SQL

This article is about the database language. For the IATA code, see San Carlos Airport (California).

SQL (/ˈɛs kjuː ˈɛl/,[4] or /ˈsiːkwəl/;[5] **Structured Query Language**[6][7][8][9]) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and a data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks."[10] Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.[11][12]

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.[13] Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, though, most SQL code is not completely portable among different database systems without adjustments.

1 History

SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s.[14] This version, initially called *SEQUEL* (*Structured English QUEry Language*), was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970s.[14] The acronym SEQUEL was later changed to SQL because "SEQUEL" was a trademark of the UK-based Hawker Siddeley aircraft company.[15]

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 June 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers.

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.[16]

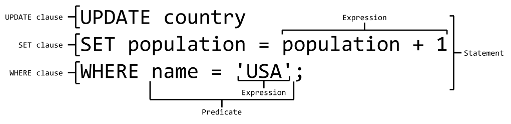
2 Design

SQL deviates in several ways from its theoretical foundation, the relational model and its tuple calculus. In that model, a table is a set of tuples, while in SQL, tables and query results are lists of rows: the same row may occur multiple times, and the order of rows can be employed in queries (e.g. in the LIMIT clause). Whether this is a practical concern is a subject of debate. Furthermore, additional features (such as NULL and views) were introduced without founding them directly on the relational model, which makes them more difficult to interpret.

Critics argue that SQL should be replaced with a language that strictly returns to the original foundation: for example, see *The Third Manifesto*.

3 Syntax

3.1 Language elements



```
UPDATE country
SET population = population + 1
WHERE name = 'USA';
```

A chart showing several of the SQL language elements that compose a single statement

The SQL language is subdivided into several language elements, including:

- *Clauses*, which are constituent components of statements and queries. (In some cases, these are optional.)[17]

- *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) (true/false/unknown) or Boolean truth values and are used to limit the effects of statements and queries, or to change program flow.
- *Queries*, which retrieve the data based on specific criteria. This is an important element of *SQL*.
- *Statements*, which may have a persistent effect on schemata and data, or 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.

3.2 Operators

Other operators have at times been suggested and/or implemented, such as the skyline operator (for finding only those records that are not 'worse' than any others).

SQL has the case/when/then/else/end expression, which was introduced in SQL-92. In its most general form, which is called a “searched case” in the SQL standard, it works like else if in other programming languages:

CASE WHEN n > 0 THEN 'positive' WHEN n < 0 THEN 'negative' ELSE 'zero' END

SQL tests WHEN conditions in the order they appear in the source. If the source does not specify an ELSE expression, SQL defaults to ELSE NULL. An abbreviated syntax—called “simple case” in the SQL standard—mirrors switch statements:

CASE n WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'I cannot count that high' END

This syntax uses implicit equality comparisons, with the usual caveats for comparing with NULL.

For the Oracle-SQL dialect, the latter can be shortened to an equivalent DECODE construct:

SELECT DECODE(n, 1, 'one', 2, 'two', 'i cannot count that high') FROM some_table;

The last value is the default; if none is specified, it also defaults to NULL. However, unlike the standard’s “simple case”, Oracle’s DECODE considers two NULLs equal with each other.[18]

3.3 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.[19]

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 include 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) to retrieve data from. 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 where 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 to use to sort the resulting data, and in which direction to sort them (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 AS 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 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;

3.3.1 Subqueries

Queries can be nested so that the results of one query can be used in another query via a relational operator or aggregation function. A nested query is also known as a *subquery*. While joins and other table operations provide computationally superior (i.e. faster) alternatives in many cases, the use of subqueries introduces a hierarchy in execution that can be useful or necessary. In the following example, the aggregation function AVG receives as input the result of a subquery:

SELECT isbn, title, price FROM Book WHERE price < (SELECT AVG(price) FROM Book) ORDER BY title;

A subquery can use values from the outer query, in which case it is known as a correlated subquery.

Since 1999 the SQL standard allows named subqueries called common table expression (named and designed after the IBM DB2 version 2 implementation; Oracle calls these subquery factoring). CTEs can also be recursive by referring to themselves; the resulting mechanism allows tree or graph traversals (when represented as relations), and more generally fixpoint computations.

