MANIPULACIÓN DE DATOS: LENGUAJES RELACIONALES - SQL2

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

- **Primer lenguaje de BD de alto nivel**. Años 70.
  - Diseñado e implementado en IBM Research (San José - California), para el SGBD Relacional experimental SYSTEM R
- **Definición de un lenguaje estándar**

  ANSI (American National Standars Institute)
  + **ISO** (Organization for International Standardization)
  - **SQL1** (ANSI 1986), extendido en 1989 (**SQL-89**)
  - SQL-92 (**SQL2**), y
  - **SQL3** (extensiones de Orientación a Objetos y otros conceptos recientes; denominado **SQL:1999**)

SQL2

- **Lenguaje de Bases de Datos COMPLETO** **(No sólo "de consulta")**
  - Definición y Manipulación de Datos (DDL + DML)
  - Definición de Vistas (VDL)
  - Creación y Destrucción de Índices (aunque en SQL2 "ya no existen")
  - Incorporación de SQL dentro de código escrito con un Lenguaje de Programación de propósito general (Pascal, C, etc.)
- Los proveedores de SGBDR comerciales implementan **variaciones de SQL**
  - algunos incluyen características que no están estandarizadas (triggers/reglas activas)
- Niveles de compatibilidad con el estándar de SQL
  - *Entry SQL*
  - *Intermediate SQL*
  - *Full SQL*

Estructura del seminario sobre SQL2

- **Consultas o Selección de datos**
- **Modificación de datos**
- **Vistas**
- **Definición y Alteración de datos**
  - **Esquemas, Dominios, Relaciones**
- **Restricciones de Integridad Generales (Asertos)**
- **Seguridad y Control de Acceso**

Anexo: resumen de la sintaxis o estructura de SQL-92

CONSULTAS BÁSICAS EN SQL

- Cláusula **SELECT**:
  Instrucción básica de obtención de información

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

donde:

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

Fecha de nacimiento y dirección del empleado llamado José B. Silva

```
SELECT fechan, dirección
FROM EMPLEADO
WHERE nombrep = 'José' AND inic = 'B' AND apellido = 'Silva';
```

- La consulta **selecciona tuplas de <lista tablas>**
  que satisfacen <condición> y
  proyecta el resultado sobre los atributos de **<lista atributos>**

CONSULTAS BÁSICAS EN SQL2

La cláusula SELECT ... FROM ... WHERE...

- No es igual a la operación Restricción σ del Álgebra Relacional

  SELECT de SQL tiene muchas más opciones y matices

- Para el caso de una **única tabla T** en <lista tablas>, es equivalente a...

  $\Pi_{\text{<lista atributos>}}(\sigma_{\text{<condición>}}(\text{<T>}))$

- **SQL2** vs. **Modelo Relacional Formal** (MRF)

  - **SQL2 permite** que las relaciones (tablas) tengan **2 o más tuplas idénticas** en todos los valores de sus atributos (el MRF no!!)

  ⇒ Tabla SQL ≠ conjunto de tuplas (en general)

  Tabla SQL = **Multiconjunto** (*bag*, bolsa, saco) **de tuplas**

  - Es posible forzar que las tablas SQL sean conjuntos de tuplas:
    - mediante opción DISTINCT (en una SELECT) o
    - con restricciones de clave

CONSULTAS BÁSICAS EN SQL

Nombre, apellido y dirección de todos los empleados del depto de Investigación

```
SELECT nombrep, apellido, dirección
FROM EMPLEADO, DEPARTAMENTO
WHERE nombred='Investigación'        → condición de selección
      AND númerod=nd;                → condición de reunión entre las tablas
```

- Cualquier número de **condiciones selección/reunión** en SELECT

Para cada proyecto ubicado en Santiago, obtener el nº del proyecto, nº del depto que lo controla y el apellido, dirección y fecha de nacimiento del gerente de ese depto

```
SELECT númerop, númd, apellido, dirección, fechan
FROM PROYECTO, DEPARTAMENTO, EMPLEADO
WHERE númd=númerod AND nssgte=nss AND lugarp='Santiago';
```

- **Una SELECT puede obtener tuplas repetidas**

Salario de los empleados de los departamentos de Administración e Investigación

```
SELECT salario
FROM EMPLEADO, DEPARTAMENTO
WHERE (nombred='Administración' OR nombred='Investigación')
      AND númerod=nd;
```

CONSULTAS BÁSICAS: OMISIÓN DE WHERE, y *

- Omisión de WHERE ⇒ **SELECCIÓN INCONDICIONAL** ≡ WHERE TRUE
  ⇒ selección de **todas las tuplas**
  - de la relación (única) de la cláusula FROM, o
  - del producto cartesiano o cruzado (si FROM contiene varias tablas)

Seleccionar todos los nss de empleados

```
SELECT nss FROM EMPLEADO;
```

Seleccionar todas las combinaciones de nss de empleados y nombres de deptos

```
SELECT nss, nombred
FROM EMPLEADO, DEPARTAMENTO;
```

- Obtención de los valores de **todos los atributos** de las tuplas seleccionadas
  - Uso del símbolo * (*TODOS LOS ATRIBUTOS*)
  - **No** es **necesario listar todos los nombres** tras cláusula SELECT

```
SELECT * FROM EMPLEADO WHERE nd=5;
SELECT * FROM EMPLEADO,DEPARTAMENTO
  WHERE nombred='Investigación' AND nd=númerod;
SELECT * FROM EMPLEADO, DEPARTAMENTO;
```


CONSULTAS BÁSICAS: CADENAS DE CARACTERES

- **COMPARACIÓN DE SUBCADENAS DE CARACTERES**

Operador LIKE

- Caracteres reservados: **'%'** y **'_'** (**comodines**)

Nombres y apellidos de los empleados cuya dirección esté en Higueras, estado de México

```
SELECT apellido, nombrep FROM EMPLEADO
WHERE dirección LIKE '%Higueras, MX%' ;
```

- **CONCATENACIÓN DE CADENAS DE CARACTERES**

Operador ||

Nombres completos de los empleados cuya dirección esté en Higueras, estado de México

```
SELECT nombrep || inic || apellido FROM EMPLEADO
WHERE dirección LIKE '%Higueras, MX%' ;
```

CONSULTAS BÁSICAS: ARITMÉTICA y TIEMPO

· **OPERACIONES ARITMÉTICAS**

– Aplicación de operadores aritméticos (+ - * /) sobre valores numéricos

Salarios de los empleados que trabajan en el proyecto ProductoX, tras un aumento del 10%

```
SELECT apellido, nombrep, 1.1*salario
FROM EMPLEADO, TRABAJA_EN, PROYECTO
WHERE nss=nsse AND númp=númerop AND nombrepr='ProductoX' ;
```

(*ojo: el valor real de los salarios en la tabla EMPLEADO **no** cambia.*)

· **OPERACIONES CON FECHAS,HORAS,MARCAS DE TIEMPO E INTERVALOS**

- **Incremento (+) y Decremento(-)** de atributos tipo **DATE, TIME, TIMESTAMP** en un **intervalo** compatible con el tipo
- Especificación del **valor de un INTERVAL como diferencia de dos** valores **DATE, TIME o TIMESTAMP**

CONSULTAS BÁSICAS: CALIFICACIÓN

☞En SQL los nombres de **atributos** deben ser **únicos dentro de** cada **relación**

- Consulta que referencia a varios **atributos de igual nombre, pero de relaciones distintas**

