Introducción al SQL

SQL (Structured Query Language) standard de hecho para RDBMS. Incluye DDL, DML y DCL.

SQL es un lenguaje declarativo (no procedural).

*** No especifica la secuencia de operaciones necesarias para obtener el resultado.**

El modelo de datos se basa estrictamente en el modelo relacional.

Las relaciones son representadas por tablas.

Reseña histórica

La estandarización comenzó en 1986.

SQL-92, definido en 1992 por ISO (International Standard Organization) y ANSI (American National Standard Institute)

SQL-1999, hace a SQL un lenguaje computacionalmente completo para objetos persistentes.

Reseña histórica

En la actualidad cada sistema tiene su propio dialecto:

* **Soporta la mayor parte de SQL-92**
* **Tiene elementos de SQL-1999**
* **Tiene características no standard**

Se verá el subconjunto más común.

Una base de datos simple:

Est_Mat_Exam

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	20/02/86	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

Cursos

N_Cur	Materia	Docente	Anio
292	Informática I	N. Berillo	1
511	Informática I	J. Calusso	1
435	Física II	R. Logiz	2

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F

Consulta de tablas SELECT

Estructura básica:

SELECT . . .

FROM . . .

WHERE . . .

Qué se quiere

Dónde está almacenado

Condición para la salida

Mostrar todos los datos

Encontrar todos los cursos de la base.
(Esto es equivalente a ver la instancia de la relación.)

Especificar la relación:

SELECT *

FROM Cursos

Todos los atributos

Qué tabla

WHERE
no se utiliza

Mostrar sólo algunos datos

Listar sólo los atributos que nos interesen.

```
SELECT   N_Cur, Materia, Anio
FROM  Cursos
```

N_Cur	Materia	Anio
292	Informática I	1
511	Informática I	1
435	Física II	2

La cláusula SELECT reporta el elenco de atributos deseados

Mostrar sólo algunos datos

La secuencia de columnas es elección del usuario, según se indique en SELECT.

SELECT Materia, Anio, N_Cur

FROM Cursos

Materia	Anio	N_Cur
Informática I	1	292
Informática I	1	511
Física II	2	435

Filas duplicadas

Si las columnas seleccionadas no contienen la clave, pueden aparecer filas duplicadas.

```
SELECT  Materia
FROM  Cursos
```

Materia
Informática I
Informática I
Física II

Esto se puede evitar utilizando la opción “DISTINCT”.

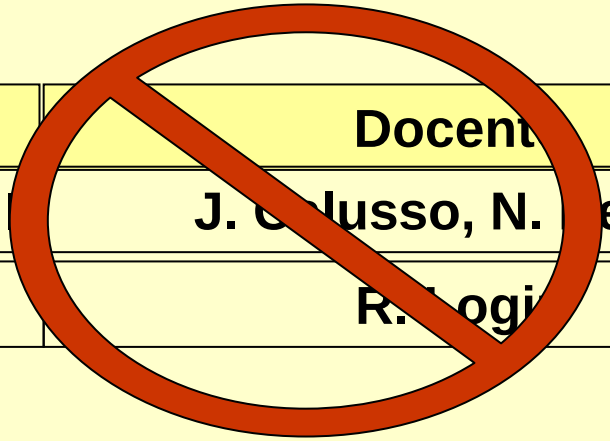
```
SELECT DISTINCT Materia
FROM  Cursos
```

Materia
Informática I
Física II

Filas duplicadas

No hay forma de impedir la duplicación parcial en Materia.

Materia	Docent
Informática I	J. C lusso, N. erillo
Física II	R. Logi



Esto no sería 1FN

Renombrar columnas

Los alias pueden ayudar a dar claridad a la lectura de resultados.

```
SELECT  Nombre AS Materia,
        Profesor AS Docente

FROM  Cursos
```

Nombre	Profesor
Informática I	N. Berillo
Informática I	J. Calusso
Física II	R. Logiz

Calcular expresiones

Antes de la presentación, es posible realizar algunos procesos.

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	20/02/86	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

```
SELECT Legajo,
       CONCAT( Nombre, ' ',Apellido)

FROM  Alumnos
```

Calcular expresiones

Legajo	
123456	Juan Pérz
135790	Ana Muro
159732	Luis Báez
175398	Nora Lorenz

```
SELECT Legajo,
       CONCAT( Nombre, ' ',Apellido)

FROM  Alumnos
```

Calcular expresiones

Antes de la presentación, es posible realizar algunos procesos.

Z

A	B
3	7
2	10
9	6
14	8

10
12
15
22

SELECT A + B

FROM Z

Calcular expresiones

Para darle a la columna resultante un nombre distinto del que se le asigna por defecto:

Z

A	B
3	7
2	10
9	6
14	8

Total
10
12
15
22

```
SELECT A + B
            as “Total”
FROM  Z
```

Ver un subconjunto de t-uplas

El Profesor Logiz quiere ver los resultados de los exámenes del curso que dicta (435):

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F
168896	435	7	P

Tendrá que especificar algo en su consulta...

