Databases for applications

# Study guide for Unit 1

## Index and views

* Index:

An index is a pointer to a row in a certain table in our database.

* Types of index
  + Unique: It is managed by means of the unique value that the table has (primary keys).
  + FullText: It is used to search on text (varchar, char and text).
  + Spatial: performs searches on data that make up geometric figures represented in the database space.
* Syntaxis:

|  |  |
| --- | --- |
| /\*crear indice\*/  CREATE INDEX idx\_name ON table\_name(attribute);  /\*consultar indice\*/  SHOW INDEX FROM table\_name; | /\*para editar un índice se elimina el índice y se vuelve a crear\*/  /\*eliminar indice\*/  ALTER TABLE table\_name  DROP INDEX idx\_name; |

* Views:

A view is a virtual table generated from the execution of several queries on one or more tables.

* Syntaxis:

|  |  |
| --- | --- |
| /\*crear vista\*/  CREATE VIEW view\_name  AS SELECT column FROM table\_name  WHERE condition;  /\*consultar vista\*/  SELECT \* FROM view\_name; | /\*editar vista\*/  ALTER VIEW view\_name  AS SELECT column FROM table\_name  Where condition;  /\*eliminar vista\*/  DROP VIEW view\_name; |

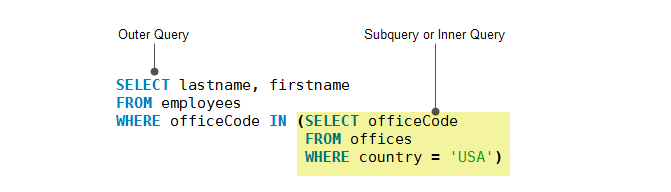
## Advanced queries salto de página

* Subqueries:

A subquery is a SELECT statement that appears inside another SELECT statement. A subquery is called an inner query while the query that contains the subquery is called an outer query.

A subquery can be used anywhere that expression is used and must be closed in parentheses.

Like this:



* Syntaxis:

We can use any,all or some to have more or less specific data.

|  |  |  |
| --- | --- | --- |
| SELECT column\_name(s)  FROM table\_name  WHERE column\_name operator ANY  (SELECT column\_name FROM table\_name WHERE condition); | SELECT column\_name(s)  FROM table\_name  WHERE column\_name operator ALL  (SELECT column\_name FROM table\_name WHERE condition); | SELECT column\_name(s)  FROM table\_name  WHERE column\_name operator SOME  (SELECT column\_name FROM table\_name WHERE condition); |

* Aggregate functions:

Aggregate functions allow us to easily produce summary data from our database.

MySQL supports all five standard aggregate functions:

* + - COUNT: Returns the number of rows that meet the WHERE.
    - SUM: Returns the total of the sum.
    - AVG: Returns the average.
    - MIN: Returns the smallest value.
    - MAX: Returns the largest value.
* Syntaxis:

|  |  |  |
| --- | --- | --- |
| SELECT COUNT(column\_name)  FROM table\_name  WHERE condition; | SELECT SUM(column\_name)  FROM table\_name  WHERE condition; | SELECT AVG(column\_name)  FROM table\_name  WHERE condition; |
| SELECT MIN(column\_name)  FROM table\_name  WHERE condition; | SELECT MAX(column\_name)  FROM table\_name  WHERE condition; |  |

* Sorting and grouping:
* Syntaxis:

|  |  |
| --- | --- |
| SELECT column1, column2, ...  FROM table\_name  ORDER BY column1, column2, ... ASC|DESC; | SELECT column\_name(s)  FROM table\_name  WHERE condition  GROUP BY column\_name(s)  HAVING condition  ORDER BY column\_name(s); |

## Triggers

A trigger is a named object within a database that is associated with a table and fires when a particular event occurs in the table.

* Type of triggers:
  + DML: They affect tables or views when an INSERT, UPDATE or DELETE statement is executed.
  + DDL: They are used to respond to DML events (CREATE, ALTER, DROP, GRANT, DENY, etc).
  + LOGON: When there is a login event.
* Syntaxis:

|  |  |  |
| --- | --- | --- |
| CREATE TRIGGER trigger\_name  {BEFORE | AFTER} {INSERT | UPDATE| DELETE }  ON table\_name FOR EACH ROW  trigger\_body; | DROP TRIGGER [schema\_name.]trigger\_name;  SHOW TRIGGERS  FROM database\_name; | DELIMITER $$ o //  CREATE TRIGGER trigger\_name  {BEFORE|AFTER}{INSERT|UPDATE|DELETE}  ON table\_name FOR EACH ROW  {FOLLOWS|PRECEDES} existing\_trigger\_name  BEGIN  -- statements  END$$ o //  DELIMITER ; |

## Stored Procedures

A stored procedure is a set of SQL commands that are stored together with the database that are used to perform transactions or operations such as inserting or modifying records.

* Syntaxis:

|  |  |  |
| --- | --- | --- |
| /\*crear un sp\*/  CREATE PROCEDURE name (parameters)  [characteristics] definition | /\*modificar sp\*/  ALTER PROCEDURE procedure\_name (parameters)  [characteristics] definition | /\*eliminar sp\*/  DROP PROCEDURE [IF EXISTS] procedure\_name |

* Types of stored procedures:

MySQL stored procedure parameters can be of three types:

* IN: It is the type of parameter used by default. The application or code that invokes the procedure will have to pass an argument for this parameter. The procedure will work with a copy of its value, the parameter having its original value at the end of the procedure's execution.
* OUT: The value of this parameter can be changed in the procedure, and also its modified value will be sent back to the code or program that invokes the procedure.
* INOUT: It is a mixture of the two previous concepts. The application or code that invokes the procedure can pass a value to it, returning the modified value at the end of the execution. If you find it confusing, take a look at the example that you will see later.

