4.3 SQL

El lenguaje relacional SQL

Structured Query Language (**lenguaje estructurado de consulta**)

- **Primer lenguaje de BD de alto nivel**. Años 70.
  - Diseñado e implementado en el IBM's Research Laboratory (San José - California), para el SGBD Relacional experimental *System R*
- **Definición de un lenguaje estándar para SGBDR**
  - **ANSI** (American National Standards Institute)
    + **ISO** (International Standardization Organization)
  - **SQL1** (ANSI 1986), extendido en 1989 (**SQL-89**)
  - **SQL-92** (SQL2), y
  - **SQL:1999** (extensiones de Orientación a Objetos, disparadores, …)
  - **SQL:2003** (incluye XML y otros conceptos recientes)
- Primeras implementaciones:
  - ORACLE (finales 70) y poco después INGRES

4.3 SQL-92

- Lenguaje de bases de datos completo (no sólo «de consulta»)
  - Definición y Manipulación de Datos (LDD + LMD)
  - Definición y destrucción de Vistas (LDV)
  - Incorporación de SQL dentro de código escrito con un Lenguaje de Programación de propósito general (Pascal, C, etc.)
  - …
- Los proveedores de SGBDR comerciales (Oracle) implementan **variaciones de SQL**
  - Algunas incluyen características que no están estandarizadas
    - índices
    - *triggers* /reglas activas ▸ incluidos en la versión SQL:1999
- Niveles de compatibilidad con el estándar de SQL
  - Entry SQL
  - Intermediate SQL
  - Full SQL

4.3 SQL-92

SQL-92 vs. Modelo Relacional Formal

- **SQL-92**
  - **No utiliza los términos formales** relación, atributo, tupla …, **sino** tabla, columna, fila…
  - **Permite** que las tablas tengan **2 o más filas idénticas** en todos los valores de sus columnas
    - En general, tabla SQL ≠ conjunto de filas, sino que

      Tabla SQL = **Multiconjunto de filas** (saco, *bag*)

    - Es posible forzar que las tablas SQL sean **conjuntos de filas**:
      - con restricciones de clave o
      - mediante opción DISTINCT en una SELECT (*se verá*)
  - Las **columnas** de una tabla están **ordenadas** (orden de creación), y es **posible** indicar un **orden de visualización de** las **filas**
  - **Una clave ajena** (externa) **puede referenciar a una clave candidata (primaria o alternativa)**

4.3 SQL-92

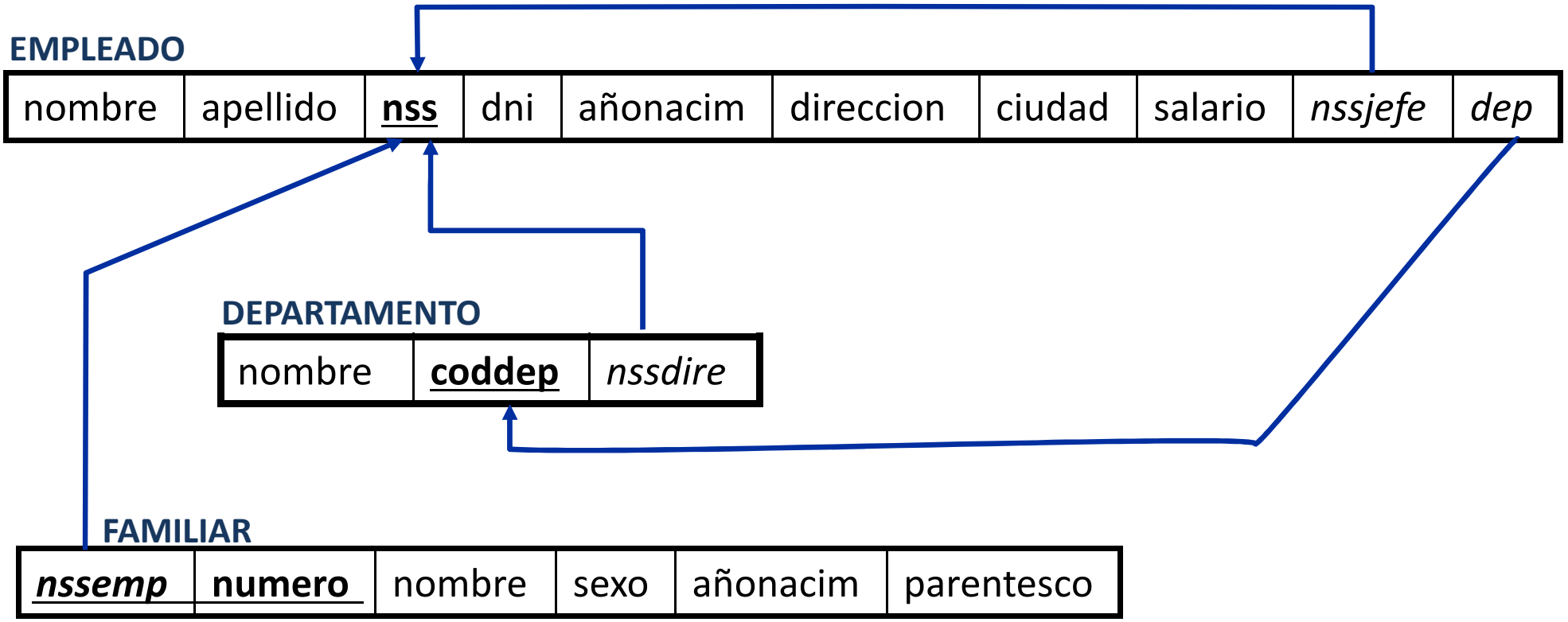
Lo que vamos a estudiar...

- Consultas o Selección de datos ◀ Sesiones de Laboratorio

- Modificación de datos ◀ Clases de Teoría
- Definición, Alteración y Eliminación de elementos de datos
  - Esquemas, Dominios, Tablas
- Vistas
- Restricciones de Integridad Generales (Asertos)

4.3 SQL-92

Esquema de base de datos “Empresa” (simplificado)



4.3 SQL-92

Un Estado del esquema de BD "Empresa" (simplificado)

EMPLEADO

nombre	apellido	**nss**	dni	añonacim	direccion	ciudad	salario	*nssjefe*	*dep*
JONÁS	SOLANO	123	11A	1945	C/PEZ, 10	MURCIA	1100	111	D1
RIGOBERTA	CALAVERA	321	21C	1974	C/BOJ, 2	YECLA	900	333	D3
EUSEBIO	MULETAS	222	22B	1969	C/RIF, 6	TOTANA	2100	123	D2
MACARENO	SOSO	111	23D	1944	C/MAR, 4	JUMILLA	1100		D1
CASIANA	FABERGÉ	333	33B	1943	C/SOL, 8	MURCIA	920	123	D3
FILOMENA	RASCAS	234	34E	1970	C/NUEZ, 3	MURCIA	1100	111	D1
GUMERSINDA	MIMOS	543	45F	1980	C/QUINTO, 5	PINOSO	850		

FAMILIAR

nssemp	**numero**	nombre	sexo	añonacim	parentesco
111	1	JONÁS	H	1992	HIJO
321	2	RÓMULA	M	1923	ABUELO
222	1	ELEUTERIO	H	2002	HIJO
321	1	RENATA	M	2002	HIJO
111	2	JULIANA	M	1936	PADRE

DEPARTAMENTO

nombre	**coddep**	*nssdire*
INVESTIGACION	D2	222
ADMINISTRACION	D1	111
PERSONAL	D3	333
TRAINING	D4	

4.3 SQL-92

Consultas básicas

- **Orden SELECT**:

  Instrucción básica de obtención de información

SELECT <lista columnas>
FROM <lista tablas>
WHERE <condición>

donde:

<lista columnas>	columnas cuyos valores va a obtener la consulta
<lista tablas>	tablas necesarias para realizar la consulta
<condición>	expresión booleana para identificar filas que obtendrá la consulta (expresión de **reunión** y/o de **selección**)