3.3.2 Inline View

An Inline view is the use of referencing an SQL subquery in a FROM clause. Essentially, the inline view is a subquery that can be selected from or joined to. Inline View functionality allows the user to reference the subquery as a table. The inline view also is referred to as a *derived table* or a *subselect*. Inline view functionality was introduced in Oracle 9i.[20]

In the following example, the SQL statement involves a join from the initial Books table to the Inline view "Sales". This inline view captures associated book sales information using the ISBN to join to the Books table. As a result, the inline view provides the result set with additional columns (the number of items sold and the company that sold the books):

Select b.isbn, b.title, b.price, sales.items_sold sales.company_nm from Book b, (select SUM(Items_Sold) Items_Sold, Company_Nm, ISBN from Book_Sales group by Company_Nm, ISBN) sales WHERE sales.isbn = b.isbn

3.3.3 Null or three-valued logic (3VL)

Main article: Null (SQL)

The concept of Null was introduced into SQL to handle missing information in the relational model. The word NULL is a reserved keyword in SQL, used to identify the Null special marker. Comparisons with Null, for instance equality (=) in WHERE clauses, results in an Unknown truth value. In SELECT statements SQL returns only results for which the WHERE clause returns a value of True; i.e., it excludes results with values of False and also excludes those whose value is Unknown.

Along with True and False, the Unknown resulting from direct comparisons with Null thus brings a fragment of three-valued logic to SQL. The truth tables SQL uses for AND, OR, and NOT correspond to a common fragment of the Kleene and Lukasiewicz three-valued logic (which differ in their definition of implication, however SQL defines no such operation).[21]

There are however disputes about the semantic interpretation of Nulls in SQL because of its treatment outside direct comparisons. As seen in the table above direct equality comparisons between two NULLs in SQL (e.g. NULL = NULL) returns a truth value of Unknown. This is in line with the interpretation that 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. However, the principle that two Nulls aren't equal to each other is effectively violated in the SQL specification for the UNION and INTERSECT operators, which do identify nulls with each other.[22] Consequently, these set operations in SQL may produce results not representing sure information, unlike operations involving explicit comparisons with NULL (e.g. those in a WHERE clause discussed above). In Codd’s 1979 proposal (which was basically adopted by SQL92) this semantic inconsistency is rationalized by arguing that removal of duplicates in set operations happens “at a lower level of detail than equality testing in the evaluation of retrieval operations.”[21] However, computer science professor Ron van der Meyden concluded that “The inconsistencies in the SQL standard mean that it is not possible to ascribe any intuitive logical semantics to the treatment of nulls in SQL.”[22]

Additionally, since SQL operators return Unknown when comparing anything with Null directly, SQL provides two Null-specific comparison predicates: IS NULL and IS NOT NULL test whether data is or is not Null.[23] Universal quantification is not explicitly supported by SQL, and must be worked out as a negated existential quantification.[24][25][26] There is also the "<row value expression> IS DISTINCT FROM <row value expression>" infixed 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>)". SQL:1999 also introduced BOOLEAN type variables, which according to the standard can also hold Unknown values. In practice, a number of systems (e.g. PostgreSQL) implement the BOOLEAN Unknown as a BOOLEAN NULL.

3.4 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 example (field1, field2, field3) VALUES ('test', 'N', NULL);

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

UPDATE example SET field1 = 'updated value' WHERE field2 = 'N';

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

DELETE FROM example WHERE field2 = 'N';

- 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".

MERGE INTO table_name USING table_reference ON (condition) WHEN MATCHED THEN UPDATE SET column1 = value1 [, column2 = value2 ...] WHEN NOT MATCHED THEN INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...])

3.5 Transaction controls

Transactions, if available, wrap DML operations:

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

CREATE TABLE tbl_1(id int); INSERT INTO tbl_1(id) VALUES(1); INSERT INTO tbl_1(id) VALUES(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 makes all data changes in a transaction permanent.
- ROLLBACK discards all data changes since the last COMMIT or ROLLBACK, leaving 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. The following example shows a classic transfer of funds transaction, where money is removed from one account and added to another. If either the removal or the addition fails, the entire transaction is rolled back.

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;

3.6 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, e.g.:

CREATE TABLE example(column1 INTEGER, column2 VARCHAR(50), column3 DATE NOT NULL, PRIMARY KEY (column1, column2));

- ALTER modifies the structure of an existing object in various ways, for example, adding a column to an existing table or a constraint, e.g.:

ALTER TABLE example ADD column4 NUMBER(3) NOT NULL;

- TRUNCATE deletes all data from a table in a very fast way, deleting the data inside the table and not the table itself. It usually implies a subsequent COMMIT operation, i.e., it cannot be rolled back (data is not written to the logs for rollback later, unlike DELETE).

