

# Structured Query Language (SQL)

## SQL Commands, Tips & Structure

Visit the Cultural View of Technology [SQL Tutorial Page](#) for videos and exercises

# Contents

## Articles

SQL	1
SQL:2003	12
SQL:2008	13
Advantage Database Server	14
Apatar	16
Call Level Interface	17
Cardinality (SQL statements)	18
Check Constraint	19
Commit (data management)	21
Common table expressions	22
Condition (SQL)	22
Correlated subquery	23
CUBRID	24
Cursor (databases)	28
Data Control Language	32
Data Definition Language	33
Data Manipulation Language	35
Database Console Commands (Transact-SQL)	36
DbForge Studio for MySQL	42
Declarative Referential Integrity	44
Devgems Data Modeler	46
Embedded SQL	47
EnterpriseDB	50
Epictetus Database Client	52
Foreign key	53
FSQL	56
Hint (SQL)	59
HSQldb	60
Log shipping	62
MaxDB	63
Meta-SQL	65
MySQL Workbench	66
NVL	69
Navicat	70

Nested SQL	72
Object-oriented SQL	72
PL/pgSQL	73
PL/SQL	74
Pro*C	82
Query optimizer	82
Query plan	85
Rollback (data management)	87
SQL Access Group	88
SQL CLR	88
SQL Problems Requiring Cursors	89
SQL injection	92
SQL/CLI	97
SQL/MED	97
SQL/OLB	98
SQL/PSM	98
SQL/Schemata	99
SQL/XML	99
SQLPro SQL Client	101
Scriptella	103
SQL PL	104
SQL/JRT	105
SQuirreL SQL Client	106
Standard Interchange Language	108
Table (database)	108
Transact-SQL	109
Truviso	111
User-defined function	113
varchar	116
View (database)	117
WQL	119
Windows Internal Database	120
XLeratorDB	121
XSQL	125

## References

Article Sources and Contributors	126
Image Sources, Licenses and Contributors	129

# Article Licenses

License

130

# SQL

---

<b>Paradigm</b>	Multi-paradigm
<b>Appeared in</b>	1974
<b>Designed by</b>	Donald D. Chamberlin Raymond F. Boyce
<b>Developer</b>	IBM
<b>Stable release</b>	SQL:2008 (2008)
<b>Typing discipline</b>	Static, strong
<b>Major implementations</b>	Many
<b>Dialects</b>	SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008
<b>Influenced by</b>	Datalog
<b>Influenced</b>	Agona, CQL, LINQ, Windows PowerShell
<b>OS</b>	Cross-platform

**SQL** (officially pronounced /ˌɛskjuːˈɛl/ like "S-Q-L" but is often pronounced /ˈsiːkwəl/ like "Sequel"),<sup>[1]</sup> often referred to as **Structured Query Language**,<sup>[2]</sup> <sup>[3]</sup> is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based upon relational algebra. Its scope includes data query and update, schema creation and modification, and data access control. SQL was one of the first languages for Edgar F. Codd's relational model in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks"<sup>[4]</sup> and became the most widely used language for relational databases.<sup>[2]</sup><sup>[5]</sup>

ANSI SQL is a Turing complete programming language.

## History

SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. This version, initially called **SEQUEL**, was designed to manipulate and retrieve data stored in IBM's original relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970s.<sup>[6]</sup> The acronym SEQUEL was later changed to SQL because "SEQUEL" was a trademark of the UK-based Hawker Siddeley aircraft company.<sup>[7]</sup>

The first *Relational Database Management System* (RDBMS) was RDMS, developed at MIT in the early 1970s, soon followed by Ingres, developed in 1974 at U.C. Berkeley. Ingres implemented a query language known as QUEL, which was later supplanted in the marketplace by SQL.<sup>[7]</sup>

In the late 1970s, Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Codd, Chamberlin, and Boyce and developed their own SQL-based RDBMS with aspirations of selling it to the U.S. Navy, Central Intelligence Agency, and other U.S. government agencies. In the summer of 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers. *Oracle V2* beat IBM's release of the System/38 RDBMS to market by a few weeks.

After testing SQL at customer test sites to determine the usefulness and practicality of the system, IBM began developing commercial products based on their System R prototype including System/38, SQL/DS, and DB2, which were commercially available in 1979, 1981, and 1983, respectively.<sup>[8]</sup>

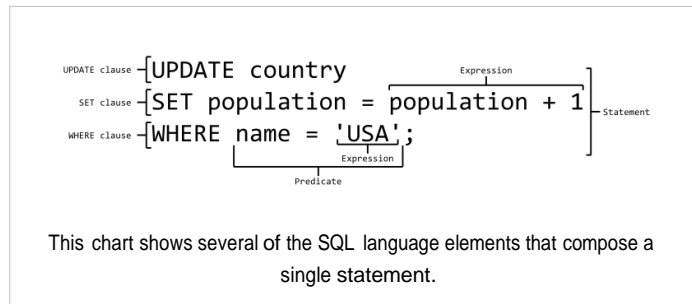
Common criticisms of SQL include a lack of cross-platform portability between vendors, inappropriate handling of missing data (see *Null (SQL)*), and unnecessarily complex and occasionally ambiguous language grammar and semantics. It also lacks the rigour of more formal languages such as relational algebra.

---

## Language elements

The SQL language is sub-divided into several language elements, including:

- *Clauses*, which are in some cases optional, constituent components of statements and queries.<sup>[9]</sup>
- *Expressions* which can produce either scalar values or tables consisting of columns and rows of data.
- *Predicates* which specify conditions that can be evaluated to SQL three-valued logic (3VL) Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- *Queries* which retrieve data based on specific criteria.
- *Statements* which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
  - SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- *Insignificant whitespace* is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.



## Queries

The most common operation in SQL is the query, which is performed with the declarative **SELECT** statement. **SELECT** retrieves data from one or more tables, or expressions. Standard **SELECT** statements have no persistent effects on the database. Some non-standard implementations of **SELECT** can have persistent effects, such as the **SELECT INTO** syntax that exists in some databases.<sup>[10]</sup>

Queries allow the user to describe desired data, leaving the database management system (DBMS) responsible for planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to be included in the final result immediately following the **SELECT** keyword. An asterisk ("\*") can also be used to specify that the query should return all columns of the queried tables. **SELECT** is the most complex statement in SQL, with optional keywords and clauses that include:

- The **FROM** clause which indicates the table(s) from which data is to be retrieved. The **FROM** clause can include optional **JOIN** subclauses to specify the rules for joining tables.
- The **WHERE** clause includes a comparison predicate, which restricts the rows returned by the query. The **WHERE** clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.
- The **GROUP BY** clause is used to project rows having common values into a smaller set of rows. **GROUP BY** is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The **WHERE** clause is applied before the **GROUP BY** clause.
- The **HAVING** clause includes a predicate used to filter rows resulting from the **GROUP BY** clause. Because it acts on the results of the **GROUP BY** clause, aggregation functions can be used in the **HAVING** clause predicate.
- The **ORDER BY** clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending). Without an **ORDER BY** clause, the order of rows returned by an SQL query is undefined.

The following is an example of a **SELECT** query that returns a list of expensive books. The query retrieves all rows from the *Book* table in which the *price* column contains a value greater than 100.00. The result is sorted in ascending order by *title*. The asterisk (\*) in the *select list* indicates that all columns of the *Book* table should be included in the

result set.

```
SELECT *  
  FROM Book  
 WHERE price > 100.00  
 ORDER BY title;
```

The example below demonstrates a query of multiple tables, grouping, and aggregation, by returning a list of books and the number of authors associated with each book.

```
SELECT Book.title,  
       count(*) AS Authors  
  FROM Book  
   JOIN Book_author ON Book.isbn = Book_author.isbn  
 GROUP BY Book.title;
```

Example output might resemble the following:

Title	Authors
SQL Examples and Guide	4
The Joy of SQL	1
An Introduction to SQL	2
Pitfalls of SQL	1

Under the precondition that *isbn* is the only common column name of the two tables and that a column named *title* only exists in the *Books* table, the above query could be rewritten in the following form:

```
SELECT title,  
       count(*) AS Authors  
  FROM Book  
   NATURAL JOIN Book_author  
 GROUP BY title;
```

However, many vendors either do not support this approach, or require certain column naming conventions in order for natural joins to work effectively.

SQL includes operators and functions for calculating values on stored values. SQL allows the use of expressions in the *select list* to project data, as in the following example which returns a list of books that cost more than 100.00 with an additional *sales\_tax* column containing a sales tax figure calculated at 6% of the *price*.

```
SELECT isbn,  
       title,  
       price,  
       price * 0.06 AS sales_tax  
  FROM Book  
 WHERE price > 100.00  
 ORDER BY title;
```

### Null and three-valued logic (3VL)

The idea of Null was introduced into SQL to handle missing information in the relational model. The introduction of Null (or Unknown) along with True and False is the foundation of three-valued logic. Null does not have a value (and is not a member of any data domain) but is rather a placeholder or "mark" for missing information. Therefore comparisons with Null can never result in either True or False but always in the third logical result, Unknown.<sup>[11]</sup>

SQL uses Null to handle missing information. It supports three-valued logic (3VL) and the rules governing SQL three-valued logic (3VL) are shown below (p and q represent logical states).<sup>[12]</sup> The word NULL is also a reserved keyword in SQL, used to identify the Null special marker.

Additionally, since SQL operators return Unknown when comparing anything with Null, SQL provides two Null-specific comparison predicates: The IS NULL and IS NOT NULL test whether data is or is not Null.<sup>[13]</sup>

Note that SQL returns only results for which the WHERE clause returns a value of True. I.e., it excludes results with values of False, but also those whose value is Unknown.

p AND q		p		
		True	False	Unknown
q	True	True	False	Unknown
	False	False	False	False
	Unknown	Unknown	False	Unknown

p OR q		p		
		True	False	Unknown
q	True	True	True	True
	False	True	False	Unknown
	Unknown	True	Unknown	Unknown

p	NOT p
True	False
False	True
Unknown	Unknown

p = q		p		
		True	False	Unknown
q	True	True	False	Unknown
	False	False	True	Unknown
	Unknown	Unknown	Unknown	Unknown

Universal quantification is not explicitly supported by SQL, and must be worked out as a negated existential quantification.<sup>[14] [15] [16]</sup>

There is also the "<row value expression> IS DISTINCT FROM <row value expression>" infix comparison operator which returns TRUE unless both operands are equal or both are NULL. Likewise, IS NOT DISTINCT FROM is defined as "NOT (<row value expression> IS DISTINCT FROM <row value expression>")

### Data manipulation

The Data Manipulation Language (DML) is the subset of SQL used to add, update and delete data:

- INSERT adds rows (formally tuples) to an existing table, e.g.,:

```
INSERT INTO My_table
    (field1, field2, field3)
VALUES
    ('test', 'N', NULL);
```

- UPDATE modifies a set of existing table rows, e.g.,:

```
UPDATE My_table
    SET field1 = 'updated value'
    WHERE field2 = 'N';
```

- DELETE removes existing rows from a table, e.g.,:

```
DELETE FROM My_table
    WHERE field2 = 'N';
```

- TRUNCATE deletes all data from a table in a very fast way. It usually implies a subsequent COMMIT operation.



- MERGE is used to combine the data of multiple tables. It combines the INSERT and UPDATE elements. It is defined in the SQL:2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called "upsert".

## Transaction controls

Transactions, if available, wrap DML operations:

- START TRANSACTION (or BEGIN WORK, or BEGIN TRANSACTION, depending on SQL dialect) mark the start of a database transaction, which either completes entirely or not at all.
- SAVE TRANSACTION (or SAVEPOINT ) save the state of the database at the current point in transaction

```
CREATE TABLE tbl_1(id int);
INSERT into tbl_1(id) value(1);
INSERT into tbl_1(id) value(2);
COMMIT;
UPDATE tbl_1 SET id=200 WHERE id=1;
SAVEPOINT id-1upd;
UPDATE tbl_1 SET id=1000 WHERE id=2;
ROLLBACK to id-1upd;
SELECT id from tbl_1;
```

- COMMIT causes all data changes in a transaction to be made permanent.
- ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, leaving the state of the data as it was prior to those changes.

Once the COMMIT statement completes, the transaction's changes cannot be rolled back.

COMMIT and ROLLBACK terminate the current transaction and release data locks. In the absence of a START TRANSACTION or similar statement, the semantics of SQL are implementation-dependent. Example: *A classic bank transfer of funds transaction.*

```
START TRANSACTION;
UPDATE Account SET amount=amount-200 WHERE account_number=1234;
UPDATE Account SET amount=amount+200 WHERE account_number=2345;
IF ERRORS=0 COMMIT;
IF ERRORS<>0 ROLLBACK;
```

## Data definition

The Data Definition Language (DDL) manages table and index structure. The most basic items of DDL are the CREATE, ALTER, RENAME, DROP and TRUNCATE statements:

- CREATE creates an object (a table, for example) in the database.
- DROP deletes an object in the database, usually irretrievably.
- ALTER modifies the structure of an existing object in various ways—for example, adding a column to an existing table.

Example:

```
CREATE TABLE My_table
(
    my_field1 INT,
    my_field2 VARCHAR(50),
    my_field3 DATE NOT NULL,
```

```
PRIMARY KEY (my_field1, my_field2)
);
```

## Data types

Each column in an SQL table declares the type(s) that column may contain. ANSI SQL includes the following datatypes.<sup>[17]</sup>

### Character strings

- CHARACTER(n) or CHAR(n) — fixed-width n-character string, padded with spaces as needed
- CHARACTER VARYING(n) or VARCHAR(n) — variable-width string with a maximum size of n characters
- NATIONAL CHARACTER(n) or NCHAR(n) — fixed width string supporting an international character set
- NATIONAL CHARACTER VARYING(n) or NVARCHAR(n) — variable-width NCHAR string

### Bit strings

- BIT(n) — an array of n bits
- BIT VARYING(n) — an array of up to n bits

### Numbers

- INTEGER and SMALLINT
- FLOAT, REAL and DOUBLE PRECISION
- NUMERIC(precision, scale) or DECIMAL(precision, scale)

SQL provides a function to round numerics or dates, called TRUNC (in Informix, DB2, PostgreSQL, Oracle and MySQL) or ROUND (in Informix, Sybase, Oracle, PostgreSQL and Microsoft SQL Server)<sup>[18]</sup>

### Date and time

- DATE — for date values (e.g., 2010-05-30)
- TIME — for time values (e.g., 14:55:37). The granularity of the time value is usually a *tick* (100 nanoseconds).
- TIME WITH TIME ZONE or TIMESTAMP — the same as TIME, but including details about the time zone in question.
- TIMESTAMP — This is a DATE and a TIME put together in one variable (e.g., 2010-05-30 14:55:37).
- TIMESTAMP WITH TIME ZONE or TIMESTAMPTZ — the same as TIMESTAMP, but including details about the time zone in question.

SQL provides several functions for generating a date / time variable out of a date / time string (TO\_DATE, TO\_TIME, TO\_TIMESTAMP), as well as for extracting the respective members (seconds, for instance) of such variables. The current system date / time of the database server can be called by using functions like NOW.

## Data control

The Data Control Language (DCL) authorizes users and groups of users to access and manipulate data. Its two main statements are:

- GRANT authorizes one or more users to perform an operation or a set of operations on an object.
- REVOKE eliminates a grant, which may be the default grant.

Example:

```
GRANT SELECT, UPDATE
ON My_table
TO some_user, another_user;
```

```

REVOKE SELECT, UPDATE
ON My_table
FROM some_user, another_user;

```

## Procedural extensions

SQL is designed for a specific purpose: to query data contained in a relational database. SQL is a set-based, declarative query language, not an imperative language such as C or BASIC. However, there are extensions to Standard SQL which add procedural programming language functionality, such as control-of-flow constructs. These are:

Source	Common Name	Full Name
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Interbase/ Firebird	PSQL	Procedural SQL
IBM	SQL PL	SQL Procedural Language (implements SQL/PSM)
Microsoft/ Sybase	T-SQL	Transact-SQL
Mimer SQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (based on Oracle PL/SQL)
PostgreSQL	PL/PSM	Procedural Language/Persistent Stored Modules (implements SQL/PSM)

In addition to the standard SQL/PSM extensions and proprietary SQL extensions, procedural and object-oriented programmability is available on many SQL platforms via DBMS integration with other languages. The SQL standard defines SQL/JRT extensions (SQL Routines and Types for the Java Programming Language) to support Java code in SQL databases. SQL Server 2005 uses the SQLCLR (SQL Server Common Language Runtime) to host managed .NET assemblies in the database, while prior versions of SQL Server were restricted to using unmanaged extended stored procedures which were primarily written in C. Other database platforms, like MySQL and Postgres, allow functions to be written in a wide variety of languages including Perl, Python, Tcl, and C.

## Criticisms of SQL

SQL is a declarative computer language for use with relational databases. Interestingly, many of the original SQL features were inspired by, but violated, the semantics of the relational model and its tuple calculus realization. Recent extensions to SQL achieved relational completeness, but have worsened the violations, as documented in *The Third Manifesto*.

Practical criticisms of SQL include:

- Implementations are inconsistent and, usually, incompatible between vendors. In particular date and time syntax, string concatenation, nulls, and comparison case sensitivity vary from vendor to vendor.
- The language makes it too easy to do a Cartesian join (joining all possible combinations), which results in "run-away" result sets when WHERE clauses are mistyped. Cartesian joins are so rarely used in practice that requiring an explicit CARTESIAN keyword may be warranted. (SQL 1992 introduced the CROSS JOIN keyword that allows the user to make clear that a Cartesian join is intended, but the shorthand "comma-join" with no predicate is still acceptable syntax, which still invites the same mistake.)

- It is also possible to misconstrue a WHERE on an update or delete, thereby affecting more rows in a table than desired. (A work-around is to use transactions or habitually type in the WHERE clause first, then fill in the rest later.)
- The grammar of SQL is perhaps unnecessarily complex, borrowing a COBOL-like keyword approach, when a function-influenced syntax could result in more re-use of fewer grammar and syntax rules.

## Cross-vendor portability

Popular implementations of SQL commonly omit support for basic features of Standard SQL, such as the DATE or TIME data types. As a result, SQL code can rarely be ported between database systems without modifications.

There are several reasons for this lack of portability between database systems:

- The complexity and size of the SQL standard means that most implementors do not support the entire standard.
- The standard does not specify database behavior in several important areas (e.g., indexes, file storage...), leaving implementations to decide how to behave.
- The SQL standard precisely specifies the syntax that a conforming database system must implement. However, the standard's specification of the semantics of language constructs is less well-defined, leading to ambiguity.
- Many database vendors have large existing customer bases; where the SQL standard conflicts with the prior behavior of the vendor's database, the vendor may be unwilling to break backward compatibility.
- Software vendors often desire to create incompatibilities with other products, as it provides a strong incentive for their existing users to remain loyal (see vendor lock-in).

## Standardization

SQL was adopted as a standard by the American National Standards Institute (ANSI) in 1986 as SQL-86<sup>[19]</sup> and International Organization for Standardization (ISO) in 1987. The original SQL standard declared that the official pronunciation for SQL is "es queue el".<sup>[2]</sup> Many English-speaking database professionals still use the nonstandard<sup>[20]</sup> pronunciation /'si:kwəl/ (like the word "sequel").

Until 1996, the National Institute of Standards and Technology (NIST) data management standards program certified SQL DBMS compliance with the SQL standard. Vendors now self-certify the compliance of their products.<sup>[21]</sup>

The SQL standard has gone through a number of revisions, as shown below:

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), <i>Entry Level</i> SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features.
2003	SQL:2003		Introduced XML-related features, <i>window functions</i> , standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it enables applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.

2008	SQL:2008		Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement. <sup>[22]</sup>
------	----------	--	---

Interested parties may purchase SQL standards documents from ISO or ANSI. A draft of SQL:2008 is freely available as a zip archive.<sup>[23]</sup>

## Standard structure

The SQL standard is divided into several parts, including:

SQL Framework, provides logical concept

SQL/Foundation, defined in ISO/IEC 9075, Part 2. This part of the standard contains the most central elements of the language. It consists of both **mandatory and optional** features.

The SQL/Bindings, specifies how SQL is to be bound to variable host languages, excluding Java.

The SQL/CLI, or **Call-Level Interface**, part is defined in ISO/IEC 9075, Part 3. SQL/CLI defines common interfacing components (structures and procedures) that can be used to execute SQL statements from applications written in other programming languages. SQL/CLI is defined in such a way that SQL statements and SQL/CLI procedure calls are treated as separate from the calling application's source code. Open Database Connectivity is a well-known superset of SQL/CLI. This part of the standard consists solely of **mandatory** features.

The SQL/PSM, or **Persistent Stored Modules**, part is defined by ISO/IEC 9075, Part 4. SQL/PSM standardizes procedural extensions for SQL, including flow of control, condition handling, statement condition signals and resignals, cursors and local variables, and assignment of expressions to variables and parameters. In addition, SQL/PSM formalizes declaration and maintenance of persistent database language routines (e.g., "stored procedures"). This part of the standard consists solely of **optional** features.

The SQL/MED, or **Management of External Data**, part is defined by ISO/IEC 9075, Part 9. SQL/MED provides extensions to SQL that define foreign-data wrappers and datalink types to allow SQL to manage external data. External data is data that is accessible to, but not managed by, an SQL-based DBMS. This part of the standard consists solely of **optional** features.

The SQL/OLB, or **Object Language Bindings**, part is defined by ISO/IEC 9075, Part 10. SQL/OLB defines the syntax and semantics of SQLJ, which is SQL embedded in Java. The standard also describes mechanisms to ensure binary portability of SQLJ applications, and specifies various Java packages and their contained classes. This part of the standard consists solely of **optional** features.

The SQL/MM (Multimedia), This extends SQL to deal intelligently with large, complex and sometimes streaming items of data, such as video, audio and spatial data.

The SQL/Schemata, or **Information and Definition Schemas**, part is defined by ISO/IEC 9075, Part 11. SQL/Schemata defines the Information Schema and Definition Schema, providing a common set of tools to make SQL databases and objects self-describing. These tools include the SQL object identifier, structure and integrity constraints, security and authorization specifications, features and packages of ISO/IEC 9075, support of features provided by SQL-based DBMS implementations, SQL-based DBMS implementation information and sizing items, and the values supported by the DBMS implementations.<sup>[24]</sup> This part of the standard contains both **mandatory and optional** features.

The SQL/JRT, or **SQL Routines and Types for the Java Programming Language**, part is defined by ISO/IEC 9075, Part 13. SQL/JRT specifies the ability to invoke static Java methods as routines from within SQL applications. It also calls for the ability to use Java classes as SQL structured user-defined types. This part of the standard consists solely of **optional** features.

The SQL/XML, or **XML-Related Specifications**, part is defined by ISO/IEC 9075, Part 14. SQL/XML specifies SQL-based extensions for using XML in conjunction with SQL. The XML data type is introduced, as well as several

routines, functions, and XML-to-SQL data type mappings to support manipulation and storage of XML in an SQL database. This part of the standard consists solely of **optional** features.

## Alternatives to SQL

A distinction should be made between alternatives to relational query languages and alternatives to SQL. Below are proposed relational alternatives to SQL. See navigational database for alternatives to relational:

- .QL - object-oriented Datalog
- 4D Query Language (4D QL)
- Datalog
- Hibernate Query Language (HQL) - A Java-based tool that uses modified SQL
- HTSQL - URL based query method
- IBM Business System 12 (IBM BS12) - one of the first fully relational database management systems, introduced in 1982
- ISBL
- Java Persistence Query Language (JPQL) - The query language used by the Java Persistence API in Java EE5
- LINQ
- Object Query Language
- QBE (Query By Example) created by Moshè Zloof, IBM 1977
- Quel introduced in 1974 by the U.C. Berkeley Ingres project.
- Tutorial D
- XQuery

## See also

- Comparison of object-relational database management systems
- Comparison of relational database management systems
- D (data language specification)
- D4 (programming language) (an implementation of D)
- Hierarchical model
- List of relational database management systems
- MUMPS
- Search suggest drop-down list
- NoSQL

## References

- [1] Beaulieu, Alan (April 2009). Mary E. Treseler. ed. *Learning SQL* (2nd ed.). Sebastapol, CA, USA: O'Reilly. ISBN 978-0-596-52083-0.
- [2] Chapple, Mike. "SQL Fundamentals" (<http://databases.about.com/od/sql/a/sqlfundamentals.htm>). *About.com: Databases*. About.com. . Retrieved 2009-01-28.
- [3] Connolly, Thomas. *Database Systems* (2nd ed.). Addison-Wesley. p. 384. ISBN 0-201-34287-1.
- [4] Codd, E.F. (June 1970). "A Relational Model of Data for Large Shared Data Banks" (<http://www.acm.org/classics/nov95/toc.html>). *Communications of the ACM* (Association for Computing Machinery) **13** (6): 377–387. doi:10.1145/362384.362685. . Retrieved 2007-06-09.
- [5] "Structured Query Language (SQL)" (<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0004100.htm>). International Business Machines. October 27, 2006. . Retrieved 2007-06-10.
- [6] Chamberlin, Donald D.; Boyce, Raymond F. (1974). "SEQUEL: A Structured English Query Language" (<http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf>) (PDF). *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control* (Association for Computing Machinery): 249–264. . Retrieved 2007-06-09.
- [7] Oppel, Andy (March 1, 2004). *Databases Demystified* (<http://www.mhprofessional.com/product.php?cat=112&isbn=0072253649>). San Francisco, CA: McGraw-Hill Osborne Media. pp. 90–91. ISBN 0-07-225364-9. .
- [8] "History of IBM, 1978" ([http://www-03.ibm.com/ibm/history/history/year\\_1978.html](http://www-03.ibm.com/ibm/history/history/year_1978.html)). *IBM Archives*. IBM. . Retrieved 2007-06-09.

- [9] ANSI/ISO/IEC International Standard (IS). Database Language SQL—Part 2: Foundation (SQL/Foundation). 1999.
  - [10] "INTO Clause (Transact-SQL)" ([http://msdn2.microsoft.com/en-us/library/ms188029\(SQL.90\).aspx](http://msdn2.microsoft.com/en-us/library/ms188029(SQL.90).aspx)). *SQL Server 2005 Books Online*. Microsoft. 2007. . Retrieved 2007-06-17;.
  - [11] ISO/IEC (2003). *ISO/IEC 9075-1:2003, "SQL/Framework"* (<http://www.iso.org>). ISO/IEC. Section 4.4.2: *The null value*. .
  - [12] Coles, Michael (2005-06-27). "Four Rules for Nulls" (<http://www.sqlservercentral.com/columnists/mcoles/fourrulesfornulls.asp>). *SQL Server Central* (Red Gate Software). .
  - [13] ISO/IEC. *ISO/IEC 9075-2:2003, "SQL/Foundation"*. ISO/IEC.
  - [14] M. Negri, G. Pelagatti, L. Sattella (1989) *Semantics and problems of universal quantification in SQL* (<http://portal.acm.org/citation.cfm?id=63224.68822&coll=GUIDE&dl=GUIDE>).
  - [15] Fratarcangeli, Claudio (1991). *Technique for universal quantification in SQL*. Retrieved from ACM.org. (<http://portal.acm.org/citation.cfm?id=126482.126484&coll=GUIDE&dl=GUIDE&CFID=5934371&CFTOKEN=55309005>)
  - [16] Kawash, Jalal (2004). *Complex quantification in Structured Query Language (SQL): a tutorial using relational calculus* - Journal of Computers in Mathematics and Science Teaching ISSN 0731-9258 Volume 23, Issue 2, 2004 AACE Norfolk, Virginia. Retrieved from Thefreelibrary.com ([http://www.thefreelibrary.com/Complex+quantification+in+Structured+Query+Language+\(SQL\):+a+tutorial...-a0119901477](http://www.thefreelibrary.com/Complex+quantification+in+Structured+Query+Language+(SQL):+a+tutorial...-a0119901477)).
  - [17] Information Technology - Database Language SQL (Proposed revised text of DIS 9075) (<http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt>).
  - [18] Arie Jones, Ryan K. Stephens, Ronald R. Plew, Alex Kriegel, Robert F. Garrett (2005), *SQL Functions Programmer's Reference*. Wiley, 127 pages.
  - [19] American National Standards Institute. X3H2 Records, 1978-1995. Finding Aid. (<http://special.lib.umn.edu/findaid/xml/cbi00168.xml>)
  - [20] Melton, Jim; Alan R Simon (1993). *Understanding the New SQL: A Complete Guide*. Morgan Kaufmann. p. 536. ISBN 1558602453. "chapter 1.2 What is SQL? SQL (correctly pronounced "ess cue ell," instead of the somewhat common "sequel"), is a..."
  - [21] Doll, Shelley (June 19, 2002). "Is SQL a Standard Anymore?" ([http://articles.techrepublic.com.com/5100-10878\\_11-1046268.html](http://articles.techrepublic.com.com/5100-10878_11-1046268.html)). *TechRepublic's Builder.com*. TechRepublic. . Retrieved 2010-01-07.
  - [22] Sybase.com (<http://iablog.sybase.com/paulley/2008/07/sql2008-now-an-approved-iso-international-standard/>).
  - [23] Zip archive of the SQL:2008 draft (<http://www.wiscorp.com/sql200n.zip>) from Whitemarsh Information Systems Corporation.
  - [24] *ISO/IEC 9075-11:2008: Information and Definition Schemas (SQL/Schemata)*. 2008. p. 1.
- "A Relational Model of Data for Large Shared Data Banks" (<http://www.acm.org/classics/nov95/toc.html>) E. F. Codd, Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377–387.
  - Discussion on alleged SQL flaws (C2 wiki)

## External links

- *1995 SQL Reunion: People, Projects, and Politics*, by Paul McJones (ed.) ([http://www.mcjones.org/System\\_R/SQL\\_Reunion\\_95/sq195.html](http://www.mcjones.org/System_R/SQL_Reunion_95/sq195.html)): transcript of a reunion meeting devoted to the personal history of relational databases and SQL.
- American National Standards Institute. X3H2 Records, 1978-1995 (<http://special.lib.umn.edu/findaid/xml/cbi00168.xml>) Charles Babbage Institute Collection documents the H2 committee's development of the NDL and SQL standards.
- Oral history interview with Donald D. Chamberlin (<http://www.cbi.umn.edu/oh/display.phtml?id=317>) Charles Babbage Institute In this oral history Chamberlin recounts his early life, his education at Harvey Mudd College and Stanford University, and his work on relational database technology. Chamberlin was a member of the System R research team and, with Raymond F. Boyce, developed the SQL database language. Chamberlin also briefly discusses his more recent research on XML query languages.
- Comparison of Different SQL Implementations (<http://troels.arvin.dk/db/rdbms/>) This comparison of various SQL implementations is intended to serve as a guide to those interested in porting SQL code between various RDBMS products, and includes comparisons between SQL:2008, PostgreSQL, DB2, MS SQL Server, MySQL, Oracle, and Informix.

# SQL:2003

---

**SQL:2003** is the fifth revision of the SQL database query language. The latest revision of the standard is SQL:2008.

## Summary

The SQL:2003 standard makes minor modifications to all parts of SQL:1999 (also known as SQL3), and officially introduces a few new features such as:<sup>[1]</sup>

- XML-related features (SQL/XML)
- Window functions
- the sequence generator, which allows standardized sequences
- two new column types: auto-generated values and identity-columns
- the new MERGE statement
- extensions to the CREATE TABLE statement, to allow "CREATE TABLE AS" and "CREATE TABLE LIKE"
- removal of the poorly-implemented "BIT" and "BIT VARYING" data types

## Documentation availability

The SQL standard is not freely available. SQL:2003 may be purchased from ISO <sup>[2]</sup> or ANSI <sup>[3]</sup>. A late draft is available as a zip archive <sup>[4]</sup> from Whitemarsh Information Systems Corporation <sup>[5]</sup>. The zip archive contains a number of PDF files that define the parts of the SQL:2003 specification.

- ISO/IEC 9075(1-4,9-11,13,14):2003 CD-ROM (352 CHF, or approximately 225 EUR, to order the CD)
  - ISO/IEC 9075-1:2003 <sup>[6]</sup> – Framework (SQL/Framework)
  - ISO/IEC 9075-2:2003 <sup>[7]</sup> – Foundation (SQL/Foundation)
  - ISO/IEC 9075-3:2003 <sup>[8]</sup> – Call-Level Interface (SQL/CLI)
  - ISO/IEC 9075-4:2003 <sup>[9]</sup> – Persistent Stored Modules (SQL/PSM)
  - ISO/IEC 9075-9:2003 <sup>[10]</sup> – Management of External Data (SQL/MED)
  - ISO/IEC 9075-10:2003 <sup>[11]</sup> – Object Language Bindings (SQL/OLB)
  - ISO/IEC 9075-11:2003 <sup>[12]</sup> – Information and Definition Schemas (SQL/Schemata)
  - ISO/IEC 9075-13:2003 <sup>[13]</sup> – SQL Routines and Types Using the Java Programming Language (SQL/JRT)
  - ISO/IEC 9075-14:2003 <sup>[14]</sup> – XML-Related Specifications (SQL/XML)

## External links

- BNF Grammar for ISO/IEC 9075-1:2003 <sup>[15]</sup> – **SQL/Framework**
- BNF Grammar for ISO/IEC 9075-2:2003 <sup>[16]</sup> – SQL/Foundation

## References

- [1] Eisenberg, Andrew; et al. (March 2004). "SQL:2003 Has Been Published" (<http://www.acm.org/sigmod/record/issues/0403/index.html#standards>) (pdf). *SIGMOD Record* **33** (1): 119. doi:10.1145/974121.974142. . Retrieved 2007-08-14.
  - [2] <http://www.iso.org/>
  - [3] <http://webstore.ansi.org/>
  - [4] [http://www.wiscorp.com/sql\\_2003\\_standard.zip](http://www.wiscorp.com/sql_2003_standard.zip)
  - [5] <http://www.wiscorp.com/>
  - [6] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34132>
  - [7] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34133>
  - [8] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34134>
  - [9] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34135>
  - [10] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34136>
-



- [11] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34137>
- [12] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=34917>
- [13] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37102>
- [14] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=35341>
- [15] <http://savage.net.au/SQL/sql-2003-1.bnf.html>
- [16] <http://savage.net.au/SQL/sql-2003-2.bnf.html>

## SQL:2008

---

**SQL:2008** is the sixth revision of the SQL database query language.

### Summary

The SQL:2008 standard was published in 2008.

### Documentation availability

The SQL standard is not freely available. The whole standard may be purchased from the ISO as *ISO/IEC 9075(1-4,9-11,13,14):2008*. The standard consists of the following parts:

- ISO/IEC 9075-1:2008<sup>[1]</sup> Framework (SQL/Framework)
- ISO/IEC 9075-2:2008<sup>[2]</sup> Foundation (SQL/Foundation)
- ISO/IEC 9075-3:2008<sup>[3]</sup> Call-Level Interface (SQL/CLI)
- ISO/IEC 9075-4:2008<sup>[4]</sup> Persistent Stored Modules (SQL/PSM)
- ISO/IEC 9075-9:2008<sup>[5]</sup> Management of External Data (SQL/MED)
- ISO/IEC 9075-10:2008<sup>[6]</sup> Object Language Bindings (SQL/OLB)
- ISO/IEC 9075-11:2008<sup>[7]</sup> Information and Definition Schemas (SQL/Schemata)
- ISO/IEC 9075-13:2008<sup>[8]</sup> SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)
- ISO/IEC 9075-14:2008<sup>[9]</sup> XML-Related Specifications (SQL/XML)

### References

- [1] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=45498](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45498)
  - [2] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38640](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38640)
  - [3] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38641](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38641)
  - [4] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38642](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38642)
  - [5] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38643](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38643)
  - [6] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38644](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38644)
  - [7] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38645](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38645)
  - [8] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38646](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38646)
  - [9] [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=45499](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45499)
-

# Advantage Database Server

---

<b>Developer(s)</b>	Sybase
<b>Initial release</b>	
<b>Written in</b>	C, C++
<b>Operating system</b>	Cross-platform
<b>Available in</b>	English
<b>Type</b>	RDBMS
<b>Website</b>	<a href="http://advantagedatabase.com">advantagedatabase.com</a> <sup>[1]</sup>

**Advantage Database Server** is a relational database management system (RDBMS) for small to medium sized businesses by Sybase iAnywhere. Database author Cary Jensen describes Advantage as follows: "Advantage has been around since 1993, when it was introduced to provide a stable solution for Clipper developers who were tired of slow performance and corrupt indexes inherent to file server-based databases. Over the years, ADS has grown in both popularity and features. Advantage is now a mature product with an impressive collection of features that rival many of the more expensive and complicated database servers"<sup>[2]</sup>.

"In short, the Advantage Database Server is a high-performance, low-maintenance, remote database server that permits you to easily build and deploy client/server applications and web-based applications"<sup>[3]</sup>.

## Uses

Advantage Database Server is most popular among application developers as a client/server backend solution for shared, networked, standalone, mobile and Internet database applications. ADS is unique among other database offerings because it provides both ISAM table-based and SQL based data access.

## Features

- Replication
  - Online Backup
  - ISAM access
  - SQL-92 compliant
  - SQL Query Optimizer
  - User Defined Functions
  - Triggers
  - Stored Procedures
  - Views
  - Server Side Aliases
  - Encrypted indexes and communications
  - DBF tables and memos greater than 4 gigabytes
  - Transactions
  - Events / Notifications
  - 64-bit support
  - Full Text Search
  - Multiple Processor support
-

- Small footprint

## Programming Languages

- Popular development environments including CodeGear Delphi, CodeGear C++Builder, Microsoft Visual Basic, Microsoft Visual C++, CA-Visual Objects, CA-Clipper and Microsoft Visual FoxPro.
- Supports standard interfaces such as ODBC, OLE DB, JDBC, PHP, and ADO.NET.