** Año de nacimiento y dirección del empleado llamado Jonás Solano*

```
SELECT añonacim, direccion
FROM EMPLEADO
WHERE apellido='RASCAS';
```

- La consulta **selecciona las filas de <lista tablas>**
  que satisfacen <condición> y
  proyecta el resultado sobre las columnas de **<lista columnas>**
- Resultado: **Tabla** con las con las columnas indicadas y las filas seleccionadas

añonacim	direccion
1970	C/NUEZ, 3

4.3 SQL-92

Consultas básicas: SELECT vs. RESTRICCIÓN

Interesante cuando se haya estudiado Álgebra Relacional

- La orden SELECT ... FROM ... WHERE...
  - **No es igual a** la operación **restricción** σ del *Álgebra Relacional*
  - **SELECT** tiene muchas más opciones y matices
  - En caso de una **única tabla T** en <lista tablas>

```
SELECT <lista columnas>
FROM   T
WHERE <condición>
```

es equivalente a esta expresión del Álgebra Relacional...

$$\prod_{\texttt{<lista columnas>}}(\sigma_{\texttt{<condición>}}\ (T))$$

y también es equivalente a esta expresión del Cálculo Relacional...

$$\{t.\text{columna}_1, \ldots\ t.\text{columna}_n \mid T(t) \text{ and } \texttt{<condición>}\}$$

4.3 SQL-92

Consultas básicas

- Cualquier nº de **condiciones** (selección/reunión) en SELECT

  ** Para cada empleado del departamento D2 o D3, que vivan en Murcia y tengan un salario no superior a 1000€, obtener su dni, nombre y dirección*

```
SELECT dni, nombre, direccion
FROM EMPLEADO
WHERE (dep='D2' OR dep='D3')
  AND ciudad='MURCIA'
  AND NOT (salario>=1000);
```

dni	nombre	direccion
33B	CASIANA	C/SOL, 8

- Una SELECT puede obtener **filas repetidas**
  - No se eliminan las filas duplicadas de forma automática

  ** Salario de los empleados*

```
SELECT salario
FROM EMPLEADO
WHERE dep='D1' OR dep='D2' OR dep='D3';
```

salario
1100
900
2100
1100
920
1100

4.3 SQL-92

Consultas básicas: uso de *

- Obtención de los valores de **todas las columnas** de las filas seleccionadas
  - **No** es **necesario listar todos los nombres** tras la cláusula SELECT
  - Uso del símbolo * (que significa “todas las columnas”)

```
SELECT *
FROM EMPLEADO
WHERE dep='D1';
```

nombre	apellido	**nss**	dni	añonacim	direccion	ciudad	salario	*nssjefe*	*dep*
JONÁS	SOLANO	123	11A	1945	C/PEZ, 10	MURCIA	1100	111	D1
MACARENO	SOSO	111	23D	1944	C/MAR, 4	JUMILLA	1100		D1
FILOMENA	RASCAS	234	34E	1970	C/NUEZ, 3	MURCIA	1100	111	D1

```
SELECT *
FROM DEPARTAMENTO
WHERE nombre='INVESTIGACION';
```

nombre	**coddep**	*nssdire*
INVESTIGACION	D2	222

4.3 SQL-92

Consultas básicas: cadenas de caracteres

- **Operador LIKE**
  - **Comparación** de cadenas de caracteres
  - Caracteres reservados: **'%'** y **'_'** (**comodines**)

** Nombres y apellidos de empleados de Las Torres de Cotillas o Cabezo de Torres*

```
SELECT nombre, apellido
FROM EMPLEADO
WHERE ciudad LIKE '%TORRES%' ;
```

- **Operador ||**
  - **Concatenación** de cadenas de caracteres

** Nombres completos, en una sola columna, de empleados de las ciudades de Aledo, Alguazas, Alhama y Alcantarilla*

```
SELECT nombre || ' ' || apellido
FROM EMPLEADO
WHERE ciudad LIKE 'AL%' ;
```

4.3 SQL-92

Consultas básicas: aritmética y tiempo

- Operaciones **aritméticas**
  - Aplicación de operadores aritméticos (**+**, **-**, *, **/**) sobre valores numéricos

** Salarios de los empleados del departamento D3, tras un aumento del 10%*

```
SELECT apellido, nombre, 1.1*salario
FROM EMPLEADO
WHERE dep='D3';
```

apellido	nombre	1.1*salario
CALAVERA	RIGOBERTA	990
FABERGÉ	CASIANA	1012

👁 el valor real actual de los **salarios** en la tabla EMPLEADO **no** cambia

- Operaciones con **fechas**, horas, marcas de tiempo e intervalos
  - Especificación del **valor de un INTERVAL** **como diferencia de dos** valores **DATE**, **TIME** o **TIMESTAMP**
  - Incremento y Decremento de valores de columnas de tipo **DATE, TIME, TIMESTAMP** en un **intervalo** compatible con el tipo

4.3 SQL-92

Consultas básicas: reunión (join)

- **Combina** las **filas relacionadas** de dos tablas **en una sola fila**
- Permite procesar los **vínculos entre tablas**
  - Datos del **empleado** 'SOLANO' junto con los de **su departamento**
    - Es necesario combinar la fila del EMPLEADO correspondiente a 'SOLANO', e, con la fila de DEPARTAMENTO cuyo valor en coddep coincida con el de dep en e
    - Se consigue aplicando la operación REUNIÓN a las dos tablas
  - Datos de cada **empleado** junto con los de **sus familiares**
    - Es necesario combinar cada fila de EMPLEADO, e, con cada fila de FAMILIAR, f, tal que el valor de nssemp en f coincida con el de nss en e
    - Se consigue aplicando la operación REUNIÓN a las dos tablas

4.3 SQL-92

Consultas básicas: reunión (join)

EMPLEADO

nombre	apellido	nss	...	dep
JONÁS	SOLANO	123	...	D1
RIGOBERTA	CALAVERA	321	...	D3
EUSEBIO	MULETAS	222	...	D2
MACARENO	SOSO	111	...	D1
CASIANA	FABERGÉ	333	...	D3
FILOMENA	RASCAS	234	...	D1
GUMERSINDA	MIMOS	543	...	

DEPARTAMENTO

nombre	coddep	nssdire
INVESTIGACION	D2	222
ADMINISTRACION	D1	111
PERSONAL	D3	333
TRAINING	D4	

```
SELECT * FROM EMPLEADO, DEPARTAMENTO WHERE dep=coddep
```

nombre	apellido	nss	...	dep	nombre	coddep	nssdire
JONÁS	SOLANO	123	...	D1	ADMINISTRACION	D1	111
RIGOBERTA	CALAVERA	321	...	D3	PERSONAL	D3	333
EUSEBIO	MULETAS	222	...	D2	INVESTIGACION	D2	222
MACARENO	SOSO	111	...	D1	ADMINISTRACION	D1	111
CASIANA	FABERGÉ	333	...	D3	PERSONAL	D3	333
FILOMENA	RASCAS	234	...	D1	ADMINISTRACION	D1	111

💣Importante: En el resultado NO ESTÁ la empleada GUMERSINDA MIMOS , ni el departamento TRAINING. ¿**Por qué**?

4.3 SQL-92

Consultas básicas: reunión (join)

EMPLEADO

nombre	apellido	nss	...	dep
JONÁS	SOLANO	123	…	D1
RIGOBERTA	CALAVERA	321	…	D3
EUSEBIO	MULETAS	222	…	D2
MACARENO	SOSO	111	…	D1
CASIANA	FABERGÉ	333	…	D3
FILOMENA	RASCAS	234	…	D1
GUMERSINDA	MIMOS	543	…	

FAMILIAR

nssemp	nombre	sexo	añonacim	parentesco
111	JONÁS	H	1992	HIJO
321	RÓMULA	M	1923	ABUELO
222	ELEUTERIO	H	2002	HIJO
321	RENATA	M	2002	HIJO
111	JULIANA	M	1936	PADRE

SELECT * FROM EMPLEADO, FAMILIAR WHERE nss=nssemp

nombre	apellido	nss	...	dep	nssemp	nombre	sexo	añonacim	parentesco
MACARENO	SOSO	111	…	D1	111	JONÁS	H	1992	HIJO
RIGOBERTA	CALAVERA	321	…	D3	321	RÓMULA	M	1923	ABUELO
EUSEBIO	MULETAS	222	…	D2	222	ELEUTERIO	H	2002	HIJO
RIGOBERTA	CALAVERA	321	…	D3	321	RENATA	M	2002	HIJO
MACARENO	SOSO	111	…	D1	111	JULIANA	M	1936	PADRE

💣 Importante: En el resultado FALTAN varios empleados ¿**por qué**?
¿Y faltan familiares? ¿por qué?