TRUNCATE TABLE example;

- DROP deletes an object in the database, usually irretrievably, i.e., it cannot be rolled back, e.g.:

DROP TABLE example;

3.7 Data types

Each column in an SQL table declares the type(s) that column may contain. ANSI SQL includes the following data types.[27]

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

For example, the number 123.45 has a precision of 5 and a scale of 2. The precision is a positive integer that determines the number of significant digits in a particular radix (binary or decimal). The scale is a non-negative integer. A scale of 0 indicates that the number is an integer. For a decimal number with scale S, the exact numeric value is the integer value of the significant digits divided by 10^S.

SQL provides a function to round numerics or dates, called TRUNC (in Informix, DB2, PostgreSQL, Oracle and MySQL) or ROUND (in Informix, SQLite, Sybase, Oracle, PostgreSQL and Microsoft SQL Server)[28]

Date and time

- DATE: for date values (e.g. 2011-05-03)
- TIME: for time values (e.g. 15:51:36). The granularity of the time value is usually a *tick* (100 nanoseconds).
- TIME WITH TIME ZONE or TIMETZ: 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. 2011-05-03 15:51:36).
- 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. The IBM Informix implementation provides the EXTEND and the FRACTION functions to increase the accuracy of time, for systems requiring sub-second precision.[29]

3.8 Data control

The Data Control Language (DCL) authorizes 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 example TO some_user, another_user; REVOKE SELECT, UPDATE ON example FROM some_user, another_user;

4 Procedural extensions

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

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 unmanaged extended stored procedures primarily written in C. PostgreSQL lets users write functions in a wide variety of languages—including Perl, Python, Tcl, and C.[30]

5 Interoperability and standardization

SQL implementations are incompatible between vendors and do not necessarily completely follow standards. In particular date and time syntax, string concatenation, NULLs, and comparison case sensitivity vary from vendor to vendor. A particular exception is PostgreSQL, which strives for standards compliance.[31]

Popular implementations of SQL commonly omit support for basic features of Standard SQL, such as the DATE or TIME data types. The most obvious such examples, and incidentally the most popular commercial and proprietary SQL DBMSs, are Oracle (whose DATE behaves as DATETIME,[32][33] and lacks a TIME type)[34] and MS SQL Server (before the 2008 version). 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 newer version of the SQL standard conflicts with the prior behavior of the vendor's database, the vendor may be unwilling to break backward compatibility.
- There is little commercial incentive for vendors to make it easier for users to change database suppliers (see vendor lock-in).
- Users evaluating database software tend to place other factors such as performance higher in their priorities than standards conformance.

SQL was adopted as a standard by the American National Standards Institute (ANSI) in 1986 as SQL-86[35] and the International Organization for Standardization (ISO) in 1987. Nowadays the standard is subject to continuous improvement by the Joint Technical Committee *ISO/IEC JTC 1, Information technology, Subcommittee SC 32, Data management and interchange*, which affiliate to ISO as well as IEC. It is commonly denoted by the pattern: *ISO/IEC 9075-n:yyyy Part n: title*, or, as a shortcut, *ISO/IEC 9075*.

ISO/IEC 9075 is complemented by *ISO/IEC 13249: SQL Multimedia and Application Packages* (SQL/MM), which defines SQL based interfaces and packages to widely spread applications like video, audio and spatial data.

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.[36]

The original standard declared that the official pronunciation for "SQL" was an initialism: /ˈɛs kjuː ˈɛl/ ("es queue el").[11] Regardless, many English-speaking database professionals (including Donald Chamberlin himself[37]) use the acronym-like pronunciation of /ˈsiːkwəl/ ("sequel"),[38] mirroring the language's pre-release development name of "SEQUEL".[14][15]

The SQL standard has gone through a number of revisions:

Interested parties may purchase SQL standards documents from ISO,[41] IEC or ANSI. A draft of SQL:2008 is freely available as a zip archive.[42]

The SQL standard is divided into nine parts.