## History

- August 1993 – Initially released as Advantage xBase server by Extended Systems.
- December 1995 – version 4.0 release, now called Advantage Database Server. This release included the first ODBC driver.
- February 1997 – Advantage Internet Server released.
- April 2000 – Advantage ODBC driver with StreamlineSQL released.
- November 2005 – Advantage Database Server: The Official Guide published<sup>[4]</sup>.
- November 2005 – Extended Systems acquired by Sybase, rolled in to iAnywhere Solutions subsidiary.
- November 2008 – The latest version was released, Advantage Database Server 9.1.

## See also

- Comparison of relational database management systems
- Sybase iAnywhere

## External links

- Advantage Database Server website <sup>[5]</sup>
- iAnywhere website on Sybase.com <sup>[6]</sup>
- JD was here – Advantage blog <sup>[7]</sup>
- Chris Franz's Advantage blog <sup>[8]</sup>
- Cary Jensen's website <sup>[9]</sup>
- Advantage Developer Zone <sup>[10]</sup>

## References

- [1] <http://www.advantagedatabase.com/>
- [2] Cary Jensen, Ph.D.Loy Anderson, Ph.D, Advantage Database Server: A Developer's Guide, February 5, 2007, 616 pages, AuthorHouse, 978-1425-7726-9
- [3] Cary Jensen, Ph.D.Loy Anderson, Ph.D, Advantage Database Server: A Developer's Guide, February 5, 2007, page 7, AuthorHouse, 978-1425-7726-9
- [4] Cary Jensen, Ph.D.Loy Anderson, Ph.D, Advantage Database Server: The Official Guide, October 29, 2003, 496 pages, McGraw-Hill Osborne Media, 0-07-223084-1
- [5] <http://www.sybase.com/products/databasemanagement/advantagedatabaseserver/>
- [6] <http://www.sybase.com/iAnywhere>
- [7] <http://jdmullin.blogspot.com/>
- [8] <http://blog.advantageevangelist.com/>
- [9] <http://www.jensendatasystems.com/>
- [10] <http://devzone.advantagedatabase.com/>

# Apatar

---

<b>Stable release</b>	v1.0.8.6 / September 21, 2007
<b>Operating system</b>	Cross-platform
<b>Type</b>	ETL, Data migration and SQL.
<b>License</b>	General Public License version 2.0
<b>Website</b>	<a href="http://apatarforge.org">apatarforge.org</a> <sup>[1]</sup>

Apatar is an open source ETL (Extract-Transform-Load) and mashup data integration software application. Other open source data integration projects are Clover.ETL, Pentaho Project, Talend Open Studio or Enhydra Octopus.

## Typical use

- Database migration.
- Application integration
- Database creation/update scripts.
- Cross-database operations, import/export.
- Automated database schema upgrade.

Apatar open source project was founded in 2005<sup>[2]</sup>. Apatar, Inc., a commercial company providing support for Apatar open source software, was founded in 2007<sup>[3]</sup>. Users of Apatar software include R.R. Donnelley and Autodesk<sup>[4]</sup>,<sup>[5]</sup>. Apatar data integration software is adopted for Amazon Web Services platform<sup>[6]</sup>.

## External links

- Apatar Website<sup>[7]</sup>
- Apatar project page on SourceForge.Net<sup>[8]</sup>
- Apatar Community Website<sup>[1]</sup>

## References

- [1] <http://www.apatarforge.org>
- [2] "Apatar Crashes Data Mashup Party" (<http://www.sdtimes.com/article/story-20070715-13.html>). SD Times (<http://www.sdtimes.com>). Retrieved 2007-09-24.
- [3] "Open Source Companies to Watch" (<http://www.networkworld.com/news/2007/082307-open-source-to-watch.html>). NetworkWorld (<http://www.networkworld.com>). Retrieved 2007-09-24.
- [4] "Two Ways To Deal With SOA's Data Integration Challenge" (<http://www.informationweek.com/software/showArticle.jhtml?articleID=200900682&pgno=2&queryText=>). InformationWeek (<http://www.informationweek.com>). Retrieved 2007-09-24.
- [5] "Young software exec moves to open source model" (<http://masshightech.bizjournals.com/masshightech/stories/2007/07/16/story12.html?page=2>). American City Business Journals, Inc. (<http://bizjournals.com>). Retrieved 2007-09-24.
- [6] "Open Source Companies to watch" (<http://www.informationweek.com/news/showArticle.jhtml?articleID=201300517>). InformationWeek (<http://www.informationweek.com>). Retrieved 2007-09-24.
- [7] <http://www.apatar.com>
- [8] <http://sourceforge.net/projects/apatar>

# Call Level Interface

---

The **Call Level Interface (CLI)** is a software standard defined in ISO/IEC 9075-3:2003. The Call Level Interface defines how a program should send SQL queries to the database management system (DBMS) and how the returned recordsets should be handled by the application in a consistent way. Developed in the early 1990s, the API was defined only for the C and COBOL programming languages.

The interface is part of what The Open Group refers to as the Common Application Environment, which is intended to be a wide standard for programming open applications, i.e. applications from different programming teams and different vendors that can interoperate in an efficient manner. SQL/CLI provides an international standard implementation-independent CLI to access SQL databases. Client-server tools can easily access databases through Dynamic-link libraries (DLL). It supports and encourages a rich set of client-server tools.

The most widespread use of the CLI standard is the basis of the ODBC (Open Database Connectivity) specification, which is widely used to allow applications to transparently access database systems from different vendors. The current version of the API, ODBC 3.52, incorporates features from both the ISO and X/Open standards (see below).

## History

The work with the Call Level Interface began in a subcommittee of the US-based SQL Access Group. In 1992, it was initially published and marketed as Microsoft's ODBC API. The CLI specification was submitted as to the ISO and ANSI standards committees in 1993; the standard has the book number ISBN 1-85912-081-4 and the internal document number is C451.

ISO SQL/CLI is an addendum to 1992 SQL standard (SQL-92). It was completed as ISO standard ISO/IEC 9075-3:1995 Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI). The current SQL/CLI effort is adding support for SQL3.

In the fourth quarter of 1994, control over the standard was transferred to the X/Open Company, which significantly expanded and updated it. The X/Open CLI interface is a superset of the ISO SQL CLI.

*This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.*

## External links

- Online definition of CLI <sup>[1]</sup> at The Open Group webpage

## References

- [1] <http://www.opengroup.org/products/publications/catalog/c451.htm>

# Cardinality (SQL statements)

---

In SQL (Structured Query Language), the term **cardinality** refers to the uniqueness of data values contained in a particular column (attribute) of a database table. The lower the cardinality, the more duplicated elements in a column. Thus, a column with the lowest possible cardinality would have the same value for every row. SQL databases use cardinality to help determine the optimal query plan for a given query.

## Values of Cardinality

When dealing with columnar value sets, there are 3 types of cardinality: high-cardinality, normal-cardinality, and low-cardinality.

**High-cardinality** refers to columns with values that are very uncommon or unique. High-cardinality column values are typically identification numbers, email addresses, or user names. An example of a data table column with high-cardinality would be a `USERS` table with a column named `USER_ID`. This column would contain unique values of 1- $n$ . Each time a new user is created in the `USERS` table, a new number would be created in the `USER_ID` column to identify them uniquely. Since the values held in the `USER_ID` column are unique, this column's cardinality type would be referred to as high-cardinality.

**Normal-cardinality** refers to columns with values that are somewhat uncommon. Normal-cardinality column values are typically names, street addresses, or vehicle types. An example of a data table column with normal-cardinality would be a `CUSTOMER` table with a column named `LAST_NAME`, containing the last names of customers. While some people have common last names, such as Smith, others have uncommon last names. Therefore, an examination of all of the values held in the `LAST_NAME` column would show "clumps" of names in some places (e.g.: a lot of Smith's ) surrounded on both sides by a long series of unique values. Since there is a variety of possible values held in this column, its cardinality type would be referred to as normal-cardinality.

**Low-cardinality** refers to columns with few unique values. Low-cardinality column values are typically status flags, boolean values, or major classifications such as gender. An example of a data table column with low-cardinality would be a `CUSTOMER` table with a column named `NEW_CUSTOMER`. This column would contain only 2 distinct values: Y or N, denoting whether the customer was new or not. Since there are only 2 possible values held in this column, its cardinality type would be referred to as low-cardinality.

## See also

- [Cardinality \(mathematics\)](#)

# Check Constraint

A **check constraint** (also known as **table check constraint**) is a condition that defines valid data when adding or updating an entry in a table of a relational database. A check constraint is applied to each row in the table. The constraint must be a predicate. It can refer to a single or multiple columns of the table. The result of the predicate can be either TRUE, FALSE, or UNKNOWN, depending on the presence of NULLs. If the predicate evaluates to UNKNOWN, then the constraint is not violated and the row can be inserted or updated in the table. This is contrary to predicates in WHERE clauses in SELECT or UPDATE statements.

For example, in a table containing products, one could add a check constraint such that the price of a product and quantity of a product is a non-negative value:

```
PRICE >= 0
```

```
QUANTITY >= 0
```

If these constraints were not in place, it would be possible to have a negative price (-\$30) or quantity (-3 items).

Check constraints are used to ensure the validity of data in a database and to provide data integrity. If they are used at the database level, applications that use the database will not be able to add invalid data or modify valid data so the data becomes invalid, even if the application itself accepts invalid data.

## Definition

Each check constraint has to be defined in the CREATE TABLE or ALTER TABLE statement using the syntax:

```
CREATE TABLE table_name (  
    ...,  
    CONSTRAINT constraint_name CHECK ( predicate ),  
    ...  
)
```

```
ALTER TABLE table_name  
    ADD CONSTRAINT constraint_name CHECK ( predicate )
```

If the check constraint refers to a single column only, it is possible to specify the constraint as part of the column definition.

```
CREATE TABLE table_name (  
    ...  
    column_name type CHECK ( predicate ),  
    ...  
)
```

## NOT NULL Constraint

A NOT NULL constraint is functionally equivalent to the following check constraint with an IS NOT NULL predicate:

```
CHECK (column IS NOT NULL)
```

Some relational database management systems are able to optimize performance when the NOT NULL constraint syntax is used as opposed to the CHECK constraint syntax given above.<sup>[1]</sup>

## Common Restrictions

Most database management systems restrict check constraints to a single row, with access to constants and deterministic functions, but not to data in other tables, or to data invisible to the current transaction because of transaction isolation.

Such constraints are not truly *table check constraints* but rather *row check constraints*. Because these constraints are generally only verified when a row is directly updated (for performance reasons,) and often implemented as implied INSERT or UPDATE triggers, integrity constraints could be violated by indirect action were it not for these limitations. Future, otherwise-valid modifications to these records would then be prevented by the CHECK constraint. Some examples of dangerous constraints include:

- CHECK ((select count(\*) from invoices where invoices.customerId = customerId) < 1000)
- CHECK (dateInserted = CURRENT\_DATE)
- CHECK (countItems = RAND())

User-defined triggers can be used to work around these restrictions. Although similar in implementation, it is semantically clear that triggers will only be fired when the table is directly modified, and that it is the designer's responsibility to handle indirect, important changes in other tables; constraints on the other hand are intended to be "true at all times" regardless of the user's actions or the designer's lack of foresight.

## References

- [1] PostgreSQL 8.3devel Documentation, Chapter 5. *Data Definition*, Section 5.3.2. *Not-Null Constraints*, Website: <http://developer.postgresql.org/pgdocs/postgres/ddl-constraints.html>, Accessed on May 5, 2007



# Commit (data management)

---

In the context of computer science and data management, **commit** refers to the idea of making a set of tentative changes permanent. A popular usage is at the end of a transaction. A *commit* is the act of committing.

## Data management

A COMMIT statement in SQL ends a transaction within a relational database management system (RDBMS) and makes all changes visible to other users. The general format is to issue a BEGIN WORK statement, one or more SQL statements, and then the COMMIT statement. Alternatively, a ROLLBACK statement can be issued, which undoes all the work performed since BEGIN WORK was issued. A COMMIT statement will also release any existing savepoints that may be in use.

In terms of transactions, the opposite of commit is to discard the tentative changes of a transaction, a rollback.

## Revision control

Commits are also done for revision control systems for source code such as Subversion or Concurrent Versions System. A commit in the context of these version control systems refers to submitting the latest changes of the source code to the repository, and making these changes part of the head revision of the repository. Thus, when other users do an UPDATE or a checkout from the repository, they will receive the latest committed version, unless they specify they wish to retrieve a previous version of the source code in the repository. Version control systems also have similar functionality to SQL databases in that they allow rolling back to previous versions easily. In this context, a commit with version control systems is not as dangerous as it allows easy rollback, even after the commit has been done.

## See also

- Atomic commit
- Two-phase commit protocol
- Three-phase commit protocol

# Common table expressions

---

A **Common Table Expression** (in SQL) is a temporary named result set, derived from a simple query and defined within the execution scope of a **SELECT**, **INSERT**, **UPDATE**, or **DELETE** statement.

CTE can be thought of as an alternatives to derived tables (subquery), views, and inline user-defined functions.

## Common table expressions

Common table expressions are supported by DB2, Firebird<sup>[1]</sup>, Microsoft SQL Server, Oracle and PostgreSQL.

Syntax:

```
WITH [RECURSIVE] with_query [, ...] SELECT...
```

with\_query looks like

```
with_query_name [ (column_name [...]) ] AS (SELECT ...)
```

## External links

- MSDN tutorial<sup>[2]</sup>

## References

[1] [http://en.wikipedia.org/wiki/Comparison\\_of\\_relational\\_database\\_management\\_systems#Database\\_capabilities](http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems#Database_capabilities)

[2] <http://msdn2.microsoft.com/en-us/library/ms190766.aspx>

# Condition (SQL)

---

A relational database management system uses SQL **conditions** or Expression (programming) in **WHERE** clauses and in **HAVING** clauses to **SELECT** subsets of data.

## Examples

To **SELECT** one row of data from a table *tab* with primary key column *pk* set to 100 — use the condition *pk = 100*:

```
SELECT * FROM tab WHERE pk = 100
```

To **SELECT** the duplicate rows of data from a table *tab* with duplicate key column *dk* set to 100 — use the condition *dk = 100* and the condition *having count(\*) > 1*:

```
SELECT * FROM tab WHERE dk = 100 having count(*) > 1
```

---

# Correlated subquery

---

A **correlated sub-query** is a term used for specific types of queries in SQL in computer databases. It is a sub-query (a query nested inside another query) that uses values from the outer query in its WHERE clause. The sub-query is evaluated once for each row processed by the outer query.

Here is an example for a typical correlated sub-query. In this example we are finding the list of employees (employee number and names) having more salary than the average salary of all employees in that employee's department.

```
SELECT empnum, name
FROM employee as e1
WHERE salary > (SELECT avg(salary)
                FROM employee
                WHERE department = e1.department);
```

In the above query the outer query is,

```
SELECT empnum, name
FROM employee as e1
WHERE salary >
```

And the inner query is,

```
(SELECT avg(salary)
 FROM employee
 WHERE department = e1.department);
```

In the above nested query the inner query has to be executed for every employee as the department will change for every row. Hence the average salary will also change.

## See also

- Select query

## External links

- <http://publib.boulder.ibm.com/infocenter/iserics/v5r3/index.jsp?topic=/sqlp/rbafycorrs.htm>

# CUBRID

---

<b>Developer(s)</b>	Search Solutions
<b>Written in</b>	C, C++, Java
<b>Operating system</b>	Cross-platform
<b>Available in</b>	English, Korean, Japanese, Chinese
<b>Type</b>	RDBMS
<b>License</b>	GNU General Public License for server engine and BSD license for APIs and CUBRID Manager tool
<b>Website</b>	<a href="http://www.cubrid.org">www.cubrid.org</a> <sup>[1]</sup>

**CUBRID** is a comprehensive open source and **complete free** relational database management system (RDBMS) **highly optimized for Web applications**, especially when complex business services process large amount of data and generate huge concurrent requests. By providing unique optimized features, CUBRID enables to process much more parallel requests at much less response time. With CUBRID, companies will benefit from superior performance and stability, scalability, and high availability required by organization to provide non-stop service for their valuable customers.

## Who else uses CUBRID?

CUBRID is being actively used by IT industry leader in Korea – NHN Corporation<sup>[2]</sup>, which breeds a farm of over 10,000 servers. In addition, large Hosting companies Cafe24<sup>[3]</sup> and Mireene<sup>[4]</sup>, Software Security company ESTsoft<sup>[5]</sup>, and many Korean local colleges manage their data with CUBRID. Two third of all CUBRID references come from the Government sector. Korea National Tax Service<sup>[6]</sup>, Korea Ministry of Public Administration and Security<sup>[7]</sup>, Korea Ministry of National Defense<sup>[8]</sup>, Busan Transportation Corporation<sup>[9]</sup>, and Korea White House<sup>[10]</sup> are major customers who deployed CUBRID as their major database management software. Just imagine how much sensitive data do they all have, how much security do they all require, and they all chose CUBRID as their database management solution. CUBRID helped many organizations gain greater results and achieve faster performance at a lower cost of ownership.

## License Policy

CUBRID, unlike other database systems, does not have an Enterprise version of its DBMS. CUBRID does not distinguish in its license policy<sup>[11]</sup> between Community and Enterprise. There is only one version of CUBRID DBMS, which is under GPL v2.0 or GPL v3.0. However, the GUI tools and API interfaces are distributed under Berkeley Software Distribution license. There is a reason for such a license policy. CUBRID wants all commercial companies to benefit from deploying its database as much as possible. BSD license allows companies to keep the source code of their proprietary software closed, thus they can make real profit.

---

## Linux Foundation Silver Sponsor

SAN FRANCISCO, April 7, 2010 – CUBRID became the newest member of The Linux Foundation<sup>[12]</sup>.

The company is joining the Linux Foundation to gain access to exclusive networking opportunities and face-to-face collaboration with members of the Linux community. Linux' strengths in the enterprise translate into major advantages for supporting web-based businesses. Its ability to enable seamless high-volume transactions and high performance server/client infrastructure are among the reasons CUBRID has become an active member of the Linux development community and the Linux Foundation.

"We are looking forward to collaboration with other Linux Foundation members at events where major Linux stakeholders meet face-to-face. It is this benefit that will help us advance our customers' web services strategies," said Byungjoo Chung<sup>[13]</sup>, CEO of CUBRID.

CUBRID also participated at the Linux Foundation Collaboration Summit on April 14–16, 2010 in San Francisco<sup>[14]</sup>.

## Product Name Origin

The name "CUBRID" is a combination of two words "Cube" and "Bridge". In case of CUBRID, "Bridge" stands of the "data bridge", while "Cube" is a sealed box which provides security for its contents. Thus, CUBRID means a secure software which holds sensitive information.

## Platforms and Interfaces

The CUBRID server and official libraries are implemented in C and C++, while CUBRID Manager, a Client Manager for CUBRID DBMS, is implemented in Java.

CUBRID runs on Linux and Microsoft Windows platforms and provides language-specific APIs, including JDBC, PHP, ODBC, and C-API.

Both CUBRID and its CUBRID Manager provide a command-line interface called CSQL and an front-end administration tool called SQL buddy for CUBRID<sup>[15]</sup> <sup>[16]</sup>.

## Features

The latest version as of December 2009 is CUBRID 2008 R2.1 including the following features;

- SQL 92 Standard
  - Transaction ACID support
  - Cross-platform support
  - Multiple granularity locking
  - Partition
  - Replication
  - High Availability support(shared-nothing clustering, fail-over and fail-back automatically)
  - Hot backup
  - Sub-SELECTs (i.e. nested SELECTs)
  - Hierarchical Query
  - Query plan caching
  - Triggers
  - Click counter
  - Updatable views
  - Java Stored procedures
  - True varchar support
-

- Cursors
- Updatable Views

[17]

## Product Development History

- 2006 - The development of CUBRID DBMS started.
- October, 2008 - First internal release followed by CUBRID 2008 R1.0 stable release.
- November, 2008 - CUBRID became an open source project. CUBRID 2008 R1.1 stable was released.
- January, 2009 - CUBRID 2008 R1.2 stable release.
- February, 2009 - CUBRID 2008 R1.3 stable release.
- March, 2009 - CUBRID 2008 R1.4 stable release.
- August, 2009 - CUBRID 2008 R2.0 stable release.
- September, 2009 - CUBRID Cluster Project has been started.
- October, 2009 - CUBRID Project <sup>[18]</sup> web site launched at Sourceforge.net. Opened Official Open Source Community web site at [www.cubrid.org](http://www.cubrid.org) <sup>[1]</sup>.
- December, 2009 - CUBRID 2008 R2.1 stable release <sup>[19]</sup>.
- February, 2010 - CUBRID Cluster *alpha version* was released.
- May, 2010 - CUBRID 2008 R2.2 stable release <sup>[20]</sup>.

## See also

- Comparison of relational database management systems
- Comparison of object-relational database management systems
- List of relational database management systems

Other RDBMS:

- Comparison of relational database management systems
- Drizzle
- Firebird (database server)
- HSQLDB
- Ingres (database)
- PostgreSQL

## External links

- CUBRID Community Web Site <sup>[1]</sup>
  - CUBRID Manual <sup>[21]</sup>
  - CUBRID Wiki <sup>[22]</sup>
  - Official CUBRID Blog <sup>[23]</sup>
  - CUBRID Forum <sup>[24]</sup>
  - CUBRID Trainings Channel on YouTube <sup>[25]</sup>
  - CUBRID Project site at Sourceforge.net <sup>[18]</sup>
  - CUBRID Project site at NAVER Development Center <sup>[26]</sup>
-

## References

- [1] <http://www.cubrid.org>
  - [2] <http://www.nhncorp.com/>
  - [3] <http://www.cafe24.com/>
  - [4] <http://www.mireene.com/>
  - [5] <http://www.estsoft.com/>
  - [6] <http://www.nts.go.kr/eng/default.html>
  - [7] <http://www.mopas.go.kr/gpms/ns/mogaha/user/nolayout/main/english/userEngMainDisplay.action>
  - [8] <http://www.mnd.go.kr/mndEng/main/index.jsp>
  - [9] <http://www.subway.busan.kr/english/main/>
  - [10] <http://english.president.go.kr/main.php>
  - [11] <http://www.cubrid.org/license>
  - [12] "CUBRID Joins Linux Foundation" (<http://www.linuxfoundation.org/node/6133>). Linux Foundation. . Retrieved 2010-05-12.
  - [13] <http://www.facebook.com/profile.php?id=100000684896663>
  - [14] "MySQL Conference Outcome" (<http://blog.cubrid.org/cubrid-life/mysql-conference-outcome/>). CUBRID Co., Ltd.. . Retrieved 2010-05-12.
  - [15] "CUBRID Introduction" (<http://cubrid.org/about>). CUBRID Co., Ltd.. . Retrieved 2009-12-29.
  - [16] "CUBRID Project site at Sourceforge.net" (<http://sourceforge.net/projects/cubrid>). CUBRID Co., Ltd.. . Retrieved 2009-12-29.
  - [17] "CUBRID Introduction" (<http://cubrid.org/about>). CUBRID Co., Ltd.. . Retrieved 2009-12-29.
  - [18] <http://sourceforge.net/projects/cubrid>
  - [19] "NAVER Developer's Center" (<http://dev.naver.com/projects/cubrid/downloads>). NHN Corp.. . Retrieved 2009-12-29.
  - [20] "New Version Release - CUBRID 2008 R2.2" (<http://blog.cubrid.org/notice/new-version-release-cubrid-2008-r2-2/>). CUBRID Co., Ltd.. . Retrieved 2010-05-07.
  - [21] <http://www.cubrid.org/manual/>
  - [22] <http://wiki.cubrid.org>
  - [23] <http://blog.cubrid.org>
  - [24] <http://forum.cubrid.org>
  - [25] <http://www.youtube.com/user/cubrid>
  - [26] <http://dev.naver.com/projects/cubrid>
-

# Cursor (databases)

---

In database packages, a **cursor** comprises a control structure for the successive traversal (and potential processing) of records in a result set.

Cursors provide a mechanism by which a database client iterates over the records in a database. Using cursors, the client can get, put, and delete database records. Database programmers use cursors for processing individual rows returned by the database system for a query. Cursors address the problem of impedance mismatch, an issue that occurs in many programming languages. Most procedural programming languages do not offer any mechanism for manipulating whole result-sets at once. In this scenario, the application must process rows in a result-set sequentially. Thus one can think of a database cursor as an iterator over the collection of rows in the result set.

Several SQL statements do not require the use of cursors. That includes the INSERT statement, for example, as well as most forms of the DELETE and UPDATE statements. Even a SELECT statement may not involve a cursor if it is used in the variation of SELECT INTO. A SELECT INTO retrieves at most a single row directly into the application.

## Working with cursors

This section introduces the ways the SQL:2003 standard defines how to use cursors in applications in embedded SQL. Not all application bindings for relational database systems adhere to that standard, and some (such as CLI or JDBC) use a different interface.

A programmer makes a cursor known to the DBMS by using a DECLARE ... CURSOR statement and assigning the cursor a (compulsory) name:

```
DECLARE cursor_name CURSOR FOR SELECT ... FROM ...
```

Before code can access the data, it must open the cursor with the OPEN statement. Directly following a successful opening, the cursor is positioned *before* the first row in the result set.

```
OPEN cursor_name
```

Programs position cursors on a specific row in the result set with the FETCH statement. A fetch operation transfers the data of the row into the application.

```
FETCH cursor_name INTO ...
```

Once an application has processed all available rows or the fetch operation is to be positioned on a non-existing row (compare scrollable cursors below), the DBMS returns a SQLSTATE '02000' (usually accompanied by an SQLCODE +100) to indicate the end of the result set.

The final step involves closing the cursor using the CLOSE statement:

```
CLOSE cursor_name
```

After closing a cursor, a program can open it again, which implies that the DBMS re-evaluates the same query or a different query and builds a new result-set.

---



## Scrollable cursors

Programmers may declare cursors as scrollable or not scrollable. The scrollability indicates the direction in which a cursor can move.

With a **non-scrollable** cursor, also known as *forward-only*, one can FETCH each row at most once, and the cursor automatically moves to the immediately following row. A fetch operation after the last row has been retrieved positions the cursor after the last row and returns SQLSTATE 02000 (SQLCODE +100).

A program may position a scrollable cursor anywhere in the result set using the FETCH SQL statement. The keyword SCROLL must be specified when declaring the cursor. The default is NO SCROLL, although different language bindings like JDBC may apply different default.

```
DECLARE cursor_name sensitivity SCROLL CURSOR FOR SELECT ... FROM ...
```

The target position for a scrollable cursor can be specified relative to the current cursor position or absolute from the beginning of the result set.

```
FETCH [ NEXT | PRIOR | FIRST | LAST ] FROM cursor_name
```

```
FETCH ABSOLUTE n FROM cursor_name
```

```
FETCH RELATIVE n FROM cursor_name
```

Scrollable cursors can potentially access the same row in the result set multiple times. Thus, data modifications (insert, update, delete operations) from other transactions could have an impact on the result set. A cursor can be SENSITIVE or INSENSITIVE to such data modifications. A sensitive cursor picks up data modifications impacting the result set of the cursor, and an insensitive cursor does not. Additionally, a cursor may be ASENSITIVE, in which case the DBMS tries to apply sensitivity as much as possible.

## "WITH HOLD"

Cursors are usually closed automatically at the end of a transaction, i.e when a COMMIT or ROLLBACK (or an implicit termination of the transaction) occurs. That behavior can be changed if the cursor is declared using the WITH HOLD clause. (The default is WITHOUT HOLD.) A holdable cursor is kept open over COMMIT and closed upon ROLLBACK. (Some DBMS deviate from this standard behavior and also keep holdable cursors open over ROLLBACK.)

```
DECLARE cursor_name CURSOR WITH HOLD FOR SELECT ... FROM ...
```

When a COMMIT occurs, a holdable cursor is positioned *before* the next row. Thus, a positioned UPDATE or positioned DELETE statement will only succeed after a FETCH operation occurred first in the transaction.

Note that JDBC defines cursors as holdable per default. This is done because JDBC also activates auto-commit per default. Due to the usual overhead associated with auto-commit and holdable cursors, both features should be explicitly deactivated at the connection level.

## Positioned update/delete statements

Cursors can not only be used to fetch data from the DBMS into an application but also to identify a row in a table to be updated or deleted. The SQL:2003 standard defines positioned update and positioned delete SQL statements for that purpose. Such statements do not use a regular WHERE clause with predicates. Instead, a cursor identifies the row. The cursor must be opened and positioned on a row already using the FETCH statement.

```
UPDATE table_name
SET    . . .
WHERE  CURRENT OF cursor_name
```

```
DELETE
FROM   table_name
WHERE  CURRENT OF cursor_name
```

The cursor must operate on an updatable result set in order to successfully execute a positioned update or delete statement. Otherwise, the DBMS would not know how to apply the data changes to the underlying tables referred to in the cursor.

## Cursors in distributed transactions

Using cursors in distributed transactions (X/Open XA Environments), which are controlled using a transaction monitor, is no different than cursors in non-distributed transactions.

One has to pay attention when using holdable cursors, however. Connections can be used by different applications. Thus, once a transaction has been ended and committed, a subsequent transaction (running in a different application) could inherit existing holdable cursors. Therefore, an application developer has to be aware of that situation.

## Cursors in XQuery

The XQuery language allows cursors to be created using the **subsequence()** function.

The format is:

```
let $displayed-sequence := subsequence($result, $start, $item-count)
```

Where **\$result** is the result of the initial XQuery, **\$start** is the item number to start and **\$item-count** is the number of items to return.

Equivalently this can also be done using a predicate:

```
let $displayed-sequence := $result[$start to $end]
```

Where **\$end** is the end sequence.

For complete examples see the XQuery Wikibook<sup>[1]</sup>.

## Disadvantages of cursors

The following information may vary from database system to database system.

Fetching a row from the cursor may result in a network round trip each time. This uses much more network bandwidth than would ordinarily be needed for the execution of a single SQL statement like DELETE. Repeated network round trips can severely impact the speed of the operation using the cursor. Some DBMSs try to reduce this impact by using block fetch. Block fetch implies that multiple rows are sent together from the server to the client. The client stores a whole block of rows in a local buffer and retrieves the rows from there until that buffer is exhausted.

Cursors allocate resources on the server, for instance locks, packages, processes, temporary storage, etc. For example, Microsoft SQL Server implements cursors by creating a temporary table and populating it with the query's result-set. If a cursor is not properly closed (*deallocated*), the resources will not be freed until the SQL session (connection) itself is closed. This wasting of resources on the server can not only lead to performance degradations but also to failures.

## See also

- Iterator

## References

- Christopher J. Date: *Database in Depth*, O'Reilly & Associates, ISBN 0-596-10012-4
- Thomas M. Connolly, Carolyn E. Begg: *Database Systems*, Addison-Wesley, ISBN 0-321-21025-5
- Ramiz Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems*, Addison-Wesley, ISBN 0-201-54263-3
- Neil Matthew, Richard Stones: *Beginning Databases with PostgreSQL: From Novice to Professional*, Apress, ISBN 1-59059-478-9
- Thomas Kyte: *Expert One-On-One: Oracle*, Apress, ISBN 1-59059-525-4
- Kevin Loney: *Oracle Database 10g: The Complete Reference*, Oracle Press, ISBN 0-07-225351-7

## External links

- Cursor Optimization Tips (for MS SQL Server)<sup>[2]</sup>
  - Descriptions from Portland Pattern Repository<sup>[3]</sup>
  - PostgreSQL Documentation<sup>[4]</sup>
  - Berkeley DB Reference Guide: Cursor operations<sup>[5]</sup>
  - ResultSet in JDBC<sup>[6]</sup>
  - Q3SqlCursor Class Reference<sup>[7]</sup>
  - OCI Scrollable Cursor<sup>[8]</sup>
  - function `oci_new_cursor`<sup>[9]</sup>
  - MySQL's Cursor Documentation<sup>[10]</sup>
  - Cursors in DB2 CLI applications<sup>[11]</sup>; Cursors in DB2 SQL stored procedures<sup>[12]</sup>
-

## References

- [1] [http://en.wikibooks.org/wiki/XQuery/Searching,Paging\\_and\\_Sorting#Paging](http://en.wikibooks.org/wiki/XQuery/Searching,Paging_and_Sorting#Paging)
- [2] <http://www.mssqlcity.com/Tips/tipCursor.htm>
- [3] <http://c2.com/cgi/wiki?DistributedCursor>
- [4] <http://www.postgresql.org/docs/8.3/interactive/plpgsql-cursors.html>
- [5] <http://sleepycat.com/docs/ref/am/cursor.html>
- [6] <http://java.sun.com/javase/6/docs/technotes/guides/jdbc/getstart/resultset.html>
- [7] <http://doc.trolltech.com/4.0/q3sqlcursor.html>
- [8] <http://www.oracle.com/technology/products/oracle9i/daily/mar15.html>
- [9] <http://de2.php.net/manual/en/function.oci-new-cursor.php>
- [10] <http://dev.mysql.com/doc/refman/5.0/en/cursors.html>
- [11] <http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.cli.doc/doc/c0007645.htm>
- [12] <http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.sql.doc/doc/c0024361.htm>

## Data Control Language

---

A **Data Control Language (DCL)** is a computer language and a subset of SQL, used to control access to data in a database.

Examples of DCL commands include:

- GRANT to allow specified users to perform specified tasks.
- REVOKE to cancel previously granted or denied permissions.

The following privileges can be GRANTED TO or REVOKED FROM a user or role:

- CONNECT
- SELECT
- INSERT
- UPDATE
- DELETE
- EXECUTE
- USAGE

In Oracle, executing a DCL command issues an implicit commit.

In PostgreSQL, executing DCL is transactional, and can be rolled back.

### See also

- Data Definition Language
  - Data Manipulation Language
-

# Data Definition Language

A **Data Definition Language (DDL)** is a computer language for defining data structures. The term was first introduced in relation to the Codasyl database model, where the schema of the database was written in a Data Definition Language describing the records, fields, and "sets" making up the user Data Model. Initially it referred to a subset of SQL, but is now used in a generic sense to refer to any formal language for describing data or information structures, like XML schemas.

## SQL

Initially, DDL was a subset of SQL statements. perhaps the most common code is CREATE TABLE code in this kind of statements.

### CREATE statements

**Create** - To make a new database, table, index, or stored query. A CREATE statement in SQL creates an object inside of a relational database management system (RDBMS). The types of objects that can be created depends on which RDBMS is being used, but most support the creation of tables, indexes, users, synonyms and databases. Some systems (such as PostgreSQL) allow CREATE, and other DDL commands, inside of a transaction and thus they may be rolled back.

#### CREATE TABLE statement

Perhaps the most common CREATE command is the CREATE TABLE command. The typical usage is:

CREATE [TEMPORARY] TABLE *[table name]* ( *[column definitions]* ) *[table parameters]*.

**Column Definitions:** A comma-separated list consisting of any of the following

- Column definition: *[column name]**[data type]* {NULL | NOT NULL} *{column options}*
- Primary key definition: PRIMARY KEY ( *[comma separated column list]* )
- CONSTRAINTS: {CONSTRAINT} *[constraint definition]*
- RDBMS specific functionality

For example, the command to create a table named **employees** with a few sample columns would be:

```
CREATE TABLE employees (
  id          INTEGER PRIMARY KEY,
  first_name  CHAR(50)  null,
  last_name   CHAR(75)  not null,
  dateofbirth DATE      null
);
```

### DROP statements

**Drop** - To destroy an existing database, table, index, or view.

A **DROP** statement in SQL removes an object from a relational database management system (RDBMS). The types of objects that can be dropped depends on which RDBMS is being used, but most support the dropping of tables, users, and databases. Some systems (such as PostgreSQL) allow DROP and other DDL commands to occur inside of a transaction and thus be rolled back.

The typical usage is simply DROP *objecttype objectname*. For example, the command to drop a table named **employees** would be:

```
DROP TABLE employees;
```

The DROP statement is distinct from the DELETE and (non-standard) TRUNCATE statements, in that they do not remove the table itself. For example, a DELETE statement might delete some (or all) data from a table while leaving the table itself in the database, whereas a DROP statement would remove the entire table from the database.

## ALTER statements

Alter - To modify an existing database object.

An ALTER statement in SQL changes the properties of an object inside of a relational database management system (RDBMS). The types of objects that can be altered depends on which RDBMS is being used.

The typical usage is ALTER *objecttype objectname parameters*. For example, the command to add (then remove) a column named **bubbles** for an existing table named **sink** would be:

```
ALTER TABLE sink ADD bubbles INTEGER;  
ALTER TABLE sink DROP COLUMN bubbles;
```

## Referential integrity statements

Finally, other kind of DDL sentence in SQL are the statements to define referential integrity relationships, usually implemented as primary key and foreign key tags in some columns of the tables.

These two statements can be included inside a CREATE TABLE or an ALTER TABLE sentence.

## XML Schema

XML Schema is an example of a pure DDL (although only relevant in the context of XML).

## DDL Tools and Related Applications

### Apache DdlUtils

Apache DdlUtils <sup>[1]</sup> is a small, easy-to-use component for working with Database Definition (DDL) files. These are XML files that contain the definition of a database schema, e.g. tables and columns. These files can be fed into DdlUtils via its Ant task or programmatically in order to create the corresponding database or alter it so that it corresponds to the DDL. Likewise, DdlUtils can generate a DDL file for an existing database.

## See also

- Data Manipulation Language
- Data Control Language

## References

[1] <http://db.apache.org/ddlutils/>

# Data Manipulation Language

---

**Data Manipulation Language (DML)** is a family of computer languages used by computer programs and/or database users to insert, delete and update data in a database. Read-only querying, i.e. SELECT, of this data may be considered to be either part of DML or outside it, depending on the context.

Currently the most popular data manipulation language is that of SQL, which is used to retrieve and manipulate data in a Relational database.<sup>[1]</sup> Other forms of DML are those used by IMS/DLI, CODASYL databases (such as IDMS), and others.