  ⇒ **AMBIGÜEDAD ---------- Solución: CALIFICACIÓN**

  Ejemplo distinto al del esquema COMPAÑÍA:
  Código, nombre y direccion de los empleados del departamento de Marketing

```
EMPLEADO(codEmp, nombre, codDepto, dirección, ...)
DEPARTAMENTO(codDepto, nombre, ...)

SELECT codEmp, EMPLEADO.nombre, dirección
FROM EMPLEADO, DEPARTAMENTO
WHERE DEPARTAMENTO.nombre ='Marketing'
    AND DEPARTAMENTO.codDepto = EMPLEADO.codDepto;
```

CONSULTAS BÁSICAS: SEUDÓNIMOS

- Puede utilizarse los **seudónimos** para **acortar** nombres de relación dentro de las **consultas con calificación**:

EMPLEADO(codEmp, **nombre**, **codDepto**, dirección, ...)
DEPARTAMENTO(**codDepto**, **nombre**, ...)

```
SELECT codEmp, E.nombre, dirección
FROM EMPLEADO E, DEPARTAMENTO D
WHERE D.nombre ='Investigación'
    AND D.codDepto = E.codDepto;
```

- Consulta que se refiere **DOS veces** a la **misma relación** (recursividad de 1 nivel)

⇒ **AMBIGÜEDAD ---------- Solución: SEUDÓNIMOS**

Obtener nombre y apellido de cada empleado y de su supervisor (jefe) inmediato

```
SELECT E.nombrep, E.apellido, S.nombrep, S.apellido
FROM EMPLEADO E, EMPLEADO AS S
WHERE E.nsssuper=S.nss;
```


CONSULTAS BÁSICAS: CAMBIO DE NOMBRE DE ATRIBUTOS

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

Nombres de cada empleado y su supervisor, cambiando al mismo tiempo los nombres de los atributos resultantes a nombre_empleado y nombre_supervisor

```
SELECT E.nombrep AS nombre_empleado,
       S.nombrep AS nombre_supervisor
FROM EMPLEADO AS E, EMPLEADO AS S
WHERE E.nsssuper = S.nss;
```

⇒ Nueva cabecera para la relación resultado

- **Seudónimos de atributos (y/o relación)** en cláusula FROM

EMPLEADO(codEmp, **nombre**, **codDepto**, dirección, ...)
DEPARTAMENTO(**codDepto**, **nombre**, ...)

```
SELECT cod, nom, dir
FROM EMPLEADO E( cod, nom, dep , dir,...), DEPARTAMENTO
WHERE nombre ='Investigación' AND dep = codDepto;
```

CONSULTAS BÁSICAS: ORDEN DE PRESENTACIÓN DE LAS TUPLAS

- **SQL** permite **presentar las tuplas resultado de** una **consulta de forma ordenada**
  - Ordenación **según valores de** uno o varios **atributos**
  - **Ascendente ASC** (por defecto) o **Descendente DESC**
  - Suele ser una operación muy costosa

☞ las tuplas NO se ordenan en disco: **se *ven* ordenadas**, pero **no lo *están***

Nombre y apellido de los empleados y proyectos en los que trabajan, en orden descendente por departamentos y, dentro de c/depto, en orden alfabético y ascendente por apellido y nombre

```
SELECT nombred, apellido, nombrep, nombrepr
FROM DEPARTAMENTO, EMPLEADO, TRABAJA_EN, PROYECTO
WHERE númerod=nd AND nss=nsse AND númp=númerop
ORDER BY nombred DESC, apellido ASC, nombrep ASC;
```


TABLAS COMO CONJUNTOS EN SQL2

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

Salario de todos los empleados

```
SELECT salario
FROM EMPLEADO;
```

Salarios distintos de los empleados (sin importar cuántos perciban cada cantidad)

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

TABLAS COMO CONJUNTOS

OPERACIONES DE CONJUNTOS

- **UNION**(∪), **INTERSECT**(∩), **EXCEPT** (—)
- Resultado: conjuntos de tuplas ---**las tuplas repetidas se eliminan**
- Para **NO eliminar duplicados**: UNION ALL, INTERSECT ALL, EXCEPT ALL
- Las relaciones han de ser **compatibles:**

igual **nº** de atributos, **en** el **mismo orden** y con el mismo **dominio**

Números de los proyectos en los que participa el empleado de apellido Silva, ya sea como trabajador, o bien como gerente del departamento que controla el proyecto

```
( SELECT númerop
  FROM PROYECTO, DEPARTAMENTO, EMPLEADO
  WHERE númd=númerod AND nssgte=nss AND apellido='Silva' )
UNION
( SELECT númerop
  FROM PROYECTO, TRABAJA_EN, EMPLEADO
  WHERE númerop=númp AND nsse=nss AND apellido='Silva' );
```


TABLAS COMO CONJUNTOS: CONJUNTOS EXPLÍCITOS

CONJUNTO EXPLÍCITO DE VALORES en cláusula WHERE

- Lista de valores encerrada entre paréntesis

nss de los empleados que trabajan en los proyectos 1, 2 ó 3

```
SELECT DISTINCT nsse FROM TRABAJA_EN
WHERE númp IN (1, 2, 3);
```

- **Operador IN**

v IN V indica si el **valor** pertenece al **conjunto de valores V**

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

nss de los empleados que trabajan en algún proyecto que no sea el 4 ni el 6

```
SELECT DISTINCT nsse FROM TRABAJA_EN
WHERE númp NOT IN (4, 6);
```

TABLAS COMO CONJUNTOS: CONJUNTOS EXPLÍCITOS

- **Operador ANY (o SOME)**

  v <op> ANY V o **v <op> SOME V** ,, donde <op> ∈ { >, ≥, <, ≤, <>, = }
  - Compara un valor individual **v** con un conjunto **V**
  - Devuelve TRUE si algún elemento **e** de **V** cumple que **v <op> e**

nss de los empleados que trabajan en alguno de los proyectos 1, 2 ó 3

```
SELECT DISTINCT nsse FROM TRABAJA_EN
WHERE númp = ANY (1, 2, 3);
```

- **Operador ALL**

  v <op> ALL V,, donde <op> ∈ { >, ≥, <, ≤, <>, = }
  - Compara un valor **v** con conjunto **V**
  - Devuelve TRUE si **para todo** elemento **e** de **V** se cumple que **v <op> e**

nss de los empleados que no trabajan en ninguno de los tres proyectos 1, 2 y 3

```
SELECT DISTINCT nsse FROM TRABAJA_EN
WHERE númp <> ALL (1, 2, 3);
```


CONSULTAS ANIDADAS EN SQL2

CONSULTA ANIDADA:

- valores obtenidos de la BD que se usan en la condición de otra consulta, para obtener otros datos
- **Es una consulta** SELECT **completa, dentro de cláusula** WHERE **de otra consulta (**consulta exterior**)**
- Es posible tener **varios niveles de consultas anidadas**

Números de los proyectos en los que participa el empleado de apellido Silva, como trabajador o como gerente del departamento que controla el proyecto

```
SELECT DISTINCT númerop  FROM PROYECTO
 WHERE númerop IN ( SELECT númerop
                    FROM PROYECTO,DEPARTAMENTO,EMPLEADO
                    WHERE númd=númerod
                       AND nssgte=nss AND apellido='Silva' )
        OR
        númerop IN (SELECT númp
                    FROM TRABAJA_EN, EMPLEADO
                    WHERE nsse=nss AND apellido='Silva' );
```

