

# Structured Query Language/Print version



*Note: current version of this book can be found at [http://en.wikibooks.org/wiki/Structured\\_Query\\_Language](http://en.wikibooks.org/wiki/Structured_Query_Language)*

Remember to click "refresh" to view this version.

## Introduction

### It's a translation and a guide

This Wikibook introduces the programming language SQL as defined by ISO/IEC. The standard — similar to most standard publications — is quite technical and neither easy to read nor understand. So there is a demand for a text document explaining key features of the language. And that is what this wikibook strives to do: present a readable, understandable introduction for everyone interested in the topic.

Manuals and white papers of database vendors are also focused mainly into technical aspects of their product. As they want to set themselves apart from each other, they tend to emphasize those aspects which go beyond the SQL standard and the products from other vendors. This is contrary to our approach: we want to emphasize the common aspects.

So the main audience of this wikibook is people who want to learn the language, maybe as a beginner or as a person with existing knowledge and some degree of experience.

### What is this wikibook not?

First of all, this wikibook is not a reference manual for the syntax of standard SQL or any implementation. Reference manuals usually consist of definitions and explanations to those definitions. By contrast, this wikibook tries to lead you to concepts and basic commands through textual descriptions and examples. Of course we will demonstrate the syntax. But you may reckon that there are slight differences to your concrete implementation.

Also, it is not a complete tutorial. First, its focus is the standard and not any concrete implementation. When you learn a computer language it is necessary to work with it and experience with your own examples. Hence, you need a concrete implementation. And most of them differ from the standard more or less. Second, this wikibook is far away from reflecting the complete standard, e.g. the central part of the standard consists of about 18 MB text in more than 1,400 pages. But you can use this wikibook as a companion on your way learning about SQL.

### How to proceed?

If you are new to SQL, you should study the chapters and pages from beginning to end. For persons having some experience with SQL and interest in a specific aspect, it should be possible to navigate directly to any page.

You need not have knowledge about any other computer language, but it will be helpful.

This wikibook consists of descriptions, definitions, and examples. You should read it with care. But, it is absolutely necessary that you do some experiments with data and data structures on your own. Hence, you need access to a concrete database system where you can do read-only and read-write tests. For those tests, you can use our example database, or you may define your own tables and data.

## Conventions

The elements of the language SQL are case-insensitive, e.g.: it makes no difference whether you write *SELECT* ..., *Select* ..., *select* ... or any combination of upper and lower case characters like *SeLeCt* .... For readability reasons, we use the convention that all language keywords are written in upper case letters and all names of user objects e.g. table and column names, are written in lower case letters.

We will write short SQL commands within one row.

```
SELECT street FROM address WHERE city = 'Duckburg';
```

For longer commands spawning multiple lines we use a *tabular format*.

```
SELECT street
FROM address
WHERE city IN ('Duckburg', 'Gotham City', 'Hobbs Lane');
```

**Advice:** Storing and retrieving **text data** is case sensitive! If you store a cityname 'Duckburg' you cannot retrieve it as 'duckburg'.

## Historical Context

One of the original scopes of computer applications was storing large amounts of data on mass storage devices and retrieving them at a later point in time. Over time user requirements increased to include not only sequential access but also random access to data records, concurrent access by parallel (writing) processes, recovery after hardware and software failures, high performance, scalability, etc. In the 1970s and 1980s, the science and computer industries developed techniques to fulfill those requests.

Define Query languages?

## What makes up a Database Management System?

Basic bricks for efficient data storage - and for this reason for all Database Management Systems (DBMS) - are implementations of fast read and write access algorithms to data located in central memory and mass storage devices like routines for B-trees, Index Sequential Access Method (ISAM), other indexing techniques as well as buffering of dirty and non-dirty blocks. These algorithms are not unique to DBMS. They also apply to file systems, some programming languages, operating systems, application server and much more.

In addition to the appropriation of these routines, a DBMS guarantees compliance with the **ACID** paradigm. This compliance means, that in a multi-user environment all changes to data within one transaction are:

**Atomic:** all changes take place or none.

**Consistent:** changes transform the database from one valid state to another valid state.

**Isolated:** transactions of different users working at the same time will not affect each other.

**Durable:** the database retains committed changes even if the system crashes afterwards.

## Classification of DBMS Design

You can distinguish between the following generations of DBMS design and implementation:

- **Hierarchical DBMS:** Data structures are designed in a hierarchical parent/child model where every child has exactly **one** parent (with the exception of the root structure, which has no parent). The result is that the data is

modeled and stored as a tree. Child rows are physically stored directly after the owning parent row. So there is no need to store the parent's ID or something like it within the child row (XML realizes a similar approach). If an application processes data in exactly this hierarchical way, it is very fast and efficient. On the other hand, other means of access are less efficient. Furthermore, hierarchical DBMSs do not provide the modeling of n:m relations. Another fault is that you have no possibility to navigate directly to data stored in lower levels. You must first navigate over the given hierarchy before reaching that data.

The best known hierarchical DBMS is IMS from IBM.

- **Network DBMS:** The network model designs data structures as a complex network with links from one or more parent nodes to one or more child nodes. Even cycles are possible. There is no need for a single root node. In general the terms *parent node* and *child node* lose their hierarchical meaning and may be referred as *link source* and *link destination*. As those links are realized as physical links within the database, applications which follow the links show good performance.
- **Relational DBMS:** The relational model designs data structures as relations (tables) with attributes (columns) and the relationship between those relations. Definitions in this model are expressed in a pure declarative way **not predetermining** any implementation issue like links from one relation to another one or a certain sequence of rows in the database. Relationships are based purely upon content. At runtime all linking and joining is done by evaluating the actual data values, e.g.: ... WHERE employee.department\_id = department.id .... The consequence is that - with the exception of explicit foreign keys - there is no meaning of a parent/child or owner/member denotation. Relationships in this model do not have any direction.

The relational model and SQL are based on the mathematical theory of relational algebra.

During the 1980s and 1990s proprietary and open source DBMS based on the relational design paradigm established themselves as market leaders.

- **Object oriented DBMS:** Nowadays most applications are written in an object oriented programming language (OOP). If in such cases the underlying DBMS belongs to the class of relational DBMS, the so called object-relational impedance mismatch arises, that is to say in contrast to the application language pure relational DBMS (prDBMS) do not support central concepts of OOP:

**Type system:** OOP offers the facility to define own classes with complex internal structures. They are built on primitive types, system classes, references to other or the same class. prDBMS knows only predefined types. Additionally prDBMS insists in first normal form, which means that attributes must be scalar. In OOP they may be sets, lists or arrays of the desired type.

**Inheritance:** Classes of OOP may inherit attributes and methods from their superclass. This concept is not known by prDBMS.

**Polymorphism:** The runtime system can decide via late binding which one of a group of methods with the same name and parameter types will be called. This concept is not known by prDBMS.

**Encapsulation:** Data and access methods to data are stored within the same class. It is not possible to access the data directly - the only way is using the access methods of the class. This concept is not known by prDBMS.

Object oriented DBMS are designed to overcome the gap between prDBMS and OOP. At their peak, they reached a weak market position in the mid and late 1990s. Afterwards some of their concepts were incorporated into the SQL standard as well as rDBMS implementations.

- **NoSQL:** The term NoSQL stands for the emerging group of DBMS which differs from others in central concepts:
  - They do not necessarily support all aspects of the ACID paradigm.
  - The data must not necessarily be structured according to any schema.
  - Their goal is support for fault-tolerant, distributed data with very huge volume.
  - Implementations differ widely in storing techniques: you can see key-value stores, document oriented databases, graph oriented databases and more.
- They do not offer a SQL interface. In 2011 an initiative started to define an alternative language: *Unstructured Query Language* as part of SQLite.

## The Theory

A relational DBMS is an implementation of data stores according to the design rules of the relational model. This approach allows operations on the data according to the relational algebra like projections, selections, joins, set operations (union, difference, intersection, ...) and more. Together with Boolean algebra (and, or, not, exists, ...) and other mathematical concepts, relational algebra builds up a complete mathematical system with basic operations, complex operations and transformation rules between the operations. Neither a DBA nor an application programmer needs to know the relational algebra. But you should know that your rDBMS is based on this mathematical foundation - and that it has the freedom to transform queries into several forms.

## The Data Model

The relational model designs data structures as relations (tables) with attributes (columns) and the relationship between those relations. The information about one entity of the real world is stored within one row of a table. In this spirit the term *one entity of the real world* must be used with care. It may be that our intellect identifies a machine like a single airplane in this vein. Depending on the information requirements you may decide to put all information into one row of a table *airplane*. But in many cases it is necessary to break up the entity into its pieces - and model the pieces as separate entities including the relation to the whole thing. If you, for example, need information about every single seat within the airplane, you need to have a second table *seat* and some way of joining seats to airplanes. And in the end you will have a great number of tables.

This way of breaking up information about real entities into a complex data model depends highly on the information requirements of the business concept. The resulting data model should conform to a so-called normal form. And the good news is: It will not predetermine the proceeding of applications. It is strictly descriptive and will not restrict the access to the data in any way.

## Some more Basics

Operations within databases must have the ability to act not only on a single row but on a set of rows. Relational algebra offers this possibility. Therefore languages based on it, e.g.: SQL, can offer a powerful syntax to manipulate a great bunch of data within one single command.

As operations within relational algebra may be replaced by different but logically equivalent operations, a language based on relational algebra should not predetermine how its syntax is mapped to operations (the execution plan). The language should describe **what** should be done and not **how** to do it. Note: This choice of operations did not concern the use or neglect of indices.

As described before the relational model tends to break up objects into sub-objects. In this and in other cases it is often necessary to collect associated information from a bunch of tables to one information unit. How is this possible without links between participating tables and rows? The answer is: All joining is done based on the **values** which are actually stored in the attributes. The rDBMS must make its own decision about how to reach all concerned rows: read all potentially affected rows and ignore those which are irrelevant (full table scan)? Or, use some kind of index and read only those which match the criteria? This value-based approach allows even the use of other operators than the equal-operator, e.g.:

```
SELECT * FROM gift JOIN box ON gift.extent < box.extent;
```

This command will join all "gift" records to all "box" records with a larger "extent" (whatever "extent" means).

## History

As outlined above rDBMS acts on the data with operations of the relational algebra like projections, selections, joins, set operations (union, difference, intersection, ...) and more. The operations of the relational algebra are denoted in a mathematical language which is highly formal and hard to understand for end users and - possibly also - for many software engineers. Therefore rDBMS offers a layer above relational algebra, which is easy to understand but nevertheless can be mapped to the underlying relational operations. Since the 1970s we have seen some languages

doing this job, one of them was SQL - another example was QUEL. In the early 1980s (after a rename from its original name *SEQUEL* due to trademark problems) SQL achieved market dominance. And in 1986 SQL was standardized for the first time. The current version is SQL 2011.

## Characteristics

The tokens and syntax of SQL are oriented on **English common speech** to keep the access barrier as small as possible. A SQL command like `UPDATE employee SET salary = 2000 WHERE id = 511;` is not far away from the sentence "Change employee's salary to 2000 for the one with id 511."

The next simplification is that all key words of SQL can be expressed in any combination of upper and lower case characters someone prefers. It makes no difference whether you write `UPDATE`, `update`, `Update`, `UpDate` or any other combination of upper and lower case characters. The keywords are **case insensitive**.

Next SQL is a **descriptive** language, not a procedural one. It does not pre-decide all aspects of the relational operations (which operation, their order, ...) which are generated from the given SQL statement. The rDBMS has the freedom to generate more than one execution plan from it and run this one it thinks is the best in the given situation. Additionally the end user is freed from all the gory details of data access, e.g.: Which one of a set of `WHERE` criteria should be evaluated first if they are combined with `AND`?

Despite those simplifications SQL is very powerful. Especially it allows the manipulation of a **set of data** records with one single statement. `UPDATE employee SET salary = salary * 1.1 WHERE salary < 2000;` will affect all employee records with an actual salary smaller than 2000: there may be thousands of those records, only a few or even zero. And you may have recognized that the operation is not a fixed manipulation. The wording `SET salary = salary * 1.1` leads to an increase of the salaries by 10%, which may be 120 for one employee and 150 for another one.

The designer of SQL tried to define the language elements **orthogonally** to each other. Among other things that means that any language element may be used at all positions of a statement where the result of that element may be used directly. E.g.: If you have a function `power()` which takes two numbers and returns another number, you can use this function at all places where numbers are allowed. The following statements are syntactically correct (if you have defined the function `power()`) - and they lead to the same data records, but this aspect is not important here.

```
SELECT salary FROM employee WHERE salary < 2048;
SELECT salary FROM employee WHERE salary < POWER(2, 11);
SELECT POWER(salary, 1) FROM employee WHERE salary < 2048;
```

Another example of orthogonality is the use of subqueries within `UPDATE`, `INSERT`, `DELETE` or inside another `SELECT` statement.

Nevertheless SQL is not free of **redundancy**. Often there are a lot of choices you can choose from to express the same situation.

```
SELECT salary FROM employee WHERE salary < 2048;
SELECT salary FROM employee WHERE NOT salary >= 2048;
SELECT salary FROM employee WHERE salary BETWEEN 0 AND 2047.99;
```

is a simple example. In complex statements you may have the choice between joins, subqueries and the *exists* construct.

## Fundamentals

Core SQL consists of statements. Statements consist of key words, operators, values, names of system and user

objects or functions. They are concluded by a semicolon. In the statement `SELECT salary FROM employee WHERE id < 100;` the tokens `SELECT`, `FROM` and `WHERE` are key words. `salary`, `employee` and `id` are object names, the "<" sign is an operator and "100" is a value.

The SQL standard arranges statements into 9 groups:

"The main classes of SQL-statements are:

SQL-schema statements; these may have a persistent effect on the set of schemas.

SQL-data statements; some of these, the SQL-data change statements, may have a persistent effect on SQL data.

SQL-transaction statements; except for the <commit statement>, these, and the following classes, have no effects that persist when an SQL-session is terminated.

SQL-control statements.

SQL-connection statements.

SQL-session statements.

SQL-diagnostics statements.

SQL-dynamic statements.

SQL embedded exception declaration."

This detailed grouping is unusual in common speech. Usually we distinguish between three groups:

*Data Definition Language (DDL)*: Managing the structure of database objects (create/drop tables, views, columns, ...)

*Data Manipulation Language (DML)*: Managing and retrieval of data with the statements `INSERT`, `UPDATE`, `DELETE`, `SELECT`, `COMMIT` and `ROLLBACK`.

*Data Control Language (DCL)*: Managing access rights.

Hint: In some publications the `SELECT` statement is said to build its own group *Data Query Language*. This group has no other statements than `SELECT`.

## Turing completeness

Core SQL as described above is not Turing complete. It misses conditional branches, variables, subroutines. But the standard as well as most implementations offers an extension to fulfill the demand for Turing completeness. In 'Part 4: Persistent Stored Modules (SQL/PSM)' of the standard there are definitions for `IF`-, `CASE`-, `LOOP`-, `Assignment`- and other statements. The existing implementations of this part have different names, different syntax and also a different scope of operation: `PL/SQL` in Oracle, `SQL/PL` in DB2, `Transact-SQL` or `T-SQL` in MS-SQL and Sybase, `PL/pgSQL` in Postgres and simply 'stored procedures' in MySQL.

## Benefit of Standardization

Like most other standards the main purpose of SQL is **portability**. Usually software designers and application developers structure and solve problems in layers. Every abstraction level is realized in its own component or sub-component: presentation to end user, business logic, data access, data storage, net and operation system demands are typical representatives of such components. They are organized as a stack and every layer offers an interface to the upper layers to use its functionality. If one of those components is realized by two different providers and both offer the same interface (as an API, Web-Service, language specification, ...) it is possible to exchange them without changing the layers which are based on them. In essence the software industry needs **stable interfaces** at the top of important layers to avoid dependence on a single provider. SQL acts as such an interface to relational database systems.

If an application uses only those SQL commands which are defined within standard SQL, it should be possible to exchange the underlying rDBMS with a different one without changing the source code of the application. In practice this is a hard job, because concrete implementations offer numerous additional features and software engineers love to use them.

A second aspect is the **conservation of know how**. If a student learns SQL, he is in a position to develop applications which are based on an arbitrary database system. The situation is comparable with any other popular programming

language. If one learns Java or C-Sharp, he can develop applications of any kind running on a lot of different hardware systems and even different hardware architectures.

## Limits

Database systems consist of many components. The access to the data is an important but not the only component. Additionally there are many more tasks: throughput optimization, physical design, backup, distributed databases, replication, 7x24 availability, ... . Standard SQL is focused mainly on data access and ignores typical DBA tasks. Even the `CREATE INDEX` statement as a widely used optimization strategy is not part of the standard. Nevertheless the standard fills thousands of pages. But most of the DBA's daily work is highly specialized to every concrete implementation and must be done in a different way when he switches to a different rDBMS. Mainly application developers benefit from SQL.

## The Standardization Process

The standardization process is organized in two levels. The first level acts in a national context. Interested companies, universities and persons of one country work within their national standardization organisation like ANSI, Deutsches Institut für Normung (DIN) or British Standards Institution (BSI), where every member has one vote. The second level is the international stage. The national organizations are members of ISO respectively IEC. In case of SQL there is a common committee of ISO and IEC named Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 32, Data management and interchange, where every national body has one vote. This committee approve the standard under the name *ISO/IEC 9075-n:yyyy*, where *n* is the part number and *yyyy* is the year of publication. The nine parts of the standard are described in short here.

If the committee releases a new version, this may concern only some of the nine parts. So it is possible that the *yyyy* denomination differs from part to part. *Core SQL* is defined mainly by the second part: *ISO/IEC 9075-2:yyyy Part 2: Foundation (SQL/Foundation)* - but it contains also some features of other parts.

Note: The API JDBC is part of Java SE and Java EE but not part of the SQL standard.

The standard is complemented by a second, closely related standard: *ISO/IEC 13249-n:yyyy SQL Multimedia and Application Packages*, which is developed by the same organizations and committee. This publication defines interfaces and package based on SQL. They focus on special kind of applications: text, pictures, data mining and spatial data applications.

## Verification of Conformance to the Standard

Until 1996 the National Institute of Standards and Technology (NIST) certified the compliance of the SQL implementation of rDBMS with the SQL standard. As NIST abandon this work, nowadays vendors self-certify the compliance of their product. They must declare the degree of conformance in a special appendix of their documentation. This documentation may be voluminous as the standard defines not only a set of base features - called *Core SQL:yyyy* - but also a lot of additional features an implementation may conform to or not.

## Implementations

To fulfill their clients' demands all major vendors of rDBMS offers - among other data access ways - the language SQL within their product. The implementations cover *Core SQL*, a bunch of additional standardized features and a huge number of additional, not standardized features. The access to standardized features may use the regular syntax or an implementation specific syntax. In essence SQL is the clamp holding everything together, but normally there are a lot of detours around the official language.

## Chapters

The main drive behind a relational database is to increase accuracy by increasing the efficiency with which data is stored. For example, the names of each of the millions of people who immigrated to the United States through Ellis Island at the turn of the 20th century were recorded by hand on large sheets of paper; people from the city of London had their country of origin entered as England, or Great Britain, or United Kingdom, or U.K., or UK, or Engl., etc. Multiple ways of recording the same information leads to future confusion when there is a need to simply know how many people came from the country now known as the United Kingdom.

The modern solution to this problem is the database. A single entry is made for each country, for example, in a reference list that might be called the Country table. When someone needs to indicate the United Kingdom, he only has one choice available to him from the list: a single entry called "United Kingdom". In this example, "United Kingdom" is the unique representation of a country, and any further information about this country can use the same term from the same list to refer to the same country. For example, a list of telephone country codes and a list of European castles both need to refer to countries; by using the same Country table to provide this identical information to both of the new lists, we've established new relationships among different lists that only have one item in common: country. A relational database, therefore, is simply a collection of lists that share some common pieces of information.

## Structured Query Language (SQL)

**SQL**, which is an abbreviation for **Structured Query Language**, is a language to request data from a database, to add, update, or remove data within a database, or to manipulate the metadata of the database.

