UT3. Queries

“Database”

DAM-DAW

Relational Algebra

- Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances. There are two kinds of query languages – relational algebra and relational calculus.
- **Relational algebra**, first created by Edgar F. Codd while at IBM, is a family of algebras with a well-founded semantics used for modelling the data stored in relational databases, and defining queries on it
- The fundamental operations of relational algebra are as follows –
  - **Select**
  - **Project**
  - **Union**
  - **Set difference**
  - **Cartesian product**
- Extended operators are those operators which can be derived from basic operators.There are mainly three types of extended operators in Relational Algebra:
  - **Join**
  - **Intersection**
  - **Divide**

Relational Algebra

Operaciones Básicas	Operaciones Unarias	- selección - proyección
	Operaciones Binarias	- unión - diferencia - producto cartesiano
Operaciones Derivadas	Intersección Cociente Combinación (join)	

Relational Algebra: Selection

Selection coge la tabla entera viendo un parámetro

R =

Nombre	Edad
Antonio	20
Juan	10
Leonor	30

$\sigma_{edad> 15}(R) =$

Nombre	Edad
Antonio	20
Leonor	30

Relational Algebra: Proyection

proyection coge una columna concreta en una tabla

R =

Nombre	Edad
Antonio	20
Juan	10
Leonor	30

$\Pi_{\text{nombre}}(R) =$

Nombre
Antonio
Juan
Leonor

Intersect, Union, Union All, Minus

- Please, watch this video
  - https://www.youtube.com/watch?v=bL5UX-p1wMc&feature=emb_rel_end

Relational Algebra:Union

- UNION

JUNTA DOS TABLAS Y MERGE LOS DUPLICADOS

  - This operation is possible if the tables have the same number of columns and compatibles data types
  - Repeatead rows will only appear.

EMPLE1

Nemple	Nombre
1001	Rosa
1005	Ana

EMPLE2

Nemple	Nombre
2001	Pilar
2010	Pepe
1005	Ana

EMPLE1 U EMPLE2

Nemple	Nombre
1001	Rosa
1005	Ana
2001	Pilar
2010	Pepe

```
select nombre from EMPLE1
UNION
select nombre from EMPLE2
```

Relational Algebra: Set Difference

ELIMINA LOS ROWS IGUALES EN DOS TABLAS DISTINTAS Y DEJA SOLO LOS ÚNICOS A CADA TABLA

- Set difference:
  - This operation is possible if the tables have the same number of columns and compatibles data types

r=

A	B
a_1	b_1
a_2	b_2
a_3	b_2
a_4	b_4

s=

A	B
a_1	b_1
a_2	b_2
a_5	b_5

EMPLE1

Nemple	Nombre
1001	Rosa
1005	Ana

EMPLE2

Nemple	Nombre
2001	Pilar
2010	Pepe
1005	Ana

r∪s=

A	B
a_1	b_1
a_2	b_2
a_3	b_2
a_4	b_4
a_5	b_5

r-s=

A	B
a_3	b_2
a_4	b_4

EMPLE1 – EMPLE2

Nemple	Nombre
1001	Rosa

EMPLE2 – EMPLE1

Nemple	Nombre
2001	Pilar
2010	Pepe

STANDARD SQL

```
SELECT nombre  from EMPLE2
MINUS
SELECT nombre from EMPLE1
```

MYSQL

```
SELECT nombre from EMPLE2
where nombre NOT IN
(SELECT nombre from EMPLE1)
```

Relational Algebra: Cartesian product

- This operation can be done among tables even if they don't have the same number of columnsThe result is "like" another table wich will have the selected columns from the different tables that take part in the operation.
- There can be ambiguity if we have columns with the same name in the tables..

A{X,Y,T} =

X	Y	T
10	22	1
11	25	2

B{W,Z} =

W	Z
33	54
37	98
42	100

C (X,Y,T,W,Z):= A * B =

X	Y	T	W	Z
10	22	1	33	54
10	22	1	37	98
10	22	1	42	100
11	25	2	33	54
11	25	2	37	98
11	25	2	42	100

Exercise: Cartesian product Ventas* Artículos

VENTAS

Cod	Fecha	Cantidad
5100	18/11/1999	100
5200	19/11/1999	120
5100	19/11/1999	45

ARTICULOS

Código	Denominacion	Existencias	Pvp
5100	cometas	500	78
5200	raquetas	250	90

mysql

```
select * from ventas v
CROSS JOIN
select * from articulos a
```

Relational Algebra: Intersection

- Intersection
  - Extended operation. The intersection between two tables is another table formed by the rows that appears in both tables.
  - The two tables must have the same number of columns as well as the same data types for each
  - Tabla 1 ∩ Tabla 2.
  - It can be expressed with the difference operation:
    - Tabla1-(Tabla1-Tabla2)