Data Manipulation Language comprises the 'SQL-data change' statements<sup>[2]</sup>, which modify stored data but not the schema or database objects. Manipulation of persistent database *objects* (e.g. tables or stored procedures) via the 'SQL-schema' statements<sup>[2]</sup>, rather than the *data* stored within them, is considered to be part of a separate Data Definition Language. In SQL these two categories are similar in their detailed syntax, data types, expressions etc., but distinct in their overall function.<sup>[2]</sup>

Data Manipulation Languages have their functional capability organized by the initial word in a statement, which is almost always a verb. In the case of SQL, these verbs are:

- SELECT ... FROM ... WHERE ...
- INSERT INTO ... VALUES ...
- UPDATE ... SET ... WHERE ...
- DELETE FROM ... WHERE ...

The purely read-only SELECT query statement is classed with the 'SQL-data' statements<sup>[2]</sup> and so is considered by the standard to be outside of DML. The SELECT ... INTO form is considered to be DML because it manipulates (i.e. modifies) data. In common practice though, this distinction is not made and SELECT is widely considered to be part of DML.<sup>[3]</sup>

Most SQL database implementations extend their SQL capabilities by providing imperative, i.e., procedural, languages. Examples of these are Oracle's PL/SQL and DB2's SQL PL.

Data manipulation languages tend to have many different flavors and capabilities between database vendors. There have been a number of standards established for SQL by ANSI,<sup>[1]</sup> but vendors still provide their own extensions to the standard while not implementing the entire standard.

There are two types of data manipulation languages:

- Procedural
- Declarative

Each SQL DML statement is a declarative command. The individual SQL statements are declarative, as opposed to imperative, in that they describe *what* the program should accomplish, rather than describing *how* to go about accomplishing it.

Data manipulation languages were initially only used by computer programs, but (with the advent of SQL) have come to be used by people as well.

## See also

- Data Definition Language
- Data Control Language

## References

- "The SQL92 standard" <sup>[4]</sup>.

[1] SQL92

[2] SQL92 4.22.2, SQL-statements classified by function

[3] "Data Manipulation Language Statements" ([http://download.oracle.com/docs/cd/B19306\\_01/server.102/b14220/sqlplsql.htm#i18503](http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/sqlplsql.htm#i18503)). Oracle. . "Data manipulation language (DML) statements *query or manipulate* data in existing schema objects."

[4] <http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt>

# Database Console Commands (Transact-SQL)

The **Database Console Commands (DBCC)** are a series of statements in Transact-SQL programming language to check the physical and logical consistency of a Microsoft SQL Server database.<sup>[1]</sup> These commands are also used to fix existing issues<sup>[1]</sup>. They are also used for administration and file management.<sup>[2]</sup>

DBCC was previously expanded as **Database Consistency Checker**.<sup>[3]</sup>

## Categories of DBCC Commands

Based on their uses, DBCC commands are made of three categories of statements. They are:

Category	Uses	Commands
Maintenance statements	Maintenance tasks	DBCC DBREINDEX, DBCC DBREPAIR, DBCC INDEXDEFRAG, DBCC SHRINKDATABASE, DBCC SHRINKFILE, DBCC UPDATEUSAGE, DBCC CLEANBETWEEN, DBCC DROPCLEANBUFFERS, DBCC FREEPROCCACHE
Status statements	Status checks	DBCC INPUTBUFFER, DBCC OPENTRAN, DBCC OUTPUTBUFFER, DBCC PROCCACHE, DBCC SHOWCONTIG, DBCC SHOW_STATISTICS, DBCC SQLPERF, DBCC TRACESTATUS, DBCC USEROPTIONS
Validation statements	Validation operations on a database and database components such as table, index, file catalog, <sup>[1]</sup> etc	DBCC CHECKALLOC, DBCC CHECKCATALOG, DBCC CHECKCONSTRAINTS, DBCC CHECKDB, DBCC CHECKFILEGROUP, DBCC CHECKIDENT, DBCC CHECKTABLE, DBCC NEWALLOC
Miscellaneous statements	Miscellaneous tasks	DBCC dlname (FREE), DBCC HELP, DBCC PINTABLE, DBCC ROWLOCK, DBCC TRACEOFF, DBCC TRACEON, DBCC UNPINTABLE
Source: MSDN Transact-SQL Reference (SQL Server 2000)		



## Operation of DBCC statements

### DBCC DBREINDEX

This statement is used to recreate the indexes for a particular table.<sup>[4]</sup> This statement rebuilds indexes in a single step.<sup>[5]</sup> It also assigns fresh pages to reduce internal and external fragmentation.<sup>[5]</sup>

### DBCC DBREPAIR

This statement is used to drop or delete a damaged database<sup>[6]</sup>. However, this command is no longer available with Microsoft SQL Server 2005 and later versions of Microsoft SQL Server<sup>[7]</sup>. Instead, it has been replaced by the DROP DATABASE Transact-SQL statement<sup>[7]</sup>

### DBCC INDEXDEFRAG

This statement is used to defragment the clustered and secondary indexes associated with the particular table.<sup>[8]</sup> The index defragmentation is carried out using the fill factor specified at the time of creation of indexes.<sup>[9]</sup> While its operation is strikingly similar to that of DBCC DBREINDEX, unlike DBCC INDEXFRAG it does not allow new fill factor to be specified.<sup>[9]</sup>

### DBCC SHRINKDATABASE

This statement is used to reduce the size of a database<sup>[10]</sup>. This statement reduces the physical size of the database log file<sup>[11] [12]</sup>. An alternate way to shrink a database is to use the commander ALTER DATABASE.<sup>[13]</sup>

### DBCC SHRINKFILE

This statement is used to reduce the size of a data file or log file of a particular database.<sup>[14] [15]</sup> The file could also be shrunk by using the SHRINKFILE attribute of the ALTER DATABASE command.<sup>[13]</sup>

### DBCC UPDATEUSAGE

This statement is used to correct inaccuracies in the page and row statistics in the views.<sup>[16]</sup>

### DBCC CLEANTABLE

This statement is used to remove spaces occupied by columns when they are removed.<sup>[17]</sup> This feature is not available with Microsoft SQL Server 2000 and has been newly introduced in Microsoft SQL Server 2005<sup>[17]</sup>

### DBCC DROPCLEANBUFFERS

This statement is used to drop clean buffers from the buffer pool.<sup>[18]</sup> This feature is not available with Microsoft SQL Server 2000 and has been newly introduced in Microsoft SQL Server 2005<sup>[18]</sup>

### DBCC FREEPROCCACHE

This statement is used to remove all elements from the procedure cache.<sup>[19]</sup> This feature is not available with Microsoft SQL Server 2000 and has been newly introduced in Microsoft SQL Server 2005<sup>[19]</sup>

### DBCC INPUTBUFFER

This statement is used to display the last statement stored in the buffer.<sup>[20]</sup>

---

**DBCC OPENTRAN**

This statement is used to display information about the oldest open transaction.<sup>[21]</sup>

**DBCC OUTPUTBUFFER**

This statement is used to return the current value of the output buffer.<sup>[22]</sup>

**DBCC PROCCACHE**

This statement is used to display information about procedure cache.<sup>[23]</sup>

**DBCC SHOWCONTIG**

This statement is used to display fragmentation information<sup>[23]</sup>

**DBCC SHOW\_STATISTICS**

This statement is used to show current distribution statistics<sup>[24]</sup>

**DBCC SQLPERF**

This statement is used to show transaction log statistics<sup>[25]</sup>

**DBCC TRACESTATUS**

This statement is used to display status of trace flags<sup>[26]</sup>

**DBCC USEROPTIONS**

This statement is used to return set as ACTIVE<sup>[27]</sup>

**DBCC CHECKALLOC**

This statement is used to check whether every extent allocated by the system has been allocated and whether there are extents that have not been allocated.<sup>[28]</sup>

**DBCC CHECKCATALOG**

This statement is used to check for consistency between system tables<sup>[29]</sup> in the system catalog. It does so through cross-referencing checks.<sup>[28]</sup>

**DBCC CHECKCONSTRAINTS**

This statement is used to check integrity of specific constraints.<sup>[30]</sup>

**DBCC CHECKDB**

This statement is used to check integrity and allocation of specific objects in database.<sup>[31]</sup> It also performs DBCC CHECKALLOC, DBCC CHECKTABLE and DBCC CHECKCATALOG in the particular order.<sup>[28]</sup>

---

## **DBCC CHECKFILEGROUP**

This statement is used to check allocation and structural integrity of tables.<sup>[32]</sup>

## **DBCC CHECKIDENT**

This statement is used to check identity value of specified table.<sup>[33]</sup>

## **DBCC CHECKTABLE**

This statement is used to check the integrity of a table<sup>[34]</sup> and all the pages and structures which comprise the table.<sup>[28]</sup> Both physical and logical checks are performed in this case.<sup>[28]</sup> However, we can also use a PHYSICAL ONLY option to check for physical consistency alone.<sup>[28]</sup>

## **DBCC NEWALLOC**

DBCC NEWALLOC is almost similar to DBCC CHECKALLOC. This statement is not supported by recent versions.<sup>[35]</sup>

## **DBCC dlname (FREE)**

This statement is used to unload a particular stored procedure DLL from memory.<sup>[35]</sup>

## **DBCC HELP**

This statement is used to return syntax information.<sup>[36]</sup>

## **DBCC PINTABLE**

This statement is used to mark a particular table to be pinned.<sup>[37]</sup>

## **DBCC ROWLOCK**

This statement is used to enable Insert Row Locking (IRL) operations.<sup>[38]</sup>

## **DBCC TRACEOFF**

This statement is used to disable a trace flag.<sup>[39]</sup>

## **DBCC TRACEON**

This statement is used to turn on a specific trace flag.<sup>[40]</sup>

## **DBCC UNPINTABLE**

This statement is used to mark a table as unpinned. In an unpinned table, the table pages in the cache could be easily removed.<sup>[41]</sup>

---

## Running a Database Console Command

A database console command could be run from (i) the command window or (ii) query analyzer window.<sup>[42]</sup>

## Advantages of Database Console Commands

Database Console Commands have a number of advantages. Their use is extremely essential in some instances

- Occasionally, there have been bad allocations of database pages.<sup>[42]</sup>
- Indexes could be destroyed or corrupted easily.<sup>[42]</sup>
- There could be misunderstandings on part of the SQL server engine.<sup>[42]</sup>
- There could be problems when a large number of updates need to be carried out.<sup>[42]</sup>
- Individual pages may lose their optimal storage footprint.<sup>[42]</sup>

## References

- *Microsoft SQL Server 2000 Database Design and Implementation*. Microsoft Press
- *Designing SQL Server 2000 Databases for .NET Enterprise Servers*. Syngress. 2001. pp. 285–286. ISBN 1928994199, ISBN 9781928994190.
- *Mastering Microsoft SQL Server 2005*. John Wiley and Sons. 2006. ISBN 0782143806, ISBN 9780782143805.
- Dušan Petkovic (2008). *Microsoft SQL Server 2008: A Beginner's Guide: A Beginner's Guide*. McGraw-Hill Professional. ISBN 0071546383, ISBN 9780071546386.
- Sajal Dam (2004). *SQL Server Query Performance Tuning Distilled*. Apress. ISBN 1590594215, ISBN 9781590594216.

## References

- [1] "DBCC - Transact-SQL Reference (SQL Server 2000)" ([http://msdn.microsoft.com/en-us/library/aa258281\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258281(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-20.
- [2] Ogle, Pg 285
- [3] *Microsoft SQL Server 2000 unleashed*. Sams Publishing. 2003. pp. 365. ISBN 0672324679, ISBN 9780672324673.
- [4] "DBCC DBREINDEX, Transact-SQL Reference (SQL Server 2000)" ([http://msdn.microsoft.com/en-us/library/aa258828\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258828(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-21.
- [5] Dam, Pg 230
- [6] "DBCC DBREPAIR, Transact-SQL Reference (SQL Server 2000)" ([http://msdn.microsoft.com/en-us/library/aa258827\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258827(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [7] "DBCC DBREPAIR, Transact-SQL Reference (SQL Server 2005)" (<http://msdn.microsoft.com/en-us/library/ms174416.aspx>). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [8] "DBCC INDEXDEFRAG, Transact-SQL Reference (SQL Server 2000)" ([http://msdn.microsoft.com/en-us/library/aa258286\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258286(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [9] Dam, Pg 236
- [10] Microsoft SQL Server 2000, Pg 142
- [11] Microsoft SQL Server 2000, Pg 398
- [12] Microsoft SQL Server 2000, Pg 402
- [13] Sack, Pg 568
- [14] "DBCC SHRINKFILE, Transact-SQL Reference (SQL Server 2005)" (<http://msdn.microsoft.com/en-us/library/ms189493.aspx>). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [15] Microsoft SQL Server 2000, Pg 367 - 370
- [16] "DBCC UPDATEUSAGE, Transact-SQL Reference (SQL Server 2005)" (<http://msdn.microsoft.com/en-us/library/ms188414.aspx>). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [17] "DBCC CLEANTABLE, Transact-SQL Reference (SQL Server 2005)" (<http://msdn.microsoft.com/en-us/library/ms174418.aspx>). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [18] "DBCC DROPCLEANBUFFERS, Transact-SQL Reference (SQL Server 2005)" (<http://msdn.microsoft.com/en-us/library/ms187762.aspx>). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
- [19] "DBCC FREEPROCCACHE, Transact-SQL Reference (SQL Server 2005)" (<http://msdn.microsoft.com/en-us/library/ms174283.aspx>). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.

- [20] "DBCC INPUTBUFFER, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258826\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258826(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [21] "DBCC OPENTRAN, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258815\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258815(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [22] "DBCC OUTPUTBUFFER, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258810\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258810(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [23] "DBCC PROCCACHE, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258820\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258820(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [24] "DBCC SHOWSTATISTICS, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258821\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258821(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [25] "DBCC SQLPERF, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258819\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258819(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [26] "DBCC TRACESTATUS, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258797\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258797(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [27] "DBCC USEROPTIONS, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258811\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258811(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [28] Petkovic, Pg 403
  - [29] "DBCC CHECKCATALOG, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258813\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258813(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [30] "DBCC CHECKCONSTRAINTS, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258282\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258282(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [31] "DBCC CHECKDB, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258278\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258278(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [32] "DBCC CHECKFILEGROUP, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258818\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258818(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [33] "DBCC CHECKIDENT, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258817\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258817(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [34] "DBCC CHECKTABLE, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258646\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258646(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [35] "DBCC NEWALLOC, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258280\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258280(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [36] "DBCC NEWALLOC, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258825\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258825(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [37] "DBCC PINTABLE, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258284\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258284(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [38] "DBCC ROWLOCK, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258812\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258812(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [39] "DBCC TRACEOFF, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258288\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258288(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [40] "DBCC TRACEON, Transact-SQL Reference (SQL Server 2005)" ([http://msdn.microsoft.com/en-us/library/aa258823\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258823(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [41] "[http://msdn.microsoft.com/en-us/library/aa258816\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258816(SQL.80).aspx)" ([http://msdn.microsoft.com/en-us/library/aa258823\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258823(SQL.80).aspx)). Microsoft Developer Network (MSDN). . Retrieved 2008-06-23.
  - [42] Arthur Fuller (December 21, 2006). "Get out of a jam by using SQL Server's DBCC" ([http://articles.techrepublic.com.com/5100-10878\\_11-6142604.html](http://articles.techrepublic.com.com/5100-10878_11-6142604.html)). .
-

# DbForge Studio for MySQL

---

	
<b>Developer(s)</b>	Devart
<b>Stable release</b>	4.50 Beta / 06/02/2010
<b>Development status</b>	Active
<b>Written in</b>	C#
<b>Operating system</b>	Microsoft Windows
<b>Size</b>	23,1MB
<b>Available in</b>	English, Russian
<b>Type</b>	RDBMS
<b>License</b>	Commercial and Non-commercial
<b>Website</b>	<a href="http://www.devart.com/dbforge/mysql/studio">www.devart.com/dbforge/mysql/studio</a> [1]

**dbForge Studio for MySQL** is a database development and administration software for the popular Relational Database Management System (RDBMS) MySQL created by Devart.

## Editions (licenses)

dbForge Studio for MySQL has three editions: Express, Standard, and Professional.

**Express** is a free software, which provides basic functionality for working with schema objects, user accounts, and SQL scripts.

**Standard** includes all must-have tools for database developers, such as a debugger, a Query Builder, code templates, object search, various export and maintenance wizards.

**Professional** is a fully-featured software for professional work with database projects, comparing schema and project data, debugging stored procedures and scripts, creating complex queries, and etc.

## History

The first version was released on 12 April 2005 under the name "**MyDeveloper Studio**".

Through 3 years of development the product has raised up to the version 3.0 (released on 01-August-2008), and got a new name - "**dbForge Studio for MySQL**" as well as a lot of new features and improvements<sup>[2]</sup>.

## Functionality

**dbForge Studio for MySQL** has got both standard features of most database tools and unique functionality.

Main features of dbForge Studio for MySQL, v4.50 include:

- Query Profiler - a new tool for database developers that shows results of internal MySQL tools like SHOW PROFILE and EXPLAIN as well as STATUS variables changes due to query execution in a convenient and clear GUI.

- Database Backup and Restore wizards - enable users to back up and restore schemas in automatic mode using Windows task scheduler, save backup options for future use, view automatically compiled log file. Besides old backup files are automatically removed based on date or quantity.
- Database Designer - a visual online database design tool for MySQL. It provides the complete picture of all the tables, foreign key relations between them, views, and stored routines of the required database; enables reverse engineering of databases to IDEF1X or IE diagrams, which can be easily printed.
- Database project for offline database development - includes the capability to integrate various SQL and script files into a single project, group them into folders, and manage in one place. You can build your project to one script file or several ones and easily deploy on the server.
- Advanced stored routine debugger - uses database server as an engine for checking user created SQL scripts and automates debugging of stored procedures, functions, and triggers.
- Schema/data comparison and synchronization tools - provide quick and clear performance, guarantee convenient analysis and management of comparison results in windows and the desired synchronization result on the first try.
- SQL code editor - includes context-sensitive code completion, automatic SQL syntax check and highlighting, customizable SQL formatting, quick code navigation, etc.
- Visual Query Builder - creates the most complex queries in a state-of-the-art diagram.
- Convenient schema object editors - simplify management of tables, views, procedures, functions, triggers, events, and user-defined functions (UDF).
- Integrated Security Manager - automates management of users and their privileges.
- Server maintenance toolkit - contains the table maintenance wizard for check, analysis, repair, and optimisation of tables and service control for starting and stopping MySQL servers.
- SQL template library: an integrated library of common SQL script templates.
- Schema object search engine: built-in schema objects search.
- Multi-format data export: customizable data export to widely-used formats: CSV, DBF, HTML, MS Access, MS Excel, ODBC, PDF, RTF, Text, and XML.

## Current version

The latest version of dbForge Studio for MySQL is 4.50. It features the following:

- Database Backup and Restored wizards
  - Query Profiler tool
  - Capability to compare and synchronize data in databases of any size
  - Advanced query building
  - Quick generating template SQL scripts for database objects
  - Extended capabilities of Schema Comparison wizard
-

## See also

- Comparison of database tools
- Devart (company)

## External links

- Devart dbForge for MySQL Studio homepage<sup>[1]</sup>
- Official Devart website<sup>[3]</sup>
- dbForge Studio for MySQL page at MySQL partners portal<sup>[4]</sup>
- Codeproject.com press room - About dbForge Studio for MySQL 3.50<sup>[5]</sup>

## References

- [1] <http://www.devart.com/dbforge/mysql/studio/>
- [2] dbForge Studio for MySQL release news (<http://devart.com/news/2010/dbforigestudio450b.html>), *Devart*
- [3] <http://www.devart.com/>
- [4] <http://solutions.mysql.com/solutions/item.php?id=849>
- [5] <http://www.codeproject.com/PressReleases/766/Devart-Releases-dbForge-Studio-for-MySQL-v-3-50-with-State-of-the-Art-Database-Designer.aspx>

# Declarative Referential Integrity

---

**Declarative Referential Integrity** (DRI) is one of the techniques in the SQL database programming language to ensure data integrity.

## Meaning in SQL

A table (called the child table) can refer to a column (or a group of columns) in another table (the parent table) by using a foreign key. The referenced column(s) in the parent table must be under a unique constraint, such as a primary key. Also, self-references are possible (not fully implemented in MS SQL Server though<sup>[1]</sup>). On inserting a new row into the child table, the relational database management system (RDBMS) checks if the entered key value exists in the parent table. If not, no insert is possible. It is also possible to specify DRI actions on UPDATE and DELETE, such as CASCADE (forwards a change/delete in the parent table to the child tables), NO ACTION (if the specific row is referenced, changing the key is not allowed) or SET NULL / SET DEFAULT (a changed/deleted key in the parent table results in setting the child values to NULL or to the DEFAULT value if one is specified).



## Product specific meaning

In Microsoft SQL Server the term DRI also applies to the assigning of permissions to users on a database object. Giving DRI permission to a database user allows them to add foreign key constraints on a table.<sup>[2]</sup>

## External links

- DRI versus Triggers <sup>[3]</sup>

## References

- [1] Microsoft Support (2007-02-11). "Error message 1785 occurs when you create a FOREIGN KEY constraint that may cause multiple cascade paths" (<http://support.microsoft.com/kb/321843/en-us>). microsoft.com. . Retrieved 2009-01-24.
- [2] Chigrik, Alexander (2003-08-13). "Managing Users Permissions on SQL Server" (<http://www.databasejournal.com/features/mssql/article.php/2246271>). Database Journal. . Retrieved 2006-12-17.
- [3] <http://www.cvalde.net/document/declaRefIntegVsTrig.htm>
-

# Devgems Data Modeler

---

<b>Developer(s)</b>	Devgems
<b>Stable release</b>	1.0
<b>Operating system</b>	Windows 2000, XP and Vista
<b>License</b>	Proprietary
<b>Website</b>	<a href="http://www.devgems.com">www.devgems.com</a> <sup>[1]</sup>

**Devgems Data Modeler** is a database design software targeting multiple database systems (MS SQL Server, Firebird, and others). It is used by data modelers, database administrators and software developers to create and manage database models and integrate with existing software development tools like CodeGear Delphi.

## Key Features

- Reverse engineering of existing databases.
- SQL (DDL) scripts generation to create databases.
- Version control and Alter SQL generation.
- Visual structure comparison.
- Target multiple RDBMS.

## Ribbon interface

Devgems Data Modeler is one of the first database design and modeling software to provide the Ribbon Interface<sup>[2]</sup>.

## Competitors

Similar software tools that are competitors are CA ERwin Data Modeler, ER/Studio, Toad Data Modeler, DeZign for Databases, among others.

## See also

- Computer-aided software engineering
  - Relational Model
  - Data modeling
  - RDBMS
  - Comparison of database tools
  - SQL Programming Tool
-

## External links

- Official page <sup>[1]</sup>
- Product feature list <sup>[3]</sup>

## References

[1] <http://www.devgems.com>

[2] Office Fluent User Interface (<http://msdn2.microsoft.com/en-us/office/aa905530.aspx>)

[3] <http://www.devgems.com/data-modeler-features.php>

# Embedded SQL

---

**Embedded SQL** is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL. Embedded SQL statements are SQL statements written inline with the program source code of the host language. The embedded SQL statements are parsed by an embedded SQL preprocessor and replaced by host-language calls to a code library. The output from the preprocessor is then compiled by the host compiler. This allows programmers to embed SQL statements in programs written in any number of languages such as: C/C++, COBOL and Fortran.

The ANSI SQL standards committee defined the embedded SQL standard in two steps: a formalism called **Module Language** was defined, then the embedded SQL standard was derived from Module Language.<sup>[1]</sup> The SQL standard defines embedding of SQL as *embedded SQL* and the language in which SQL queries are embedded is referred to as the *host language*. A popular host language is C. The mixed C and embedded SQL is called Pro\*C in Oracle and Sybase database management systems. In the PostgreSQL database management system this precompiler is called ECPG. Other embedded SQL precompilers are Pro\*Ada, Pro\*COBOL, Pro\*FORTRAN, Pro\*Pascal, and Pro\*PL/I.

## Systems that support Embedded SQL

### IBM DB2

IBM DB2 version 9 for Linux, UNIX and Windows supports embedded SQL for C, C++, Java, COBOL, FORTRAN and REXX although support for FORTRAN and REXX has been deprecated.<sup>[2]</sup>

### Oracle Corporation

Ada

**Pro\*Ada** was officially desupported by Oracle in version 7.3. Starting with Oracle8, Pro\*Ada has been replaced by SQL\*Module but appears to have not been updated since.<sup>[3]</sup> SQL\*Module is a module language that offers a different programming method from embedded SQL. SQL\*Module supports the Ada83 language standard for Ada.

C/C++

Pro\*C became Pro\*C/C++ with Oracle8. Pro\*C/C++ is currently supported as of Oracle Database 11g.

COBOL

Pro\*COBOL is currently supported as of Oracle Database 11g.

Fortran

Pro\*FORTRAN is no longer updated as of Oracle8 but Oracle will continue to issue patch releases as bugs are reported and corrected.<sup>[4]</sup>

---

## Pascal

Pro\*Pascal was not released with Oracle8.<sup>[4]</sup>

## PL/I

Pro\*PL/I was not released with Oracle8. The *Pro\*PL/I Supplement to the Oracle Precompilers Guide*, however, continues to make appearances in the Oracle Documentation Library (current as of release 11g).<sup>[4]</sup>

## PostgreSQL

### C/C++

ECPG is part of PostgreSQL since version 6.3.

### COBOL

Cobol-IT <sup>[5]</sup> is now distributing a COBOL precompiler for PostgreSQL

## Altibase

### C/C++

SESC is an embedded SQL precompiler provided by Altibase Corp. for its DBMS server.

## Data Access Corporation

With DataFlex 3.2 and Visual DataFlex you can pass SQL statements via one of the Data Access CLI connectivity kits to Microsoft SQL Server, IBM DB2 or any ODBC supporting database. The results can be retrieved and processed.

## Microsoft SQL Server

### COBOL

Cobol-IT <sup>[5]</sup> is distributing a Embedded SQL precompiler for COBOL.

## MySQL

### COBOL

Cobol-IT <sup>[5]</sup> is distributing a Embedded SQL precompiler for COBOL.

## Systems that do not support Embedded SQL

### Microsoft SQL Server

Embedded SQL for C has been deprecated as of Microsoft SQL Server 2008 although earlier versions of the product support it.<sup>[6]</sup>

### MySQL

MySQL does not support Embedded SQL.<sup>[7]</sup>

### Sybase

Embedded SQL support has been discontinued by Sybase.<sup>[8]</sup>

## See also

- Pro\*C/C++
-

## External links

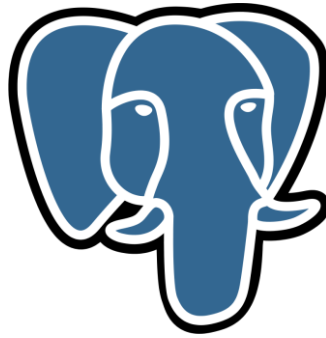
- Introduction to Pro\*C Embedded SQL <sup>[9]</sup>
- Embedded SQL with Pro\*C <sup>[10]</sup>
- SQL\*Module for Ada Programmer's Guide, Release 8.0 <sup>[11]</sup>
- ECPG, PostgreSQL 8.3 Documentation <sup>[12]</sup>

## References

- [1] "The Module Language Concept" ([http://download.oracle.com/docs/cd/B10501\\_01/appdev.920/a58231/ch1.htm#2889](http://download.oracle.com/docs/cd/B10501_01/appdev.920/a58231/ch1.htm#2889)). *SQL\*Module for Ada Programmer's Guide, Release 8.0, Chapter 1. Introduction to SQL\*Module*. Oracle Corporation. . Retrieved 2008-07-14.
- [2] "DB2 Database for Linux, UNIX and Windows" (<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.apdv.embed.doc/doc/c0007014.htm>). *Developing Embedded SQL Applications*. IBM. . Retrieved 2009-04-10.
- [3] "Ada Support in Version 8" ([http://download.oracle.com/docs/cd/B10501\\_01/server.920/a96530/migcompa.htm#1010868](http://download.oracle.com/docs/cd/B10501_01/server.920/a96530/migcompa.htm#1010868)). *Oracle9i Database Migration, Release 2 (9.2), Chapter 5. Compatibility and Interoperability*. Oracle Corporation. . Retrieved 2008-07-14.
- [4] "Language Alternatives" ([http://download.oracle.com/docs/cd/A64702\\_01/doc/server.805/a58232/ch01.htm#505](http://download.oracle.com/docs/cd/A64702_01/doc/server.805/a58232/ch01.htm#505)). *Pro\*COBOL Precompiler Programmer's Guide, Release 8.0, Chapter 1. Introduction*. Oracle Corporation. . Retrieved 2008-07-14.
- [5] <http://www.cobol-it.com>
- [6] "Deprecated Database Engine Features in SQL Server 2008" (<http://msdn.microsoft.com/en-us/library/ms143729.aspx>). *SQL Server 2008 Books Online (March 2009)*. Microsoft. . Retrieved 2009-04-10.
- [7] "An Open Pre-Compiler for Embedded SQL" (<http://dev.mysql.com/tech-resources/articles/precompiler-for-embedded-sql.html>). *MySQL DevZone Articles*. Mysql. . Retrieved 2009-04-10.
- [8] "EmbeddedSQL" (<http://www.sybase.com/products/archivedproducts/embeddedsql>). *Sybase Products*. Sybase. . Retrieved 2009-04-10.
- [9] <http://infolab.stanford.edu/%7Eullman/fcdb/oracle/or-proc.html>
- [10] <http://www.oreillynet.com/pub/a/databases/2006/12/07/embedded-sql-with-pro-c.html>
- [11] [http://tahiti.oracle.com/pls/db92/db92.show\\_toc?partno=a58231](http://tahiti.oracle.com/pls/db92/db92.show_toc?partno=a58231)
- [12] <http://www.postgresql.org/docs/8.3/interactive/ecpg.html>

# EnterpriseDB

---



<b>Type</b>	Private
<b>Industry</b>	Software & Programming
<b>Founded</b>	March 2004 <sup>[1]</sup>
<b>Headquarters</b>	Westford, Massachusetts, USA
<b>Key people</b>	Ed Boyajian, CEO
<b>Products</b>	Postgres Plus RDBMS
<b>Website</b>	<a href="http://www.enterprisedb.com">www.enterprisedb.com</a> <sup>[2]</sup>

**EnterpriseDB** is a privately-held company that provides enterprise class support for PostgreSQL through its product Postgres Plus Standard Server, a relational database management system (RDBMS) built as a superset of PostgreSQL with additional open source packages and targeted for the enterprise market. The company also offers Postgres Plus Advanced Server, which adds compatibility software to simplify application migration from other databases.

## History

Founded in March 2004, EnterpriseDB is headquartered in Westford, Massachusetts and has other offices in the United States, Europe, and Asia.

## Products

In addition to providing a mirror site to download PostgreSQL, Enterprise offers two distributions based on PostgreSQL that have additional features and enterprise-class support. These distributions are available for free download and are supported on several different platforms, including Linux, Windows, and Solaris. They include connectors for the most common enterprise programming languages and environments, including: JDBC, ODBC, .NET, ESQL / C++, Perl, Python, PHP. All three versions of the database server are available for free download.

- **PostgreSQL** - the bits developed by the Postgres community
- **Postgres Plus Standard Server** - all the features of PostgreSQL plus additional QA testing, integrated components, tuning and one-click install
- **Postgres Plus Advanced Server** - all the features of Postgres Standard Server plus Oracle compatibility, scalability features, and DBA and developer tools [it can run Oracle applications written for Oracle databases by revising the core of PostgreSQL to recognize Oracle's PL/SQL <sup>[3]</sup> <sup>[4]</sup> as well as handle data replication to and from Oracle <sup>[5]</sup> ]

EnterpriseDB also offers the following services and support options to aid in the development and deployment of Postgres-based applications:

- **Remote DBA** - administration of a Postgres-based application
- **Developer support subscription** - 8x5 support, product updates and case management
- **Deployment support subscription** - 8x5 or 24x7 support, product updates and case management
- **Getting Started with Postgres training** - half day of training designed for beginners
- **Postgres administration training** - four days of training covering all aspects of the database
- **Packages service: Health Check** - detailed architectural review
- **Packages service: Partitioning** - define a partitioning strategy for an existing installation
- **Packages service: Replication** - define a replication strategy for an existing installation
- **Packages service: High Availability** - define a high availability strategy for an existing installation

## External links

- EnterpriseDB official website <sup>[2]</sup>

## References

- [1] About EnterpriseDB Company (<http://www.enterprisedb.com/company/index.do>)
  - [2] <http://www.enterprisedb.com/>
  - [3] Babcock, Charles (August 7, 2007). "EnterpriseDB Seeks New Role As A Data Warehouse" (<http://www.intelligententerprise.com/channels/performance/showArticle.jhtml?articleID=201204243>). Intelligent Enterprise. . Retrieved 2007-09-07. "It has revised the core PostgreSQL open source system to recognize Oracle's PL/SQL version of the standard data access language and run applications designed for the Oracle database"
  - [4] LewisC (July 23, 2007). "EnterpriseDB FAQ - What does compatibility mean?" (<http://blogs.ittoolbox.com/oracle/guide/archives/enterprisedb-faq-what-does-compatibility-mean-17822>). ITtoolbox Blogs. .
  - [5] "FTD Replicates Oracle to Postgresql" (<http://www.enterprisedb.com/community/success/ftd.do>). February 2, 2009. .
-

# Epictetus Database Client

---

<b>Developer(s)</b>	Antilogic Software
<b>Stable release</b>	1.0 beta / June 17, 2009
<b>Operating system</b>	Cross-platform
<b>Platform</b>	Java
<b>Type</b>	Database administration tool
<b>License</b>	Freeware
<b>Website</b>	antilogics.com <sup>[1]</sup>

**Epictetus** is free cross-platform multi-database administration tool. Epictetus requires JVM to run. Epictetus is distributed under the Freeware License.

## Feature Summary

- Multithreading
- Autocomplete
- Primary and foreign keys highlight
- BLOB, CLOB autoload
- SQL syntax highlight
- Query history
- Schema browser
- Edit table data
- Restore last opened windows
- Autoupdate

## History

The Epictetus was developed by the Antilogic Software<sup>[2]</sup>.

## Supported databases

- Oracle Database 8i, 9i, 10g, 11g
  - Microsoft SQL Server
  - MySQL
  - InterBase/Firebird
  - PostgreSQL
  - HSQLDB
  - H2
-



## External links

- Antilogic Software - Epictetus <sup>[3]</sup>
- Interview: Konstantin Chikarev on Epictetus Database Manager <sup>[4]</sup>

## References

[1] <http://antilogics.com/>

[2] <http://www.antilogics.com/>

[3] <http://www.antilogics.com/epictetus.html>

[4] <http://java.dzone.com/news/interview-konstantin-chikarev->

## Foreign key

In the context of relational databases, a **foreign key** is a referential constraint between two tables.<sup>[1]</sup> The foreign key identifies a column or a set of columns in one (referencing) table that refers to set of columns in another (referenced) table. The columns in the referencing table must be the primary key or other candidate key in the referenced table. The values in one row of the referencing columns must occur in a single row in the referenced table. Thus, a row in the referencing table cannot contain values that don't exist in the referenced table (except potentially NULL). This way references can be made to link information together and it is an essential part of database normalization. Multiple rows in the referencing table may refer to the same row in the referenced table. Most of the time, it reflects the one (master table, or referenced table) to many (child table, or referencing table) relationship.

The referencing and referenced table may be the same table, i.e. the foreign key refers back to the same table. Such a foreign key is known in SQL:2003 as a **self-referencing** or **recursive** foreign key.

A table may have multiple foreign keys, and each foreign key can have a different referenced table. Each foreign key is enforced independently by the database system. Therefore, cascading relationships between tables can be established using foreign keys.

Improper foreign key/primary key relationships or not enforcing those relationships are often the source of many database and data modeling problems.