SQL is generally pronounced as the three letters in the name, e.g. *ess-cue-ell*, or in some people's usage, as the word *sequel*.

SQL is a declarative language in which the expected result or operation is given without the specific details about how to accomplish the task. The steps required to execute SQL statements are handled transparently by the SQL database. Sometimes SQL is characterized as *non-procedural* because procedural languages generally require the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, or flushing buffers and writing data to filesystems. Therefore, SQL is considered to be designed at a higher conceptual level of operation than procedural languages because the lower level logical and physical operations aren't specified and are determined by the SQL engine or server process that executes it.

Instructions are given in the form of statements, consisting of a specific SQL statement and additional parameters and operands that apply to that statement. SQL statements and their modifiers are based upon official SQL standards and certain extensions to that each database provider implements. Commonly used statements are grouped into the following categories:

### Data Query Language (DQL)

- **SELECT** - Used to retrieve certain records from one or more tables.

### Data Manipulation Language (DML)

- **INSERT** - Used to create a record.
- **UPDATE** - Used to change certain records.
- **DELETE** - Used to delete certain records.

### Data Definition Language (DDL)

- **CREATE** - Used to create a new table, a view of a table, or other object in database.
- **ALTER** - Used to modify an existing database object, such as a table.
- **DROP** - Used to delete an entire table, a view of a table or other object in the database.

### Data Control Language (DCL)

- **GRANT** - Used to give a privilege to someone.
- **REVOKE** - Used to take back privileges granted to someone.

Before learning SQL, relational databases have several concepts that are important to learn first. Databases store the data of an information system. We regroup data by groups of comparable data (all the employees, all the projects, all



the offices...). For each group of comparable data, we create a *table*. This table is specially designed to suit this type of data (its attributes). For instance, a table named `employee` which stores all the employees would be designed like this:

<b>employee</b> <small>the table</small>	
<u>id_employee</u> <small>the primary key</small>	an integer
firstname <small>a column</small>	a string of characters <small>a column type</small>
lastname	a string of characters
phone	10 numbers
mail	a string of characters

And the company employees would be stored like this:

<b>employee</b>				
<u>id_employee</u>	firstname	lastname	phone	mail
1 <small>a column value</small>	Big	BOSS	936854270	big.boss@company.com
2	John	DOE	936854271	john.doe@company.com
3	Linus	TORVALDS	936854272	linus.torvalds@company.com
4	Jimmy	WALES	936854273	jimmy.wales@company.com
5	Larry	PAGE	936854274	larry.page@company.com

The data stored in a table is called *entities*. As a table is usually represented as an array, the data attributes (first name, last name...) are called *columns* and the records (the employees) are called *rows*. `id_employee` is a database specific technical identifier called a *primary key*. It is used to link the entities from a table to another. To do so, it must be unique for each row. A primary key is usually underlined. Any unique attribute (for instance, the mail) or group of attributes (for instance, the first name and last name) can be the table primary key but it is recommended to use an additional technical id (`id_employee`) for primary key.

Let's create a second table called `project` which stores the company projects:

<b>employee</b>	
<u>id_employee</u>	an integer
firstname	a string of characters
lastname	a string of characters
phone	10 numbers
mail	a string of characters

<b>project</b>	
<u>id_project</u>	an integer
name	a string of characters
created_on	a date
ended_on	a date
# manager	an integer

And the company projects would be stored like this:

<b>employee</b>				
<u>id_employee</u>	firstname	lastname	phone	mail
1	Big	BOSS	936854270	big.boss@company.com
2	John	DOE	936854271	john.doe@company.com
3	Linus	TORVALDS	936854272	linus.torvalds@company.com
4	Jimmy	WALES	936854273	jimmy.wales@company.com
5	Larry	PAGE	936854274	larry.page@company.com

<b>project</b>		
<u>id_project</u>	name	created
1	Google	1998-09
2	Linux	1991-01
3	Wikipedia	2001-01

`id_project` is the primary key of the project table and `manager` is a *foreign key*. A foreign key is a technical id which is equal to one of the primary keys stored in another table (here, the employee table). Doing this, the Google project is linked to the employee Larry PAGE. This link is called a *relationship*. A foreign key is usually preceded by a sharp. Note that several projects can point to a common manager so an employee can be the manager of several projects.

Now, we want to create, not a single link, but multiple links. So we create a *junction table*. A junction table is a table that isn't used to store data but links the entities of other tables. Let's create a table called `members` which links employees to project:

employee	
<u>id_employee</u>	an integer
firstname	a string of characters
lastname	a string of characters
phone	10 numbers
mail	a string of characters

members	
# <u>id_employee</u>	an integer
# <u>id_project</u>	an integer

project	
<u>id_project</u>	an integer
name	a string of characters
created_on	a date
ended_on	a date
# manager	an integer

And the employees and the projects can be linked like this:

employee				
<u>id_employee</u>	firstname	lastname	phone	mail
1	Big	BOSS	936854270	big.boss@company.com
2	John	DOE	936854271	john.doe@company.com
3	Linus	TORVALDS	936854272	linus.torvalds@company.com
4	Jimmy	WALES	936854273	jimmy.wales@company.com
5	Larry	PAGE	936854274	larry.page@company.com
6	Max	THE GOOGLER	936854275	max.the-googler@company.com
7	Jenny	THE WIKIPEDIAN	936854276	jenny.the-wikipedian@company.com

proj		
<u>id_project</u>	name	creat
1	Google	1998-
2	Linux	1991-
3	Wikipedia	2001-

members	
# <u>id_employee</u>	# <u>id_project</u>
3	2
2	1
4	3
5	1
2	3
6	1
7	3

An employee can be associated to several projects (John DOE with Google and Wikipedia) and a project can be associated to several employees (Wikipedia with Jimmy, John and Jenny), which is impossible with just a foreign key. A junction table hasn't its own primary key. Its primary key is the couple of foreign keys, as this couple is unique. A junction table can link more than two entity tables by containing more columns.

# Relationships

So let's list the different types of relationships:

- One to one,
- One to many (for instance, the manager of a project),
- Many to many (for instance, the members of the projects).

For each type of relationships, there is a way to link the entities :

- One to many relationship: create a foreign key from an entity table to the other,
- Many to many relationship: create a junction table,
- One to one relationship: just merge the two tables.

Now you know how to design a database schema and to put the data of your information system into it.

Data Query Language is used to extract data from the database. It doesn't modify any data in the database. It describes only one query: SELECT.

## SQL data types

Each column has a type. Here are the standard SQL data types:

Data type	Explanation	Allowed values	Example
VARCHAR(n)	A string with a maximum length of n	[0-9a-zA-Z]{n}	"foo"
CHAR(n)	A string with a fixed length of n	[0-9a-zA-Z]{n}	"foo"
SMALLINT	A 16 bits signed integer	\-?[0-9]+	584
INTEGER	A 32 bits signed integer	\-?[0-9]+	-8748
FLOAT	A decimal floating point	\-?[0-9]+[\.[0-9]+]?	48.96
NUMBER(n,[d])	A number with n digits (and d decimal digits if mentioned)	\-?[0-9]+[\.[0-9]+]?	484.65
DATE	A date (YYYY-MM-DD)	[0-9][0-9][0-9][0-9]\-[0-1][0-9]\-[0-3][0-9]	2009-03-24
TIME	A time period of sixty minutes; one twenty-fourth of a day	[0-2][0-9]\:[0-5][0-9]\:[0-5][0-9]	11:24:56
TIMESTAMP	A date and hour	[0-9]+	18648689595962
BLOB	Any binary data	Any	

There is no boolean type. Integers are used instead.

## SELECT query

The exhaustive syntax of the SELECT query is as follows:

```
SELECT [ ALL | DISTINCT ] <COLUMN name> [[ AS ] <alias> ], [ ALL | DISTINCT ] <COLUMN name> [[ AS ] <alias> ] *
FROM <table> [[ AS ] <alias> ] [ [ FULL | LEFT | RIGHT | OUTER | INNER ] JOIN <table> ON <expression> ]
[ , <table> [[ AS ] <alias> ] [ [ FULL | LEFT | RIGHT | OUTER | INNER ] JOIN <table> ON <expression> ] ] *
[ WHERE <predicate> [ { AND | OR } <predicate> ] * ]
[ GROUP BY <COLUMN name> [ , <COLUMN name> ] * ]
[ HAVING <predicate> [ { AND | OR } <predicate> ] * ]
]
[ ORDER BY <COLUMN name> [ ASC | DESC ] [ , <COLUMN name> [ ASC | DESC ] ] * ]
[ FETCH FIRST <count> ROWS ONLY ] ;
```

## First query

Let's create the table `reunion` with many columns:

reunion	
<u>id_reunion</u>	INTEGER
name	VARCHAR(20)
description	VARCHAR(255)
priority	CHAR(1)
planned	SMALLINT
date	DATE
hour	TIME
duration	INTEGER
# id_office	INTEGER
pdf_report	BLOB

...and let's fill it:

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528

Let's do a simple query. The following query just returns the content of the `reunion` table:

### ■ Query:

```
SELECT *  
FROM reunion;
```

### ■ Result:

id_reunion	name	description	priority	planned	date	hour	day
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	6
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	3
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	9
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	1
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	6
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	1

The form of the result depends on the client application. It can be returned as a text output (backend), a HTML page (thin client), a program object (middleware) etc... The statements, queries, clauses (SELECT, FROM...), instructions and operators are not case sensitive but they are commonly written in uppercase for readability.

The SELECT and FROM clauses are the two required clauses of a SELECT query:

- FROM : list the tables the query uses to return the data,
- SELECT : list the data to return.

## WHERE clause

The WHERE clause doesn't influence the columns the query returns but the rows. It filters the rows applying *predicates* on it. A *predicate* specifies conditions that can be true or false. SQL can handle conditions whose result is unknown. For example, the following query returns the reunions which have a B priority level:

### ▪ Query:

```
SELECT *
FROM reunion
WHERE reunion.priority = 'B';
```

### ▪ Result:

id_reunion	name	description	priority	planned	date	hour	day
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	9
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	6
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	1

The table name can be omitted if it is not ambiguous.

## Predicate

Compared to the second operand, the first operand can be :

- equal : =
- different : <>
- lesser : <
- lesser or equal : <=
- greater : >
- greater or equal : >=

The following query returns the reunions which have another priority level than B:

### ▪ Query:

```
SELECT *
```

```
FROM reunion
WHERE priority <> 'B';
```

▪ **Result:**

id_reunion	name	description	priority	planned	date	hour	duration
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	15

## Operators

The WHERE clause can have several conditions using the operators AND (all the conditions must be true) and OR (only one condition needs to be true). The operator OR is inclusive (several conditions can be true). The order of evaluation can be indicated with brackets. NOT inverts a condition. The following query returns the reunions which have a B priority level and last more than an hour or which take place on 2008/05/12:

▪ **Query:**

```
SELECT *
FROM reunion
WHERE (priority = 'B' AND NOT duration <= 60) OR DATE = '2008-05-12';
```

▪ **Result:**

id_reunion	name	description	priority	planned	date	hour	duration
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	15

## LIKE

LIKE allows simplified regular expression matching. It can be applied on the text columns (CHAR, VARCHAR,...).

- *Alphanumerical characters* only match identical text,
- % is a wildcard that matches any text,
- \_ is a wildcard that matches any single character,

The following query returns the reunions which end with "ing" and which contain " the " in its description:

▪ **Query:**

```
SELECT *
FROM reunion
WHERE name LIKE '%ing' AND description LIKE '% the %';
```

▪ **Result:**

id_reunion	name	description	priority	planned	date	hour	duration
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60

5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60
---	-----------	----------------------------------	---	---	------------	----------	----

## BETWEEN and IN

BETWEEN matches a range of values that can be numbers, dates or times. IN matches a list of allowed values. The following query returns the reunions which take place between 2008-04-01 and 2009-04-01 and have an A, B or D priority level:

### ■ Query:

```
SELECT *
FROM reunion
WHERE DATE BETWEEN '2008-04-01' AND '2009-04-01' AND priority IN ('A', 'B', 'D');
```

### ■ Result:

id_reunion	name	description	priority	planned	date	hour	duration
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60

## EXISTS

EXISTS is usually used with a subselect. This predicate is true if the list (i.e. the result set of a subselect) is not empty. This keyword allows to filter the returned rows using data that are not directly associated to the returned rows (i.e. they are not joined, not linked, not related... to the returned rows) so you can not use junction in this case. For instance, we want to retrieve all the reunions for which there is at least one reunion two times longer:

### ■ Query:

```
SELECT *
FROM reunion r1
WHERE EXISTS (
  SELECT r2.id_reunion
  FROM reunion r2
  WHERE r2.duration = r1.duration * 2
);
```

### ■ Result:

id_reunion	name	description	priority	planned	date	hour	duration
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60

The duration of another reunion is used in this query whereas there is no join, no link and no relationship between the two rows. This condition can not be done without EXISTS. Note that the subselect uses the alias r1 whereas this alias is defined in the main query.

EXISTS is also used to match a lack of data. Let's remember the employee table and the members table:

employee				
<u>id_employee</u>	firstname	lastname	phone	mail
1	Big	BOSS	936854270	big.boss@company.com
2	John	DOE	936854271	john.doe@company.com
3	Linus	TORVALDS	936854272	linus.torvalds@company.com
4	Jimmy	WALES	936854273	jimmy.wales@company.com
5	Larry	PAGE	936854274	larry.page@company.com
6	Max	THE GOOGLER	936854275	max.the- googler@company.com
7	Jenny	THE WIKIPEDIAN	936854276	jenny.the- wikipedian@company.com

members	
<u>#</u> <u>id_employee</u>	<u>#</u> <u>id_project</u>
3	2
2	1
4	3
5	1
2	3
6	1
7	3

The following query returns the employees who are not linked to any project (i.e. the ones there is no relationship for them in the members table):

▪ **Query:**

```
SELECT *
FROM employees e
WHERE NOT EXISTS (
  SELECT m.id_employee
  FROM members m
  WHERE m.id_employee = e.id_employee
);
```

▪ **Result:**

id_employee	firstname	lastname	phone	mail
1	Big	BOSS	936854270	big.boss@company.com

## IS NULL

IS NULL tests if a column is filled. It is often used for foreign key columns.

## FROM clause

The FROM clause defines the tables that are used for the query but it can also join tables. A JOIN builds a *super* table with the columns of two tables to be used for the query. To explain what a join is, we consider two archaic tables without primary keys nor foreign keys:



table_1	
common_value	specific_value_1
red	9999
grey	6666
white	0000
purple	7777
purple	2222
black	8888

table_2	
common_value	specific_value_2
green	HHHHHH
yellow	PPPPPP
black	FFFFFF
red	OOOOOO
red	LLLLLL
blue	RRRRRR

We want to associate values from columns of different tables matching values on a given column in each table.

## FULL OUTER JOIN

A JOIN is made matching a column on a table to a column on the other table. After a FULL OUTER JOIN, for a given value (red), for a given row with this value on one table ([ red | 9999 ]), one row is created for each row that matches on the other table ([ red | OOOOOO ] and [ red | LLLLLL ]). If a value exists in only one table, then a row is created and is completed with NULL columns.

common_value	specific_value
red	9999
red	9999
grey	6666
white	0000
purple	7777
purple	2222
black	8888
green	NULL
yellow	NULL
blue	NULL

```
FROM table_1 FULL OUTER JOIN table_2 ON table_1.common_value = table_2.common_value
```

## RIGHT OUTER JOIN

The RIGHT OUTER JOIN is like the FULL OUTER JOIN but it doesn't create row for values that don't exist on the left table.

```
FROM table_1 RIGHT OUTER JOIN table_2 ON table_1.common_value = table_2.common_value
```

common_value	specific_value
red	9999
red	9999
black	8888
green	NULL
yellow	NULL
blue	NULL

## LEFT OUTER JOIN

The LEFT OUTER JOIN is like the FULL OUTER JOIN but it doesn't create row for values that don't exist on the right table.

```
FROM table_1 LEFT OUTER JOIN table_2 ON table_1.common_value = table_2.common_value
```

common_value	specific_v
red	9999
red	9999
grey	6666
white	0000
purple	7777
purple	2222
black	8888

## INNER JOIN

The INNER JOIN is like the FULL OUTER JOIN but it creates row only for values that exist on both the left table and the right table.

```
FROM table_1 INNER JOIN table_2 ON table_1.common_value = table_2.common_value
```

common_value	specific_value
red	9999
red	9999
black	8888

## Alias

The FROM clause can declare several tables, separated by , and aliases can be defined for table name with the keyword AS, which allows the user to make several joins with the same tables. The following query is equivalent to the INNER JOIN above:

### ■ Query:

```
SELECT *  
FROM table_1 AS t1, table_2 AS t2  
WHERE t1.common_value = t2.common_value
```

The keyword AS can be omitted.

## SELECT clause

The SELECT clause doesn't influence the data processed by the query but the data returned to the user. \* return all the data processed after joining and filtering. Otherwise, the SELECT clause lists expressions separated by , .

The expressions can be a table name, a table name and a column name separated by a dot or simply a column name if it is not ambiguous. The SELECT clause also allows evaluated expressions like addition, subtraction, concatenation, ... An expression can be followed by an alias with the keyword AS. The keyword AS can be omitted.

Here is an example:

■ Query:

```
SELECT reunion.id_reunion, concat(name, ' : ', reunion.description) n, priority AS p, planned * 10 AS plan, duration AS duration
FROM reunion;
```

■ Result:

id_reunion	n	p	plan	reunion_length
1	Planning : We need to plan the project.	A	10	70
2	Progress : What we have done.	C	10	40
3	Change : What we need to change in the project.	B	10	100
4	Presentation : Presentation of the project.	D	0	130
5	Reporting : Explanation to the new beginner.	B	10	70
6	Learning : A new software version has been install...	B	10	130

The expressions can be also the following aggregation functions:

- count(\*): the count of rows returned,
- max(<column\_name>): the greatest value of the column,
- min(<column\_name>): the lowest value of the column.

Here is a new example:

■ Query:

```
SELECT COUNT(*) * 10 AS c, MAX(DATE) AS latest_date, MIN(reunion.DATE) oldest_date
FROM reunion;
```

■ Result:

c	latest_date	oldest_date
60	2009-09-21	2008-03-24

## ORDER BY clause

The ORDER BY clause sorts the rows returned by the query by one or several columns. The sort is done with the first column mentioned. The second column is used to sort the rows which have the same value in the first column and so on. The keywords ASC or DESC can be added after each column. ASC indicates an ascending sort. DESC indicates a descending sort. Default is a descending sort. Let's do two simple requests, the first sorting by only one column and the second sorting by two columns:

■ Query:

```
SELECT *
FROM reunion
ORDER BY priority ASC;
```

■ Result:

id_reunion	name	description	priority	planned	date	hour	duration
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120

#### ■ Query:

```
SELECT *
FROM reunion
ORDER BY priority ASC, duration DESC;
```

#### ■ Result:

id_reunion	name	description	priority	planned	date	hour	duration
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120

## GROUP BY clause

The GROUP BY clause is used for aggregation operations. It gathers the rows into groups, for instance, all the rows that have the same value in a given column. After gathering rows into groups, any aggregation operation is applied on each group instead of a unique big group of rows. As a consequence, an aggregation operation will return as many result as the number of groups. Groups can be formed with all the rows that have the same value for a given column or the same combination of values for several given columns. For instance, we want to know the number of reunions for each type of priority:

#### ■ Query:

```
SELECT COUNT(*) AS NUMBER, priority
FROM reunion
GROUP BY priority;
```

#### ■ Result:

number	priority
1	A
3	B
1	C
1	D

Due to the GROUP BY clause, the aggregation function count(\*) doesn't return a global count but a count for each priority level (A, B, C and D).

#### ■ Query:

```
SELECT COUNT(*) AS NUMBER, planned, duration
```

```
FROM reunion
GROUP BY planned, duration;
```

▪ **Result:**

number	planned	duration
1	0	120
1	1	30
2	1	60
1	1	90
1	1	120

Note that there are four groups with 1 for the column `planned` and there are two groups with 120 for the column `duration`. However, you can see that there is no group with the same combination of values from the two columns.