EMPLE1

Nemple	Nombre
1001	Rosa
1005	Ana

EMPLE2

Nemple	Nombre
2001	Pilar
2010	Pepe
1005	Ana

NumEmp	Nombre
1005	Ana

Relationa Algebra: Divide

- Optional for the student

Tomaremos como tablas de partida:

Tabla1

A	B	C
1	6	4
4	3	1
7	0	2
1	2	3

Tabla2

B	C
2	3
7	1

Para obtener la tabla cociente primero calculamos una tabla intermedia con las columnas de Tabla1 que no estén en Tabla2.

A
1
4
7
1

A continuación concatenamos esta tabla con Tabla2 (TablaIntermedia x Tabla2)y obtenemos la tabla intermedia:

A	B	C
1	**2**	**3**
1	7	1
4	2	3
4	7	1
7	2	3
7	7	1
1	**2**	**3**
1	7	1

De esta tabla extraemos las filas que estén contenidas en Tabla1, y de esta fila tomamos solo la columna A de Tabla1

A
1

Relational Algebra: Join

- Subset of cartesian product.
- A JOIN is a means for combining columns from one (self-join) or more tables by using values common to each.
- Representation:
  - (tabla1 * tabla 2) $_{\text{criterio}}$

VENTAS

Cod	Fecha	Cantidad
5100	18/11/1999	100
5200	19/11/1999	120
5100	19/11/1999	45

ARTICULOS

Código	Denominacion	Existencias	Pvp
5100	cometas	500	78
5200	raquetas	250	90

(Ventas * Artículos) Cod= Código

Cod	Fecha	Cantidad	Denominación	Enxistencias	PvP
5100	18/11/1999	100	Cometas	500	78
5200	19/11/1999	120	Raquetas	25	90
5100	19/11/1999	45	Cometas	500	78

Example of tables

Tabla **EMP**

Campo	Nulo	Tipo
EMPNO	NOT NULL	integer
ename		text(10)
mgr		integer
job		text(9)
hiredate		DATE
sal		long
comm		long
deptno	NOT NULL	byte

Tabla **DEPT**

Campo	NULL	Tipo
DEPTNO	NOT NULL	Byte
DNAME		text(14)
LOC		text(13)

Comparison operators

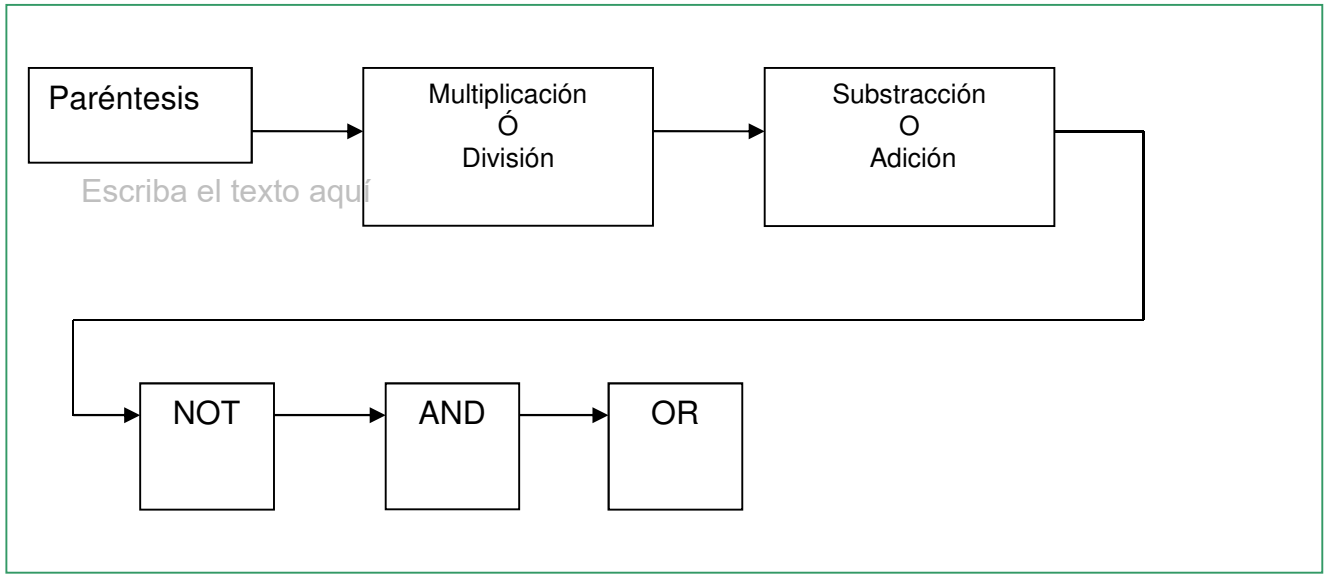
Comando	Descripción
<, LOWER THAN, LT	Menor que
>, GREATER THAN, GT	Mayor que
<>, NOT	Distinto de
<=, LE	Menor o igual que
>=, GE	Mayor o igual que
=, EQUAL, EQ	Igual que
BETWEEN	Especifica un intervalo de valores desde, hasta
LIKE	➢ Utilizado para comparar con un modelo "____%" (o '____*' en algunos SGBD)
IN	➢ Utilizado para especificar los valores de un dominio o conjunto IN (v1, v2, v3,…)