CONSULTAS ANIDADAS: COMPARACIÓN DE CONJUNTOS

- **Operador IN** (otro uso del mismo operador)
  t IN S indica si la **tupla** t pertenece al **conjunto de tuplas S** (subconsulta)

Nombre y dirección de los empleados que trabajan en algún proyecto.

```
SELECT nombrep, dirección FROM EMPLEADO
WHERE nss  IN ( SELECT nsse  FROM TRABAJA_EN );
```

Nºs de seguridad social de aquellos empleados que trabajan en algún proyecto en el que trabaje el empleado José B. Silva, de forma tal que ambos tengan la misma combinación (proyecto, horas), es decir todo empleado que trabaje las mismas horas que José B. Silva, en cada proyecto en el que trabajen ambos. El nss de José B. Silva es '123456789'.

```
SELECT DISTINCT nsse FROM TRABAJA_EN
WHERE (númp, horas)  IN ( SELECT númp, horas  FROM TRABAJA_EN
                          WHERE nsse='123456789');
```


CONSULTAS ANIDADAS: COMPARACIÓN DE CONJUNTOS

- **Operador ANY** o **SOME** (otro uso del mismo operador)
  t <op> ANY S o **t <op> SOME S,,** donde <op> ∈ { >, ≥, <, ≤, ≠, = }
  - Compara una tupla **t** con las tuplas resultado de una consulta anidada
  - Devuelve TRUE si alguna tupla **e** de **S** cumple que **t <op> e**

- **Operador ALL** (otro uso del mismo operador)
  t <op> ALL S,, donde <op> ∈ { >, ≥, <, ≤, ≠, = }
  - Compara una tupla **t** con las tuplas resultado de una consulta anidada
  - Devuelve TRUE si **para toda** tupla **e** de **S** se cumple que **t <op> e**

Nombres y apellidos de los empleados cuyo salario es menor que el de TODOS los empleados del departamento 5

```
SELECT nombrep, apellido FROM EMPLEADO
WHERE salario < ALL ( SELECT salario
                      FROM EMPLEADO
                      WHERE nd=5 );
```

CONSULTAS ANIDADAS: ATRIBUTOS AMBIGUOS

• Colisión de nombres de atributos entre consulta exterior y anidada ⇒ **Ambigüedad**

Nombres y apellidos de cada empleado con dependientes de igual nombre y sexo que él

```
SELECT nombrep, apellido FROM EMPLEADO E
WHERE nss IN ( SELECT nsse  FROM DEPENDIENTE
                    WHERE nsse=nss AND nombrep=nombre_dep
                       AND sexo=sexo);
```

Solución:

• REGLA: **"una referencia a un *atributo no calificado*, se refiere a la relación declarada en la *consulta anidada más interior*"**

• Si es necesario **hacer referencia desde** una consulta **anidada**, a atributos de **relaciones** declaradas en una consulta **externa** ⇒ **CALIFICAR**

Nombres y apellidos de cada empleado con dependientes de igual nombre y sexo que él

```
SELECT nombrep, apellido FROM EMPLEADO E
WHERE nss IN ( SELECT nsse  FROM DEPENDIENTE
                    WHERE nss=nsse AND nombrep=nombre_dep
                       AND E.sexo=sexo);
```


CONSULTAS ANIDADAS: CORRELACIÓN

· **CONSULTAS CORRELACIONADAS**:

– Una condición de la consulta anidada hace referencia a un atributo de una relación declarada en la consulta exterior

– La **consulta anidada se evalúa una vez para cada tupla** (o combinación de tuplas) **de la consulta exterior**

```
SELECT nombrep, apellido FROM EMPLEADO E
WHERE nss IN ( SELECT nsse  FROM DEPENDIENTE
                    WHERE nss=nsse AND nombrep=nombre_dep AND E.sexo=sexo);
```

· Evaluar la consulta anidada para cada tupla de EMPLEADO,

· Si el valor de nss de la tupla EMPLEADO está en el resultado de la consulta anidada, seleccionar la tupla EMPLEADO para el resultado total

• Una **CONSULTA ANIDADA que use el operador =** o **IN siempre puede expresarse** como una **ÚNICA CONSULTA**

```
SELECT E.nombrep, E.apellido
  FROM EMPLEADO E, DEPENDIENTE D
  WHERE E.nss=D.nsse AND E.nombrep=D.nombre_dep AND E.sexo=D.sexo;
```

CONSULTAS ANIDADAS: EXISTS y UNIQUE

- **COMPROBACIÓN DE RELACIONES VACÍAS: EXISTS (S)**
  - Devuelve TRUE si la relación S contiene al menos una tupla
  - Devuelve FALSE si S es una relación vacía (sin tuplas)

S suele ser una consulta anidada correlacionada

Nombres y apellidos de empleados con algún dependientes de igual nombre y sexo que él

```
SELECT E.nombrep, E.apellido FROM EMPLEADO E
WHERE EXISTS ( SELECT * FROM DEPENDIENTE
               WHERE E.nss=nsse AND sexo=E.sexo AND E.nombrep=nombre_dep);
```

Nombres de empleados sin personas dependientes

```
SELECT nombrep, apellido FROM EMPLEADO E
WHERE NOT EXISTS ( SELECT * FROM DEPENDIENTE WHERE nss=nsse );
```

- **COMPROBACIÓN DE TUPLAS DUPLICADAS: UNIQUE (S)**
  - Devuelve TRUE si NO hay tuplas repetidas en S

S suele ser una consulta anidada correlacionada

Nombres y apellidos de Empleados que trabajan en un único proyecto

```
SELECT nombrep, apellido FROM EMPLEADO
WHERE UNIQUE ( SELECT nsse FROM TRABAJA_EN WHERE nss=nsse );
```


VALORES NULOS (NULL)

(*
Recordamos...
Cada **NULL** es **distinto de todos** los demás **NULL**
⇒ en una REUNIÓN, las tuplas con NULL en los atributos de reunión NO se incluyen en el resultado (excepto si REUNIÓN EXTERNA)
*)

- **Operador IS NULL ,, IS NOT NULL**
  - **v IS NULL** → es TRUE si el valor **v** es NULL (información desconocida)
  - **v IS NOT NULL** → es TRUE si el valor **v** es un valor no NULL

Nombres de empleados sin supervisores

```
SELECT nombrep, apellido FROM EMPLEADO
WHERE nsssuper IS NULL;
```

FUNCIONES AGREGADAS

- **Operación COUNT( )**
  - Cuenta el número de tuplas o valores especificados en una consulta
- **Operación SUM( ), MAX( ), MIN( ), AVG( )**
  - Suma, máximo, mínimo y media aritmética
  - Aplicadas a un multiconjunto (bag, saco) de valores numéricos
- Pueden ser usadas en cláusula SELECT o en HAVING (*se verá*)

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

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

Suma de los salarios, salario máximo, salario mínimo y salario medio, pero de los empleados del departamento de Investigación

```
SELECT SUM(salario), MAX(salario), MIN(salario), AVG(salario)
FROM EMPLEADO, DEPARTAMENTO
WHERE nd=númerod AND nombred='Investigación';
```


FUNCIONES AGREGADAS

- **USO DE * EN LAS FUNCIONES AGREGADAS**

Número total de empleados de la compañía

```
SELECT COUNT(*) FROM EMPLEADO
```
(cuenta filas o tuplas)

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

```
SELECT COUNT(nsssuper) FROM EMPLEADO;
```
(no cuenta tuplas con nsssuper a NULL)

Número de empleados en el departamento de Investigación

```
SELECT COUNT(*) FROM EMPLEADO, DEPARTAMENTO
WHERE nd=númerod AND nombred='Investigación';
```

- **USO DE DISTINCT EN LAS FUNCIONES AGREGADAS**

Contar el número 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 !!