## Defining Foreign Keys

Foreign keys are defined in the ANSI SQL Standard, through a FOREIGN KEY constraint. The syntax to add such a constraint to an existing table is defined in SQL:2003 as shown below. Omitting the column list in the REFERENCES clause implies that the foreign key shall reference the primary key of the referenced table.

```
ALTER TABLE <table identifier>
  ADD [ CONSTRAINT <constraint identifier> ]
    FOREIGN KEY ( <column expression> {, <column expression>}... )
    REFERENCES <table identifier> [ ( <column expression> {, <column expression>}... )
    [ ON UPDATE <referential action> ]
    [ ON DELETE <referential action> ]
```

Likewise, foreign keys can be defined as part of the CREATE TABLE SQL statement.

```
CREATE TABLE table_name (
  id    INTEGER PRIMARY KEY,
  col2  CHARACTER VARYING(20),
  col3  INTEGER,
  ...
```

```
CONSTRAINT col3_fk FOREIGN KEY(col3)
REFERENCES other_table(key_col) ON DELETE CASCADE,
... )
```

If the foreign key is a single column only, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (
  id    INTEGER PRIMARY KEY,
  col2  CHARACTER VARYING(20),
  col3  INTEGER REFERENCES other_table(column_name),
  ... )
```

## Referential Actions

Because the Database Management System enforces referential constraints, it must ensure data integrity if rows in a referenced table are to be deleted (or updated). If dependent rows in referencing tables still exist, those references have to be considered. SQL:2003 specifies 5 different **referential actions** that shall take place in such occurrences:

- CASCADE
- RESTRICT
- NO ACTION
- SET NULL
- SET DEFAULT

### CASCADE

Whenever rows in the master (referenced) table are deleted, the respective rows of the child (referencing) table with a matching foreign key column will get deleted as well. This is called a cascade delete.

Example Tables: *Customer(customer\_id, cname, caddress)* and *Order(customer\_id, products, payment)*

Customer is the master table and Order is the child table, where 'customer\_id' is the foreign key in Order and represents the customer who placed the order. When a row of Customer is deleted, any Order row matching the deleted Customer's customer\_id will also be deleted.

### RESTRICT

A value cannot be updated or deleted when a row exists in a foreign key table that references the value in the referenced table.

Similarly, a row cannot be deleted as long as there is a reference to it from a foreign key table.

### NO ACTION

NO ACTION and RESTRICT are very much alike. The main difference between NO ACTION and RESTRICT is that with NO ACTION the referential integrity check is done after trying to alter the table. RESTRICT does the check before trying to execute the UPDATE or DELETE statement. Both referential actions act the same if the referential integrity check fails: the UPDATE or DELETE statement will result in an error.

In other words, when an UPDATE or DELETE statement is executed on the referenced table using the referential action NO ACTION, the DBMS verifies at the end of the statement execution that none of the referential relationships are violated. This is different from RESTRICT, which assumes at the outset that the operation will violate the constraint. Using NO ACTION, the triggers or the semantics of the statement itself may yield an end state in which no foreign key relationships are violated by the time the constraint is finally checked, thus allowing the statement to complete successfully.

## SET NULL

The foreign key values in the referencing row are set to NULL when the referenced row is updated or deleted. This is only possible if the respective columns in the referencing table are nullable. Due to the semantics of NULL, a referencing row with NULLs in the foreign key columns does not require a referenced row.

## SET DEFAULT

Similarly to SET NULL, the foreign key values in the referencing row are set to the column default when the referenced row is updated or deleted.

## Triggers

Referential actions are generally implemented as implied triggers (i.e. triggers with system-generated names, often hidden.) As such, they are subject to the same limitations as user-defined triggers, and their order of execution relative to other triggers may need to be considered; in some cases it may become necessary to replace the referential action with its equivalent user-defined trigger to ensure proper execution order, or to work around mutating-table limitations.

Another important limitation appears with transaction isolation: your changes to a row may not be able to fully cascade because the row is referenced by data your transaction cannot "see", and therefore cannot cascade onto. An example: while your transaction is attempting to renumber a customer account, a simultaneous transaction is attempting to create a new invoice for that same customer; while a CASCADE rule may fix all the invoice rows your transaction can see to keep them consistent with the renumbered customer row, it won't reach into another transaction to fix the data there; because the database cannot guarantee consistent data when the two transactions commit, one of them will be forced to rollback (often on a first-come-first-served basis.)

## Example

As a first example to illustrate foreign keys, suppose an accounts database has a table with invoices and each invoice is associated with a particular supplier. Supplier details (such as address or phone number) are kept in a separate table; each supplier is given a 'supplier number' to identify it. Each invoice record has an attribute containing the supplier number for that invoice. Then, the 'supplier number' is the primary key in the Supplier table. The foreign key in the Invoices table points to that primary key. The relational schema is the following. Primary keys are marked in bold, and foreign keys are marked in italics.

```
Supplier ( SupplierNumber, Name, Address, Type )
Invoices ( InvoiceNumber, SupplierNumber, Text )
```

The corresponding Data Definition Language statement is as follows.

```
CREATE TABLE Supplier (
  SupplierNumber INTEGER NOT NULL,
  Name           VARCHAR(20) NOT NULL,
  Address        VARCHAR(50) NOT NULL,
  Type           VARCHAR(10),
  CONSTRAINT supplier_pk PRIMARY KEY(SupplierNumber),
  CONSTRAINT number_value CHECK (SupplierNumber > 0) )

CREATE TABLE Invoices (
  InvoiceNumber INTEGER NOT NULL,
  SupplierNumber INTEGER NOT NULL,
```

```
Text          VARCHAR(4096),
CONSTRAINT invoice_pk PRIMARY KEY(InvoiceNumber),
CONSTRAINT inumber_value CHECK (InvoiceNumber > 0),
CONSTRAINT supplier_fk FOREIGN KEY(SupplierNumber)
REFERENCES Supplier(SupplierNumber)
ON UPDATE CASCADE ON DELETE RESTRICT )
```

## See also

- Alternate key
- Candidate key
- Compound key
- Superkey
- Junction table

## External links

- Cascading Referential Integrity Constraints in MS SQL Server<sup>[2]</sup>

## References

[1] "Database Basics - Foreign Keys" ([http://www.visualcase.com/kbase/database\\_basics\\_-\\_foreign\\_keys.htm](http://www.visualcase.com/kbase/database_basics_-_foreign_keys.htm)). . Retrieved 2010-03-13.

[2] [http://technet.microsoft.com/en-us/library/ms186973\(SQL.90\).aspx](http://technet.microsoft.com/en-us/library/ms186973(SQL.90).aspx)

# FSQL

**FSQL**, **Fuzzy Structured Query Language**, or **Fuzzy SQL**, means, fuzzy SQL language. It is an extension of the SQL language that allows users to write flexible conditions in their queries. using the fuzzy logic defined by Lofti A. Zadeh.

Some of the main extensions to SQL are:

- **Linguistic Labels:** If an attribute is capable of fuzzy treatment then linguistic labels can be defined on it. These labels will be preceded with the symbol \$ to distinguish them easily. There are two types of labels and they will be used in different fuzzy attribute types:
  1. Labels for attributes with an ordered underlined fuzzy domain: Every label of this type has associated a trapezoidal possibility distribution.
  2. Labels for attributes with an non ordered fuzzy domain. Here, there is a similarity relation defined between each two labels in the domain. The similarity degree is in the interval [0,1].
- **Possibility and Necessity Fuzzy Comparators:** Besides the typical comparators (=, >...), FSQL includes the fuzzy comparators in the following table. Like in SQL, fuzzy comparators may compare one column with one constant or two columns of the same type.

Comparator for Possibility	Comparator for Necessity	Significance
FEQ or F=	NFEQ or NF=	Fuzzy Equal (Possibly/Necessarily Equal)
FDIF, F!= or F<>	NFDIF, NF!= or NF<>	Fuzzy DIfferent
FGT or F>	NFGT or NF>	Fuzzy Greater Than
FGEQ or F>=	NFGEQ or NF>=	Fuzzy Greater or Equal
FLT or F<	NFLT or NF<	Fuzzy Less Than
FLEQ or F<=	NFLEQ or NF<=	Fuzzy Less or Equal
MGT or F>>	NMGT or NF>>	Much Greater Than
MLT or F<<	NMLT or NF<<	Much Less Than

Possibility comparators are more general than necessity comparators are. Then, necessity comparators retrieve less tuples and these comply with the conditions necessarily.

- **Inclusion Fuzzy Comparators:** FSQL includes two comparators for the inclusion operation: FINCL (Fuzzy Included in) and INCL (Included in).
- **Fulfilment thresholds (T):** For each simple condition a fulfilment threshold may be established (default is 1). with the format:

condition THOLD T

indicating that the condition must be satisfied with minimum degree T in [0,1] to be considered. The reserved word THOLD is optional and may be substitute for a traditional crisp comparator (=, <...), modifying the query meaning. The word THOLD is equivalent to using the crisp comparator greater or equal. The constant T maybe substituted by a qualifier (a predefined label) like \$High or \$Low. Example: Give me all persons with fair hair (in minimum degree 0.5) that they are possibly taller than label \$Tall (in minimum degree 0.8):

```
SELECT      * FROM Person
WHERE Hair FEQ $Fair THOLD 0.5 AND
          Height FGT $Tall THOLD 0.8
```

- **Function CDEG(attribute):** It shows a column with the fulfilment degree of the condition of the query, for a specific attribute, which is expressed between brackets as the argument. If logic operators appear, the calculation of this compatibility degree is carried out in the following way: We use the minimum T-norm and the maximum T-conorm, but the user may change these values by default modifying only a view (FSQL\_NOTANDOR). In this view the user can set the function to use for every logic operator (NOT, AND, OR). Obviously, that function must be implemented in the FSQL Server or may be implemented by the user himself.
- **Basic Fuzzy Constants:** Firstly, FSQL defined the syntax for some fuzzy constants. These fuzzy constants that we can use in FSQL are detailed in the following table:

Fuzzy Constant	Significance
UNKNOWN	Unknown value but the attribute is applicable
UNDEFINED	The attribute is not applicable or it is meaningless
NULL	Total ignorance: We know nothing about it
$\$[a,b,c,d]$	Fuzzy trapezium ( $a \leq b \leq c \leq d$ )
$\$label$	Linguistic Label: It may be a trapezium or a scalar (defined in FMB).
$[n,m]$	Interval "Between n and m" ( $a=b=n$ and $c=d=m$ ).
$\#n$	Fuzzy value "Approximately n" ( $b=c=n$ and $n-a=d-n=\text{margin}$ ).

- **Extended Fuzzy Constants:** In FSQL we can also use the following fuzzy constants:

Fuzzy Constant	Significance
$\$[a,b,c,d,P1/N1,...,Pn/Nn]$	Extended fuzzy trapezoid (with some points $P_i/N_i$ where all the $N_i$ are between a and b or between c and d). Values b and c are both optional. If they do not exist, then this constant is a fuzzy value without kernel.
$n \pm m$	Fuzzy value "Approximately n": triangle $n \pm m$ .
$\{P1/L1, P2/L2, ..., Pn/Ln\}$	Non-continuous possibility distribution on labels, where $P1, P2, ..., Pn$ are the possibility values and $L1, L2, ..., Ln$ are the labels.
$\{L1, L2, ..., Ln\}$	Non-continuous possibility distribution on labels, where $L1, L2, ..., Ln$ are the labels, with possibility degrees 1 for all of them: $\{1/L1, ..., 1/Ln\}$ .
$\{P1/N1, P2/N2, ..., Pn/Nn\}$	Non-continuous possibility distribution on numbers, where $P1, P2, ..., Pn$ are the possibility values and $N1, N2, ..., Nn$ are the numbers.
$\{N1, N2, ..., Nn\}$	Non-continuous possibility distribution on numbers, where $N1, N2, ..., Nn$ are the numbers, with possibility 1 for all of them: $\{1/N1, ..., 1/Nn\}$ .

- **More characteristics:** fuzzy quantifiers in queries, dynamic change of functions in logic operations, fuzzy set operators (FUNION, FINTERSECT and FMINUS), the ALTER FSQL statement, some useful functions to handle fuzzy attributes and fuzzy values (cardinality, normalization, concentration or dilatation, contrast intensification or fuzzification...), some DDL statements of FSQL and fuzzy comparators for fuzzy time in FSQL (F\_BEFORE, NF\_BEFORE, F\_AFTER, NF\_AFTER, F\_MUCH\_BEFORE, NF\_MUCH\_BEFORE, F\_OVERLAPS...).

## External links

- Web page about FSQL <sup>[1]</sup>: References, software and links.
- Galindo J., Urrutia A., Piattini M., "Fuzzy Databases: Modeling, Design and Implementation" <sup>[2]</sup>. Idea Group Publishing Hershey, USA, 2006.
- Galindo, J. (Ed.), (2008). Handbook of Research on Fuzzy Information Processing in Databases <sup>[3]</sup>. Hershey, PA, USA: Information Science Reference (<http://www.info-sci-ref.com>).

## References

- [1] <http://www.lcc.uma.es/~ppgg/FSQL/>  
 [2] <http://www.idea-group.com/books/details.asp?id=5246/>  
 [3] <http://www.lcc.uma.es/~ppgg/fuzzydb/>

# Hint (SQL)

---

In various SQL implementations, a **hint** is an addition to the SQL standard that instructs the database engine on how to execute the query. For example, a hint may tell the engine to use as little memory as possible (even if the query will run slowly), or to use or not use an index even if the query optimizer would decide otherwise.

Different database engines use different approaches in implementing hints. MySQL uses its own extension to the SQL standard, where a table name may be followed by `USE INDEX`, `FORCE INDEX` or `IGNORE INDEX` keywords<sup>[1]</sup>. Oracle implements hints by using specially crafted comments in the query that begin with a `+` symbol, thus not influencing SQL compatibility<sup>[2]</sup>.

## See also

- Query optimizer
- Query plan

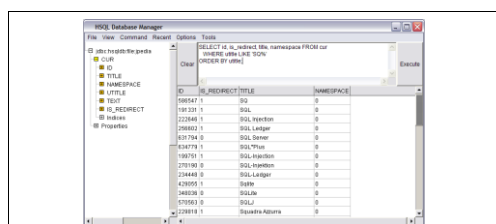
## References

[1] MySQL 5.0 Reference Manual: 12.2.8.2 Index Hint Syntax (<http://dev.mysql.com/doc/refman/5.0/en/index-hints.html>)

[2] Mike Ault: Oracle SQL Hints Tuning ([http://www.dba-oracle.com/t\\_sql\\_hints\\_tuning.htm](http://www.dba-oracle.com/t_sql_hints_tuning.htm))

---

# HSQLDB



HSQL Database Manager

<b>Stable release</b>	2.0.0 / June 7, 2010
<b>Preview release</b>	2.0.0 RC 9 / March 19, 2010
<b>Development status</b>	Active
<b>Written in</b>	Java
<b>Operating system</b>	Cross-platform
<b>Size</b>	1100 KB
<b>Type</b>	RDBMS
<b>License</b>	BSD license
<b>Website</b>	<a href="http://hsqldb.org">http://hsqldb.org</a>

**HSQLDB** (Hyper Structured Query Language Database) is a relational database management system written in Java. HSQLDB is based on the discontinued Hypersonic SQL Project.<sup>[1]</sup>

HSQLDB is available under a BSD license.

It has a JDBC driver and supports a large subset of SQL-92, SQL-99, and SQL:2003 standards.<sup>[2]</sup> It offers a fast,<sup>[3]</sup> small (around 600 kilobytes in the standard version) database engine which offers both in-memory and disk-based tables. Embedded and server modes are available.

Additionally, it includes tools such as a minimal web server, in-memory query and management tools (can be run as applets), and a number of demonstration examples. It can run on Java runtimes from version 1.1 upwards, including free Java runtimes such as Kaffe.

HSQLDB is currently being used as a database and persistence engine in many open source software projects, such as OpenOffice.org Base and the Standalone Roller Demo,<sup>[4]</sup> as well as in commercial products, such as Mathematica or InstallAnywhere (starting with version 8.0).<sup>[5]</sup>

Over 350 book titles document the use of HSQLDB for application development with frameworks such as Spring Framework or Hibernate.<sup>[6]</sup>



## Transaction support

HSQLDB version 2.0 has three transaction control modes. It supports read committed and serializable isolation levels with table level locks or with multiversion concurrency control (MVCC), or a combination of locks and MVCC. version 1.8.1 supports transaction isolation level 0 (read uncommitted) only. <sup>[7]</sup>

## Data storage

HSQLDB has two main table types used for durable read-write data storage (i.e. if transaction has been successfully committed, it is guaranteed that the data will survive system failure and will keep its integrity).

The default MEMORY type stores all data changes to the disk in the form of a SQL script. During engine start up, these commands are executed and data is reconstructed into the memory. While this behavior is not suitable for very large tables, it provides highly regarded performance benefits and is easy to debug.

Another table type is CACHED, which allows one to store gigabytes of data, at the cost of the slower performance. HSQLDB engine loads them only partially and synchronizes the data to the disk on transaction commits. However, the engine always loads all rows affected during an update into the memory. This renders very large updates impossible without splitting the work into smaller parts. <sup>[8]</sup>

Other table types allow for read-write CSV-file access (these tables can participate, for example, in queries with JOINS and simplify spreadsheet processing) and read-write non-durable in-memory data storage.

## See also

- List of relational database management systems
- Comparison of relational database management systems
- OpenOffice.org Base
- H2
- Apache Derby
- Change control

## External links

- HSQLDB at SourceForge.net <sup>[9]</sup>
- Software using HSQLDB <sup>[10]</sup>
- Books referring to HSQLDB (Google Books) <sup>[11]</sup>

## References

- [1] "The new HSQLDB" (<http://hsqldb.org/>). hsqldb.org. .
- [2] "HSQLDB SQL Syntax" (<http://hsqldb.org/doc/guide/ch09.html>). hsqldb.org. .
- [3] "HSQLDB Performance Comparison" ([http://hsqldb.org/images/imola\\_retrieve.jpg](http://hsqldb.org/images/imola_retrieve.jpg)). hsqldb.org. .
- [4] "Standalone Roller Demo" ([http://rollerweblogger.org/page/roller/20040707#try\\_roller\\_it\\_s\\_easy](http://rollerweblogger.org/page/roller/20040707#try_roller_it_s_easy)). rollerweblogger.org. .
- [5] "Software using HSQLDB" (<http://hsqldb.org/web/hsqUsing.html>). hsqldb.org. .
- [6] *Books referring to HSQLDB* (<http://books.google.com/books?q=hsqldb&btnG=Search+Books>). Google Books. .
- [7] "HSQLDB Documentation" ([http://hsqldb.org/doc/2.0/guide/sessions-chapt.html#sqlgeneral\\_trans\\_cc-sect](http://hsqldb.org/doc/2.0/guide/sessions-chapt.html#sqlgeneral_trans_cc-sect)). .
- [8] "HSQLDB Documentation" (<http://hsqldb.org/doc/guide/ch05.html#N10DED>). .
- [9] <http://hsqldb.org>
- [10] <http://hsqldb.org/web/hsqUsing.html>
- [11] <http://books.google.com/books?q=hsqldb&btnG=Search+Books>

# Log shipping

---

In Microsoft SQL Server, **log shipping** is the process of automating the backup of a database and transaction log files on a primary (production) server, and then restoring them onto a standby server<sup>[1]</sup>. Similar to replication, the primary purpose of log shipping is to increase database availability by maintaining a backup server that can replace production server quickly.

Although the actual failover mechanism in log shipping is manual, this implementation is often chosen due to its low cost in human and server resources, and ease of implementation. As comparison, SQL server clusters enable automatic failover, but at the expense of much higher storage and license costs. Compared to database replication, log shipping does not provide as much reporting capabilities, but backs up also system tables along with data tables, and locks standby server from users' modifications.<sup>[2]</sup> A replicated server can be modified (e.g. views) and therefore is not suitable for failover purposes.

## External links

- Log Shipping<sup>[3]</sup>, MS SQL Server implementation.

## References

- [1] How to Perform SQL Server Log Shipping ([http://www.sql-server-performance.com/articles/clustering/log\\_shipping\\_70\\_p1.aspx](http://www.sql-server-performance.com/articles/clustering/log_shipping_70_p1.aspx)), "What is Log Shipping". Retrieved on 2008-12-16.
- [2] Ibison, Paul. "Log Shipping vs Replication" (<http://www.sqlservercentral.com/articles/Replication/logshippingvsreplication/1399/>). *SQLServerCentral.com*. Retrieved 2009-08-07.
- [3] [http://msdn.microsoft.com/en-us/library/ms190016\(SQL.90\).aspx](http://msdn.microsoft.com/en-us/library/ms190016(SQL.90).aspx)

# MaxDB

---

<b>Stable release</b>	7.6.06.10/7.7.07.16 / March, 2010 <sup>[1]</sup>
<b>Operating system</b>	Cross-platform
<b>Type</b>	RDBMS
<b>License</b>	SAP freeware license agreement for MaxDB (closed source)
<b>Website</b>	[2]

**MaxDB** is an ANSI SQL-92 (entry level) compliant relational database management system (RDBMS) from SAP AG, which was delivered also by MySQL AB from 2003 to 2007. MaxDB is targeted for large SAP environments e.g. mySAP Business Suite and other applications that require enterprise-level database functionality. It is able to run terabyte-range data in continuous operation.

## History

The database development started in 1977 as a research project at the Technical University of Berlin. In the early 1980s it became a database product that subsequently was owned by Nixdorf Computer, Siemens-Nixdorf, Software AG and today by SAP AG. Along this line it has been named VDN, Reflex, Supra 2, DDB/4, Entire SQL-DB-Server and Adabas D. In 1997 SAP took over the software from Software AG and renamed it to SAP DB. Since October 2000 SAP DB sources additionally were released as open source under the GNU General Public License. In 2003 SAP AG and MySQL AB joined a partnership and re-branded the database system to MaxDB.

In October 2007 this reselling was terminated and sales and support of the database reverted back to SAP <sup>[3]</sup>. SAP AG is now managing MaxDB development, distribution, and support. Source code of MaxDB is no longer available under the GNU General Public License. SAP also stated that "Further commercial support concepts to cover mission critical use requirements outside of SAP scenarios are currently subject to discussion."<sup>[4]</sup>

Version 7.5 of MaxDB is a direct advancement of the SAP DB 7.4 code base. Therefore, the MaxDB software version 7.5 can be used as a direct upgrade of previous SAP DB versions starting 7.2.04 and higher.

MaxDB is subjected to SAP AG's complete quality assurance process before it is shipped with SAP solutions or provided as a download on the 'SAP Network' <sup>[5]</sup>.

## Features

MaxDB is delivered with a set of administration and development tools. Most tools are GUI based and have CLI (Command Line Interface) based counterparts. It offers bindings for JDBC; ODBC; SQLDBC (native C/C++ interface); Precompiler; PHP; Perl; Python; WebDAV; OLE DB, ADO, DAO, RDO and .NET via ODBC; Delphi and Tcl via Third Party Programming Interfaces. MaxDB is Cross-platform, offering releases for HP-UX, IBM AIX, Linux, Solaris, Microsoft Windows 2000, Microsoft Windows Server 2003, and Microsoft Windows XP. SAP users should check the details of the platform availability on the SAP product pages for the product that will be used together with MaxDB.

## Distinguishing features

MaxDB offers built-in hot backup, does not need any online reorganizations and claims to be SQL 92 Entry-Level compatible. One current development goal is "zero administration" and by this "low TCO". It is also fair to expect good online transaction processing (OLTP) performance combined with relatively low hardware requirements.

High availability mode and small fault record is another historically observed feature of MaxDB, which holds some appraisal from the open source software and db communities.

## Future releases

The next release will have the name MaxDB 7.7.00. One possible future feature for 7.7.00 is the use of Multiversion Concurrency Control (MVCC) instead of the current lock based implementation.

## Licensing

MaxDB was licensed under the GNU GPL from versions 7.2 through 7.6. Programming interfaces were licensed under the GPL with exceptions for projects released under other Open Source licenses.

SAP DB 7.3 and 7.4 were licensed as GPL with LGPL drivers. MaxDB 7.5 was offered under dual licensing, i.e. licensed as GPL with GPL drivers or a commercial license.

From version 7.5 through version 7.6 onwards distribution of MaxDB (previously SAP DB) to the open source community was provided by MySQL AB, the same company that develops the open-source software database, MySQL. Development was done by SAP AG, MySQL AB and the open-source software community.

In October 2007, SAP assumed full sales and commercial support for MaxDB. MaxDB 7.6 is now closed source, available free-of-charge (without support, and with usage restrictions) for use with non-SAP applications. Commercial support models for using MaxDB outside of SAP environments are under consideration.

## See also

- List of relational database management systems
- Comparison of relational database management systems
- Comparison of database tools

## External links

- SAP MaxDB - The SAP Database System <sup>[6]</sup> on SAP Developer Network (SDN)
- MaxDB Wiki <sup>[7]</sup> on SAP Developer Network (SDN)

## References

- [1] Kristen, Andrea (24 March 2010). "Updates for SAP MaxDB 7.6 and 7.7" (<http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/18319>). . Retrieved 6 May 2010.
- [2] <https://www.sdn.sap.com/irj/sdn/maxdb>
- [3] MySQL AB :: MySQL AB to Optimize its Open Source Database for SAP NetWeaver ([http://www.mysql.com/news-and-events/press-release/release\\_2007\\_40.html](http://www.mysql.com/news-and-events/press-release/release_2007_40.html))
- [4] *MaxDB back under the SAP roof!* (<https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/7514>)
- [5] <http://sdn.sap.com/irj/sdn/maxdb>
- [6] <http://www.sdn.sap.com/irj/sdn/maxdb>
- [7] <http://www.sdn.sap.com/irj/scn/wiki?path=/display/MaxDB/Main>

# Meta-SQL

---

## 'Meta-SQL' Use (with reference to PeopleSoft)

Meta-SQL expands to platform-specific SQL substrings, causes another function to be called, or substitutes a value. Meta-SQL constructs are used in functions that pass SQL strings, such as the following:

- SQLExec.
- Scroll buffer functions (ScrollSelect and its relatives).
- PeopleSoft Application Designer dynamic and SQL views.
- Some Rowset class methods (Select, SelectNew, Fill, and so on.).
- The SQL class.
- PeopleSoft Application Engine programs.
- Some Record class methods (Insert, Update, and so on.).
  - COBOL functions.

## Meta-SQL Element

Types There are three types of meta-SQL elements:

- Construct

Constructs are a direct substitution of a value, and help to build or modify a SQL statement. Examples include %Bind, %InsertSelect, and %List.

- Function

Functions perform actions or cause another function to be called. Examples include %ClearCursor, %Execute, and %ExecuteEdits.

- Meta-variable.

Meta-variables enable substitution of text within SQL statements. Examples include %AsOfDate, %Comma, and %JobInstance.

---

Meta-SQL Placement Considerations: Not all meta-SQL can be used by all programs. Some meta-SQL can be used only in Application Engine programs. Other meta-SQL can only be used as part of a SQL statement in a SQL or dynamic view. The following table lists available meta-SQL elements and where each element can be used.

If a meta-SQL construct, function, or meta-variable is supported in PeopleCode, it is supported in all types of PeopleCode programs; that is, in Application Engine PeopleCode programs (actions), component interface PeopleCode programs, and so on.

Note. Even if a meta-SQL element is used in PeopleCode, you cannot use meta-SQL like a built-in function. You can use meta-SQL in the SQLExec function, the Select method, the Fill method, and so on. Note. Meta-SQL is not available in SQR

---

# MySQL Workbench

<b>Developer(s)</b>	Oracle Corporation
<b>Stable release</b>	5.1.18 / September 3, 2009
<b>Written in</b>	C# / C++ / Objective-C (Depending on platform)
<b>Operating system</b>	Cross-platform
<b>License</b>	GNU General Public License or proprietary EULA
<b>Website</b>	<a href="http://wb.mysql.com/">http://wb.mysql.com/</a>

**MySQL Workbench** is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single, seamless environment for the MySQL database system. It is the successor to DBDesigner 4 from fabFORCE.net, and replaces the previous package of software, MySQL GUI Tools Bundle.

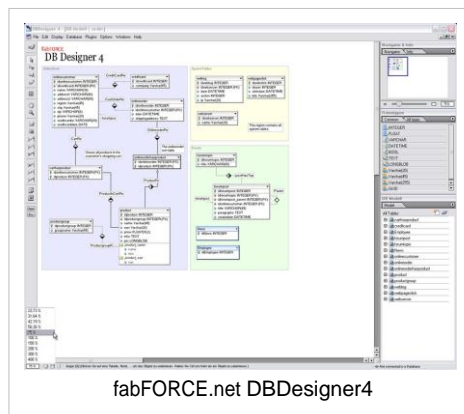
## History

### fabFORCE.net DBDesigner4

**DBDesigner4** is an open source visual database design and querying tool for the MySQL database released under the GPL.<sup>[1]</sup> It was written in 2002/2003 by the Austrian programmer Michael G. Zinner for his fabFORCE.net platform using Delphi 7 / Kylix 3.<sup>[2] [3]</sup>

While being a physical-modeling only tool DBDesigner4 offers a comprehensive feature set including reverse engineering of MySQL databases, model-to-database synchronization, model poster printing, basic version control of schema models and a SQL query builder.<sup>[4]</sup> It is available for MS Windows and Linux.<sup>[5]</sup>

Zinner was approached by representatives from MySQL AB and joined the company in late 2003 to take over the development of graphical user interface (GUI) tools for MySQL which lead to the creation of the MySQL GUI Tools Bundle.<sup>[6]</sup>

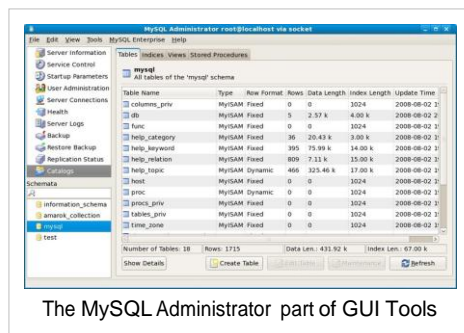


fabFORCE.net DBDesigner4

### MySQL GUI Tools Bundle

The **MySQL GUI Tools Bundle** is a cross-platform open source suite of desktop applications for the administration of MySQL database servers, and for building and manipulating the data within MySQL databases. It was developed by MySQL AB and later by Sun Microsystems and released under the GPL. Development on the GUI Tools bundle has stopped, and is now only preserved under the Download Archives of the MySQL site.<sup>[7]</sup>

The GUI Tools bundle has been superseded by MySQL Workbench, and reached its End-of-Life with the beta releases of MySQL



The MySQL Administrator part of GUI Tools

Workbench 5.2. However, the MySQL Support team will continue to provide assistance for the bundle until June 30, 2010.<sup>[8]</sup> Future releases of MySQL Workbench will add a migration plugin, providing features comparable to the

MySQL Migration Toolkit component within the MySQL GUI Tools bundle.

## Releases

The first preview version of MySQL Workbench was released in September 2005,<sup>[9]</sup> and was not included in the MySQL GUI Tools Bundle. Development was started again in 2007 and MySQL Workbench was set to become the MySQL GUI flagship product.<sup>[10]</sup>

Version numbering was started 5.0 to emphasize the fact that MySQL Workbench was developed as the successor to DBDesigner4.<sup>[11]</sup>

## MySQL Workbench 5.0 and 5.1

MySQL Workbench 5.0 and 5.1 are specialized visual database design tools for the MySQL database. While MySQL Workbench 5.0 was a MS Windows only product cross-platform support was added to MySQL Workbench 5.1 and later.<sup>[12]</sup> <sup>[13]</sup>

## MySQL Workbench 5.2

Starting with MySQL Workbench 5.2 the application has evolved to a general database GUI application. Apart from physical database modeling it features a SQL Editor and a database server administration interface, replacing the old MySQL GUI Tools Bundle.

## Features

Prominent features of MySQL Workbench 5.2 are:

- General
    - Database Connection & Instance Management
    - Wizard driven action items
    - Fully scriptable with Python and Lua
    - Support for custom plugins
  - SQL Editor
    - Schema Object browsing
    - SQL Syntax Highlighter and Statement Parser
    - Multiple-, editable Result Sets
    - SQL Snippets Collections
    - SSH Connection Tunneling
    - Unicode Support
  - Data Modeling
    - ER Diagramming
    - Drag'n'Drop visual modeling
    - Reverse Engineering from SQL Scripts and Live Database
    - Forward Engineering to SQL Scripts and Live Database
    - Schema Synchronization
    - Printing of Models
    - Import from fabFORCE.net DBDesigner4
  - Database Administration
    - Start & Stop of Database Instances
    - Instance Configuration
-

- Database Account Management
- Instance Variables Browsing
- Log File Browsing
- Data Dump Export/Import

## Licensing and editions

MySQL Workbench is the first MySQL family of products that offer two different editions - an open source and a commercial edition.<sup>[14]</sup> The "Community Edition" is a full featured product that is not crippled in any way. Being the foundation for all other editions it will benefit from all future development efforts. The commercial "Standard Edition" extends the Community Edition with a series of modules and plugins.<sup>[15]</sup>

As this business decision was announced soon after the takeover of MySQL by Sun Microsystems this caused speculation about the future licensing of the MySQL database in the press.<sup>[16] [17]</sup>

## Community reception and reviews

Since its introduction MySQL Workbench has become popular within the MySQL community. Since 2010 it is now the 2nd most downloaded product from the MySQL website with more than 250.000 downloads a month.<sup>[18]</sup> Before that it was voted Database Tool of the Year 2009 on Developer.com.<sup>[19]</sup>

MySQL Workbench has been reviewed by the open source community and print magazines.<sup>[20] [21] [22] [23]</sup> While some limitation of the Beta releases have been noted MySQL Workbench usage is generally recommended.

## See also

- HeidiSQL
- Navicat
- SQLyog

## External links

- MySQL Workbench Product home page<sup>[24]</sup>, MySQL.com. Retrieved 2010-03-26.
- MySQL Workbench Developer Central webpage<sup>[25]</sup>, wb.MySQL.com. Retrieved 2010-03-26.

## References

- [1] DBDesigner4 Webpage (<http://www.fabforce.net/dbdesigner4/>), fabFORCE.net. Retrieved 2010-03-26.
- [2] fabFORCE.net About Page (<http://www.fabforce.net/about.php>), fabFORCE.net. Retrieved 2010-03-26.
- [3] DBDesigner4 Source Code Download (<http://www.fabforce.net/downloadfile.php?idownloadfile=7>), fabFORCE.net. Retrieved 2010-03-26.
- [4] DBDesigner4 Feature List (<http://www.fabforce.net/dbdesigner4/features.php>), fabFORCE.net. Retrieved 2010-03-26.
- [5] DBDesigner4 Download Page (<http://www.fabforce.net/downloads.php>), fabFORCE.net. Retrieved 2010-03-26.
- [6] Arjen Lentz, "Interview with Michael G. Zinner" (<http://dev.mysql.com/tech-resources/interviews/mike-zinner.html>), MySQL.com. Retrieved 2010-03-26.
- [7] MySQL GUI Tools Bundle: Archived Downloads (<http://dev.mysql.com/downloads/gui-tools/5.0.html>), MySQL.com, Retrieved 2010-03-26.
- [8] MySQL Product Support EOL Announcements (<http://dev.mysql.com/support/eol-notice.html>), MySQL.com, Retrieved 2010-03-26.
- [9] MySQL GUI Bundle announcement (without MySQL Workbench) (<http://forums.mysql.com/read.php?3,146587,146587#msg-146587>), MySQL.com Forum Archive, Retrieved 2010-03-26.
- [10] Workbench Schedule Announcement (<http://forums.mysql.com/read.php?113,142329,142329#msg-142329>), MySQL.com Forum Archive, Retrieved 2010-03-26.
- [11] MySQL Workbench FAQ - General ([http://wb.mysql.com/?page\\_id=28](http://wb.mysql.com/?page_id=28)), MySQL Workbench Blog, Retrieved 2010-03-26.
- [12] Michael G. Zinner, "Why Release on Windows First" (<http://wb.mysql.com/?p=19>), MySQL Workbench Blog, Retrieved 2010-03-26.
- [13] MySQL Workbench Releases ([http://wb.mysql.com/?page\\_id=49](http://wb.mysql.com/?page_id=49)), MySQL Workbench Blog, Retrieved 2010-03-26.



- [14] MySQL Workbench Editions ([http://dev.mysql.com/workbench/?page\\_id=11](http://dev.mysql.com/workbench/?page_id=11)), MySQL Workbench Blog. Retrieved 2010-03-26.
- [15] Michael G. Zinner, "Beta testers wanted!" (<http://lists.mysql.com/gui-tools/2022>), MySQL Mailing List Archives, 2007-11-19. Retrieved 2010-03-26.
- [16] Sun Introduces MySQL Workbench ([http://www.oreillynet.com/conferences/blog/2008/04/news\\_release\\_sun\\_introduces\\_my.html](http://www.oreillynet.com/conferences/blog/2008/04/news_release_sun_introduces_my.html)), O'Reilly Conference News, Retrieved 2010-03-26.
- [17] Sean Michael Kerner, "MySQL Not Going Closed Source?" (<http://www.internetnews.com/dev-news/article.php/3741616/MySQL+Not+Going+Closed+Source.htm>), internetnews.com. Retrieved 2010-03-26.
- [18] MySQL Workbench Contribute Page ([http://wb.mysql.com/?page\\_id=17](http://wb.mysql.com/?page_id=17)), MySQL Workbench Blog. Retrieved 2010-03-26.
- [19] Winners - Product of the Year 2009 (<http://www.developer.com/java/other/article.php/3795991/Winners-of-the-Developercom-Product-of-the-Year-2009-Are-Announced.htm>), Developer.com, 2009-01-14. Retrieved 2010-03-26.
- [20] Jack Wallen, "Get to Know MySQL Workbench" (<http://www.linux.com/learn/tutorials/293621-get-to-know-mysql-workbench>), Linux.com, 2010-03-16. Retrieved 2010-03-26.
- [21] Konstantin Mirin, "MySQL Workbench – The Database Modeling Tool for MySQL" (<http://programmersnotes.info/2009/03/01/mysql-workbench-the-database-modeling-tool-for-mysql/>), programmersnotes.info, 2009-03-01. Retrieved 2010-03-26.
- [22] Cal Evans, "MySQL Workbench – A Superficial Review" (<http://blog.calevans.com/2009/07/05/mysql-workbench-a-superficial-review/>), blog.calevans.com, 2009-07-05. Retrieved 2010-03-26.
- [23] MySQL Workbench 5.1: Too many tables and too many complex relationships? Visualize your MySQL database with MySQL Workbench. (<http://www.linuxpromagazine.com/Resources/Current-Issue>), Linux Magazine, 2001-03-19. Retrieved 2010-03-26.
- [24] <http://mysql.com/products/workbench/>
- [25] <http://wb.mysql.com/>

## NVL

---

In Oracle/PLSQL, the **NVL** function lets you substitute a value when a null value is encountered.

The syntax for the NVL function is:

```
NVL( string1, replace_with_if_null )
```

string1 is the string to test for a null value. replace\_with is the value returned if string1 is null.

Both parameters of the function have to be of the same data type. You cannot use this function to replace a null integer by a string unless you call the TO\_CHAR function on that value:

```
NVL(TO_CHAR(numeric_column), 'some string')
```

### See also

- COALESCE, the standard equivalent

# Navicat

---

<b>Developer(s)</b>	PremiumSoft
<b>Operating system</b>	Cross-platform
<b>Available in</b>	Multilingual (8)
<b>Type</b>	SQL Database Management and Development
<b>License</b>	Shareware
<b>Website</b>	<a href="http://www.navicat.com">http://www.navicat.com</a>

PremiumSoft **Navicat** is a series of graphical database management and development software for MySQL, Oracle, SQLite and PostgreSQL. It has an Explorer-like interface and supports multiple database connections for local and remote databases.

Navicat allows database administrators and developers to manage, develop and monitor MySQL, Oracle, PostgreSQL and SQLite databases. Navicat works in Microsoft Windows, Macintosh and Linux platforms.

Some of its features include:

- Visual query builder
- SSH and HTTP tunneling
- Data and structure synchronization
- Import and export and backup of data
- Report builder
- Task scheduling and wizards tool

There are differences in the features available across operating systems.<sup>[1]</sup>

## History

Navicat for MySQL is the first product of PremiumSoft which is released in 2002. Navicat for PostgreSQL was released in 2005, whereas Navicat for Oracle is released in August 2008 and Navicat for SQLite is released in April 2010.

In June 2009, Navicat Premium version 8 is released. Navicat Premium combines all Navicat versions into one version and can connect to different database types including MySQL, Oracle, and PostgreSQL simultaneously, allowing users to do data migration between cross databases.

In April 2010, PremiumSoft has released version 9 of Navicat Premium, in this new version, it added the connectivity of SQLite database to Navicat Premium, allowing Navicat Premium to connect to MySQL, Oracle, Postgresql and SQLite in a single application.

## Free version

Navicat Lite is a free version of the software made available in October 2007 for non-commercial use only. Previously, Navicat was an exclusively commercial software package since its release in 2002.

The Lite editions of Navicat lack several features found in the Standard editions, including form view, record filtering, visual query building, and options for import, export and backup of data.<sup>[1]</sup>

## References

[1] Navicat Feature Matrix, for MySQL (<http://mysql.navicat.com/feature.html>), PostgreSQL (<http://pgsql.navicat.com/feature.html>), Oracle (<http://oracle.navicat.com/feature.html>). Retrieved on 14 December 2008.

- Connecting MySQL GUI Client to a Remote MySQL Server (<http://java.dzone.com/articles/connecting-mysql-gui-client-a->) (by kevin from dzone.com)
- Navicat MySQL GUI Review ([http://www.lasplash.com/TechTalk/navicat\\_mysql\\_gui\\_review.php](http://www.lasplash.com/TechTalk/navicat_mysql_gui_review.php)) (by lasplash.com)
- Navicat Review (<http://www.onlamp.com/pub/a/onlamp/2004/12/22/navicat.html>) (by O'Reilly OnLamp.com)
- MySQL Database Management Tools For Windows (<http://www.techmixer.com/navicat-mysql-mysql-database-management-tools-for-windows>) (by techmixer.com)
- The Era of Open Source: Migrate your Data from MS SQL to MySQL (<http://databasejournal.com/features/mysql/article.php/3405841>) (by Allen from databasejournal.com)
- How to Import Excel, Access or XML Data into MySQL Using Navicat (<http://www.phpbuilder.com/columns/kenlin20031229.php3>)

## External links

- Navicat (<http://www.navicat.com/>)

# Nested SQL

---

A **nested table** is a table that is embedded within another table.

Individual elements can be inserted, updated, and deleted in a nested table. Since individual elements can be modified in a nested table, they are more flexible than an array because elements in an array can only be modified as a whole, not individually. A nested table doesn't have a maximum size, and an arbitrary number of elements can be stored in it.

## See also

- Query optimizer

## External links

- Oracle Collections: A Definition in Plain English Part 2<sup>[1]</sup>
- Oracle Database 10g: SQL (Oracle Press)<sup>[2]</sup>

## References

[1] <http://blogs.ittoolbox.com/oracle/guide/archives/oracle-collections-a-definition-in-plain-english-part-2-6003>

[2] <http://www.oraclepressonline.com/toc.jsp?bookid=8172&prevbkid=&nextbkid=>

# Object-oriented SQL

---

A functional language, a superset of SQL, used in Hewlett-Packard's OpenODB database system.

SQL1999, formerly known as SQL3, is an Object-Oriented SQL.

# PL/pgSQL

---

**PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language)** is a procedural language supported by the PostgreSQL RDBMS. It closely resembles Oracle's PL/SQL language.

PL/pgSQL, as a fully featured programming language, allows much more procedural control than SQL, including the ability to use loops and other control structures. Functions created in the PL/pgSQL language can be called from an SQL statement, or as the action that a trigger performs.

PL/pgSQL was created to be able to perform more complex operations and computations than SQL, while being easy to use, and is able to be defined as trusted by the server.<sup>[1]</sup>

Lot of examples of PLpgSQL using are in PL/pgSQL tutorial <sup>[2]</sup>.

PL/pgSQL is the only "PL" language installed by default for PostgreSQL, but many others are available, including PL/Java <sup>[3]</sup>, PL/Perl <sup>[4]</sup>, plPHP <sup>[5]</sup>, PL/Python <sup>[6]</sup>, PL/R <sup>[7]</sup>, PL/Ruby <sup>[8]</sup>, PL/sh <sup>[9]</sup>, and PL/Tcl <sup>[10]</sup>.

## References

- [1] "PL/pgSQL - SQL Procedural Language" (<http://www.postgresql.org/docs/current/static/plpgsql.html>). . Retrieved 2007-11-15.
  - [2] [http://www.pgsql.cz/index.php/PL/pgSQL\\_%28en%29](http://www.pgsql.cz/index.php/PL/pgSQL_%28en%29)
  - [3] <http://gborg.postgresql.org/project/pljava/projdisplay.php>
  - [4] <http://www.postgresql.org/docs/current/interactive/plperl.html>
  - [5] <http://www.commandprompt.com/community/plphp/>
  - [6] <http://www.postgresql.org/docs/current/interactive/plpython.html>
  - [7] <http://www.joeconway.com/plr/>
  - [8] <http://raa.ruby-lang.org/project/pl-ruby>
  - [9] <http://plsh.projects.postgresql.org/>
  - [10] <http://www.postgresql.org/docs/current/interactive/pltcl.html>
-

# PL/SQL

---

**PL/SQL** (Procedural Language/Structured Query Language) is Oracle Corporation's procedural extension language for SQL and the Oracle relational database. PL/SQL's general syntax resembles that of Ada.

**PL/SQL** is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java.

PL/SQL is available in Oracle Database (since version 7), TimesTen in-memory database (since version 11.2.1),<sup>[1]</sup> IBM DB2 (since version 9.7)<sup>[2]</sup>.

## Introduction

PL/SQL supports variables, conditions, loops and exceptions. Arrays are also supported, though in a somewhat unusual way, involving the use of PL/SQL collections. PL/SQL collections are a slightly advanced topic.

Implementations from version 8 of Oracle Database onwards have included features associated with object-orientation.

PL/SQL program units (essentially code containers) can be compiled into the Oracle database. Programmers can thus embed PL/SQL units of functionality into the database directly. They also can write scripts containing PL/SQL program units that can be read into the database using the Oracle SQL\*Plus tool.

Once the program units have been stored into the database, they become available for execution at a later time.

While programmers can readily embed Data Manipulation Language (DML) statements directly into their PL/SQL code using straight forward SQL statements, Data Definition Language (DDL) requires more complex "Dynamic SQL" statements to be written in the PL/SQL code. However, DML statements underpin the majority of PL/SQL code in typical software applications.

In the case of PL/SQL dynamic SQL, early versions of the Oracle Database required the use of a complicated Oracle DBMS\_SQL package library. More recent versions have however introduced a simpler "Native Dynamic SQL", along with an associated EXECUTE IMMEDIATE syntax.

Oracle Corporation customarily extends package functionality with each successive release of the Oracle Database.

## PL/SQL program units

### Anonymous Blocks

Anonymous blocks<sup>[3]</sup> form the basis of the simplest PL/SQL code, and have the following structure:

```
<<label>>
DECLARE
    Type / item / function / procedure declarations
BEGIN
    Statements
EXCEPTION
    Exception handlers
END label;
```

The <<label>> and the DECLARE and EXCEPTION sections are optional.

Exceptions, errors which arise during the execution of the code, have one of two types:

1. Predefined exceptions
2. User-defined exceptions.

User-defined exceptions are always raised explicitly by the programmers, using the `RAISE` or `RAISE_APPLICATION_ERROR` commands, in any situation where they have determined that it is impossible for normal execution to continue. `RAISE` command has the syntax:

```
RAISE <exception name>;
```

Oracle Corporation has pre-defined several exceptions like `NO_DATA_FOUND`, `TOO_MANY_ROWS`, *etc.* Each exception has a SQL Error Number and SQL Error Message associated with it. Programmers can access these by using the `SQLCODE` and `SQLERRM` functions.

The `DECLARE` section defines and (optionally) initialises variables. If not initialised specifically, they default to `NULL`.

For example:

```
declare
  number1 number(2);
  number2 number(2) := 17;           -- value default
  text1   varchar2(12) := 'Hello world';
  text2   date       := SYSDATE;     -- current date and time
begin
  SELECT street_number
    INTO number1
   FROM address
  WHERE name = 'INU';
end;
```

The symbol `:=` functions as an assignment operator to store a value in a variable.

The major datatypes in PL/SQL include `NUMBER`, `INTEGER`, `CHAR`, `VARCHAR2`, `DATE`, `TIMESTAMP`, `TEXT` *etc.*

## Functions

Functions in PL/SQL are a collection of SQL and PL/SQL statements that perform a task and should return a value to the calling environment. User defined functions are used to supplement the many hundreds of functions built in by Oracle.

There are two different types of functions in PL/SQL. The traditional function is written in the form:

```
CREATE OR REPLACE FUNCTION <function_name> [(input/output variable declarations)] RETURN return_type
[AUTHID <CURRENT USER | DEFINER>] <IS|AS>
  [declaration block]
BEGIN
  <PL/SQL block with return statement>
  RETURN <return_value>;
[Exception
  exception block]
  RETURN <return_value>;
END;
```

Pipelined Table Functions are written in the form:

```
CREATE OR REPLACE FUNCTION <function_name> [(input/output variable declarations)] RETURN return_type
[AUTHID <CURRENT USER | DEFINER>] [<AGGREGATE | PIPELINED>] <IS|USING>
```

```

    [declaration block]
BEGIN
    <PL/SQL block with return statement>
    PIPE ROW <return type>;
    RETURN;
[Exception
    exception block]
    PIPE ROW <return type>;
    RETURN;
END;

```

There are three types of parameter: IN, OUT and IN OUT. An IN parameter is used as input only. An IN parameter is passed by reference and thus cannot be changed by the called program. An OUT parameter is initially NULL. The program assigns the parameter a value and that value is returned to the calling program. An IN OUT parameter may or may not have an initial value. That initial value may or may not be modified by the called program. Any changes made to the parameter are returned to the calling program by default by copying but, with the NOCOPY hint may be passed by reference.

## Procedures

Procedures are similar to Functions, in that they can be executed to perform work. The primary difference is that procedures cannot be used in a SQL statement and although they can have multiple out parameters they do not "RETURN" a value.

Procedures are traditionally the workhorse of the coding world and functions are traditionally the smaller, more specific pieces of code. PL/SQL maintains many of the distinctions between functions and procedures found in many general-purpose programming languages, but in addition, functions can be called from SQL, while procedures cannot.

## Packages

Packages are groups of conceptually linked Functions, Procedures, Variable, PL/SQL table and record TYPE statements, Constants & Cursors etc. The use of packages promotes re-use of code. Packages usually have two parts, a specification and a body, although sometimes the body is unnecessary. The specification (spec for short) is the interface to your applications; it declares the types, variables, constants, exceptions, cursors, and subprograms available for use. The body fully defines cursors and subprograms, and so implements the spec. They are stored permanently in USER\_SOURCE system table. Advantages 1. Modular approach, Encapsulation/hiding of business logic, security, performance improvement, re-usability. They support OOPS features like function over loading, encapsulation. 2. Using we can declare session level (scoped) variables. Since variable declared in pack spec has session scope.

## Variables

### Numeric variables

```
variable_name number(P[,S]) := value;
```

To define a numeric variable, the programmer appends the variable type **NUMBER** to the name definition. To specify the (optional) precision(P) and the (optional) scale (S), one can further append these in round brackets, separated by a comma. ("Precision" in this context refers to the number of digits which the variable can hold, "scale" refers to the number of digits which can follow the decimal point.)



A selection of other datatypes for numeric variables would include: `binary_float`, `binary_double`, `dec`, `decimal`, `double precision`, `float`, `integer`, `int`, `numeric`, `real`, `smallint`, `binary_integer`

## Character variables

```
variable_name varchar2(L) := 'Text';
```

To define a character variable, the programmer normally appends the variable type `VARCHAR2` to the name definition. There follows in brackets the maximum number of characters which the variable can store.

Other datatypes for character variables include: `varchar`, `char`, `long`, `raw`, `long raw`, `nchar`, `nchar2`, `clob`, `blob`, `bfile`

## Date variables

```
variable_name date := '01-Jan-2005';
```

Oracle provides a number of data types that can store dates (`DATE`, `DATETIME`, `TIMESTAMP etc.), however DATE is most commonly used.`

Programmers define date variables by appending the datatype code "`DATE`" to a variable name. The `TO_DATE` function can be used to convert strings to date values. The function converts the first quoted string into a date, using as a definition the second quoted string, for example:

```
to_date('31-12-2004','dd-mm-yyyy')
```

or

```
to_date ('31-Dec-2004','dd-mon-yyyy', 'NLS_DATE_LANGUAGE = American')
```

To convert the dates to strings one uses the function `TO_CHAR` (`date_string`, `format_string`).

PL/SQL also supports the use of ANSI date and interval literals.<sup>[4]</sup> The following clause gives an 18-month range:

```
WHERE dateField BETWEEN DATE '2004-12-31' - INTERVAL '1-6' YEAR TO MONTH
      AND DATE '2004-12-31'
```

## Datatypes for specific columns

Variable\_name **Table\_name**.Column\_name%*type*;

This syntax defines a variable of the type of the referenced column on the referenced tables.

Programmers specify user-defined datatypes with the syntax:

```
type data_type is record (field_1 type_1 :=xyz, field_2 type_2 :=xyz, ..., field_n type_n :=xyz);
```

For example:

```
declare
    type t_address is record (
        name address.name%type,
        street address.street%type,
        street_number address.street_number%type,
        postcode address.postcode%type);
    v_address t_address;
begin
    select name, street, street_number, postcode into v_address from address where rownum = 1;
end;
```

This sample program defines its own datatype, called *t\_address*, which contains the fields *name*, *street*, *street\_number* and *postcode*.

So according to the example, we are able to copy the data from the database to the fields in the program.

Using this datatype the programmer has defined a variable called *v\_address* and loaded it with data from the ADDRESS table.

Programmers can address individual attributes in such a structure by means of the dot-notation, thus:  
*"v\_address.street := 'High Street';"*

## Conditional Statements

The following code segment shows the IF-THEN-ELSIF construct. The ELSIF and ELSE parts are optional so it is possible to create simpler IF-THEN or, IF-THEN-ELSE constructs.

```
IF x = 1 THEN
    sequence_of_statements_1;
ELSIF x = 2 THEN
    sequence_of_statements_2;
ELSIF x = 3 THEN
    sequence_of_statements_3;
ELSIF x = 4 THEN
    sequence_of_statements_4;
ELSIF x = 5 THEN
    sequence_of_statements_5;
ELSE
    sequence_of_statements_N;
END IF;
```

The CASE statement simplifies some large IF-THEN-ELSE structures.

```
CASE
    WHEN x = 1 THEN sequence_of_statements_1;
    WHEN x = 2 THEN sequence_of_statements_2;
    WHEN x = 3 THEN sequence_of_statements_3;
    WHEN x = 4 THEN sequence_of_statements_4;
    WHEN x = 5 THEN sequence_of_statements_5;
    ELSE sequence_of_statements_N;
END CASE;
```

CASE statement can be used with predefined selector:

```
CASE x
    WHEN 1 THEN sequence_of_statements_1;
    WHEN 2 THEN sequence_of_statements_2;
    WHEN 3 THEN sequence_of_statements_3;
    WHEN 4 THEN sequence_of_statements_4;
    WHEN 5 THEN sequence_of_statements_5;
    ELSE sequence_of_statements_N;
END CASE;
```

## Array handling

PL/SQL refers to arrays as "collections". The language offers three types of collections:

1. Index-by tables (associative arrays)
2. Nested tables
3. Varrays (variable-size arrays)

Programmers must specify an upper limit for varrays, but need not for index-by tables or for nested tables. The language includes several collection methods used to manipulate collection elements: for example FIRST, LAST, NEXT, PRIOR, EXTEND, TRIM, DELETE, etc. Index-by tables can be used to simulate associative arrays, as in this example of a memo function for Ackermann's function in PL/SQL.

## Looping

As a procedural language by definition, PL/SQL provides several iteration constructs, including basic LOOP statements, WHILE loops, FOR loops, and Cursor FOR loops.

### LOOP statements

Syntax <sup>[5]</sup>:

```
<<parent_loop>>
LOOP
    statements

    <<child_loop>>
    loop
        statements
        exit parent_loop when <condition>; -- Terminates both loops
        exit when <condition>; -- Returns control to parent_loop
    end loop;

    exit when <condition>;
END LOOP parent_loop;
```

Loops can be terminated by using the EXIT keyword, or by raising an exception.

### Cursor FOR loops

```
FOR RecordIndex IN (SELECT person_code FROM people_table)
LOOP
    DBMS_OUTPUT.PUT_LINE(RecordIndex.person_code);
END LOOP;
```

Cursor-for loops automatically open a cursor, read in their data and close the cursor again

As an alternative, the PL/SQL programmer can pre-define the cursor's SELECT-statement in advance in order (for example) to allow re-use or to make the code more understandable (especially useful in the case of long or complex queries).

```
DECLARE
    CURSOR cursor_person IS
        SELECT person_code FROM people_table;
```

```
BEGIN
  FOR RecordIndex IN cursor_person
  LOOP
    DBMS_OUTPUT.PUT_LINE(RecordIndex.person_code);
  END LOOP;
END;
```

The concept of the person\_code within the FOR-loop gets expressed with dot-notation ("."):

```
RecordIndex.person_code
```

## Example

```
declare
  var number;
begin
  /*N.B. for loop variables in pl/sql are new declarations, with scope only inside the loop */
  for var in 0 .. 10 loop
    dbms_output.put_line(var);
  end loop;

  if (var is null) then
    dbms_output.put_line('var is null');
  else
    dbms_output.put_line('var is not null');
  end if;
end;
```

Output:

```
0
1
2
3
4
5
6
7
8
9
10
var is null
```

## Similar languages

PL/SQL functions analogously to the embedded procedural languages associated with other relational databases. Sybase ASE and Microsoft SQL Server have Transact-SQL, PostgreSQL has PL/pgSQL (which tries to emulate PL/SQL to an extent), and IBM DB2 includes SQL Procedural Language,<sup>[6]</sup> which conforms to the ISO SQL's SQL/PSM standard.

The designers of PL/SQL modelled its syntax on that of Ada. Both Ada and PL/SQL have Pascal as a common ancestor, and so PL/SQL also resembles Pascal in numerous aspects. The structure of a PL/SQL package closely resembles the basic Pascal program structure or a Borland Delphi unit. Programmers can define global data-types, constants and static variables, public and private, in a PL/SQL package.

PL/SQL also allows for the definition of classes and instantiating these as objects in PL/SQL code. This resembles usages in object-oriented programming languages like Object Pascal, C++ and Java. PL/SQL refers to a class as an "Abstract Data Type" (ADT) or "User Defined Type" (UDT), and defines it as an Oracle SQL data-type as opposed to a PL/SQL user-defined type, allowing its use in both the Oracle SQL Engine and the Oracle PL/SQL engine. The constructor and methods of an Abstract Data Type are written in PL/SQL. The resulting Abstract Data Type can operate as an object class in PL/SQL. Such objects can also persist as column values in Oracle database tables.

PL/SQL does not resemble Transact-SQL, despite superficial similarities. Porting code from one to the other usually involves non-trivial work, not only due to the differences in the feature sets of the two languages, but also due to the very significant differences in the way Oracle and SQL Server deal with concurrency and locking.

The Fyralce project aims to enable the execution of PL/SQL code in the open-source Firebird database.

The StepSqlite product is a PL/SQL compiler for the popular small database SQLite.

## See also

- Relational database management systems

## References

- [1] [http://otndnld.oracle.co.jp/document/products/V16967\\_01/doc/timesten.1121/e13076/intro.htm](http://otndnld.oracle.co.jp/document/products/V16967_01/doc/timesten.1121/e13076/intro.htm)
- [2] <http://www.ibm.com/developerworks/data/library/techarticle/dm-0907oracleappsondb2/index.html>
- [3] [http://download.oracle.com/docs/cd/B19306\\_01/appdev.102/b14261/block\\_declaration.htm](http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/block_declaration.htm)
- [4] "Literals" ([http://download.oracle.com/docs/cd/B19306\\_01/server.102/b14200/sql\\_elements003.htm#sthref365](http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/sql_elements003.htm#sthref365)). *Oracle Database SQL Reference 10g Release 2 (10.2)*. Oracle. . Retrieved 2009-03-20.
- [5] [http://download.oracle.com/docs/cd/B19306\\_01/appdev.102/b14261/controlstructures.htm#sthref921](http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/controlstructures.htm#sthref921)
- [6] SQL PL (<http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/ad/c0011916.htm>)
- Feuerstein, Steven; with Bill Pribyl (2005). *Oracle PL/SQL Programming* (4th ed. ed.). O'Reilly & Associates. ISBN 0-596-00977-1.
- Naudé, Frank (June 9, 2005). "Oracle PL/SQL FAQ rev 2.08" (<http://www.orafaq.com/faqplsql.htm>).

## External links

- Oracle FAQ: PL/SQL ([http://www.orafaq.com/wiki/PL/SQL\\_FAQ](http://www.orafaq.com/wiki/PL/SQL_FAQ))
- Oracle Technology Center ([http://www.oracle.com/technology/tech/pl\\_sql/index.html](http://www.oracle.com/technology/tech/pl_sql/index.html))
- Oracle Tahiti Search Engine (<http://tahiti.oracle.com>)
- Morgan's Library (<http://www.morganslibrary.org/library.html>)

# Pro\*C

---

**Pro\*C** (also known as **Pro\*C/C++**) is an embedded SQL programming language used by Oracle Database and Sybase SQL Server database management systems. Pro\*C uses either C or C++ as its host language. During compilation, the embedded SQL statements are interpreted by a precompiler and replaced by C or C++ function calls to their respective SQL library. The output from the Pro\*C precompiler is standard C or C++ code that is then compiled by any one of several C or C++ compilers into an executable.

## See also

- Embedded SQL

## External links

- Introduction to Pro\*C Embedded SQL <sup>[9]</sup>
- Embedded SQL with Pro\*C <sup>[10]</sup>
- Oracle 10.2 Pro\*C/C++ Programmer's Guide <sup>[1]</sup>

## References

- [1] [http://download.oracle.com/docs/cd/B19306\\_01/appdev.102/b14407/toc.htm](http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14407/toc.htm)

# Query optimizer

---

The **query optimizer** is the component of a database management system that attempts to determine the most efficient way to execute a query. The optimizer considers the possible query plans for a given input query, and attempts to determine which of those plans will be the most efficient. Cost-based query optimizers assign an estimated "cost" to each possible query plan, and choose the plan with the smallest cost. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations required, the CPU requirements, and other factors determined from the data dictionary. The set of query plans examined is formed by examining the possible access paths (e.g. index scan, sequential scan) and join algorithms (e.g. sort-merge join, hash join, nested loops). The search space can become quite large depending on the complexity of the SQL query.

Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs. However, some database engines allow guiding the query optimizer with hints.

## Implementation

Most query optimizers represent query plans as a tree of "plan nodes". A plan node encapsulates a single operation that is required to execute the query. The nodes are arranged as a tree, in which intermediate results flow from the bottom of the tree to the top. Each node has zero or more child nodes—those are nodes whose output is fed as input to the parent node. For example, a join node will have two child nodes, which represent the two join operands, whereas a sort node would have a single child node (the input to be sorted). The leaves of the tree are nodes which produce results by scanning the disk, for example by performing an index scan or a sequential scan.

## Join ordering

The performance of a query plan is determined largely by the order in which the tables are joined. For example, when joining 3 tables A, B, C of size 10 rows, 10,000 rows, and 1,000,000 rows, respectively, a query plan that joins B and C first can take several orders-of-magnitude more time to execute than one that joins A and C first. Most query optimizers determine join order via a dynamic programming algorithm pioneered by IBM's System R database project. This algorithm works in two stages:

1. First, all ways to access each relation in the query are computed. Every relation in the query can be accessed via a sequential scan. If there is an index on a relation that can be used to answer a predicate in the query, an index scan can also be used. For each relation, the optimizer records the cheapest way to scan the relation, as well as the cheapest way to scan the relation that produces records in a particular sorted order.
2. The optimizer then considers combining each pair of relations for which a join condition exists. For each pair, the optimizer will consider the available join algorithms implemented by the DBMS. It will preserve the cheapest way to join each pair of relations, in addition to the cheapest way to join each pair of relations that produces its output according to a particular sort order.
3. Then all three-relation query plans are computed, by joining each two-relation plan produced by the previous phase with the remaining relations in the query.

In this manner, a query plan is eventually produced that joins all the queries in the relation. Note that the algorithm keeps track of the sort order of the result set produced by a query plan, also called an *interesting order*. During dynamic programming, one query plan is considered to beat another query plan that produces the same result, only if they produce the same sort order. This is done for two reasons. First, a particular sort order can avoid a redundant sort operation later on in processing the query. Second, a particular sort order can speed up a subsequent join because it clusters the data in a particular way.

Historically, System-R derived query optimizers would often only consider *left-deep* query plans, which first join two base tables together, then join the intermediate result with another base table, and so on. This heuristic reduces the number of plans that need to be considered ( $n!$  instead of  $4^n$ ), but may result in not considering the optimal query plan. This heuristic is drawn from the observation that join algorithms such as nested loops only require a single tuple (aka row) of the outer relation at a time. Therefore, a left-deep query plan means that fewer tuples need to be held in memory at any time: the outer relation's join plan need only be executed until a single tuple is produced, and then the inner base relation can be scanned (this technique is called "pipelining").

Subsequent query optimizers have expanded this plan space to consider "bushy" query plans, where both operands to a join operator could be intermediate results from other joins. Such bushy plans are especially important in parallel computers because they allow different portions of the plan to be evaluated independently.

## Query planning for nested SQL queries

A SQL query to a modern relational DBMS does more than just selections and joins. In particular, SQL queries often nest several layers of SPJ blocks (Select-Project-Join), by means of group by, exists, and not exists operators. In some cases such nested SQL queries can be flattened into a select-project-join query, but not always. Query plans for nested SQL queries can also be chosen using the same dynamic programming algorithm as used for join ordering, but this can lead to an enormous escalation in query optimization time. So some database management systems use an alternative rule-based approach that uses a query graph model.

## Cost estimation

One of the hardest problems in query optimization is to accurately estimate the costs of alternative query plans. Optimizers cost query plans using a mathematical model of query execution costs that relies heavily on estimates of the cardinality, or number of tuples, flowing through each edge in a query plan. Cardinality estimation in turn depends on estimates of the selection factor of predicates in the query. Traditionally, database systems estimate selectivities through fairly detailed statistics on the distribution of values in each column, such as histograms. This technique works well for estimation of selectivities of individual predicates. However many queries have conjunctions of predicates such as `select count(*) from R where R.make='Honda' and R.model='Accord'`. Query predicates are often highly correlated (for example, `model='Accord'` implies `make='Honda'`), and it is very hard to estimate the selectivity of the conjunct in general. Poor cardinality estimates and uncaught correlation are one of the main reasons why query optimizers pick poor query plans. This is one reason why a DBA should regularly update the database statistics, especially after major data loads/unloads.

## References

- Chaudhuri, Surajit (1998). "An Overview of Query Optimization in Relational Systems" <sup>[1]</sup>. *Proceedings of the ACM Symposium on Principles of Database Systems* <sup>[1]</sup>. pp. pages 34–43. doi:10.1145/275487.275492.
- Ioannidis, Yannis (March 1996). "Query optimization" <sup>[2]</sup>. *ACM Computing Surveys* **28** (1): 121–123. doi:10.1145/234313.234367.
- Selinger, P. G.; Astrahan, M. M.; Chamberlin, D. D.; Lorie, R. A.; Price, T. G. (1979). "Access Path Selection in a Relational Database Management System". *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. pp. 23–34. doi:10.1145/582095.582099

## See also

- Join Selection Factor
- Query optimization

## References

- [1] <http://citeseer.ist.psu.edu/chaudhuri98overview.html>  
[2] <http://citeseer.ist.psu.edu/487912.html>



# Query plan

A **query plan** (or **query execution plan**) is an ordered set of steps used to access or modify information in a SQL relational database management system. This is a specific case of the relational model concept of access plans.

Since SQL is declarative, there are typically a large number of alternative ways to execute a given query, with widely varying performance. When a query is submitted to the database, the query optimizer evaluates some of the different, correct possible plans for executing the query and returns what it considers the best alternative. Because query optimizers are imperfect, database users and administrators sometimes need to manually examine and tune the plans produced by the optimizer to get better performance.

## Generating query plans

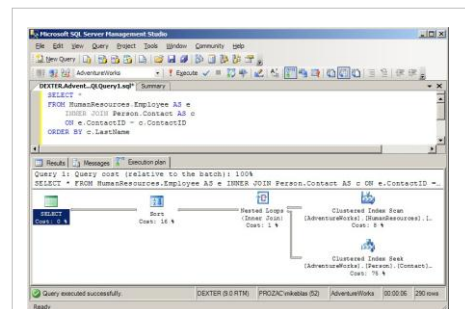
A given database management system may offer one or more mechanisms for returning the plan for a given query. Some packages feature tools which will generate a graphical representation of a query plan. Other tools allow a special mode to be set on the connection to cause the DBMS to return a textual description of the query plan. Another mechanism for retrieving the query plan involves querying a virtual database table after executing the query to be examined.

## Graphical plans

The SQL Server Management Studio tool which ships with Microsoft SQL Server, for example, shows this graphical plan when executing this two-table join against a sample database:

```
SELECT *
FROM HumanResources.Employee AS e
INNER JOIN Person.Contact AS c
ON e.ContactID = c.ContactID
ORDER BY c.LastName
```

The UI allows exploration of various attributes of the operators involved in the query plan, including the operator type, the number of rows each operator consumes or produces, and the expected cost of each operator's work.



Microsoft SQL Server Management Studio displaying a sample query plan.

## Textual plans

The textual plan given for the same query in the screenshot is shown here:

```
StmtText
----

|--Sort(ORDER BY:([c].[LastName] ASC))

|--Nested Loops(InnerJoin, OUTER REFERENCES:([e].[ContactID],[Expr1004]) WITH UNORDERED PREFETCH)

|--Clustered Index Scan(OBJECT:([AdventureWorks].[HumanResources].[Employee].[PK_Employee_EmployeeID] AS [e]))

|--Clustered Index Seek(OBJECT:([AdventureWorks].[Person].[Contact].[PK_Contact_ContactID] AS [c]),

SEEK:([c].[ContactID]=[AdventureWorks].[HumanResources].[Employee].[ContactID] as [e].[ContactID]) ORDERED FORWARD)
```

It indicates that the query engine will do a scan over the primary key index on the Employee table and a matching seek through the primary key index (the ContactID column) on the Contact table to find matching rows. The resulting rows from each side will be shown to a nested loops join operator, sorted, then returned as the result set to the connection.

In order to tune the query, the user must understand the different operators that the database may use, and which ones might be more efficient than others while still providing semantically correct query results.

## Database tuning

Reviewing the query plan can present opportunities for new indexes or changes to existing indexes. It can also show that the database is not properly taking advantage of existing indexes (see query optimizer).

## Query tuning

The query optimizer will not always choose the best query plan for a given query. In some databases the query plan can be reviewed, problems found, and then the query optimizer given hints on how to improve it. In other databases alternatives to express the same query (other queries that return the same results) can be tried. Some query tools can generate embedded hints in the query, for use by the optimizer.

Some databases like Oracle provide a Plan table for query tuning. This plan table will return the cost and time for executing a Query. In Oracle there are 2 optimization techniques:

1. CBO or Cost Based Optimization
2. RBO or Rule Based Optimization

The RBO is slowly being deprecated. For CBO to be used, all the tables referenced by the query must be analyzed. To analyze a table, a package DBMS\_STATS can be made use of.

The others methods for query optimization include:

1. SQL Trace
2. Oracle Trace
3. TKPROF

- Video tutorial on how to perform SQL performance tuning with reference to Oracle <sup>[1]</sup>

## References

[1] <http://seeingwithc.org/sqltuning.html>

---

# Rollback (data management)

---

In database technologies, a **rollback** is an operation which returns the database to some previous state. Rollbacks are important for database integrity, because they mean that the database can be restored to a clean copy even after erroneous operations are performed. They are crucial for recovering from database server crashes; by rolling back any transaction which was active at the time of the crash, the database is restored to a consistent state.

In SQL, ROLLBACK is a command that causes all data changes since the last BEGIN WORK, or START TRANSACTION to be discarded by the relational database management system (RDBMS), so that the state of the data is "rolled back" to the way it was before those changes were made.

A ROLLBACK statement will also release any existing savepoints that may be in use.

In most SQL dialects, ROLLBACKs are connection specific. This means that if two connections are made to the same database, a ROLLBACK made in one connection will not affect any other connections. This is vital for proper concurrency.

The rollback feature is usually implemented with a transaction log, but can also be implemented via multiversion concurrency control.

A cascading rollback occurs in database systems when a transaction (T1) causes a failure and a rollback must be performed. Other transactions dependent on T1's actions must also be rolled back due to T1's failure, thus causing a cascading effect.

## References

- "ROLLBACK Transaction" <sup>[1]</sup>, Microsoft SQL Server.
- "Sql Commands" <sup>[2]</sup>, Microsoft SQL Server.

## See also

- Savepoint
- Commit
- Norton Ghost

## References

[1] <http://msdn2.microsoft.com/en-us/library/ms181299.aspx>

[2] <http://www.pantz.org/software/mysql/mysqlcommands.html>

---

# SQL Access Group

---

The **SQL Access Group (SAG)** was a group of software companies that was formed in 1989 to define and promote standards for database portability and interoperability. Initial members were Oracle Corporation, Informix, Ingres, DEC, Tandem, Sun and HP.

The SAG started the development of the SQL Call Level Interface which later was published as an X/Open specification.

In 1992, Microsoft released version 1.0 of ODBC which was based on the X/Open SQL CLI specification.

The SQL Access Group transferred its activities and assets to X/Open in the fourth quarter of 1994.

## External links

- Introduction to SAG CLI by the SAG Chairman on Dr. Dobbs <sup>[1]</sup>
- Data Management: SQL Call Level Interface (CLI) ISBN 1-85912-081-4 Apr 1995 <sup>[1]</sup>

## References

[1] <http://www.ddj.com/database/184410032>

# SQL CLR

---

**SQL CLR** or **SQLCLR** (SQL Common Language Runtime) is technology for hosting of the Microsoft .NET common language runtime engine within SQL Server. The SQLCLR allows managed code to be hosted by, and run in, the Microsoft SQL Server environment.

This technology, introduced in Microsoft SQL Server 2005, allow users for example to create the following types of managed code objects in SQL Server in .NET languages such as C# or VB.NET.

- Stored procedures (SPs) which are analogous to *procedures* or *void functions* in procedural languages like VB or C,
- triggers which are stored procedures that fire in response to Data Manipulation Language (DML) or Data Definition Language (DDL) events,
- User-defined functions (UDFs) which are analogous to functions in procedural languages,
- User-defined aggregates (UDAs) which allow developers to create custom aggregates that act on sets of data instead of one row at a time,
- User-defined types (UDTs) that allow users to create simple or complex data types which can be serialized and deserialized within the database.

The SQL CLR relies on the creation, deployment, and registration of .NET assemblies, which are physically stored in managed code dynamic load libraries (DLLs). These assemblies may contain .NET namespaces, classes, functions and properties.

---

## External links

- MSDN: Using CLR Integration in SQL Server 2005<sup>[1]</sup>
- MSDN Forum on .NET Framework in SQL Server<sup>[2]</sup>
- SqlClr.net Independent site<sup>[3]</sup>
- SQL CLR Team Blog (No posts since 2006, might be dead)<sup>[4]</sup>

## References

- [1] <http://msdn2.microsoft.com/en-us/library/ms345136.aspx>  
[2] <http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=86&SiteID=1>  
[3] <http://www.sqlclr.net/>  
[4] <http://blogs.msdn.com/sqlclr/>

# SQL Problems Requiring Cursors

---

A cursor is a construct available in most implementations of SQL that allows the programmer to handle data in a row-by-row manner.

The trouble with cursor processing is that the row-by-row nature cannot easily be made to occur in parallel by database optimizers. This results in a failure of the cursor processing to take full advantage of the processing power available on most database platforms.

To overcome this problem, cursor logic can often be converted into set-based SQL queries. Set-based SQL uses the syntax "SELECT...FROM..." with various clauses to perform the data manipulation work. Database optimizers have little trouble dividing such queries into parallel threads, thus fully utilizing the database hardware.

The resulting performance gain can be literal orders of magnitude: hours become minutes, and minutes become seconds.

Despite the tremendous performance gains from set-based SQL, it can be challenging to find the SQL that replaces fairly straight-forward cursors.

This article discusses examples of such conversions.

## Constraints

In this article, the following constraints apply:

- **The term "cursor" includes all cursor-like behavior.** For example, using a loop in a shell script that loops across SQL queries or the output of SQL queries is cursor-like behavior and does not achieve the goal of true set-based processing within the database.
  - **All set-based SQL must be ANSI SQL.** A number of vendors provide some extremely powerful proprietary extensions. The goal is to avoid such extensions in favor of ANSI SQL.
  - **The solution must be generalizable.** In one or more examples below, specific values may be used for demonstration purposes, but any solution must scale to any number feasible within the power of the database software and machine resources.
-

## Example: Insert rows based on a count in the table itself

### Problem

The table below represents marbles. The four text columns represent four characteristics of marbles. Each characteristic has two values for a total of 16 types of marbles.

The "quantity" column represents how many of that marble we have. But instead of having a quantity, we want to have one row for each marble.

Thus, the target table would have the four text columns, and a total of  $40 + 20 + 20 + 10 + \dots + 10 + 5 = 270$  rows.

Source table:

QUANTITY	TEXTURE	APPEARANCE	SHAPE	COLOR
40	smooth	shiny	round	blue
20	smooth	shiny	warped	blue
20	smooth	dull	round	blue
10	smooth	dull	warped	blue
20	rough	shiny	round	blue
10	rough	shiny	warped	blue
10	rough	dull	round	blue
5	rough	dull	warped	blue
40	rough	dull	warped	red
20	rough	dull	round	red
20	rough	shiny	warped	red
10	rough	shiny	round	red
20	smooth	dull	warped	red
10	smooth	dull	round	red
10	smooth	shiny	warped	red
5	smooth	shiny	round	red

Table to generate:

TEXTURE	APPEARANCE	SHAPE	COLOR	
smooth	shiny	round	blue	-- 1
smooth	shiny	round	blue	-- 2
...				-- and so on
smooth	shiny	round	blue	-- 40
smooth	shiny	warped	blue	-- 1
smooth	shiny	warped	blue	-- 2
...				-- and so on
smooth	shiny	warped	blue	-- 20
...				-- and so on
smooth	shiny	round	red	-- 1
smooth	shiny	round	red	-- 2
smooth	shiny	round	red	-- 3
smooth	shiny	round	red	-- 4
smooth	shiny	round	red	-- 5

### Solution in cursor form

Generating the target table with a cursor is fairly simple.

