

SQL Avanzado



Tablas de verdad en SQL

- NULL puede tener muchos significados
- ¿Cómo se comporta NULL al compararlo con valores booleanos?

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN
OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN
NOT			
TRUE	FALSE		
FALSE	TRUE		
UNKNOWN	UNKNOWN		


Tablas de verdad en SQL

- Para comparar un valor contra NULL es mejor utilizar el operador IS o IS NOT
- En SQL NULL = NULL no necesariamente es true.

```
SELECT    Fname, Lname
FROM      EMPLOYEE
WHERE     Super_ssn IS NULL;
```

Consultas anidadas

- Son consultas dentro de otra consulta
- Normalmente aparecen como condición en el WHERE, pero también pueden usarse en el HAVING o en la lista de columnas seleccionadas



**SELECT
FROM
WHERE**

DISTINCT Pnumber
PROJECT
Pnumber **IN**

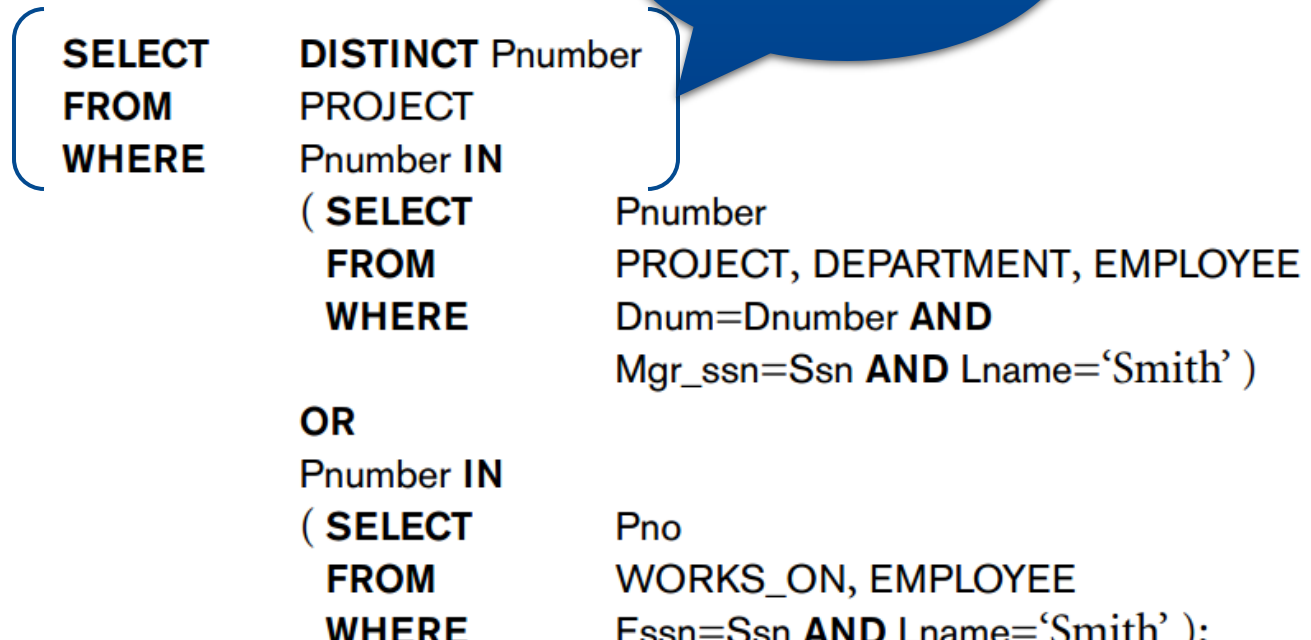
(**SELECT** Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber **AND**
Mgr_ssn=Ssn **AND** Lname='Smith')

OR
Pnumber **IN**
(**SELECT**
FROM
WHERE

Pno
WORKS_ON, EMPLOYEE
Essn=Ssn **AND** Lname='Smith');

Consultas anidadas


- Son consultas dentro de otra consulta
- Normalmente aparecen como condiciones de selección, pero también pueden usarse en el HAVING o en la lista de columnas seleccionadas



```
SELECT  
FROM  
WHERE  
DISTINCT Pnumber  
PROJECT  
Pnumber IN  
(  
  SELECT  
  FROM  
  WHERE  
  Pnumber  
  PROJECT, DEPARTMENT, EMPLOYEE  
  Dnum=Dnumber AND  
  Mgr_ssn=Ssn AND Lname='Smith' )  
OR  
Pnumber IN  
(  
  SELECT  
  FROM  
  WHERE  
  Pno  
  WORKS_ON, EMPLOYEE  
  Essn=Ssn AND Lname='Smith' );
```