## HAVING clause

The `HAVING` clause is used with the `GROUP BY` clause. The `HAVING` clause contains a predicate and removes from the returned rows the groups for which the predicate is false. For example, we want to retrieve only the priorities for which there are at least two reunions with the same priority level:

▪ **Query:**

```
SELECT priority
FROM reunion
GROUP BY priority
HAVING COUNT(*) > 1;
```

▪ **Result:**

priority
B

## FETCH FIRST clause

The `FETCH FIRST` clause is used to limit the number of returned rows. Only the first rows are returned. The number of returned rows is the number indicated in the clause.

▪ **Query:**

```
SELECT *
FROM reunion
FETCH FIRST 4 ROWS ONLY;
```

▪ **Result:**

id_reunion	name	description	priority	planned	date	hour	duration
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120

This clause is often used not to return useless rows for test or to improve the performance.

Now you can explore all the data of an already existing database.

## SQL Functions

- COUNT
- AVG
- MIN
- MAX
- SUM

Eg:

```
SELECT 'COUNT(*)' FROM reunion
```

returns the number of rows in the table **reunion**.

---

- See also: [\[\[1\]\] \(http://en.wikibooks.org/wiki/Oracle\\_Programming/SQL\\_Cheatsheet\)](http://en.wikibooks.org/wiki/Oracle_Programming/SQL_Cheatsheet)

Data Manipulation Language is used to modify the records in the database. It never modifies the schema of the database (table features, relationships, ...). It describes three statements: INSERT, UPDATE and DELETE.

## INSERT statement

The exhaustive syntax of the INSERT statement is as follows:

```
INSERT INTO <TABLE name>[ (<COLUMN name>[, <COLUMN name>]*)]
{
  VALUES (<value>[, <value>]*)
|
  SELECT [ALL | DISTINCT] <COLUMN name> [, [ALL | DISTINCT] <COLUMN name>]*
  FROM <table> [[AS | =] <alias> | [[FULL | LEFT | RIGHT] OUTER | INNER] JOIN <table> ON <expression>]
  [, <table> [[AS | =] <alias> | [[FULL | LEFT | RIGHT] OUTER | INNER] JOIN <table> ON <expression>]]*

  [WHERE <predicate> [{AND | OR} <predicate>]*]
  [GROUP BY <COLUMN name> [, <COLUMN name>]*]
  [HAVING <predicate> [{AND | OR} <predicate>]]*
}
[ORDER BY <COLUMN name> [ASC | DESC] [, <COLUMN name> [ASC | DESC]]*]
[LIMIT <count>]
};
```

The INSERT statement is used to add new records (rows) in a table. For instance, we want to add a new reunion:

- Its primary key is 7,
- Its name is "Job interview",
- Its description is "Meeting with Mr. SPENCER",
- Its priority is B,
- Its planned,
- Its date is on October 28, 2009,
- Its hour is 18:30:00,
- Its duration is 30,
- Its office technical id is 23,
- There is no pdf report.

■ The table before the statement:

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528

■ Query:

```
INSERT INTO reunion (id_reunion, name, description, priority, planned, DATE, HOUR, duration, id_office, pdf_report)
VALUES (7, 'Job interview', 'Meeting with Mr. SPENCER', B, 1, 2009-10-28, 18:30:00, 30, 23, NULL);
```

■ The table after the statement:

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	

The INTO clause contains the name of the table where the record needs to be inserted. It can be followed by a list of columns in brackets. The VALUES clause contains the values to insert in brackets. If the column names are omitted, the VALUES clause must contain as many values as the number of columns of the table. The values are inserted in the table columns in the same order that the order in which the columns have been declared in the table. If the column names are mentioned, there must be as many column names as values. The values are respectively inserted into the named columns. If a column in the table is omitted, a NULL value is inserted instead.

The VALUES clause can be replaced by an inner SELECT statement. In this case, the INSERT statement can insert several rows in the table. For example, we want to plan twice all the reunion with a B priority level, one year later:

- **The table before the statement:**



reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	

■ Query:

```
INSERT INTO reunion (id_reunion, name, description, priority, planned, DATE, HOUR, duration, id_office)
SELECT id_reunion + MAX(id_reunion), name, description, priority, 1, DATE + 0001-00-00, HOUR, duration, id_offi:
FROM reunion
WHERE priority = 'B';
```

■ The table after the statement:

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	
10	Change	What we need to change in the project.	B	1	2009-06-03	9:30:00	90	41	
12	Reporting	Explanation to the new beginner.	B	1	2010-03-15	14:00:00	60	7	
13	Learning	A new software version has been installed.	B	1	2010-09-21	16:00:00	120	11	
14	Job interview	Meeting with Mr. SPENCER	B	1	2010-10-28	18:30:00	30	23	

## UPDATE statement

The exhaustive syntax of the UPDATE statement is as follows:

```

UPDATE <table name>
SET <column name> = <value>[, <column name> = <value>]*
WHERE <predicate> [{AND | OR} <predicate>]*;

```

The UPDATE statement is used to modify already existent records in a table. The UPDATE clause is followed by the table name in which the rows need to be changed. The SET clause is followed by couples of column name and value. The values will be inserted in the given columns. The WHERE clause contains predicates. If the predicates are true

for an existent row, this row will be modified.

For instance, we want to change the date, the hour and the description of the reunion with id 14:

■ **The table before the statement:**

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	
10	Change	What we need to change in the project.	B	1	2009-06-03	9:30:00	90	41	
12	Reporting	Explanation to the new beginner.	B	1	2010-03-15	14:00:00	60	7	
13	Learning	A new software version has been installed.	B	1	2010-09-21	16:00:00	120	11	
14	Job interview	Meeting with Mr. SPENCER	B	1	2010-10-28	18:30:00	30	23	

■ **Query:**

```
UPDATE reunion
SET description = 'Meeting with Ms. JOHNSON', DATE = '2010-02-11', HOUR = '08:00:00'
```

```
WHERE id_reunion = '14';
```

■ The table after the statement:

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	
10	Change	What we need to change in the project.	B	1	2009-06-03	9:30:00	90	41	
12	Reporting	Explanation to the new beginner.	B	1	2010-03-15	14:00:00	60	7	
13	Learning	A new software version has been installed.	B	1	2010-09-21	16:00:00	120	11	
14	Job interview	Meeting with Ms. JOHNSON	B	1	2010-02-11	08:00:00	30	23	

## DELETE statement

The exhaustive syntax of the DELETE statement is as follows:

```
DELETE FROM <table name>  
[WHERE <predicate> [{AND | OR} <predicate>]*];
```

The DELETE statement is used to remove specific rows in a table with conditions. The FROM clause is followed by the table name in which the rows need to be removed. The WHERE clause contains predicates. If the predicates are true for an row, this row will be removed. If the predicates are false for all the rows, the statement do nothing. A DELETE statement without WHERE clause empties the table.

For example, we want to remove all the reunions that last two hours:

■ **The table before the statement:**

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...484654
4	Presentation	Presentation of the project.	D	0	2008-09-11	15:30:00	120	27	
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
6	Learning	A new software version has been installed.	B	1	2009-09-21	16:00:00	120	11	785278...37528
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	
10	Change	What we need to change in the project.	B	1	2009-06-03	9:30:00	90	41	
12	Reporting	Explanation to the new beginner.	B	1	2010-03-15	14:00:00	60	7	
13	Learning	A new software version has been installed.	B	1	2010-09-21	16:00:00	120	11	
14	Job interview	Meeting with Ms. JOHNSON	B	1	2010-02-11	08:00:00	30	23	

■ **Query:**

```
DELETE FROM reunion
WHERE duration = 120;
```

■ The table after the statement:

reunion									
<u>id_reunion</u>	name	description	priority	planned	date	hour	duration	# id_office	pdf_report
1	Planning	We need to plan the project.	A	1	2008-03-24	10:30:00	60	35	48644...846348
2	Progress	What we have done.	C	1	2008-05-12	14:00:00	30	13	9862...15676
3	Change	What we need to change in the project.	B	1	2008-06-03	9:30:00	90	41	34876...4846548
5	Reporting	Explanation to the new beginner.	B	1	2009-03-15	14:00:00	60	7	19739...37718
7	Job interview	Meeting with Mr. SPENCER	B	1	2009-10-28	18:30:00	30	23	
10	Change	What we need to change in the project.	B	1	2009-06-03	9:30:00	90	41	
12	Reporting	Explanation to the new beginner.	B	1	2010-03-15	14:00:00	60	7	
14	Job interview	Meeting with Ms. JOHNSON	B	1	2010-02-11	08:00:00	30	23	

Now you can use an already existing database schema to handle your own data.

Data Definition Language is used to modify the schema of the database. It will never impact the user rights for the database. Otherwise, it can erase records in some tables. It describes three statements: CREATE, ALTER and DROP.

## CREATE statement

The exhaustive syntax of the CREATE statement for the tables is as follows:

```
CREATE TABLE <TABLE name>
(<COLUMN name> <COLUMN type>[ NOT NULL][ PRIMARY KEY | DEFAULT <value>][, <COLUMN name> <COLUMN type>[ NOT NULL]
[, [ CONSTRAINT <CONSTRAINT name>]
{
  PRIMARY KEY (<COLUMN name>[, <COLUMN name>]*)
|
  UNIQUE ([VALUE|<COLUMN name>[, <COLUMN name>]*)
|
  FOREIGN KEY (<COLUMN name>[, <COLUMN name>]*) REFERENCES <TABLE name> (<COLUMN name>[, <COLUMN name>]*) [ ON
|
  CHECK (<predicate>[{ AND | OR } <predicate>]*)
```

```

}
]*
);

```

The CREATE statement is used to create a new table with no record. Let's create the table `office`. The records in the `office` table will contain a technical id, the name of the office, a description, the number of available places, the availability and the date for the next office security control:

▪ **Query:**

```

CREATE TABLE office
(
  id_office INTEGER PRIMARY KEY NOT NULL,
  name VARCHAR(20) NOT NULL,
  description VARCHAR(255),
  place_number INTEGER NOT NULL,
  available SMALLINT NOT NULL DEFAULT 1,
  next_inspection DATE NOT NULL
);

```

▪ **The table after the statement:**

office	
<u>id_office</u>	INTEGER
name	VARCHAR(20)
description	VARCHAR(255)
place_number	INTEGER
available	SMALLINT
next_inspection	DATE

Now the table `office` can be used and filled as the tables `reunion`, `employee`, `project` and `members`:

office					
<u>id_office</u>	name	description	place_number	available	next_inspection
1	Show room		100	1	2011-03-24
2	Big room	The biggest room.	200	1	2010-06-03
3	Open space	The developer open space.	50	1	2011-03-15
4	Hall	The entrance.	20	1	2010-10-28
5	Reunion room		20	1	2010-05-12
6	Actual office	This office is under construction.	5	0	2010-06-03
7	Temporary office	The office used while the actual is under construction.	5	1	2011-03-15
8	Coffee machine	The room where you can pause.	5	1	2011-02-11

The statement starts with CREATE TABLE, to indicate that what we want to create is a table. It's followed by the name of the table (i.e. `office`). The name of the table is followed by parentheses which describe all the columns of the table. The descriptions of the columns are separated by a comma. Each description contains the column name (for instance, `id_office`), the column type (INTEGER, VARCHAR, CHAR, DATE, etc...), an optional nullability information (nothing to indicate that the column can be null or NOT NULL to indicate that the column can't be null) and the optional keyword DEFAULT followed by a default value or the optional keyword PRIMARY KEY to indicate that the column is a primary key. If no default value is defined, NULL is the default value. If NOT NULL is defined,

the column can't have NULL as default value.

You can see that the column `id_office` has been defined as a primary key, the column `description` can be null and the column `available` has 1 as default value.

## ALTER statement

The exhaustive syntax of the ALTER statement for the tables is as follows:

```
ALTER TABLE <TABLE name>
{
  ADD[ COLUMN ] <COLUMN name> <COLUMN type>[ NOT NULL][ PRIMARY KEY | DEFAULT <value>]
|
  ALTER[ COLUMN ] <COLUMN name>[ SET DEFAULT <DEFAULT option>| DROP DEFAULT ]
|
  DROP[ COLUMN ] <COLUMN name>
|
  ADD[ CONSTRAINT <CONSTRAINT name>]
  {
    PRIMARY KEY (<COLUMN name>[, <COLUMN name>]*)
  |
    UNIQUE ([VALUE|<COLUMN name>[, <COLUMN name>]*)
  |
    FOREIGN KEY (<COLUMN name>[, <COLUMN name>]*) REFERENCES <TABLE name> (<COLUMN name>[, <COLUMN name>]*)[ ON I
  |
    CHECK (<predicate>[{ AND | OR } <predicate>]*)
  }
|
  DROP CONSTRAINT <CONSTRAINT name>
};
```

The ALTER statement is used to modify a table. It can be used on a table with records in it.

### ADD CONSTRAINT clause

This clause allows to add a constraint on the table as it could be done at the table creation time. Let's add a unicity constraint on both the name and the description of the office:

- **Query:**

```
ALTER TABLE office ADD CONSTRAINT unique_name_and_description UNIQUE (name, description);
```

Now we can not insert a row with the same name and description of an already existing row and we can not update a row with the same name and description of another row. However, we can insert a row with only the same name or only the same description.

### DROP CONSTRAINT clause

This clause allows to remove an existing constraint on the table by its name. Let's remove the preceding unicity constraint on both the name and the description of the office:

- **Query:**

```
ALTER TABLE office DROP CONSTRAINT unique_name_and_description;
```

Now we can insert a row with the same name and description of an already existing row and we can update a row with the same name and description of another row once again.



## ADD COLUMN clause

Let's add a new column `has_video_projector` to indicate if we can project a slideshow:

- The table before the statement:

office					
<u>id_office</u>	name	description	place_number	available	next_inspection
1	Show room		100	1	2011-03-24
2	Big room	The biggest room.	200	1	2010-06-03
3	Open space	The developer open space.	50	1	2011-03-15
4	Hall	The entrance.	20	1	2010-10-28
5	Reunion room		20	1	2010-05-12
6	Actual office	This office is under construction.	5	0	2010-06-03
7	Temporary office	The office used while the actual is under construction.	5	1	2011-03-15
8	Coffee machine	The room where you can pause.	5	1	2011-02-11

- Query:

```
ALTER TABLE office ADD has_video_projector SMALLINT DEFAULT 0;
```

- The table after the statement:

office						
<u>id_office</u>	name	description	place_number	available	next_inspection	has_video_projector
1	Show room		100	1	2011-03-24	0
2	Big room	The biggest room.	200	1	2010-06-03	0
3	Open space	The developer open space.	50	1	2011-03-15	0
4	Hall	The entrance.	20	1	2010-10-28	0
5	Reunion room		20	1	2010-05-12	0
6	Actual office	This office is under construction.	5	0	2010-06-03	0
7	Temporary office	The office used while the actual is under construction.	5	1	2011-03-15	0
8	Coffee machine	The room where you can pause.	5	1	2011-02-11	0

The column `has_video_projector` has been added at the end. The column has been filled with the default value.

## DROP COLUMN clause

Now let's remove the column `next_inspection`:

- The table before the statement:

office						
<u>id_office</u>	name	description	place_number	available	next_inspection	has_video_projector
1	Show room		100	1	2011-03-24	0
2	Big room	The biggest room.	200	1	2010-06-03	0
3	Open space	The developer open space.	50	1	2011-03-15	0
4	Hall	The entrance.	20	1	2010-10-28	0
5	Reunion room		20	1	2010-05-12	0
6	Actual office	This office is under construction.	5	0	2010-06-03	0
7	Temporary office	The office used while the actual is under construction.	5	1	2011-03-15	0
8	Coffee machine	The room where you can pause.	5	1	2011-02-11	0

■ **Query:**

```
ALTER TABLE office DROP COLUMN next_inspection;
```

■ **The table after the statement:**

office					
<u>id_office</u>	name	description	place_number	available	has_video_projector
1	Show room		100	1	0
2	Big room	The biggest room.	200	1	0
3	Open space	The developer open space.	50	1	0
4	Hall	The entrance.	20	1	0
5	Reunion room		20	1	0
6	Actual office	This office is under construction.	5	0	0
7	Temporary office	The office used while the actual is under construction.	5	1	0
8	Coffee machine	The room where you can pause.	5	1	0

The column `next_inspection` has been removed. If you want to remove a column, you need to remove any constraint applied on it (for instance, you could not remove the name or the description column if there is still the `unique_name_and_description` unicity constraint).

## DROP statement

The exhaustive syntax of the DROP statement for the tables is as follows:

```
DROP TABLE <TABLE name>;
```

The DROP statement is used to remove table.

Now you can use a database for any type of data.

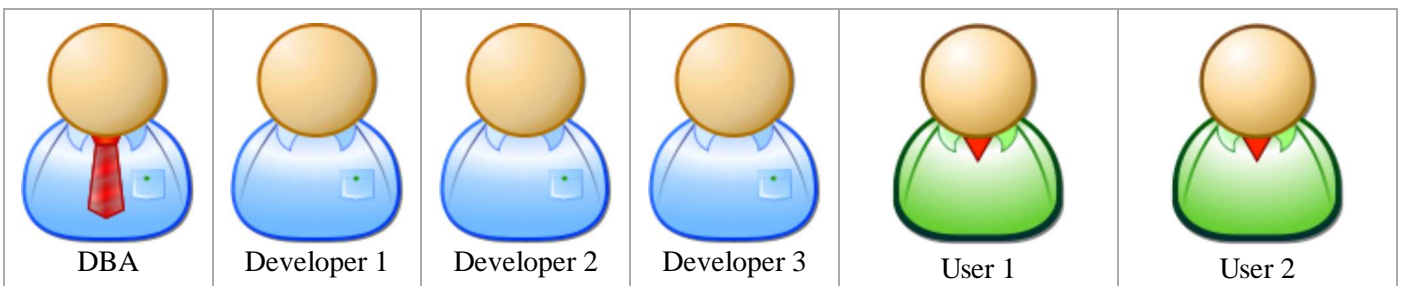
Data Control Language is used to modify the user rights for the database. It describes two statements: GRANT and REVOKE.

## GRANT statement



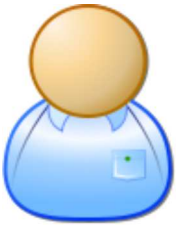
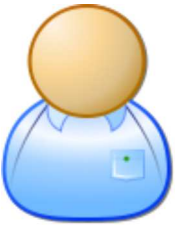
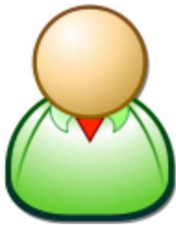
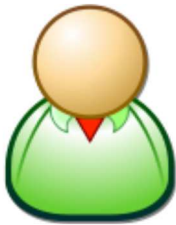
The exhaustive syntax of the GRANT statement is as follows:

```
GRANT
{
  ALL PRIVILEGES ON[ TABLE] <TABLE OR VIEW name>
|
  {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}
  [, {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}]* ON[ TABLE] <TABLE OR VIEW name>
|
  USAGE ON
  {DOMAIN <DOMAIN name>|COLLATION <collation name>|CHARACTER SET <charset name>|TRANSLATION <translation name>}
|
  REFERENCES <COLUMN name>[, <COLUMN name>]* ON <TABLE name>
}
[,
{
  ALL PRIVILEGES ON[ TABLE] <TABLE OR VIEW name>
|
  {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}
  [, {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}]* ON[ TABLE] <TABLE OR VIEW name>
|
  USAGE ON
  {DOMAIN <DOMAIN name>|COLLATION <collation name>|CHARACTER SET <charset name>|TRANSLATION <translation name>}
|
  REFERENCES <COLUMN name>[, <COLUMN name>]* ON <TABLE name>
}
]* TO {PUBLIC|<USER name>}[, {PUBLIC|<USER name>}]* [ WITH GRANT OPTION];
```



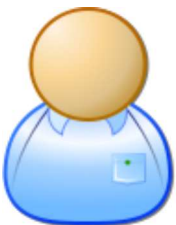
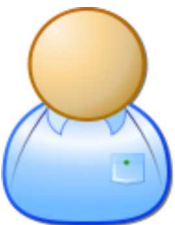
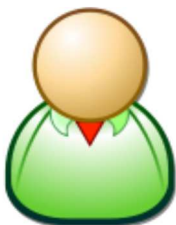
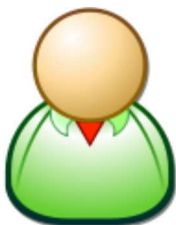
The GRANT statement is used to give a privilege to someone. Any SQL operations are done using a user name. The user name are created by the *database management system*.