```
pre> declare
```

```
cursor c is select * from marbles_seed;
```

```
begin
```

```
  for r in c loop
    for i in 1..r.quantity loop
```

```
      000000000000r.co
```

```
      lor_predicted,
```

```
          r.accuracy,
          r.coverage
        );
    end loop;
  end loop;
```

```
end; </pre>
```

### Solution in ANSI set-based SQL

Solving the problem with ANSI SQL is a bit more code, but requires a bit more creative thought than the nested loop approach of cursors.

#### Number Table

To reach the solution requires an intermediate table. The table has one column of type NUMBER that has the values 0 to whatever number of rows is needed. For this discussion, we'll limit it to one million rows. The code is as follows:

```
<
```

#### Solution Core

Assume the source table above is named `marbles_seed` and the target table is named `marbles`. The code that generates the needed 270 rows is:

```
insert into marbles
select m.texture,
       m.appearance,
       m.shape,
       m.color_actual,
       m.hits,
       m.color_predicted,
       m.accuracy,
       m.coverage
from   marbles_seed m,
       numbers n
where  m.quantity > n.n
```

# SQL injection

**SQL injection** is a code injection technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another. SQL injection attacks are also known as SQL insertion attacks.<sup>[1]</sup>

## Forms of vulnerability

### Incorrectly filtered escape characters

This form of SQL injection occurs when user input is not filtered for escape characters and is then passed into an SQL statement. This results in the potential manipulation of the statements performed on the database by the end user of the application.

The following line of code illustrates this vulnerability:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "userName" variable as

```
' or '1'='1
```

renders this SQL statement by the parent language:

```
SELECT * FROM users WHERE name = '' or '1'='1';
```

If this code were to be used in an authentication procedure then this example could be used to force the selection of a valid username because the evaluation of '1'='1' is always true.

The following value of "userName" in the statement below would cause the deletion of the "users" table as well as the selection of all data from the "userinfo" table (in essence revealing the information of every user), using an API that allows multiple statements:

```
a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't
```

This input renders the final SQL statement as follows:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM  
userinfo WHERE 't' = 't';
```

While most SQL server implementations allow multiple statements to be executed with one call in this way, some SQL APIs such as PHP's `mysql_query()` do not allow this for security reasons. This prevents attackers from injecting entirely separate queries, but doesn't stop them from modifying queries.



## Incorrect type handling

This form of SQL injection occurs when a user supplied field is not strongly typed or is not checked for type constraints. This could take place when a numeric field is to be used in a SQL statement, but the programmer makes no checks to validate that the user supplied input is numeric. For example:

```
statement := "SELECT * FROM userinfo WHERE id = " + a_variable + " ;"
```

It is clear from this statement that the author intended `a_variable` to be a number correlating to the "id" field. However, if it is in fact a string then the end user may manipulate the statement as they choose, thereby bypassing the need for escape characters. For example, setting `a_variable` to

```
1;DROP TABLE users
```

will drop (delete) the "users" table from the database, since the SQL would be rendered as follows:

```
SELECT * FROM userinfo WHERE id=1;DROP TABLE users;
```

## Vulnerabilities inside the database server

Sometimes vulnerabilities can exist within the database server software itself, as was the case with the MySQL server's `mysql_real_escape_string()` function<sup>[2]</sup>. This would allow an attacker to perform a successful SQL injection attack based on bad Unicode characters even if the user's input is being escaped. This bug was patched with the release of version 5.0.22 (released on 24th May 06).

## Blind SQL injection

Blind SQL Injection is used when a web application is vulnerable to an SQL injection but the results of the injection are not visible to the attacker. The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page. This type of attack can become time-intensive because a new statement must be crafted for each bit recovered. There are several tools that can automate these attacks once the location of the vulnerability and the target information has been established.<sup>[3]</sup>

### Conditional responses

One type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen.

```
SELECT booktitle from booklist where bookId = 'OOk14cd' AND 1=1 ;
```

will result in a normal page while

```
SELECT booktitle from booklist where bookId = 'OOk14cd' AND 1=2 ;
```

will likely give a different result if the page is vulnerable to a SQL injection. An injection like this may suggest to the attacker that a blind SQL injection is possible, leaving the attacker to devise statements that evaluate to true or false depending on the contents of another column or table outside of the SELECT statement's column list.<sup>[4]</sup>

### Conditional errors

This type of blind SQL injection causes an SQL error by forcing the database to evaluate a statement that causes an error if the WHERE statement is true. For example,

```
SELECT 1/0 from users where username='Ralph';
```

the division by zero will only be evaluated and result in an error if user Ralph exists.

### Time delays

Time Delays are a type of blind SQL injection that cause the SQL engine to execute a long running query or a time delay statement depending on the logic injected. The attacker can then measure the time the page takes to load to determine if the injected statement is true.

## Preventing SQL injection

To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, parameterized statements must be used (preferred), or user input must be carefully escaped or filtered.

### Parameterized statements

With most development platforms, parameterized statements can be used that work with parameters (sometimes called placeholders or bind variables) instead of embedding user input in the statement. In many cases, the SQL statement is fixed, and each parameter is a scalar, not a table. The user input is then assigned (bound) to a parameter. This is an example using Java and the JDBC API:

```
PreparedStatement prep = conn.prepareStatement("SELECT * FROM USERS  
WHERE USERNAME=? AND PASSWORD=?");  
prep.setString(1, username);  
prep.setString(2, password);  
prep.executeQuery();
```

### Enforcement at the database level

Currently only the H2 Database Engine supports the ability to enforce query parameterization.<sup>[5]</sup> However, one drawback is that query by example may not be possible or practical because it's difficult to implement query by example using parametrized queries.

### Enforcement at the coding level

Using object-relational mapping libraries avoids the need to write SQL code. The ORM library in effect will generate parameterized SQL statements from object-oriented code.

### Escaping

A straight-forward, though error-prone, way to prevent injections is to *escape* characters that have a special meaning in SQL. The manual for an SQL DBMS explains which characters have a special meaning, which allows creating a comprehensive blacklist of characters that need translation. For instance, every occurrence of a single quote (') in a parameter must be replaced by two single quotes (") to form a valid SQL string literal. In PHP, for example, it is usual to escape parameters using the function `mysql_real_escape_string` before sending the SQL query:

```
$query = sprintf("SELECT * FROM Users where UserName='%s' and  
Password='%s'",  
                mysql_real_escape_string($Username),
```

```
mysql_real_escape_string($Password));  
mysql_query($query);
```

This is error prone because it is easy to forget to escape a given string.

## Real-world examples

- On November 1, 2005, a high school student used SQL injection to break into the site of a Taiwanese information security magazine from the Tech Target group and steal customers' information.<sup>[6]</sup>
- On January 13, 2006, Russian computer criminals broke into a Rhode Island government web site and allegedly stole credit card data from individuals who have done business online with state agencies.<sup>[7]</sup>
- On March 29, 2006, Susam Pal discovered an SQL injection flaw in an official Indian government tourism site.<sup>[8]</sup>
- On March 2, 2007, Sebastian Bauer discovered an SQL injection flaw in the knorr.de login page.<sup>[9]</sup>
- On June 29, 2007, a computer criminal defaced the Microsoft U.K. website using SQL injection.<sup>[10] [11]</sup> U.K. website *The Register* quoted a Microsoft spokesperson acknowledging the problem.
- In January 2008, tens of thousands of PCs were infected by an automated SQL injection attack that exploited a vulnerability in application code that uses Microsoft SQL Server as the database store.<sup>[12]</sup>
- On April 13, 2008, the Sexual and Violent Offender Registry of Oklahoma shut down its website for 'routine maintenance' after being informed that 10,597 Social Security numbers from sex offenders had been downloaded via an SQL injection attack<sup>[13]</sup>
- In May 2008, a server farm inside China used automated queries to Google's search engine to identify SQL server websites which were vulnerable to the attack of an automated SQL injection tool.<sup>[12] [14]</sup>
- In 2008, at least April through August, a sweep of attacks began exploiting the SQL injection vulnerabilities of Microsoft's IIS web server and SQL Server database server. The attack doesn't require guessing the name of a table or column, and corrupts all text columns in all tables in a single request.<sup>[15]</sup> A HTML string that references a malware JavaScript file is appended to each value. When that database value is later displayed to a website visitor, the script attempts several approaches at gaining control over a visitor's system. The number of exploited web pages is estimated at 500,000<sup>[16]</sup>
- On August 17, 2009, the United States Justice Department charged an American citizen Albert Gonzalez and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack. In reportedly "the biggest case of identity theft in American history", the man stole cards from a number of corporate victims after researching their payment processing systems. Among the companies hit were credit card processor Heartland Payment Systems, convenience store chain 7-Eleven, and supermarket chain Hannaford Brothers.<sup>[17]</sup>
- In December 2009, an attacker breached a RockYou! plaintext database containing the unencrypted usernames and passwords of about 32 million users using an SQL injection attack.<sup>[18]</sup>

## External links

- WASC Threat Classification - SQL Injection Entry <sup>[19]</sup>, by the Web Application Security Consortium
- Why SQL Injection Won't Go Away <sup>[20]</sup>, by Stuart Thomas
- SQL Injection Attacks by Example <sup>[21]</sup>, by Steve Friedl

## References

- [1] Watson, Carli (2006) *Beginning C# 2005 databases* ISBN 978-0-470-04406-3, pages 201-5
- [2] "E.1.7. Changes in MySQL 5.0.22 (24 May 2006)" (<http://dev.mysql.com/doc/refman/5.0/en/news-5-0-22.html>). MySQL AB. May 4, 2006. . Retrieved May 16, 2008., "An SQL-injection security hole has been found in multi-byte encoding processing", retrieved March 20, 2008
- [3] "Using SQLBrute to brute force data from a blind SQL injection point" (<http://www.justinclarke.com/archives/2006/03/sqlbrute.html>). Justin Clarke. . Retrieved October 18, 2008.
- [4] Ofer Maor and Amichai Shulman. "Blind SQL Injection: Getting the syntax right" ([http://www.imperva.com/resources/adcs/blind\\_sql\\_server\\_injection.html#getting\\_syntax\\_right](http://www.imperva.com/resources/adcs/blind_sql_server_injection.html#getting_syntax_right)). Imperva. . Retrieved May 16, 2008. "This is usually the trickiest part in the blind SQL injection process. If the original queries are simple, this is simple as well. However, if the original query was complex, breaking out of it may require a lot of trial and error."
- [5] "SQL Injections: How Not To Get Stuck" (<http://thecodist.com/article/sql-injections-how-not-to-get-stuck>). The Codist. May 8, 2007. . Retrieved February 1, 2010.
- [6] "WHID 2005-46: Teen uses SQL injection to break to a security magazine web site" (<http://www.xiom.com/whid-2005-46>). Web Application Security Consortium. November 1, 2005. . Retrieved December 1, 2009.
- [7] "WHID 2006-3: Russian hackers broke into a RI GOV website" (<http://www.xiom.com/whid-2006-3>). Web Application Security Consortium. January 13, 2006. . Retrieved May 16, 2008.
- [8] "WHID 2006-27: SQL Injection in incredibleindia.org" (<http://www.xiom.com/whid-2006-27>). Web Application Security Consortium. March 29, 2006. . Retrieved March 12, 2010.
- [9] "WHID 2007-12: SQL injection at knorr.de login page" (<http://www.xiom.com/whid-2007-12>). Web Application Security Consortium. March 2, 2007. . Retrieved March 12, 2010.
- [10] Robert (June 29, 2007). "Hacker Defaces Microsoft U.K. Web Page" (<http://www.cgisecurity.net/2007/06/hacker-defaces.html>). cgisecurity.net. . Retrieved May 16, 2008.
- [11] Keith Ward (June 29, 2007). "Hacker Defaces Microsoft U.K. Web Page" (<http://rcpmag.com/news/article.aspx?editorialid=8762>). Redmond Channel Partner Online. . Retrieved May 16, 2008.
- [12] Sumner Lemon, IDG News Service (May 19, 2008). "Mass SQL Injection Attack Targets Chinese Web Sites" ([http://www.pcworld.com/businesscenter/article/146048/mass\\_sql\\_injection\\_attack\\_targets\\_chinese\\_web\\_sites.html](http://www.pcworld.com/businesscenter/article/146048/mass_sql_injection_attack_targets_chinese_web_sites.html)). PCWorld. . Retrieved May 27, 2008.
- [13] Alex Papadimoulis (April 15, 2008). "Oklahoma Leaks Tens of Thousands of Social Security Numbers, Other Sensitive Data" (<http://thedailywtf.com/Articles/Oklahoma-Leaks-Tens-of-Thousands-of-Social-Security-Numbers,-Other-Sensitive-Data.aspx>). The Daily WTF. . Retrieved May 16, 2008.
- [14] Michael Zino (May 1, 2008). "ASCII Encoded/Binary String Automated SQL Injection Attack" (<http://www.bloombit.com/Articles/2008/05/ASCII-Encoded-Binary-String-Automated-SQL-Injection.aspx>). .
- [15] Giorgio Maone (April 26, 2008). "Mass Attack FAQ" (<http://hackademix.net/2008/04/26/mass-attack-faq/>). .
- [16] Gregg Keizer (April 25, 2008). "Huge Web hack attack infects 500,000 pages" (<http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9080580>). .
- [17] "US man 'stole 130m card numbers'" (<http://news.bbc.co.uk/2/hi/americas/8206305.stm>). BBC. August 17, 2009. . Retrieved August 17, 2009.
- [18] O'Dell, Jolie (December 16, 2009). "RockYou Hacker - 30% of Sites Store Plain Text Passwords" (<http://www.nytimes.com/external/readwriteweb/2009/12/16/16readwriteweb-rockyou-hacker-30-of-sites-store-plain-text-13200.html>). *The New York Times*. . Retrieved May 23, 2010.
- [19] <http://projects.webappsec.org/SQL-Injection>
- [20] <http://docs.google.com/leaf?id=0BykNNUTb95yzYTRjMjNjMWEtODBmNS00YzgWLTImMGYtNWZmODI2MTNmZWYw&sort=name&layout=list&num=50>
- [21] <http://unixwiz.net/techtips/sql-injection.html>

# SQL/CLI

---

The **SQL/CLI**, or *Call-Level Interface*, extension to the SQL standard is defined in ISO/IEC 9075-3:2003. This extension defines common interfacing components (structures and procedures) that can be used to execute SQL statements from applications written in other programming languages. The SQL/CLI extension is defined in such a way that SQL statements and SQL/CLI procedure calls are treated as separate from the calling application's source code.

## See also

- SQL
- SQL:2003

# SQL/MED

---

The **SQL/MED**, or *Management of External Data*, extension to the SQL standard is defined by ISO/IEC 9075-9:2003. SQL/MED provides extensions to SQL that define foreign-data wrappers and datalink types to allow SQL to manage external data. External data is data that is accessible to, but not managed by, an SQL-based DBMS. This standard can be used in the development of Federated database systems

## See also

- SQL
- SQL:2003

## External links

SQL/MED - A Status Report <sup>[1]</sup>

## References

- [1] <http://www.sigmod.org/record/issues/0209/jimmelman.pdf>

# SQL/OLB

---

The **SQL/OLB**, or *Object Language Bindings*, extension to the SQL standard is defined by ISO/IEC 9075-10:2003. SQL/OLB defines the syntax and semantics of SQLJ, which is SQL embedded in Java. The standard also describes mechanisms to ensure binary portability of SQLJ applications, and specifies various Java packages and their contained classes.

## See also

- SQL
- SQL:2003

## External links

- Short description <sup>[1]</sup>

## References

- [1] <http://www.jcc.com/sql.htm>

# SQL/PSM

---

**SQL/PSM** stands for Structured Query Language/Persistent Stored Modules, and was developed by the American National Standards Institute (ANSI) as an extension to SQL.<sup>[1]</sup> It was first adopted in 1996,<sup>[2]</sup> and it provides procedural programmability in addition to the querying commands of SQL.

The SQL/PSM extension is defined by ISO/IEC 9075-4:2003. SQL/PSM standardizes procedural extensions for SQL, including flow of control, condition handling, statement condition signals and resignals, cursors and local variables, and assignment of expressions to variables and parameters. In addition, SQL/PSM formalizes declaration and maintenance of persistent database language routines (e.g., "stored procedures").

## See also

- SQL
- SQL:2003

## External links

Some samples of SQL/PSM functions <sup>[3]</sup>

## References

- [1] "Database Language SQL" ([http://www.itl.nist.gov/div897/ctg/dm/sql\\_info.html](http://www.itl.nist.gov/div897/ctg/dm/sql_info.html)). NIST SQL Project. . Retrieved February 17, 2010.
- [2] Celko, Joe (January 1997). "The Future of SQL Programming" (<http://web.archive.org/web/20070927045233/http://www.dbmsmag.com/9701d06.html>). *DBMS Magazine* (Miller Freeman). Archived from the original (<http://www.dbmsmag.com/9701d06.html>) on September 27, 2007. . Retrieved February 17, 2010.
- [3] [http://www.pgsql.cz/index.php/SQL/PSM\\_Manual](http://www.pgsql.cz/index.php/SQL/PSM_Manual)

# SQL/Schemata

---

The **SQL/Schemata**, or *Information and Definition Schemas*, part to the SQL standard is defined by ISO/IEC 9075-11:2008. SQL/Schemata defines the Information Schema and Definition Schema, providing a common set of tools to make SQL databases and objects self-describing. These tools include the SQL object identifier, structure and integrity constraints, security and authorization specifications, features and packages of ISO/IEC 9075, support of features provided by SQL-based DBMS implementations, SQL-based DBMS implementation information and sizing items, and the values supported by the DBMS implementations.<sup>[1]</sup> SQL/Schemata defines a number of features, some of which are mandatory.

## See also

- Data dictionary
- SQL
- SQL:2003

## References

[1] ISO/IEC 9075-11:2008: *Information and Definition Schemas (SQL/Schemata)*, 2008, pp. p. 1

# SQL/XML

---

The **SQL/XML**, or *XML-Related Specifications*, extension to the SQL standard is defined by ISO/IEC 9075-14:2003 (see SQL:2003). SQL/XML specifies SQL-based extensions for using XML in conjunction with SQL. The XML data type is introduced, as well as several routines, functions, and XML-to-SQL data type mappings to support manipulation and storage of XML in a SQL database.

The SQL/XML specification includes functions to construct XML data. These functions allow the user to construct new XML elements or attributes with values e.g. from relational tables. Other functions such as XMLCONCAT or XMLAGG can be used to combine small XML fragments into larger ones. The list of available construction functions includes:

- XMLELEMENT
- XMLATTRIBUTES
- XMLFOREST
- XMLCONCAT
- XMLNAMESPACES
- XMLCOMMENT
- XMLPI
- XMLDOCUMENT
- XMLAGG
- etc.

SQL/XML also defines functions which allow the user to embed XQuery expressions in SQL statements. These functions include:

- XMLQUERY
- XMLTABLE

While XMLQUERY returns values of type XML, the function XMLTABLE can take XML data as input and produce a relational table as output. Predicates on XML data, such as search conditions, can be expressed with the

---

XMLEXISTS predicate, typically in the WHERE clause of a SQL statement.

Further information and examples of the SQL/XML functions are provided in the external links below.

## External links

- Advancements in SQL/XML <sup>[1]</sup> Andrew Eisenberg, Jim Melton; SIGMOD Record 33(3): 79-86 2004.
- SQL/XML is Making Good Progress <sup>[2]</sup> Andrew Eisenberg, Jim Melton; SIGMOD Record 31(2): 101-108 2002.
- SQLXML.org <sup>[3]</sup>
- SQLX.org <sup>[4]</sup>

## References

- [1] <http://www.sigmod.org/sigmod/record/issues/0409/11.JimMelton.pdf>
  - [2] <http://acm.org/sigmod/record/issues/0206/standard.pdf>
  - [3] <http://sqlxml.org/>
  - [4] <http://sqlx.org/>
-



# SQLPro SQL Client

---

<b>Original author(s)</b>	Vive Corp.
<b>Stable release</b>	1.4 / February 4, 2008
<b>Operating system</b>	Windows
<b>Type</b>	SQL Programming Tool, Programming tool, Database administration and automation
<b>Website</b>	[1] <sup>[1]</sup>

**SQLPro** is a proprietary, visual database management and development tool for multiple databases. SQLPro comes with a three-pane interface and has integrated SQL editor with many useful SQL editing features. SQLPro connects directly (using native drivers) to most popular database servers using one single interface. It supports native data source connection for Oracle, PostgreSQL, MySQL, Microsoft SQL Server, Microsoft Access and SQLite. Provides quick view of the database schema, table schema and results of the query execution.

Amenities include color-coding of the SQL, drag-and-drop of objects into the SQL pane to save you from typing their name, retrieval of SQL for things like stored procedures and triggers from the underlying database, and one-click creation of SELECT and INSERT statements. You can save SQL files and print result sets.

## Feature Summary

- SQLPro runs on all 32-bit Windows platforms, including Windows 95, 98, NT, 2000, XP, and Vista.
- No specific hardware requirements.
- Color coding of SQL keywords
- Retrieve stored procedures, triggers and functions
- No other software or third party drivers to install. SQLPro comes with full set of drivers.
- Multiple host/database connections.

## Supported Databases

- Microsoft Access
  - Microsoft SQL Server
  - MySQL
  - Oracle 8i, 9i, 10g and 11g
  - PostgreSQL
  - SQLite
-

## External links

- Official Web Site <sup>[2]</sup>
- Review: SQLPro <sup>[3]</sup>

## References

[1] <http://www.vive.net/products/sqlpro.htm>


[2] <http://www.vive.net/>

[3] <http://www.larkware.com/Reviews/sqlpro.html>

---

# Scriptella

---

	
<b>Stable release</b>	1.0 / 5 May 2010
<b>Operating system</b>	Cross-platform
<b>Type</b>	ETL, Data migration and SQL.
<b>License</b>	Apache Software License
<b>Website</b>	scriptella.javaforge.com <sup>[1]</sup>

**Scriptella** is an open source ETL (Extract-Transform-Load) and script execution tool written in Java. Its primary focus is simplicity. It doesn't require the user to learn another complex XML-based language to use it, but allows the use of SQL or another scripting language suitable for the data source to perform required transformations.

Potential users should be aware that Scriptella does not offer any graphical user interface.

## Typical use

- Database migration.
- Database creation/update scripts.
- Cross-database ETL operations, import/export.
- Alternative for Ant `<sql>` task.
- Automated database schema upgrade.

## Features

- **Simple XML syntax** for scripts. Add dynamics to your existing SQL scripts by creating a thin wrapper XML file:

```
<!DOCTYPE etl SYSTEM "http://scriptella.javaforge.com/dtd/etl.dtd">
<etl>
  <connection driver="$driver" url="$url" user="$user" password="$password"/>
  <script>
    <include href="PATH_TO_YOUR_SCRIPT.sql"/>
    -- And/or directly insert SQL statements here
  </script>
</etl>
```

- Support for **multiple datasources** (or multiple connections to a single database) in a ETL file.
  - Support for many useful **JDBC features**, e.g. parameters in SQL including file blobs and JDBC escaping.
  - **Performance.** Performance and low memory usage are one of our primary goals.
  - Support for **evaluated expressions and properties** (JEXL syntax)
  - Support for **cross-database ETL scripts** by using `<dialect>` elements
  - **Transactional execution**
  - **Error handling** via `<onerror>` elements
  - **Conditional scripts/queries execution** (similar to Ant if/unless attributes but more powerful)
  - **Easy-to-Use** as a standalone tool or Ant task. No deployment/installation required.
  - **Easy-To-Run** ETL files directly from Java code.
-

- **Built-in adapters for popular databases** for a tight integration. Support for any database with JDBC/ODBC compliant driver.
- Service Provider Interface (SPI) for interoperability with non-JDBC DataSources and integration with scripting languages. Out of the box support for JSR 223 (Scripting for the Java Platform) compatible languages.
- Built-In CSV, TEXT, XML, LDAP, Lucene, Velocity, JEXL and Janino providers. Integration with Java EE, Spring Framework, JMX and JNDI for enterprise ready scripts.

## External links

- Scriptella ETL Site <sup>[1]</sup>
- Discussion forum <sup>[2]</sup>
- Scriptella ETL Author's Blog <sup>[3]</sup>
- Example code snippets for Scriptella ETL <sup>[4]</sup>
- Scriptella ETL <sup>[5]</sup> at Ohloh

## References

- [1] <http://scriptella.javaforge.com>  
[2] [http://www.javaforge.com/proj/forum/browseForum.do?forum\\_id=3126](http://www.javaforge.com/proj/forum/browseForum.do?forum_id=3126)  
[3] <http://jroller.com/page/ejboy>  
[4] <http://snippets.dzone.com/tag/scriptella>  
[5] <http://www.ohloh.net/projects/4526>

# SQL PL

---

**SQL PL** stands for Structured Query Language Procedural Language and was developed by IBM as a set of commands that extend the use of SQL in the IBM DB2 (DB2 UDB Version 7) database system.<sup>[1]</sup> It provides procedural programmability in addition to the querying commands of SQL.

## External links

- SQL PL Guide for developing Stored Procedures in DB2 <sup>[2]</sup>

## References

- [1] IBM Info Center (<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.apdv.sql.doc/doc/c0011916.htm>)  
[2] <http://www.sqlpl-guide.com>
-

# SQL/JRT

---

The **SQL/JRT**, or *SQL Routines and Types for the Java Programming Language*, extension to the SQL standard is defined by ISO/IEC 9075-13:2003. SQL/JRT specifies the ability to invoke static Java methods as routines from within SQL applications. It also calls for the ability to use Java classes as SQL structured user-defined types.

## See also

- SQL:2003

# Squirrel SQL Client

---

<b>Developer(s)</b>	Colin Bell, Gerd Wagner, Rob Manning and others
<b>Stable release</b>	3.1 / March 3, 2010
<b>Operating system</b>	Cross-platform
<b>Platform</b>	Java
<b>Type</b>	Database administration tool
<b>License</b>	LGPL
<b>Website</b>	<a href="http://www.squirrelsql.org">www.squirrelsql.org</a> <sup>[1]</sup>

The **Squirrel SQL Client** is a database administration tool. It uses JDBC to allow users to explore and interact with databases via a JDBC driver. It provides an editor that offers code completion and syntax highlighting for standard SQL. It also provides a plugin architecture that allows plugin writers to modify much of the application's behavior to provide database-specific functionality or features that are database-independent. As this desktop application is written entirely in Java with Swing UI components, it should run on any platform that has a JVM.<sup>[2]</sup>

Squirrel SQL Client is free as open source software that is distributed under the GNU Lesser General Public License.

## Feature Summary

- Object Tree allows for browsing database objects such as catalogs, schemas, tables, triggers, views, sequences, procedures, UDTs, etc.
- The SQL Editor is based on the NetBeans core editing components to provide syntax highlighting. It can be used to open, create, save and execute files containing SQL statements.
- Supports simultaneous sessions with multiple databases.
- It runs on any platform that has a JVM.
- Plugin architecture to facilitate database vendor-specific extensions (Information or actions not available using standard JDBC - see Squirrel SQL Client Plugin API for more details)
- Translations for the user interface are available in 8 languages (Bulgarian, Brazilian Portuguese, Chinese, Czech, French, German, Italian, Spanish).
- Graph capabilities to create charts showing table relationships.<sup>[3]</sup>
- Bookmarks, which are user-defined code templates. Squirrel comes with predefined example bookmarks for the most common SQL and DDL statements.<sup>[3]</sup>

## History

The Squirrel SQL project was developed by a team of Java developers around the world and led by Colin Bell. It has been hosted as a SourceForge project since 2001, and is still being developed today.

## Supported databases

- Axion Java RDBMS.
- Apache Derby
- Daffodil (One\$DB)
- Fujitsu Siemens SESAM/SQL-Server with the SESAM/SQL JDBC driver
- Firebird with the JayBird JCA/JDBC Driver
- FrontBase
- HSQLDB
- Hypersonic SQL
- H2 (DBMS)
- IBM DB2 for Linux, OS/400 and Windows
- Informix
- Ingres (and OpenIngres)
- InstantDB
- InterBase
- Mckoi SQL Database
- Microsoft Access with the JDBC/ODBC bridge.
- Microsoft SQL Server
- Mimer SQL
- MySQL
- Netezza
- Oracle Database 8i, 9i, 10g, 11g
- Pointbase
- PostgreSQL 7.1.3 and higher
- SAPDB
- Sybase
- Sunopsis XML Driver (JDBC Edition)
- Teradata Warehouse
- ThinkSQL RDBMS
- Vertica Analytic Database

## References

- [1] <http://www.squirrelsql.org/>
  - [2] Wagner, Gerd; Griffin, Glenn. "Squirrel, a Universal SQL Client" ([http://squirrel-sql.sourceforge.net/paper/Squirrel\\_us.pdf](http://squirrel-sql.sourceforge.net/paper/Squirrel_us.pdf)). . Retrieved 2009-04-15.
  - [3] "Squirrel SQL Client Home Page" (<http://squirrel-sql.sourceforge.net/>). SourceForge.net. . Retrieved 2009-04-15.
-

# Standard Interchange Language

---

The **Standard Interchange Language (SIL)** <sup>[1]</sup> is a data interchange language standard developed by the Food Distribution Retail Systems Group (FDRSG) for the interchange of information between software programs. It is a subset of SQL (Structured Query Language) and acts as an interface standard for transferring data between proprietary store systems like DSD (Direct Store Delivery) and POS (Point Of Sale). It was introduced in 1989 in the United States.

## External links

- SIL — Data interchange language <sup>[2]</sup>
- Expert: standard interchange language <sup>[3]</sup>

## References

- [1] Thayer, Warren. "Can SIL break the computer language barrier? The Standard Interchange Language — a data exchange standard designed with wholesalers in mind — may give retail systems integration a big boost", *Progressive Grocer*, January 1991.
- [2] <http://hopl.murdoch.edu.au/showlanguage.prx?exp=7474&language=SIL>
- [3] <http://www.intota.com/multisearch.asp?strSearchType=all&strQuery=standard+interchange+language>

# Table (database)

---

In relational databases and flat file databases, a **table** is a set of data elements (values) that is organized using a model of vertical columns (which are identified by their name) and horizontal rows. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a candidate key.

Table is another term for relations; although there is the difference in that a table is usually a multi-set (bag) of rows whereas a relation is a set and does not allow duplicates. Besides the actual data rows, tables generally have associated with them some meta-information, such as constraints on the table or on the values within particular columns.

The data in a table does not have to be physically stored in the database. Views are also relational tables, but their data are calculated at query time. Another example are nicknames, which represent a pointer to a table in another database.

## Comparisons with other data structures

In non-relational systems, hierarchical databases, the distant counterpart of a table is a structured file, representing the rows of a table in each record of the file and each column in a record.

Unlike a spreadsheet, the datatype of field is ordinarily defined by the schema describing the table. Some relational systems are less strict about field datatype definitions.

## Tables versus relations

In terms of the relational model of databases, a table can be considered a convenient representation of a relation, but the two are not strictly equivalent. For instance, an SQL table can potentially contain duplicate rows, whereas a true relation cannot contain duplicate tuples. Similarly, representation as a table implies a particular ordering to the rows and columns, whereas a relation is explicitly unordered. However, the database system does not guarantee any ordering of the rows unless an ORDER BY clause is specified in the SELECT statement that queries the table.

---



An equally valid representation of a relation is as an  $n$ -dimensional chart, where  $n$  is the number of attributes (a table's columns). For example, a relation with two attributes and three values can be represented as a table with two columns and three rows, or as a two-dimensional graph with three points. The table and graph representations are only equivalent if the ordering of rows is not significant, and the table has no duplicate rows.

## See also

- Relation (database)
- Table (information)

# Transact-SQL

**Transact-SQL (T-SQL)** is Microsoft's and Sybase's proprietary extension to SQL. Transact-SQL is central to using SQL Server. All applications that communicate with an instance of SQL Server do so by sending Transact-SQL statements to the server, regardless of the user interface of the application.

Transact-SQL augments SQL with certain additional features:

- Control-of-flow language
- Local variables
- Various support functions for string processing, date processing, mathematics, etc.
- Changes to DELETE and UPDATE statements

These additional features make Transact-SQL Turing complete.

## Flow control

Keywords for flow control in Transact-SQL include BEGIN and END, BREAK, CONTINUE, GOTO, IF and ELSE, RETURN, WAITFOR, and WHILE.

IF and ELSE allow conditional execution. This batch statement will print "It is the weekend" if the current date is a weekend day, or "It is a weekday" if the current date is a weekday.

```
IF DATEPART(dw, GETDATE()) = 7 OR DATEPART(dw, GETDATE()) = 1
    PRINT 'It is the weekend.'
ELSE
    PRINT 'It is a weekday.'
```

BEGIN and END mark a block of statements. If more than one statement is to be controlled by the conditional in the example above, we can use BEGIN and END like this:

```
IF DATEPART(dw, GETDATE()) = 7 OR DATEPART(dw, GETDATE()) = 1
BEGIN
    PRINT 'It is the weekend.'
    PRINT 'Get some rest!'
END
ELSE
BEGIN
    PRINT 'It is a weekday.'
    PRINT 'Get to work!'
END
```

WAITFOR will wait for a given amount of time, or until a particular time of day. The statement can be used for delays or to block execution until the set time.

RETURN is used to immediately return from a stored procedure or function.

BREAK ends the enclosing WHILE loop, while CONTINUE causes the next iteration of the loop to execute. An example of a WHILE loop is given below.

## Changes to DELETE and UPDATE statements

In Transact-SQL, both the DELETE and UPDATE statements allow a FROM clause to be added, which allows joins to be included.

This example deletes all users who have been flagged with the 'Idle' flag.

```
DELETE users
FROM users as u
JOIN user_flags as f
ON u.id=f.id
WHERE f.name = 'Idle'
```

## BULK INSERT

BULK INSERT is a Transact-SQL statement that implements a bulk data-loading process, inserting multiple rows into a table, reading data from an external sequential file. Use of BULK INSERT results in better performance than processes that issue individual INSERT statements for each row to be added. Additional details are available on Microsoft's MSDN page <sup>[1]</sup>.

## See also

- Adaptive Server Enterprise (Sybase)
- PL/SQL (Oracle)
- SQL (ANSI)
- SQL Server (Microsoft)

## External links

- Sybase Transact-SQL User's Guide <sup>[2]</sup>
- Transact-SQL Reference for SQL Server 2000 (MSDN) <sup>[3]</sup>
- Transact-SQL Reference for SQL Server 2005 (MSDN) <sup>[4]</sup>
- Transact-SQL Reference for SQL Server 2008 (MSDN) <sup>[5]</sup>

## References

- [1] <http://msdn2.microsoft.com/en-us/library/ms188365.aspx>
- [2] [http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase\\_15.0.sqlug/html/sqlug/title.htm](http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.sqlug/html/sqlug/title.htm)
- [3] [http://msdn2.microsoft.com/en-us/library/aa260642\(SQL.80\).aspx](http://msdn2.microsoft.com/en-us/library/aa260642(SQL.80).aspx)
- [4] <http://msdn2.microsoft.com/en-us/library/ms189826.aspx>
- [5] [http://msdn.microsoft.com/en-us/library/bb510741\(SQL.100\).aspx](http://msdn.microsoft.com/en-us/library/bb510741(SQL.100).aspx)
-

# Truviso

---

	
<b>Industry</b>	Software
<b>Founded</b>	2006
<b>Headquarters</b>	 Foster City, CA, USA
<b>Key people</b>	Tom Rowley Michael J. Franklin Sailesh Krishnamurthy Tom Kuhr Kim MacPherson
<b>Website</b>	<a href="http://www.truviso.com">http://www.truviso.com</a>

**Truviso** (pronounced *true-VEE-so*) is a continuous analytics, venture-backed, startup headquartered in Foster City, California developing and supporting its solution leveraging PostgreSQL, to deliver a proprietary analytics solutions for net-centric customers.

Truviso was founded in 2006<sup>[1]</sup> by UC Berkeley professor Michael J. Franklin<sup>[2]</sup> and his Ph.D. student Sailesh Krishnamurthy<sup>[3]</sup>, advancing on the research of Berkeley's Telegraph project<sup>[4]</sup>.

Truviso's TruCQ product leverages and extends the open source PostgreSQL database<sup>[5]</sup> to enable analysis of streaming data, including queries that combine those streams with other streaming data or with historical/staged data.<sup>[6] [7]</sup> One public example of Truviso's customers using continuous analytics is the dynamic tag cloud visualization of blog indexer Technorati.<sup>[8]</sup>

Truviso is one of the pioneers in the continuous analytics space which seeks to alter how business intelligence is done -- rather than accumulating data first and then running queries on the data set stored in a relational database or a data warehouse, Truviso has always-on queries which process streaming data as it arrives, continuously. For many queries this approach yields results hundreds or thousands of times faster and more efficiently.