4.3 SQL-92

Consultas básicas: reunión (join)

- Para especificar una **reunión** (join) entre varias tablas, estas se incluyen en la cláusula FROM separadas por comas, y
- Las **condiciones de reunión** se incluyen en la cláusula WHERE, quizá mezcladas con las condiciones de selección de filas

```
SELECT <lista columnas>
FROM   R, S
WHERE <condición de reunión>
```

** Apellido, dirección y departamento de los empleados que tengan algún hijo*

```
SELECT apellido, direccion, dep
FROM EMPLEADO, FAMILIAR      → reunión o join de tablas
WHERE nss=nssemp             → condición de reunión entre tablas ◄
  AND parentesco='HIJO'      → condición de selección;
```

4.3 SQL-92

Consultas básicas: omisión de WHERE

- Selección incondicional
  - No incluir WHERE equivale a una condición TRUE para todas las filas
    - Selección de **todas las filas** de...
      - una **tabla** (si la cláusula FROM sólo contiene una tabla), o
      - el **producto cartesiano** entre varias tablas (si FROM incluye más de una)

** Seleccionar los nss de todos los empleados*

```
SELECT nss
FROM EMPLEADO;
```

** Obtener todas las **combinaciones** de los datos de empleados con los de departamentos*

```
SELECT *
FROM EMPLEADO, DEPARTAMENTO;
```

4.3 SQL-92

Consultas básicas: omisión de WHERE en la reunión

nombre	apellido	nss	dni	añonacim	direccion	ciudad	salario	nssjefe	dep	nombre	coddep	nssdire
JONÁS	SOLANO	123	11A	1945	C/PEZ, 10	MURCIA	1100	111	D1	ADMINISTRACION	D1	111
RIGOBERTA	CALAVERA	321	21C	1974	C/BOJ, 2	YECLA	900	333	D3	ADMINISTRACION	D1	111
EUSEBIO	MULETAS	222	22B	1969	C/RIF, 6	TOTANA	2100	123	D2	ADMINISTRACION	D1	111
MACARENO	SOSO	111	23D	1944	C/MAR, 4	JUMILLA	1100		D1	ADMINISTRACION	D1	111
CASIANA	FABERGÉ	333	33B	1943	C/SOL, 8	MURCIA	920	123	D3	ADMINISTRACION	D1	111
FILOMENA	RASCAS	234	34E	1970	C/NUEZ, 3	MURCIA	1100	111	D1	ADMINISTRACION	D1	111
JONÁS	SOLANO	123	11A	1945	C/PEZ, 10	MURCIA	1100	111	D1	INVESTIGACION	D2	222
RIGOBERTA	CALAVERA	321	21C	1974	C/BOJ, 2	YECLA	900	333	D3	INVESTIGACION	D2	222
EUSEBIO	MULETAS	222	22B	1969	C/RIF, 6	TOTANA	2100	123	D2	INVESTIGACION	D2	222
MACARENO	SOSO	111	23D	1944	C/MAR, 4	JUMILLA	1100		D1	INVESTIGACION	D2	222
CASIANA	FABERGÉ	333	33B	1943	C/SOL, 8	MURCIA	920	123	D3	INVESTIGACION	D2	222
FILOMENA	RASCAS	234	34E	1970	C/NUEZ, 3	MURCIA	1100	111	D1	INVESTIGACION	D2	222
JONÁS	SOLANO	123	11A	1945	C/PEZ, 10	MURCIA	1100	111	D1	PERSONAL	D3	333
RIGOBERTA	CALAVERA	321	21C	1974	C/BOJ, 2	YECLA	900	333	D3	PERSONAL	D3	333
EUSEBIO	MULETAS	222	22B	1969	C/RIF, 6	TOTANA	2100	123	D2	PERSONAL	D3	333
MACARENO	SOSO	111	23D	1944	C/MAR, 4	JUMILLA	1100		D1	PERSONAL	D3	333
CASIANA	FABERGÉ	333	33B	1943	C/SOL, 8	MURCIA	920	123	D3	PERSONAL	D3	333
FILOMENA	RASCAS	234	34E	1970	C/NUEZ, 3	MURCIA	1100	111	D1	PERSONAL	D3	333

4.3 SQL-92

Consultas básicas: calificación

- En SQL los **nombres** de las **columnas** deben ser **únicos dentro de** cada **tabla**, pero distintas tablas pueden tener columnas denominadas igual.
- Consulta que referencia a varias **columnas de igual nombre, pero de tablas distintas... ▸ AMBIGÜEDAD**
  - **Solución: CALIFICACIÓN**

** Nombre, apellido y dirección de los empleados del departamento de INVESTIGACION*

```
SELECT nombre, apellido, direccion
FROM EMPLEADO, DEPARTAMENTO
WHERE dep=coddep
   AND nombre='INVESTIGACION'; -- Error: columna ambigua
-- Sentencia correcta:
SELECT EMPLEADO.nombre, apellido, direccion
FROM EMPLEADO, DEPARTAMENTO
WHERE dep=coddep
   AND DEPARTAMENTO.nombre='INVESTIGACION';
```

4.3 SQL-92

Consultas básicas: pseudónimos

- Puede utilizarse **pseudónimos** (alias) para **acortar** nombres de tabla dentro de las **consultas con calificación**:

```
SELECT E.nombre, apellido, direccion
FROM EMPLEADO E, DEPARTAMENTO AS D    ← AS es opcional
WHERE dep=coddep
   AND D.nombre='INVESTIGACION';
```

  - En SQL de Oracle **no** se permite usar AS, basta con separar el nombre de la tabla y el del alias con un espacio en blanco

- Consulta que se refiere **dos veces** a la **misma tabla**
  - **AMBIGÜEDAD**
    - **Solución: PSEUDÓNIMOS**

** Obtener nombre y apellido de cada empleado y de su supervisor inmediato*

```
SELECT E.nombre, E.apellido, S.nombre, S.apellido
FROM EMPLEADO E, EMPLEADO S
WHERE E.nssjefe=S.nss;
```

4.3 SQL-92

Consultas básicas: renombrar columnas

- En el **resultado** de evaluar la consulta

  ** Nombres de cada empleado y su supervisor, cambiando al mismo tiempo los nombres de las columnas resultantes a 'empleado' y 'supervisor'*

```
SELECT E.nombre AS empleado, S.nombre AS supervisor
FROM EMPLEADO E, EMPLEADO S
WHERE E.nssjefe = S.nss;
```

Sin los alias:

E.nombre	S.nombre
JONÁS	MACARENO
RIGOBERTA	CASIANA
EUSEBIO	JONÁS
CASIANA	JONÁS
FILOMENA	MACARENO

Con los alias:

empleado	supervisor
JONÁS	MACARENO
RIGOBERTA	CASIANA
EUSEBIO	JONÁS
CASIANA	JONÁS
FILOMENA	MACARENO

- **Nueva cabecera** para la tabla resultado
  - En SQL de Oracle basta con separar el nombre de la columna y el del alias con un espacio en blanco, aunque también se puede usar el AS

4.3 SQL-92

Consultas básicas: orden de presentación

- SQL permite **presentar las filas** resultado de una consulta de forma **ordenada**: **Cláusula ORDER BY**
  - Ordenación **según valores de** una o varias **columnas**
  - **Ascendente ASC** (por defecto) o **Descendente DESC**
  - Suele ser una operación muy costosa