Consultas anidadas

- Son consultas dentro de otra consulta
- Normalmente aparecen como condición en el WHERE, pero también pueden usarse en el HAVING o en la lista de columnas seleccionadas



**SELECT
FROM
WHERE**

DISTINCT Pnumber
PROJECT
Pnumber **IN**

(**SELECT
FROM
WHERE**

Pnumber
PROJECT, DEPARTMENT, EMPLOYEE
Dnum=Dnumber AND
Mgr_ssn=Ssn AND Lname='Smith')


OR
Pnumber **IN**
(**SELECT
FROM
WHERE**

Pno
WORKS_ON, EMPLOYEE
Essn=Ssn AND Lname='Smith');

Nested query

Consultas anidadas

- Son consultas dentro de otra consulta
- Normalmente aparecen como condición en el WHERE, pero también pueden usarse en el HAVING o en la lista de columnas seleccionadas



**SELECT
FROM
WHERE**

DISTINCT Pnumber
PROJECT
Pnumber **IN**
(**SELECT**
FROM
WHERE

Pnumber
PROJECT, DEPARTMENT
Dnum=Dnumber **AND**
Mgr_ssn=Ssn **AND** Lname

OR
Pnumber **IN**
(**SELECT**
FROM
WHERE


Pno
WORKS_ON, EMPLOYEE
Essn=Ssn **AND** Lname='Smith');



Nested query

Consultas anidadas

- Son consultas dentro de otra consulta
- Normalmente aparecen como condición en el WHERE, pero también pueden usarse en el HAVING o en la lista de columnas seleccionadas



**SELECT
FROM
WHERE**

DISTINCT Pnumber
PROJECT
Pnumber **IN**
(**SELECT**
FROM
WHERE

OR
Pnumber **IN**
(**SELECT**
FROM
WHERE

Pnumber
PROJECT, DEPARTMENT, EMPLOYEE
Dnum=Dnumber **AND**
Mgr_ssn=Ssn **AND** Lname='Smith')


Pno
WORKS_ON, EMPLOYEE
Essn=Ssn **AND** Lname='Smith');

Puede haber
ambigüedad

Consultas anidadas

- Son consultas dentro de otra consulta
- Normalmente aparecen como condición en el WHERE, pero pueden usarse en el HAVING o en la lista de columnas

Se asume que cualquier atributo no calificado es de la consulta más anidada



**SELECT
FROM
WHERE**

DISTINCT Pnumber
PROJECT
Pnumber **IN**
(**SELECT**
FROM
WHERE


Pnumber
PROJECT, DEPARTMENT, EMPLOYEE
Dnum=Dnumber **AND**
Mgr_ssn=Ssn **AND** Lname='Smith')

OR
Pnumber **IN**
(**SELECT**
FROM
WHERE

Pno
WORKS_ON, EMPLOYEE
Essn=Ssn **AND** Lname='Smith');

Consultas anidadas


- Si la consulta anidada hace referencia a un atributo de la consulta externa, se dice que es una consulta anidada correlacionada



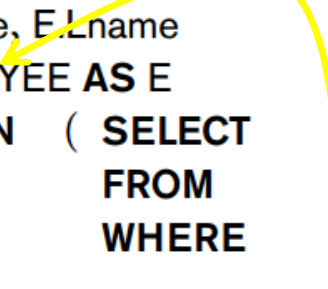
```
SELECT      E.Fname, E.Lname
FROM        EMPLOYEE AS E
WHERE       E.Ssn IN ( SELECT      Essn
                        FROM        DEPENDENT AS D
                        WHERE       E.Fname=D.Dependent_name
                        AND E.Sex=D.Sex );
```

Consultas anidadas

- Si la consulta anidada hace referencia a un atributo de la consulta externa, se dice que es una consulta anidada correlacionada




```
SELECT      E.Fname, E.Lname
FROM        EMPLOYEE AS E
WHERE       E.Ssn IN ( SELECT
                        FROM
                        WHERE
                        Essn
                        DEPENDENT AS D
                        E.Fname=D.Dependent_name
                        AND E.Sex=D.Sex );
```



Consultas anidadas con EXISTS

- La misma consulta se puede reformular de la siguiente forma:



```
SELECT      E.Fname, E.Lname
FROM        EMPLOYEE AS E
WHERE       EXISTS ( SELECT      *
                        FROM        DEPENDENT AS D
                        WHERE       E.Ssn=D.Essn AND E.Sex=D.Sex
                                AND E.Fname=D.Dependent_name);
```