Truviso has received funding from ONSET Ventures,<sup>[9]</sup> Diamondhead Ventures,<sup>[10]</sup> and the UPS Strategic Enterprise Fund.<sup>[11]</sup>

## Technology

Truviso's analytics approach is to have always-on queries analyzing streaming data. This strategy for handling continuously flowing data is different from traditional business intelligence approaches of first accumulating data and then running batch queries for reporting and analysis.

Truviso has developed a continuous analytics solution to solving the challenge of high-volume, always-on data analysis. Truviso's solution is based on a scalable PostgreSQL platform capable of concurrent query execution, utilizing standard SQL against live streams of data. Truviso's approach enables analysis of heterogeneous data regardless of whether the data is flowing, staged, or some combination of the two.

- Queries are continuous and always running so new results are delivered when the downstream application or use require them
  - Data does not need to be stored or modified, so the system can keep up with enormous data volumes
  - Thousands of concurrent queries can be run continuously and simultaneously on a single server
  - Queries can be run over both real-time and historical data
-

- Incoming data can be optionally persisted for replay, backtesting, drill-down, or benchmarking

On May 4, 2010, Truviso announced that the company developed a specific application for web analytics called Visitor Insight & Analytics<sup>[12]</sup>.

## External links

- Truviso's website <sup>[13]</sup>
- TDWI interview with Professor Michael Franklin <sup>[14]</sup>
- James Taylor's "First Look" on Truviso <sup>[15]</sup>
- Truviso's profile on DEMO.com <sup>[16]</sup>
- Continuous Analytics white paper <sup>[17]</sup>
- Real-time data streams and the web <sup>[18]</sup>

## See also

- Streaming Media List of streaming media companies
- Event Stream Processing (ESP) is technology that focuses on processing streams of related data.
- Event Driven Architecture (EDA) software architecture pattern promoting the production, detection, consumption of, and reaction to events.
- Real-time business intelligence Application of business intelligence to dynamic, real-time business processes
- Real-time computing CEP systems are typically real-time systems
- Operational Intelligence Both CEP and ESP are technologies that underpin operational intelligence.

## References

- [1] Truviso: New tricks with old SQL ([http://www.infoworld.com/article/07/05/26/moes-amalgamated\\_1.html](http://www.infoworld.com/article/07/05/26/moes-amalgamated_1.html))
- [2] Michael Franklin's bio (<http://www.cs.berkeley.edu/~franklin/>)
- [3] Sailesh Krishnamurthy's bio (<http://sailesh.googlepages.com/>)
- [4] The Telegraph Project at UC Berkeley (<http://telegraph.cs.berkeley.edu/>)
- [5] PostgreSQL Application Software Catalogue (<http://www.postgresql.org/download/products/7.html>)
- [6] Truviso Offers Another Option For Business Intelligence ([http://www.informationweek.com/news/business\\_intelligence/analytics/showArticle.jhtml?articleID=207404020](http://www.informationweek.com/news/business_intelligence/analytics/showArticle.jhtml?articleID=207404020))
- [7] Truviso: shaking up the Streaming market ([http://www.it-director.com/blogs/Robin\\_Bloor/2007/6/TruViso\\_shaking\\_up\\_the\\_Streaming\\_market.html](http://www.it-director.com/blogs/Robin_Bloor/2007/6/TruViso_shaking_up_the_Streaming_market.html))
- [8] Truviso shows off dynamic database with Technorati tag cloud (<http://venturebeat.com/2008/05/28/truviso-shows-of-dynamic-database-with-technorati-tag-cloud>)
- [9] ONSET Ventures Portfolio: Applications and Infrastructure Software ([http://www.onsetventures.com/portfolio/portfolio\\_software.html](http://www.onsetventures.com/portfolio/portfolio_software.html))
- [10] Diamondhead Ventures Portfolio Companies (<http://www.dhven.com/portfolio/companies.html>)
- [11] UPS Strategic Enterprise Fund Portfolio Companies (<http://www.ups.com/sef/portfolio.html>)
- [12] Truviso Enters the High-End Web Analytics Market with Visitor Insight & Analytics Software (<http://www.b-eye-network.com/view/13614>)
- [13] <http://www.truviso.com>
- [14] <http://www.tdwi.org/News/display.aspx?ID=9334>
- [15] <http://jtonedm.com/2009/03/03/first-look-truviso/>
- [16] <http://www.demo.com/community/?q=node/172191>
- [17] [http://neilconway.org/docs/cidr2009\\_franklin.pdf](http://neilconway.org/docs/cidr2009_franklin.pdf)
- [18] <http://gigaom.com/2009/07/12/big-data-and-real-time-web-a-confluence-of-streams/>

# User-defined function

A **User-Defined Function**, or **UDF**, is a function provided by the user of a program or environment, in a context where the usual assumption is that functions are built into the program or environment.

## BASIC language

In some old implementations of the BASIC programming language, user defined functions are defined using the "DEF FN" syntax. More modern dialects of BASIC are influenced by the structured programming paradigm, where most or all code is written as user defined functions or procedures, and the concept becomes practically redundant.

## Databases

In SQL databases, a user-defined function provides a mechanism for extending the functionality of the database server by adding a function that can be evaluated in SQL statements. The SQL standard distinguishes between scalar and table functions. A scalar function returns only a single value (or NULL), whereas a table function returns a (relational) table comprising zero or more rows, each row with one or more columns.

User-defined functions in SQL are declared using the CREATE FUNCTION statement. For example, a function that converts Celsius to Fahrenheit might be declared like this:

```
CREATE FUNCTION dbo.CtoF(Celsius FLOAT)
  RETURNS FLOAT
  RETURN (Celsius * 1.8) + 32
```

Once created, a user-defined function may be used in expressions in SQL statements. For example, it can be invoked where most other intrinsic functions are allowed. This also includes SELECT statements, where the function can be used against data stored in tables in the database. Conceptually, the function is evaluated once per row in such usage. For example, assume a table named ELEMENTS, with a row for each known chemical element. The table has a column named BoilingPoint for the boiling point of that element, in Celsius. This query:

```
SELECT Name, CtoF(BoilingPoint)
FROM Elements
```

would retrieve the name and the boiling point from each row. It invokes the CtoF user-defined function as declared above in order to convert the value in the column to a value in Fahrenheit.

Each user-defined function carries certain properties or characteristics. The SQL standard defines the following properties:

- **language** - defines the programming language in which the user-defined function is implemented; examples are SQL, C, or Java.
- **parameter style** - defines the conventions that are used to pass the function parameters and results between the implementation of the function and the database system (only applicable if language is not SQL).
- **specific name** - a name for the function that is unique within the database. Note that the function name does not have to be unique, considering overloaded functions. Some SQL implementations, such as Microsoft SQL Server, require that function names are unique within a database, and overloaded functions are not allowed.
- **determinism** - specifies whether the function is deterministic or not. The determinism characteristic has an influence on the query optimizer when compiling a SQL statement.
- **SQL-data access** - tells the database management system whether the function contains no SQL statements (NO SQL), contains SQL statements but does not access any tables or views (CONTAINS SQL), reads data from tables or views (READS SQL DATA), or actually modifies data in the database (MODIFIES SQL DATA).

User-defined functions should not be confused with stored procedures. Stored procedures allow the user to group a set of SQL commands. A procedure can accept parameters and execute its SQL statements depending on those parameters. A procedure is not an expression and, thus, cannot be used like user-defined functions.

Some database management systems allow the creation of user defined functions in languages other than SQL. Microsoft SQL Server, for example, allows the user to use .NET languages including C# for this purpose. DB2 and Oracle support user-defined functions written in C or Java programming languages.

## SQL Server 2000

There are three types of UDF in Microsoft SQL Server 2000:

**#Scalar functions.**

**#Inline table-valued functions.**

**#Multistatement table-valued functions.**

Scalar functions return a single data value (not a table) with RETURNS clause. Scalar functions can use all scalar data types, with exception of timestamp and user-defined data types. Inline table-valued functions return the result set of a single SELECT statement. Multistatement table-valued functions return a table, which was built with many TRANSACT-SQL statements.

User-defined functions can be invoked from a query like built-in functions such as OBJECT\_ID, LEN, DATEDIFF, or can be executed through an EXECUTE statement like stored procedures.

Performance Notes:

1. On Microsoft SQL Server 2000 a table-valued function which 'wraps' a View may be much faster than the View itself. The following MyFunction is an example of a 'function-wrapper' which runs faster than the underlying view MyView:

```
CREATE FUNCTION MyFunction ()
RETURNS @Tbl TABLE
(
  StudentID VARCHAR(255),
  SAS_StudentInstancesID INT,
  Label VARCHAR(255),
  Value MONEY,
  CMN_PersonsID INT
)
AS

BEGIN

INSERT @Tbl
(
  StudentID ,
  SAS_StudentInstancesID ,
  Label ,
  Value ,
  CMN_PersonsID
) SELECT
StudentID ,
```

```

SAS_StudentInstancesID ,
Label ,
Value ,
CMN_PersonsID
FROM MyView -- where MyView selects (with joins) the same columns from large table(s)

RETURN

END

```

2. On Microsoft SQL Server 2005 the result of the same code execution is the opposite: view is executed faster than the 'function-wrapper'.

User-defined functions are subroutines made of one or more Transact-SQL statements that can be used to encapsulate code for reuse. It takes zero or more arguments and evaluates a return value. Has both control-flow and DML statements in its body similar to stored procedures. Does not allow changes to any Global Session State, like modifications to database or external resource, such as a file or a network. Does not support output parameter. DEFAULT keyword must be specified to pass the default value of parameter. Errors in UDF cause UDF to abort which, in turn, aborts the statement that invoked the UDF.

```

CREATE FUNCTION CubicVolume
-- Input dimensions in centimeters
    (@CubeLength decimal(4,1),
     @CubeWidth decimal(4,1) ,
     @CubeHeight decimal(4,1) )
RETURNS decimal(12,3)
AS
BEGIN
    RETURN ( @CubeLength * @CubeWidth * @CubeHeight )
END

```

Data type supported in Microsoft SQL Server 2000 Like a temporary table used to store results Mostly used to define temporary variable of type (table) and the return value of a UDF The scope is limited to function, stored procedure, or batch in which it is defined Assignment operation is not allowed between (Table) variables May be used in SELECT, INSERT, UPDATE, and DELETE CREATE FUNCTION to create UDF ALTER FUNCTION to change the characteristics of UDF DROP FUNCTION to remove UDF

## External links

- Microsoft SQL Server reference for CREATE FUNCTION <sup>[1]</sup>
- MySQL manual section on UDFs <sup>[2]</sup>
- DB2 CREATE FUNCTION statement <sup>[3]</sup>
- Using user defined functions written in SQL in Sybase ASE <sup>[4]</sup>
- Using user defined functions written in Java in Sybase ASE <sup>[5]</sup>

## References

- [1] [http://msdn2.microsoft.com/en-us/library/aa258261\(SQL.80\).aspx](http://msdn2.microsoft.com/en-us/library/aa258261(SQL.80).aspx)
- [2] <http://dev.mysql.com/doc/refman/5.0/en/adding-functions.html>
- [3] <http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0000917.htm>
- [4] [http://www.petersap.nl/SybaseWiki/index.php/User\\_defined\\_SQL\\_functions](http://www.petersap.nl/SybaseWiki/index.php/User_defined_SQL_functions)
- [5] [http://www.petersap.nl/SybaseWiki/index.php/User\\_defined\\_Java\\_functions](http://www.petersap.nl/SybaseWiki/index.php/User_defined_Java_functions)

## varchar

---

A **varchar** or **Variable Character Field** is a set of character data of indeterminate length. The term **varchar** specifically refers to a data type of a field (or column) in a database management system. Varchar fields can be of any size up to the limit. The limit differs from types of databases, an Oracle 9i Database has a limit of 4000 bytes, a MySQL Database has a limit of 65,535 bytes (for the entire row) and Microsoft SQL Server 2005 8000 bytes (unless varchar(max) is used, which has a maximum storage capacity of 2,147,483,648 bytes).

**varchar** data types are popular amongst all database management systems (DBMS). This is mostly because, unlike the fixed-size char data-type, **varchar** does not store any blank characters, reducing the size of a database when the full length of the field is not used, although the length of the used size is stored, adding a small overhead.

## References

- CHAR and VARCHAR documentation <sup>[1]</sup> for Microsoft SQL Server 2008.
- VARCHAR documentation <sup>[2]</sup> for Apache Derby.
- CHAR and VARCHAR documentation <sup>[3]</sup> for MySQL 5.1 <sup>[4]</sup>.
- CHAR and VARCHAR documentation <sup>[5]</sup> for IBM\_Informix.

## References

- [1] <http://msdn.microsoft.com/en-us/library/ms176089.aspx>
  - [2] <http://db.apache.org/derby/docs/dev/ref/>
  - [3] <http://dev.mysql.com/doc/refman/5.1/en/char.html>
  - [4] <http://dev.mysql.com/>
  - [5] <http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.sqls.doc/sqls984.htm>
-



# View (database)

---

In database theory, a **view** consists of a stored query accessible as a virtual table composed of the result set of a query. Unlike ordinary tables (base tables) in a relational database, a view does not form part of the physical schema: it is a dynamic, virtual table computed or collated from data in the database. Changing the data in a table alters the data shown in subsequent invocations of the view.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table
- Views can join and simplify multiple tables into a single virtual table
- Views can act as aggregated tables, where the database engine aggregates data (sum, average etc) and presents the calculated results as part of the data
- Views can hide the complexity of data; for example a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data it presents
- Depending on the SQL engine used, views can provide extra security
- Views can limit the degree of exposure of a table or tables to the outer world

Just as functions (in programming) can provide abstraction, so database users can create abstraction by using views. In another parallel with functions, database users can manipulate nested views, thus one view can aggregate data from other views. Without the use of views the normalization of databases above second normal form would become much more difficult. Views can make it easier to create lossless join decomposition.

Just as rows in a base table lack any defined ordering, rows available through a view do not appear with any default sorting. A view is a relational table, and the relational model defines a table as a set of rows. Since sets are not ordered - by definition - the rows in a view are not ordered, either. Therefore, an ORDER BY clause in the view definition is meaningless. The SQL standard (SQL:2003) does not allow an ORDER BY clause in a subselect in a CREATE VIEW statement, just as it is not allowed in a CREATE TABLE statement. However, sorted data can be obtained from a view, in the same way as any other table - as part of a query statement. Nevertheless, some DBMS (such as Oracle and SQL Server) allow a view to be created with an ORDER BY clause in a subquery, affecting how data is displayed.

## Read-only vs. updatable views

Database practitioners can define views as read-only or updatable. If the database system can determine the reverse mapping from the view schema to the schema of the underlying base tables, then the view is updatable. INSERT, UPDATE, and DELETE operations can be performed on updatable views. Read-only views do not support such operations because the DBMS cannot map the changes to the underlying base tables. A view update is done by key preservation.

Some systems support the definition of INSTEAD OF triggers on views. This technique allows the definition of other logic for execution in place of an insert, update, or delete operation on the views. Thus database systems can implement data modifications based on read-only views. However, an INSTEAD OF trigger does not change the read-only or updatable property of the view itself.

---

## Advanced view features

Various database management systems have extended the views from read-only subsets of data.

The Oracle database introduced the concept of materialized views: pre-executed, non-virtual views commonly used in data warehousing. They give a static snapshot of the data and may include data from remote sources. The accuracy of a materialized view depends on the frequency or trigger mechanisms behind its updates. DB2 provides so-called "materialized query tables" (MQTs) for the same purpose. Microsoft SQL Server introduced in its 2000 version indexed views which only store a separate index from the table, but not the entire data.

## Equivalence

A view is equivalent to its source query. When queries are run against views, the query is modified. For example, if there exists a view named `Accounts_view` with the content as follows:

`accounts view:`

```
-----
SELECT name,
       money_received,
       money_sent,
       (money_received - money_sent) AS balance,
       address,
       ...
FROM table_customers c
JOIN accounts_table a
  ON a.customerid = c.customer_id
```

then the application could run a simple query such as:

`Sample query`

```
-----
SELECT name,
       balance
FROM accounts_view
```

The RDBMS then takes the simple query, replaces the equivalent view, then sends the following to the optimiser:

`Preprocessed query:`

```
-----
SELECT name,
       balance
FROM (SELECT name,
            money_received,
            money_sent,
            (money_received - money_sent) AS balance,
            address,
            ...
      FROM table_customers c JOIN accounts_table a
        ON a.customerid = c.customer_id
      )
```

From this point on the optimizer takes the query, removes unnecessary complexity (for example: it is not necessary to read the address, since the parent invocation does not make use of it) and then sends the query to the SQL engine

for processing.

## External links

- Views in Microsoft SQL Server 2005 <sup>[1]</sup>
- Views in MySQL <sup>[2]</sup>
- Views in PostgreSQL <sup>[3]</sup>
- Views in SQLite <sup>[4]</sup>
- Views in Oracle <sup>[5]</sup>
- Materialized Views in Oracle <sup>[6]</sup>

## References

- [1] <http://msdn.microsoft.com/en-us/library/ms187956.aspx>  
[2] <http://dev.mysql.com/doc/refman/5.1/en/views.html>  
[3] <http://www.postgresql.org/docs/current/interactive/tutorial-views.html>  
[4] [http://www.sqlite.org/lang\\_createview.html](http://www.sqlite.org/lang_createview.html)  
[5] [http://download.oracle.com/docs/cd/E11882\\_01/server.112/e10713/schemaob.htm#CNCPT311](http://download.oracle.com/docs/cd/E11882_01/server.112/e10713/schemaob.htm#CNCPT311)  
[6] [http://download.oracle.com/docs/cd/E11882\\_01/server.112/e10713/schemaob.htm#CNCPT411](http://download.oracle.com/docs/cd/E11882_01/server.112/e10713/schemaob.htm#CNCPT411)

# WQL

**Windows Management Instrumentation Query Language** (WQL) is Microsoft's implementation of the CIM Query Language (CQL), a query language for the Common Information Model standard from the Distributed Management Task Force (DMTF). It is a subset of the standard ANSI SQL with minor semantic changes.<sup>[1]</sup> A basic WQL query remains fairly understandable for people with basic SQL knowledge.

WQL is dedicated to WMI and is designed to perform queries against the CIM repository to retrieve information or get event notifications.

## Example

As an example, the following WQL query selects all the drives on a computer that have less than 2 MB of free space.<sup>[2]</sup>

```
SELECT * FROM Win32_LogicalDisk WHERE FreeSpace < 2097152
```

## See also

- Windows Management Instrumentation
- Common Information Model
- Web-Based Enterprise Management
- Windows PowerShell

## External links

- Querying with WQL <sup>[3]</sup>
- WQL Operators <sup>[4]</sup>
- WQL-Supported Date Formats <sup>[5]</sup>
- WQL-Supported Time Formats <sup>[6]</sup>
- WQL (SQL for WMI) <sup>[7]</sup>
- Using WQL with the WMI Provider for Server Events <sup>[8]</sup>
- WMI Queries <sup>[9]</sup>

## References

- [1] WQL (SQL for WMI) (<http://msdn2.microsoft.com/en-us/library/aa394606.aspx>)
- [2] WMI Queries ([http://msdn2.microsoft.com/en-us/library/ms186146\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/ms186146(VS.80).aspx))
- [3] <http://msdn2.microsoft.com/en-us/library/aa392902.aspx>
- [4] <http://msdn2.microsoft.com/en-us/library/aa394605.aspx>
- [5] <http://msdn2.microsoft.com/en-us/library/aa394607.aspx>
- [6] <http://msdn2.microsoft.com/en-us/library/aa394608.aspx>
- [7] <http://msdn2.microsoft.com/en-us/library/aa394606.aspx>
- [8] <http://msdn2.microsoft.com/en-us/library/ms180524.aspx>
- [9] [http://msdn2.microsoft.com/en-us/library/ms186146\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/ms186146(VS.80).aspx)

# Windows Internal Database

**Windows Internal Database** (codenamed WYukon, sometimes referred to as SQL Server Embedded Edition) is a variant of SQL Server Express 2005 that is included with Windows Server 2008, and is included with other free Microsoft products released after 2007 that require an SQL Server database backend. Windows SharePoint Services 3.0 and Windows Server Update Services 3.0 both include Windows Internal Database, which can be used as an alternative to using a retail edition of SQL Server. WID is a 32-bit application, even as component of Windows Server 2008 64-bit, which installs in the path C:\Program Files (x86)\Microsoft SQL Server.

Windows Internal Database is not available as a standalone product for use by end-user applications; Microsoft provides SQL Server Express and Microsoft SQL Server for this purpose. Additionally, it is designed to only be accessible to Windows Services running on the same machine.

Several components of Windows Server 2008 use Windows Internal Database for their data storage: Active Directory Rights Management Services, Windows System Resource Manager, UDDI Services, and Windows SharePoint Services. On Windows Server 2003, SharePoint and Windows Server Update Services will install Windows Internal Database and use it as a default data store if a retail SQL Server database instance is not provided. A Knowledge Base article published by Microsoft states that Windows Internal Database does not identify itself as a removable component, and provides instructions how it may be uninstalled by calling Windows Installer directly.<sup>[1]</sup>

SQL Server Management Studio Express can be used to connect to an instance of Windows Internal Database using `\\.\pipe\MSSQL$MICROSOFT##SSEE\sql\query` as instance name. But this will only work locally, as Remote Connections cannot be enabled for this edition of SQL Server.

## External links

- Planning and Architecture for Windows SharePoint Services 3.0 Technology <sup>[2]</sup>
- Release Notes for Microsoft Windows Server Update Services 3.0 <sup>[3]</sup>
- <http://www.mssqltips.com/tip.asp?tip=1577> <sup>[4]</sup>

## References

- [1] "Windows Internal Database is not listed in the Add or Remove Programs tool and is not removed when you remove Windows SharePoint Services 3.0 from the computer (MSKB920277)" (<http://support.microsoft.com/kb/920277>). *Knowledge Base*. Microsoft. .
- [2] <http://go.microsoft.com/fwlink/?LinkId=79600>
- [3] <http://go.microsoft.com/fwlink/?LinkId=71220>
- [4] <http://www.mssqltips.com/tip.asp?tip=1577>

# XLeratorDB

---



**XLeratorDB** is a suite of database function libraries that enable Microsoft SQL Server to perform a wide range of additional (non-native) business intelligence and ad hoc analytics. The libraries, which are embedded and run centrally on the database, include more than 300 individual functions similar to those found in Microsoft Excel spreadsheets. The individual functions are grouped and sold as six separate libraries based on usage: finance, statistics, math, engineering, unit conversions and strings. WestClinTech, the company that developed **XLeratorDB**, claims it is "the first commercial function package add-in for Microsoft SQL Server."<sup>[1]</sup>

## Company history

WestClinTech (LLC), founded by software industry veterans Charles Flock and Joe Stampf in 2008, is located in Irvington, New York, USA. Flock was a co-founder of The Frustum Group, developer of the OPICS enterprise banking and trading platform, which was acquired by London-based Misys, PLC in 1996.<sup>[2]</sup> Stampf joined Frustum in 1994 and with Flock remained active with the company after acquisition, helping to develop successive generations of OPICS now employed by over 150 leading financial institutions worldwide.<sup>[3]</sup>

Following a full year of research, development and testing, WestClinTech introduced and recorded its first commercial sale of **XLeratorDB** in April 2009.<sup>[4]</sup> <sup>[5]</sup> In September 2009, **XLeratorDB** became available to all Federal agencies through NASA's Strategic Enterprise-Wide Procurement (SEWP-IV) program, a government-wide acquisition contract.<sup>[6]</sup>

## Technology

**XLeratorDB** uses Microsoft SQL CLR(Common Language Runtime) technology.<sup>[7]</sup> SQL CLR allows managed code to be hosted by, and run in, the Microsoft SQL Server environment. SQL CLR relies on the creation, deployment and registration of .NET Framework assemblies that are physically stored in managed code dynamic-link libraries (DLL). The assemblies may contain .NET namespaces, classes, functions, and properties. Because managed code compiles to native code prior to execution, functions using SQL CLR can achieve significant performance increases versus the equivalent functions written in T-SQL in some scenarios.<sup>[8]</sup>

**XLeratorDB** requires Microsoft SQL Server 2005 or SQL Server 2005 Express editions, or later (compatibility mode 90 or higher).<sup>[9]</sup> The product installs with PERMISSION\_SET=SAFE. SAFE mode, the most restrictive permission set, is accessible by all users. Code executed by an assembly with SAFE permissions cannot access external system resources such as files, the network, the internet, environment variables, or the registry.<sup>[10]</sup>

## Functions

In computer science, a function is a portion of code within a larger program which performs a specific task and is relatively independent of the remaining code. As used in database and spreadsheet applications these functions generally represent mathematical formulas widely used across a variety of fields. While this code may be user-generated, it is also embedded as a pre-written sub-routine in applications. These functions are typically identified by common nomenclature which corresponds to their underlying operations: e.g. **IRR** identifies the function which calculates Internal Rate of Return on a series of periodic cash flows.

### Function uses

As sub-routines functions can be integrated and used in a variety of ways, and in a wide variety of larger, more complicated applications. Within large enterprise applications they may, for example, play an important role in defining business rules or risk management parameters, while remaining virtually undetected by end users. Within database management systems and spreadsheets, however, these kinds of functions also represent discrete sets of tools; they can be accessed directly and utilized on a stand-alone basis, or in more complex, user-defined configurations. In this context, functions can be used for business intelligence and ad hoc analysis of data in fields such as finance, statistics, engineering, math, etc.

### Function types

**XLeratorDB** uses three kinds of functions to perform analytic operations: scalar, aggregate, and a hybrid form which WestClinTech calls **Range Queries**. Scalar functions take a single value, perform an operation and return a single value.<sup>[11]</sup> An example of this type of function is **LOG**, which returns the logarithm of a number to a specified base.<sup>[12]</sup> Aggregate functions operate on a series of values but return a single, summarizing value. An example of this type of function is **AVG**, which returns the average of values in a specified group.<sup>[13]</sup>

In **XLeratorDB** there are some functions which have characteristics of aggregate functions (operating on multiple series of values) but cannot be processed in SQL CLR using single column inputs, such as **AVG** does. For example, irregular internal rate of return (**XIRR**), a financial function, operates on a collection of cash flow values from one column, but must also apply variable period lengths from another column and an initial iterative assumption from a third, in order to return a single, summarizing value. WestClinTech documentation notes that **Range Queries** specify the data to be included in the result set of the function independently of the WHERE clause associated with the T-SQL statement, by incorporating a SELECT statement into the function as a string argument; the function then traps that SELECT statement, executes it internally and processes the result.<sup>[14]</sup>

Some **XLeratorDB** functions that employ **Range Queries** are: **NPV**, **XNPV**, **IRR**, **XIRR**, **MIRR**, **MULTINOMIAL**, and **SERIESSUM**. Within the application these functions are identified by a "\_q" naming convention: e.g. **NPV\_q**, **IRR\_q**, etc.<sup>[15]</sup>

## Analytic functions

### SQL Server functions

Microsoft SQL Server is the #3 selling database management system (DBMS), behind Oracle and IBM.<sup>[16]</sup> (While versions of SQL Server have been on the market since 1987<sup>[17]</sup>, **XLeratorDB** is compatible with only the 2005 edition and later.) Like all major DBMS, SQL Server performs a variety of data mining operations by returning or arraying data in different views (also known as drill-down). In addition, SQL Server uses Transact-SQL (T-SQL)<sup>[18]</sup> to execute four major classes of pre-defined functions in native mode.<sup>[19]</sup> Functions operating on the DBMS offer several advantages over client layer applications like Excel: they utilize the most up-to-date data available; they can process far larger quantities of data; and, the data is not subject to exporting and transcription errors<sup>[20]</sup>.

SQL Server 2008 includes a total of 58 functions that perform relatively basic aggregation (12), math (23) and string manipulation (23) operations useful for analytics; it includes no native functions that perform more complex operations directly related to finance, statistics or engineering.<sup>[21]</sup>

### Excel functions

Microsoft Excel, a component of Microsoft Office suite, is one of the most widely used spreadsheet applications on the market today.<sup>[22]</sup> In addition to its inherent utility as a stand-alone desktop application, Excel overlaps and complements the functionality of DBMS in several ways: storing and arraying data in rows and columns; performing certain basic tasks such as pivot table<sup>[23]</sup> and aggregating values; and facilitating sharing, importing and exporting of database data. Excel's chief limitation relative to a true database is capacity; Excel 2003 is limited to some 65k rows and 256 columns; Excel 2007 extends this capacity to roughly 1million rows and 16k columns.<sup>[24]</sup> By comparison, SQL Server is able to manage over 500k terabytes of memory.<sup>[25]</sup>

Excel offers, however, an extensive library of specialized pre-written functions which are useful for performing ad hoc analysis on database data. Excel 2007 includes over 300 of these pre-defined functions, although customized functions can also be created by users, or imported from third party developers as add-ons. Excel functions are grouped by type:<sup>[26]</sup>

#### Excel Functions

Financial	Statistical	Engineering	Math and trig
Information	Date and time	Text and data	Logical
Add-ins and automation	Lookup and reference	Cube	Database and list management

### Excel business intelligence functions

Operating on the client computing layer Excel plays an important role as a business intelligence tool<sup>[27]</sup> because it:

- performs a wide array of complex analytic functions not native to most DBMS software
- offers far greater ad hoc reporting and analytic flexibility than most enterprise software
- provides a medium for sharing and collaborating because of its ubiquity throughout the enterprise

Microsoft reinforces this positioning with Business Intelligence documentation that positions Excel in a clearly pivotal role.<sup>[28]</sup>

## XLeratorDB vs. Excel functions

While operating within the database environment, **XLeratorDB** functions utilize the same naming conventions and input formats, and in most cases, return the same calculation results as Excel functions.<sup>[29]</sup> **XLeratorDB**, coupled with SQL Server's native capabilities, compares to Excel's function sets as follows:

Excel 2007		XLeratorDB + SQL Server			
Function Type	Total	Total	Match	New	Native
Financial	52	63	50	13	0
Statistics	83	131	65	54	12
Math	59	73	34	16	23
Engineering	39	42	38	4	0
Conversions*	49	77	0	77	0
Strings	26	58	11	24	23
*Microsoft includes these functions within Engineering using variable input configurations					

## External links

- XLeratorDB website<sup>[30]</sup>
- Microsoft SQL Server Documentation<sup>[31]</sup>
- Microsoft Excel Documentation<sup>[32]</sup>

## References

- [1] <http://WestClinTech.com/Home/tabid/36/Default.aspx>
- [2] <http://www.fundinguniverse.com/company-histories/Misys-PLC-Company-History.html>
- [3] [http://www.misys.com/tcm/markets/otc\\_derivative\\_trading\\_solutions-opics\\_plus.html](http://www.misys.com/tcm/markets/otc_derivative_trading_solutions-opics_plus.html) [4] <http://www.sqlmag.com/Articles/ArticleID/102126/102126.html?Ad=1>
- [5] <http://WestClinTech.com/CaseStudyNewlandCommunities/tabid/178/Default.aspx>
- [6] <http://www.sewp.nasa.gov/index.shtml>
- [7] [http://WestClinTech.com/wct\\_local/Documentation/AboutXLeratorDB/tabid/148/topic/Technology/Default.aspx](http://WestClinTech.com/wct_local/Documentation/AboutXLeratorDB/tabid/148/topic/Technology/Default.aspx)
- [8] This needs a reference
- [9] [http://WestClinTech.com/Support/FAQ/tabid/143/Default.aspx#PL\\_A1\\_RequiredToInstall](http://WestClinTech.com/Support/FAQ/tabid/143/Default.aspx#PL_A1_RequiredToInstall)
- [10] See SQL Server DBMS documentation at: <http://msdn.microsoft.com/en-us/library/ms130214.aspx>
- [11] Also see: <http://msdn.microsoft.com/en-us/library/ms174318.aspx>
- [12] Excel definition at: <http://office.microsoft.com/en-us/excel/HP100624401033.aspx>
- [13] SQL Server definition at: <http://msdn.microsoft.com/en-us/library/ms177677.aspx>
- [14] [http://WestClinTech.com/wct\\_local/Documentation/AboutXLeratorDB/tabid/148/topic/Range%20Queries/Default.aspx](http://WestClinTech.com/wct_local/Documentation/AboutXLeratorDB/tabid/148/topic/Range%20Queries/Default.aspx)
- [15] See XLeratorDB Function Packages information box, above
- [16] IDC COMPETITIVE ANALYSIS: Worldwide Relational Database Management Systems 2007 Vendor Shares, Carl W. Olofson, June 2008, IDC #212840, Volume: 1, Tab: Markets
- [17] <http://e-articles.info/e/a/title/A-Brief-History-of-Microsoft-SQL-Server/>
- [18] <http://msdn.microsoft.com/en-us/library/ms166026%28SQL.90%29.aspx>
- [19] <http://msdn.microsoft.com/en-us/library/ms174318.aspx>
- [20] IBM refers to this as "no-paste analytics. See Data Warehousing documentation.
- [21] <http://msdn.microsoft.com/en-us/library/ms174318%28v=SQL.100%29.aspx>
- [22] Various sources suggest Office/Excel market share exceeds 90%, but this needs a specific source/citation
- [23] [http://searchsqlserver.techtarget.com/sDefinition/0,,sid87\\_gci875976,00.html](http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci875976,00.html)
- [24] See Excel documentation
- [25] <http://msdn.microsoft.com/en-us/library/ms143432.aspx>
- [26] <http://office.microsoft.com/en-us/excel/HA102775241033.aspx>



- [27] See general Business Intelligence documentation Microsoft and IBM, for example: [http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/cognos/pdfs/factsheets/fs\\_cognos8bi\\_analysis\\_for\\_microsoft\\_excel.pdf](http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/cognos/pdfs/factsheets/fs_cognos8bi_analysis_for_microsoft_excel.pdf)
- [28] <http://www.microsoft.com/presspass/newsroom/office/BusinessIntelligenceFS.msp>
- [29] <http://www.WestClinTech.com/Blog/tabid/132/EntryID/19/Default.aspx>
- [30] <http://www.WestClinTech.com/>
- [31] <http://www.microsoft.com/sqlserver/2008/en/us/default.aspx>
- [32] <http://office.microsoft.com/en-us/excel/default.aspx>

# XSQL

---

**XSQL** combines the power of XML and SQL to provide a language and database independent means to store and retrieve SQL queries and their results.

## Description

XSQL is the combination of XML (Extensible Markup Language) and SQL (Structured Query Language) to provide a language and database independent means for storing SQL queries, clauses and query results. XSQL development is still in its infancy and welcomes suggestions for improvement (especially in the form of patches).

Currently, the XSQL project has a DTD (Document Type Definition) to define the structure of an XSQL document and researchers are currently working on modifying the XML Generator, DBI Perl module to be able to parse XSQL documents and provide a tree- and event-based API (Application Programming Interface) to their elements. These modifications are being submitted as patches to the modules maintainer, Matt Sergeant. Thus, the source code does not live at this site.

It is hoped that XSQL will provide an end-to-end solution for handling SQL in Perl (other languages can be supported if there is interest). Creating XSQL implementations in other languages will allow all databases to support XML without having to alter the database source code in any way. The XSQL implementations can take care of turning XSQL in SQL and turning results into XSQL.