ⓘ las filas no se ordenan en disco: **se verán ordenadas**, pero no tienen por qué estarlo

```
SELECT coddep,nombre
FROM DEPARTAMENTO
ORDER BY nombre;
```

```
SELECT apellido,nombre,añonacim
FROM EMPLEADO
WHERE ciudad='MURCIA'
ORDER BY apellido, nombre;
```

```
SELECT nssemp,nombre,añonacim,parentesco
FROM FAMILIAR
ORDER BY nssemp, añonacim;
```

4.3 SQL-92

Tablas como conjuntos

- SQL **no elimina filas repetidas** del resultado de una consulta, porque...
  - Eliminar duplicados es **costoso** (ordenar+recorrer+eliminar)
  - El usuario puede desear ver las filas repetidas en el resultado
  - Si se aplica una función agregada a filas, rara vez deben eliminarse las duplicadas
- **Operador DISTINCT**:
  - Para eliminar **filas repetidas** del resultado de una consulta SQL
    - Resultado = Relación del Modelo Relacional Formal (conjunto de filas)

** Salarios de los empleados*

```
SELECT salario FROM EMPLEADO;
```

** Salarios **distintos** de los empleados*

```
SELECT DISTINCT salario
  FROM EMPLEADO;
```

salario
1100
900
2100
1100
920
1100
850

salario
1100
900
2100
920
850

4.3 SQL-92

Tablas como conjuntos

- **Operaciones de conjuntos: UNION, INTERSECT, EXCEPT**
  - Resultado: conjunto de filas ▸ **las filas repetidas se eliminan**
  - Las tablas operando han de ser **compatibles en tipo:**
    - igual **nº de columnas**, y
    - columnas "correspondientes" con igual **dominio** (o de tipos de datos compatibles)

(❑ En SQL de Oracle se usa **MINUS** en lugar de EXCEPT)

** Empleados veteranos (nacidos antes de 1965), o bien con familiares mayores a su cargo*

```
  (SELECT nss FROM EMPLEADO
   WHERE añonacim<1965)
UNION
  (SELECT nssemp FROM FAMILIAR
   WHERE parentesco='ABUELO');
```

nss
123
111
333

UNION

nssemp
321

=

nss
111
123
321
333

4.3 SQL-92

Tablas como conjuntos

** Empleados jefes de otros y que también dirigen algún departamento*

```
  (SELECT nssjefe
   FROM EMPLEADO)
INTERSECT
  (SELECT nssdire
   FROM DEPARTAMENTO);
```

nssjefe
111
333
123
123
111

INTERSECT

nssdire
222
111
333

=

nssjefe
111
333

** Empleados que cobran menos de 950€ y que no tienen hijos*

```
  (SELECT nss
   FROM EMPLEADO
   WHERE salario<950)
EXCEPT --MINUS
  (SELECT nssemp
   FROM FAMILIAR
   WHERE parentesco ='HIJO');
```

nss
321
333
543

EXCEPT

nssemp
111
222
321

=

nss
333
543

- Para **no eliminar duplicados**…
  - UNION **ALL**, INTERSECT **ALL**, EXCEPT **ALL** (❑En SQL de Oracle es MINUS ALL)

4.3 SQL-92

Tablas como conjuntos: conjuntos explícitos

- **Un conjunto explícito de valores** es una lista de valores, separados por comas, encerrada entre paréntesis
- Puede aparecer en la cláusula WHERE

  ** Nss de los empleados que tienen padres/madres, abuelos/as o tíos a su cargo*

```
SELECT DISTINCT nssemp FROM FAMILIAR
WHERE parentesco IN ('PADRE', 'ABUELO','TIO');
```

- **Operador IN**

  v IN V

  – Indica si el **valor v** pertenece al **conjunto de valores V**

  – Devuelve TRUE si **algún** elemento **e** de **V** cumple que **v = e**

  ** Nss de los empleados que tengan a su cargo familares que no sean hijos ni sobrinos*

```
SELECT DISTINCT nssemp FROM FAMILIAR
WHERE parentesco NOT IN ('HIJO', 'SOBRINO');
```

4.3 SQL-92

Tablas como conjuntos: conjuntos explícitos

- **Operador ANY** (o **SOME**)

  v <op> ANY V o **v <op> SOME V** ,, <op> ∈ { >, ≥, <, ≤, **< >**, = }

  – Compara un valor individual **v** con los elementos de un conjunto **V**

  – Devuelve TRUE si **algún** elemento **e** de **V** cumple que **v <op> e**

  ** Nss de los empleados que tienen a su cargo algún padre, abuelo o tío*

```
SELECT DISTINCT nssemp FROM FAMILIAR
WHERE parentesco = ANY ('PADRE','ABUELO','TIO');
```

- **Operador ALL**

  v <op> ALL V,, <op> ∈ { >, ≥, <, ≤, **<>**, = }

  – Compara un valor **v** con los elementos de un conjunto **V**

  – Devuelve TRUE si **para todo** elemento **e** de **V** se cumple **v <op> e**

  ** Obtener el nss de los empleados que no tienen hijos ni sobrinos a su cargo*

```
SELECT DISTINCT nssemp FROM FAMILIAR
WHERE parentesco <> ALL ('HIJO', 'SOBRINO');
```

4.3 SQL-92

Consultas anidadas

- **Es una consulta** SELECT **completa, dentro de cláusula** WHERE **de otra consulta** (consulta exterior)
- Obtiene **valores de la BD que se usan en la condición de otra consulta**, para obtener otros datos

** Familiares del empleado 'RIGOBERTA CALAVERA'*

```
SELECT nombre FROM FAMILIAR
WHERE nssemp IN (SELECT nss FROM Empleado
                 WHERE nombre='RIGOBERTA'
                   AND apellido ='CALAVERA');
```

** Empleados (nss y nombre) del departamento de Investigación*

```
SELECT nss, nombre FROM EMPLEADO
WHERE dep IN (SELECT coddep FROM DEPARTAMENTO
              WHERE nombre = 'INVESTIGACION');
```

4.3 SQL-92

Consultas anidadas

** Empleados (nss y nombre) del departamento de Investigación*

```
SELECT nss, nombre
FROM EMPLEADO
WHERE dep IN (SELECT coddep FROM DEPARTAMENTO
              WHERE nombre = 'INVESTIGACION');
```

Primero se evalúa la subconsulta

coddep
D2

Después se seleccionan las filas de EMPLEADO cuyo valor de columna dep está entre los obtenidos en la subconsulta

nombre	apellido	**nss**	...	*nssjefe*	*dep*
JONÁS	SOLANO	123		111	D1
RIGOBERTA	CALAVERA	321		333	D3
EUSEBIO	MULETAS	222		123	D2
MACARENO	SOSO	111			D1
CASIANA	FABERGÉ	333		123	D3
FILOMENA	RASCAS	234		111	D1
GUMERSINDA	MIMOS	543			

Por último, se proyecta en las columnas indicadas

nss	nombre
222	EUSEBIO

4.3 SQL-92

Consultas anidadas

- Es posible tener **varios niveles de consultas anidadas**

** Empleados (dni y nombre) del departamento dirigido por 'MACARENO SOSO'.*

```
SELECT dni, nombre
FROM EMPLEADO
WHERE dep IN (SELECT coddep
              FROM DEPARTAMENTO
              WHERE nssdire IN (SELECT nss
                                FROM EMPLEADO
                                WHERE nombre='MACARENO'
                                  AND apellido='SOSO'));
```

4.3 SQL-92

Consultas anidadas: comparar conjuntos

- **Operador IN** (otro uso del mismo operador)

  t IN S

  – Indica **si la fila t pertenece al conjunto de filas S** (subconsulta)

** Nombre y dirección de los empleados que tienen algún familiar.*

```
SELECT nombre, direccion FROM EMPLEADO
WHERE nss IN (SELECT nssemp FROM FAMILIAR);
```