Retorna true si la
consulta anidada
devolvió al menos
un registro


Convertir el resultado en un conjunto

- Mediante la palabra reservada DISTINCT un *multiset* con tuplas repetidas se puede convertir en un conjunto:

SELECT	DISTINCT Essn
FROM	WORKS_ON
WHERE	Pno IN (1, 2, 3);

JOIN


- JOIN es el equivalente del operador relacional \bowtie



```
SELECT      Fname, Lname, Address  
FROM        (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)  
WHERE       Dname='Research';
```

JOIN

- JOIN es el equivalente del operador relacional \bowtie




```
SELECT  Fname, Lname, Address
FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE   Dname='Research';
```



```
SELECT  Fname, Lname, Address
FROM    EMPLOYEE, DEPARTMENT
WHERE   Dname='Research' AND Dnumber=Dno;
```

JOIN

- JOIN es el equivalente del operador relacional \bowtie




**SELECT
FROM
WHERE**

Fname, Lname, Address
(EMPLOYEE **JOIN** DEPARTMENT **ON** Dno=Dnumber)
Dname='Research';



**SELECT
FROM
WHERE**


Fname, Lname, Address
EMPLOYEE, DEPARTMENT
Dname='Research' **AND** Dnumber=Dno;



Mucho más
clara

JOIN

- JOIN es el equivalente del operador relacional \bowtie



```
SELECT      Fname, Lname, Address
FROM        (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE       Dname='Research';
```




```
SELECT      Fname, Lname, Address
FROM        EMPLOYEE, DEPARTMENT
WHERE       Dname='Research' AND Dnumber=Dno;
```

No es tan fácil entender cual
es la condición de join y la
de select

JOIN


- Para las variantes del JOIN existe el operador LEFT JOIN, RIGHT JOIN y FULL OUTER JOIN



```
SELECT    E.Lname AS Employee_name,  
          S.Lname AS Supervisor_name  
FROM      (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S  
          ON E.Super_ssn=S.Ssn);
```

JOIN

- Cuando un JOIN involucra múltiples tablas, se le llama multiway joy




```
SELECT      Pnumber, Dnum, Lname, Address, Bdate
FROM        ((PROJECT JOIN DEPARTMENT ON Dnum=Dnumber)
JOIN EMPLOYEE ON Mgr_ssn=Ssn)
WHERE       Plocation='Stafford';
```

Funciones agregadas en SQL

- Las funciones agregadas se utilizan para resumir información de múltiples tuplas en una sola
- El agrupamiento se utiliza para crear subgrupos antes de resumir la información
- Para poder utilizar funciones agregadas, es necesario agrupar. Implícitamente las funciones agregadas agrupan.

Funciones agregadas en SQL



SELECT
FROM

SUM (Salary), **MAX** (Salary), **MIN** (Salary), **AVG** (Salary)
EMPLOYEE;

SELECT
FROM
WHERE

SUM (Salary), **MAX** (Salary), **MIN** (Salary), **AVG** (Salary)
(EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
Dname='Research';


SELECT
FROM

COUNT (*)
EMPLOYEE;

SELECT
FROM
WHERE

COUNT (*)
EMPLOYEE, DEPARTMENT
DNO=DNUMBER AND DNAME='Research';

Funciones agregadas en SQL



SELECT
FROM

SUM (Salary), **MAX** (Salary), **MIN** (Salary), **AVG** (Salary)
EMPLOYEE;

Implica
agrupamiento
por esta columna

SELECT
FROM
WHERE

SUM (Salary), **MAX** (Salary), **MIN** (Salary), **AVG** (Salary)
(**EMPLOYEE** **JOIN** **DEPARTMENT** **ON** Dno=Dnumber)
Dname='Research';

SELECT
FROM


COUNT (*)
EMPLOYEE;

SELECT
FROM
WHERE

COUNT (*)
EMPLOYEE, **DEPARTMENT**
DNO=**DNUMBER** **AND** **DNAME**='Research';

Funciones agregadas en SQL

```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM      EMPLOYEE;
```



```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE      Dname='Research';
```


```
SELECT      COUNT (*)
FROM      EMPLOYEE;
```

```
SELECT      COUNT (*)
FROM      EMPLOYEE, DEPARTMENT
WHERE      DNO=DNUMBER AND DNAME='Research';
```

