



SQL DML

Esc. Normal Superior N° 10
Analista de Sistemas
Seminario 2022
Prof.: Fabio Dos Santos

Struct Query Language (SQL)

Lenguaje de consultas estructurado

- Lenguaje declarativo: describe lo que el programa debe lograr, en lugar de describir cómo lograrlo
- Tiene una clara conexión con su resultado matemático porque esta basado en el Álgebra Relacional y el Calculo Relacional de tuplas.
- Incluye capacidades como la de definir la estructura de los datos y la actualización de los mismos.
- Además suele también combinarse con lenguaje imperativo y administrativo

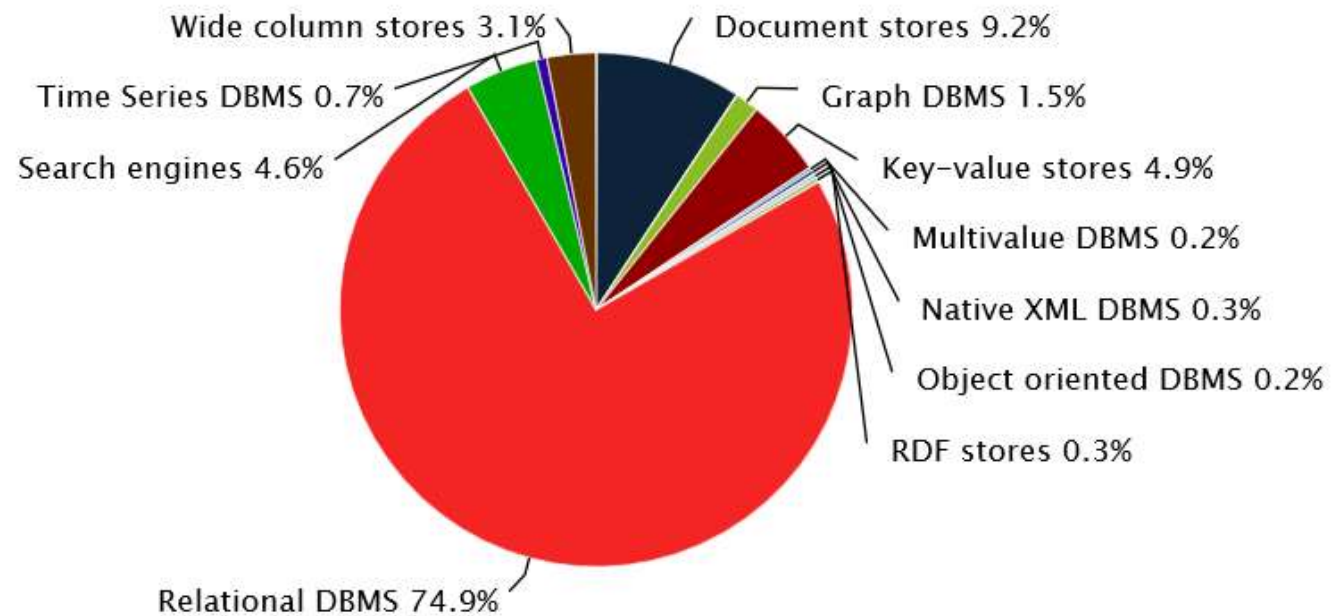


Fue definido inicialmente por D.D. Chamberlin en los años 70 y se denominó SEQUEL (Struct English Query Language). Fue desarrollado con el prototipo de base de datos relacional de IBM, el System R.

Imagen: mappingthejourney.com

SQL

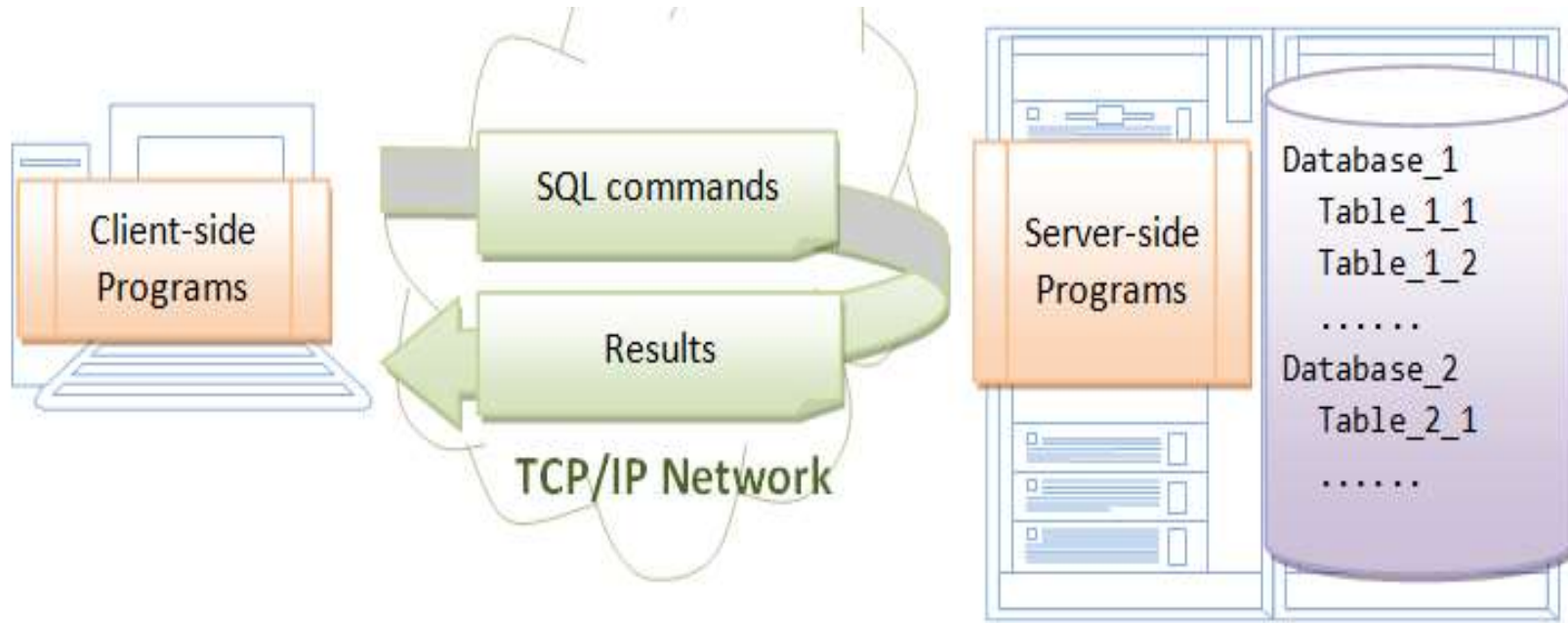
- SQL es el lenguaje de consulta de mayor influencia, el mas implementado en conjunto con las base de datos relacionales.





Evolución de SQL

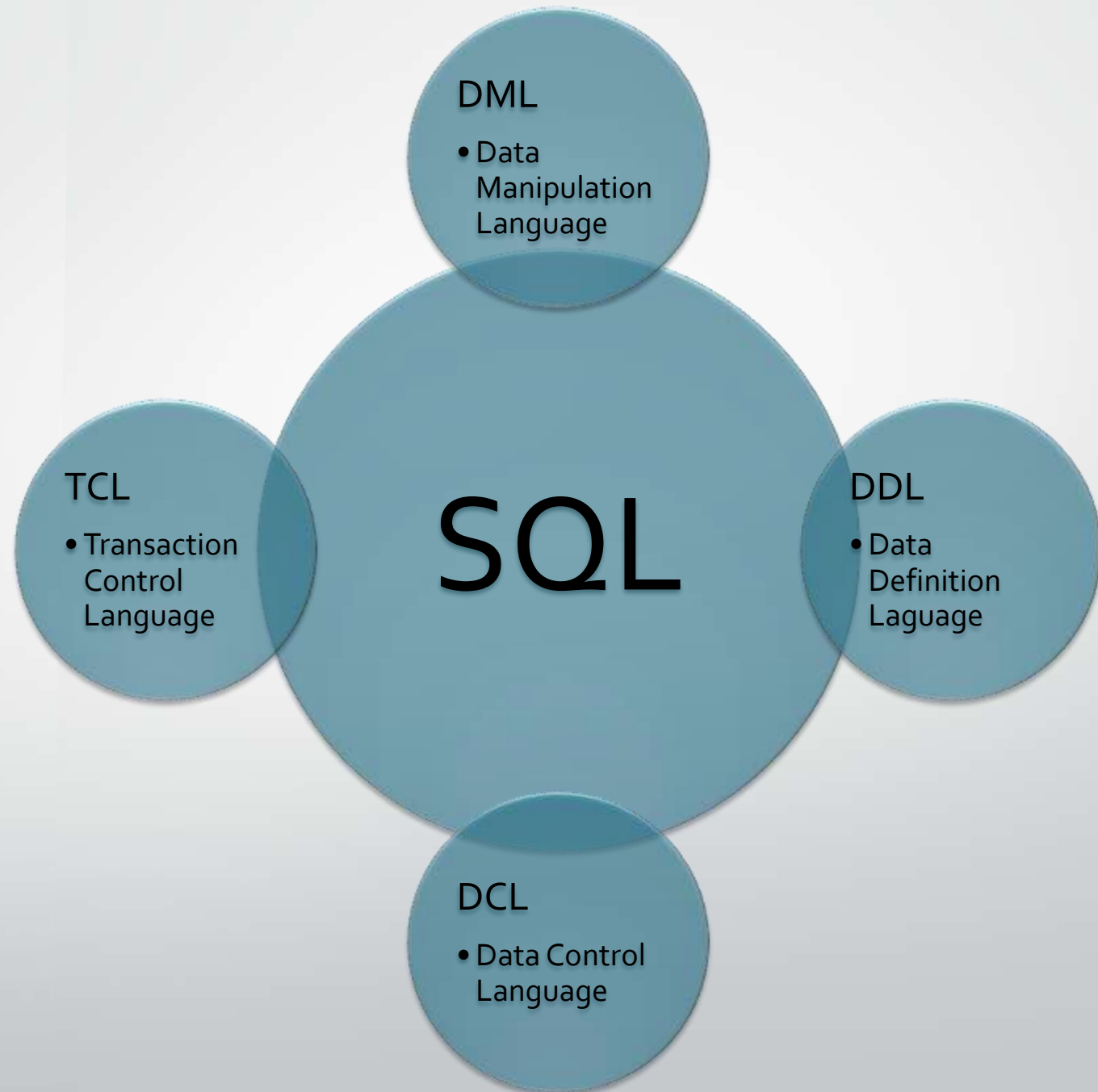
Principios 1970s	Modelo relacional
1974-77	IBM System R
Principios 1980s	Primeros RDBMS
1986	SQL-Standard By Ansi e ISO un año después
1989	Mejoras de integridad aprobadas por ISO
1992	Nueva versión llamada SQL2
1999	SQL3 también llamado SQL-99
2003	XML y otras características de secuencia
2006	Importación y exportación XML por la IOS/IEC