- ISO/IEC 9075-1:2011 Part 1: *Framework* (SQL/Framework). It provides logical concepts.
- ISO/IEC 9075-2:2011 Part 2: *Foundation* (SQL/Foundation). It contains the most central elements of the language and consists of both *mandatory and optional* features.
- ISO/IEC 9075-3:2008 Part 3: *Call-Level Interface* (SQL/CLI). It defines interfacing components (structures, procedures, variable bindings) that can be used to execute SQL statements from applications written in Ada, C respectively C++, COBOL, Fortran, MUMPS, Pascal or PL/I. (For Java see part 10.) 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.
- ISO/IEC 9075-4:2011 Part 4: *Persistent Stored Modules* (SQL/PSM) It 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.
- ISO/IEC 9075-9:2008 Part 9: *Management of External Data* (SQL/MED). It 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.
- ISO/IEC 9075-10:2008 Part 10: *Object Language Bindings* (SQL/OLB). It defines the syntax and semantics of SQLJ, which is SQL embedded in Java (see also part 3). 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, as opposed to SQL/OLB JDBC, which is not part of the SQL standard, which defines an API.
- ISO/IEC 9075-11:2011 Part 11: *Information and Definition Schemas* (SQL/Schemata). It 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.[43] This part of the standard contains both *mandatory and optional* features.
- ISO/IEC 9075-13:2008 Part 13: *SQL Routines and Types Using the Java Programming Language* (SQL/JRT). It specifies the ability to invoke static Java methods as routines from within SQL applications ('Java-in-the-database'). 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.
- ISO/IEC 9075-14:2011 Part 14: *XML-Related Specifications* (SQL/XML). It 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.[39] This part of the standard consists solely of *optional* features.

ISO/IEC 9075 is complemented by ISO/IEC 13249 *SQL Multimedia and Application Packages*. This closely related but separate standard is developed by the same committee. It defines interfaces and packages based on SQL. The aim is a unified access to typical database applications like text, pictures, data mining or spatial data.

- ISO/IEC 13249-1:2007 Part 1: *Framework*
- ISO/IEC 13249-2:2003 Part 2: *Full-Text*

- ISO/IEC 13249-3:2011 Part 3: *Spatial*
- ISO/IEC 13249-5:2003 Part 5: *Still image*
- ISO/IEC 13249-6:2006 Part 6: *Data mining*
- ISO/IEC 13249-8:xxxx Part 8: *Metadata registries (MDR)* (work in progress)

6 Alternatives

A distinction should be made between alternatives to SQL as a language, and alternatives to the relational model itself. Below are proposed relational alternatives to the SQL language. See navigational database and NoSQL for alternatives to the relational model.

- .QL: object-oriented Datalog
- 4D Query Language (4D QL)
- BQL: a superset that compiles down to SQL
- Datalog: critics suggest that Datalog has two advantages over SQL: it has cleaner semantics, which facilitates program understanding and maintenance, and it is more expressive, in particular for recursive queries.[44]
- HTSQL: URL based query method
- IBM Business System 12 (IBM BS12): one of the first fully relational database management systems, introduced in 1982
- ISBL
- jOOQ: SQL implemented in Java as an internal domain-specific language
- Java Persistence Query Language (JPQL): The query language used by the Java Persistence API and Hibernate persistence library
- LINQ: Runs SQL statements written like language constructs to query collections directly from inside .Net code.
- Object Query Language
- OttoQL
- QBE (Query By Example) created by Moshè Zloof, IBM 1977
- Quel introduced in 1974 by the U.C. Berkeley Ingres project.
- Tutorial D
- XQuery

7 Distributed SQL processing

Distributed Relational Database Architecture (DRDA) was designed by a work group within IBM in the period 1988 to 1994. DRDA enables network connected relational databases to cooperate to fulfill SQL requests. [45] [46]

An interactive user or program can issue SQL statements to a local RDB and receive tables of data and status indicators in reply from remote RDBs. SQL statements can also be compiled and stored in remote RDBs as packages and then invoked by package name. This is important for the efficient operation of application programs that issue complex, high-frequency queries. It is especially important when the tables to be accessed are located in remote systems.

The messages, protocols, and structural components of DRDA are defined by the Distributed Data Management Architecture.

8 See also

- Comparison of object-relational database management systems
- Comparison of relational database management systems
- D (data language specification)
- D4 (programming language)
- Hierarchical model
- List of relational database management systems
- MUMPS
- NoSQL
- Transact-SQL
- Online analytical processing (OLAP)
- Online transaction processing (OLTP)
- Data warehouse
- relational data stream management system
- Star schema
- Snowflake schema
- DB2 SQL return codes

9 Notes

[1] "Media Type registration for application/sql". Internet Assigned Numbers Authority. 10 April 2013. Retrieved 10 April 2013.