Arithmetic and logical operators

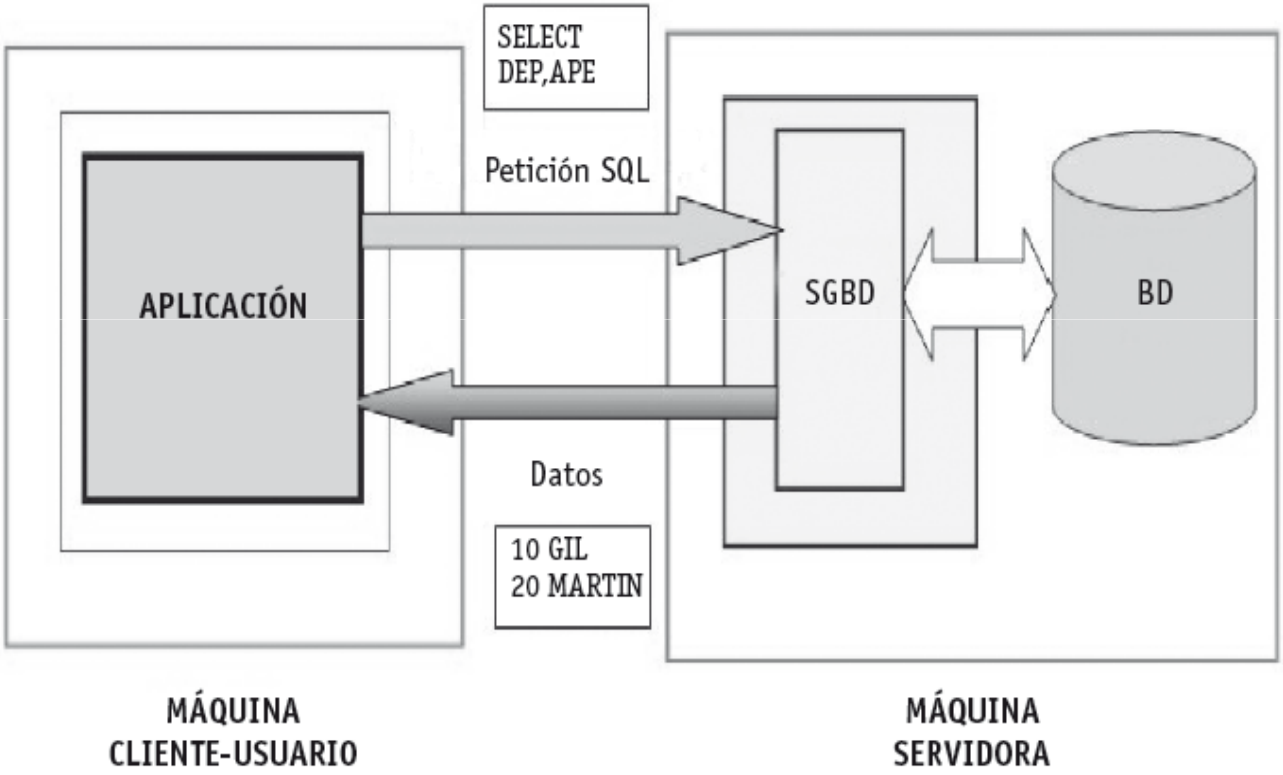
- **Logical operators**

Comando	Descripción
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor verdadero sólo si ambas son ciertas
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor verdadero sólo con que una condición sea verdadera
NOT	Negación lógica. Devuelve el valor contrario de la expresión