www3.ntu.edu.sg

Modelo
cliente
servidor

Componentes de SQL



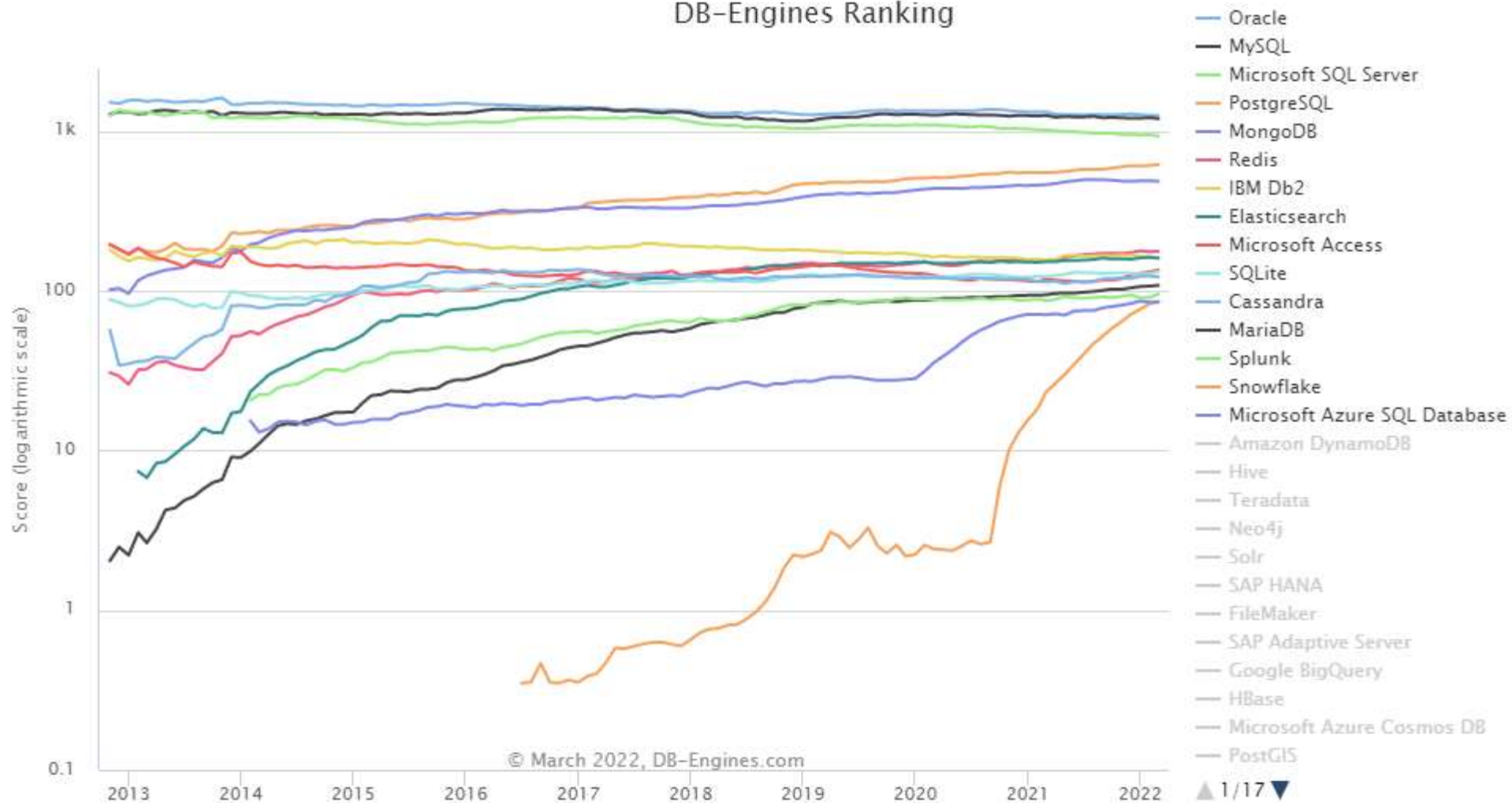


- Es un sistema de manejo de bases de datos del modelo relacional desarrollado por la empresa Microsoft.
- El lenguaje de desarrollo es utilizado por línea de comandos o mediante la interfaz gráfica de Management Studio.
- Se distribuye en varias ediciones: Enterprise, Standard Express y Developer.
- El código fuente original de SQL Server que fue utilizado en las versiones previas a la versión 7.0 habría sido comprado de Sybase, pero fue actualizado en las versiones 7.0 y 2000, y reescrito en la versión 2005. Los lenguajes utilizados son C y C++.
- Generalmente, cada 2-3 años, una nueva versión es lanzada y, entre estos lanzamientos, se proponen service packs con mejoras y correcciones de bugs, y hotfixes por problemas urgentes en el sistema de seguridad o bugs críticos.

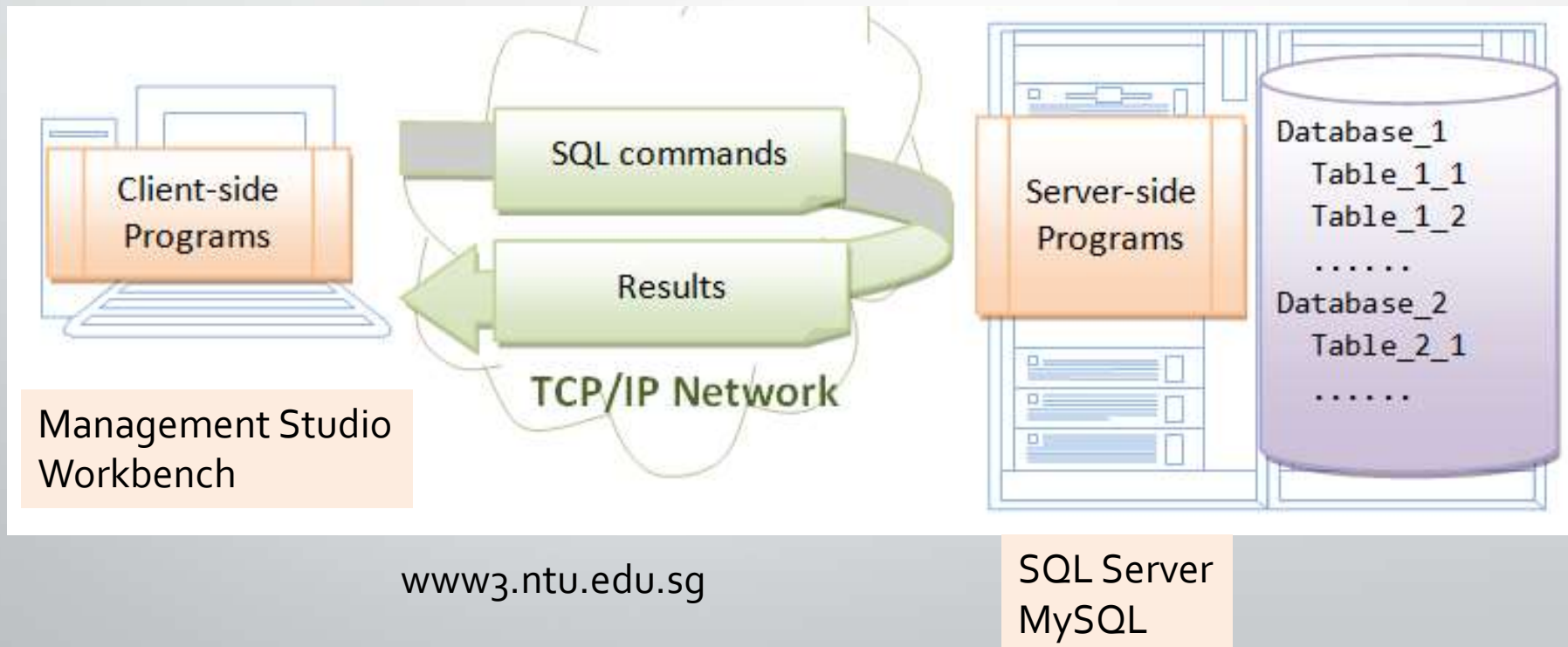


- Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.
- MySQL fue inicialmente desarrollado por MySQL AB que fue adquirida por Sun Microsystems en 2008, y ésta a su vez por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.
- Se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise.
- Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