FUNCIONES AGREGADAS

- **FUNCIONES AGREGADAS EN CONSULTAS CORRELACIONADAS**
  - Subconsulta que incluye la función, dentro del WHERE de la consulta exterior

Nombres de los empleados con 2 o más dependientes

```
SELECT apellido, nombrep FROM EMPLEADO
WHERE 2 ≤ ( SELECT COUNT(*)
            FROM dependiente
            WHERE nss=nsse   );
```


AGRUPACIÓN

- **Cláusula GROUP BY**
  - Subgrupos de tuplas dentro de una relación
  - Los grupos se forman según el valor de algunos atributos: atributos de agrupación
  - Las tuplas de cada grupo tendrán igual valor en los atributos de agrupación

- **APLICACIÓN DE FUNCIONES AGREGADAS A GRUPOS**

Para cada departamento, obtener su nº y nombre, el nº de empleados de dicho departamento y el salario medio de los empleados del mismo

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

- **Los atributos de agrupación deben aparecer en cláusula SELECT,** para que su valor (único para cada grupo) aparezca junto al resultado de aplicar la función a cada grupo

AGRUPACIÓN

- **Cláusula HAVING**
  - Siempre junto a GROUP BY
  - **Condición** que **deben cumplir** los ***grupos*** **de tuplas** asociados a cada valor de los atributos de agrupación
  - Un ***grupo*** que NO cumple la condición, no se selecciona para el resultado

Para cada proyecto **en el que trabajen más de dos empleados**, obtener número y nombre del proyecto, y el nº de empleados que trabajan en él

```
SELECT númerop, nombrepr, COUNT(*)
FROM PROYECTO, TRABAJA_EN
WHERE númerop=númp
GROUP BY númerop, nombrepr
HAVING COUNT(*) > 2 ;
```


AGRUPACIÓN

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

Nº total de empleados cuyos salarios rebasan los 40.000$ en cada departamento, pero sólo en el caso de departamentos en los que trabajen más de 5 empleados

```
(* Consulta incorrecta ¿por qué? *)

SELECT nombred, COUNT(*)
FROM DEPARTAMENTO, EMPLEADO
WHERE númerod=nd
    AND salario>40000
GROUP BY nombred
HAVING COUNT(*) > 5 ;

(* pista: orden de ejecución *)
```

```
(* Consulta correcta *)

SELECT nombred, COUNT(*)
FROM DEPARTAMENTO, EMPLEADO
WHERE númerod=nd
    AND salario>40000
    AND nd IN (SELECT nd
                FROM EMPLEADO
                GROUP BY nd
                HAVING COUNT(*) > 5)
GROUP BY nombred ;
```

REUNIÓN DE RELACIONES (JOIN)

Nombres y dirección de empleados del departamento de Investigación

```
SELECT nombrep, apellido, dirección
FROM EMPLEADO, DEPARTAMENTO →  reunión (join) de tablas
WHERE nombred='Investigacion' AND nd=númerod; →  condición de reunión
```

- **RELACIONES o TABLAS REUNIDAS** (nuevo en SQL2)

  Especificación **en la cláusula FROM** de la tabla resultante de una REUNIÓN

  – Antes se especificaba en cláusulas FROM y WHERE

  – Consulta más comprensible (no mezcla condiciones de reunión y de selección)

Nombres y dirección de empleados del departamento de Investigación

```
SELECT nombrep, apellido, dirección
FROM (EMPLEADO JOIN DEPARTAMENTO ON nd=númerod)→ única tabla reunida
WHERE nombred='Investigacion';
```


REUNIÓN DE RELACIONES (JOIN)

- **ANIDAMIENTO DE ESPECIFICACIONES DE REUNIÓN**

Para cada proyecto ubicado en Santiago, obtener nº de proyecto, nº del departamento que lo controla y apellido, dirección y fecha de nacimiento del gerente de ese departamento

```
SELECT númerop, númd, apellido, dirección, fechan
FROM ( ( PROYECTO JOIN DEPARTAMENTO ON númd=númerod )
        JOIN EMPLEADO ON nssgte=nss )
WHERE lugarp='Santiago';
```

REUNIÓN INTERNA DE RELACIONES

- **REUNIÓN INTERNA (inner join)**
  - **Tipo de reunión por defecto**

```
SELECT ...
FROM (R1 JOIN R2 ON <cond_reunión>)
WHERE ...
```

  - Si existe una tupla t1 en una de las relaciones y otra tupla t2 en la otra, tales que cumplen la condición de reunión, la tabla resultado (reunida) incluirá la tupla resultado de combinar t1 y t2

```
SELECT E.nombrep AS nombre_empleado, S.nombrep AS nombre_supervisor
FROM (EMPLEADO E JOIN EMPLEADO S ON E.nsssuper = S.nss);
```

    - Son excluidas las tuplas EMPLEADO con NULL en nsssuper

  - También puede especificarse como R1 **INNER JOIN** R2 **ON** <cond_reunión>

REUNIÓN NATURAL DE RELACIONES

- **REUNIÓN NATURAL (natural join)**
  - SIN condición de reunión

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

  - EQUIRREUNIÓN IMPLÍCITA
    para cada par de atributos de igual nombre en una y otra relación
    - ✓ Cada par de atributos sólo se incluye UNA VEZ en el resultado
    - ✓ Si no coinciden los nombres de los atributos
      RENOMBRAR uno de los atributos, con AS en la cláusula FROM

```
SELECT nombrep, apellido, dirección
FROM (EMPLEADO
        NATURAL JOIN (DEPARTAMENTO AS DEP(nombred, nd, nssg, fech)))
WHERE nombred='Investigacion';
```

REUNIÓN EXTERNA DE RELACIONES

· **REUNIÓN EXTERNA (outer join)**

– Se necesita considerar tuplas con valor NULL en los atributos de reunión

SELECT E.nombrep AS nombre_empleado, S.nombrep AS nombre_supervisor
FROM (EMPLEADO E **LEFT OUTER JOIN** EMPLEADO S **ON** nsssuper=nss);

- INCLUIDAS las tuplas de EMPLEADO E con NULL en nsssuper

· **LEFT** [OUTER] **JOIN** **R**eunión externa **izquierda**
· **RIGHT** [OUTER] **JOIN** Reunión externa **derecha**
· **FULL** [OUTER] **JOIN** Reunión externa **completa** o **total**

EVALUACIÓN DE CONSULTAS SQL2

En una consulta SQL...