Funciones agregadas en SQL

```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM      EMPLOYEE;
```

```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE      Dname='Research';
```



```
SELECT      COUNT (*)
FROM      EMPLOYEE;
```


```
SELECT      COUNT (*)
FROM      EMPLOYEE, DEPARTMENT
WHERE      DNO=DNUMBER AND DNAME='Research';
```


Funciones agregadas en SQL

```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM      EMPLOYEE;
```


```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE      Dname='Research';
```

```
SELECT      COUNT (*)
FROM      EMPLOYEE;
```



```
SELECT      COUNT (*)
FROM      EMPLOYEE, DEPARTMENT
WHERE      DNO=DNUMBER AND DNAME='Research';
```

Funciones agregadas en SQL



SELECT
FROM


COUNT (DISTINCT Salary)
EMPLOYEE;

SELECT
FROM
WHERE

Lname, Fname
EMPLOYEE
(SELECT COUNT (*)
FROM DEPENDENT
WHERE Ssn=Essn) >= 2;

Funciones agregadas en SQL

```
SELECT  
FROM      COUNT (DISTINCT Salary)  
EMPLOYEE;
```



```
SELECT      Lname, Fname  
FROM        EMPLOYEE  
WHERE       ( SELECT      COUNT (*)  
              FROM DEPENDENT  
              WHERE      Ssn=Essn ) >= 2;
```

Funciones agregadas en SQL


- Funciones disponibles
 - COUNT
 - SUM
 - MAX
 - MIN
 - AVG

Agrupamiento

- Para aplicar funciones agregadas a subgrupos de la tabla se utiliza GROUP BY
- Por ejemplo: “Obtener el salario promedio de los empleados de cada departamanto”
- Para resolver esta consulta se necesita particionar la tabla en grupos de tuplas por un atributo arbitrario: el número de departamento
- Este atributo arbitrario se llama atributo de agrupamiento
- Se aplica la función agregada a cada subgrupo

Agrupamiento


- El atributo de agrupamiento debe seleccionarse también para poder identificar el valor agregado y el valor por el que se agrupó



```
SELECT    Dno, COUNT (*), AVG (Salary)
FROM      EMPLOYEE
GROUP BY  Dno;
```

Agrupamiento

- El atributo de agrupamiento debe seleccionarse también para poder identificar el valor agregado y el valor por el que se agrupó




Fname	Minit	Lname	<u>Ssn</u>	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24


Agrupamiento




```
SELECT    Pnumber, Pname, COUNT (*)
FROM      PROJECT, WORKS_ON
WHERE     Pnumber=Pno
GROUP BY  Pnumber, Pname;
```

¿Cómo se
resuelve esta
consulta?

Agrupamiento




```
SELECT Pnumber, Pname, COUNT (*)  
FROM PROJECT, WORKS_ON  
WHERE Pnumber=Pno  
GROUP BY Pnumber, Pname;
```



Primero, se traen
las dos tablas


Agrupamiento



```
SELECT Pnumber, Pname, COUNT (*)  
FROM PROJECT, WORKS_ON  
WHERE Pnumber=Pno  
GROUP BY Pnumber, Pname;
```

Segundo, filtra las
tuplas según la
condición de JOIN

Agrupamiento




```
SELECT Pnumber, Pname, COUNT (*)  
FROM PROJECT, WORKS_ON  
WHERE Pnumber=Pno  
GROUP BY Pnumber, Pname;
```



Por último agrupa las
filas resultantes

Agrupamiento


- La cláusula HAVING permite filtrar a nivel de grupos
- Por ejemplo, solo mostrar los grupos que cumplen cierta condición



```
SELECT      Pnumber, Pname, COUNT (*)  
FROM        PROJECT, WORKS_ON  
WHERE       Pnumber=Pno  
GROUP BY    Pnumber, Pname  
HAVING      COUNT (*) > 2;
```

Agrupamiento

- La cláusula HAVING permite filtrar a nivel de grupos
- Por ejemplo, solo mostrar los grupos que cumplen cierta condición




```
SELECT    Pnumber, Pname, COUNT (*)  
FROM      PROJECT, WORKS_ON  
WHERE     Pnumber=Pno  
GROUP BY  Pnumber, Pname  
HAVING    COUNT (*) > 2;
```



Es diferente del
WHERE

Agrupamiento

- La cláusula HAVING permite filtrar a nivel de grupos
- Por ejemplo, solo mostrar los grupos que cumplen cierta condición




```
SELECT Pnumber, Pname, COUNT (*)  
FROM PROJECT, WORKS_ON  
WHERE Pnumber=Pno  
GROUP BY Pnumber, Pname  
HAVING COUNT (*) >
```

Es diferente del
WHERE

Se puede entender
como un WHERE que se
aplica a nivel de grupos

Agrupamiento


- Hay que tener cuidado a la hora de entender el orden de aplicación del WHERE y del HAVING
- Ejemplo: “recuperar el número total de empleados cuyos salarios exceden \$40000 en cada departamento pero solo para departamento que tengan más de 5 empleados”



```
SELECT      Dname, COUNT (*)
FROM        DEPARTMENT, EMPLOYEE
WHERE       Dnumber=Dno AND Salary>40000
GROUP BY    Dname
HAVING      COUNT (*) > 5;
```

Agrupamiento

- Hay que tener cuidado a la hora de entender el orden de aplicación del WHERE y del HAVING
- Ejemplo: “recuperar el número total de empleados cuyos salarios exceden \$40000 en cada departamento pero solo para departamento que tengan más de 5 empleados”




```
SELECT      Dname, COUNT (*)
FROM        DEPARTMENT, EMPLOYEE
WHERE       Dnumber=Dno AND Salary>40000
GROUP BY    Dname
HAVING      COUNT (*) > 5;
```

¿Da el resultado
esperado?

Agrupamiento

- Hay que tener cuidado a la hora de entender el orden de aplicación del WHERE y del HAVING
- Ejemplo: “recuperar el número total de empleados cuyos salarios exceden \$40000 en cada departamento pero solo para departamento que tengan más de 5 empleados”




```
SELECT  Dname, COUNT (*)  
FROM    DEPARTMENT, EMPLOYEE  
WHERE   Dnumber=Dno AND Salary>40000  
GROUP BY Dname  
HAVING  COUNT (*) > 5;
```

No!

Agrupamiento

- Hay que tener cuidado a la hora de entender el orden de aplicación del WHERE y del HAVING
- Ejemplo: “recuperar el número total de empleados cuyos salarios exceden \$40000 en cada departamento pero solo para departamento que tengan más de 5 empleados”




```
SELECT Dname, COUNT (*)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno AND Salary>40000
GROUP BY Dname
HAVING COUNT (*) > 5;
```

El WHERE se aplica antes de agrupar

Agrupamiento

- Hay que tener cuidado a la hora de entender el orden de aplicación del WHERE y del HAVING
- Ejemplo: “recuperar el número total de empleados cuyos salarios exceden \$40000 en cada departamento pero solo para departamento que tengan más de 5 empleados”




```
SELECT Dname, COUNT(*)
FROM DEPARTMENT, EMPLOYEES
WHERE Dnumber=Dno AND salary>40000
GROUP BY Dname
HAVING COUNT (*) > 5;
```

Cuando se aplique el
HAVING ya los
empleados se habrán
filtrado

Agrupamiento

- Hay que tener cuidado a la hora de entender el orden de aplicación del WHERE y del HAVING
- Ejemplo: “recuperar el número total de empleados cuyos salarios exceden \$40000 en cada departamento pero solo para departamento que tengan más de 5 empleados”



```
SELECT D1.Dnumber, COUNT(*)
FROM DEPARTMENT AS D1, EMPLOYEE AS E1
WHERE D1.Dnumber = E1.Dno AND E1.Salary > 40000
      AND E1.Dno IN (SELECT E2.Dno
                     FROM EMPLOYEE AS E2
                     GROUP BY E2.Dno
                     HAVING COUNT(*) > 2)
GROUP BY D1.DNumber;
```

En resumen...

SELECT <attribute and function list>

FROM <table list>

[**WHERE** <condition>]

[**GROUP BY** <grouping attribute list>]

[**HAVING** <group condition>]

[**ORDER BY** <attribute list>]