DB-Engines Ranking



Modelo cliente servidor



Tarea



DESCARGAR

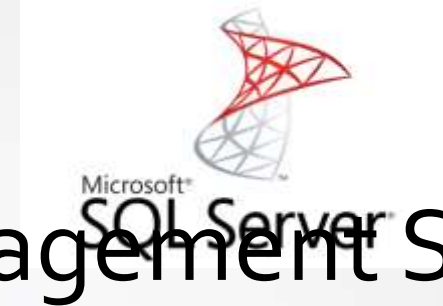


INSTALAR



Instalación de SQL Server Developer Edition

- Tener una cuenta Microsoft
- <https://www.microsoft.com/es-es/sql-server/sql-server-downloads>



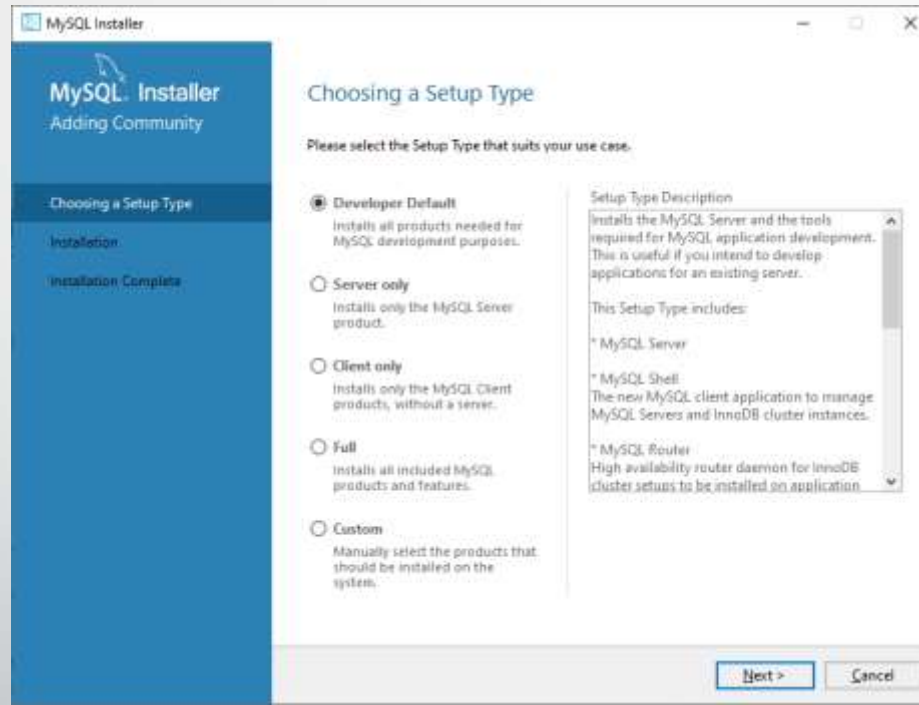
Instalación de SQL Server Management Studio

- <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms>
- **Version information**
- Release number: 18.4
- Build number: 15.0.18206.0
- Release date: November 04, 2019

Sobre la instalación

- Features Selection (características) sólo se necesita marcar DataBase Engine Services (servicios de motor de base de datos)
- Authentication Mode (modo de autenticación): Dejar Windows... y click en Add current user (agregar usuario actual) – es el administrador de la BD

MySQL



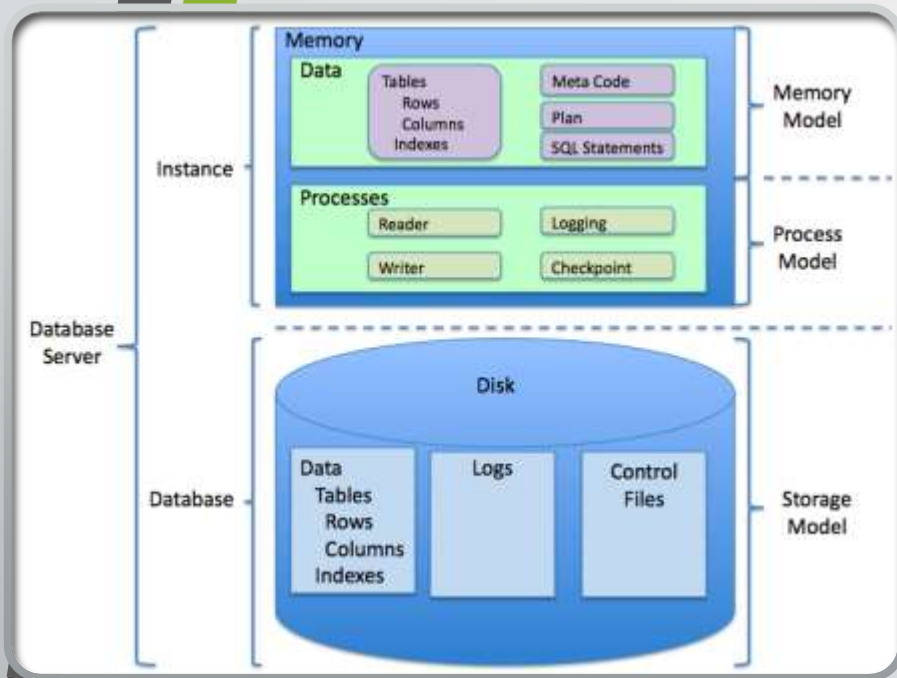
Base de datos Database

- Una **base de datos** o **banco de datos** (*BD*) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.



Sistema de gestión de bases de datos (SGBD)

Database management system (DBMS)

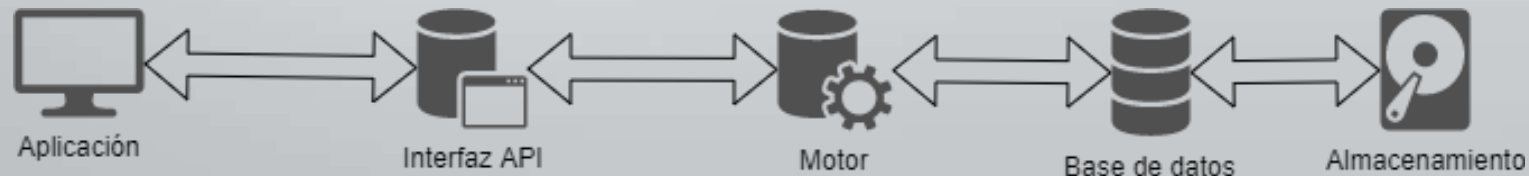


- El sistema de gestión de bases de datos (DBMS) es el software que interactúa con los usuarios finales, las aplicaciones y la propia base de datos para capturar y analizar los datos.
- El software DBMS abarca además el núcleo principal proporcionado para administrar la base de datos.
- La suma total de la base de datos, el DBMS y las aplicaciones asociadas se puede denominar "sistema de base de datos".
- A menudo, el término "base de datos" también se utiliza para hacer referencia libremente a cualquiera de los DBMS, el sistema de base de datos o una aplicación asociada a la base de datos.