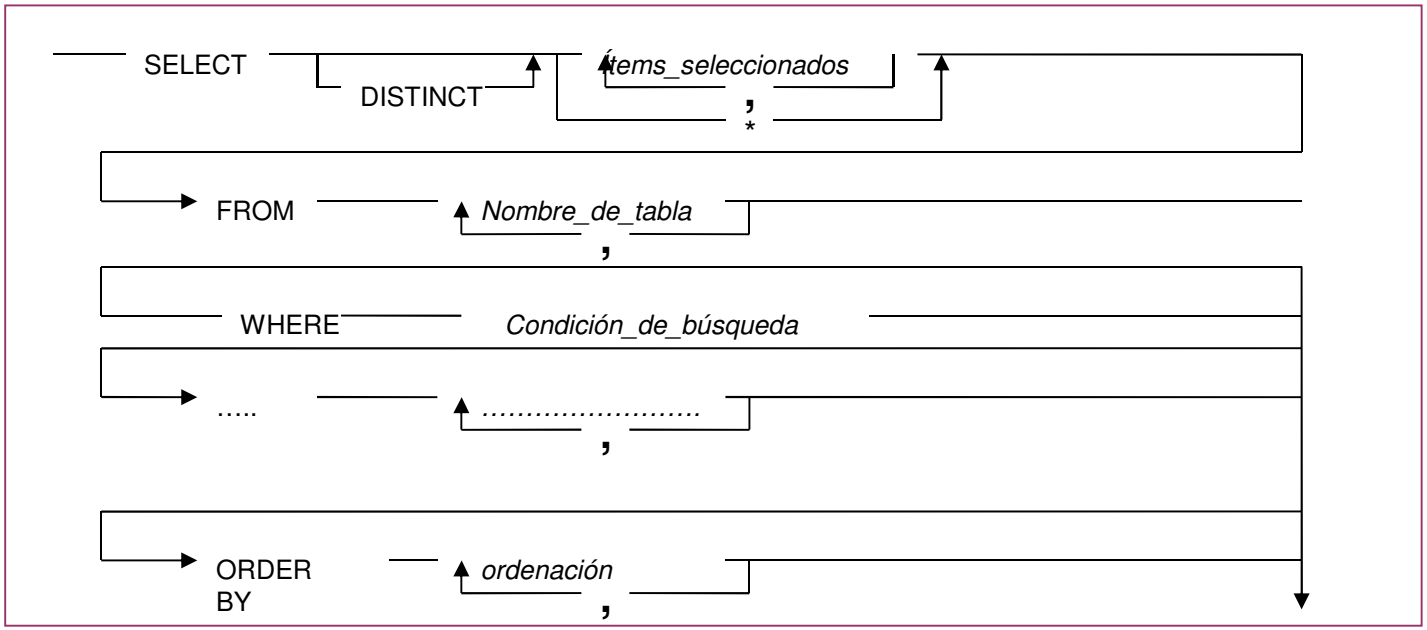
- **Operator precedence:**



SELECT


SELECT
DEP,APE
Petición SQL
APLICACIÓN
SGBD
BD
Datos
10 GIL
20 MARTIN
MÁQUINA
CLIENTE-USUARIO
MÁQUINA
SERVIDORA

SELECT: Reduced Syntax



- **Select clause**
  - FROM: to select the table/s we want to read
  - WHERE: to filter at row level
  - ORDER BY: to order the results by one or more columns (ASC or DESC). Default: ASC
  - GROUP BY: To group files according to field contents
  - HAVING: To filter at group level. It allows aggregate functions

SELECT…. FROM ….

- Reduced Syntax:

  SELECT [ALL|DISTINCT] [expre_colum1, expre_colum2, ..., expre_column | *]

  FROM [nombre_tabla1, nombre_tabla2, ..., nombre_tablan]
- Select * from emp ;
- Select eno, dname from emp;
- Select emp.eno, emp.dname from emp;
- Select distinct loc from dept;
- Select distinct a,b from T;
- Constants: Select “the name is ”, ename from emp;
- Expressions: Select sal*10/100 from emp
- Exercise
  - **http://www.w3schools.com/sql/sql_select.asp**

SELECT …FROM… WHERE…

- **FROM**
  - Table qualifier
    - Select * from **nombrebaseDatos_esquema.nombreTabla**
    - Select * from (select * from emp); Where is the table name?
- **USO DE ALIAS**
  - https://www.w3schools.com/sql/sql_alias.asp
  - In tables: Select t1.ename from emp t1;
  - In columns: select sal + 5 “SubidaSalarial” from emp;
- **WHERE: Filter the files**
  - The where clause can be writen in different ways. We have always to think about using the indexes in case the are defined
    - E.g.
      - Select * from emp where UPPER(ename) = “FELIPE”;
      - In this case the index, if it were defined by ename column, wouldn’t be used

SELECT …FROM… WHERE…

- **Comparison operators**
  - Select * from emp where sal > 100
  - Select ename from emp where sal > 100 and comm > 0
  - **Between**
    - Select * from emp where sal between 10 and 100;

    or
    - Select * from emp where sal > 0 and sal < 100;
  - **In**
    - Select * from emp where empno in (10,20)

    or
    - Select * from emp where empno = 10 or empno = 20;

  - **LIKE**
    - For comparing strings
      - ? : substitutes one characer(in some RDBMS _)
      - * : substitutes any number of characters (in some RDMBS : %)
    - The following sentences doesn't use index if it's defined
      - Select * from emp where ename like '%an%';
      - Select * from emp where ename like '%ez';
    - Not too bad for index use
      - Select * from emp where ename like 'F%';
  - **NULL VALUES**
    - Select * from emp where comm IS NULL (= NULL doesn't work)
    - Select * from emp where comm IS NOT NULL (<> doesn't work)

SELECT …FROM… WHERE…ORDER BY..

- ORDER BY
  - The select sentence doesn't return the rows in a speficic order. We have to use ORDER BY if we want some order
  - ASC, DESC. By default is ASC

  E.g:
  - Select ename from emp order by ename
  - Select * from emp order by ename, sal
  - Select ename, sal from emp order by 1, 2
  - Select ename, sal from emp where sal > 10 order by sal DESC, ename ASC
- The order by clause affects the response time of the sentence

MySQL functions

- The index, if defined, won't be used if we use functions at the left of the where clause
- There are:
  - Numeric functions
  - String functions
  - Date Functions
  - ..
- The functions can be different between RDBMS

- https://www.w3schools.com/sql/sql_ref_mysql.asp

SQL. Aggregate Functions

- **COUNT**(attribute)
- **AVG**(attribute)
- **MIN**(attribute)
- **MAX**(attribute)
- **SUM**(attribute)

e.g.

```
Select MIN(SAL),MAX(SAL),AVG(SAL),COUNT(*) from EMP
```

- **Difference between**
  - **count(*) :** number of rows
  - **count(attribute):** number of rows where attribute is not null

SELECT: GROUP BY and HAVING

- **Grop by**
  - –https://www.w3schools.com/sql/sql_groupby.asp
  - –E.g.
    - OK: SELECT DEPT_NO, COUNT(*) FROM EMPLE **GROUP BY** DEPT_NO;
    - OK: Select deptno, sum(comm+sal) from emp where ename like 'F%' group by deptno
    - Error
      - Select deptno, sal from emp group by deptno;
      - How can you fix it?
- **Having**
  - To filter the groups obtained by group by clause
  - Having requires GROUP BY.
  - E.g.
    - Select deptno, SUM(sal)
      FROM EMP
      GROUP BY Deptno
      HAVING Deptno IN (1,3,5)
      - Rewrite the previous sentence with where
    - Select deptno, count(*)
      FROM EMP
      GROUP BY depteno
      HAVING count(*) >4

SQL: Subqueries

- **Subqueries can be nested**

  Select * From EMP
  WHERE SAL > (select avg(sal) from EMP)

- **Subqueries will be executed before the outer query**

- **Comparison in subqueries (>, <, <>, <=, >=, =). They must return only one value or none**

  SELECT ename FROM EMP WHERE job = (SELECT job FROM EMP WHERE ename ='John');
  What happens if the subquery returns mor than one value?