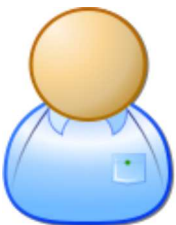
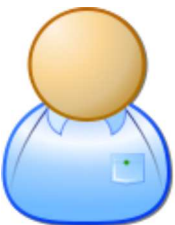
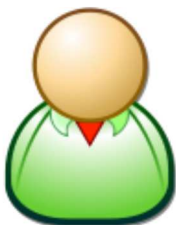
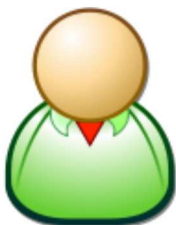
The privileges apply on the tables (i.e. employee, office, etc...), the views, their columns, the domain, the collation, the charset and the translation.

 DBA	 Developer 1	 Developer 2	 Developer 3	 User 1	 User 2
employee office ...	employee office ...	employee office ...	employee office ...	employee office ...	employee office ...

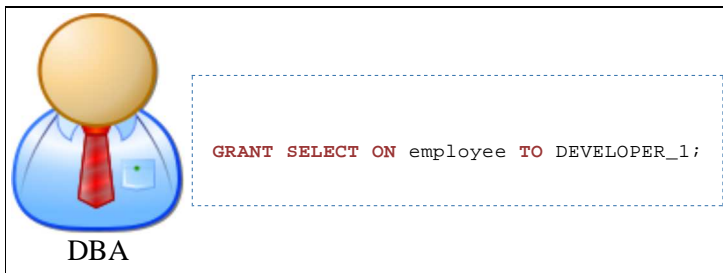
The privileges can allow to process SELECT ("s"), INSERT ("i"), UPDATE ("u") and DELETE ("d") statements (not CREATE, ALTER or DROP statements). Let's say that only the DataBase Administrator has the privileges.

 DBA	 Developer 1	 Developer 2	 Developer 3	 User 1	 User 2
employee s i u d office s i u d ...	employee office ...	employee office ...	employee office ...	employee office ...	employee office ...

For each privilege ("s", "i", "u" and "d"), there is also a meta-privilege ("S", "I", "U" and "D") : a user can send a privilege to another user. Let's say that only the DataBase Administrator has the meta-privileges.

 DBA	 Developer 1	 Developer 2	 Developer 3	 User 1	 User 2
employee S I U D s i u d office S I U D s i u d ...	employee office ...	employee office ...	employee office ...	employee office ...	employee office ...

The DBA wants to allow DEVELOPER\_1 to select columns on the table `employee`:

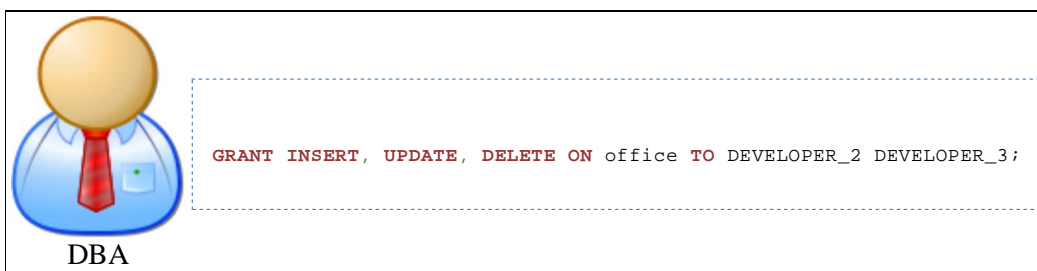


The rights for DEVELOPER\_1 have changed:



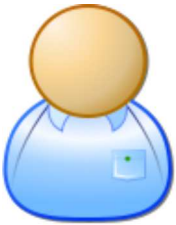
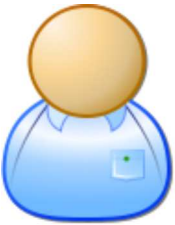
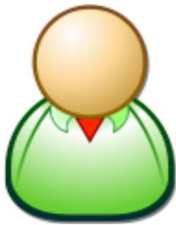
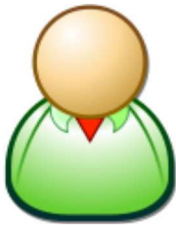
DBA	Developer 1	Developer 2	Developer 3	User 1	User 2																										
<table border="1"> <tr><td>employee</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> ...	employee	S I U D	s i u d	office	S I U D	s i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee	s	office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office	
employee																															
S I U D																															
s i u d																															
office																															
S I U D																															
s i u d																															
employee																															
s																															
office																															
employee																															
office																															
employee																															
office																															
employee																															
office																															
employee																															
office																															

`SELECT` indicates that we want to sent the `SELECT` privilege. The keyword `ON` followed by `employee` indicates that the privilege applies on the table `employee`. The keyword `TO` followed by `DEVELOPER_1` indicates that the privilege has been sent to `DEVELOPER_1`.

The DBA wants to allow `DEVELOPER_2` and `DEVELOPER_3` to insert, update and delete rows on the table `office`:




The rights for `DEVELOPER_2` and `DEVELOPER_3` have changed:



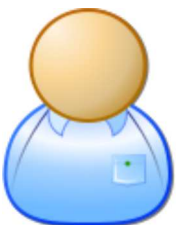
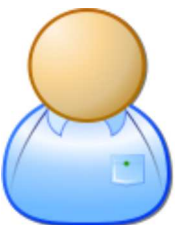


 DBA	 Developer 1	 Developer 2	 Developer 3	 User 1	 User 2																										
<table border="1"> <tr><td>employee</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> ...	employee	S I U D	s i u d	office	S I U D	s i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee	s	office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>i u d</td></tr> </table> ...	employee		office	i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>i u d</td></tr> </table> ...	employee		office	i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office	
employee																															
S I U D																															
s i u d																															
office																															
S I U D																															
s i u d																															
employee																															
s																															
office																															
employee																															
office																															
i u d																															
employee																															
office																															
i u d																															
employee																															
office																															
employee																															
office																															

Whereas you can send several privileges on a table to several users at once, you can't send privileges on several tables at once. If you want to send all the privileges (SELECT, INSERT, UPDATE and DELETE), you can replace the list of privileges by the keywords ALL PRIVILEGES.


Now, the DBA wants to allow USER\_1 to insert on the table `employee` and allow him to send this privilege to other users:

 DBA	<pre>GRANT INSERT ON employee TO USER_1 WITH GRANT OPTION;</pre>
--	--

The rights for USER\_1 have changed:

 DBA	 Developer 1	 Developer 2	 Developer 3	 User 1	 User 2																											
<table border="1"> <tr><td>employee</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> ...	employee	S I U D	s i u d	office	S I U D	s i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee	s	office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>i u d</td></tr> </table> ...	employee		office	i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>i u d</td></tr> </table> ...	employee		office	i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>S</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee	S	s	office		<table border="1"> <tr><td>employee</td></tr> <tr><td></td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> </table> ...	employee		office	
employee																																
S I U D																																
s i u d																																
office																																
S I U D																																
s i u d																																
employee																																
s																																
office																																
employee																																
office																																
i u d																																
employee																																
office																																
i u d																																
employee																																
S																																
s																																
office																																
employee																																
office																																

The keyword WITH GRANT OPTION indicates that we want to send privileges with the meta-privileges to the user. Now, USER\_1 can send the SELECT privilege on the table `employee` to the other users. Let's say that USER\_1 wants to allow anyone to process SELECT on the table `employee`:



```
GRANT SELECT ON TABLE employee TO PUBLIC;
```

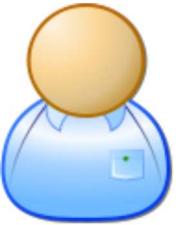
User 1

The rights of all the users have changed:

DBA	Developer 1	Developer 2	Developer 3	User 1	User 2																																																																																																																																
<table border="1"> <tr><td colspan="4">employee</td></tr> <tr><td>S</td><td>I</td><td>U</td><td>D</td></tr> <tr><td>s</td><td>i</td><td>u</td><td>d</td></tr> </table> <table border="1"> <tr><td colspan="4">office</td></tr> <tr><td>S</td><td>I</td><td>U</td><td>D</td></tr> <tr><td>s</td><td>i</td><td>u</td><td>d</td></tr> </table> <p>...</p>	employee				S	I	U	D	s	i	u	d	office				S	I	U	D	s	i	u	d	<table border="1"> <tr><td colspan="4">employee</td></tr> <tr><td>s</td><td></td><td></td><td></td></tr> </table> <table border="1"> <tr><td colspan="4">office</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> <p>...</p>	employee				s				office												<table border="1"> <tr><td colspan="4">employee</td></tr> <tr><td>s</td><td></td><td></td><td></td></tr> </table> <table border="1"> <tr><td colspan="4">office</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td>i</td><td>u</td><td>d</td><td></td></tr> </table> <p>...</p>	employee				s				office								i	u	d		<table border="1"> <tr><td colspan="4">employee</td></tr> <tr><td>s</td><td></td><td></td><td></td></tr> </table> <table border="1"> <tr><td colspan="4">office</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td>i</td><td>u</td><td>d</td><td></td></tr> </table> <p>...</p>	employee				s				office								i	u	d		<table border="1"> <tr><td colspan="4">employee</td></tr> <tr><td>S</td><td></td><td></td><td></td></tr> <tr><td>s</td><td></td><td></td><td></td></tr> </table> <table border="1"> <tr><td colspan="4">office</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> <p>...</p>	employee				S				s				office												<table border="1"> <tr><td colspan="4">employee</td></tr> <tr><td>s</td><td></td><td></td><td></td></tr> </table> <table border="1"> <tr><td colspan="4">office</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> <p>...</p>	employee				s				office											
employee																																																																																																																																					
S	I	U	D																																																																																																																																		
s	i	u	d																																																																																																																																		
office																																																																																																																																					
S	I	U	D																																																																																																																																		
s	i	u	d																																																																																																																																		
employee																																																																																																																																					
s																																																																																																																																					
office																																																																																																																																					
employee																																																																																																																																					
s																																																																																																																																					
office																																																																																																																																					
i	u	d																																																																																																																																			
employee																																																																																																																																					
s																																																																																																																																					
office																																																																																																																																					
i	u	d																																																																																																																																			
employee																																																																																																																																					
S																																																																																																																																					
s																																																																																																																																					
office																																																																																																																																					
employee																																																																																																																																					
s																																																																																																																																					
office																																																																																																																																					



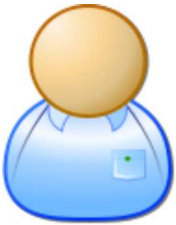
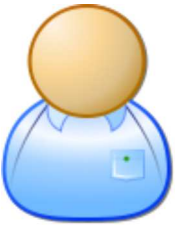
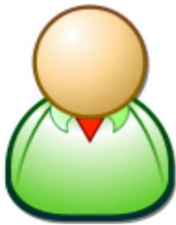
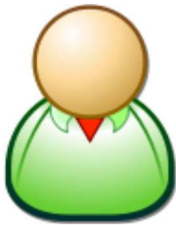
The keyword PUBLIC indicates that we want to send privileges to all the users and the new future ones.

Let's say that DEVELOPER\_3 tries to allow USER\_2 to insert records into the table `office`:



```
GRANT INSERT ON TABLE office TO USER_2;
```

Developer 3

 DBA	 Developer 1	 Developer 2	 Developer 3	 User 1	 User 2																																
<table border="1"> <tr><td>employee</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>S I U D</td></tr> <tr><td>s i u d</td></tr> </table> ...	employee	S I U D	s i u d	office	S I U D	s i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> <tr><td></td></tr> </table> ...	employee	s	office			<table border="1"> <tr><td>employee</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>i u d</td></tr> </table> ...	employee	s	office	i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>S</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td>I</td></tr> <tr><td>i u d</td></tr> </table> ...	employee	S	s	office	I	i u d	<table border="1"> <tr><td>employee</td></tr> <tr><td>S</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> <tr><td></td></tr> </table> ...	employee	S	s	office			<table border="1"> <tr><td>employee</td></tr> <tr><td>s</td></tr> </table> <table border="1"> <tr><td>office</td></tr> <tr><td></td></tr> <tr><td></td></tr> </table> ...	employee	s	office		
employee																																					
S I U D																																					
s i u d																																					
office																																					
S I U D																																					
s i u d																																					
employee																																					
s																																					
office																																					
employee																																					
s																																					
office																																					
i u d																																					
employee																																					
S																																					
s																																					
office																																					
I																																					
i u d																																					
employee																																					
S																																					
s																																					
office																																					
employee																																					
s																																					
office																																					

The operation has been refused because DEVELOPER\_3 hasn't enough privileges.

## Sending privileges on columns

You can send privileges on columns only (only for INSERT and UPDATE):

```
GRANT INSERT (name, description) ON TABLE office TO USER_2;
GRANT UPDATE (id_office, name) ON TABLE office TO USER_2;
```

For INSERT, make all the columns that the user can't fill have default values, are automatically generated or are filled by a trigger before the insertion. Otherwise, the privilege is just useless.

## REVOKE statement


The exhaustive syntax of the REVOKE statement is as follows:

```
REVOKE[ GRANT OPTION FOR]
{
  ALL PRIVILEGES ON[ TABLE] <TABLE OR VIEW name>
  | {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}
  | [, {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}]* ON[ TABLE] <TABLE OR VIEW name>
  | USAGE ON
  | {DOMAIN <DOMAIN name>|COLLATION <collation name>|CHARACTER SET <charset name>|TRANSLATION <translation name>}
  | REFERENCES <COLUMN name>[, <COLUMN name>]* ON <TABLE name>
}
[,
{
  {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}
  | [, {SELECT|DELETE|{INSERT|UPDATE}[ (<COLUMN name>[, <COLUMN name>]*)]}]* ON[ TABLE] <TABLE OR VIEW name>
  | USAGE ON
  | {DOMAIN <DOMAIN name>|COLLATION <collation name>|CHARACTER SET <charset name>|TRANSLATION <translation name>}
  | REFERENCES <COLUMN name>[, <COLUMN name>]* ON <TABLE name>
}
]* FROM {PUBLIC|<USER name>}[, {PUBLIC|<USER name>}]*[ RESTRICT|CASCADE]
```



The REVOKE statement is used to take back privileges granted to someone. This revocation may be more complicated than you expect. To completely remove a privilege to a user, this privilege must be taken back by all the users that have sent the privilege.

For instance, the DBA wants to remove the INSERT and DELETE privileges on the table `employee` to `DEVELOPER_2` and `DEVELOPER_3`:




```
REVOKE INSERT, DELETE ON TABLE office FROM DEVELOPER_2, DEVELOPER_3;
```

DBA

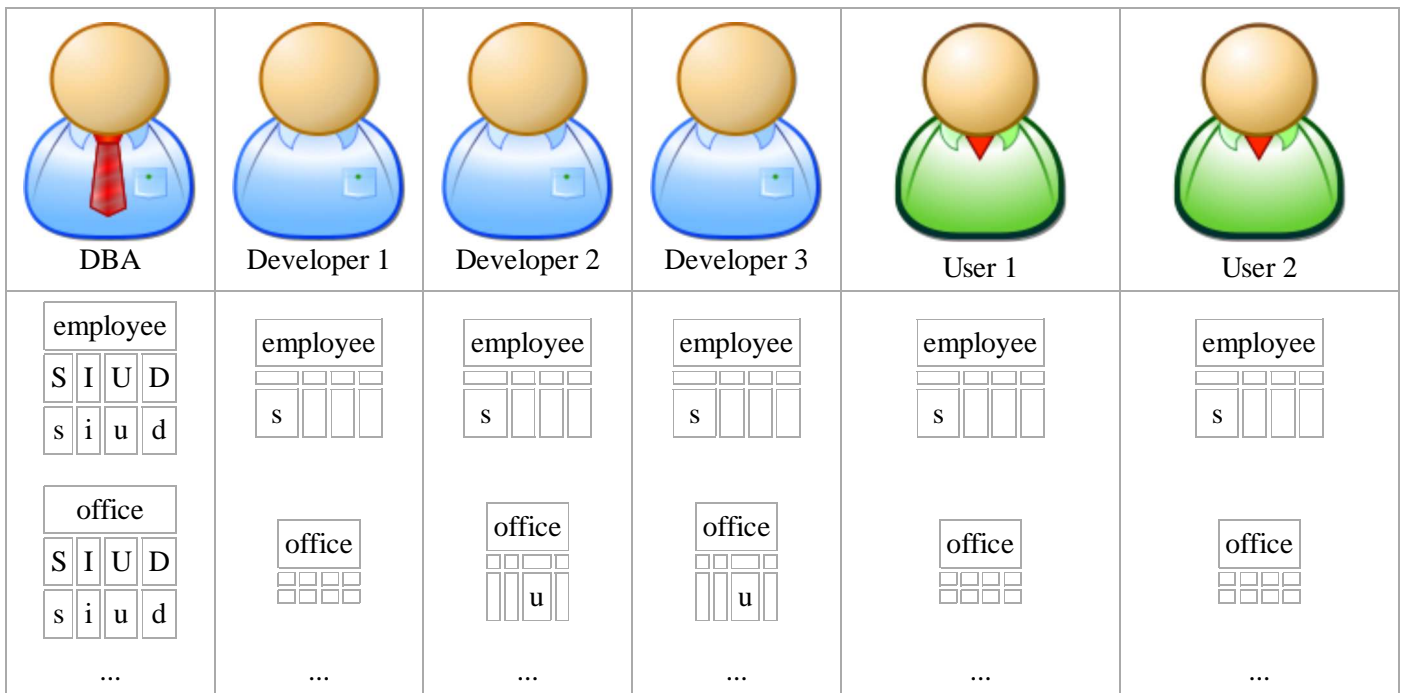
DBA	Developer 1	Developer 2	Developer 3	User 1	User 2
<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 2px;">employee</div> <div style="border: 1px solid gray; padding: 2px;">S I U D</div> <div style="border: 1px solid gray; padding: 2px;">s i u d</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid gray; padding: 2px;">office</div> <div style="border: 1px solid gray; padding: 2px;">S I U D</div> <div style="border: 1px solid gray; padding: 2px;">s i u d</div> </div> <p>...</p>	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 2px;">employee</div> <div style="border: 1px solid gray; padding: 2px;">s</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid gray; padding: 2px;">office</div> </div> <p>...</p>	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 2px;">employee</div> <div style="border: 1px solid gray; padding: 2px;">s</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid gray; padding: 2px;">office</div> <div style="border: 1px solid gray; padding: 2px;">u</div> </div> <p>...</p>	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 2px;">employee</div> <div style="border: 1px solid gray; padding: 2px;">s</div> <div style="border: 1px solid gray; padding: 2px;">s</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid gray; padding: 2px;">office</div> </div> <p>...</p>	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 2px;">employee</div> <div style="border: 1px solid gray; padding: 2px;">s</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid gray; padding: 2px;">office</div> </div> <p>...</p>	

If you remove a privilege to a user who was also sent the related meta-privilege (for example, SELECT privilege to `USER_1`), the operation also removes the meta-privilege. To remove only meta-privileges, add the keywords `GRANT OPTION FOR`:



```
REVOKE GRANT OPTION FOR SELECT ON TABLE employee FROM USER_1;
```

DBA



Now you can administrate a database.

## Result set column

It is not recommended to use `*` in a `SELECT` clause, due to performance issues. You should only return columns you want to use. As a consequence, you should replace any `count(*)` by a count on one column only.