Motor de base de datos

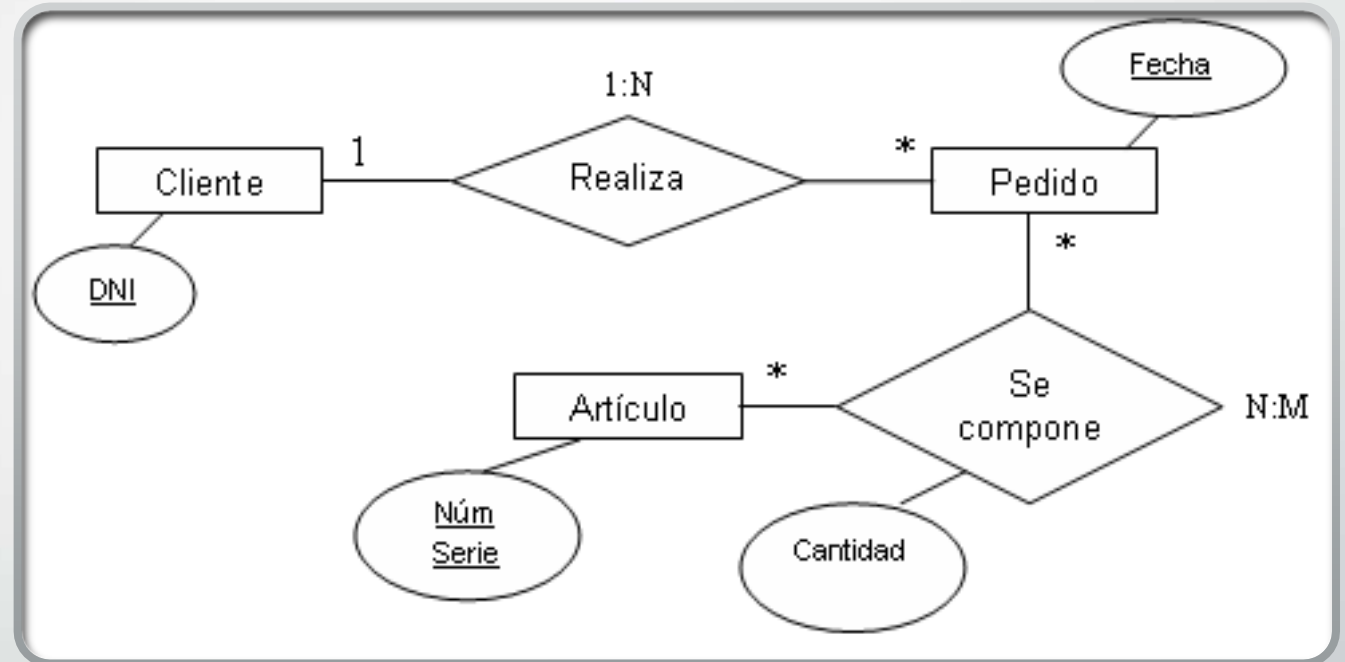
Database engine or storage engine

- Es el componente de software subyacente que un sistema de administración de bases de datos (DBMS) utiliza para crear, leer, actualizar y eliminar datos (CRUD) de una base de datos.
- La mayoría de los sistemas de administración de bases de datos incluyen su propia interfaz de programación de aplicaciones (API) que permite al usuario interactuar con su motor subyacente sin pasar por la interfaz de usuario del DBMS.
- El término "motor de base de datos" se utiliza con frecuencia indistintamente con "servidor de base de datos" o "sistema de gestión de bases de datos".
- Una 'instancia de base de datos' hace referencia a los procesos y estructuras de memoria del motor de base de datos en ejecución.



Modelo entidad-relación

- *Entidad:* es el objeto sobre el cual se requiere mantener o almacenar información.
- *Atributo:* son las propiedades que describen y califican una entidad. Ej: Entidad cliente(nombre, apellido, dirección, edad)
- *Relación:* es la asociación significativa y estable entre dos entidades



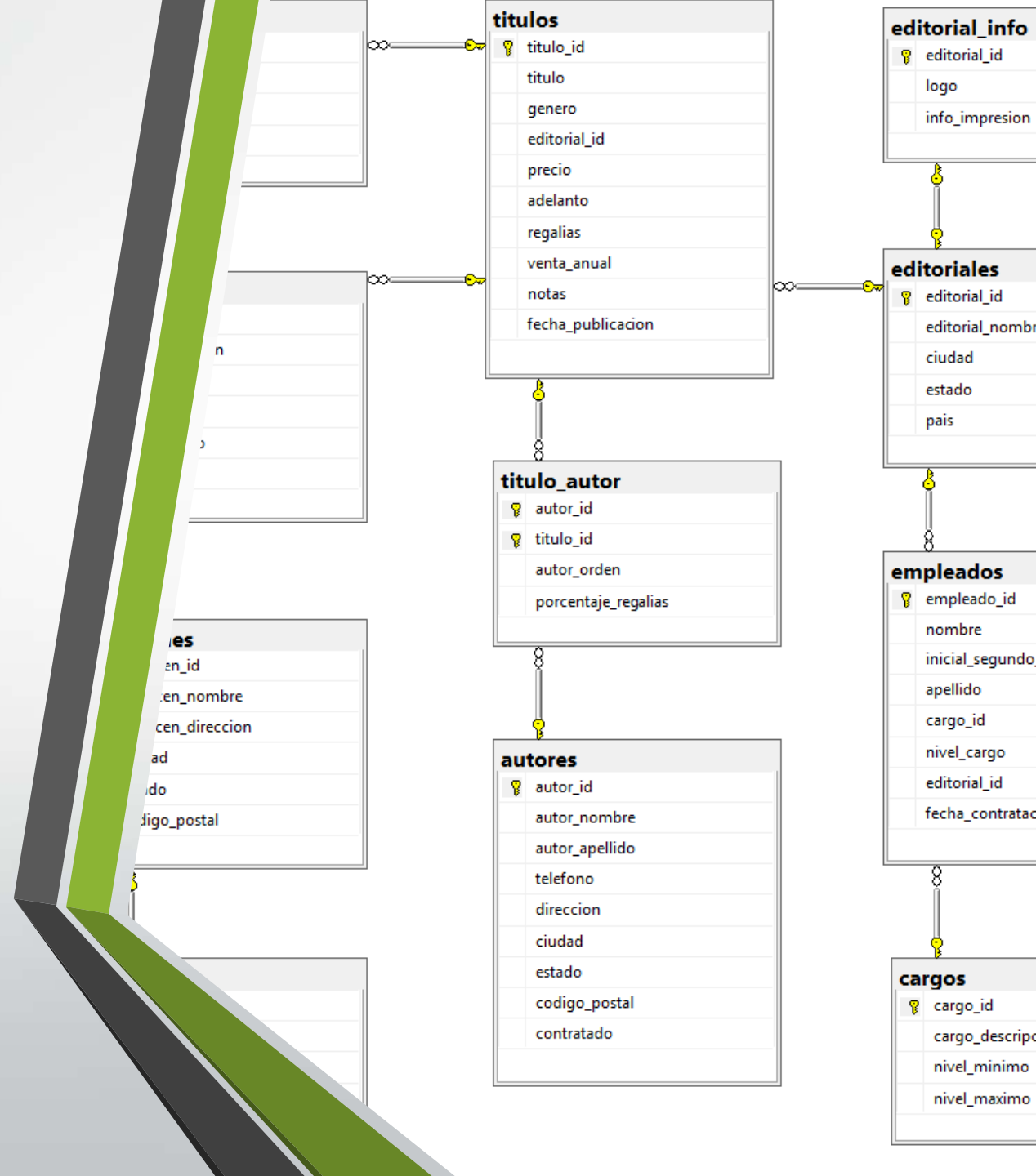
Modelo relacional

- Se usa para el modelado y gestión de base de datos
- Su idea fundamental es el uso de relaciones.
- Las relaciones son un conjunto de tuplas.
- Las tuplas son una secuencia de valores agrupados

ALUMNOS(dni, apellido, nombre, direccion, telefono, sexo, fecha_nacimiento)

CLIENTE(cuit, razon, direccion, telefono)

PEDIDOS(numero, fecha, fecha_entrega)



Algebra Relacional

- El álgebra relacional es un conjunto de operaciones que describen paso a paso cómo computar una respuesta sobre las relaciones, tal y como éstas son definidas en el modelo relacional.
- Describe el aspecto de la manipulación de datos. Estas operaciones se usan como una representación intermedia de una consulta a una base de datos y, debido a sus propiedades algebraicas, sirven para obtener una versión más optimizada y eficiente de dicha consulta.

relational algebra

set operations



set union



set intersection



set difference



cartesian product

relational database specific operations



selection



projection



join



set division

set functions

sum

avg

count

any

max

min

Convenciones utilizadas

Lenguaje de programación	Diagrama Entidad Relación	Modelo Relacional	Base de datos
Archivo	Entidad	Relación	Tabla
Campo	Atributo	Atributo	Columna
Registro	Registro	Tupla	Fila
	Relación	Operación	Asociación

Estructura básica de una consulta

- SELECT: corresponde al operador Proyección del Álgebra Relacional, se utiliza para dar la lista de los atributos que se quieren obtener en la consulta. también pueden ser utilizadas expresiones.
- FROM: indica la o las relaciones del Álgebra Relacional, que intervienen en la consulta. Si hay mas de una relación, se realiza el Producto Cartesiano entre las relaciones que intervengan.
- WHERE: corresponde al operador Selección del Álgebra Relacional, es decir es un predicado que se aplica a las tuplas de la relación indicada en la cláusula FROM o a las tuplas del producto cartesiano si interviene mas de una relación en la consulta.
- SELECT <lista de columnas>
- [FROM <lista de tablas>
- [WHERE <predicado>]]

Columnas calculadas

- Se pueden utilizar valores constantes, funciones o variables del sistema y si no se coloca la cláusula FROM se obtiene una relación con un solo atributo y 1 sola tupla.

```
SELECT 5*4
```

```
SELECT getdate()
```

```
SELECT @@VERSION
```

Columnas calculadas

- Se pueden utilizar expresiones aritméticas, alfanuméricas o funciones aplicadas a los atributos de una relación.

```
SELECT legajo, sueldo_mensual,  
       sueldo_mensual + premios * 1.5  
FROM asalariado
```

Selección de columnas

- Corresponde a la operación Proyección del Álgebra Relacional (Π).
- Se puede utilizar el * para indicar todos los campos de las relaciones que intervienen o bien el nombre de una relación.* para indicar todos los campos pero solo de esa relación.
SELECT *
FROM autores
- Listaría todos los atributos de la relación autores.
SELECT editores.*, Info_impresion
FROM editores, editor_info
WHERE editores.editor_id = editor_Info.editor_id
- Produce como resultado una relación con todos los campos de la relación Editor mas la Info_impresion de editor_info.