** Dni, nombre y ciudad de los empleados que son jefes de departamento.*

```
SELECT dni, nombre, ciudad FROM EMPLEADO
WHERE nss IN (SELECT nssdire FROM DEPARTAMENTO);
```

** Nombre y año de nacimiento de los familiares de los empleados del departamento 'D1'.*

```
SELECT nombre, añonacim FROM FAMILIAR
WHERE nssemp IN (SELECT nss FROM EMPLEADO WHERE dep='D1');
```

4.3 SQL-92

Consultas anidadas

- El operador **IN** en realidad compara **filas (tuplas)**

** NSS de empleados que tienen la misma edad y ciudad que algún otro empleado.*

```
SELECT nss
FROM EMPLEADO E1
WHERE (añonacim, ciudad) IN (SELECT añonacim, ciudad
                             FROM EMPLEADO E2
                             WHERE E1.nss <> E2.nss);
```

**Familiares con igual nombre que el empleado del que dependen*

```
SELECT *
FROM FAMILIAR
WHERE (nombre, nssemp) IN (SELECT nombre, nss
                           FROM EMPLEADO);
```

4.3 SQL-92

Consultas anidadas

- **Si** la consulta anidada **devuelve una sola columna y una única fila,** el resultado es un valor escalar, y **se permite usar el comparador = en lugar del operador IN**

** Empleados (nss y nombre) del departamento de Investigación*

```
SELECT nss, nombre
FROM EMPLEADO
WHERE dep = (SELECT coddep FROM DEPARTAMENTO
             WHERE nombre = 'INVESTIGACION');
```

** Familiares del empleado 'RIGOBERTA CALAVERA'*

```
SELECT nombre FROM FAMILIAR
WHERE nssemp = (SELECT nss
                FROM Empleado
                WHERE nombre = 'RIGOBERTA'
                  AND apellido ='CALAVERA');
```

4.3 SQL-92

Consultas anidadas: comparar conjuntos

- **Operador ANY** o **SOME** (otro uso del mismo operador)

  t <op> ANY S o **t <op> SOME S,,** <op> $\in \{ >, \geq, <, \leq, < >, = \}$

  – Compara una fila **t** con las filas resultado de una consulta anidada **S**

  – Devuelve TRUE si **alguna** fila **e** de **S** cumple que **t <op> e**

- **Operador ALL** (otro uso del mismo operador)

  t <op> ALL S,, <op> $\in \{ >, \geq, <, \leq, < >, = \}$

  – Compara una fila **t** con filas resultado de una consulta anidada **S**

  – Devuelve TRUE si **para toda** fila **e** de **S** se cumple que **t <op> e**

** Nombres y apellidos de los empleados cuyo salario es menor que el de <u>todos</u> los empleados del departamento D3*

```
SELECT nombre, apellido FROM EMPLEADO
WHERE salario < ALL (SELECT salario
                     FROM EMPLEADO
                     WHERE dep='D3' );
```

❍ ¿Sería "mejor" usar DISTINCT en la subconsulta?

4.3 SQL-92

Consultas anidadas

** Nombres y apellidos de los empleados cuyo salario es menor que el de <u>todos</u> los empleados del departamento D3*

```
SELECT nombre, apellido FROM EMPLEADO
WHERE salario < ALL (SELECT salario FROM EMPLEADO WHERE dep='D3');
```

Primero se evalúa la subconsulta

salario
900
920

Después se seleccionan las filas de EMPLEADO cuyo valor de columna `salario` es inferior a todos los valores obtenidos en la subconsulta

nombre	apellido	**nss**	…	salario	*nssjefe*	*dep*
JONÁS	SOLANO	123		1100	111	D1
RIGOBERTA	CALAVERA	321		900	333	D3
EUSEBIO	MULETAS	222		2100	123	D2
MACARENO	SOSO	111		1100		D1
CASIANA	FABERGÉ	333		920	123	D3
FILOMENA	RASCAS	234		1100	111	D1
GUMERSINDA	MIMOS	543		**850**		

Por último, se proyecta en las columnas indicadas

nombre	apellido
GUMERSINDA	MIMOS

4.3 SQL-92

Consultas anidadas: columnas ambiguas

- **Coincidencia de nombres** de columnas en las consultas exterior y anidada ▸ **Ambigüedad**

  ** Nombre y apellidos de cada empleado con familiares de igual nombre que él*

```
SELECT nombre, apellido FROM EMPLEADO
WHERE nss IN (SELECT nssemp FROM FAMILIAR
                    WHERE nombre=nombre);
```

◂ ¿cómo evitar esta ambigüedad?

- ❑ Regla: **una columna no calificada se refiere a la tabla declarada en la consulta anidada más interior**
- ❑ Si en una consulta **anidada** es necesario **usar** columnas de **tablas** declaradas en una consulta **exterior** ⇨ **calificar**

  ** Nombre y apellidos de cada empleado con familiares de igual nombre que él*

```
SELECT nombre, apellido
FROM EMPLEADO E
WHERE nss IN (SELECT nssemp FROM FAMILIAR
                  WHERE nombre=E.nombre);
```

4.3 SQL-92

Consultas anidadas: correlación

- Una consulta exterior y otra anidada están **correlacionadas** si **una condición de la anidada contiene columnas de una tabla declarada en la consulta exterior**

** Nombre y apellido de los empleados que tienen algún abuelo con igual nombre que él*

```
SELECT nombre, apellido
FROM EMPLEADO E
WHERE nss IN (SELECT nssemp FROM FAMILIAR
              WHERE E.nombre=nombre AND parentesco='ABUELO');
```

** Código de los departamentos cuyo director pertenece a un departamento distinto al que dirige ◗ detección de errores*

```
SELECT coddep
FROM DEPARTAMENTO
WHERE nssdire IN (SELECT nss FROM EMPLEADO
                  WHERE dep<>coddep);
```

4.3 SQL-92

Consultas anidadas: correlación

- **La consulta anidada correlacionada se evalúa una vez para cada fila** (o combinación de filas) **de la consulta exterior** ☹
  - Trate de ejecutar "a mano", utilizando la tabla de ejemplo, la consulta de la diapositiva anterior que obtiene **Nombre y apellido de los empleados que tienen algún abuelo con igual nombre que él*
  - Esta circunstancia suele implicar que este tipo de consultas correlacionadas sean poco eficientes… y sea deseable redactarlas de otro modo.
- Una **consulta anidada que use el operador =** o **IN siempre puede expresarse** como una **reunión** (JOIN)

```
SELECT E.nombre, apellido
FROM EMPLEADO E, FAMILIAR F
WHERE nss=nssemp
  AND E.nombre=F.nombre
  AND parentesco='ABUELO';
```

4.3 SQL-92

Consultas anidadas: EXISTS

- **Operador EXISTS (S)**: comprobación de tablas vacías
  - Devuelve TRUE si la tabla S contiene al menos una fila
  - Devuelve FALSE si S es una tabla vacía (sin filas)

ⓘ S suele ser una consulta anidada correlacionada

** Nombre y apellido de cada empleado con familiares de igual nombre que él*

```
SELECT nombre, apellido FROM EMPLEADO E
WHERE EXISTS (SELECT * FROM FAMILIAR
              WHERE nssemp=nss AND nombre=E.nombre);
```

- Suele usarse más en sentido negativo: NOT EXISTS (S)

** Nombres y apellidos de los empleados sin familiares*

```
SELECT nombre, apellido
FROM EMPLEADO
WHERE NOT EXISTS (SELECT * FROM FAMILIAR
                  WHERE nssemp=nss);
```

4.3 SQL-92

Nulos

- **Null**
  - Es una marca que indica **ausencia** o **desconocimiento** de información
  - Comparar (>, ≥, <, ≤, < >, =) NULL con cualquier cosa da FALSE
    - NULL no es un valor, y es **distinto a cualquier otra cosa**, incluso a otro NULL

- **Operador IS NULL , IS NOT NULL**

  v IS NULL

  – Devuelve TRUE si **v** es NULL

  v IS NOT NULL

  – Devuelve TRUE si **v** es un valor no NULL

** Nombres y apellidos de los empleados sin supervisores*

```
SELECT nombre, apellido FROM EMPLEADO
WHERE nssjefe IS NULL;
```

nombre	apellido
MACARENO	SOSO
GUMERSINDA	MIMOS

4.3 SQL-92

Funciones agregadas

- **Funciones SUM( ), MAX( ), MIN( ), AVG( )**
  - Suma, máximo, mínimo y media aritmética (promedio)
  - Aplicadas a un multiconjunto (saco, *bag*) de valores numéricos

ⓘ Pueden aparecer en cláusula SELECT

** Suma de los salarios y salario máximo, mínimo y medio de los empleados*

```
SELECT SUM(salario),MAX(salario),MIN(salario),AVG(salario)
FROM EMPLEADO;
```

SUM(salario)	MAX(salario)	**MIN(salario)**	**AVG(salario)**
8070	2100	850	1152,86

** Suma de salarios y salario máximo, mínimo y medio de empleados del dep. Investigación*

```
SELECT SUM(salario),MAX(salario),MIN(salario),AVG(salario)
FROM EMPLEADO
WHERE dep IN (SELECT coddep FROM DEPARTAMENTO
              WHERE nombre='INVESTIGACION');
```

ⓘ También pueden aparecer en cláusula **HAVING** (*se verá*)

4.3 SQL-92

Funciones agregadas

- **Función COUNT( )**
  - Cuenta el número de **filas**, o de **valores no nulos** en atributos

** ¿Cuántos empleados hay?*

```
SELECT COUNT(nss) FROM EMPLEADO;
```

COUNT(nss)
7

** ¿Cuántos familiares tiene la empleada 'RIGOBERTA CALAVERA'?*

```
SELECT COUNT(nombre) FROM FAMILIAR
WHERE nssemp IN (SELECT nss FROM EMPLEADO
                 WHERE nombre='RIGOBERTA' AND apellido='CALAVERA');
```

COUNT(nombre)
2

- También pueden aparecer en cláusula **HAVING** (*se verá*)

4.3 SQL-92

Funciones agregadas: uso de * y de DISTINCT

- **Uso de ***

** Número total de empleados de la empresa*

```
SELECT COUNT(*) FROM EMPLEADO
```
◂ **cuenta filas**

COUNT(*)
7

** Contar el número de empleados de la empresa **que tienen un jefe***

```
SELECT COUNT(nssjefe)
  FROM Empleado;
```

COUNT(nssjefe)
5

◂ **cuenta filas con nssjefe NOT NULL**

** Número de empleados en el departamento de Investigación*

```
SELECT COUNT(*) FROM EMPLEADO, DEPARTAMENTO D
WHERE dep=coddep AND D.nombre='INVESTIGACION';
```

COUNT(*)
1

- **Uso de DISTINCT**

** Contar el nº de **valores distintos** de salario que pueden cobrar los empleados*

```
SELECT COUNT(salario) FROM EMPLEADO;
```

◂ **Error**: NO se eliminan duplicados, así que COUNT(salario) ≡ COUNT(*)

```
SELECT COUNT(DISTINCT salario) FROM EMPLEADO;
```
◂ **OK !!**

4.3 SQL-92

Funciones agregadas: uso de * y de DISTINCT

- Ejemplo

  ** Cuántos empleados son jefes de algún otro*

```
-- Con correlación:
SELECT COUNT(*)
FROM EMPLEADO J
WHERE EXISTS (SELECT * FROM EMPLEADO E
              WHERE E.nssjefe=J.nss);
-- Sin correlación:
SELECT COUNT(*)
FROM EMPLEADO
WHERE nss IN (SELECT nssjefe FROM EMPLEADO);


-- Sin subconsulta (la mejor opción):
SELECT COUNT(DISTINCT nssjefe)
FROM EMPLEADO;
```

4.3 SQL-92

Funciones agregadas y correlación

- Es posible que una consulta anidada y correlacionada con otra exterior, incluya una función agregada

  ** Apellidos y nombres de los empleados con 2 o más familiares*

```
SELECT apellido, nombre
FROM EMPLEADO
WHERE 2 ≤ (SELECT COUNT(*)
           FROM FAMILIAR
           WHERE nss=nssemp);
```

4.3 SQL-92

Agrupación

- **Cláusula GROUP BY**
  - Para formar **grupos de filas** dentro de una tabla
  - Los grupos se forman según el valor de las columnas de agrupación
  - Las **filas de cada grupo** tendrán el **mismo valor en las columnas de agrupación**
- Aplicación de **funciones agregadas a grupos de filas**

** Para cada departamento, obtener su número, cuántos empleados tiene dicho departamento y el salario medio de los empleados del mismo*

```
SELECT dep, COUNT(*), AVG(salario)
FROM EMPLEADO
GROUP BY dep ;
```

← una columna de agrupación

En la cláusula SELECT, las **columnas de agrupación deben aparecer antes de cualquier función agregada**, para que su valor (único para cada grupo) aparezca junto al resultado de aplicar la función al grupo

4.3 SQL-92

Agrupación

** Para cada departamento, obtener su número, cuántos empleados tiene dicho departamento y el salario medio de los empleados del mismo*

```
SELECT dep, COUNT(*), AVG(salario)
FROM EMPLEADO
GROUP BY dep ;
```

Primero se construyen los grupos en la tabla EMPLEADO

nombre	apellido	salario	...	dep
JONÁS	SOLANO	1100	...	D1
MACARENO	SOSO	1100	...	D1
FILOMENA	RASCAS	1100	...	D1
EUSEBIO	MULETAS	2100	...	D2
RIGOBERTA	CALAVERA	900	...	D3
CASIANA	FABERGÉ	920	...	D3
GUMERSINDA	MIMOS	850	...	

Después se aplica la cláusula SELECT a cada grupo

dep	COUNT(*)	AVG(*)
D1	3	1100
D2	1	2100
D3	2	910
NULL	1	850

4.3 SQL-92

Agrupación

- **Cláusula HAVING**
  - **Siempre junto a GROUP BY**
  - **Condición** que **deben cumplir** los **grupos de filas** asociados a cada valor de las columnas de agrupación
  - Un grupo que no cumple la condición, no es seleccionado para el resultado

** Para cada departamento en el que **el salario medio de los empleados sea inferior a 1200 euros**, mostrar el nombre del departamento y el código de su director*

```
SELECT D.nombre, nssdire
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep=dep
GROUP BY dep
HAVING AVG(salario)<1200 ;
```

4.3 SQL-92

Agrupación

```
SELECT D.nombre, nssdire
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep=dep
GROUP BY dep
HAVING AVG(salario)<1200 ;
```

Primero se ejecuta el **join**

nombre	apellido	nss	...	dep	nombre	coddep	nssdire
JONÁS	SOLANO	123	…	D1	ADMINISTRACION	D1	111
RIGOBERTA	CALAVERA	321	…	D3	PERSONAL	D3	333
EUSEBIO	MULETAS	222	…	D2	INVESTIGACION	D2	222
MACARENO	SOSO	111	…	D1	ADMINISTRACION	D1	111
CASIANA	FABERGÉ	333	…	D3	PERSONAL	D3	333
FILOMENA	RASCAS	234	…	D1	ADMINISTRACION	D1	111

4.3 SQL-92

Agrupación

```
SELECT D.nombre, nssdire
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep=dep
GROUP BY dep
HAVING AVG(salario)<1000 ;
```

Después se **particiona en grupos** según el valor de la columna dep

nombre	apellido	salario	...	dep	nombre	coddep	nssdire
JONÁS	SOLANO	1100	…	D1	ADMINISTRACION	D1	111
MACARENO	SOSO	1100	…	D1	ADMINISTRACION	D1	111
FILOMENA	RASCAS	1100	…	D1	ADMINISTRACION	D1	111
EUSEBIO	MULETAS	2100	…	D2	INVESTIGACION	D2	222
RIGOBERTA	CALAVERA	900	…	D3	PERSONAL	D3	333
CASIANA	FABERGÉ	920	…	D3	PERSONAL	D3	333

4.3 SQL-92

Agrupación

```
SELECT D.nombre, nssdire
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep=dep
GROUP BY dep
HAVING AVG(salario)<1200 ;
```

Ahora se **seleccionan** los grupos que cumplen lo indicado en el HAVING

nombre	apellido	salario	...	dep	D.nombre	coddep	nssdire	AVG(salario)
JONÁS	SOLANO	1100	…	D1	ADMINISTRACION	D1	111	1100 ☑
MACARENO	SOSO	1100	…	D1	ADMINISTRACION	D1	111	
FILOMENA	RASCAS	1100	…	D1	ADMINISTRACION	D1	111	
EUSEBIO	MULETAS	2100	…	D2	INVESTIGACION	D2	222	2100
RIGOBERTA	CALAVERA	900	…	D3	PERSONAL	D3	333	910 ☑
CASIANA	FABERGÉ	920	…	D3	PERSONAL	D3	333	

4.3 SQL-92

Agrupación

```
SELECT D.nombre, nssdire
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep=dep
GROUP BY dep
HAVING AVG(salario)<1200 ;
```

Por último, se aplica la **cláusula SELECT** a cada grupo

D.nombre	D.nssdire
ADMINISTRACION	111
PERSONAL	333

** Para cada departamento **en el que trabajen más de diez empleados**, obtener el código y nombre del departamento, y el nº de empleados que trabajan en él*

```
SELECT coddep, D.nombre, COUNT(*)
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep=dep
GROUP BY coddep, D.nombre
HAVING COUNT(*) > 10 ;
```

4.3 SQL-92

Agrupación: HAVING vs. WHERE

- **WHERE...** se aplica a **filas individuales**
- **HAVING...** se aplica a **grupos de filas**

** Nombre del departamento y nº de empleados cuyos salarios superan los 1.800€, pero sólo en el caso de departamentos en los que trabajen más de 5 empleados.*

```
SELECT D.nombre, COUNT(*)
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep= dep
   AND salario > 1800
GROUP BY D.nombre
HAVING COUNT(*) > 5 ;

/* Consulta incorrecta ¿por qué? */
/* Pista: orden de ejecución */
```

```
SELECT D.nombre, COUNT(*)
FROM DEPARTAMENTO D, EMPLEADO
WHERE coddep = dep
 AND salario > 1800
 AND dep IN (SELECT dep
             FROM EMPLEADO
             GROUP BY dep
             HAVING COUNT(*)>5)
GROUP BY D.nombre ;
                  /* Consulta correcta */
```

4.3 SQL-92

Tablas reunidas

- Reunión especificada **en la cláusula FROM** de una consulta

  ** Nombre, apellido y dirección de cada empleado del departamento de Investigación*

```
SELECT E.nombre, apellido, direccion
FROM (EMPLEADO E JOIN DEPARTAMENTO D          ← tabla reunida
      ON dep=coddep)
WHERE D.nombre='INVESTIGACION';
```

- Hasta ahora la hemos especificado en cláusulas FROM y WHERE

  ** Nombre, apellido y dirección de cada empleado del departamento de Investigación*

```
SELECT E.nombre, apellido, direccion
FROM EMPLEADO E, DEPARTAMENTO D          ← reunión de tablas
WHERE dep=coddep                         ← condición de reunión
  AND D.nombre='INVESTIGACION';
```

- Usar tablas reunidas genera consultas más comprensibles: separa las condiciones de reunión de las de selección

4.3 SQL-92

Tablas reunidas: reunión interna

- Formato

  SELECT ...
  FROM **(R1 JOIN R2 ON <condición_reunión>)**
  WHERE ...

- Si existe una fila t1 en R1 y otra fila t2 en R2, tales que cumplen la condición de reunión, la tabla resultado (reunida) incluirá la fila obtenida al combinar t1 y t2

  ** Nombres de cada empleado y de su jefe o supervisor*

```
SELECT E.nombre empleado, S.nombre supervisor
FROM (EMPLEADO E JOIN EMPLEADO S ON E.nssjefe = S.nss);
```

  – IMPORTANTE: son excluidas las filas EMPLEADO con NULL en `nssjefe`

4.3 SQL-92

Tablas reunidas: anidamiento y tipos

- Es posible anidar varias especificaciones de reunión de tablas

** Empleados (dni y nombre) del departamento dirigido por 'MACARENO SOSO'.*

```
SELECT dni, E.nombre
FROM (EMPLEADO E
      JOIN (SELECT coddep
            FROM (DEPARTAMENTO JOIN EMPLEADO J ON nssdire=nss)
            WHERE J.nombre='MACARENO' AND apellido='SOSO')
      ON dep=coddep);
```

- El concepto de tabla reunida también permite especificar **diferentes tipos de join**, además de la interna (que ya hemos visto):
  - Reunión Natural
  - Reunión Externa

4.3 SQL-92

Reunión Natural de tablas (natural join)

- Formato

  SELECT ...
  FROM **(R1 NATURAL JOIN R2)**
  WHERE ...

- NO necesita condición de reunión
  - No incluye la cláusula ON condición
- (El sistema) Asume **una condición de reunión para cada par de columnas con igual nombre en una y otra tabla**
- Y sólo se incluye **una de estas columnas** en el resultado

4.3 SQL-92

Reunión Natural de tablas (natural join)

- Sea la relación

OFICINA(*coddep*, oficina, ubicacion)

que almacena las distintas oficinas de cada departamento,
Y donde ***coddep*** es una referencia a **DEPARTAMENTO.coddep**

```
SELECT *
FROM (OFICINA NATURAL JOIN DEPARTAMENTO);
```

- Lo que asume y ejecuta el sistema:

```
SELECT *
FROM (OFICINA O JOIN DEPARTAMENTO D ON (O.coddep = D.coddep)
```

- Relación resultado: (**coddep**, oficina, ubicacion, nombre, nssdire)
- Si en vez de todos, sólo se seleccionan algunas columnas…

```
SELECT coddep, nombre, oficina
FROM (DEPARTAMENTO NATURAL JOIN OFICINA);
```

  - No es necesario calificar la columna coddep

4.3 SQL-92

Reunión Natural de tablas (natural join)

OFICINA

coddep	oficina	ubicacion
D1	O1	Murcia
D3	O1	Madrid
D2	O1	Alicante
D1	O3	Toledo
D3	O2	Alicante
D1	O2	Albacete

DEPARTAMENTO

nombre	coddep	nssdire
INVESTIGACION	D2	222
ADMINISTRACION	D1	111
PERSONAL	D3	333

OFICINA NATURAL JOIN DEPARTAMENTO

coddep	oficina	ubicacion	nombre	nssdire
D1	O1	Murcia	ADMINISTRACION	111
D3	O1	Madrid	PERSONAL	333
D2	O1	Alicante	INVESTIGACION	222
D1	O3	Toledo	ADMINISTRACION	111
D3	O2	Alicante	PERSONAL	333
D1	O2	Albacete	ADMINISTRACION	111

4.3 SQL-92

Reunión Natural de tablas (natural join)

- El SQL de Oracle **sí** implementa la reunión natural, pero si se usan alias para las tablas, **no** se puede calificar una columna de reunión
  - Al anteponer el nombre de la tabla, las columnas no tendrían el mismo nombre…
  - Es decir, esto daría ERROR:

```
SELECT *
FROM (DEPARTAMENTO D NATURAL JOIN OFICINA O )
WHERE D.coddep <> 'D1';
```

-- Se debe quitar el prefijo **D.** … y funcionará!