## Avoid the value expressions

Whenever it is possible, avoid the use of value expressions in the `WHERE` clause like this:

```

1. SELECT id_reunion
2. FROM reunion
3. WHERE duration - 60 <= 0;

```

It forces the rDBMS to compute the value for each line, which is very expensive. You should rather compute yourself the values with literals (`0 + 60` in this case):

```

1. SELECT id_reunion
2. FROM reunion
3. WHERE duration <= 60;

```

## Index

If you often select records sorting or filtering by a given column, you may add an index on this column. The database behavior should not change. The index may make query faster. However, don't add useless indexes as it makes insertion a little bit slower.

The exhaustive syntax of the CREATE statement for the indexes is as follows:

```
1. CREATE [ UNIQUE ] INDEX <INDEX name> ON <TABLE name> (<COLUMN name> [ , <COLUMN name> ] * );
```

The keyword UNIQUE indicates that all the group of values in the columns must be distinct.

## Appendices

ACID	An acronym for the 4 properties <i>atomicity</i> , <i>consistency</i> , <i>isolation</i> and <i>durability</i> . Any transaction must conform to them. <i>Atomicity</i> means that either all or no data modification will take place. <i>Consistency</i> ensures that transactions transforms the database from one valid state to another valid state. <i>Isolation</i> requires that transactions will not affect each other, even if they run at the same time. <i>Durability</i> means that the modifications will keep into the database even if the system crashes immediately after the transaction. q.v.: ACID
Attribute	A set of properties (name, datatype, size, ...) used to characterize the data items of entities. A group of attributes constructs an entity-type (or table), i.e.: all values of a certain column must conform to the same attributes. Attributes are optionally complemented by constraints.
Block	Aggregation of one or more physical blocks of a mass device. Usually a block contains numerous rows of one or more tables. Sometimes one row is distributed across several blocks. q.v.: dirty block
Clause	A certain language element as part of a statement. E.g.: the <i>WHERE clause</i> defines search criterias.
Column	A set of values of a single table which resides on the same position within its rows.
Constraint	Similar to attributes constraints define rules at a higher level, data items must conform to. E.g.: nullability, primary and foreign key, uniqueness, default value, user-defined-criterias like <code>STATUS &lt; 10</code> .
Cursor	A cursor is a mechanism by which the rows of a table may be acted on (e.g., returned to a host programming language) one at a time.
Database	A set of tables. Those tables contain user data and the Data Dictionary.
Database Management System (DBMS)	A set of computer programs that controls the creation, maintenance and usage of the database. q.v.: DBMS
Data Dictionary (DD)	A set of predefined tables where the DBMS stores information about all user defined objects (tables, views, constraints, ...).
Data Control Language (DCL)	A class of statements which defines the access rights to data, e.g: <code>GRANT . . . , REVOKE, . . . .</code>
Data Definition Language (DDL)	A class of statements which defines logical and physical design of a database, e.g.: <code>CREATE TABLE . . . .</code>
Data Manipulation Language (DML)	A class of statements which retrieves and manipulates data, e.g.: <code>SELECT . . . , INSERT . . . , UPDATE . . . , DELETE . . . , COMMIT, ROLLBACK.</code>
Dirty Block	A block whose content has changed in memory, but is still not written to disc.
Entity	An identifiable object like an <i>employee</i> or a <i>department</i> . An entity is an instance of an entity-type. Usually there are many instances of a certain entity-type. Every entity is stored in one row. Entities of same entity-type are stored in rows of the same table. So entities are a logical construct and rows a physical implementation.
Entity-type	A group of attributes describing the structure of entities. As entities of same entity-type are stored in rows of the same table it can be said, that an entity-type describes a table. (Many people tend to use the term entity as a synonym for entity-type.)
Expression	A certain language element as part of a statement. It can produce either scalar values or a table.
Foreign key	A value used to reference a primary key. Its value will match a primary key value outside own table.
Index	An index is a construct containing copies of original values and backreferences to their original rows. It's purpose is the provision of a fast access to the original data. To achieve this, an index contains some kind of collocation.  Remark: Indexes are not part of the SQL standard. Nevertheless they are part of nearly every DBMS.

Junction table	If more than one row of table T1 refers to more than one row of table T2 (many-to-many relationship) you need an intermediate table to store this relationship. The rows of the intermediate table contains the primary keys of T1 and T2 as values. q.v.: Junction_table
Normalization	Tables should conform to special rules - namely <i>First-</i> , <i>Second-</i> and <i>Third-Normal Form</i> . The process of rearranging columns over tables is called <i>normalization</i> .
NULL	If no value is stored in the a column of a row, we say it stores the <i>null value</i> . So, NULL is a special value that is used to indicate the absence of any data value. For example it makes a difference whether a temperature is measured and stored as 0 degrees or whether the temperature is not measured and hence not stored. One consequence of the existence of <i>null values</i> is that SQL knows not only the boolean values TRUE and FALSE but also a third one: UNKNOWN.
Predicate	A language element of the <i>WHERE clause</i> which specifies conditions that evaluates to the SQL three-valued logic (true/false/unknown). Predicates are used to find rows.
Primary key	A value or a set of values used to identify a single row uniquely.
Query	An often used statement which retrieves data from the database. It is introduced by the keyword SELECT and usually contains a predicate.
Relationship	A reference between two different or the same entity. References are not implemented as links. They base upon the values of the entities.
Relational Model	A method (and a mathematical theory) to model data as tables (relations), the relationships among each other and all operations on the data.
Row	One record in a table containing information about one single entity. A row has exactly one value for each of its columns - in accordance with <i>First Normal Form</i> . This value may be NULL.
Statement	A single command which is executed by the DBMS. There are 3 main classes of statements: DML, DDL and DCL.
Table (=Relation)	A set of rows of a certain entity-type, i.e. all rows of a certain table have the same structure.
Three Value Logic (3VL)	SQL knows three boolean values: TRUE, FALSE and UNKNOWN. See: NULL. q.v.: 3VL
Transaction	A logical unit of work consisting of one or more modifications to the database. The ACID criterium must be achieved. A transaction is either saved by the COMMIT statement or completely canceled by the ROLLBACK statement.
Value	Implementation of a single data item within a certain column of a certain row. (You can think of a cell within a spreadsheet.)
View	A virtual table containing only its definition and no real data. The definition consists of a query to one or more real tables or views. Queries to the view are processed as queries to the underlying real tables.

**Some of the above terms correlate to each other at the logical and implementation level.**

Logical Design	Implementation
entity-type	table
entity	row
	column
data item	value

**SQL return codes** are used for the diagnosis of programming failures as a result of SQL calls by DB2 programs. An important feature of DB2 programs is the error processing. The error diagnostic containing the SQL return code is held in the field **SQLCODE** within the DB2 SQLCA block.

# SQLCA

The SQL communications area (SQLCA) structure is used within the DB2 program to return error information to the application program. This information in the SQLCA and the SQLCODE field is updated after every API call for the SQL statement.

# SQLCODE

The SQLCODE field contains the SQL return code. The code can be zero (0), negative or positive.

```
-----  
 0 means successful execution.  
  
Negative means unsuccessful execution with an error.  
An example is -911 which means a timeout has occurred with a rollback.  
  
Positive means successful execution with a warning.  
An example is +100 which means no rows found.  
-----  
  
-----  
If you have an SQL code with a letter in it look at Letter Codes below.  
-----
```

Here is a more comprehensive list of the SQLCODEs for DB2:

## Zero (Successful)

```
-----  
 0 Successful  
-----
```

## o Successful **Bold text**

## Negative (through -251)

```
-----  
-007 STATEMENT CONTAINS THE ILLEGAL CHARACTER character  
-010 String constant beginning string is NOT TERMINATED.  
-029 INTO Clause Required.  
-060 Invalid type Specification: "Spec"  
-079 QUALIFIER FOR DECLARED GLOBAL TEMPORARY TABLE table-name MUST BE SESSION, NOT qualifier  
-084 Unacceptable SQL Statement.  
-097 THE USE OF LONG VARCHAR OR LONG VARGRAPHIC IS NOT ALLOWED IN THIS CONTEXT  
-101 Statement is Too Long, or Too Complex.  
-102 String constant is too long.  
-103 Literal is an invalid Numeric Literal.  
-104 Illegal Symbol token.  
-105 Invalid String.  
-107 The Name, "NAME" is Too Long, Maximum Allowable size is "SIZE".  
-108 THE NAME name IS QUALIFIED INCORRECTLY  
-109 "CLAUSE" clause is not permitted.  
-110 Invalid Hexadecimal Literal Beginning "STRING"  
-111 A Column Function does not include a column name.  
-112 The Operand of a Column function is Another Column Function.  
-113 Invalid Character found in Name: "NAME", Reason Code- "REASON-CODE".  
-114 THE LOCATION NAME location DOES NOT MATCH THE CURRENT SERVER  
-115 A PREDICATE Is Invalid because the comparison operator "OPERATOR" is followed by a 'parenthesized list' or "P  
or "All" with out a Sub-query  
-117 The number of values in the INSERT does not match the number of columns.  
-118 THE OBJECT TABLE OR VIEW OF THE DELETE OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE  
-119 A COLUMN IDENTIFIED IN A HAVING CLAUSE IS NOT INCLUDED IN THE GROUP BY CLAUSE  
-120 A WHERE CLAUSE, SET CLAUSE, VALUES CLAUSE, OR A SET HOST-VARIABLE STATEMENT INCLUDES A COLUMN FUNCTION  
-121 THE COLUMN name IS IDENTIFIED MORE THAN ONCE IN THE INSERT OR UPDATE OR SET TRANSITION VARIABLE STATEMENT  
-122 A SELECT STATEMENT WITH NO GROUP BY CLAUSE CONTAINS A COLUMN NAME AND A COLUMN FUNCTION IN THE SELECT CLAUSE  
COLUMN NAME IS CONTAINED IN THE SELECT CLAUSE BUT NOT IN THE GROUP BY CLAUSE  
-123 THE PARAMETER IN POSITION n IN THE FUNCTION name MUST BE A CONSTANT OR KEYWORD  
-125 AN INTEGER IN THE ORDER BY CLAUSE DOES NOT IDENTIFY A COLUMN OF THE RESULT  
-126 THE SELECT STATEMENT CONTAINS BOTH AN UPDATE CLAUSE AND AN ORDER BY CLAUSE  
-127 DISTINCT IS SPECIFIED MORE THAN ONCE IN A SUBSELECT  
-128 INVALID USE OF NULL IN A PREDICATE  
-129 THE STATEMENT CONTAINS TOO MANY TABLE NAMES  
-130 THE ESCAPE CLAUSE CONSISTS OF MORE THAN ONE CHARACTER, OR THE STRING PATTERN CONTAINS AN INVALID OCCURRENCE  
-----
```

```

ESCAPE CHARACTER
-131 STATEMENT WITH LIKE PREDICATE HAS INCOMPATIBLE DATA TYPES
-132 AN OPERAND OF value IS NOT VALID
-133 A COLUMN FUNCTION IN A SUBQUERY OF A HAVING CLAUSE IS INVALID BECAUSE ALL COLUMN REFERENCES IN ITS ARGUMENT
NOT CORRELATED TO THE GROUP BY RESULT THAT THE HAVING CLAUSE IS APPLIED TO
-134 IMPROPER USE OF LONG STRING COLUMN column-name OR AN EXPRESSION THAT RESOLVES TO A LONG STRING
-136 SORT CANNOT BE EXECUTED BECAUSE THE SORT KEY LENGTH IS GREATER THAN 4000 BYTES
-137 THE LENGTH RESULTING FROM operation IS GREATER THAN maximum-length
-138 THE SECOND OR THIRD ARGUMENT OF THE SUBSTR FUNCTION IS OUT OF RANGE
-142 THE SQL STATEMENT IS NOT SUPPORTED
-144 INVALID SECTION NUMBER number
-147 ALTER FUNCTION function-name FAILED BECAUSE SOURCE FUNCTIONS CANNOT BE ALTERED
-148 THE SOURCE TABLE source-name CANNOT BE RENAMED OR ALTERED
-150 THE OBJECT OF THE INSERT, DELETE, OR UPDATE STATEMENT IS A VIEW OR TRANSITION TABLE FOR WHICH THE REQUESTED
OPERATION IS NOT PERMITTED
-151 THE UPDATE STATEMENT IS INVALID BECAUSE THE CATALOG DESCRIPTION OF COLUMN column-name INDICATES THAT IT CANNOT
UPDATED
-152 THE DROP clause CLAUSE IN THE ALTER STATEMENT IS INVALID BECAUSE constraint-name IS A constraint-type
-153 THE STATEMENT IS INVALID BECAUSE THE VIEW OR TABLE DEFINITION DOES NOT INCLUDE A UNIQUE NAME FOR EACH COLUMN
-154 THE STATEMENT FAILED BECAUSE VIEW OR TABLE DEFINITION IS NOT VALID
-156 THE STATEMENT DOES NOT IDENTIFY A TABLE
-157 ONLY A TABLE NAME CAN BE SPECIFIED IN A FOREIGN KEY CLAUSE. object-name IS NOT THE NAME OF A TABLE.
-158 THE NUMBER OF COLUMNS SPECIFIED FOR THE VIEW IS NOT THE SAME AS THE NUMBER OF COLUMNS SPECIFIED BY THE SELECT
CLAUSE, OR THE NUMBER OF COLUMNS SPECIFIED IN THE CORRELATION CLAUSE IN A FROM CLAUSE IS NOT THE SAME AS THE
NUMBER OF COLUMNS IN THE CORRESPONDING TABLE, VIEW, TABLE EXPRESSION, OR TABLE FUNCTION
-159 DROP OR COMMENT ON object IDENTIFIES A(N) object-type1 RATHER THAN A(N) object-type2
-160 THE WITH CHECK OPTION CANNOT BE USED FOR THE SPECIFIED VIEW
-161 THE INSERT OR UPDATE IS NOT ALLOWED BECAUSE A RESULTING ROW DOES NOT SATISFY THE VIEW DEFINITION
-164 auth-id1 DOES NOT HAVE THE PRIVILEGE TO CREATE A VIEW WITH QUALIFICATION authorization-ID
-170 THE NUMBER OF ARGUMENTS SPECIFIED FOR function-name IS INVALID
-171 THE DATA TYPE, LENGTH, OR VALUE OF ARGUMENT nn OF function-name IS INVALID
-173 UR IS SPECIFIED ON THE WITH CLAUSE BUT THE CURSOR IS NOT READ-ONLY
-180 THE DATE, TIME, OR TIMESTAMP VALUE value IS INVALID
-181 THE STRING REPRESENTATION OF A DATETIME VALUE IS NOT A VALID DATETIME VALUE
-182 AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE IS INVALID
-183 AN ARITHMETIC OPERATION ON A DATE OR TIMESTAMP HAS A RESULT THAT IS NOT WITHIN THE VALID RANGE OF DATES
-184 AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE CONTAINS A PARAMETER MARKER
-185 THE LOCAL FORMAT OPTION HAS BEEN USED WITH A DATE OR TIME AND NO LOCAL EXIT HAS BEEN INSTALLED
-186 THE LOCAL DATE LENGTH OR LOCAL TIME LENGTH HAS BEEN INCREASED AND EXECUTING PROGRAM RELIES ON THE OLD LENGTH
-187 A REFERENCE TO A CURRENT DATE/TIME SPECIAL REGISTER IS INVALID BECAUSE THE MVS TOD CLOCK IS BAD OR THE MVS PA
IS OUT OF RANGE
-188 THE STRING REPRESENTATION OF A NAME IS INVALID
-189 CCSID ccsid IS UNKNOWN OR INVALID FOR THE DATA TYPE OR SUBTYPE
-190 ATTRIBUTES OF COLUMN column-name IN TABLE table-name ARE NOT COMPATIBLE WITH THE EXISTING COLUMN
-191 A STRING CANNOT BE USED BECAUSE IT IS INVALID MIXED DATA
-197 QUALIFIED COLUMN NAMES IN ORDER BY CLAUSE NOT PERMITTED WHEN UNION OR UNION ALL SPECIFIED
-198 THE OPERAND OF THE PREPARE OR EXECUTE IMMEDIATE STATEMENT IS BLANK OR EMPTY
-199 Illegal use of the specified keyword.
-203 A REFERENCE TO COLUMN column-name IS AMBIGUOUS
-204 Object not defined to DB2.
-205 Column name not in table.
-206 Column does not exist in any table of the SELECT.
-208 THE ORDER BY CLAUSE IS INVALID BECAUSE COLUMN name IS NOT PART OF THE RESULT TABLE
-212 name IS SPECIFIED MORE THAN ONCE IN THE REFERENCING CLAUSE OF A TRIGGER DEFINITION
-214 AN EXPRESSION IN THE FOLLOWING POSITION, OR STARTING WITH position-or-expression-start IN THE clause-type CL
IS NOT VALID. REASON CODE = reason-code
-216 Not the same number of expressions on both sides of the comparison in a SELECT .
-219 THE REQUIRED EXPLANATION TABLE table-name DOES NOT EXIST
-220 THE COLUMN column-name IN EXPLANATION TABLE table-name IS NOT DEFINED PROPERLY
-221 "SET OF OPTIONAL COLUMNS" IN EXPLANATION TABLE table-name IS INCOMPLETE. OPTIONAL COLUMN column-name IS MISS
-222 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE USING cursor-name
-223 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST AN UPDATE HOLE USING cursor-name
-224 FETCH cannot make an INSENSITIVE cursor SENSITIVE.
-225 FETCH STATEMENT FOR cursor-name IS NOT VALID BECAUSE THE CURSOR IS NOT DEFINED AS SCROLL
-228 FOR UPDATE CLAUSE SPECIFIED FOR READ-ONLY CURSOR cursor-name
-229 The locale specified in a SET LOCALE statement was not found.
-240 THE PART CLAUSE OF A LOCK TABLE STATEMENT IS INVALID
-243 SENSITIVE CURSOR cursor-name CANNOT BE DEFINED FOR THE SPECIFIED SELECT STATEMENT
-244 SENSITIVITY sensitivity SPECIFIED ON THE FETCH IS NOT VALID FOR CURSOR cursor-name
-245 THE INVOCATION OF FUNCTION ROUTINE-NAME IS AMBIGUOUS
-250 THE LOCAL LOCATION NAME IS NOT DEFINED WHEN PROCESSING A THREE-PART OBJECT NAME
-251 TOKEN name IS NOT VALID

```

## Negative (-300 to -499)

```

-300 THE STRING CONTAINED IN HOST VARIABLE OR PARAMETER position-number IS NOT NUL-TERMINATED
-301 THE VALUE OF INPUT HOST VARIABLE OR PARAMETER NUMBER position-number CANNOT BE USED AS SPECIFIED BECAUSE OF
DATA TYPE
-302 THE VALUE OF INPUT VARIABLE OR PARAMETER NUMBER position-number IS INVALID OR TOO LARGE FOR THE TARGET COLUMN
THE TARGET VALUE
-303 A VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER position-number BECAUSE THE DATA TYPES ARE NOT COM
-304 A VALUE WITH DATA TYPE data-type1 CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE
OF THE HOST VARIABLE IN POSITION position-number WITH DATA TYPE data-type2
-305 THE NULL VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER position-number BECAUSE NO INDICATOR VARIABLE
SPECIFIED
-309 A PREDICATE IS INVALID BECAUSE A REFERENCED HOST VARIABLE HAS THE NULL VALUE
-310 DECIMAL HOST VARIABLE OR PARAMETER number CONTAINS NON-DECIMAL DATA
-311 THE LENGTH OF INPUT HOST VARIABLE NUMBER position-number IS NEGATIVE OR GREATER THAN THE MAXIMUM