Calificadores para columnas

- Permite especificar desde que tabla proviene la columna
- <tabla>.<columna>
- Cuando se tiene mas de una tabla con el mismo nombre de columna para evitar ambigüedad hay que especificar de que tabla es la columna.

Alias de columna

- Las columnas cambian en conjunto de resultados resultante el nombre del atributo por el alias.
- Para renombrar atributos hay 3 expresiones equivalentes:
 - 1. **SELECT atributo as alias**
 - 2. **SELECT atributo alias**
 - 3. **SELECT alias=atributo**
- Ejemplo:
SELECT legajo, fecha_ing as Ingreso, fecha_baja as Baja
FROM empleado
- El renombramiento en los atributos es la ultima operación que se realiza, por ende no pueden utilizarse los alias en otras cláusulas.
- Por ejemplo no seria valido:
SELECT empleado.legajo as ap, sueldo_mensual
FROM empleado
WHERE ap='Perez'

Alias de tablas

- Para renombrar las relaciones que intervienen en una consulta, se utiliza el alias luego del nombre de la relación o entre ambos se coloca la palabra reservada AS.

```
SELECT E.legajo, apellido, nombres, sueldo_mensual  
FROM empleado AS E, asalariado AS Asa  
WHERE E.legajo=Asa.legajo
```
- El renombramiento de las relaciones que intervienen en la cláusula FROM es especialmente útil para comparar 2 filas de la misma relación.
- Ejemplo: obtener los legajos de las asalariados que ganan mas que algún otro asalariado.

```
SELECT DISTINCT A1.legajo  
FROM asalariado A1, asalariado A2  
WHERE A1.sueldo_mensual > A2.sueldo_mensual
```

Filas sin repetición

- Como nombramos anteriormente, el resultado de una consulta en SQL es una nueva relación. Por ejemplo:
-- Listar los generos de los libros

```
SELECT genero  
FROM titulos
```

En el resultado estarían los nombres de cada una de las tuplas de la relación empleados.
- Si queremos eliminar los duplicados, se inserta la palabra clave Distinct después de SELECT. Si no se utiliza la palabra clave Distinct se asume que no se eliminarán los duplicados (también puede ser indicado implícitamente con la palabra clave ALL).
-- Listar los generos de los libros

```
SELECT DISTINCT genero  
FROM titulos
```

Filas superiores

- Si se quiere restringir la consulta a las primeras n tuplas de la relación resultante, se utiliza TOP n para indicar que se listen solo las primeras n tuplas de la relación resultante.

```
SELECT [TOP n [PERCENT]]
```

- Ejemplo

```
SELECT TOP 5 *
```

```
FROM empleados
```

- Listaría las primeras 5 tuplas de la relación empleado.

- Ejemplo

```
SELECT TOP 10 PERCENT *
```

```
FROM empleados
```

- Listaría EL 10% de las tuplas de la relación empleado.

Cláusula WHERE

- Tiene 2 funciones primordiales:
 - Establecer la selección de registros (filtrar información)
 - Definir la correspondencia de registros entre varias tablas que están unidas a través de claves foráneas
- En la misma se especifican predicados. Se utilizan conectores lógicos y operadores de comparación
- Sintaxis:

```
SELECT <lista columnas>
```

```
FROM <lista tablas>
```

```
WHERE <nombre columna> <operador> <expresión> [<operador lógico> <nombre columna>  
<operador> <expresión>]
```

Operadores

Operador	Descripción
=	Igual a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	Desigual a
!=	
!<	No mayor que
!>	No menor que
()	Orden de precedencia
AND	Unión necesaria (Y)
OR	Unión alternativa (O)

Junta de tablas

- La cláusula FROM realiza un producto de las relaciones que intervienen. Para realizar una junta natural, es necesario igualar en el predicado los atributos comunes en la cláusula WHERE
- Ejemplo:

```
SELECT empleado.*, tarea_desc  
FROM empleado, tarea  
WHERE empleado.tarea_id = tarea.tarea_id
```
- Como en el Álgebra Relacional se realiza el producto cartesiano y luego la selección de las tuplas que cumplen con la **condición** de junta, pero a diferencia del Álgebra Relacional si se utiliza el * en el SELECT, no se eliminan instancias de los atributos comunes.
- Se pueden incluir otros operadores de comparación que no sea el igual.

Claúsula Where

- Ejemplo:

```
SELECT *  
FROM autores  
WHERE contratado
```

```
SELECT empleado_id, apellido, nombre  
FROM empleados  
WHERE year(fecha_contratacion) = 2003 AND month(fecha_ing) = 2
```

Consultas basadas en rango

- Between: Permite recuperar filas basadas en un rango de valores.
- Sintaxis:

SELECT <lista de columnas>

FROM <lista de tablas>

WHERE <columna> [NOT] BETWEEN <valor inicial> AND
<valor final>

Recomendaciones para el uso de condiciones

- En primer lugar las condiciones de relación.
- En segundo lugar las condiciones de selección
- Las condiciones de relación y de selección van conectadas por “AND”

Operaciones sobre cadenas

- Para comparación de cadenas de caracteres, se utiliza el operador LIKE con caracteres especiales para definir patrones.
- Dichos caracteres son:
 - % : especifica cualquier subcadena sin limites en la cantidad de caracteres.
 - _ : especifica cualquier carácter en dicha posición (uno solo).
 - [] : para especificar coincidencias del carácter en esa posición contra rangos o listas de caracteres.
 - [^] : para especificar no coincidencias del carácter en esa posición contra rangos o listas de caracteres.
- Ejemplos: Todos los empleados cuyos apellidos empiezan con P

```
SELECT *
FROM empleado
WHERE apellido LIKE 'P%'
```
- Todos los empleados que contienen la cadena de caracteres GARCIA en su apellido

```
SELECT *
FROM empleado
WHERE apellido LIKE '%GARCIA%'
```

Operaciones sobre cadenas

- Todos los empleados que contienen al menos 3 letras en el apellido

```
SELECT *  
FROM empleado  
WHERE apellido LIKE '___%'
```
- Todos los empleados cuyos apellidos empiezan con la letra A o la letra C

```
SELECT *  
FROM empleado  
WHERE apellido LIKE '[A,C]%'
```
- Todos los empleados cuyos apellidos no empiezan con la letras R, S o T

```
SELECT *  
FROM empleado  
WHERE apellido LIKE '[^R-T]%'
```
- Si el patrón de búsqueda contiene los caracteres % o _ , SQL permite cambiar el sentido de dichos caracteres por medio de un carácter escape. El mismo se define con la palabra reservada ESCAPE seguido del carácter que debe tener una interpretación literal. Por ejemplo: LIKE 'ab/%cd%' ESCAPE '/' busca todas las cadenas que empiezan con ab%cd también se puede utilizar la negación de la cláusula LIKE para especificar discordancias.

```
SELECT *  
FROM empleado  
WHERE apellido NOT LIKE '%GARCIA%'
```
- Produciría como resultado los datos de los empleados que no contengan la cadena GARCIA en su apellido.

Valores nulos

- En Sql, el valor nulo (null) pertenece a todos los dominios.
- Para definir predicados con atributos que contengan el valor nulo, se utiliza el operador IS NULL.
- Ejemplo:

```
SELECT *  
FROM empleado  
WHERE fecha_baja IS NULL.
```
- Para la negación se utiliza IS NOT NULL.
- La función ISNULL(expresión , valor_a_retornar) permite cambiar el valor null de la expresión en cuestión por el valor contenido en el segundo parámetro.
- Ejemplo:

```
SELECT legajo, apellido, ISNULL( nombres, 'FALTA DATO' )  
FROM empleado
```

Cláusula Order By

- Permite ordenar las tuplas resultantes de una consulta. Se puede especificar los atributos por los cuales se lista o un numero que indica el numero de columna (esto es especialmente útil cuando la columna que se lista no tiene un nombre).
- También se puede especificar por cada columna si el orden será Ascendente (ASC,default) o Descendente (DESC).

```
SELECT legajo, apellido, nombres  
FROM empleado  
ORDER BY apellido DESC, nombres ASC
```

- Ordena el resultado en forma descendente por apellido, pero a iguales apellidos ordena en forma ascendente por el atributo nombres.

- Es equivalente a:

```
SELECT legajo, apellido, nombres  
FROM empleado  
ORDER BY 2 DESC, 3 ASC
```

Cláusula Order By

- Evite especificar enteros en la cláusula ORDER BY como representaciones posicionales de las columnas en la lista de selección. Por ejemplo, aunque una instrucción como `SELECT ProductID, Name FROM Production.Production ORDER BY 2` es válida pero otros usuarios no la entenderán tan bien como si especificase el nombre de la columna real. Además, para realizar cambios en la lista de selección, como modificar el orden de las columnas o agregar otras nuevas, es necesario modificar la cláusula ORDER BY a fin de evitar resultados inesperados.
- En una instrucción `SELECT TOP (N)`, use siempre una cláusula ORDER BY. Esta es la única manera de indicar previsiblemente a qué filas afecta TOP.