## External links

- [XSQL project website](#) <sup>[1]</sup>

## References

- [1] <http://xsql.sourceforge.net/>
-

# Article Sources and Contributors

**SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=368422026> *Contributors:* 62.253.64.xxx, 64.168.29.xxx, A.R., AFriedman, Aaron Brenneman, Aarthib123, ActiveSelective, Adashiel, Admrboltz, Adrian.walker, AgentCDE, Ahoerstemeier, Aidan W, Ajcumming, Aka042, Alai, Alalia 17, Alansohn, Alexius08, AllanManangan, Allstarecho, Alon, Alpha 4615, Alterego, Alvin-cs, AnandKumria, Anclation, Andri3w, Andre Engels, Andrisi, Angela, Anoko moonlight, Aou, Apiency, Arcann, Aresgunther, Arvi, Aseld, Ashley Y, Asix, Avillia, Avé, Az1568, B15nes7, BL, Badseed, Baiji, Barefootguru, Basil.bourque, Beetstra, Behringerdj, Ben D., Ben-Zin, BenFrantzDale, Beno1000, Bernd vdB, Bevo, Bill Huffman, Bkwillwm, BlindEagle, Blonkm, Bobdc, Bobstay, Bodmerb, Bonadea, Booles, Bovineone, Branko, Brassrat70s, Brick Thrower, Bryan Derksen, Bunnyhop11, Burgher, Butterscotch, CALR, CF84, CRGreathouse, Caltrop, CambridgeBayWeather, Camw, Can't sleep, clown will eat me, Canterbury Tail, Channard, Charles Matthews, Cherry Cotton, Chitirell, Cic, Closedmouth, Coffeehood, Cometstyles, Conversion script, Coolboy1234, Coremayo, Countersubject, Cpl Syx, Craig Stuntz, Cuchullain, Cuckoosnest, Cwolsfsheep, Cww, Cybercobra, D6, DH85868993, Da monster under your bed, Damian Yerrick, Dan Atkinson, DanBishop, DanP, Danakil, Dancter, Davep.org, DavidWikiuser, Dbtfz, Dcoetzee, Decon1985, Decoy, Denelson83, Dennis forbes, DennisWithem, Danelos, DigitalEnthusiast, Dittaeva, Dlohcierkim's sock, Dlyons493, Dmsar, Docu, DopefishJustin, Download, Dpr, Dreadstar, Dreftymac, Dgrussell, Dsan, Duncharris, Dysprosia, Dlugosz, EarthPerson, Eboyjr, Ed Poor, Edward, Egrabczewski, Ehajiyev, Ehheh, ElectricRay, Ellguy, Elkman, Eloquence, Elwikipedista, Emperorbma, Entoaggie09, Enviroboy, Epr123, Escape Orbit, Evildeathmath, Ezeu, Fastily, FatalError, Favonian, FelisLeo, Ferdna, Fleurydj, FrancoGG, Frانس2000, Frando, Frani, Frecklefoot, Fred Bradstadt, Furrykef, GHe, Gabrielsroka, Gaddium, Galoubet, Gawa123, Gennaro Prota, GhettoBlaster, Gil Gamesh, Gilliam, Gioto, GnuDoYng, Goethean, Goki, Gopher292, Goplat, Gosub, GraemeL, Graham87, GrandPoohBah, Grauenwolf, Greg Lindahl, GregCovey, Gregbard, GregorB, Ground Zero, Gzuckier, Hairy Dude, Hakkinen, Hankwang, Hardeeps, Hari, Harlock jds, Harperska, Haza-w, Herbythyme, Heron, Hirzel, Hongooi, Huzarus, Hydrargyrum, Iancarter, Iffcool, Incnis Mrsi, Inonit, Intr, Ioannes Pragensis, Iolar, J.delanoy, JCLately, JLaTondre, Jacek79, JamesNK, Jamesday, Jan Hidders, Jarash, Jay, Jaysweet, Jc35th, Jdlambert, Jeenware, Jeffg, Jeffrey Smith, Jerome Charles Potts, Jezmck, Jimbo Wales, Jimhark, Jimmytharpe, Joelm, Joeyjojo Junior, John Vandenberg, JohnCD, JohnOwens, Johnmc, Joieko, Jon Awbrey, Jonearles, Jonik, Jorge Stolfi, Jpk, Jsmethers, Jsrx, Julesd, Jusdafax, Just James, Justinkaz, Jwissick, Jzcool, Jzel6201, KD5TV1, Kaimiddleton, Karl-Henner, KeyStroke, Kingsleydehen, Kitch, Kizor, Klausness, Kop, Kovianyoy, Kpelt, Kreca, Kubra, Kuru, Kvdveer, Kwamikagami, KymFarnik, Lambiam, Lankiveil, Larsinio, Lavarock86, Leandrod, Lfstevens, Liao, Libcub, Littlebluenick, Logiphile, Lost.goblin, Lotje, Love1710, Loveenatayal, Lowellian, Ltruett, Lucian Sunday, LuoShengli, M7, Macaldo, Mackseem, Magnus Manske, Mahamoty, Mahanga, Mark Renier, Mark T, Master2841, MathiasRav, Mathmo, Matt.smart, MaxEnt, Maximaximax, Mdcchachi, Mdd, Meh222, MeltBanana, Mendel, Mentifisto, Merphand, Mersperio, Metroking, Michael Hardy, MichaelSpeer, Michele.alessandrini, Mikeblas, Mild Bill Hiccup, Mindmatrix, Minghong, MinorContributor, Miskin, Mitch Ames, MoRsE, Modster, Mlodaddy, MrOllie, Mxn, Mzajac, NYKevin, Neilc, Netfriend, Neum, NiceGuyEduardo, Nikola Smolenski, Nohat, Norm mit, Nricardo, Nsd, Nurg, Octahedron80, Od Mishehu, Oli Fiith, Olivier Debre, Oljtn, Ondrejkl, Opticyclic, Orzetto, Osmodiar, Paul Foxworthy, Paul Mackay, Paul.mcjones, Paulmieberman, Peruvianlama, Peter.vanroose, Philippe Perrault, Piano non troppo, PierreAbbat, Piet Delpont, Plcysys, Plrk, Plugwash, Pmcjones, Possum, Pouya, Power piglet, Powerlord, Pparazorbak, Prashanthns, Pukkie, Qu1j0t3, Quantum00, Quebec99, Quintote, RnB, RW Marloe, Racklever, RadioKirk, RainbowOfLight, Randomned, Ravi555, Razorx, Rdeleopn, Rdsmith4, RedWolf, Reedy, Reikon, RevRagnarok, Rfc1394, Rhobite, Ribaw1, Riffic, Rjwilmsi, Roadrunner, RobertG, RockMFR, Ronstone, Rp, Ruakh, Rursus, Ruud Koot, Rwww, S.figueiredo, S.Orvarr.S, SF007, Salix alba, SallyBoseman, Sam Korn, Sander Sade, SarekOfVulcan, Saric, Sbbv, Scanos, Schaeef, Schemtzm, Severa, Sgkay, Shizhao, Simetrical, SiobhanHansa, Sippin, Sjakalle, Sjc, SkyWalker, Slavingdog, Smartweb, Smeyerinnewton, Smu95ip, Somme111, SouthLake, Spectrum266, Sql pol, SqlPac, Sstrader, Starbeard, Starius, Stephenb, Steve Alvesteffer, Stevietheman, Stifle, Stratadrake, Subflux, SuezanneC Baskerville, SummerWithMorons, Svick, Swanboy2, Ta bu shi da yu, Tablizer, Tagtool, Tasc, Tcnv, Tcutcher, Tgeller, The Anome, The Letter J, The Thing That Should Not Be, Theone256, Thetorpedodog, Thirtyfootscrew, Three-quarter-ten, Thumperward, Thunderbritches, TigerShark, Tiha, TimTay, Timwi, Tizio, Tobias Bergemann, Todd Vierling, Tomtheeditor, Toreau, Traal, Traxer, Trevorlofin, Trlovejoy, Troels Arvin, Turnstep, Ucanlookitup, Ulf Abrahamsson, Ummit, Uncle G, Unforgettableid, Unknown W. Brackets, Variable, Ville Oikarinen, Vincenzo.romano, Viridity, Visor, Vitund, Vivek.pandey, Vivio Testarossa, VoluntarySlave, Vonfragnioff, Wadamja, Washburnmaw, Wei.cs, Weialawaga, Wernher, Wesley, Weylinp, Wikiklrsc, William Avery, Winchelsea, Wingnut909, Wiretse, Wisq, Wragge, Wulfila, XSTRIKEx6864, Xdenizen, Xibe, Xjhx001, Yeneesha, Yeng-Wang-Yeh, Yzchang, Zhenqinli, Zigger, Zundark, Ævar Arnfrjörð Bjarnason, 1241 anonymous edits

**SQL:2003** *Source:* <http://en.wikipedia.org/w/index.php?oldid=345521742> *Contributors:* Alastairbrown, Aureliengg, Avé, Bovineone, CRGreathouse, CYCC, Duster.Cleaner, Friendlydta, Galaxy07, JVz, Jeltz, John Vandenberg, Mariuz, Mark Renier, Pekaje, SqlPac, The Fortunate Unhappy, TheParanoidOne, Touko vk, 20 anonymous edits

**SQL:2008** *Source:* <http://en.wikipedia.org/w/index.php?oldid=345521773> *Contributors:* AllanManangan, Allstarecho, CRGreathouse, Dfetter, GregorB, Highpush76, Itai, The Fortunate Unhappy, Touko vk, Troels Arvin, 2 anonymous edits

**Advantage Database Server** *Source:* <http://en.wikipedia.org/w/index.php?oldid=367900775> *Contributors:* Cander0000, DMacks, Databaseguy, Doctorfluffy, Kbrumback, Mdd, 2 anonymous edits

**Apatar** *Source:* <http://en.wikipedia.org/w/index.php?oldid=363890312> *Contributors:* Cander0000, D6, Greenrd, Iness b, Murrayflynn, Pegship, Renatko, Shaakunthala, Tyrell perera, 7 anonymous edits

**Call Level Interface** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319780166> *Contributors:* Andy Dingley, BonsaiViking, Cyrius, Dataphile, David Gerard, David Wahler, Dysprosia, GhettoBlaster, Jamelan, John Vandenberg, Michael Hardy, Mindmatrix, Modster, Naddy, Nixdorf, Paul Foxworthy, R. S. Shaw, Spoileralerter, SqlPac, The Anome, Tin, Xypron, 2 anonymous edits

**Cardinality (SQL statements)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=348615513> *Contributors:* BostonRed, Closedmouth, DaBest1, DerekAsirvadem, Eric Burnett, GregorB, Little Mountain 5, Mark Renier, Mathman1550, Rjwilmsi, Thal3s, Troels Arvin, 3 anonymous edits

**Check Constraint** *Source:* <http://en.wikipedia.org/w/index.php?oldid=348299311> *Contributors:* Andreas Kaufmann, LegitimateAndEvenCompelling, MER-C, Quaker5567, Soluch, SqlPac, Stolze, Tizio, UberCryxik, Unordained, 5 anonymous edits

**Commit (data management)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=354467286> *Contributors:* Ajgorhoe, Amalas, Arashb31, Donovan Heinrich, G7yungbi, Gpvos, Gtrmp, JCLately, Jthiesen, Lmalecki, Mark Renier, Michael Slone, Mikeblas, Nbla, Radagast83, RedWolf, RockMFR, Schapel, SqlPac, Stevertigo, Yuvysingh, 23 anonymous edits

**Common table expressions** *Source:* <http://en.wikipedia.org/w/index.php?oldid=343722712> *Contributors:* Akimoto rika, Avonwyss, Dawynn, Dfetter, Enigmaman, Hertzprung, Labreuer, Pamar, 6 anonymous edits

**Condition (SQL)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=344600352> *Contributors:* ClaimJumperJohn, Deviloper, Dna-webmaster, Intelligentsium, Wwmbes, Xaje, 1 anonymous edits

**Correlated subquery** *Source:* <http://en.wikipedia.org/w/index.php?oldid=346539438> *Contributors:* Atendulk, CultureDrone, KathrynLybarger, 1 anonymous edits

**CUBRID** *Source:* <http://en.wikipedia.org/w/index.php?oldid=367901147> *Contributors:* Cander0000, CommonsDelinker, Cubridorg, Ficell, Kadishmal, PamD, Prozaker, Rich Farmbrough, Rpyle731, Tomtheeditor, 1 anonymous edits

**Cursor (databases)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=364764175> *Contributors:* Agujero Negro, Aitias, Alik Kirillovich, BlastOButter42, Catgut, Cwolsfsheep, Danielx, DarkFalls, Darth Panda, DigitalEnthusiast, Dmccreary, Doug Bell, Ejdzje, Epr123, Fieldday-sunday, Ilyanep, Jamestochter, Janigabor, Jerome Charles Potts, Larsinio, Mark Renier, Mikeblas, Mindmatrix, Miten.morakhia, MrOllie, Mwtows, Nagae, NawlinWiki, NeonMerlin, NewEnglandYanke, OsamaK, PhilHibbs, RHaworth, Reedy, Richi, S.K., S.Orvarr.S, Sachzn, SarekOfVulcan, SimonP, Sippin, SkyWalker, Stolze, Tbannist, Tobias Bergemann, Underpants, Wallie, Wjasongilmore, Wknight94, Xiphoris, 66 anonymous edits

**Data Control Language** *Source:* <http://en.wikipedia.org/w/index.php?oldid=368179895> *Contributors:* Alleborgo, Altenmann, Chupon, CodeNaked, Dfetter, GregorB, Gwernol, Isotope23, OMouse, Slipstream, SqlPac, The Thing That Should Not Be, 22 anonymous edits

**Data Definition Language** *Source:* <http://en.wikipedia.org/w/index.php?oldid=362006918> *Contributors:* 16@r, Academic Challenger, Aitias, Aleenf1, Anna Lincoln, Aymath2, BL, Babbage, Chupon, CodeNaked, Cometstyles, Cybersprocket, Decoy, Eags, Eliazar, Elonka, Emvee, Fresheneesz, Gcdinmore, Goplat, GregorB, Heimstern, Isotope23, JLaTondre, JoB, Juansempere, KeyStroke, Lectoran, Luna Santin, Mhkay, Mindmatrix, Modster, MrRadioGuy, Nate Silva, Riki, Santosjhd, Sigilbertz, Skew-t, SkyWalker, SqlPac, Svick, Tobias382, Trevorbjork, Tumble, Unyoyega, Voyagerfan5761, Vy0123, Wei&Nix, WikHead, Wwwwolf, Zanimum, 103 anonymous edits

**Data Manipulation Language** *Source:* <http://en.wikipedia.org/w/index.php?oldid=367649020> *Contributors:* Andy Dingley, Aneeshpu, Booyabazooka, CanisRufus, Chrisahn, Chupon, CodeNaked, Danlev, Discospinster, Frazzydee, Isotope23, Jay, Jcrided, KeyStroke, KnightRider, Minghong, Mr. Wheely Guy, Nevcao, OMouse, Shanes, SqlPac, Tavahom, Tony Fox, Viveksharma474, Vmenkov, 33 anonymous edits

**Database Console Commands (Transact-SQL)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=364652828> *Contributors:* Cander0000, EagleFan, Lilac Soul, Marek69, Ravichandar84, Rjwilmsi, Shadowjams

**DbForge Studio for MySQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=368370348> *Contributors:* Anas2048, Apparition11, CommonsDelinker, Devart, Doscmaker, Imnotminkus, JLaTondre, Oddharmonic, Oracleofottawa, Rich Farmbrough, The Thing That Should Not Be, 14 anonymous edits

**Declarative Referential Integrity** *Source:* <http://en.wikipedia.org/w/index.php?oldid=355601353> *Contributors:* Bertport, Bluemoose, DerekAsirvadem, Emersoni, PPOST, Remember the dot, Veatch, Wikiafreeman, 7 anonymous edits

**Devgems Data Modeler** *Source:* <http://en.wikipedia.org/w/index.php?oldid=349635328> *Contributors:* Beeblebrox, Dethlock99, LilHelpa, Wsandiego

**Embedded SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=358881581> *Contributors:* Chiragjayswal, Docu, EagleOne, Elonka, Fred Bradstadt, Ghetoblaster, HeavyStorm, Informationanalyst, Jwoodger, Krauss, Liao, Meskes, Mrzaus, Rwww, Shamatt, Shawn Y.H. Kim, Taketarou, Yurik, 15 anonymous edits

**EnterpriseDB** *Source:* <http://en.wikipedia.org/w/index.php?oldid=353341057> *Contributors:* Bovineone, Cander0000, Csabo, Ctkene, Deathphoenix, Hmains, Integr, JLaTondre, JamesBWatson, Jonah.harris, Liempt, Lmxspice, Mindmatrix, Mralston, Neilc, Nikkhils, Nisselua, Oberiko, Pgsnake, Postgressor01, Prequent, Rcsheets, Reactor, Riaanvn, Sappy, Staecker, Superm401, SvBiker, Technobadger, Tom harrison, Valrith, Xompanthy, Yurik, 36 anonymous edits

**Epictetus Database Client** *Source:* <http://en.wikipedia.org/w/index.php?oldid=366485474> *Contributors:* Frap, Maddcast, 13 anonymous edits

**Foreign key** *Source:* <http://en.wikipedia.org/w/index.php?oldid=367971096> *Contributors:* 16@r, Amix.pal, Arichnad, Arunsinghil, AutumnSnow, Beesforan, Biochemza, Brick Thrower, Can't sleep, clown will eat me, Cander0000, Cww, DHN, Darth Panda, Derek Balsam, DireWolf, Dobi, FatalError, Frap, Govorun, GregorB, Gsm1011, IanHarvey, JHunTerJ, Jadrman, Jlenthe, Joebeone, KeyStroke, Kf4bdy, Larsinio, Marcusfriedman, Mark Renier, Mikeblas, MikeyTheK, Mindmatrix, Mormegil, Natalie Erin, Obradovic Goran, PPOST, Pbwest, Peak, RIL-sv, Reedy, Rjwilmsi, Rror, Salvatore Ingala, Selfworm, Sempfer, Shreyasjoshis, Species8473, Staecker, Stolze, Svenmattijssen, Tarquin, The Thing That Should Not Be, Timhowardriley, TobiasPersson, Troels Arvin, Unordained, Waskyo, ZipzOp, 121 anonymous edits

**FSQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=325601208> *Contributors:* 1ForTheMoney, Amine Brikci N, Drilnoth, Mark Renier, PepeGalindo, The Thing That Should Not Be, 2 anonymous edits

**Hint (SQL)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=316315123> *Contributors:* JLaTondre, Miyagawa, Nikola Smolenski, Yms

**HSQLDB** *Source:* <http://en.wikipedia.org/w/index.php?oldid=368504845> *Contributors:* Betacommand, Bryan Derksen, Cakruege, Christian Storm, Cristan, Dougher, Elwikipedista, ErikWarmelin, Exceeder, Fchoong, Frap, Gokudo, Haakon, Indefatigable, JesseHogan, Jmoller99, Joehni, Jpkole, M4bwav, Mikeblas, Minghong, Neilc, Odoepner, Pavel Vozenilek, Pilotguy, Reedy, Rharnier, Rjwilmsi, SF007, Sam Hocevar, Skinrider, Sqliinfo, Stephenbez, TheParanoidOne, Turnstep, Two Bananas, Vlad, Zero0w, 46 anonymous edits

**Log shipping** *Source:* <http://en.wikipedia.org/w/index.php?oldid=325812110> *Contributors:* Bsoo, Cerebellum, Kubanczyk, Mrh30, Quality check, Roleplayer, 2 anonymous edits

**MaxDB** *Source:* <http://en.wikipedia.org/w/index.php?oldid=360533887> *Contributors:* AlistairMcMillan, Andareed, Bobprime, Cander0000, Chillywillycd, Cjcollier, Clowess, Cool Hand Luke, Craesh, Deeahbz, DocendoDiscimus, Doco, Dreadstar, Edward nz, Gurch, JasonLax, Jerome Charles Potts, Jlundell, Joel Alcalay, Jon vs, Jonah.harris, Jtdirl, K1Bond007, Karnesky, Kent Wang, Kingdon, Knaveofhearts, Minghong, Nainawalli, Neilc, Nihiltres, Nono64, Odie5533, RN, Reedy, Rjwilmsi, Seb35, Sharcho, Skychutw, Sleske, Technobadger, Tim baroon, Unyoyega, WatchAndObserve, Who, Zero0w, Zondor, 39 anonymous edits

**Meta-SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=346332563> *Contributors:* Dlohrer2003, Gary King, GregorB, Gunnala, RockMFR, Serenaacw, Squids and Chips, 6 anonymous edits

**MySQL Workbench** *Source:* <http://en.wikipedia.org/w/index.php?oldid=364111604> *Contributors:* AlexLibman, Andreas Kaufmann, Avenger cn, Billism, CDV, Capiscuas, Chealer, Daniel.Cardenas, Dfrankow, DreamGYM, Europimp, Faisal.akeel, Frap, Goa103, M4gnum0n, Magioladitis, Mariuz, Minnaert, Neovov, Ogre lawless, Ronz, RossPatterson, Scott.stratford, Seifried, Thumperward, Tim baroon, Tomjenkins52, Valenajarro, Vapor75, Voidxor, 56 anonymous edits

**NVL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=356244270> *Contributors:* BlueNovember, Fsmatovu, Kylebrennan1, NickelShoe, Random832, Sasawat, 2 anonymous edits

**Navicat** *Source:* <http://en.wikipedia.org/w/index.php?oldid=363895121> *Contributors:* 16@r, Anakin101, Bob barker rox my sox, Cander0000, Closedmouth, Dreftymac, FSElias, Fiftyquid, Frap, JLaTondre, Jak123, Jerazol, Jonny-mt, Manycat, Nkocharh, Premiumsoft, Redvers, Rwww, Scarian, The Evil Spartan, 27 anonymous edits

**Nested SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=291047094> *Contributors:* Pdn30, Sinneed, Slashme, SwiftGus, Victor falk, WriterListener, 2 anonymous edits

**Object-oriented SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=309689673> *Contributors:* Adolescence, Bearcat, Jeepday, Segv11, 1 anonymous edits

**PL/pgSQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=352486888> *Contributors:* AlexKarpman, Andy Dingley, Avinashm, Brianray, Georgeryp, Integr, John Vandenberg, Martijn Hoekstra, Mwtoews, Neilc, Nickdc, Northernhenge, Rjwilmsi, The RedBurn, Turnstep, 8 anonymous edits

**PL/SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=368123629> *Contributors:* Adreamsoul, Aegicen, Akadruid, Alansohn, Alex.g, Andy Dingley, Anna Lincoln, Aravind730, Bovineone, Brianray, Bwefler, Can't sleep, clown will eat me, CanisRufus, Centrx, Chappy84, Chowbok, Chrissawer, Chutzpan, ClemMcGann, Cm10497, Cometstyles, Comp123, Conversion script, Craigy144, Cwjoiley, Cybercobra, Derbeth, DevastatorILC, Drovetto, E0steven, EagleOne, Edans.sandes, Engmark, Eustress, Fdesing, Fsilva27, Fubar Obfusco, Gianfranco, GrandPoohBah, Greensburger, GregorB, Helix84, Hroptatyr, Hu12, Hydrogen Iodide, Icoy, Japo, Jdforrester, Jlam, Johayek, John Vandenberg, Jonathan321, Josemanimala, Kalapratik, Kokedada, Kmorozov, Krishchik, Kuru, Kusunose, Leandro, Lerdsuwa, Lexikorn, Linlasj, Loren.wilton, Luiscosta, Madhukar83, Maheshvenna, Mark Renier, Michael Devore, Michig, Mikeblas, Miker@sundialservices.com, Mintleaf, MrOllie, Naudelf, Nickdc, Nicolas1981, Nuno Tavares, PaulBoxley, Pedant17, Peterl, Pjetter, Quadell, Quale, Rajithmohan, Ramkrish, RedWolf, Rednblu, Rjwilmsi, Ruud Koot, S.K., SarekOfVulcan, Scrollwheel, Seb az86556, Securiger, Sef96121, Shadowjams, Shell Kinney, Shimeru, ShlomoS, Shoaler, Sippsin, Sjc, SlowJog, Soosed, Stevietheman, TMSTKSBB, Tertrih, The cat ncl uk, ThisIsDennis, Turnstep, VMS Mosaic, VictorAnyakin, Vipinhari, Wernher, Whitehatnetizen, Yonatan, Ziguang, 343 anonymous edits

**Pro\*C** *Source:* <http://en.wikipedia.org/w/index.php?oldid=219915444> *Contributors:* Closedmouth, EagleOne, Kingturtle, Shamatt, 2 anonymous edits

**Query optimizer** *Source:* <http://en.wikipedia.org/w/index.php?oldid=366713427> *Contributors:* AAA!, Andreas Kaufmann, Arcann, Bevo, Blauerflummi, CanonMR, Danhuby, Friendlydata, Hardeeps, Hertzsprung, Hjzla, Kdau, Larsinio, Manop, Mark Renier, Mikeblas, Mindmatrix, Minghong, Moltonel3xCombo, MvS, Neilc, Nikola Smolenski, Nothings, Reedy, Rshankar13, Ruud Koot, Samunoz1, Sdorance, Shell Kinney, Snodnipper, TechPurism, Thorwald, Thue, 40 anonymous edits

**Query plan** *Source:* <http://en.wikipedia.org/w/index.php?oldid=348122507> *Contributors:* Aaronbrick, Ammar.w, Ancheta Wis, Arcann, Bevo, Cww, Freezegravity, Grace Note, Hardeeps, James barton, Larsinio, Mark Renier, Mbarbier, Mdesmet, Mikeblas, Mindmatrix, Neilc, Reedy, RI, SimonP, Sippsin, Slaniel, TheParanoidOne, ZoBlitz, 11 anonymous edits

**Rollback (data management)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=365459310> *Contributors:* Alexius08, Brick Thrower, Craig Stuntz, Emperorbma, Gadmio, JCLately, JonHarder, Khalid hassani, Mattisse, Mikeblas, Oxyeron83, Pegship, Piotrus, Pne, Rfl, SimonP, Turnstep, Unknown W. Brackets, Vedant, Wikiklrsc, 16 anonymous edits

**SQL Access Group** *Source:* <http://en.wikipedia.org/w/index.php?oldid=244225330> *Contributors:* Andreas Kaufmann, DrorHarari, Fabrictramp, JLaTondre, Lowellian, 2 anonymous edits

**SQL CLR** *Source:* <http://en.wikipedia.org/w/index.php?oldid=361655394> *Contributors:* Alai, Amalas, Colonies Chris, Dkent600, Dojikami, Echuck215, Jmkehayias, KathrynLybarger, Malcolm, Rabin06, Ridor.cz, Rogerd, Soumyasch, Torc2, Warren, Xpclient, 8 anonymous edits

**SQL Problems Requiring Cursors** *Source:* <http://en.wikipedia.org/w/index.php?oldid=309732042> *Contributors:* Fabrictramp, Malcolm, Somewherepurple, 6 anonymous edits

**SQL injection** *Source:* <http://en.wikipedia.org/w/index.php?oldid=367864378> *Contributors:* - ), .anaconda, Af648, Alansohn, Alerante, Alex Marandon, Americasroof, Aminadav, AndyDent, AndyHassall, Aneeshjoseph, AnnaFinotera, Anna Lincoln, Antientropic, Apokrif, ArielGold, ArnoldReinhold, Ayolucas, Badgernet, BaldPark, Balusc, Belal qudah, Benjamin Pineau, Bensonwu, Bevnet, Bevo, Biskeh, Blake-, BobKeim, Bookbrad, Bradjamesbrown, Btx40, CAN, CKlunck, Caesura, Caim, Catrope, Cdean, Cenizc, Ch'marr, Cheesieluv, Chris-marsh-usa, Chrisj2, ChristianEdwardGruber, Chuunen Baka, CliffC, CoJaBo, Cybercobra, Damian Yerrick, DamnFools, Dandv, Danhash, Danielosneto, Davidhorman, Daydreamer302000, DerHexer, Discospinster, Dmitry Evteev, Drol, Elkman, Elwikipedista, Enigmasoldier, Enric Naval, Erik9, Everyking, Excirial, Fedevela, Feezo, Ferkelparade, Finnigall, Folajimi, Freedomlinux, Furrykef, Garylhwitt, Gmoose1, Gogo Dodo, Golbez, GregorB, H4ckf0rs4k3, HDCase, HalJor, Hariva, HeW6, Hede2000, Hurrn, Husky, II MusLiM HyBRiD II, Indy90, IntergalacticRabbit, Island, Ixtd64, JLEm, Jamesooders, Javawizard, Jeffg, Jeffrey Mall, Jeterfan428, Jmanico, Jnarvey, JoeSmack, Jonsiddle, Jjacques, KD5TV1, Kahina, Kalimantan kid, Kate, Kenkku, KeyStroke, Kingpin13, Kitchen, Klizza, Lawrencegold, Ldo, Leirith, Liftarn, Little Mountain 5, Luna Santin, Maghnus, Martin Hinks, Max613, Mboverload, Mcgyver5, Mchl, MeekMark, MentisQ, Michael Slone, MichaelCoates, Michaelhodgins, MightyWarrior, Miko3k, Mild Bill Hiccup, Milo99, Mindmatrix, Mirko051, Mopatop, Moreschi, MrZanzi, Mrdehate, N5iln, Nabieh, Nbertram, Nic tester, Nickgalea, Nidheeshks, Njan, Nosbig, Od Mishehu, Offli, Oli Filth, Oskar Sigvardsson, Oxyeron83, Panoptical, Pearl's sun, Peterl, Pharaoh of the Wizards, Piano non troppo, Pinecar,

Pingveno, Plumbago, Portablegeek, Prescriptiononline, Project2501a, Public Menace, RadioActive, Rand20s, Ratfox, Raysonho, Rztus, Rd232 public, Reedy, Revivethespirit, ReyBrujo, Rjanag, Rodney viana, Roman Lagunov, RonhJones, Roshenc, Rpkrawczyk, SP-KP, Samngms, ScottW, Shabbirbhimani, Shadowjams, SheeEtin, Shlomif, Shirlitz, Sniper1rfa, Societebi, SpK, SteinbDJ, Storm Rider, Straussian, Suei8423, Superm401, Taka, Terrifictrifid, TheBilly, TheRingess, ThomasMueller, Tide rolls, Tjkiesel, Tobi Kellner, Tom-, Trevor MacInnis, Troels Arvin, Unlox775, VASTA zx, Vasil, Vis says, Vladocar, Vupen, WTucker, Wbrice83186, Werikba, WhiteCrane, WibWobble, Wikilost, WikipedianMarlith, Wkeevers96, Wknight94, Wwwwolf, XDanielx, Yamamoto Ichiro, Z29plurazAlpha, Zedlander, Zgdot, Zhyale, Zzuuzz, 591 anonymous edits

**SQL/CLI** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319841353> *Contributors:* SqlPac, Touko vk, 1 anonymous edits

**SQL/MED** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319863593> *Contributors:* Fundou, SqlPac, Touko vk, 5 anonymous edits

**SQL/OLB** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319866276> *Contributors:* Jackie, SqlPac, Touko vk, 1 anonymous edits

**SQL/PSM** *Source:* <http://en.wikipedia.org/w/index.php?oldid=344645108> *Contributors:* Greenrd, GregorB, Jdlambert, RossPatterson, SqlPac, 2 anonymous edits

**SQL/Schemata** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319869577> *Contributors:* Friendlydata, SqlPac, Touko vk, Troels Arvin, 1 anonymous edits

**SQL/XML** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319877546> *Contributors:* FloatingMind, Klausness, SqlPac, Touko vk, 8 anonymous edits **SQLPro**

**SQL Client** *Source:* <http://en.wikipedia.org/w/index.php?oldid=341442806> *Contributors:* JLaTondre, Nice999, Woohookitty, 1 anonymous edits

**Scriptella** *Source:* <http://en.wikipedia.org/w/index.php?oldid=365140502> *Contributors:* 7Volk, Centrx, Chowbok, Dgies, EagleOne, Hangy, Khalid hassani, PigFlu Oink, Tkpwms, William Avery, 38 anonymous edits

**SQL PL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=339518087> *Contributors:* Berny68, Jdlambert, Malcolm, Rwww, Squids and Chips, 3 anonymous edits

**SQL/JRT** *Source:* <http://en.wikipedia.org/w/index.php?oldid=319870938> *Contributors:* Download, Mark Renier, SqlPac, Touko vk, 1 anonymous edits

**Squirrel SQL Client** *Source:* <http://en.wikipedia.org/w/index.php?oldid=364290680> *Contributors:* EagleOne, Fiercel, KI4m-AWB, Lowellian, MRqtH2, Manngr, MickScott, Patriotic dissent, Rich Farmbrough, Satya.sid, Tgabor, TreasuryTag, 7 anonymous edits

**Standard Interchange Language** *Source:* <http://en.wikipedia.org/w/index.php?oldid=335282017> *Contributors:* Ausgangskontrolle, Erechtheus, Gary King, Jpbowen, Vivafelis

**Table (database)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=362659152> *Contributors:* 12george1, 16@r, Ajraddatz, Alai, Arcann, AutumnSnow, Blanchardb, Bobo192, Bongwarrior, Bruxism, Cbrunschen, Cyfal, Drefymac, Epbr123, Funnyfarmofdoom, Gurch, IMSoP, IanCarter, Jamelan, Jerazol, Jerome Charles Potts, Larsinio, LeonardoGregianin, Mark Renier, Mblumber, Mikeblas, Mindmatrix, Morad86, N0nr3s, Nibs208, Ofus, Quentar, S.K., Sietse Snel, SimonP, Sippsin, Sonett72, SqlPac, Stolze, TheParanoidOne, Turnstep, Txomin, Ummit, Versageek, Zhenqinli, 68 anonymous edits

**Transact-SQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=357831239> *Contributors:* Apokrif, Ashmoo, Big Smooth, Bongwarrior, Brick Thrower, Chrisahn, Donarreiskoffer, Fotek, Greystork, Hbent, IDX, Jamelan, Jogloran, Klapi, MIT Trekkie, Marius, Masgatoitkaca, Mdchachi, Mikeblas, Mwtoews, Norm mit, Nricardo, Oskar Liljeblad, Pascal666, PatrickFisher, Paulineward, Perfecto, Phil Boswell, Pxma, Quilokos, Qxz, Rhobite, Rsocel, Rturnham, Ruud Koot, S.Övrr.S, Shanes, Spolster, SummerWithMorons, The Thing That Should Not Be, Troels Arvin, Warren, 99 anonymous edits

**Truviso** *Source:* <http://en.wikipedia.org/w/index.php?oldid=361313695> *Contributors:* El jefe, GregorySmith, Jgonion, Kerberus13, Neilc, NurseryRhyme, 2 anonymous edits

**User-defined function** *Source:* <http://en.wikipedia.org/w/index.php?oldid=351713704> *Contributors:* AKGhetto, ANNAfoxlover, Amcguinn, Andrzej P. Wozniak, Chowbok, Chriscf, Crystallina, Giraffedata, Gpvos, Lankiveil, Mercy, Mikeblas, Mitsukai, Mpin, Petersap, RevRagnarok, ReyBrujo, Seqsea, SmilesALot, Stolze, ZacBowling, 23 anonymous edits

**varchar** *Source:* <http://en.wikipedia.org/w/index.php?oldid=323592692> *Contributors:* AlisonW, Anphanax, Brenthale, Elwikipedista, MasKa, Mikeblas, NicM, Octahedron80, Shamespwns, SueHay, 24 anonymous edits

**View (database)** *Source:* <http://en.wikipedia.org/w/index.php?oldid=364383817> *Contributors:* Acjelen, Alai, Andrew.george.hammond, Blowdart, BullRunner2009, ClementSeveillac, Dcoetzee, Dfmg.msc, Elcasc, FernandoAires, Fragment, JDHeinzmann, JForget, JLaTondre, Jerazol, Jobbin, Jtgerman, Jwoodger, Kibbled bits, Kku, Kuru, Larsinio, Mark Renier, Mathmo, Mikeblas, Mindmatrix, MrOllie, Muchium, Nothings, Quentar, RaBa, Raeky, Rjwilmsi, Roux, S.K., Sappy, Scrool, Sippsin, Smalljim, Stolze, TheDamian, Toyota prius 2, Troels Arvin, WmLGann, Wwphx, Z.E.R.O., Zhenqinli, 86 anonymous edits

**WQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=354784455> *Contributors:* Bryan Derksen, Drclue, GhettoBlaster, Mindmatrix, Rypcord, Tarekradi, TubularWorld, Warren, 1 anonymous edits

**Windows Internal Database** *Source:* <http://en.wikipedia.org/w/index.php?oldid=350002133> *Contributors:* BenStrauss, FleetCommand, Soumyasch, Warren, 9 anonymous edits

**XLeratorDB** *Source:* <http://en.wikipedia.org/w/index.php?oldid=355575851> *Contributors:* Belovedfreak, Cander0000, Difu Wu, Drbreznjev, Dthomsen8, J04n, Jpbowen, R'nB, Sqltool, 1 anonymous edits

**XSQL** *Source:* <http://en.wikipedia.org/w/index.php?oldid=362589846> *Contributors:* Bunnyhop11, Cander0000, FatalItY, HJWeng, Intrgr, Legoktm, Levin, Melab-1, Tabletop, Xezbeth, 4 anonymous edits

# Image Sources, Licenses and Contributors

**File:SQL ANATOMY wiki.svg** Source: [http://en.wikipedia.org/w/index.php?title=File:SQL\\_ANATOMY\\_wiki.svg](http://en.wikipedia.org/w/index.php?title=File:SQL_ANATOMY_wiki.svg) License: GNU Free Documentation License Contributors: :User:SqlPac, modified by Ferdna. Original uploader was Ferdna at en.wikipedia

**File:Dbforgestudio-small.jpg** Source: <http://en.wikipedia.org/w/index.php?title=File:Dbforgestudio-small.jpg> License: GNU Free Documentation License Contributors: Devart (company)

**Image:Postgresql elephant.svg** Source: [http://en.wikipedia.org/w/index.php?title=File:Postgresql\\_elephant.svg](http://en.wikipedia.org/w/index.php?title=File:Postgresql_elephant.svg) License: BSD Contributors: Jeff MacDonald

**Image:Hsql.png** Source: <http://en.wikipedia.org/w/index.php?title=File:Hsql.png> License: unknown Contributors: Cijk, Sfan00 IMG

**File:Dbd4 ss simplemodel.png** Source: [http://en.wikipedia.org/w/index.php?title=File:Dbd4\\_ss\\_simplemodel.png](http://en.wikipedia.org/w/index.php?title=File:Dbd4_ss_simplemodel.png) License: Creative Commons Attribution-Sharealike 3.0 Contributors: User:MichaelGZinner

**File:MySQLAdministrator1.png** Source: <http://en.wikipedia.org/w/index.php?title=File:MySQLAdministrator1.png> License: GNU General Public License Contributors: User:Nevetsjc

**Image:SQLServer QueryPlan.png** Source: [http://en.wikipedia.org/w/index.php?title=File:SQLServer\\_QueryPlan.png](http://en.wikipedia.org/w/index.php?title=File:SQLServer_QueryPlan.png) License: unknown Contributors: Mikeblas

**Image:Scriptella logo.png** Source: [http://en.wikipedia.org/w/index.php?title=File:Scriptella\\_logo.png](http://en.wikipedia.org/w/index.php?title=File:Scriptella_logo.png) License: Public Domain Contributors: Ejboy

**Image:Truviso\_logo.JPG** Source: [http://en.wikipedia.org/w/index.php?title=File:Truviso\\_logo.JPG](http://en.wikipedia.org/w/index.php?title=File:Truviso_logo.JPG) License: unknown Contributors: Fetchcomms, Kerberus13, Rettetast, Skier Dude

**File:Flag of the United States.svg** Source: [http://en.wikipedia.org/w/index.php?title=File:Flag\\_of\\_the\\_United\\_States.svg](http://en.wikipedia.org/w/index.php?title=File:Flag_of_the_United_States.svg) License: Public Domain Contributors: User:Dbenbenn, User:Indolences, User:Jacobolus, User:Technion, User:Zscout370

**Image:WestClinTech box prod.jpg** Source: [http://en.wikipedia.org/w/index.php?title=File:WestClinTech\\_box\\_prod.jpg](http://en.wikipedia.org/w/index.php?title=File:WestClinTech_box_prod.jpg) License: Creative Commons Attribution-Sharealike 3.0 Contributors: Sqltool

# License

---

Creative Commons Attribution-Share Alike 3.0 Unported  
<http://creativecommons.org/licenses/by-sa/3.0/>

---