# Study guide for Unit 2

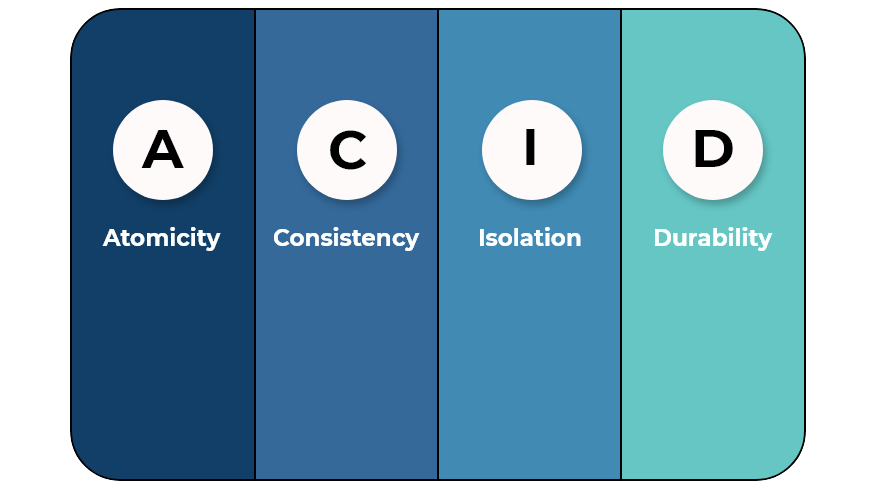
## Transaction handling

* Concept of transaction

A transaction is a unit of work composed of several tasks, the end result of which must be that all or none of them are executed.

There are three types of transaction depending on the way it is started:

* **Automatic confirmation:** Where the SGBD performs this confirmation every time there is a change (such as data update) in the data within the database.
* **Implicit:** Where the DBMS makes said confirmation only if the programmer (or program) confirms or rejects it first.
* **Explicit:** Where the confirmation is done "by hand" using SQL statements.
* Fundamental properties of transactions

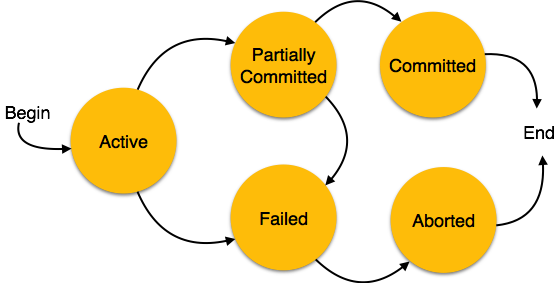


* Atomicity: Ensures that events are carried out or not, a transaction cannot be incomplete (example: application error).
* Consistency: Execute only events that do not break the rules and guidelines of the integrity of the database.
* Isolation: They must be activated when many transactions are executed at the same time by one or more users so that no query or modification is affected.
* Durability: It is the persistence of the database, if the system fails, the changes already made should not be compromised.
* Structure of a transaction

Start o Begin Rollback Commit

(inicia la transacción) (Deshace las transacciones (Finaliza la

en caso de haber un problema) transacción)



## Consistency and insulation levels

* concept of consistency

It is a broader term than integrity. It could be defined as the consistency between all the data in the database.

* A completed transaction (partially confirmed) may not be definitively confirmed (consistency).
* The execution of a transaction must lead to a consistent database state (meeting all defined integrity constraints).
* If the system is definitively confirmed, it ensures the persistence of the changes made in the database.
* If you cancel the changes you have made are undone.
* A transaction that completes successfully is said to be committed.
* concept of insulation levels

Transactions specify an isolation level that defines the degree to which a transaction should be isolated from resource or data modifications made by other transactions. Isolation levels are described based on the side effects of concurrency that are allowed, such as dirty data reads or phantom reads (those that serve us only to view information).

* **Confirmed reading:**A transaction will not be able to see changes from other connections until they have been committed or dropped.
* **Unconfirmed reading:**They do not affect the locks produced by other connections to the data reading.
* **Repeatable reading:**It guarantees that the data read cannot be changed by other transactions, during that transaction.
* **Serializable:**Other transactions will not be allowed to insert, update or delete data used by our transaction. It blocks them while it lasts.

# Study guide for Unit 3

## Database encryption

* Concept of data encryption

In the computer world, encryption is the conversion of data from a readable format to an encrypted format, which can only be read or processed after it has been decrypted.

* Objectives that crypto should offer
  + **Privacy or confidentiality:** only authorized persons can obtain the information.
  + **Integrity:** it must be possible to verify that the message has not been modified during its journey.
  + **Authentication:** when communication is secure between two interlocutors.
  + **Non-repudiation:** neither party must subsequently deny that it has performed a certain action or that it has transmitted certain information.
* Encryption methods

**Symmetric key encryption:** is one that uses the same key to encrypt and decrypt the message and that the sender and receiver must first know.

**Asymmetric cryptography:** it is one that uses two keys. The public and the private.