[2] "The application/sql Media Type, RFC 6922". Internet Engineering Task Force. April 2013. p. 3. Retrieved 10 April 2013.

[3] Paul, Ryan. "A guided tour of the Microsoft Command Shell". Ars Technica. Retrieved 10 April 2011.

[4] Beaulieu, Alan (April 2009). Mary E Treseler, ed. *Learning SQL* (2nd ed.). Sebastapol, CA, USA: O'Reilly. ISBN 978-0-596-52083-0.

[5] "SQL, n.". *Oxford English Dictionary*. Oxford University Press. Retrieved 2014-11-27.

[6] Encyclopedia Britannica. "SQL". Retrieved 2013-04-02.

[7] Oxford Dictionaries. "SQL".

[8] IBM. "SQL Guide".

[9] Microsoft. "Structured Query Language (SQL)".

[10] Codd, Edgar F (June 1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* (Association for Computing Machinery) **13** (6): 377–87. doi:10.1145/362384.362685. Retrieved 2007-06-09.

[11] Chapple, Mike. "SQL Fundamentals". *Databases*. About.com. Retrieved 2009-01-28.

[12] "Structured Query Language (SQL)". International Business Machines. October 27, 2006. Retrieved 2007-06-10.

[13] "ISO/IEC 9075-1:2008: Information technology – Database languages – SQL – Part 1: Framework (SQL/Framework)".

[14] Chamberlin, Donald D; Boyce, Raymond F (1974). "SEQUEL: A Structured English Query Language" (PDF). *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control* (Association for Computing Machinery): 249–64. Retrieved 2007-06-09.

[15] Oppel, Andy (February 27, 2004). *Databases Demystified*. San Francisco, CA: McGraw-Hill Osborne Media. pp. 90–1. ISBN 0-07-146960-5.

[16] "History of IBM, 1978". *IBM Archives*. IBM. Retrieved 2007-06-09.

[17] ANSI/ISO/IEC International Standard (IS). Database Language SQL—Part 2: Foundation (SQL/Foundation). 1999.

[18] "DECODE". Docs.oracle.com. Retrieved 2013-06-14.

[19] "Transact-SQL Reference". *SQL Server Language Reference*. SQL Server 2005 Books Online. Microsoft. 2007-09-15. Retrieved 2007-06-17.

[20] "Derived Tables". ORACLE.

[21] Hans-Joachim, K. (2003). "Null Values in Relational Databases and Sure Information Answers". *Semantics in Databases. Second International Workshop Dagstuhl Castle, Germany, January 7–12, 2001. Revised Papers*. Lecture Notes in Computer Science **2582**. pp. 119–138. doi:10.1007/3-540-36596-6_7. ISBN 978-3-540-00957-3.

[22] Ron van der Meyden, "Logical approaches to incomplete information: a survey" in Chomicki, Jan; Saake, Gunter (Eds.) *Logics for Databases and Information Systems*, Kluwer Academic Publishers ISBN 978-0-7923-8129-7, p. 344

[23] ISO/IEC. *ISO/IEC 9075-2:2003, "SQL/Foundation"*. ISO/IEC.

[24] M. Negri, G. Pelagatti, L. Sbattella (1989) *GUIDE Semantics and problems of universal quantification in SQL*

[25] Fratarcangeli, Claudio (1991). *Technique for universal quantification in SQL*. ACM.org.

[26] 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. Thefreelibrary.com

[27] "Information Technology: Database Language SQL". CMU. (proposed revised text of DIS 9075).

[28] Arie Jones, Ryan K. Stephens, Ronald R. Plew, Alex Kriegel, Robert F. Garrett (2005), *SQL Functions Programmer's Reference*. Wiley, 127 pages.

[29]

[30] PostgreSQL contributors (2011). "PostgreSQL server programming". *PostgreSQL 9.1 official documentation*. postgresql.org. Retrieved 2012-03-09.