Funciones agregadas

- Las funciones agregadas se aplican a un conjunto de valores y producen un único valor. En otras palabras múltiples filas se combinan en una.
- Dichas funciones son:
 - AVG: promedio
 - COUNT: contar
 - SUM: suma
 - MIN: mínimo
 - MAX: máximo
- Sintaxis:
SELECT <función([DISTINCT] columna)>
FROM <tabla>

Funciones agregadas

- Para las funciones AVG, SUM y COUNT se puede utilizar la palabra DISTINCT para promediar, sumar o contar solo valores que no se repitan o ALL (default) para indicar que se consideren todos los valores.

```
SELECT COUNT(DISTINCT sueldo_mensual)
```

```
FROM asalariado
```

- Daría como resultado la cantidad de sueldos mensuales distintos que se pagan en la empresa.
- La función COUNT admite como argumento a un asterisco debido a que dicha función cuenta filas sin depender del atributo. Sin embargo si se desean contar valores no repetidos o no considerar los valores nulos se debe hacer referencia a un atributo.
- Las funciones COUNT, AVG y SUM no consideran valores nulos, excepto COUNT si es utilizado el * como argumento.
- No se permiten combinar las funciones agregadas, es decir no sería válido por ejemplo utilizar COUNT(MAX(...))

Agrupación

- GROUP BY agrupa tuplas por un conjunto de atributos.

```
SELECT <columna a agrupar> [, <columna a agrupar>], <funciones
    agregadas>
FROM <lista tablas>
[WHERE <predicado>]
[GROUP BY <columna a agrupar> [, <columna a agrupar>]]
```
- El agrupamiento se realiza luego de haber aplicado la cláusula WHERE si esta existe
- Por cada valor del conjunto de valores especificados en la cláusula Group By se genera una tupla en el resultado.
- Si se utilizan funciones agregadas en la cláusula SELECT, las mismas se aplican a los grupos.

```
SELECT tarea_id, COUNT(*)
FROM empleado
GROUP BY tarea_id
```
- Produciría como resultado una relación donde cada fila corresponde a un código de tarea y por cada código de tarea se muestra la cantidad de empleados que tienen esa tarea asignada.

Agrupación

- Si se listan atributos en el SELECT cuando existe la cláusula Group By, los mismos deben estar incluidos en los atributos por los cuales se agrupa. No pueden listarse atributos que no estén en la cláusula Group By.
- Obtiene la cantidad de empleados por año de contratación

```
SELECT year(fecha_contratacion), COUNT(*)
FROM empleados
GROUP BY year(fecha_contratacion)
```
- Listar la cantidad empleados contratados cada año por cada editor

```
SELECT editor_id, year(fecha_contratacion), COUNT(*)
FROM empleados
GROUP BY editor_id, year(fecha_contratacion)
ORDER BY editor_id
```

Predicado de grupo

- Having indica un predicado que se aplica a los grupos (a diferencia del WHERE que es también un predicado pero aplicable a las tuplas).
- Una vez que se formaron los grupos se aplica el predicado a dichos grupos y solo se seleccionan los grupos que cumplan con dicho predicado.
- Por lo general se utilizan funciones agregadas dentro de la cláusula Having
- Ejemplo: Listar las ocurrencias en la cual un editor haya contratado más de un empleado en el año. Mostrar código de editor, año y cantidad empleados contratados

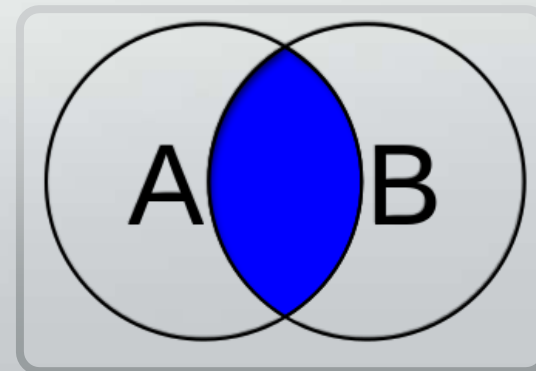
```
SELECT editor_id, year(fecha_contratacion), COUNT(*)  
FROM empleados  
GROUP BY editor_id, year(fecha_contratacion)  
HAVING COUNT(*) > 1  
ORDER BY editor_id
```


Junta

- Las condiciones de junta de 2 o mas tablas pueden ser especificadas en la cláusula WHERE o en la cláusula FROM.
- Si se especifica en la cláusula FROM, existen 2 tipos de juntas: INNER JOIN o OUTER JOIN.

Junta interna

- Inner Join retorna las filas del producto cartesiano que cumplan con la condición de junta, exigiendo que exista una tupla en la primer relación y una tupla en la segunda relación que cumplan con dicha condición.
- Es equivalente a realizar la junta natural expresando la condición de junta en la cláusula WHERE. Luego de realizado el producto cartesiano, se retornan solo las tuplas que cumplen con la condición de junta.

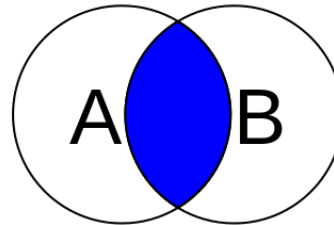


Junta interna

R1

A	B	C
a1	a1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

R2



C	D
c1	d1
c2	d2
c3	d3
c6	d6
c7	d7

SELECT
FROM
ON

*
R1 INNER JOIN R2
R1.C = R2.C

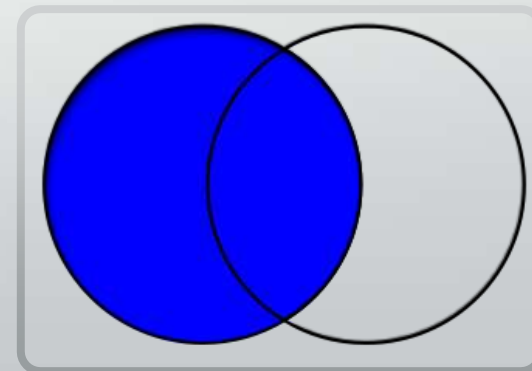
A	B	C	C	D
A1	b1	c1	c1	d1
a2	b2	c2	c2	d2
a3	b3	c3	c3	d3

Junta externa

- Outer Join retorna todas las filas de una de las 2 relaciones o de las 2.
- A su vez OUTER puede ser LEFT, RIGHT o FULL.

Junta externa por izquierda

- LEFT Outer retornara todas las filas de la relación que esta a la izquierda del operador Outer.
 - Primero se realiza el producto cartesiano,
 - Luego se aplica la condición de junta sobre las tuplas resultantes
 - Por ultimo se traen de la relación que esta a la izquierda las tuplas que no fueron incluidas completando con valores nulos en los atributos de la relación que figura a la derecha.

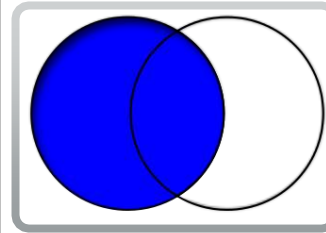


Junta externa por izquierda

R1

A	B	C
a1	a1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

R2



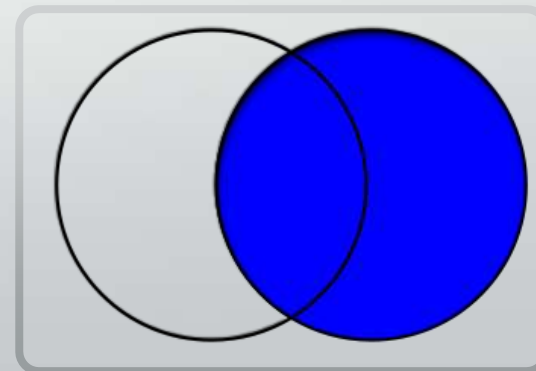
C	D
c1	d1
c2	d2
c3	d3
c6	d6
c7	d7

```
SELECT      *  
FROM        R1 LEFT OUTER JOIN  R2  
ON          R1.C = R2.C
```

A	B	C	C	D
a1	b1	c1	c1	d1
a2	b2	c2	c2	d2
a3	b3	c3	c3	d3
a4	b4	c4	null	null
a5	b5	c5	null	null

Junta externa por derecha

- Retorna todas las filas de la relación que esta a la derecha del operador Outer.
- Pasos:
 - Primero se realiza el producto cartesiano,
 - Luego se aplica la condición de junta sobre las tuplas resultantes
 - Por ultimo se traen de la relación que esta a la derecha las tuplas que no fueron incluidas completando con valores nulos en los atributos de la relación que figura a la izquierda.