```

```

-312 variable-name IS AN UNDEFINED OR UNUSABLE HOST VARIABLE OR IS USED IN A DYNAMIC SQL STATEMENT OR A TRIGGER
DEFINITION
-313 THE NUMBER OF HOST VARIABLES SPECIFIED IS NOT EQUAL TO THE NUMBER OF PARAMETER MARKERS
-314 THE STATEMENT CONTAINS AN AMBIGUOUS HOST VARIABLE REFERENCE
-327 THE ROW CANNOT BE INSERTED BECAUSE IT IS OUTSIDE THE BOUND OF THE PARTITION RANGE FOR THE LAST PARTITION
-330 A STRING CANNOT BE USED BECAUSE IT CANNOT BE TRANSLATED. REASON reason-code, CHARACTER code-point, HOST VARIABLE
position-number
-331 A STRING CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE IT CANNOT BE TRANSLATED. REASON reason-code, CHARACTER
code-point, POSITION position-number
-332 CHARACTER CONVERSION BETWEEN CCSID from-ccsid TO to-ccsid REQUESTED BY reason-code IS NOT SUPPORTED
-333 THE SUBTYPE OF A STRING VARIABLE IS NOT THE SAME AS THE SUBTYPE KNOWN AT BIND TIME AND THE DIFFERENCE CANNOT
RESOLVED BY TRANSLATION
-338 AN ON CLAUSE IS INVALID
-339 THE SQL STATEMENT CANNOT BE EXECUTED FROM AN ASCII BASED DRDA APPLICATION REQUESTOR TO A V2R2 DB2 SUBSYSTEM
-350 INVALID SPECIFICATION OF A LARGE OBJECT COLUMN
-351 AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION position-number OF THE SELECT-LIST
-352 AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION position-number OF THE INPUT-LIST
-355 A LOB COLUMN IS TOO LARGE TO BE LOGGED
-359 THE RANGE OF VALUES FOR THE IDENTITY COLUMN IS EXHAUSTED
-372 ONLY ONE ROWID OR IDENTITY COLUMN IS ALLOWED IN A TABLE
-373 DEFAULT CANNOT BE SPECIFIED FOR IDENTITY COLUMN column-name
-390 THE FUNCTION function-name, SPECIFIC NAME specific-name, IS NOT VALID IN THE CONTEXT IN WHICH IT OCCURS
-392 SQLDA PROVIDED FOR CURSOR cursor HAS BEEN CHANGED FROM THE PREVIOUS FETCH
-396 object-type object-name ATTEMPTED TO EXECUTE AN SQL STATEMENT DURING FINAL CALL PROCESSING
-397 THE OPTION GENERATED IS SPECIFIED WITH A COLUMN THAT IS NOT A ROW ID OR DISTINCT TYPE BASED ON A ROW ID
-399 INVALID VALUE ROWID WAS SPECIFIED
-400 THE CATALOG HAS THE MAXIMUM NUMBER OF USER DEFINED INDEXES
-401 THE OPERANDS OF AN ARITHMETIC OR COMPARISON OPERATION ARE NOT COMPARABLE
-402 AN ARITHMETIC FUNCTION OR OPERATOR arith-fop IS APPLIED TO CHARACTER OR DATETIME DATA
-404 THE SQL STATEMENT SPECIFIES A STRING THAT IS TOO LONG
-405 THE NUMERIC LITERAL literal CANNOT BE USED AS SPECIFIED BECAUSE IT IS OUT OF RANGE
-406 A CALCULATED OR DERIVED NUMERIC VALUE IS NOT WITHIN THE RANGE OF ITS OBJECT COLUMN
-407 AN UPDATE, INSERT, OR SET VALUE IS NULL, BUT THE OBJECT COLUMN column-name CANNOT CONTAIN NULL VALUES
-408 THE VALUE IS NOT COMPATIBLE WITH THE DATA TYPE OF ITS TARGET
-409 INVALID OPERAND OF A COUNT FUNCTION
-410 THE FLOATING POINT LITERAL literal CONTAINS MORE THAN 30 CHARACTERS
-411 CURRENT SQLID CANNOT BE USED IN A STATEMENT THAT REFERENCES REMOTE OBJECTS
-412 THE SELECT CLAUSE OF A SUBQUERY SPECIFIES MULTIPLE COLUMNS
-413 OVERFLOW OCCURRED DURING NUMERIC DATA TYPE CONVERSION
-414 A LIKE PREDICATE IS INVALID BECAUSE THE FIRST OPERAND IS NOT A STRING
-415 THE CORRESPONDING COLUMNS, column-number, OF THE OPERANDS OF A UNION OR A UNION ALL DO NOT HAVE COMPARABLE CO
DESCRIPTIONS
-416 AN OPERAND OF A UNION CONTAINS A LONG STRING COLUMN
-417 A STATEMENT STRING TO BE PREPARED INCLUDES PARAMETER MARKERS AS THE OPERANDS OF THE SAME OPERATOR
-418 A STATEMENT STRING TO BE PREPARED CONTAINS AN INVALID USE OF PARAMETER MARKERS
-419 THE DECIMAL DIVIDE OPERATION IS INVALID BECAUSE THE RESULT WOULD HAVE A NEGATIVE SCALE
-420 THE VALUE OF A STRING ARGUMENT WAS NOT ACCEPTABLE TO THE function-name FUNCTION
-421 THE OPERANDS OF A UNION OR UNION ALL DO NOT HAVE THE SAME NUMBER OF COLUMNS
-423 INVALID VALUE FOR LOCATOR IN POSITION position-#
-426 DYNAMIC COMMIT NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED
-427 DYNAMIC ROLLBACK NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED
-430 routine-type routine-name (SPECIFIC NAME specific-name) HAS ABNORMALLY TERMINATED
-433 VALUE value IS TOO LONG
-435 AN INVALID SQLSTATE sqlstate IS SPECIFIED IN THE FUNCTION RAISE sql_error OR IN A SIGNAL SQLSTATE STATEMENT
-438 APPLICATION RAISED ERROR WITH DIAGNOSTIC TEXT: text
-440 NO routine-type BY THE NAME routine-name HAVING COMPATIBLE ARGUMENTS WAS FOUND
-441 INVALID USE OF 'DISTINCT' OR 'ALL' WITH SCALAR FUNCTION function-name
-443 ROUTINE routine-name (SPECIFIC NAME specific-name) HAS RETURNED AN ERROR SQLSTATE WITH DIAGNOSTIC TEXT msg-tes
-444 USER PROGRAM name COULD NOT BE FOUND
-449 CREATE OR ALTER STATEMENT FOR FUNCTION OR PROCEDURE routine-name CONTAINS AN INVALID FORMAT OF THE EXTERNAL N
CLAUSE OR IS MISSING THE
-450 USER-DEFINED FUNCTION OR STORED PROCEDURE name, PARAMETER NUMBER parmnum, OVERLAYED STORAGE BEYOND ITS DECLAR
LENGTH.
-451 THE data-item DEFINITION, IN THE CREATE FUNCTION FOR function-name CONTAINS DATA TYPE type WHICH IS NOT
APPROPRIATE FOR AN EXTERNAL FUNCTION WRITTEN IN THE GIVEN LANGUAGE
-453 THERE IS A PROBLEM WITH THE RETURNS CLAUSE IN THE CREATE FUNCTION STATEMENT FOR function-name
-454 THE SIGNATURE PROVIDED IN THE CREATE FUNCTION STATEMENT FOR function-name MATCHES THE SIGNATURE OF SOME OTHER
FUNCTION ALREADY EXISTING IN THE SCHEMA
-455 IN CREATE FUNCTION FOR function-name, THE SCHEMA NAME schema-name1 PROVIDED FOR THE SPECIFIC NAME DOES NOT MA
THE SCHEMA NAME schema-name2 OF THE FUNCTION
-456 IN CREATE FUNCTION FOR function-name, THE SPECIFIC NAME specific-name ALREADY EXISTS IN THE SCHEMA
-457 A FUNCTION OR DISTINCT TYPE CANNOT BE CALLED name SINCE IT IS RESERVED FOR SYSTEM USE
-458 IN A REFERENCE TO FUNCTION function-name BY SIGNATURE, A MATCHING FUNCTION COULD NOT BE FOUND
-461 A VALUE WITH DATA TYPE source-data-type CANNOT BE CAST TO TYPE target-data-type
-463 EXTERNAL ROUTINE routine-name (SPECIFIC NAME specific-name) HAS RETURNED AN INVALID SQLSTATE sqlstate, WITH
DIAGNOSTIC TEXT text
-469 SQL CALL STATEMENT MUST SPECIFY AN OUTPUT HOST VARIABLE FOR PARAMETER number
-470 SQL CALL STATEMENT SPECIFIED A NULL VALUE FOR INPUT PARAMETER number, BUT THE STORED PROCEDURE DOES NOT SUPPO
NULL VALUES.
-471 INVOCATION OF FUNCTION OR PROCEDURE name FAILED DUE TO REASON rc
-472 CURSOR cursor-name WAS LEFT OPEN BY EXTERNAL FUNCTION function-name (SPECIFIC NAME specific-name)
-473 A USER DEFINED DATA TYPE CANNOT BE CALLED THE SAME NAME AS A SYSTEM PREDEFINED TYPE (BUILT-IN TYPE)
-475 THE RESULT TYPE type -1 OF THE SOURCE FUNCTION CANNOT BE CAST TO THE RETURNS TYPE type -2 OF THE USER-DEFINED
FUNCTION function-name
-476 REFERENCE TO FUNCTION function-name WAS NAMED WITHOUT A SIGNATURE, BUT THE FUNCTION IS NOT UNIQUE WITHIN ITS
-478 DROP OR REVOKE ON OBJECT TYPE type1 CANNOT BE PROCESSED BECAUSE OBJECT name OF TYPE type2 IS DEPENDENT ON IT
-480 THE PROCEDURE procedure-name HAS NOT YET BEEN CALLED
-482 The procedure returned no locators.
-483 IN CREATE FUNCTION FOR function-name STATEMENT, THE NUMBER OF PARAMETERS DOES NOT MATCH THE NUMBER OF PARAMET
OF THE SOURCE FUNCTION
-487 object-type object-name ATTEMPTED TO EXECUTE AN SQL STATEMENT WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE
NOT SPECIFY THIS ACTION

```



```

-490 NUMBER number DIRECTLY SPECIFIED IN AN SQL STATEMENT IS OUTSIDE THE RANGE OF ALLOWABLE VALUES IN THIS CONTEXT
      (minval, maxval)
-491 CREATE STATEMENT FOR USER-DEFINED FUNCTION function-name MUST HAVE A RETURNS CLAUSE AND: THE EXTERNAL CLAUSE,
      OTHER REQUIRED KEYWORDS; THE RETURN STATEMENT AND PARAMETER NAMES; OR THE SOURCE CLAUSE
-492 THE CREATE FUNCTION FOR function-name HAS A PROBLEM WITH PARAMETER NUMBER number. IT MAY INVOLVE A MISMATCH W
      SOURCE FUNCTION
-495 ESTIMATED PROCESSOR COST OF estimate-amount1 PROCESSOR SECONDS (estimate-amount2 SERVICE UNITS) IN COST CATEG
      cost-category EXCEEDS A RESOURCE LIMIT ERROR THRESHOLD OF limit- amount SERVICE UNITS
-496 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT REFERENCES A RESULT SET THAT WAS NOT CREATED BY THE CURRENT S
-497 THE MAXIMUM LIMIT OF INTERNAL IDENTIFIERS HAS BEEN EXCEEDED FOR DATABASE database-name
-499 CURSOR cursor-name HAS ALREADY BEEN ASSIGNED TO THIS OR ANOTHER RESULT SET FROM PROCEDURE procedure-name.

```

## Negative (-500 to -697)

```

-500 THE IDENTIFIED CURSOR WAS CLOSED WHEN THE CONNECTION WAS DESTROYED
-501 THE CURSOR IDENTIFIED IN A FETCH OR CLOSE STATEMENT IS NOT OPEN
-502 Opening cursor that is already open.
-503 Updating column needs to be specified.
-504 THE CURSOR NAME cursor-name IS NOT DEFINED
-507 THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT OPEN
-508 THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT POSITIONED ON A ROW
-509 THE TABLE IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT THE SAME TABLE DESIGNATED BY THE CURSOR
-510 THE TABLE DESIGNATED BY THE CURSOR OF THE UPDATE OR DELETE STATEMENT CANNOT BE MODIFIED
-511 THE FOR UPDATE CLAUSE CANNOT BE SPECIFIED BECAUSE THE TABLE DESIGNATED BY THE CURSOR CANNOT BE MODIFIED
-512 STATEMENT REFERENCE TO REMOTE OBJECT IS INVALID
-513 THE ALIAS alias-name MUST NOT BE DEFINED ON ANOTHER LOCAL OR REMOTE ALIAS
-514 THE CURSOR cursor-name IS NOT IN A PREPARED STATE
-516 THE DESCRIBE STATEMENT DOES NOT SPECIFY A PREPARED STATEMENT
-517 CURSOR cursor-name CANNOT BE USED BECAUSE ITS STATEMENT NAME DOES NOT IDENTIFY A PREPARED SELECT STATEMENT
-518 THE EXECUTE STATEMENT DOES NOT IDENTIFY A VALID PREPARED STATEMENT
-519 THE PREPARE STATEMENT IDENTIFIES THE SELECT STATEMENT OF THE OPENED CURSOR cursor-name
-525 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT WAS IN ERROR AT BIND TIME FOR SECTION = sectno PACKAGE = pkg;
      CONSISTENCY TOKEN = X'contoken'
-526 THE REQUESTED OPERATION OR USAGE DOES NOT APPLY TO table type TEMPORARY TABLE table name
-530 THE INSERT OR UPDATE VALUE OF FOREIGN KEY constraint-name IS INVALID
-531 PARENT KEY IN A PARENT ROW CANNOT BE UPDATED BECAUSE IT HAS ONE OR MORE DEPENDENT ROWS IN RELATIONSHIP
      constraint-name
-532 THE RELATIONSHIP constraint-name RESTRICTS THE DELETION OF ROW WITH RID X'rid-number'
-533 INVALID MULTIPLE-ROW INSERT
-534 THE PRIMARY KEY CANNOT BE UPDATED BECAUSE OF MULTIPLE-ROW UPDATE
-536 THE DELETE STATEMENT IS INVALID BECAUSE TABLE table-name CAN BE AFFECTED BY THE OPERATION
-537 THE PRIMARY KEY CLAUSE, A FOREIGN KEY CLAUSE, OR A UNIQUE CLAUSE IDENTIFIES COLUMN column-name MORE THAN ONCE
-538 FOREIGN KEY name DOES NOT CONFORM TO THE DESCRIPTION OF A PARENT KEY OF TABLE table-name
-539 TABLE table-name DOES NOT HAVE A PRIMARY KEY
-540 THE DEFINITION OF TABLE table-name IS INCOMPLETE BECAUSE IT LACKS A PRIMARY INDEX OR A REQUIRED UNIQUE INDEX
-542 column-name CANNOT BE A COLUMN OF A PRIMARY KEY, A UNIQUE CONSTRAINT, OR A PARENT KEY BECAUSE IT CAN CONTAIN
      VALUES
-543 A ROW IN A PARENT TABLE CANNOT BE DELETED BECAUSE THE CHECK CONSTRAINT check-constraint RESTRICTS THE DELETIO
-544 THE CHECK CONSTRAINT SPECIFIED IN THE ALTER TABLE STATEMENT CANNOT BE ADDED BECAUSE AN EXISTING ROW VIOLATES
      CHECK CONSTRAINT
-545 THE REQUESTED OPERATION IS NOT ALLOWED BECAUSE A ROW DOES NOT SATISFY THE CHECK CONSTRAINT check-constraint
-546 THE CHECK CONSTRAINT constraint-name IS INVALID
-548 A CHECK CONSTRAINT THAT IS DEFINED WITH column-name IS INVALID
-549 THE statement STATEMENT IS NOT ALLOWED FOR object_type1 object_name BECAUSE THE BIND OPTION DYNAMICRULES(RUN)
      NOT IN EFFECT FOR object_type2
-551 auth-id DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION operation ON OBJECT object-name
-552 auth-id DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION operation
-553 auth-id SPECIFIED IS NOT ONE OF THE VALID AUTHORIZATION IDS
-554 AN AUTHORIZATION ID CANNOT GRANT A PRIVILEGE TO ITSELF
-555 AN AUTHORIZATION ID CANNOT REVOKE A PRIVILEGE FROM ITSELF
-556 authid2 CANNOT HAVE THE privilege PRIVILEGE on_object REVOKED BY authid1 BECAUSE THE REVOKEE DOES NOT POSSESS
      PRIVILEGE OR THE REVOKER DID NOT MAKE THE GRANT
-557 INCONSISTENT GRANT/REVOKE KEYWORD keyword. PERMITTED KEYWORDS ARE keyword-list
-558 INVALID CLAUSE OR COMBINATION OF CLAUSES ON A GRANT OR REVOKE
-559 ALL AUTHORIZATION FUNCTIONS HAVE BEEN DISABLED
-567 bind-type AUTHORIZATION ERROR USING auth-id AUTHORITY PACKAGE = package-name PRIVILEGE = privilege
-571 THE STATEMENT WOULD RESULT IN A MULTIPLE SITE UPDATE
-573 TABLE table-name DOES NOT HAVE A UNIQUE KEY WITH THE SPECIFIED COLUMN NAMES
-574 THE SPECIFIED DEFAULT VALUE OR IDENTITY ATTRIBUTE VALUE CONFLICTS WITH THE DEFINITION OF COLUMN column-name
-577 object-type object-name ATTEMPTED TO MODIFY DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPE
      THIS ACTION
-579 object-type object-name ATTEMPTED TO READ DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPEC
      THIS ACTION
-580 THE RESULT-EXPRESSIONS OF A CASE EXPRESSION CANNOT ALL BE NULL
-581 THE DATA TYPES OF THE RESULT-EXPRESSIONS OF A CASE EXPRESSION ARE NOT COMPATIBLE
-582 THE SEARCH-CONDITION IN A SEARCHED-WHEN-CLAUSE CANNOT BE A QUANTIFIED PREDICATE, IN PREDICATE, OR AN EXISTS
      PREDICATE.
-583 THE USE OF FUNCTION function-name IS INVALID BECAUSE IT IS NOT DETERMINISTIC OR HAS AN EXTERNAL ACTION
-585 THE SCHEMA NAME schema-name CANNOT APPEAR MORE THAN ONCE IN THE CURRENT PATH
-586 THE TOTAL LENGTH OF THE CURRENT PATH SPECIAL REGISTER CANNOT EXCEED 254 CHARACTERS
-587 A LIST OF item-references ARE NOT IN THE SAME FAMILY
-590 PARAMETER NAME parameter-name IS NOT UNIQUE IN THE CREATE FOR ROUTINE routine-name
-592 NOT AUTHORIZED TO CREATE FUNCTIONS OR PROCEDURES IN WLM ENVIRONMENT env-name
-593 NOT NULL MUST BE SPECIFIED FOR ROWID OR DISTINCT TYPE COLUMN column-name
-601 THE NAME OF THE OBJECT TO BE CREATED OR THE TARGET OF A RENAME STATEMENT IS IDENTICAL TO THE EXISTING NAME n
      THE OBJECT TYPE obj-type
-602 TOO MANY COLUMNS SPECIFIED IN A CREATE INDEX
-603 A UNIQUE INDEX CANNOT BE CREATED BECAUSE THE TABLE CONTAINS ROWS WHICH ARE DUPLICATES WITH RESPECT TO THE VA