[31] PostgreSQL contributors (2012). "About PostgreSQL". *PostgreSQL 9.1 official website*. PostgreSQL Global Development Group. Retrieved March 9, 2012. PostgreSQL prides itself in standards compliance. Its SQL implementation strongly conforms to the ANSI-SQL:2008 standard

[32] Lorentz, Diana; Roeser, Mary Beth; Abraham, Sundeep; Amor, Angela; Arora, Geeta; Arora, Vikas; Ashdown, Lance; Baer, Hermann; Bellamkonda, Shrikanth (October 2010) [1996]. "Basic Elements of Oracle SQL: Data Types". *Oracle Database SQL Language Reference 11g Release 2 (11.2)*. Oracle Database Documentation Library. Redwood City, CA: Oracle USA, Inc. Retrieved December 29, 2010. For each DATE value, Oracle stores the following information: century, year, month, date, hour, minute, and second

[33] Lorentz, Diana; Roeser, Mary Beth; Abraham, Sundeep; Amor, Angela; Arora, Geeta; Arora, Vikas; Ashdown, Lance; Baer, Hermann; Bellamkonda, Shrikanth (October 2010) [1996]. "Basic Elements of Oracle SQL: Data Types". *Oracle Database SQL Language Reference 11g Release 2 (11.2)*. Oracle Database Documentation Library. Redwood City, CA: Oracle USA, Inc. Retrieved December 29, 2010. The datetime data types are DATE...

[34] Lorentz, Diana; Roeser, Mary Beth; Abraham, Sundeep; Amor, Angela; Arora, Geeta; Arora, Vikas; Ashdown, Lance; Baer, Hermann; Bellamkonda, Shrikanth (October 2010) [1996]. "Basic Elements of Oracle SQL: Data Types". *Oracle Database SQL Language Reference 11g Release 2 (11.2)*. Oracle Database Documentation Library. Redwood City, CA: Oracle USA, Inc. Retrieved December 29, 2010. Do not define columns with the following SQL/DS and DB2 data types, because they have no corresponding Oracle data type:... TIME

[35] "Finding Aid". *X3H2 Records, 1978–95*. American National Standards Institute.

[36] Doll, Shelley (June 19, 2002). "Is SQL a Standard Anymore?". *TechRepublic's Builder.com*. TechRepublic. Archived from the original on 2013-01-02. Retrieved 2010-01-07.

[37] Gillespie, Patrick. "Pronouncing SQL: S-Q-L or Sequel?". *Pronouncing SQL: S-Q-L or Sequel?*. Retrieved 12 February 2012.

[38] Melton, Jim; Alan R Simon (1993). "1.2. What is SQL?". *Understanding the New SQL: A Complete Guide*. Morgan Kaufmann. p. 536. ISBN 1-55860-245-3. SQL (correctly pronounced "ess cue ell," instead of the somewhat common "sequel")...

[39] Wagner, Michael (2010). *SQL/XML:2006 - Evaluierung der Standardkonformität ausgewählter Datenbanksysteme*. Diplomica Verlag. p. 100. ISBN 3-8366-9609-6.

[40] "SQL:2008 now an approved ISO international standard". Sybase. July 2008.

[41] "ISO/IEC 9075-2:2011: Information technology -- Database languages -- SQL -- Part 2: Foundation (SQL/Foundation)".

[42] "SQL:2008 draft" (Zip). Whitemarsh Information Systems Corporation.

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

[44]

[45] Reinsch, R. (1988). "Distributed database for SAA". *IBM Systems Journal* **27** (3): 362–389. doi:10.1147/sj.273.0362.

[46] *Distributed Relational Database Architecture Reference*. IBM Corp. SC26-4651-0. 1990.

10 References

- Codd, Edgar F (June 1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* **13** (6): 377–87. doi:10.1145/362384.362685.
- Discussion on alleged SQL flaws (C2 wiki)
- C. J. Date with Hugh Darwen: *A Guide to the SQL standard : a users guide to the standard database language SQL, 4th ed.*, Addison Wesley, USA 1997, ISBN 978-0-201-96426-4

11 External links

- *1995 SQL Reunion: People, Projects, and Politics*, by Paul McJones (ed.): transcript of a reunion meeting devoted to the personal history of relational databases and SQL.
- American National Standards Institute. X3H2 Records, 1978–1995 Charles Babbage Institute Collection documents the H2 committee's development of the NDL and SQL standards.
- Oral history interview with Donald D. Chamberlin 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 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.
- Event stream processing with SQL - An introduction to real-time processing of streaming data with continuous SQL queries

12 Text and image sources, contributors, and licenses

12.1 Text