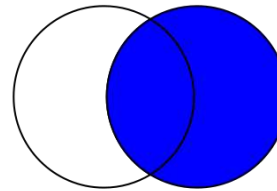
Junta externa por derecha

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

R2

C	D
c1	d1
c2	d2
c3	d3
c6	d6
c7	d7



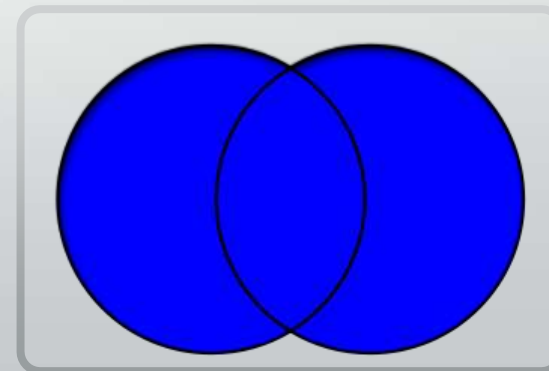
SELECT
FROM
ON

*
R1 RIGHT OUTER JOIN R2
R1.C = R2.C

A	B	C	C	D
a1	b1	c1	c1	d1
a2	b2	c2	c2	d2
a3	b3	c3	c3	d3
null	null	null	c6	d6
null	null	null	c7	d7

Junta externa completa

- Full Outer retorna todas las filas de la relación que esta a la izquierda y a la derecha del operador outer.
- Primero se realiza el producto cartesiano
- Luego se aplica la condición de junta sobre las tuplas resultantes
- Por ultimo se traen de la relación que esta a la izquierda las tuplas que no fueron incluidas completando con valores nulos en los atributos de la relación que figura a la derecha y de la relación que esta a la derecha las tuplas que no fueron incluidas completando con valores nulos en los atributos de la relación que figura a la izquierda.

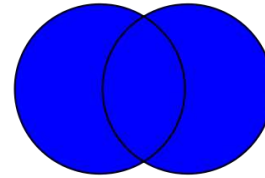


Junta externa completa

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

R2



C	D
c1	d1
c2	d2
c3	d3
c6	d6
c7	d7

SELECT
FROM
ON

*
R1 FULL OUTER JOIN R2
R1.C = R2.C

A	B	C	C	D
a1	b1	c1	c1	d1
a2	b2	c2	c2	d2
a3	b3	c3	c3	d3
a4	b4	c4	null	null
a5	b5	c5	null	null
null	null	null	c6	d6
null	null	null	c7	d7

Filtro en condición de junta externa

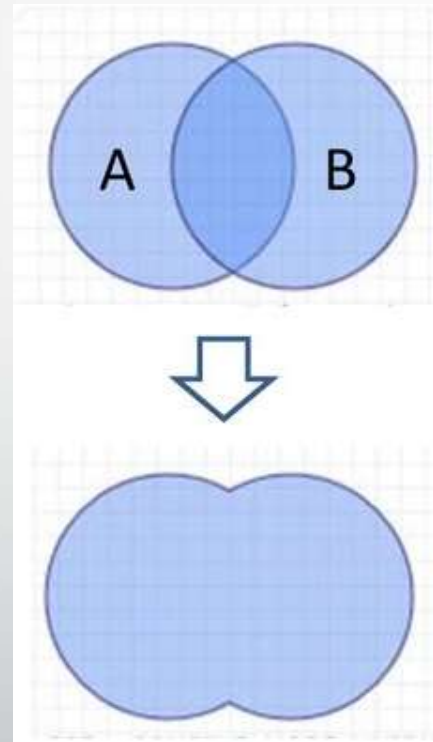
- Al aplicar el filtro en la cláusula WHERE para la tabla opuesta de un Outer Join, semánticamente se aplica el filtro al resultado final de la combinación, por lo que la junta externa pasa a ser una junta interna.
- Solución: el filtro debe aplicarse en la cláusula ON junto a un operador AND

Operador Unión

```
SELECT <lista de columnas>  
FROM <lista de tablas>  
[<otras clausulas>]
```

UNION

```
SELECT <lista de columnas>  
FROM <lista de tablas>  
[<otras clausulas>]  
[ORDER BY <columnas>]
```



El operador UNION combina los resultados de dos o más consultas y crea un conjunto de resultados que incluye filas recuperadas de las consultas participantes en UNION.

Todas las consultas se ejecutan de forma independiente, pero su salida se combina.

Reglas para la unión

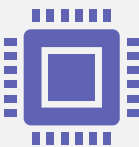
- El número de columnas y el orden de las columnas de todas las consultas deben ser los mismos.
- Los tipos de datos de las columnas en la tabla que implica en cada consulta deben ser iguales o compatibles.
- Por lo general, los nombres de columna devueltos se toman de la primera consulta.
- De forma predeterminada, UNION se comporta como UNION [DISTINCT], es decir, eliminó las filas duplicadas; sin embargo, el uso de la palabra clave ALL con UNION devuelve todas las filas, incluidos los duplicados



Los pasos siguientes muestran el orden de procesamiento lógico, u orden de enlaces, de una instrucción SELECT.



Este orden determina cuándo los objetos definidos en un paso están disponibles para las cláusulas en pasos posteriores.



Observe que la ejecución física real de la instrucción está determinada por el procesador de consultas y el orden puede variar en esta lista.

Orden de
procesamiento
lógico de
SELECT

Orden de procesamiento lógico de SELECT



1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. HAVING
7. SELECT
8. DISTINCT
9. ORDER BY
10. TOP

- Los alias creados en la lista SELECT no se pueden utilizar en pasos anteriores.
- En algunas bases de datos (como MySQL), el uso de alias creados en la lista SELECT está permitido en la cláusula GROUP BY y HAVING.
- Los alias de expresión no se pueden utilizar en otras expresiones dentro de la misma lista SELECT. Por ejemplo no se admite: `SELECT a + 1 AS x, x + 1 AS y`
- Cuando se utiliza INNER JOIN, no importa si especifica las expresiones lógicas en la cláusula WHERE o en la cláusula ON.
- La cláusula DISTINCT es redundante cuando se utiliza GROUP BY. Por lo tanto, no quitaría ninguna fila del conjunto de registros.

Subconsultas

- Es una sentencia SELECT que aparece dentro de otra sentencia SELECT.
- La subconsulta se puede encontrar en:
 - la lista de selección
 - en la cláusula FROM
 - EN LA cláusula WHERE
 - en la cláusula HAVING de la consulta principal.

Subconsultas

- Una subconsulta tiene la misma sintaxis que una sentencia SELECT normal exceptuando que aparece encerrada entre paréntesis.
- Tiene las siguientes restricciones:
 - No puede contener la cláusula ORDER BY
 - No puede ser la UNION de varias sentencias SELECT
 - Si la subconsulta aparece en la lista de selección o está asociada a un operador igual "=" solo puede devolver un único registro.

Subconsultas como una relación

- También se puede anidar subconsultas en la clausula FROM y estas serán tratadas como si fueran una relación más.
- Su uso es recomendado cuando se quieren obtener resultados de funciones agregadas con grupos y luego mostrar junto a columnas que no son necesarias en el grupo
- Por ejemplo: Informar el total de unidades vendidas por cada almacén

```
select a.almacen_nombre, v.total
from almacenes as a
inner join (
select almacen_id, sum(cantidad) as total
from ventas
group by almacen_id) as v on v.almacen_id = a.almacen_id
order by total desc
```

Subconsultas como una relación

- Por ejemplo: Obtener un conjunto de resultados donde cada fila corresponde a un código de tarea y por cada código de tarea se calcula la cantidad de empleados que tienen asignados esa tarea. Mostrar la descripción de la tarea y la cantidad calculada. Mostrar todas las tareas inclusive la que no tengan empleados asignados

```
SELECT t.tarea_desc, ISNULL(te.cantidad,0) AS cantidad
FROM Tarea AS t
LEFT OUTER JOIN (
    SELECT tarea_id, COUNT(*) AS cantidad
    FROM empleado
    GROUP BY tarea_id) AS te ON te.tarea_id = t.tarea_id
```

Pertenencia a conjuntos

- La conectiva IN o su negación NOT IN verifica la pertenencia de un valor en un conjunto de valores.

SELECT <lista de columnas>

FROM <lista de tablas>

WHERE <columna> [NOT] IN (<lista valores>)

- Ejemplo: Mostrar los autores que residen en las ciudades de Covelo, Palo Alto y San Jose.

select *

from autores

where ciudad in ('Covelo','Palo Alto', 'San Jose')

Pertenencia a conjuntos

- Las consultas que involucran la operación de intersección pueden ser implementados con el operador IN o EXISTS.
- La diferencia puede plantearse utilizando NOT IN o NOT EXISTS.
- Sintaxis:

```
SELECT <lista de columnas>
```

```
FROM <lista de tablas>
```

```
WHERE <columna> [NOT] IN (
```

```
    SELECT <columna>
```

```
    FROM <lista de tablas>)
```

- In determina que un valor debe existir en el resultado de la subconsulta.
- La subconsulta debe devolver una única columna.

Pertenencia a conjuntos

- Ejemplo: Listar los autores que escribieron libros

```
SELECT *  
FROM AUTORES  
WHERE AUTOR_ID IN (  
  SELECT AUTOR_ID  
  FROM TITULO_AUTOR)
```

- Ejemplo: Listar los autores que no escribieron libros

```
SELECT *  
FROM AUTORES  
WHERE AUTOR_ID NOT IN (  
  SELECT AUTOR_ID  
  FROM TITULO_AUTOR)
```

Comprobación de relaciones vacías

- La cláusula EXISTS devuelve verdadero si el resultado de una subconsulta no es vacío, es decir si devuelve al menos 1 fila.
- Sintaxis:

SELECT <lista de columnas>

FROM <lista de tablas>

WHERE EXIST (SELECT <lista de columnas>

FROM <lista de tablas>

WHERE <valor consulta principal> = <valor subconsulta>)

- La cláusula EXISTS devuelve verdadero si el resultado de una subconsulta tiene resultados, es decir si la subconsulta devuelve al menos 1 fila.
- No importa lo que se obtenga en la lista de columnas, de hecho puede ser un literal, lo que importa es la existencia de resultados.
- Toda consulta que utiliza la conectiva IN o NOT IN puede ser rescrita utilizando EXISTS o NOT EXISTS.

Comprobación de relaciones vacías

- Ejemplo: Listar los autores que escribieron libros:

```
SELECT *  
FROM AUTORES AS A  
WHERE EXISTS (  
  SELECT 1  
  FROM TITULO_AUTOR AS TA  
  WHERE TA.AUTOR_ID = A.AUTOR_ID)
```

- Ejemplo: Listar los autores que no escribieron libros:

```
SELECT *  
FROM AUTORES AS A  
WHERE NOT EXISTS (  
  SELECT 1  
  FROM TITULO_AUTOR AS TA  
  WHERE TA.AUTOR_ID = A.AUTOR_ID)
```


Comparación de un valor en un conjunto de valores

- Para comparar un valor contra un conjunto de valores retornados por una subconsulta, se utilizan los operadores SOME (también se permite ANY) y ALL a continuación del operador de comparación.
- SOME o ANY producen el valor verdadero cuando el valor que se compara cumple la condición de comparación con alguno de la lista de valores, ALL retorna el valor verdadero cuando el valor que se compara cumple la condición de comparación con todos los valores de la lista de valores.

Comparación de un valor en un conjunto de valores

- Sintaxis:

```
SELECT <lista de columnas>  
FROM <lista de tablas>  
WHERE <columna> <operador> SOME (SELECT <columna>  
FROM <lista de tablas>)
```

- Ejemplo: Listar los empleados cuyo nivel de cargo no es el mínimo.

```
select *  
from empleados  
where nivel_cargo > some (  
    select nivel_cargo  
    from empleados)
```

Comparación de un valor en un conjunto de valores

- Sintaxis:

```
SELECT <lista de columnas>
```

```
FROM <lista de tablas>
```

```
WHERE <columna> <operador> ALL (SELECT <columna>
```

```
FROM <lista de tablas>)
```

- Listar los legajos del personal contratado cuyo valor horario de 100 horas es superior a todos los sueldos mensuales

```
SELECT legajo
```

```
FROM contratado
```

```
WHERE valor_hora*100 > ALL ( SELECT sueldo_mensual
```

```
FROM asalariado)
```

Anidar subconsultas

- Las subconsultas pueden anidarse de forma que una subconsulta aparezca en la cláusula WHERE (por ejemplo) de otra subconsulta que a su vez forma parte de otra consulta principal.
- Por ejemplo mostrar los empleados que son “secretaria” y no son “ingeniero”

```
SELECT LEGAJO, APELLIDO, NOMBRE  
FROM EMPLEADO  
WHERE LEGAJO IN (SELECT LEGAJO  
FROM SECRETARIA  
WHERE LEGAJO NOT IN (SELECT LEGAJO  
FROM INGENIERO)  
)
```

Inserción

- Insert agrega una o varias tuplas en una relación.
- La inserción de una tupla se expresa de la forma:

```
INSERT [INTO] r [(r1, r2, r3, ..., rn)]  
VALUES (valor1, valor2, ..., valorn)
```
- La especificación del nombre de los atributos es optativa, de no especificarse se insertan los valores según el orden de los atributos en el esquema de relación r (es decir valor1 se insertara en el primer atributo, valor n en el ultimo atributo).
- Si la tabla tiene reglas de validación estas deben ser aprobadas para completar la operación.
- Por ejemplo:

```
INSERT INTO EMPLEADOS  
VALUES ('ABC90016M', 'DIEGO', 'X', 'SAMPAOLI', 6, 180, 9901, '20150630')
```

O bien

```
INSERT INTO EMPLEADOS (EMPLEADO_ID, NOMBRE, APELLIDO, CARGO_ID, NIVEL_CARGO)  
VALUES ('ABC90015M', 'DIEGO', 'MARADONA', 5, 150)
```

Inserción con subconsulta

- También se puede insertar en una relación el resultado de una subconsulta, siempre que los valores que se insertan pertenezcan a los dominios de los atributos. En este caso la forma general es:

```
INSERT [INTO] r [(r1, r2, r3, ..., rn)]  
SELECT valor1, valor2, ..., valorn  
<cláusulas>
```

- Ejemplo: Agregar un “Descuento Cliente” a aquellos almacenes que no lo tengan. La cantidad mínima será de 5, cantidad máxima 25 y el descuento 5%.

```
INSERT INTO descuentos  
SELECT 'DESCUENTO CLIENTE', ALMACEN_ID, 5, 25, 5  
FROM ALMACENES  
WHERE ALMACEN_ID NOT IN (  
    SELECT ALMACEN_ID FROM DESCUENTOS WHERE ALMACEN_ID IS NOT NULL  
)
```

Modificación

- Se puede modificar el valor de una o mas columnas pertenecientes a una o mas filas mediante el comando Update

```
UPDATE r
```

```
SET r1 = valor1, r2 = valor2, ... , rn = valor n
```

```
[ WHERE p ]
```

- Ejemplo: incrementar el nivel de cargo de todos los empleados en 5 puntos

```
UPDATE empleados
```

```
SET nivel_cargo = nivel_cargo + 5
```

Modificación

- Lo mas habitual es limitar la acción a un determinado numero de filas, utilizando para ello la cláusula where.
- Si se omite la cláusula WHERE se actualizan todas las filas de la relación.
- Ejemplo: incrementar el nivel de cargo del empleado con id 'ABC90016M' en 5 puntos

```
UPDATE empleados  
SET nivel_cargo = nivel_cargo + 5  
WHERE empleado_id = 'ABC90016M'
```


Modificación

- También en el comando UPDATE puede utilizarse la cláusula FROM para modificar tuplas de una relación que se juntan con tuplas de otra relación.
- Ejemplo: establecer el nivel de cargo máximo a los empleados que estén sobre el mismo

```
UPDATE empleados
```

```
SET nivel_cargo = nivel_maximo
```

```
From empleados
```

```
inner join cargos as c on c.cargo_id = empleados.cargo_id
```

```
where nivel_cargo > nivel_maximo
```

Borrado

- La operación de borrado consiste en la eliminación de una o mas tuplas de una relación.
DELETE [FROM] r
[WHERE p]
- Donde r representa una relación y p un predicado.
- La sentencia DELETE eliminara de la relación r todas las tuplas que cumplan con el predicado p , la cláusula WHERE es optativa, si no esta presente se eliminan todas las tuplas de la relación r .
- Ejemplos: Borrar el contenido de la tabla técnico
DELETE FROM técnico
- Ejemplo: Borrar al ingeniero de legajo 1324
delete from empleados
where empleado_id = 'ZXY90016M'

Borrado

- El predicado p ser un predicado que incluya un nuevo select (siempre y cuando se determine una condición lógica).
- Ejemplo: Borrar de la tabla secretaria a las secretarias que ganan mas de 1500 \$ mensuales

```
DELETE FROM secretaria  
WHERE legajo IN (SELECT legajo  
FROM asalariado  
WHERE sueldo_mensual > 1500 )
```
- Ejemplo: Idem anterior utilizando operador Inner Join

```
DELETE FROM secretaria s  
INNER JOIN asalariado a  
ON s.legajo = a.legajo  
WHERE a.sueldo_mensual > 1500
```

Paginación

- Permite limitar el número de filas en el conjunto de resultados.
 - `OFFSET {cantidad_filas} { ROW | ROWS }`
 - `[FETCH { FIRST | NEXT } {cantidad_filas} { ROW | ROWS } ONLY]`
- OffSet indica cuantas filas a saltar
- Fetch número de filas a retornar
- Limitaciones:
 - Es obligatorio el uso de Order By
 - No se puede combinar con Top
 - La cantidad de filas debe ser una expresión numérica entera y no soporta subconsulta.

Paginación

- Por ejemplo mostrar los empleados omitiendo los cinco primeros

```
SELECT LEGAJO, APELLIDO, NOMBRE  
FROM EMPLEADO  
ORDER BY LEGAJO  
OFFSET 5 ROWS
```

- Por ejemplo mostrar 5 empleados omitiendo los diez primeros

```
SELECT LEGAJO, APELLIDO, NOMBRE  
FROM EMPLEADO  
ORDER BY LEGAJO  
OFFSET 10 ROWS  
FETCH NEXT 5 ROWS ONLY
```