- **IN operator. We can use it for subqueries that return more than one value**

  SELECT ename FROM EMP WHERE job IN (SELECT job FROM EMP WHERE deptno=20);

- **EXISTS , NOT EXISTS**
  - **https://www.w3schools.com/sql/sql_exists.asp**
  - Departments with employees:

  SELECT Dname, DEPTNO FROM DEPT WHERE
  EXISTS (SELECT * FROM EMP WHERE EMP.DEPTNO = DEPT.DEPTNO;

- **ANY y ALL**
  - **https://www.w3schools.com/sql/sql_any_all.asp**

  SELECT * FROM EMP WHERE SALARIO = ANY (SELECT SALARIO FROM EMP WHERE DEPTNO=30);

  SELECT * FROM EMP WHERE SALARIO < ALL (SELECT SALARIO FROM EMP WHERE DEPTNO=30);

SQL. Set operators

- In the following operations the number of columns and data types must match.
- **UNION**
  - Combines the results of the different selects.
  - Duplicate rows are not considered only once.
  - **UNION ALL**: unlike 'UNION', it considers duplicate rows

```
SELECT COL1, COL2, ... FROM TABLA1 WHERE CONDICION
UNION
SELECT COL1, COL2, ... FROM TABLA2 WHERE CONDICION
```

- **INTERSECT**
  - Returns the rows that are identical in both
  - Duplicate rows are considered only once

```
SELECT COL1, COL2, ... FROM TABLA1 WHERE CONDICION
INTERSECT
SELECT COL1, COL2, ... FROM TABLA2 WHERE CONDICION
```

- **MINUS**
  - Returns those rows which are in the first select and not in the second.
  - Duplicated files in the first set are only considered before comparing with the second set
  - Syntax

```
SELECT COL1, COL2, ... FROM TABLA1 WHERE CONDICION
MINUS
SELECT COL1, COL2, ... FROM TABLA2 WHERE CONDICION;
```

  - Exercise: how to do a Minus operation in MySQL?

SQL. Cartesian Product and Join

- **Cartesian Product**
  - Syntax
    Select col1, col2,… from tabla1, tabla 2;
- **JOIN**
  - Subset of a Cartesian product
  - Is the result of adding a where condition to a Cartesian product
  - https://www.w3schools.com/sql/sql_join.asp
  - Possible ambiguity if there are columns in different tables with the same name
    - Suggestion: use always the alias
  - Syntax
    SELECT tabla1.c1, tabla1.c2,…. tabla2.c1, tabla2.c2,….
    FROM tabla1, tabla2,...
    WHERE tabla1.c1 = tabla2.c1 …
    or
    SELECT tabla1.c1, tabla1.c2,…. tabla2.c1, tabla2.c2,….
    FROM tabla1 INNER JOIN tabla 2 ON tabla1.c1 = tabla2.c1 …

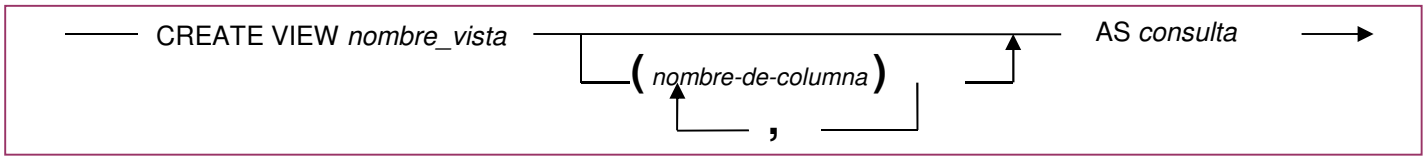
  - If there is no Where in a join, it can be a Cartesian product !!!!
  - E.g.
    Select e.eno, e.deptno, d.loc from emp e, dept d
    where e.deptno = d.deptno

SELECT: OUTER JOIN

- **OUTER JOIN**
  - **https://www.w3schools.com/sql/sql_join.asp**
  - **LEFT (OUTER) JOIN**:
    - Return all records from the left table, and the matched records from the right table
  - **RIGHT (OUTER) JOIN**:
    - Return all records from the right table, and the matched records from the left table
  - **FULL (OUTER) JOIN**:
    - Return all records when there is a match in either left or right table

Views

https://www.w3schools.com/sql/sql_view.asp



Select * from nombre_vista;
Select nombre-de-columna,…. From nombre_vista;

References

- MySQL:
  - http://dev.mysql.com
- Tutorial:
  - http://www.w3schools.com/
  - http://www.aulaclic.es/sql/
  - https://www.w3resource.com/mysql/mysql-tutorials.php