- Hay un máximo de **6** cláusulas
- Sólo son **obligatorias SELECT y FROM**

· **ORDEN DE ESPECIFICACIÓN**

SELECT <lista atributos>	atributos o funciones que se van a obtener
FROM <lista tablas>	relaciones necesarias (incluso las reunidas)
WHERE <condición para tuplas>	condiciones para selección de tuplas
GROUP BY <lista atributos agrupación>	especificación del agrupamiento de tuplas
HAVING <condición para grupos>	condición para selección de grupos de tuplas
ORDER BY <lista atributos ordenación>	orden de presentación del resultado de la consulta

· **ORDEN DE EVALUACIÓN**

1) **FROM** (es decir reunión o join de relaciones, si se especifica más de una)
2) **WHERE**
3) **GROUP BY**
4) **HAVING**
5) **ORDER BY**

EVALUACIÓN DE CONSULTAS SQL2

- **FLEXIBILIDAD DE LAS CONSULTAS SQL**
  - **muchas formas** de especificar la **misma consulta**

  ☺ el **usuario elige** la técnica o enfoque más **cómodo**

  Ejemplo: especificación de una consulta con
  - a) condiciones de reunión en cláusula WHERE, o
  - b) relaciones reunidas en la cláusula FROM, o
  - c) consultas anidadas y el operador de comparación IN

  ☹
  - Confusión del usuario ¿qué técnica uso?
  - Algunas técnicas son más eficientes que otras

    ⇒ el usuario debe determinar cuál

  Consejo (optimización de consultas):
  Consultas con **mínimo anidamiento correlacionado y mínimo ordenamiento implícito**

  – En **condiciones ideales**...
  - Usuario: sólo debería preocuparse de **especificar** la **consulta correctamente**
  - SGBD: se ocupa de **ejecutar** la **consulta** de manera **eficiente**

  – En la **práctica** NO es así

  ⇒ mejor si el usuario **sabe** qué tipos de consultas son más y menos costosos

MODIFICACIÓN DE DATOS en SQL2: INSERCIÓN

- ***Orden* INSERT**
  - **Añade** una **tupla completa** a una relación
  - Incluye **nombre de la relación** y **lista de valores para los atributos**, escritos en igual **orden** al especificado en la instrucción CREATE TABLE

```
INSERT INTO EMPLEADO
VALUES ( "Ricardo", 'C', "Martínez", "653298653", "30-DIC-52",
         "Olmo 98, Cedros, MX", 'M', 37000, "987654321", 4 ) ;
```

  - Si se desea poner los valores de los atributos en cualquier orden, hay que especificar los nombres de los atributos en ese mismo orden

```
INSERT INTO EMPLEADO
 (nombrep, inic,apellido,nss,nd,salario, nsssuper, direccion, fechan, sexo)
VALUES ( "Ricardo", 'C', "Martínez", "653298653", 4, 37000, "987654321"
          "Olmo 98, Cedros, MX", "30-DIC-52", 'M' );
```

MODIFICACIÓN DE DATOS: INSERCIÓN

- **INSERCIÓN DE VARIAS TUPLAS en una sola instrucción INSERT**
  - Tuplas separadas por comas
  - Cada tupla se encierra entre paréntesis

- **ESPECIFICACIÓN EXPLÍCITA DE ALGUNOS ATRIBUTOS** (y no todos)
  - **Omisión de atributos cuyo valor se desconoce**
  - Los **atributos no especificados** tomarán el **valor**...
    - Por omisión: valor tomado de su cláusula **DEFAULT**, o bien
    - **NULL**: si no se definió cláusula DEFAULT para el atributo y permite nulos

Inserción de un nuevo empleado del que sólo se conoce su nombre, apellidos y nss

```
INSERT INTO EMPLEADO (nombrep, apellido, nss)
VALUES ( "Ricardo", "Martínez", "653298653" ) ;
```


MODIFICACIÓN DE DATOS: INSERCIÓN

- **MANEJO Y MANTENIMIENTO DE RESTRICCIONES DE INTEGRIDAD**

✓**SGBD con implementación total de SQL2**
- El SGBD maneja e impone **toda RI** especificada con DDL

✓**SGBD con implementación de algunas RI** (menor complejidad, mayor eficiencia)
- **SGBD implementa comprobaciones** para hacer cumplir las **RI que sí maneja**

```
INSERT INTO EMPLEADO (nombrep, apellido, nss, nd)
VALUES ( "Roberto", "Huertas", 2 ) ;
```

» Inserción rechazada: no se incluye valor para el atributo nss, declarado NOT NULL

- **Programador asegura** la **NO violación de las RI no verificadas** por el DBMS

```
INSERT INTO EMPLEADO (nombrep, apellido, nss, nd)
VALUES ( "Roberto", "Huertas", "980760540", 2 ) ;
```

- Si el **SGBD NO maneja la Integridad Referencial** ⇒ inserción permitida
  ¡¡¡El programador debe asegurar que esto NO pase!!!
- Si el **SGBD SÍ maneja la Integridad Referencial** ⇒ inserción rechazada
  no existe departamento con número=2

MODIFICACIÓN DE DATOS: INSERCIÓN

- **INSERCIÓN DE TUPLAS RESULTADO DE UNA CONSULTA**
  - **Carga** de una relación con información sinóptica de la BD

Sea una relación INFO_DEPTOS vacía, en la que queremos almacenar los nombres de cada departamento, su nº de empleados y el salario conjunto de los empleados del mismo

```
INFO_DEPTOS ( nombre_depto, núm_emps, sal_total)
INSERT INTO INFO_DEPTOS ( nombre_depto, núm_emps, sal_total )
                SELECT nombred, COUNT(*), SUM(salario)
                FROM DEPARTAMENTO, EMPLEADO
                WHERE númerod=nd
                GROUP BY nombred ;
```

Ahora es posible hacer SELECT ... FROM INFO_DEPTOS ...

- Ojo: la tabla INFO_DEPTOS puede contener información NO actualizada
  Si se modifica información en EMPLEADO y/o DEPARTAMENTO, **los cambios NO se reflejarán en la relación** INFO_DEPTOS
  - » Es necesario crear una relación de este tipo mediante una **VISTA**, para que esté siempre contenga los datos más actuales

MODIFICACIÓN DE DATOS: ELIMINACIÓN

- **Orden DELETE**
  - **Elimina tuplas completas** de una única relación
  - **Sólo 1 relación en cláusula** FROM
  - Cláusula **WHERE** para **seleccionar las tuplas** que eliminar
    - Si no hay WHERE, se eliminan **todas** las tuplas
    - La tabla permanece, pero queda vacía
- **PROPAGACIÓN DE ELIMINACIONES**
  - Según acciones especificadas en cada Restricción de Integridad Referencial escritas con DDL en los CREATE TABLE

```
DELETE FROM EMPLEADO WHERE apellido='Bojórquez';      elimina 0 tuplas
DELETE FROM EMPLEADO WHERE nss='123456789' ;          elimina 1 tupla
DELETE FROM EMPLEADO
  WHERE nd IN ( SELECT númerod FROM DEPARTAMENTO
                WHERE nombre='Investigación') ;       elimina 4 tuplas
DELETE FROM EMPLEADO ;                                elimina todas las tuplas
```

MODIFICACIÓN DE DATOS: *ACTUALIZACIÓN*

- **Orden UPDATE**
  - **Modifica valores de atributos** en **una o más tuplas** de una relación
  - Se modifican tuplas de una **única tabla** a la vez
  - **Cláusula SET** especifica **atributos que modificar y nuevos valores**
  - **Cláusula WHERE** para **seleccionar tuplas que actualizar**
    - Si no hay WHERE, se aplica la modificación a **todas** las tuplas

- **PROPAGACIÓN DE MODIFICACIONES**

  Si **cambia** el valor de **clave candidata**, este cambio **se propaga** a valores de **clave ajena** de tuplas de otras relaciones, si así se especificó en las RIs Referencial en la definición de la tabla con CREATE TABLE

Para el proyecto número 10, cambiar el lugar a Belen y el nº del departamento controlador al 5.

```
UPDATE PROYECTO SET lugarp = 'Belen', númd = 5
WHERE numerop=10 ;
```


MODIFICACIÓN DE DATOS: *ACTUALIZACIÓN*

- **MODIFICACIÓN DE VARIAS TUPLAS A LA VEZ EN UPDATE**

Otorgar a todos los empleados del departamento de Investigación un aumento salarial del 10%

```
UPDATE EMPLEADO SET salario = salario*1.1
WHERE nd IN (SELECT númerod
             FROM DEPARTAMENTO
             WHERE nombred='Investigación') ;
```

- **NULL O DEFAULT COMO NUEVO VALOR DE UN ATRIBUTO**

```
UPDATE EMPLEADO SET salario = DEFAULT;

UPDATE EMPLEADO SET nsssuper = NULL WHERE ... ;
```

DEFINICIÓN DE DATOS (DDL) EN SQL2

- **ESQUEMA de Base de Datos Relacional**
  - Agrupa relaciones (tablas) y otros elementos, de una misma aplicación
  - 1as versiones de SQL: todas las tablas dentro del un esquema único y global a todas las aplicaciones que accedían a la BD

- **Orden CREATE SCHEMA**: sentencia de definición/creación de esquemas

  CREATE SCHEMA COMPAÑÍA **AUTHORIZATION** JSILVA **;**

  - **nombre de esquema**: identifica el esquema (COMPAÑÍA)
  - **identificador de autorización**: usuario/cuenta propietario del esquema (JSILVA)
  - Conjunto de **definiciones de cada elemento contenido en el esquema**

- **Elementos del Esquema:**
  Tablas, Vistas, Dominios, autorizaciones, asertos, etc.

DDL: DEFINICIÓN DE TABLAS

- **Orden CREATE TABLE**
  - Define (crea) una relación: **nombre**, atributos y restricciones
  - Para cada **atributo**:
    - nombre, tipo de datos (dominio) y restricciones de atributo
  - **Restricciones** de tabla
    - de clave candidata e integridad de entidad,
    - de integridad referencial, o
    - restricciones de otro tipo

```
CREATE TABLE EMPLEADO (
  nombrep ...
  inic ...
  apellido ...
  nss ...
  fechan ...
  direccion ...
  sexo ...
  salario ...
  nsssuper ...
  nd ...
);
```

DDL: DEFINICIÓN DE TABLAS

· **INDICACIÓN DEL ESQUEMA AL QUE PERTENECE UNA RELACIÓN**

- Esquema Explícito

  CREATE TABLE COMPAÑÍA.EMPLEADO ...

- Esquema Implícito en el contexto

  CREATE TABLE EMPLEADO **...**

· **ORDENAMIENTO DE ATRIBUTOS Y TUPLAS EN SQL2**

- **Atributos ordenados tal como aparecen en** CREATE TABLE
- **Las filas (tuplas) NO están ordenadas**

• Las relaciones creadas con CREATE TABLE son relaciones BASE

- El SGBD las almacena físicamente en algún fichero de la base de datos

DDL: DEFINICIÓN DE TABLAS

· **ESPECIFICACIÓN DEL TIPO DE DATOS DE UN ATRIBUTO**

1) Especificar **directamente** el **TIPO DE DATO** tras el nombre del atributo

```
CREATE TABLE EMPLEADO (
  nombrep VARCHAR(15) ...
... );
```

2) Declarar un DOMINIO y usar su nombre como "tipo de datos"

- Facilita cambio del tipo de datos usado por muchos atributos
- Esquema más comprensible

```
CREATE DOMAIN NOMBRES AS CHAR(15);
...
CREATE TABLE EMPLEADO (
  nombrep NOMBRES ...
... );
```

DDL: DEFINICIÓN DE TABLAS

- **TIPOS DE DATOS**
  - **Numéricos**
    - **Enteros y Reales** **INTEGER, SMALLINT, FLOAT, REAL, DOUBLE PRECISION**
    - **Con formato** **DECIMAL(p,e)** (DEC(p,e), NUMERIC(p,e))
      p: precisión, **e**: escala. Valor por omisión **escala e = 0**
  - **Cadena de caracteres**
    - **Longitud fija** **CHAR(n)** (n: nº caracteres)
    - **Longitud variable** **VARCHAR(n)** (n: máximo nº caracteres)
  - **Cadena de Bits**
    - **Longitud fija** **BIT(n)** (n: nº bits) ,
    - **Longitud variable** **BIT VARYING(n)** n: máx nº bits. Valor por omisión **n = 1**
  - **DATE** (10 posiciones) **= YEAR , MONTH , DAY** (yyyy-mm-dd)
  - **TIME** (8 posiciones) **= HOUR , MINUTE , SECOND** (hh:mm:ss)
    - Sólo permitidas **fechas y horas válidas**
  - **TIMESTAMP** (marca de tiempo) = DATE, TIME, fracciones de segundo y desplazamiento respecto al huso horario estándar (WITH TIME ZONE)
  - **INTERVAL** Valor relativo para incr/decrementar el v.a. de fecha, hora o *timestamp*
    - Se califica con YEAR/MONTH ó DAY/TIME para indicar su naturaleza

DDL: DEFINICIÓN DE DOMINIOS

- **DOMINIOS DE DATOS**

CREATE DOMAIN <nombre dominio> <tipo de datos>
[DEFAULT <valor>]
[<lista de definición de restricciones de dominio>] ;

- El <tipo de datos> ha de ser uno de los proporcionados por el sistema (built-in)
- Valor por defecto: (opcional)
  - Especifica el **valor por omisión** para atributos definidos de este dominio
  - Será asignado a cada columna definida sobre el dominio, si no tiene ya su propia DEFAULT
- Definición de Restricciones de Integridad de Dominio: (opcional)
  - Conjunto de RI que se aplican a cada columna definida sobre el dominio
  - Cada restriccion puede tener un nombre (cláusula CONSTRAINT <nombre_RI>)
  - Ejemplo: enumeración de posibles valores componentes del dominio

CREATE DOMAIN COLOR VARCHAR(8) DEFAULT 'sinColor'
CONSTRAINT color_valido
CHECK (VALUE IN ('rojo', 'amarillo' , 'azul', 'verde', ' sinColor')) ;

DDL: DEFINICIÓN DE TABLAS

· **ESPECIFICACIÓN DE RESTRICCIONES DE ATRIBUTO**

- OPCIÓN DE NULO
  - Indica si un atributo puede contener **NULL**
  - Restricción **NULL o NOT NULL**

```
CREATE TABLE EMPLEADO (
  nombrep VARCHAR(15) NOT NULL, ... );
```

  – La restricción **NOT NULL** es **obligatoria para** atributos de **clave primaria**

- VALOR POR OMISIÓN (o por defecto)
  - Cláusula **DEFAULT <valor>**

```
CREATE TABLE EMPLEADO ( ...
  salario DECIMAL(10,2) NULL DEFAULT 100000 ... );
```

  - Si un atributo no tiene cláusula DEFAULT, su valor por defecto es...
    · El de su dominio, si su tipo de datos es un dominio y éste incluye DEFAULT
    · El NULL, en cualquier otro caso, siempre que el atributo permita NULL

DDL: DEFINICIÓN DE TABLAS

· **ESPECIFICACIÓN DE RESTRICCIONES DE TABLA**

- **PRIMARY KEY (...)** Especifica atributos que componen la **Clave Primaria**
- **UNIQUE (...)** Indica **Claves Alternativas**
- **FOREIGN KEY (...)** Especifica **Claves Externas** (Integridad Referencial)
- **CHECK (...)** Indica **condición** que debe cumplir **toda tupla** de la tabla

```
CREATE TABLE EMPLEADO (
  nombrep   VARCHAR(15)  NOT NULL,
  ...,
  nss       CHAR (9)     NOT NULL,
  nif       CHAR(9)      NOT NULL,
  ...,
  nsssuper  CHAR(9)      NULL,
  nd        INTEGER      NOT NULL,
  PRIMARY KEY ( nss ),
  UNIQUE ( nif ),
  CHECK ( nsssuper <> nss ),
  FOREIGN KEY (nsssuper) REFERENCES EMPLEADO(nss),
  FOREIGN KEY (nd) REFERENCES DEPARTAMENTO(numerod)
  );
```

DDL: DEFINICIÓN DE TABLAS

- **ACCIONES DISPARADAS POR INTEGRIDAD REFERENCIAL**
  - Políticas de mantenimiento de la Integridad Referencial

  ON DELETE <acción> **ON UPDATE** <acción>

  donde <acción> ∈ { **NO ACTION**, **SET NULL, CASCADE, SET DEFAULT** }

- **ESPECIFICACIÓN DE RESTRICCIONES DE TABLA NOMBRADAS**
  - Es opcional **CONSTRAINT** <nombre_RI> <restricción CHECK>
  - Nombres de restricción **únicos** dentro del mismo **esquema**
  - Identifican una restricción, por si después debe ser sustituida por otra

```
CREATE TABLE EMPLEADO (
  ...,
  CONSTRAINT pk_empleado      PRIMARY KEY ( nss ),
  CONSTRAINT nif_unico        UNIQUE ( nif ),
  CONSTRAINT no_auto_ jefe    CHECK ( nsssuper <> nss ),
  CONSTRAINT jefe_emp         FOREIGN KEY (nsssuper) REFERENCES EMPLEADO(nss)
                              ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT dep_emp          FOREIGN KEY (nd) REFERENCES DEPARTAMENTO(numerod)
                              ON DELETE SET NULL ON UPDATE CASCADE
);
```


VISTAS EN SQL2

- **VISTA - VIEW**
  - **Es una Relación derivada de otras relaciones**
  - Las **tablas o vistas** de las que se deriva la vista son sus **Tablas Base**
  - Son **relaciones (tablas) virtuales** (no necesariamente existen en forma física)

    CREATE VIEW <nombre_vista> [<lista_nombres_atributos>]
    AS <consulta_de_definición>
  - La consulta de definición determina el **contenido de la vista**

```
CREATE VIEW EMPLEADO_DEPENDIENTE
AS SELECT nombrep, nombre_dependiente, parentesco
    FROM EMPLEADO, DEPENDIENTE
    WHERE nss = nsse;
```

- **Las vistas pueden utilizarse como...**
  - Mecanismo de simplificación de consultas
  - Mecanismo de seguridad
- **Característica fundamental de las vistas**
  - **Actualización Permanente** (responsable: el SGBD)

  » La vista NO se **crea** cuando se define, sino cuando se **consulta**

ESPECIFICACIÓN DE VISTAS: CREATE VIEW

- Por defecto la vista hereda los nombres de los atributos...
  - ✓... seleccionados desde las tablas base (también llamadas "tablas de definición")
  - ✓... siempre que ningún atributo sea el resultado de una función u operación aritmética

```
CREATE VIEW VISTA_TRABAJA_EN
AS SELECT nombrep, apellido, nombrepr, horas
      FROM EMPLEADO, PROYECTO, TRABAJA_EN
      WHERE nss = nsse AND númp = número ;

CREATE VIEW INFO_DEPTO (nombre_depto, núm_de_emps, sal_total)
AS SELECT nombred, COUNT(*), SUM(salario)
    FROM DEPARTAMENTO, EMPLEADO
    WHERE númerod = nd
    GROUP BY nombred ;
```

- **No tienen ninguna limitación** en operaciones de **Consulta**

  Apellido y nombre de empleados que trabajan en el proyecto llamado "ProductoX"

```
SELECT nombrepr, nombrep, apellido
FROM VISTA_TRABAJA_EN
WHERE nombrepr='ProductoX' ;
```

- **Tienen algunas limitaciones en** operaciones **de Modificación**

MODIFICACIÓN DE VISTAS EN SQL

- Es complicada y puede ser ambigua

  Modificar **vistas definidas sobre múltiples tablas** suele dar problemas:

```
UPDATE VISTA_TRABAJA_EN
SET nombrepr = 'ProductoY'
WHERE apellido = 'Silva' AND nombrep='José' AND nombrepr = 'ProductoX' ;
```

  Puede traducirse a **DOS** posibles **modificaciones** de las **relaciones base**:

  a)
```
UPDATE TRABAJA_EN
    SET númp = (SELECT número FROM PROYECTO WHERE nombrepr = 'ProductoY')
    WHERE nsse = (SELECT nss FROM EMPLEADO
                  WHERE apellido = 'Silva' AND nombrep = 'José')
      AND númp = (SELECT número FROM PROYECTO
                  WHERE nombrepr = 'ProductoX') ;
```

  » Relaciona José Silva con la tupla 'ProductoY' de PROYECTO y no con la de 'ProductoX'

  b)
```
UPDATE PROYECTO SET nombrepr = 'ProductoY'
    WHERE nombrepr = 'ProductoX' ;
```

  » Mismo efecto que **a)** pero modificando el nombrepr en PROYECTO
  ⇒ PERO modifica todas las tuplas de la vista cuyo nombrepr = 'ProductoX' ¡ ☹ !

MODIFICACIÓN DE VISTAS EN SQL

- **Algunas modificaciones de vistas carecen de sentido**

```
UPDATE INFO_DEPTO
SET sal_total = 100000
WHERE nombred='Investigación' ;
```

Pues sal_total se define como 'suma de los salarios individuales de los empleados' y muchas actualizaciones de las tablas base satisfarían esta actualización

- **NO se garantiza que "toda vista sea actualizable"**
- **Una vista sería actualizable**...
  - ✓ **Si implicara** una **única actualización posible** de las **tablas definición**
  - ✓ **Si existieran varias posibles actualizaciones, pero existiera un** procedimiento específico de actualización de las tablas base, tal que...
    - El usuario pudiera elegir el procedimiento, especificándolo en la definición de la vista, o bien
    - El SGBD pudiera elegir el procedimiento, según la actualización más probable

MODIFICACIÓN DE VISTAS EN SQL

En general...

- Una vista con **UNA única tabla de definición**

  Actualizable si los atributos **contienen** la **clave primaria** u otra **candidata** de la **relación base** (se establece una transformación de cada tupla de la vista a una única tupla base)
- Una vista definida sobre **MÚLTIPLES tablas** mediante reuniones

  NO es **actualizable**
- Una vista definida mediante **AGRUPACIÓN y funciones AGREGADAS**

  NO es **actualizable**

- **OPCIÓN DE VERIFICACIÓN DE VISTAS**

  Cláusula WITH CHECK OPTION
  - Al **final de la definición** de toda vista que se vaya a utilizar para modificación
  - Permite al sistema:
    - Comprobar si la vista es actualizable, y
    - Planear una estrategia de ejecución de actualizaciones

IMPLEMENTACIÓN DE VISTAS EN SQL

Dos enfoques:

1. ESTRATEGIA DE **ACTUALIZACIÓN DE CONSULTAS DE DEFINICIÓN**
- **Consulta** sobre la **vista** se traduce a una **Consulta** sobre **Tablas Definición**
- La vista se **rellena de tuplas** a partir de la ejecución de la **consulta**
- Desventaja: **poco eficiente** cuando...
  - la vista se define con **consulta compleja**, con tiempo de ejecución apreciable,
  - y se aplican **muchas consultas** sobre la vista **en poco tiempo**

2. ESTRATEGIA DE **MATERIALIZACIÓN DE VISTAS**
- **1ª Consulta** sobre la vista ⇒ **creación de Tabla Temporal Física**
- Se conserva la tabla para posteriores consultas sobre la vista
- Necesaria estrategia para actualización incremental de la Tabla Temporal tras cualquier modificación de tablas base (⇒ **actualización permanente**)
- Si no se hace referencia a la vista tras un tiempo, el sistema la eliminará (y la recalculará en una consulta futura)

DDL: ALTERACIÓN DE TABLAS

- **MODIFICACION DE LA ESTRUCTURA DE LAS RELACIONES BASE DEL ESQUEMA**
- Posibilidades:
  - Adición y Eliminación de Atributos (columnas)
  - Modificación de la Definición de Atributos
  - Adición y Eliminación de Restricciones de Tabla

ALTER TABLE <nombre_relacion> ... ;

DDL: ALTERACIÓN DE TABLAS

- **ADICIÓN DE ATRIBUTO (COLUMNA)**

ALTER TABLE <nombre_relacion> ADD <definición_atributo> ;

Añadir a la relación EMPLEADO del esquema COMPAÑÍA, un atributo para contener el puesto de trabajo

ALTER TABLE COMPAÑÍA.EMPLEADO **ADD** puesto VARCHAR(12);

– Todas las tuplas de EMPLEADO tendrán puesto a NULL
⇒ NO está permitida la restricción NOT NULL en la definición del nuevo atributo
(si es necesaria, podrá establecerse después)

– Para introducir un **valor** para **puesto**, en cada **tupla existente** en EMPLEADO:

(a) Especificar la cláusula DEFAULT al añadir el atributo:
ALTER TABLE EMPLEADO ADD puesto VARCHAR(12) **DEFAULT "aprendiz"**;

(b) Utilizar después una orden UPDATE *(se verá)*

DDL: ALTERACIÓN DE TABLAS

- **ELIMINACIÓN DE UN ATRIBUTO (COLUMNA)**

ALTER TABLE <nombre_relacion> DROP <nombre_atributo> <opcion> ;

<opcion> puede ser...

CASCADE: elimina el atributo y toda restricción o vista que le haga referencia

RESTRICT: sólo elimina el atributo si ninguna vista ni restricción le referencia

Eliminación del atributo dirección de la relación EMPLEADO del esquema COMPAÑÍA

ALTER TABLE COMPAÑÍA.EMPLEADO DROP dirección CASCADE;

DDL: ALTERACIÓN DE TABLAS

- **MODIFICACION DE LA DEFINICIÓN DE UN ATRIBUTO (COLUMNA)**

  ALTER TABLE <nombre_relacion> ALTER <nombre_atributo> <accion> ;

  Modificación de la cláusula por omisión

  – **Eliminación** de la cláusula DEFAULT existente

  ALTER TABLE COMPAÑÍA.DEPARTAMENTO **ALTER** nssgte **DROP DEFAULT**;

  – **Definición** de una **nueva** cláusula por omisión

  ALTER TABLE DEPARTAMENTO **ALTER** nssgte **SET DEFAULT "123456789"**;

DDL: ALTERACIÓN DE TABLAS

– La restricción de tabla que se desea modificar debe tener un nombre!!

- **ELIMINACIÓN DE UNA RESTRICCIÓN DE TABLA**

  ALTER TABLE <nom_relacion> DROP CONSTRAINT <nombre_RI> <opcion>;

  ALTER TABLE COMPAÑÍA.EMPLEADO DROP CONSTRAINT jefe_emp CASCADE;

- **ADICIÓN DE UNA RESTRICCIÓN DE TABLA**

  ALTER TABLE <nom_relacion> ADD CONSTRAINT <nombre_RI> <defin_RI>;

  ALTER TABLE EMPLEADO ADD CONSTRAINT salario_ok CHECK (salario > 0);

DDL: ALTERACIÓN DE DOMINIOS

- Orden **ALTER DOMAIN <nombre_dominio> <acción>;**

  - Eliminación y Reemplazo de una definición por defecto

```
ALTER DOMAIN <nom_dominio> DROP DEFAULT;
ALTER DOMAIN <nom_dominio> SET DEFAULT <valor>;
```

  - Eliminación y/o especificación de nuevas restricciones de dominio

```
ALTER DOMAIN <nom_dominio> DROP CONSTRAINT <nom_RI_dominio>;
ALTER DOMAIN <nom_dominio>
                    ADD [CONSTRAINT <nom_RI>] <restricción_CHECK>;
```


DDL: ELIMINACIÓN DE ELEMENTOS DEL ESQUEMA

- **ELIMINACIÓN DE UNA VISTA. Orden DROP VIEW**
  - Destruye una relación derivada, junto con su definición en el catálogo

  DROP VIEW <nombre_vista> ;

- **ELIMINACIÓN DE UN DOMINIO. Orden DROP DOMAIN**
  - Destruye un dominio de datos, junto con su definición en el catálogo

  DROP DOMAIN <nombre_dominio> <opción> ;

  <opcion> puede ser...
  - RESTRICT: sólo destruye el dominio si no hay ningún atributo definido sobre él
  - CASCADE: se destruye el dominio y, todo atributo definido sobre él, pasa a tener el tipo de datos sobre el que se había definido el dominio

DDL: ELIMINACIÓN DE ELEMENTOS DEL ESQUEMA

- **ELIMINACIÓN DE UNA RELACIÓN. Orden DROP TABLE**
  - Destruye una relación base junto con su definición en el catálogo

  DROP TABLE <nombre_relación> <opcion>;

  <opcion> puede ser...
    - RESTRICT: Destruye la tabla sólo si no se le hace referencia desde ninguna otra relación (clave ajena), ni es tabla base de una vista
    - CASCADE: Elimina la tabla junto con restricciones y vistas que la referencian

- **ELIMINACIÓN DE UN ESQUEMA. Orden DROP SCHEMA**
  - Destruye un esquema de BD junto con su definición en el catálogo

  DROP SCHEMA <nombre_esquema> <opcion>;

  <opcion> puede ser...
    - RESTRICT: Destruye el esquema sólo si no contiene ningún elemento
    - CASCADE: Elimina el esquema, y los tablas, dominios y demás elementos contenidos en el esquema

DDL: CATÁLOGO RELACIONAL

- ***CATÁLOGO* de una Base de Datos Relacional** (nuevo concepto en SQL2)
  - ***Colección nombrada de esquemas*** en un entorno SQL
  - **Existe un esquema especial, INFORMATION_SCHEMA,** con **datos sobre** la definición de todos los **elementos de todos** los **esquemas existentes en el catálogo**
  - Sólo pueden definirse **Restricciones de Integridad** (RI Referencial, etc.) **entre relaciones** que existan **en esquemas del mismo catálogo**
  - **Es posible compartir elementos** (definiciones de dominio, etc.) **entre esquemas del mismo catálogo**