4.3 SQL-92

Reunión Externa de tablas (outer join)

- Necesaria si en una reunión se necesita obtener las filas…
  - con valor NULL en las columnas de reunión, o
  - sin correspondencia en la otra tabla
- Tipos de reunión externa:
  - **LEFT** [OUTER] **JOIN** Reunión externa **izquierda**
  - **RIGHT** [OUTER] **JOIN** Reunión externa **derecha**
  - **FULL** [OUTER] **JOIN** Reunión externa **completa** o **total**

** Nombre de cada empleado y de su jefe, incluyendo los empleados que no tienen jefe*

```
SELECT E.nombre AS empleado, S.nombre AS supervisor
FROM (EMPLEADO E
      LEFT OUTER JOIN EMPLEADO S ON E.nssjefe=S.nss);
```

** Nombres de cada empleado y de su departamento, admitiendo los empleados sin departamento asignado y los departamentos sin empleados*

4.3 SQL-92

Reunión Externa de tablas (outer join)

EMPLEADO

nombre	apellido	nss	...	dep
JONÁS	SOLANO	123	...	D1
RIGOBERTA	CALAVERA	321	...	D3
EUSEBIO	MULETAS	222	...	D2
MACARENO	SOSO	111	...	D1
CASIANA	FABERGÉ	333	...	D3
FILOMENA	RASCAS	234	...	D1
GUMERSINDA	MIMOS	543	...	

DEPARTAMENTO

nombre	coddep	nssdire
INVESTIGACION	D2	222
ADMINISTRACION	D1	111
PERSONAL	D3	333
TRAINING	D4	

** Nombres de empleados y de sus departamentos, incluyendo...*

1.- *los* ***empleados que no tienen departamento***

```
SELECT E.nombre, D.nombre
 FROM (EMPLEADO E LEFT OUTER JOIN DEPARTAMENTO D ON dep=coddep);
```

2.- *los* ***departamentos que no tienen empleados***

```
SELECT E.nombre, D.nombre
 FROM (EMPLEADO E RIGHT OUTER JOIN DEPARTAMENTO D ON dep=coddep);
```

3.- ***tanto empleados sin departamento, como departamentos sin empleados***

```
SELECT E.nombre, D.nombre
FROM (EMPLEADO E FULL OUTER JOIN DEPARTAMENTO D ON dep=coddep);
```

4.3 SQL-92

Reunión Externa de tablas (outer join)

EMPLEADO LEFT OUTER JOIN DEPARTAMENTO ON dep=coddep

nombre	apellido	nss	...	dep	nombre	coddep	nssdire
JONÁS	SOLANO	123	…	D1	ADMINISTRACION	D1	111
RIGOBERTA	CALAVERA	321	…	D3	PERSONAL	D3	333
EUSEBIO	MULETAS	222	…	D2	INVESTIGACION	D2	222
MACARENO	SOSO	111	…	D1	ADMINISTRACION	D1	111
CASIANA	FABERGÉ	333	…	D3	PERSONAL	D3	333
FILOMENA	RASCAS	234	…	D1	ADMINISTRACION	D1	111
GUMERSINDA	MIMOS	543	…				

EMPLEADO RIGHT OUTER JOIN DEPARTAMENTO ON dep=coddep

nombre	apellido	nss	...	dep	nombre	coddep	nssdire
JONÁS	SOLANO	123	…	D1	ADMINISTRACION	D1	111
RIGOBERTA	CALAVERA	321	…	D3	PERSONAL	D3	333
EUSEBIO	MULETAS	222	…	D2	INVESTIGACION	D2	222
MACARENO	SOSO	111	…	D1	ADMINISTRACION	D1	111
CASIANA	FABERGÉ	333	…	D3	PERSONAL	D3	333
FILOMENA	RASCAS	234	…	D1	ADMINISTRACION	D1	111
					TRAINING	D4	

4.3 SQL-92

Reunión Externa de tablas (outer join)

EMPLEADO FULL OUTER JOIN DEPARTAMENTO ON dep=coddep

nombre	apellido	nss	...	dep	nombre	coddep	nssdire
JONÁS	SOLANO	123	…	D1	ADMINISTRACION	D1	111
RIGOBERTA	CALAVERA	321	…	D3	PERSONAL	D3	333
EUSEBIO	MULETAS	222	…	D2	INVESTIGACION	D2	222
MACARENO	SOSO	111	…	D1	ADMINISTRACION	D1	111
CASIANA	FABERGÉ	333	…	D3	PERSONAL	D3	333
FILOMENA	RASCAS	234	…	D1	ADMINISTRACION	D1	111
GUMERSINDA	MIMOS	543	…				
					TRAINING	D4	

- El SQL de Oracle **sí** implementa todos los tipos de reunión externa
  - Además mantiene la notación **(+)** por compatibilidad con versiones anteriores a Oracle9i

4.3 SQL-92

❑Vistas en línea (*online views*) en SQL de Oracle

- Es una **subconsulta** (un SELECT…) **dentro de la cláusula FROM** de otra consulta
- **No es una subconsulta anidada** puesto que no aparece dentro de la cláusula WHERE de la consulta exterior, sino en la FROM

** Para cada departamento, mostrar los datos de los empleados que cobran el máximo salario:*

```
SELECT X.dep, E.nss, E.nombre, E.apellido, E.salario
FROM  EMPLEADO E,
      (SELECT dep, MAX(salario) AS max_sal
       FROM EMPLEADO
       GROUP BY dep) X
WHERE E.dep=X.dep AND E.salario=X.max_sal;
```

4.3 SQL-92

Evaluación de consultas

- En una consulta SQL hay un máximo de **6** cláusulas
- Sólo son **obligatorias SELECT** y **FROM**
- Debe terminar con un **;**

- **Orden de especificación** (escritura) de las cláusulas:

SELECT <lista columnas> -- columnas o funciones que desea obtener

FROM <lista tablas> -- tablas necesarias (incluso las reunidas)

WHERE <condición para filas> -- condiciones para selección de filas o de reunión

GROUP BY <lista columnas agrupación> -- especificación del agrupamiento de filas

HAVING <condición para grupos> -- condición de selección de grupos de filas

ORDER BY <lista columnas ordenación> -- orden de presentación del resultado

4.3 SQL-92

Evaluación de consultas

- **Orden de evaluación** de las cláusulas:
  1) **FROM** (es decir, la reunión o JOIN de tablas, si se especifica más de una)
  2) **WHERE**
  3) **GROUP BY**
  4) **HAVING**
  5) **ORDER BY**
  6) **SELECT**
- **Diversas formas** de especificar una **misma consulta**

  Ejemplo: es posible expresar una consulta utilizando...

  a) Lista de tablas en el FROM y condiciones de reunión en el WHERE, o

  b) Tablas reunidas en la cláusula FROM, o

  c) Consultas anidadas (correlacionadas o no) y comparación mediante IN **...**

  - Flexibilidad

Evaluación de consultas

- Ventajas e inconvenientes de esta flexibilidad:
  - ☺ – El **usuario elige** la técnica o enfoque más **cómodo**
  - ☹ – Confusión del usuario: ¿qué técnica uso?
    - – Algunas técnicas son más eficientes que otras
      - El usuario debe determinar cuál... ¡probar!
- En **condiciones ideales**...
  - – Usuario: se preocupa sólo de **especificar** la consulta **correctamente**
  - – SGBD: se ocupa de **ejecutar** la consulta de manera **eficiente**
- Pero en la **práctica** no suele ser así...
  - conviene **saber** qué tipos de consulta son más y menos costosos

4.3 SQL-92

Evaluación de consultas: recomendaciones...

- Escribir las consultas con el mínimo **anidamiento correlacionado** y con el mínimo **ordenamiento implícito**
  - Evitar consultas correlacionadas y
  - No usar innecesariamente las cláusulas ORDER BY, DISTINCT, GROUP BY…
- Por orden de **elegancia**
  1. IN
  2. EXISTS
  3. JOIN
- Por orden de **eficiencia**
  1. IN (sin correlación)
  2. JOIN
  3. EXISTS
  - Aunque depende mucho del esquema de BD, de las estructuras de almacenamiento empleadas, y del SGBD (optimizador) de que se trate
  - Es necesario estudiar los manuales de uso del SGBD concreto (Oracle, por ejemplo) y ¡probar!