```

```

OF THE IDENTIFIED COLUMNS
-604 A DATA TYPE DEFINITION SPECIFIES AN INVALID LENGTH, PRECISION, OR SCALE ATTRIBUTE
-607 OPERATION OR OPTION operation IS NOT DEFINED FOR THIS OBJECT
-611 ONLY LOCKMAX 0 CAN BE SPECIFIED WHEN THE LOCK SIZE OF THE TABLESPACE IS TABLESPACE OR TABLE
-612 column-name IS A DUPLICATE COLUMN NAME
-613 THE PRIMARY KEY OR A UNIQUE CONSTRAINT IS TOO LONG OR HAS TOO MANY COLUMNS
-614 THE INDEX CANNOT BE CREATED OR THE LENGTH OF A COLUMN CANNOT BE CHANGED BECAUSE THE SUM OF THE INTERNAL LENGTH
THE IDENTIFIED COLUMNS IS GREATER THAN THE ALLOWABLE MAXIMUM
-615 operation-type IS NOT ALLOWED ON A PACKAGE IN USE
-616 obj-type1 obj-name1 CANNOT BE DROPPED BECAUSE IT IS REFERENCED BY obj-type2 obj-name2
-617 A TYPE 1 INDEX IS NOT VALID FOR TABLE table-name
-618 OPERATION operation IS NOT ALLOWED ON SYSTEM DATABASES
-619 OPERATION DISALLOWED BECAUSE THE DATABASE IS NOT STOPPED
-620 KEYWORD keyword IN stmt type STATEMENT IS NOT PERMITTED FOR A space type SPACE IN THE database type DATABASE
-621 DUPLICATE DBID dbid WAS DETECTED AND PREVIOUSLY ASSIGNED TO database-name
-622 FOR MIXED DATA IS INVALID BECAUSE THE MIXED DATA INSTALL OPTION IS NO
-623 A CLUSTERING INDEX ALREADY EXISTS ON TABLE table-name
-624 TABLE table-name ALREADY HAS A PRIMARY KEY OR UNIQUE KEY CONSTRAINT WITH SPECIFIED COLUMNS
-625 TABLE table-name DOES NOT HAVE AN INDEX TO ENFORCE THE UNIQUENESS OF THE PRIMARY OR UNIQUE KEY
-626 THE ALTER STATEMENT IS NOT EXECUTABLE BECAUSE THE PAGE SET IS NOT STOPPED
-627 THE ALTER STATEMENT IS INVALID BECAUSE THE PAGESSET HAS USER-MANAGED DATA SETS
-628 THE CLAUSES ARE MUTUALLY EXCLUSIVE
-629 SET NULL CANNOT BE SPECIFIED BECAUSE FOREIGN KEY name CANNOT CONTAIN NULL VALUES
-630 THE WHERE NOT NULL SPECIFICATION IS INVALID FOR TYPE 1 INDEXES
-631 FOREIGN KEY name IS TOO LONG OR HAS TOO MANY COLUMNS
-632 THE TABLE CANNOT BE DEFINED AS A DEPENDENT OF table-name BECAUSE OF DELETE RULE RESTRICTIONS
-633 THE DELETE RULE MUST BE delete-rule
-634 THE DELETE RULE MUST NOT BE CASCADE
-635 THE DELETE RULES CANNOT BE DIFFERENT OR CANNOT BE SET NULL
-636 THE PARTITIONING KEYS FOR PARTITION part-num ARE NOT SPECIFIED IN ASCENDING OR DESCENDING ORDER
-637 DUPLICATE keyword KEYWORD
-638 TABLE table-name CANNOT BE CREATED BECAUSE COLUMN DEFINITION IS MISSING
-639 A NULLABLE COLUMN OF A FOREIGN KEY WITH A DELETE RULE OF SET NULL CANNOT BE A COLUMN OF THE KEY OF A PARTITIONED
INDEX
-640 LOCKSIZE ROW CANNOT BE SPECIFIED BECAUSE TABLE IN THIS TABLESPACE HAS TYPE 1 INDEX
-643 CHECK CONSTRAINT EXCEEDS MAXIMUM ALLOWABLE LENGTH
-644 INVALID VALUE SPECIFIED FOR KEYWORD keyword IN stmt-type STATEMENT
-646 TABLE table-name CANNOT BE CREATED IN SPECIFIED TABLE SPACE table-space-name BECAUSE IT ALREADY CONTAINS A TABLE
-647 BUFFERPOOL bp-name CANNOT BE SPECIFIED BECAUSE IT HAS NOT BEEN ACTIVATED
-650 THE ALTER INDEX CANNOT BE EXECUTED, REASON reason
-651 TABLE DESCRIPTION EXCEEDS MAXIMUM SIZE OF OBJECT DESCRIPTOR.
-652 VIOLATION OF INSTALLATION DEFINED EDIT OR VALIDATION PROCEDURE proc-name
-653 TABLE table-name IN PARTITIONED TABLE SPACE tspace-name IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT
BEEN CREATED
-655 THE CREATE OR ALTER STOGROUP IS INVALID BECAUSE THE STORAGE GROUP WOULD HAVE BOTH SPECIFIC AND NON-SPECIFIC
VOLUME IDS
-658 A object-type CANNOT BE DROPPED USING THE statement STATEMENT
-660 INDEX index-name CANNOT BE CREATED OR ALTERED ON PARTITIONED TABLE SPACE tspace-name BECAUSE KEY LIMITS ARE NOT
SPECIFIED
-661 INDEX index-name CANNOT BE CREATED ON PARTITIONED TABLE SPACE tspace-name BECAUSE THE NUMBER OF PART
SPECIFICATIONS IS NOT EQUAL TO THE NUMBER OF PARTITIONS OF THE TABLE SPACE
-662 A PARTITIONED INDEX CANNOT BE CREATED ON A NON-PARTITIONED TABLE SPACE tspace-name
-663 THE NUMBER OF KEY LIMIT VALUES IS EITHER ZERO, OR GREATER THAN THE NUMBER OF COLUMNS IN THE KEY OF INDEX index-name
-665 THE PART CLAUSE OF AN ALTER STATEMENT IS OMITTED OR INVALID
-666 stmt-verb object CANNOT BE EXECUTED BECAUSE function IS IN PROGRESS
-667 THE CLUSTERING INDEX FOR A PARTITIONED TABLE SPACE CANNOT BE EXPLICITLY DROPPED
-668 THE COLUMN CANNOT BE ADDED TO THE TABLE BECAUSE THE TABLE HAS AN EDIT PROCEDURE
-669 THE OBJECT CANNOT BE EXPLICITLY DROPPED. REASON reason-code
-670 THE RECORD LENGTH OF THE TABLE EXCEEDS THE PAGE SIZE LIMIT
-671 THE BUFFERPOOL ATTRIBUTE OF THE TABLE SPACE CANNOT BE ALTERED AS SPECIFIED BECAUSE IT WOULD CHANGE THE PAGE SIZE
OF THE TABLE SPACE
-672 OPERATION DROP NOT ALLOWED ON TABLE table_name
-676 ONLY A 4K PAGE BUFFERPOOL CAN BE USED FOR AN INDEX
-677 INSUFFICIENT VIRTUAL STORAGE FOR BUFFERPOOL EXPANSION
-678 THE LITERAL literal SPECIFIED FOR THE INDEX LIMIT KEY MUST CONFORM TO THE DATA TYPE data-type OF THE CORRESPONDING
COLUMN column-name
-679 THE OBJECT name CANNOT BE CREATED BECAUSE A DROP IS PENDING ON THE OBJECT
-680 TOO MANY COLUMNS SPECIFIED FOR A TABLE, VIEW OR TABLE FUNCTION
-681 COLUMN column-name IN VIOLATION OF INSTALLATION DEFINED FIELD PROCEDURE. RT: return-code, RS: reason-code, MS:
message-token
-682 FIELD PROCEDURE procedure-name COULD NOT BE LOADED
-683 THE SPECIFICATION FOR COLUMN, DISTINCT TYPE, FUNCTION, OR PROCEDURE data-item CONTAINS INCOMPATIBLE CLAUSES
-684 THE LENGTH OF LITERAL LIST BEGINNING string IS TOO LONG
-685 INVALID FIELD TYPE, column-name
-686 COLUMN DEFINED WITH A FIELD PROCEDURE CAN NOT COMPARE WITH ANOTHER COLUMN WITH DIFFERENT FIELD PROCEDURE
-687 FIELD TYPES INCOMPARABLE
-688 INCORRECT DATA RETURNED FROM FIELD PROCEDURE, column-name, msgno
-689 TOO MANY COLUMNS DEFINED FOR A DEPENDENT TABLE
-690 THE STATEMENT IS REJECTED BY DATA DEFINITION CONTROL SUPPORT. REASON reason-code
-691 THE REQUIRED REGISTRATION TABLE table-name DOES NOT EXIST
-692 THE REQUIRED UNIQUE INDEX index-name FOR DDL REGISTRATION TABLE table-name DOES NOT EXIST
-693 THE COLUMN column-name IN DDL REGISTRATION TABLE OR INDEX table-name (index-name) IS NOT DEFINED PROPERLY
-696 THE DEFINITION OF TRIGGER trigger-name INCLUDES AN INVALID USE OF CORRELATION NAME OR TRANSITION TABLE NAME
REASON CODE=reason-code
-697 OLD OR NEW CORRELATION NAMES ARE NOT ALLOWED IN A TRIGGER DEFINED WITH THE FOR EACH STATEMENT CLAUSE. OLD_TABLE
NEW_TABLE NAMES ARE NOT ALLOWED IN A TRIGGER WITH THE BEFORE CLAUSE.

```

## Negative (-713 to -991)

```

-713 THE REPLACEMENT VALUE value FOR special-register IS INVALID
-715 PROGRAM program-name WITH MARK release-dependency-mark FAILED BECAUSE IT DEPENDS ON FUNCTIONS OF THE RELEASE,
WHICH FALLBACK HAS OCCURRED
-716 PROGRAM program-name PRECOMPILED WITH INCORRECT LEVEL FOR THIS RELEASE
-717 bind-type FOR object-type object-name WITH MARK release-dependency-mark FAILED BECAUSE object-type DEPENDS ON
FUNCTIONS OF THE RELEASE FROM WHICH FALLBACK HAS OCCURRED
-718 REBIND OF PACKAGE package-name FAILED BECAUSE IBMREQD OF ibmreqd IS INVALID
-719 BIND ADD ERROR USING auth-id AUTHORITY PACKAGE package-name ALREADY EXISTS
-720 BIND ERROR, ATTEMPTING TO REPLACE PACKAGE = package_name WITH VERSION = version2 BUT THIS VERSION ALREADY EXI
-721 BIND ERROR FOR PACKAGE = pkg-id CONTOKEN = contoken'X IS NOT UNIQUE SO IT CANNOT BE CREATED
-722 bind-type ERROR USING auth-id AUTHORITY PACKAGE package-name DOES NOT EXIST
-723 AN ERROR OCCURRED IN A TRIGGERED SQL STATEMENT IN trigger-name. INFORMATION RETURNED: SQLCODE: sqlerror, SQL
sqlstate, MESSAGE TOKENS token-list, SECTION NUMBER section-number
-724 THE ACTIVATION OF THE object-type OBJECT object-name WOULD EXCEED THE MAXIMUM LEVEL OF INDIRECT SQL CASCADING
-725 THE SPECIAL REGISTER register AT LOCATION location WAS SUPPLIED AN INVALID VALUE
-726 BIND ERROR ATTEMPTING TO REPLACE PACKAGE = package-name. THERE ARE ENABLE OR DISABLE ENTRIES CURRENTLY ASSOCI
WITH THE PACKAGE
-728 DATA TYPE data-type IS NOT ALLOWED IN DB2 PRIVATE PROTOCOL PROCESSING
-729 A STORED PROCEDURE SPECIFYING COMMIT ON RETURN CANNOT BE THE TARGET OF A NESTED CALL STATEMENT
-730 THE PARENT OF A TABLE IN A READ-ONLY SHARED DATABASE MUST ALSO BE A TABLE IN A READ-ONLY SHARED DATABASE
-731 USER-DEFINED DATASET dsname MUST BE DEFINED WITH SHAREOPTIONS(1,3)
-732 THE DATABASE IS DEFINED ON THIS SUBSYSTEM WITH THE ROSHARE READ ATTRIBUTE BUT THE TABLE SPACE OR INDEX SPACE
NOT BEEN DEFINED ON THE OWNING SUBSYSTEM
-733 THE DESCRIPTION OF A TABLE SPACE, INDEX SPACE, OR TABLE IN A ROSHARE READ DATABASE MUST BE CONSISTENT WITH IT
DESCRIPTION IN THE OWNER SYSTEM
-734 THE ROSHARE ATTRIBUTE OF A DATABASE CANNOT BE ALTERED FROM ROSHARE READ
-735 DATABASE dbid CANNOT BE ACCESSED BECAUSE IT IS NO LONGER A SHARED DATABASE
-736 INVALID OBID obid SPECIFIED
-737 IMPLICIT TABLE SPACE NOT ALLOWED
-739 CREATE OR ALTER FUNCTION function-name FAILED BECAUSE FUNCTIONS CANNOT MODIFY DATA WHEN THEY ARE PROCESSED IN
PARALLEL.
-740 FUNCTION name IS DEFINED WITH THE OPTION MODIFIES SQL DATA WHICH IS NOT VALID IN THE CONTEXT IN WHICH IT WAS
INVOKED
-741 A database-type DATABASE IS ALREADY DEFINED FOR MEMBER member-name
-742 DSNDB07 IS THE IMPLICIT WORK FILE DATABASE
-746 THE SQL STATEMENT IN AN EXTERNAL FUNCTION, TRIGGER, OR IN STORED PROCEDURE name VIOLATES THE NESTING SQL
RESTRICTION
-747 TABLE table-name IS NOT AVAILABLE UNTIL THE AUXILIARY TABLES AND INDEXES FOR ITS EXTERNALLY STORED COLUMNS HA
BEEN CREATED
-748 AN INDEX ALREADY EXISTS ON AUXILIARY TABLE table-name
-750 THE SOURCE TABLE source-name CANNOT BE RENAMED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS OR TRIGG
DEFINITIONS
-751 object-type object-name (SPECIFIC NAME specific name) ATTEMPTED TO EXECUTE AN SQL STATEMENT statement THAT IS
ALLOWED
-752 THE CONNECT STATEMENT IS INVALID BECAUSE THE PROCESS IS NOT IN THE CONNECTABLE STATE
-763 INVALID TABLE SPACE NAME
-764 A LOB TABLE SPACE AND ITS ASSOCIATED BASE TABLE SPACE MUST BE IN THE SAME DATABASE
-765 TABLE IS NOT COMPATIBLE WITH DATABASE
-766 THE OBJECT OF A STATEMENT IS AN AUXILIARY TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED
-767 MISSING OR INVALID COLUMN SPECIFICATION FOR INDEX
-768 AN AUXILIARY TABLE ALREADY EXISTS FOR THE SPECIFIED COLUMN OR PARTITION
-769 SPECIFICATION OF CREATE AUX TABLE DOES NOT MATCH THE CHARACTERISTICS OF THE BASE TABLE
-770 TABLE table-name CANNOT HAVE A LOB COLUMN UNLESS IT ALSO HAS A ROWID COLUMN
-771 INVALID SPECIFICATION OF A ROWID COLUMN
-797 ATTEMPT TO CREATE TRIGGER trigger-name WITH AN UNSUPPORTED TRIGGERED SQL STATEMENT
-798 YOU CANNOT INSERT A VALUE INTO A COLUMN THAT IS DEFINED WITH THE OPTION GENERATED ALWAYS COLUMN column-name
-802 EXCEPTION ERROR exception-type HAS OCCURRED DURING operation-type OPERATION ON data-type DATA, POSITION
position-number
-803 Duplicate key on insert or update.
-804 AN ERROR WAS FOUND IN THE APPLICATION PROGRAM INPUT PARAMETERS FOR THE SQL STATEMENT, REASON reason
-805 DBRM or package not found in plan.
-807 ACCESS DENIED: PACKAGE package-name IS NOT ENABLED FOR ACCESS FROM connection-type connection-name
-808 THE CONNECT STATEMENT IS NOT CONSISTENT WITH THE FIRST CONNECT STATEMENT
-811 More than one row retrieved in SELECT INTO.
-812 THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE A BLANK COLLECTION-ID WAS FOUND IN THE CURRENT PACKAGESET SPECI
REGISTER while trying to FORM A QUALIFIED PACKAGE NAME FOR PROGRAM program-name consistency-token USING PLAN.
-815 A GROUP BY OR HAVING CLAUSE IS IMPLICITLY OR EXPLICITLY SPECIFIED IN A SUBSELECT OF A BASIC PREDICATE OR THE
CLAUSE OF AN UPDATE STATEMENT
-817 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE STATEMENT WILL RESULT IN A PROHIBITED UPDATE OPERATION.
-818 Plan and program: timestamp mismatch.
-819 THE VIEW CANNOT BE PROCESSED BECAUSE THE LENGTH OF ITS PARSE TREE IN THE CATALOG IS ZERO
-820 THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE catalog-table CONTAINS A VALUE THAT IS NOT VALID IN THIS RELEAS
-822 THE SQLDA CONTAINS AN INVALID DATA ADDRESS OR INDICATOR VARIABLE ADDRESS
-840 TOO MANY ITEMS RETURNED IN A SELECT OR INSERT LIST
-842 A CONNECTION TO location-name ALREADY EXISTS
-843 THE SET CONNECTION OR RELEASE STATEMENT MUST SPECIFY AN EXISTING CONNECTION
-846 INVALID SPECIFICATION OF AN IDENTITY COLUMN
-867 INVALID SPECIFICATION OF A ROWID COLUMN
-870 THE NUMBER OF HOST VARIABLES IN THE STATEMENT IS NOT EQUAL TO THE NUMBER OF DESCRIPTORS
-872 A VALID CCSID HAS NOT YET BEEN SPECIFIED FOR THIS SUBSYSTEM
-873 DATA ENCODED WITH DIFFERENT ENCODING SCHEMES CANNOT BE REFERENCED IN THE SAME SQL STATEMENT
-874 THE ENCODING SCHEME SPECIFIED FOR THE object-type MUST BE THE SAME AS THE CONTAINING TABLE SPACE OR OTHER
PARAMETERS
-875 operand CANNOT BE USED WITH THE ASCII DATA REFERENCED
-876 'object' CANNOT BE CREATED, REASON 'reason'
-877 CCSID ASCII OR CCSID UNICODE IS NOT ALLOWED FOR THIS DATABASE OR TABLE SPACE
-878 THE PLAN_TABLE USED FOR EXPLAIN CANNOT BE ASCII OR UNICODE
-879 CREATE OR ALTER STATEMENT FOR obj-name CANNOT DEFINE A COLUMN, DISTINCT TYPE, FUNCTION OR STORED PROCEDURE
PARAMETER WITH ENCODING SCHEME encoding-scheme
-880 SAVEPOINT savepoint-name DOES NOT EXIST OR IS INVALID IN THIS CONTEXT
-881 A SAVEPOINT WITH NAME savepoint-name ALREADY EXISTS, BUT THIS SAVEPOINT NAME CANNOT BE REUSED

```

```

-882 SAVEPOINT DOES NOT EXIST
-900 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE APPLICATION PROCESS IS NOT CONNECTED TO AN APPLICATION SERV
-901 UNSUCCESSFUL EXECUTION CAUSED BY A SYSTEM ERROR THAT DOES NOT PRECLUDE THE SUCCESSFUL EXECUTION OF SUBSEQUENT
STATEMENTS
-902 POINTER TO THE ESSENTIAL CONTROL BLOCK (CT/RDA) HAS VALUE 0, REBIND REQUIRED
-904 Unavailable resource. Someone else is locking your data.
-905 UNSUCCESSFUL EXECUTION DUE TO RESOURCE LIMIT BEING EXCEEDED, RESOURCE NAME = resource-name LIMIT = limit-amount
CPU SECONDS (limit-amount2 SERVICE UNITS) DERIVED FROM limit-source
-906 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THIS FUNCTION IS DISABLED DUE TO A PRIOR ERROR
-908 bind-type ERROR USING auth-id AUTHORITY. BIND, REBIND OR AUTO-REBIND OPERATION IS NOT ALLOWED
-909 THE OBJECT HAS BEEN DELETED
-910 THE SQL STATEMENT CANNOT ACCESS AN OBJECT ON WHICH A DROP OR ALTER IS PENDING
-911 Rollback has been done due to Deadlock or timeout.
-913 Deadlock or timeout has occurred.
-917 BIND PACKAGE FAILED
-918 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE A CONNECTION HAS BEEN LOST
-919 A ROLLBACK OPERATION IS REQUIRED
-922 AUTHORIZATION FAILURE: error-type ERROR. REASON reason-code (Authorization needed).
-923 CONNECTION NOT ESTABLISHED: DB2 condition REASON reason-code, TYPE resource-type, NAME resource-name
-924 DB2 CONNECTION INTERNAL ERROR, function-code, return-code, reason-code
-925 COMMIT NOT VALID IN IMS, CICS OR RRSF ENVIRONMENT
-926 ROLLBACK NOT VALID IN IMS, CICS OR RRSF ENVIRONMENT
-927 The language interface was called but no connection had been made.
-929 FAILURE IN A DATA CAPTURE EXIT: token
-939 ROLLBACK REQUIRED DUE TO UNREQUESTED ROLLBACK OF A REMOTE SERVER
-947 THE SQL STATEMENT FAILED BECAUSE IT WILL CHANGE A TABLE DEFINED WITH DATA CAPTURE CHANGES, BUT THE DATA CANNOT
PROPAGATED
-948 DISTRIBUTED OPERATION IS INVALID
-950 THE LOCATION NAME SPECIFIED IN THE CONNECT STATEMENT IS INVALID OR NOT LISTED IN THE COMMUNICATIONS DATABASE
-981 THE SQL STATEMENT FAILED BECAUSE THE RRSF CONNECTION IS NOT IN A STATE THAT ALLOWS SQL OPERATIONS, REASON
reason-code.
-991 CALL ATTACH WAS UNABLE TO ESTABLISH AN IMPLICIT CONNECT OR OPEN TO DB2. RC1= rc1 RC2= rc2
-----