Listar los
atributos/funciones que
se quieren obtener

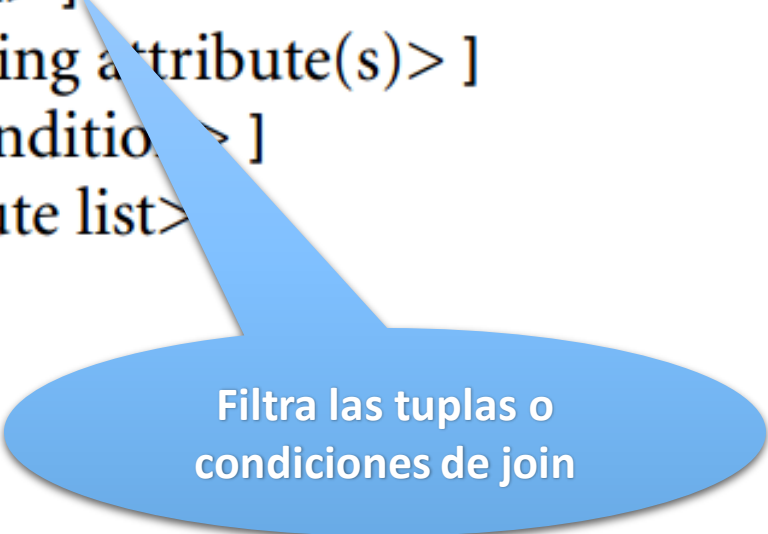
En resumen...

```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```

Relaciones involucradas
incluidos los JOIN, pero
no consultas anidadas

En resumen...

SELECT <attribute and function list>
FROM <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>]



Filtra las tuplas o
condiciones de join

En resumen...

```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```



Agrupar las filas filtradas

En resumen...

```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
ORDER BY <attribute list> ];
```



Filtra los grupos

En resumen...

¿En qué orden se
resuelve la consulta?

```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```


En resumen...

SELECT <attribute and function list>
FROM <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>];

¿En qué orden se
resuelve la consulta?

Depende mucho del
motor


En resumen...



```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```

1. FROM: Identifica todas las tablas involucradas


En resumen...



```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```

2. WHERE: Aplica la condición de selección y join a las tuplas


En resumen...



```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```

3. GROUP BY y HAVING: crea los grupos y filtra

En resumen...



```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```

4. SELECT: Calcula los atributos y/o funciones

En resumen...

```
SELECT <attribute and function list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <grouping attribute(s)> ]  
[ HAVING <group condition> ]  
[ ORDER BY <attribute list> ];
```


5. ORDER BY: Ordena el resultado

En resumen...

- Una consulta **puede formularse de múltiples formas**:
 - El programador puede utilizar la técnica que más conozca o se sienta cómodo **(+)**
 - El programador puede confundirse pues es difícil escoger la técnica para una consulta específica **(-)**
 - Una técnica puede ser más eficiente que otra **(-)**
- Como regla general entre **menos consultas anidadas** se tengan, más eficiente la consulta

Triggers

- Permiten establecer acciones que se ejecutan cuando ciertas condiciones se dan
- Para crear un trigger se usa el comando CREATE TRIGGER



```
CREATE TRIGGER SALARY_VIOLATION  
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN  
ON EMPLOYEE  
  
FOR EACH ROW  
WHEN ( NEW.SALARY > ( SELECT SALARY FROM EMPLOYEE  
WHERE SSN = NEW.SUPERVISOR_SSN ) )  
INFORM_SUPERVISOR(NEW.Supervisor_ssn,  
NEW.Ssn );
```


Vistas en SQL

- Una vista es una tabla derivada de otras tablas
- Una vista puede derivarse de tablas o de otras vistas
- Una vista se considera una tabla virtual (se ve pero no existe)
 - Limita las posibles actualizaciones que se pueden realizar sobre una vista

Vistas en SQL

- ¿Cuándo utilizar una vista?
- Suponga que una aplicación hace una consulta donde hace JOIN de varias tablas, filtra cierto registros y agrupa por ciertas columnas. El resultado, una enorme consulta
- Dicha consulta tiene que ejecutarse constantemente...
- La forma recomendada es construir una vista que encapsule la complejidad detrás de dicha consulta.
 - Optimizada por el DBMS
 - Mejor mantenibilidad

Vistas en SQL




```
CREATE VIEW WORKS_ON1
AS SELECT
    Fname, Lname, Pname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Ssn=Essn AND Pno=Pnumber;
```

```
CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT
    Dname, COUNT (*), SUM (Salary)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno
GROUP BY Dname;
```

Vistas en SQL


```
CREATE VIEW WORKS_ON1
AS SELECT
    Fname, Lname, Pname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Ssn=Essn AND Pno=Pnumber;
```



```
CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT
    Dname, COUNT (*), SUM (Salary)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno
GROUP BY Dname;
```

Vistas en SQL

```
CREATE VIEW WORKS_ON1
AS SELECT
    Fname, Lname, Pname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Ssn=Essn AND Pno=Pnumber;
```



```
CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT
    Dname, COUNT (*), SUM (Salary)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno
GROUP BY Dname;
```



Columnas de la vista

SQL Avanzado