Cláusula WHERE: condiciones

Expresar una condición lógica: es decir, una expresión booleana que sea cierta para un subconjunto de t-uplas.

En este caso, la expresión será:

N_Cur = 435

Legajo	N_Cur	Nota	Tipo
123456	435	8	F
168896	435	7	P

```
SELECT *
FROM Evaluac
WHERE N_Cur = 435
```

Resumen

Elegir las t-uplas que interesen (WHERE).

Proyectar los atributos que interesen (SELECT).

Legajo	Nota	Tipo
123456	8	F
168896	7	P

```
SELECT  Legajo, Nota,Tipo
FROM  Evaluac
WHERE N_Cur = 435
```

Condiciones compuestas

La “regla” para seleccionar los datos que interesan puede ser más complicada.

Por ejemplo: El Profesor Logiz desea saber qué estudiantes obtuvieron más de 7.

El resultado deseado es:

Legajo	Nota	Tipo
123456	8	F

Condiciones compuestas

El resultado involucra dos condiciones:

Una con respecto al número de curso (N_Cur=435).

Una con respecto a la nota (Nota>7).

WHERE N_Cur = 435

??? **Nota > 7**

¿Cómo combinarlas?

Condiciones compuestas

Si estamos interesados en t-uplas que satisfagan ambas condiciones, el operador es AND:

```
SELECT  *
FROM Evaluac
WHERE N_Cur = 435
AND   Nota > 7
```

Legajo	N_Cur	Nota	Tipo
123456	435	8	F

Condiciones compuestas

Si estamos interesados en t-uplas que satisfagan al menos una condición, el operador es OR:

```
SELECT  *
FROM Evaluac
WHERE N_Cur = 435
OR    Nota > 7
```

Legajo	N_Cur	Nota	Tipo
123456	435	8	F
168896	435	7	P

Ejercicio

Dada la estructura:

Pedidos(Num_Ped, Fecha, Cliente, Monto, Tasa_IVA)

Se requiere:

La lista de pedidos a partir de 2005, del cliente Negri, mostrando el número de pedido, la fecha y el importe (más IVA).

Operadores

se usa junto con comodines:

LIKE , busca cadenas de caracteres, de acuerdo con algún patrón.

Equivale a:

_	**“cualquier carácter individual”**
%	**“cualquier cadena”**

Operadores

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	20/02/86	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

Buscar los Legajos y Apellidos de los alumnos cuyos correos electrónicos tengan una “*a*” en la tercera posición y terminen en “*.ar*” .

Operadores

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	20/02/86	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

```
SELECT  Legajo, Apellido
FROM Alumnos
WHERE  Correo_el  LIKE '_ _a%.ar'
```

Operadores

Legajo	Apellido
159732	Báez

```
SELECT  Legajo, Apellido
FROM Alumnos
WHERE  Correo_el  LIKE '_ _a%.ar'
```

Operadores

BETWEEN, es verdadera cuando un atributo pertenece al intervalo cerrado.

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F

Buscar los exámenes con nota entre 7 y 9:

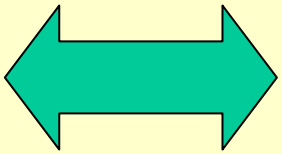
Operadores

BETWEEN, es verdadera cuando un atributo pertenece al intervalo cerrado.

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F

```
SELECT  *
FROM Evaluac
WHERE  Nota
BETWEEN 7 AND 9
```

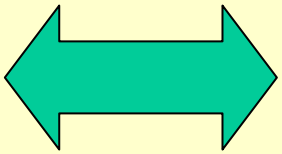


```
SELECT  *
FROM Evaluac
WHERE  Nota>=7
AND Nota<=9
```

Operadores

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
123456	435	8	F

```
SELECT  *
FROM Evaluac
WHERE  Nota
BETWEEN 7 AND 9
```



```
SELECT  *
FROM Evaluac
WHERE  Nota>=7
AND Nota<=9
```

Operadores

IN, es verdadera cuando el valor de un atributo pertenece a un conjunto de valores.

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F
144456	916	8	P

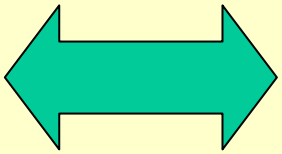
Buscar los exámenes de los cursos 435 y 916:

Operadores

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	292	10	P
159732	292	6	F
123456	435	8	F
144456	916	8	P

```
SELECT  *
FROM Evaluac
WHERE  N_Cur IN
(435,916)
```

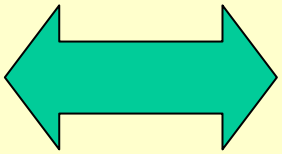


```
SELECT  *
FROM Evaluac
WHERE  N_Cur=435
OR N_Cur=916
```

Operadores

Legajo	N_Cur	Nota	Tipo
123456	435	8	F
144456	916	8	P

```
SELECT *
FROM Evaluac
WHERE Nota IN
(435,916)
```



```
SELECT *
FROM Evaluac
WHERE N_Cur=435
OR N_Cur=916
```

Null values

Los NULL VALUES pueden dar origen a resultados extraños:

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	20/02/86	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

Sea un cambio en los datos: Suponer que hay un NULL VALUE en la fecha de nacimiento de Ana Muro.

Null values

Los NULL VALUES pueden dar origen a resultados extraños:

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	NULL	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

Legajo	Apellido	Nomb	FeNac	Correo_el
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

```
SELECT *
FROM Alumnos
WHERE FeNac > 31/12/85
```

Null values

Ninguna condición acerca de la fecha de nacimiento (FeNac) puede seleccionar la fila de Ana Muro.

Buscar Null values

Cualquier condición falla siempre con NULL VALUES, salvo que se la explore explícitamente con el operador IS.

Buscar Null values

```
SELECT *
FROM Alumnos
WHERE FeNac IS NULL
```

Legajo	Apellido	Nomb	FeNac	Correo_el
135790	Muro	Ana	20/02/86	amu@utn.edu.ar

Lo opuesto es:

NOT (FeNac IS NULL), verdadera cuando los valores son no nulos. También expresado como: FeNac IS NOT NULL

Null values y condiciones compuestas

ATENCIÓN:

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	NULL
135790	Muro	Ana	NULL	amu@utn.edu.ar
159732	Báez	Luis	26/04/88	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@gmail.com

```
SELECT  Legajo
FROM Alumnos
WHERE  FeNac <= 31/12/87
AND Correo_el LIKE  '%.com'
```

Legajo
17539-8

Null values y condiciones compuestas

ATENCIÓN:

Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	NULL
135790	Muro	Ana	NULL	amu@utn.edu.ar
159732	Báez	Luis	26/04/88	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@gmail.com

```
SELECT  Legajo
FROM Alumnos
WHERE  FeNac <= 31/12/87
OR Correo_el LIKE  '%.com'
```

Legajo
123456
175398

Ordenar resultados

El resultado de un SELECT trae filas en un orden impredecible.

La cláusula ORDER BY produce una salida ordenada.

Ordenar resultados

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F

```
SELECT  *
FROM EVALUAC
ORDER BY  Tipo,
          Nota DESC
```

Legajo	N_Cur	Nota	Tipo
123456	435	8	F
123456	292	7	F
159732	292	6	F
135790	511	10	P

Cambiar el contenido de la base

Operaciones habituales:

Agregar t-uplas a la base de datos.
Eliminar t-uplas de la base de datos.
Modificar t-uplas de la base de datos.

Cambiar el contenido de la base

INSERT puede usar el resultado de una consulta para agregar varias filas simultáneamente.

DELETE y **UPDATE** pueden usar condiciones para especificar cuáles t-uplas se han de borrar o modificar.

Insertar una fila

Cursos

N_Cur	Materia	Docente	Anio
292	Informática I	N. Berillo	1
511	Informática I	J. Calusso	1
435	Física II	R. Logiz	2

Especificando valores para cada atributo:

```
INSERT INTO
Cursos(N_Cur,Docente,Materia,Anio).
VALUES (230,Durza,Álgebra,1)
```

Insertar una fila

Cursos

C_Mat	Materia	Docente	Anio
292	Informática I	N. Berillo	1
511	Informática I	J. Calusso	1
435	Física II	R. Logiz	2

Omitiendo el listado de atributos (Se usa implícitamente el orden utilizado en la definición de atributos):

```
INSERT INTO Cursos
VALUES (230, Álgebra,Durza,1)
```

Insertar varias filas

A veces, las filas se ext
y se insertan en otra.

Esto puede hacerse en u co paso:

Las estructuras de entrada y salida tienen que ser compatibles

```
INSERT INTO
EstudSinCorr(Legajo pellido,Nombre)
SELECT (Legajo,Apellido,Nomb)
FROM Alumnos
WHERE  Correo_el IS NULL
```

Borrar filas

Se usa una condición para especificar las filas a eliminar.

```
DELETE FROM Cursos
WHERE Docente = 'J. Logiz'
```

Atención

```
DELETE FROM Cursos
```

Borra todo

Borrar filas

ATENCIÓN:

Restricción de integridad referencial.
Se podría violar al borrar filas.

Modificar t-uplas

Mediante una condición se especifica qué filas han de ser borradas.

Los nuevos valores se especifican mediante expresiones.

```
UPDATE Cursos
SET Docente= 'M. Duren', Anio=1
WHERE  Docente= 'J. Logiz'
```

Modificar t-uplas

Mediante una condición se especifica qué filas han de ser borradas.

Los nuevos valores se especifican mediante expresiones.

```
UPDATE Empleados
SET S_Basico= 1,15 * S_Basico
WHERE  Cargo='Gerente'
```

DDL Definición de tablas

CREATE TABLE define la estructura de una tabla y crea una instancia vacía.

Para cada atributo se especifica:

- *** Nombre y dominio**
- *** Valor opcional por defecto**
- *** Restricciones opcionales**

Opcionalmente, se especifican restricciones a nivel de tabla.

DDL Definición de tablas Ejemplo

```
CREATE TABLE Alumnos  (

Legajo   CHAR (6)  PRIMARY KEY,     {Clave principal}
DNI  CHAR (8) UNIQUE NOT NULL, {Clave secundaria}
Apellido  VARCHAR (25)  NOT NULL,
Nomb VARCHAR (20)  NOT NULL,
FeNac  DATE,
Correo_el  VARCHAR (50)    ) ;
```

DDL Definición de tablas Ejemplo

```
CREATE TABLE Cursos  (

N_Cur   INT    PRIMARY KEY,    {Clave principal}
Materia  CHAR (20)  NOT NULL,
Docente  VARCHAR (30),
Anio INT DEFAULT 1 CHECK(Anio>0)    ) ;
```

DDL Definición de tablas Ejemplo

```
CREATE TABLE Evaluac  (
Legajo CHAR (6) NOT NULL REFERENCES
                                    Alumnos,
N_Cur  INT  NOT NULL REFERENCES  Cursos,
Nota  INT  NOT NULL  CHECK (Nota BETWEEN
                                    0 AND 10),
Tipo  CHAR (1)  NOT NULL  CHECK( Tipo
                                    IN('F','P')) ,
PRIMARY KEY(Legajo,N_Cur)  ) ;
```

NULL VALUES y DEFAULT

NOT NULL Excluye la posibilidad de NULL VALUES en un atributo.

Los valores por defecto (DEFAULT) se utilizarán si, al momento de la inserción de una t-upla no se suministra el valor para algún atributo.

```
Correo_el  VARCHAR(50)
DEFAULT'nn@utn.edu.ar'
INSERT INTO
Alumnos(Legajo,Apellido,Nomb,FeNac)
VALUES(234567,'Bertold','Mario',29/09/84)
```

CLAVES

La restricción UNIQUE especifica una clave alternativa.

DNI CHAR(8) UNIQUE

O, en el caso de claves múltiples:

UNIQUE(Apellido,Nombre)

La especificación:

UNIQUE(Apellido)
UNIQUE(Nombre)

sería mucho más restrictiva...

CLAVE PRINCIPAL

SIMPLE

A continuación de la definición del atributo:

Legajo CHAR(7) PRIMARY KEY

MÚLTIPLE

A nivel de tabla:

PRIMARY KEY(Apellido,Nomb)

CLAVE PRINCIPAL

Observaciones:

La clave principal puede ser omitida (pero muchos DBMS emiten un mensaje de advertencia)

Como máximo una clave principal por tabla.

Los componentes de la clave principal son implícitamente NOT NULL, en la mayoría de los DBMS.

CLAVE EXTERNA Restricciones

Se especifica la clave externa y la tabla a la que se hace referencia.

A nivel de atributo:

```
N_Cur  INT  REFERENCES
       Cursos(N_Cur)
```

CLAVE EXTERNA Restricciones

Se especifica la clave externa y la tabla a la que se hace referencia.

A nivel de tabla:

FOREIGN KEY (N_Cur) REFERENCES Cursos(N_Cur)

Evaluac es la tabla referenciante.
Cursos es la tabla de destino.

Las columnas a las que se hace referencia tienen que ser claves (No necesariamente primaria).

Restricciones genéricas

CHECK permite especificar restricciones genéricas, utilizando todo el poder expresivo de SQL..

CHECK (Condición)

Restricciones genéricas

Se verifica al insertar o modificar una t-upla.

Sueldo INT CHECK (Sueldo>0),

NULL VALUES: no son detectados como violación.

Restricciones genéricas

CHECK a nivel de tabla, permite expresiones multi-atributo.

CHECK ((Nota>=7)) OR (Tipo='P'))

Restricciones con nombre

Útil para interpretar los mensajes del DBMS en caso de violaciones.

```
Nota  INT NOT NULL
CONSTRAINT Notaval
CHECK (Nota BETWEEN 0 AND 10),
```

```
CONSTRAINT ClavextCursos
FOREIGN KEY(N_Cur)
REFERENCES Cursos(N_Cur)
```

QUERY Consulta

Hasta el momento estamos en condiciones de:

*** Definir la estructura DB, con restricciones.**

*** Insertar, modificar y borrar datos.**

*** Escribir consultas a una tabla por vez.**

Pero un principio fundamental del modelo relacional es la distribución de información relacionada en distintas tablas.

QUERY

¿Qué hacer si se quiere los Apellidos y Nombres de los alumnos del curso del profesor Berillo que aprobaron el final de Informática I?

¿Qué hacer si se quiere saber cuántos alumnos de cada profesor se presentaron al final?

QUERY

Afortunadamente la primera búsqueda produjo un solo resultado.

Ejemplo:

Se quiere … urso del profes… …mática I.

```
SELECT N_Cur
FROM Cursos
WHERE  Materia = “Informática I”
AND Docente = “Berillo”
```

N_Cur
292

```
SELECT Legajo
FROM Evaluac
WHERE  Curso = 292
```

Legajo
123456
159732

QUERY a

¿Qué ocurre si rindió 20 exámenes?

Poco práctico.

Ejemplo:

Se quiere cono

exámenes aprobad

```
SELECT N_Cur
FROM Evaluac
WHERE  Legajo = '123
```

N_Cur
292
435

```
SELECT Docente
FROM Cursos
WHERE  N_Cur IN (292,435)
```

Docente
N. Berillo
R. Logiz

QUERY sobre varias tablas

Otro problema:

Se quiere una lista con la estructura indicada a continuación:

Legajo	Apellido	Nomb	N_Cur	Nota	Tipo

Imposible obtenerla accediendo una tabla por vez.

QUERY sobre varias tablas

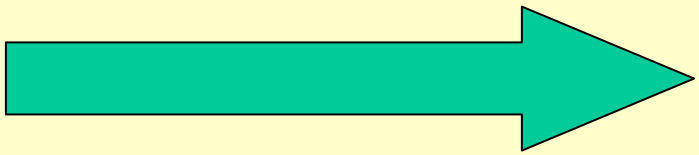
Alumnos

Legajo	Apellido	Nomb	FeNac	Correo_el
123456	Pérez	Juan	12/10/85	jper@utn.edu.ar
135790	Muro	Ana	20/02/86	amu@utn.edu.ar
159732	Báez	Luis	26/04/85	lbae@utn.edu.ar
175398	Lorenz	Nora	21/08/87	hlor@utn.edu.ar

Evaluac

Legajo	N_Cur	Nota	Tipo
123456	292	7	F
135790	511	10	P
159732	292	6	F
123456	435	8	F

El resultado sería:



QUERY sobre varias tablas

Legajo	Apellido	Nomb	N_Cur	Nota	Tipo
123456	Pérez	Juan	292	7	F
135790	Muro	Ana	511	10	P
159732	Báez	Luis	292	6	F
123456	Pérez	Juan	435	8	F

¿Qué pasos serían necesarios para realizar “a mano” la tarea que produzca el resultado deseado?

QUERY sobre varias tablas

Pasos:

1. **Considerar las tablas Evaluac y Alumnos.**
2. **“Aparear” las t-uplas de Evaluac con las de Alumnos teniendo en cuenta el valor de Legajo.**
3. **Considerar sólo las columnas de interés.**

QUERY sobre varias tablas

Los pasos 1 y 3 son similares al caso de tabla única.

Para el paso 2 es necesario hacer explícita la condición de apareo “JOIN”.

QUERY sobre varias tablas

La condición, expresada en palabras:

Aparear t-uplas de Evaluac con t-uplas de Alumnos si tienen el mismo valor en Legajo.

Legajo = Legajo

No es útil

Ambas referencias de atributo son ambiguas (¿Qué tabla?)

QUERY Referencia a tributos

Cuando se trabaja con tablas que contienen atributos con el mismo nombre, se antepone el nombre de la tabla al del atributo:

Evaluac.Legajo = Alumnos.Legajo

Esto siempre se puede hacer, aunque no sea estrictamente necesario.

QUERY Alias para nombres de tablas

A veces las tablas tienen nombres muy largos, o complejos.

Es posible agregar alias a la cláusula FROM:

```
SELECT . . .

FROM Evaluac  E

WHERE  E.Legajo . . .
```

QUERY sobre varias tablas

1 Considerar las tablas Evaluac y Alumnos

FROM Evaluac E, Alumnos A

2 Aparear cada t-upla de Evaluac con la correspondiente t-upla de Alumnos, utilizando Legajo

WHERE E.Legajo = A.Legajo

3 Hacer la proyección de los atributos que se desean

SELECT E.Legajo, A.Apellido, A.Nomb,
E.N_Cur, E.Nota, E.Tipo

QUERY sobre varias tablas

Reuniendo todo:

```
SELECT E.Legajo, A.Apellido, A.Nomb,
                 E.N_Cur, E.Nota, E.Tipo
FROM Evaluac E, Alumnos A
WHERE E.Legajo = A.Legajo
```

QUERY sobre varias tablas

Ejemplo:

Nombre de los docentes de los cursos aprobados por el estudiante de Legajo 123456.

```
SELECT C.Docente

FROM Evaluac E, Cursos C

WHERE E.N_Cur = C.N_Cur
AND   E.Legajo = '123456'
```

JOIN explícito

“Juntar” tablas en la cláusula FROM

```
SELECT C.Docente
FROM Cursos C  JOIN Evaluac  E  ON
(C:N_Cur = E.N_Cur)
WHERE  E.Legajo = ‘123456’
```

QUERY . . . con más tablas

Ejemplo:

Nombre de los docentes de los cursos aprobados por el estudiante Juan Pérez.

3 tablas => 2 JOIN

Los JOIN pueden ser fácilmente generalizados para el caso de tablas múltiples.

QUERY . . . con más tablas

Ejemplo:

Nombre de los docentes de los cursos aprobados por el estudiante Juan Pérez.

```
SELECT C.Docente

FROM Evaluac E, Cursos C, Alumnos A

WHERE   E.Legajo = A.Legajo
  AND   A.Apellido = 'Pérez'
  AND   A.Nomb = 'Juan'
```

JOIN sobre sí misma

A veces, se realiza el JOIN de una tabla consigo misma.
Es habitual para las tablas derivadas de relaciones cíclicas.

Padres P1

Padre	Hijo
Silvia	Ana
Lucas	Ana
Pedro	Lucas
María	Silvia
Elisa	Lucas
Luis	Silvia

Padres P2

Padre	Hijo
Silvia	Ana
Lucas	Ana
Pedro	Lucas
María	Silvia
Elisa	Lucas
Luis	Silvia

¿Los abuelos de Ana?

Abuelos
Pedro
María
Elisa
Luis

QUERY Resumen

Los QUERY sobre múltiples tablas requieren condiciones de JOIN para especificar cómo se aparearán las t-uplas..

Cuando los nombres de columna son iguales, es necesario referirse a ellos con el formato extendido, anteponiendo el nombre de la tabla.

Resultados con cálculo

Se ha visto cómo extraer información de t-uplas individuales (en algunos casos con JOIN).

¿Qué hacer cuando se necesita información acerca de grupos de t-uplas?

Cantidad de exámenes rendidos por el alumno de Legajo 123456

Nota promedio de los exámenes de primer año.

Resultados con cálculo

SQL tiene algunas herramientas para eso:

*** Funciones de agregación.**

*** Cláusula de agrupamiento:**
GROUP BY

Nueva DB para ejemplos:

Emplead

NEmp	Apellido	Suc	Posic	Sueldo
E001	López	1	ASist	2000
E002	Buno	2	AFun	1500
E003	Baer	1	Progr	1000
E004	Vargas	3	Progr	1000
E005	Pérez	2	ASist	2500
E006	Mergui	1	AFun	1100
E007	Rest	1	Progr	1000
E008	Daub	2	Progr	1200

Sucurs

Suc	Jefe	Ciudad
1	Forcas	Buenos Aires
2	Mateos	Bahía Blanca
3	Lorenzi	Buenos Aires

Funciones de agregación (columna)

MIN → mínimo
MAX → máximo
SUM → suma
AVG → media aritmética
STDEV → desviación standard
VARIANCE → varianza
COUNT → contador

Funciones de agregación

Ejemplo:

```
SELECT SUM(Sueldo) AS TotSuel
FROM Emplead
WHERE Suc='1'
```

TotSuel
5100

Funciones de agregación

Una función de agregación puede tener como argumento cualquier expresión válida en la lista de selección (pero no otra función de agregación)

```
SELECT SUM(Sueldo*12) AS SuelAnu
FROM Emplead
WHERE Suc='1'
```

SuelAnu
61200

Funciones de agregación

Todas las funciones, salvo COUNT, ignoran los NULL VALUES.

El resultado es NULL si todos los valores son NULL.

La opción DISTINCT considera sólo los valores distintos.

```
SELECT SUM(DISTINCT Sueldo)
FROM Emplead
WHERE Suc='1'
```

4100

COUNT y NULL VALUES

COUNT (*) Cuenta cantidad de t-uplas en el resultado.

NEmp	. . .	Suc	Sueldo
E001	. . .	1	2000
E002	. . .	2	1500
E003	. . .	1	1000
E004	. . .	3	NULL
E005	. . .	2	2500
E006	. . .	1	NULL
E007	. . .	1	1000
E008	. . .	2	1200

```
SELECT COUNT(*)
AS CantEmpS1

FROM Emplead

WHERE Suc='1'
```

CantEmpS1
4

COUNT y NULL VALUES

Una especificación de columna dentro del COUNT (*) hace que cambie el comportamiento: las t-uplas con NULL en esas columnas son ignoradas.

NEmp	. . .	Suc	Sueldo
E001	. . .	1	2000
E002	. . .	2	1500
E003	. . .	1	1000
E004	. . .	3	NULL
E005	. . .	2	2500
E006	. . .	1	NULL
E007	. . .	1	1000
E008	. . .	2	1200

```
SELECT
COUNT(Sueldo)
AS CaEmSue1

FROM Emplead

WHERE Suc='1'
```

CaEmSue1
3

SELECT y funciones de agregación

Las funciones de agregación no pueden ser utilizadas con expresiones que contienen nombres de atributos.

SELECT Apellido, MIN(Sueldo)

FROM Empleado

¿Qué apellido aparecería con el mínimo sueldo?

SELECT y funciones de agregación

Las funciones de agregación devuelven un único valor, mientras que las referencias a columnas, habitualmente devuelven un conjunto de valores (entre los que puede haber elementos repetidos).

```
SELECT MAX(Sueldo) , MIN(Sueldo)
FROM Emplead
WHERE Suc='1'
```

Correcto

GROUP BY y funciones de agregación

Las funciones de agregación sintetizan valores de todas las t-uplas que satisfacen la condición WHERE.

A veces esos valores son dados por "grupos homogéneos" (por ej. Empleados de la misma sucursal)

La cláusula GROUP BY permite la definición de tales grupos y especifica una o más columnas: las t-uplas se agrupan sobre la base de los valores de las columnas de agrupación.

GROUP BY y funciones de agregación

```
SELECT Suc , COUNT(*) AS CanProg
FROM Emplead
WHERE Posic='Progr'
GROUP BY  Suc
```

Suc	CanProg
1	2
2	1
3	1

La lista del SELECT puede incluir las columnas agrupadas, pero no otras.

Cómo trabaja GROUP BY

NEmp	Apellido	Suc	Posic	Sueldo
E003	Baer	1	Progr	1000
E004	Vargas	3	Progr	1000
E007	Rest	1	Progr	1000
E008	Daub	2	Progr	1200

NEmp	Apellido	Suc	Posic	Sueldo
E003	Baer	1	Progr	1000
E007	Rest	1	Progr	1000
E008	Daub	2	Progr	1200
E004	Vargas	3	Progr	1000

Se buscan las t-uplas que cumplen la cláusula WHERE...

...se agrupa por la columna indicada por GROUP BY ...

Cómo trabaja GROUP BY

Suc	CanProg
1	2
2	1
3	1

...La función de agregación se aplica para cada grupo.

GROUP BY

Ejemplo 1:

Para cada sucursal de Buenos Aires, encontrar el sueldo promedio.

```
SELECT E.Suc, AVG(Sueldo) AS PromSue
FROM Emplead E, Sucurs S
WHERE S.Ciudad='Buenos Aires'  AND
      S.Suc=E.Suc
GROUP BY  E.Suc
```

Suc	PromSue
1	1275
3	1000

GROUP BY

Ejemplo 2:

Para cada posición y sucursal de Buenos Aires, encontrar el sueldo promedio.

Suc	Posic	Sueldo
1	ASist	2000
1	AFun	1100
1	Progr	1000
3	Progr	1000

```
SELECT E.Suc, E.Posic,
AVG(Sueldo) AS PromSue

FROM Emplead E, Sucurs S

WHERE S.Ciudad='Buenos Aires'
    AND  S.Suc=E.Suc

GROUP BY  E.Suc, E.Posic
```

Agrupamiento y proyección

Un SELECT con agrupamiento por columna, produce el mismo resultado que la eliminación de duplicados con DISTINCT.

```
SELECT Suc
FROM Sucurs
GROUP BY  Suc
```

Suc
1
2
3

Equivale a:

```
SELECT DISTINCT Suc
FROM Sucurs
```

Agrupamiento y proyección

Los agrupamientos pueden ser seleccionados sobre la base de sus propiedades “de conjunto”, es decir, los valores de las funciones de agregación.

```
SELECT Suc
COUNT(*) AS CanEmp
FROM Emplead
GROUP BY  Suc
HAVING COUNT(*) > 2
```

Suc	CanEmp
1	4
2	3

HAVING tiene para grupos el mismo significado que WHERE para t-uplas

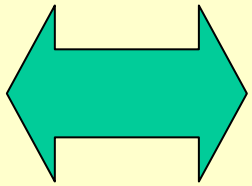
Condiciones para grupos

HAVING admite dos tipos de condiciones:

*** Condiciones con funciones de agregación (v.g. COUNT(*)>2)**

*** Condiciones de agrupamiento de columnas (También podrían incluirse en la cláusula WHERE)**

```
SELECT Suc
COUNT(*) AS CanEmp
FROM Emplead
GROUP BY  Suc
HAVING Suc< > '1'
```



Suc	CanEmp
2	3
3	1

```
SELECT Suc
COUNT(*) AS CanEmp
FROM Emplead
WHERE  Suc<A>'1'
GROUP BY Suc
```

Un ejemplo comprehensivo

Para cada sucursal en Buenos Aires que tenga no menos de 3 empleados, se quiere saber el sueldo promedio. El resultado se quiere ordenado de modo decreciente por valor de media de sueldo y por valor creciente de sucursal.

```
SELECT E.Suc, AVG(Sueldo) AS PrSue
FROM Emplead E, Sucurs S
WHERE  E.Suc=S.Suc
     AND  S.Ciudad='Buenos Aires'
GROUP BY E.Suc
HAVING Count(*) >=3
ORDER BY PrSue  DESC, Suc
```

El orden de las cláusulas es siempre el indicado

Un ejemplo comprehensivo

Para cada sucursal en Buenos Aires que tenga no menos de 3 empleados, se quiere saber el sueldo promedio. El resultado se quiere ordenado de modo decreciente por valor de media de sueldo y por valor creciente de sucursal.

```
SELECT E.Suc, AVG(Sueldo) AS PrSue
FROM Emplead E, Sucurs S
WHERE  E.Suc=S.Suc
     AND  S.Ciudad='Buenos Aires'
GROUP BY E.Suc
HAVING Count(*) >=3
ORDER BY PrSue  DESC, Suc
```

Solamente SELECT y FROM son obligatorias

Un ejemplo comprehensivo

Para cada sucursal en Buenos Aires que tenga no menos de 3 empleados, se quiere saber el sueldo promedio. El resultado se quiere ordenado de modo decreciente por valor de media de sueldo y por valor creciente de sucursal.

```
SELECT E.Suc, AVG(Sueldo) AS PrSue
FROM Emplead E, Sucurs S
WHERE  E.Suc=S.Suc
     AND  S.Ciudad='Buenos Aires'
GROUP BY E.Suc
HAVING Count(*) >=3
ORDER BY PrSue  DESC, Suc
```

GROUP BY no implica ordenamiento del resultado

Definición de vistas

La cláusula DEFINE VIEW define una vista, que es una tabla virtual.

Las t-uplas de la vista son el resultado de un QUERY, que es dinámicamente calculado toda vez que la vista es accedida.

Definición de vistas

```
CREATE VIEW Progra(Nemp,Suc,Ciudad)  AS
SELECT  E.Nemp,S.Suc,S.Ciudad
FROM Emplead E, Sucurs
WHERE  E.Suc=S.Suc
     AND  Posic='Progr'

SELECT  (*)
FROM Progra
WHERE  Ciudad='Buenos Aires'
```

NEmp	Suc	Ciudad
E003	1	Buenos Aires
E004	3	Buenos Aires
E007	1	Buenos Aires
E008	2	Bahía Blanca

NEmp	Suc	Ciudad
E003	1	Buenos Aires
E004	3	Buenos Aires
E007	1	Buenos Aires

Uso de vistas

*** Permite al usuario tener una visión personalizada de la base de datos, ajustada a sus necesidades específicas.**

→(Nivel externo)

*** En caso de modificación en el nivel lógico, las vistas pueden reproducir las tablas preexistentes.**

El usuario y los programas pueden realizar QUERY sobre las relaciones, como antes.

Uso de vistas

*** Control de acceso:**
Un Perfil de usuarios puede ser autorizada a ver parte de la tabla, en función de una definición de vista.

*** Una definición de vista puede hacer referencia a otras vistas.**

Actualización de vistas

* Las vistas pueden ser consultadas (QUERY) como tablas.

* La actualización de vistas tiene algunas limitaciones.

```
CREATE VIEW EmporSuc(Suc,CanEmp)  AS
SELECT  Suc COUNT(*)
FROM Emplead
GROUP BY Suc
```

Suc	CanEmp
1	4
2	3
3	1

```
UPDATE  CanEmp
SET CanEmp = CanEmp + 1
WHERE  Suc ='3'
```

¿Qué significa?

Actualización de vistas

*** Una vista no puede ser actualizada si en su definición aparece:**

*** GROUP BY**

*** DISTINCT**

*** JOIN**

*** Funciones de agregación**

Regla práctica:

Una vista puede ser actualizada si es posible determinar unívocamente qué t-uplas de las tablas base se actualizarán en razón de la actualización de la vista.

Fin de la presentación

Referencias:

*