```

## Negative (Smaller than -991)

```

-1760 CREATE PROCEDURE FOR procedure-name MUST HAVE VALID LANGUAGE AND EXTERNAL CLAUSES
-2001 THE NUMBER OF HOST VARIABLE PARAMETERS FOR A STORED PROCEDURE IS NOT EQUAL TO THE NUMBER OF EXPECTED HOST
VARIABLE PARAMETERS. ACTUAL NUMBER sqlданum, EXPECTED NUMBER opnum
-5012 HOST VARIABLE host-variable IS NOT EXACT NUMERIC WITH SCALE ZERO
-20003 GBPCACHE NONE CANNOT BE SPECIFIED FOR TABLESPACE OR INDEX IN GRECP
-20004 8K or 16K BUFFERPOOL PAGESIZE INVALID FOR A WORKFILE OBJECT
-20005 THE INTERNAL ID LIMIT OF limit HAS BEEN EXCEEDED FOR OBJECT TYPE object-type
-20006 LOBS CANNOT BE SPECIFIED AS PARAMETERS WHEN NO WLM ENVIRONMENT IS SPECIFIED
-20008 UNSUPPORTED OPTION keyword SPECIFIED
-20070 AUXILIARY TABLE table-name CANNOT BE CREATED BECAUSE COLUMN column-name IS NOT A LOB COLUMN
-20071 WLM ENVIRONMENT NAME MUST BE SPECIFIED function-name
-20072 csect-name bind-type bind-subtype ERROR USING auth-id AUTHORITY OPERATION IS NOT ALLOWED ON A TRIGGER PACKAGE
package-name
-20073 THE FUNCTION function-name CANNOT BE ALTERED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS
-20074 THE OBJECT object-name CANNOT BE CREATED BECAUSE THE FIRST THREE CHARACTERS ARE RESERVED FOR SYSTEM OBJECTS
-20091 A VIEW NAME WAS SPECIFIED AFTER LIKE IN ADDITION TO THE INCLUDING IDENTITY COLUMN ATTRIBUTES CLAUSE
-20092 A VIEW WAS SPECIFIED FOR LIKE BUT IT INCLUDES A ROWID COLUMN
-20100 AN ERROR OCCURRED WHEN BINDING A TRIGGERED SQL STATEMENT. INFORMATION RETURNED: SECTION NUMBER : section-number
SQLCODE sqlerror, SQLSTATE sqlstate, AND MESSAGE TOKENS token-list
-20101 THE FUNCTION function FAILED WITH REASON rc
-20102 CREATE OR ALTER STATEMENT FOR ROUTINE routine-name SPECIFIED THE option OPTION WHICH IS NOT ALLOWED FOR THE
OF ROUTINE
-20104 AN ATTEMPT TO ALTER A CCSID FROM from-ccsid TO to-ccsid FAILED
-20106 THE CCSID FOR TABLE SPACE OR DATABASE CANNOT BE CHANGED BECAUSE THE TABLE SPACE OR DATABASE ALREADY CONTAINS
TABLE THAT IS REFERENCED IN EXISTING VIEW DEFINITIONS
-20107 HOST VARIABLE OR PARAMETER NUMBER position-number CANNOT BE USED AS SPECIFIED BECAUSE REASON reason
-20108 A RESULT SET CONTAINS AN UNSUPPORTED DATA TYPE IN POSITION NUMBER position-number FOR CURSOR cursor-name OR
BY STORED PROCEDURE procedure-name
-20110 CANNOT IMPLICITLY CONNECT TO A REMOTE SITE WITH A SAVEPOINT OUTSTANDING
-20111 CANNOT ISSUE SAVEPOINT, RELEASE SAVEPOINT, ROLLBACK TO SAVEPOINT FROM A TRIGGER, FROM A USER-DEFINED FUNCTION
OR FROM A GLOBAL TRANSACTION
-20123 CALL TO STORED PROCEDURE procedure FAILED BECAUSE THE RESULT SET RETURNED FOR CURSOR cursor IS SCROLLABLE,
THE CURSOR IS NOT POSITIONED BEFORE THE FIRST ROW
-20124 OPEN CURSOR cursor FAILED BECAUSE THE CURSOR IS SCROLLABLE BUT THE CLIENT DOES NOT SUPPORT THIS
-20125 CALL TO STORED PROCEDURE procedure FAILED BECAUSE THE RESULT SET FOR CURSOR cursor IS SCROLLABLE, BUT THE CLIENT
DOES NOT SUPPORT THIS
-20126 CURSOR cursor IS DEFINED AS SCROLLABLE, BUT THE ENVIRONMENT INVOLVES A HOP SITE
-20127 VALUE SPECIFIED ON FETCH STATEMENT FOR ABSOLUTE OR RELATIVE IS TOO LARGE FOR DRDA
-20129 LOCAL SPECIAL REGISTER IS NOT VALID AS USED
-20200 THE INSTALL_JAR OR REPLACE_JAR PROCEDURE FOR jar-id FAILED AS url COULD NOT BE LOCATED.
-20201 THE INSTALL_JAR, REPLACE_JAR, OR REMOVE_JAR PROCEDURE FOR jar-name FAILED AS THE JAR NAME IS INVALID
-20202 THE REPLACE_JAR OR REMOVE_JAR PROCEDURE FOR jar-name FAILED AS class IS IN USE
-20203 USER DEFINED FUNCTION OR PROCEDURE name HAS A JAVA METHOD WITH AN INVALID SIGNATURE. THE ERROR IS AT OR NEAR
PARAMETER number. THE SIGNATURE IS signature
-20204 THE USER-DEFINED FUNCTION OR PROCEDURE routine-name WAS UNABLE TO MAP TO A SINGLE JAVA METHOD
-20207 THE INSTALL_JAR OR REMOVE_JAR PROCEDURE FOR jar-name SPECIFIED THE USE OF A DEPLOYMENT DESCRIPTOR
-20210 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT WAS AT A LEVEL THAT IS INCOMPATIBLE WITH THE CURRENT VALUE
THE ENCODING BIND OPTION OR SPECIAL REGISTER
-20212 USER-DEFINED ROUTINE name ENCOUNTERED AN EXCEPTION ATTEMPTING TO LOAD JAVA CLASS class-name FROM JAR jar-name
ORIGINAL EXCEPTION: exception-string.
-20213 STORED PROCEDURE procedure-name HAS RETURNED A DYNAMIC RESULT SET OF AN INVALID CLASS. PARAMETER number IS
DB2 RESULT SET
-30000 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT WILL NOT AFFECT THE SUCCESSFUL EXECUTION OF
-----

```

```

SUBSEQUENT COMMANDS OR SQL STATEMENTS: REASON reason-code (sub-code)
-30002 THE SQL STATEMENT CANNOT BE EXECUTED DUE TO A PRIOR CONDITION IN A CHAIN OF STATEMENTS
-30020 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT CAUSED DEALLOCATION OF THE CONVERSATION: REASON,
<reason-code (sub-code)>
-30021 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT WILL AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT
COMMANDS OR SQL STATEMENTS: MANAGER manager AT LEVEL level NOT SUPPORTED ERROR
-30030 COMMIT REQUEST WAS UNSUCCESSFUL, A DISTRIBUTION PROTOCOL VIOLATION HAS BEEN DETECTED, THE CONVERSATION HAS
DEALLOCATED. ORIGINAL SQLCODE=original-sqlcode AND ORIGINAL SQLSTATE=original-sqlstate
-30040 EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL NOT AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT
COMMANDS OR SQL STATEMENTS. REASON reason-code TYPE OF RESOURCE resource-type RESOURCE NAME resource-name PRG
ID pppvvrmm RDBNAME rdbname
-30041 EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMM
AND SQL STATEMENTS. REASON <reason-code> TYPE OF RESOURCE <resource-type> RESOURCE NAME <resource-name> PRG
<pppvvrmm> RDBNAME <rdbname>
-30050 <command-or-sql-statement-type COMMAND OR SQL STATEMENT INVALID WHILE BIND PROCESS IN PROGRESS
-30051 BIND PROCESS WITH SPECIFIED PACKAGE NAME AND CONSISTENCY TOKEN NOT ACTIVE
-30052 PROGRAM PREPARATION ASSUMPTIONS ARE INCORRECT
-30053 OWNER AUTHORIZATION FAILURE
-30060 RDB AUTHORIZATION FAILURE
-30061 RDB NOT FOUND
-30070 command COMMAND NOT SUPPORTED ERROR
-30071 object-type OBJECT NOT SUPPORTED ERROR
-30072 parameter subcode PARAMETER NOT SUPPORTED ERROR
-30073 parameter subcode PARAMETER VALUE NOT SUPPORTED ERROR
-30074 REPLY MESSAGE WITH codepoint (svrcod) NOT SUPPORTED ERROR
-30080 COMMUNICATION ERROR code (subcode)
-30081 prot COMMUNICATION ERROR DETECTED. API=api, LOCATION=loc, FUNCTION=func, ERROR CODES=rc1 rc2 rc3
-30082 CONNECTION FAILED FOR SECURITY REASON reason-code (reason-string)
-30090 REMOTE OPERATION INVALID FOR APPLICATION EXECUTION ENVIRONMENT
-30104 ERROR IN BIND OPTION option AND BIND VALUE value
-30105 BIND OPTION option1 IS NOT ALLOWED WITH BIND OPTION option2

```

## Positive Values (Warnings)

```

+012 THE UNQUALIFIED COLUMN NAME column-name WAS INTERPRETED AS A CORRELATED REFERENCE
+098 A DYNAMIC SQL STATEMENT ENDS WITH A SEMICOLON.
+100 Row not found or end of cursor.
+111 THE SUBPAGES OPTION IS NOT SUPPORTED FOR TYPE 2 INDEXES
+117 THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS
+162 TABLESPACE database-name.tablespace-name HAS BEEN PLACED IN CHECK PENDING
+203 THE QUALIFIED COLUMN NAME column-name WAS RESOLVED USING A NON-UNIQUE OR UNEXPOSED NAME
+204 name IS AN UNDEFINED NAME
+206 column-name IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE
+218 THE SQL STATEMENT REFERENCING A REMOTE OBJECT CANNOT BE EXPLAINED
+219 THE REQUIRED EXPLANATION TABLE table-name DOES NOT EXIST
+220 THE COLUMN column-name IN EXPLANATION TABLE table-name IS NOT DEFINED PROPERLY
+222 Trying to fetch a row within a DELETE statement.
+223 Trying to fetch a row within an UPDATE statement.
+231 FETCH after a BEFORE or AFTER but not on a valid row.
+236 SQLDA INCLUDES integer1 SQLVAR ENTRIES, BUT integer2 ARE REQUIRED FOR integer3 COLUMNS
+237 SQLDA INCLUDES integer1 SQLVAR ENTRIES, BUT integer2 ARE REQUIRED BECAUSE AT LEAST ONE OF THE COLUMNS BEING
DESCRIBED IS A DISTINCT TYPE
+238 SQLDA INCLUDES integer1 SQLVAR ENTRIES, BUT integer2 SQLVAR ENTRIES ARE NEEDED FOR integer3 COLUMNS BECAUSE
LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A LOB
+239 SQLDA INCLUDES integer1 SQLVAR ENTRIES, BUT integer2 ARE REQUIRED FOR integer3 COLUMNS BECAUSE AT LEAST ONE
COLUMNS BEING DESCRIBED IS A DISTINCT TYPE
+304 Value cannot be assigned to this host variable because it is out of range.
+331 THE NULL VALUE HAS BEEN ASSIGNED TO A HOST VARIABLE BECAUSE THE STRING CANNOT BE TRANSLATED. REASON reason-co
CHARACTER code-point, HOST VARIABLE position-number
+335 DB2 CONVERTED A HOST VARIABLE, PARAMETER, OR COLUMN NUMBER var-num var-name-or-num TO COLUMN NAME,HOST VARIAP
OR EXPRESSION NUMBER col-name-or-num FROM from ccsid TO to-ccsid, AND RESULTING IN SUBSTITUTION CHARACTERS.
+339 THE SQL STATEMENT HAS BEEN SUCCESSFULLY EXECUTED, BUT THERE MAY BE SOME CHARACTER CONVERSION INCONSISTENCIES
+394 USER SPECIFIED OPTIMIZATION HINTS USED DURING ACCESS PATH SELECTION
+395 USER SPECIFIED OPTIMIZATION HINTS ARE INVALID (REASON CODE = reason-code). THE OPTIMIZATION HINTS ARE IGNORE
+402 LOCATION location IS UNKNOWN
+403 THE LOCAL OBJECT REFERENCED BY THE CREATE ALIAS STATEMENT DOES NOT EXIST
+434 OPTION keyword IS A DEPRECATED FEATURE
+445 VALUE value HAS BEEN TRUNCATED
+462 EXTERNAL FUNCTION OR PROCEDURE name (SPECIFIC NAME specific-name) HAS RETURNED A WARNING SQLSTATE, WITH DIAGN
TEXT text
+464 PROCEDURE proc RETURNED num QUERY RESULT SETS, WHICH EXCEEDS THE DEFINED LIMIT integer
+466 PROCEDURE proc RETURNED num QUERY RESULTS SETS
+494 NUMBER OF RESULT SETS IS GREATER THAN NUMBER OF LOCATORS
+495 ESTIMATED PROCESSOR COST OF estimate-amount1 PROCESSOR SECONDS (estimate-amount2 SERVICE UNITS) IN COST CATEG
cost-category EXCEEDS A RESOURCE LIMIT WARNING THRESHOLD OF limit- amount SERVICE UNITS
+535 THE RESULT OF THE POSITIONED UPDATE OR DELETE MAY DEPEND ON THE ORDER OF THE ROWS
+541 THE REFERENTIAL OR UNIQUE CONSTRAINT name HAS BEEN IGNORED BECAUSE IT IS A DUPLICATE
+551 auth-id DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION operation ON OBJECT object-name
+552 auth-id DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION operation
+558 THE WITH GRANT OPTION IS IGNORED
+561 THE ALTER, INDEX, REFERENCES, AND TRIGGER PRIVILEGES CANNOT BE GRANTED PUBLIC AT ALL LOCATIONS
+562 A GRANT OF A PRIVILEGE WAS IGNORED BECAUSE THE GRANTEE ALREADY HAS THE PRIVILEGE FROM THE GRANTOR
+585 THE SCHEMA NAME schema-name APPEARS MORE THAN ONCE IN THE CURRENT PATH
+599 COMPARISON FUNCTIONS ARE NOT CREATED FOR A DISTINCT TYPE BASED ON A LONG STRING DATA TYPE
+610 A CREATE/ALTER ON OBJECT object-name HAS PLACED OBJECT IN utility PENDING
+645 WHERE NOT NULL IS IGNORED BECAUSE THE INDEX KEY CANNOT CONTAIN NULL VALUES
+650 THE TABLE BEING CREATED OR ALTERED CANNOT BECOME A DEPENDENT TABLE
+653 TABLE table-name IN PARTITIONED TABLESPACE tspace-name IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS

```

```

NOT BEEN CREATED
+655 STOGROUP stogroup_name HAS BOTH SPECIFIC AND NON-SPECIFIC VOLUME IDS. IT WILL NOT BE ALLOWED IN FUTURE RELEASES
+658 THE SUBPAGES VALUE IS IGNORED FOR THE CATALOG INDEX index-name
+664 THE INTERNAL LENGTH OF THE LIMIT-KEY FIELDS FOR THE PARTITIONED INDEX index-name EXCEEDS THE LENGTH IMPOSED BY
+738 DEFINITION CHANGE OF object object_name MAY REQUIRE SIMILAR CHANGE ON READ-ONLY SYSTEMS
+799 A SET STATEMENT REFERENCES A SPECIAL REGISTER THAT DOES NOT EXIST AT THE SERVER SITE
+802 The null indicator was set to -2 as an arithmetic statement didn't work.
+806 BIND ISOLATION LEVEL RR CONFLICTS WITH TABLESPACE LOCKSIZE PAGE OR LOCKSIZE ROW AND LOCKMAX 0
+807 THE RESULT OF DECIMAL MULTIPLICATION MAY CAUSE OVERFLOW
+863 THE CONNECTION WAS SUCCESSFUL BUT ONLY SBCS WILL BE SUPPORTED
+883 ROLLBACK TO SAVEPOINT OCCURRED WHEN THERE WERE OPERATIONS THAT CANNOT BE UNDONE, OR AN OPERATION THAT CANNOT
UNDONE OCCURRED WHEN THERE WAS A SAVEPOINT OUTSTANDING
+2000 TYPE 1 INDEXES WITH SUBPAGES GREATER THAN 1 CANNOT BECOME GROUP BUFFER POOL DEPENDENT IN A DATA SHARING
ENVIRONMENT
+20002 THE GBPCACHE SPECIFICATION IS IGNORED, bpname DOES NOT ALLOW CACHING
+20007 USE OF OPTIMIZATION HINTS IS DISALLOWED BY A DB2 SUBSYSTEM PARAMETER. THE SPECIAL REGISTER 'OPTIMIZATION HINTS'
IS SET TO THE DEFAULT VALUE OF BLANKS.
+20122 DEFINE NO OPTION IS NOT APPLICABLE IN THE CONTEXT SPECIFIED
+20141 TRUNCATION OF VALUE WITH LENGTH length OCCURRED FOR hv-or-parm-number
+20267 OPTION clause IS NOT SUPPORTED IN THE CONTEXT IN WHICH IT WAS SPECIFIED
+30100 OPERATION COMPLETED SUCCESSFULLY BUT A DISTRIBUTION PROTOCOL VIOLATION HAS BEEN DETECTED. ORIGINAL
SQLCODE=original-sqlcode AND ORIGINAL SQLSTATE=original-sqlstate
-----

```

## Letter SQL Codes

```

-----
If you have a code in the type of SQL-0000##A these are still valid but a bit harder to decipher.
Use the chart below to map the letter to the last number of the value.
This was created when SQL Codes were only able to be unsigned.
Example   SQL-000010} => This translates to SQL CODE: 100.
Example   SQL-000010N => This translates to SQL CODE: -105.
Example   SQL-000092P => This translates to SQL CODE: -927.

-----
 0  1  2  3  4  5  6  7  8  9
}  A  B  C  D  E  F  G  H  I
-----
-1 -2 -3 -4 -5 -6 -7 -8 -9
  J  K  L  M  N  O  P  Q  R
-----

```

## External resources

- A more extensive list (<http://www.theamericanprogrammer.com/programming/sqlcodes.shtml>) of the codes

## License

### GNU Free Documentation License

Version 1.3, 3 November 2008 Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.  
<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is

instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards

disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## **2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## **3. COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## **4. MODIFICATIONS**

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the



Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## **10. FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME.  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.3  
or any later version published by the Free Software Foundation;  
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the section entitled "GNU  
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the  
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Retrieved from "[http://en.wikibooks.org/w/index.php?title=Structured\\_Query\\_Language/Print\\_version&](http://en.wikibooks.org/w/index.php?title=Structured_Query_Language/Print_version&)

- This page was last modified on 7 October 2013, at 15:30.
- Text is available under the Creative Commons Attribution/Share-Alike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy.