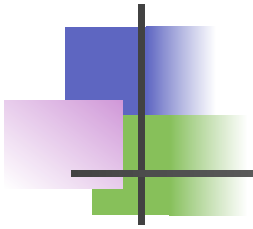
Departamento de
Ciencias de la Computación y Tecnologías de Información
Universidad del Bío-Bío
Sede Chillán

Bases de Datos

Lenguaje SQL

Mª Angélica Caro Gutiérrez
http://www.face.ubiobio.cl/~mcaro/
mcaro@ubiobio.cl



Lenguaje SQL

- Introducción
- Conceptos básicos
- Consultas básicas en SQL
- Consultas complejas en SQL
- Vistas (tablas virtuales) en SQL

Consultas Complejas en SQL

- SELECT: Consulta para recuperar datos de la BD
- La sintaxis de la orden SELECT consta básicamente de las cláusulas SELECT y FROM como obligatorias y de otras varias cláusulas opcionales:

<cláusula SELECT> <cláusula FROM>
[<cláusula WHERE>]
[<cláusula GROUP BY> [<cláusula HAVING>]]
[<cláusula ORDER BY>]

Consultas Complejas en SQL

- GROUP BY y HAVING
  - "Contar la cantidad de libros de cada clase"
  - Otra forma de resolverlo sería:

```
SELECT clase, COUNT(*)
FROM libro
GROUP BY clase;
```

  - Resultado:

	clase integer	count bigint
1	3	2
2	1	1

Consultas Complejas en SQL

- GROUP BY y HAVING
  - Una consulta general en SQL tiene la siguiente forma:

```
SELECT [DISTINCT] lista-selección
FROM lista(tablas,vistas)
WHERE condición
GROUP BY lista-para-formar-grupos
HAVING condición-sobre-grupos
```

  - La lista-selección en la cláusula SELECT consiste de:
    - 1. Una lista de nombres de atributos
    - 2. Una lista de términos de la forma **OpAgreg**(nombre-columna) **AS** nuevo-nombre
  - Todos los atributos que aparecen en (1) deben aparecer en lista-para-formar-grupos

Consultas Complejas en SQL

- GROUP BY y HAVING

```
SELECT [DISTINCT] lista-selección
FROM lista(tablas,vistas)
WHERE condición
GROUP BY lista-para-formar-grupos
HAVING condición-sobre-grupos
```

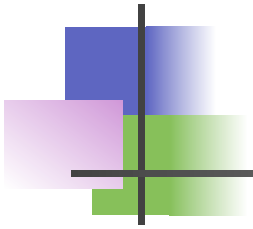
  - Cada fila del resultado de la consulta se corresponde con un grupo, que es un conjunto de filas que concuerdan con los valores para las columnas de lista-para-formar-grupos
  - Las expresiones en condición-sobre-grupos de la cláusula HAVING deben tener un único valor por grupo
  - Si se omite GROUP BY, se considera a toda la tabla como un solo grupo

Consultas Complejas en SQL

- Consultas con GROUP BY y HAVING
  - Dado el esquema: ALUMNO(ID,NOMBRE,EDAD, CIUDAD)
  - Encontrar la edad del alumno más joven (por sobre los 18 años) para cada ciudad que tenga como mínimo dos alumnos

```
SELECT CIUDAD, MIN(EDAD) AS minEDAD
FROM ALUMNO
WHERE EDAD > 18
GROUP BY CIUDAD
HAVING COUNT (*) > 1
```

  - Los duplicados no se eliminan a menos que se especifique con DISTINCT



Consultas Complejas en SQL

- Cláusula HAVING
  - No se pueden usar funciones de agrupación en la cláusula WHERE de un SELECT. O sea, no se puede usar el WHERE para, de forma selectiva eliminar datos que no interesan del resultado de una consulta agrupada.
  - Por ejemplo, en la tabla

    examenes(id_asignatura, nro_estudiante, nota, fecha)
  - Si hacemos:

```
SELECT nro_estudiante, avg(nota)
FROM examenes
WHERE avg (nota) >6
GROUP BY nro_estudiante;
```

  - Daría un error por usar una función de agrupamiento en el WHERE

Consultas Complejas en SQL

- Cláusula HAVING

  - La cláusula HAVING hace una función parecida a la del WHERE cuando se trabaja con este tipo de funciones. Así, para listar aquellos alumnos cuya media es mayor que 6 sería:

```
SELECT nro_estudiante, avg(nota)
FROM examenes
GROUP BY nro_estudiante
HAVING avg (nota) >6;
```

  - El campo referenciado en la cláusula HAVING no puede tener más de un valor por grupo. Esto significa que, en la práctica, HAVING sólo puede referenciar a funciones de agregación y columnas que se están usando en el GROUP by

Consultas Complejas en SQL

- Cláusula HAVING

  - De cada proyecto, en el que trabajen más de dos empleados, recupere su número, su nombre y el número de empleados que trabajan en él.

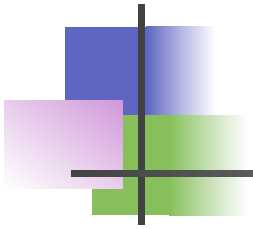
```
SELECT NUMPROY, NOMBREPROYECTO, COUNT(*)
FROM (PROYECTO JOIN TRABAJA_EN
        ON NUMPROY = NUMPROYECTO)
GROUP BY NUMPROY, NOMBREPROYECTO
HAVING COUNT(*) > 2;
```

Consultas Complejas en SQL

- Cláusula HAVING

  - De cada proyecto, recupere su número, su nombre y el número de empleados del departamento 5 que trabajan en él.

```
SELECT P.NUMPROY, P.NOMBREPROYECTO, COUNT(*)
FROM ((PROYECTO AS P JOIN TRABAJA_EN
        ON NUMPROY = NUMPROYECTO)
        JOIN EMPLEADO ON DNI=DNIEMPLEADO)
WHERE DNO = 5
GROUP BY NUMPROY, NOMBREPROYECTO;
```

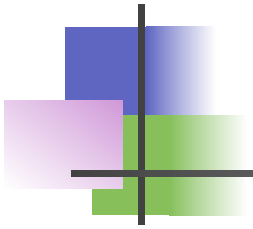


Ejercicios

Dado el esquema:

DEPARTAMENTOS (codigo, nombre), AREAS (**codigo**, nombre, departamento)
PROFESORES (**codigo**, apellido1, apellido2, nombre, activo, categoria, dedicacion, area)

1. Obtener el número de profesores activos (activo=1) de cada área agrupados según su categoría.
2. Obtener el número de profesores activos de cada departamento y área agrupados según su categoría.
3. Listar el código y nombre de cada departamento y el número de profesores activos (activo=1) que tiene.
4. Listar el código de cada departamento con más de 5 profesores activos.

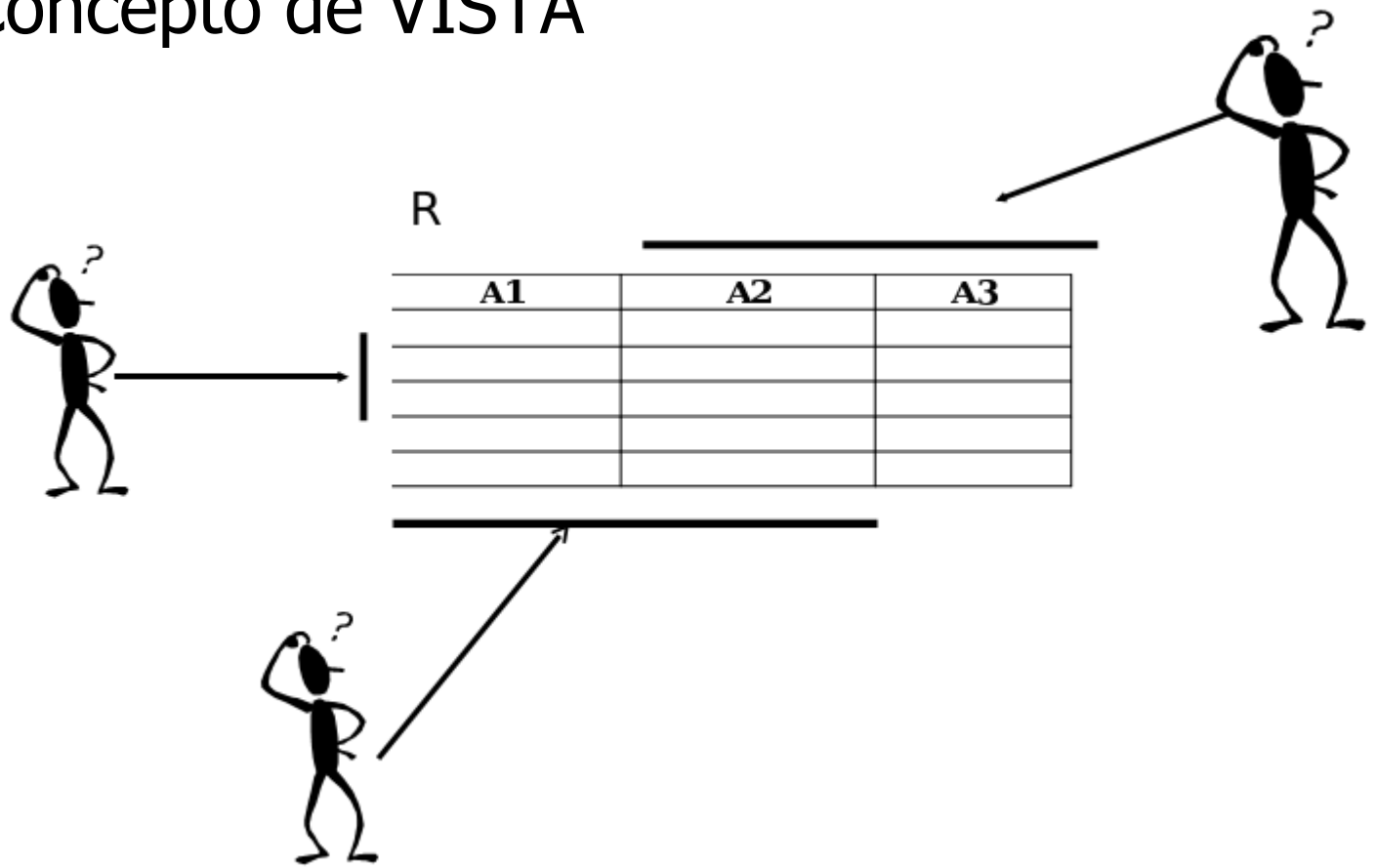


Unidad 5: Lenguaje SQL

- Introducción
- Conceptos Básicos
- Consultas básicas y complejas en SQL
- Sentencias de inserción, eliminación y de actualización en SQL
- Vistas (tablas virtuales) en SQL

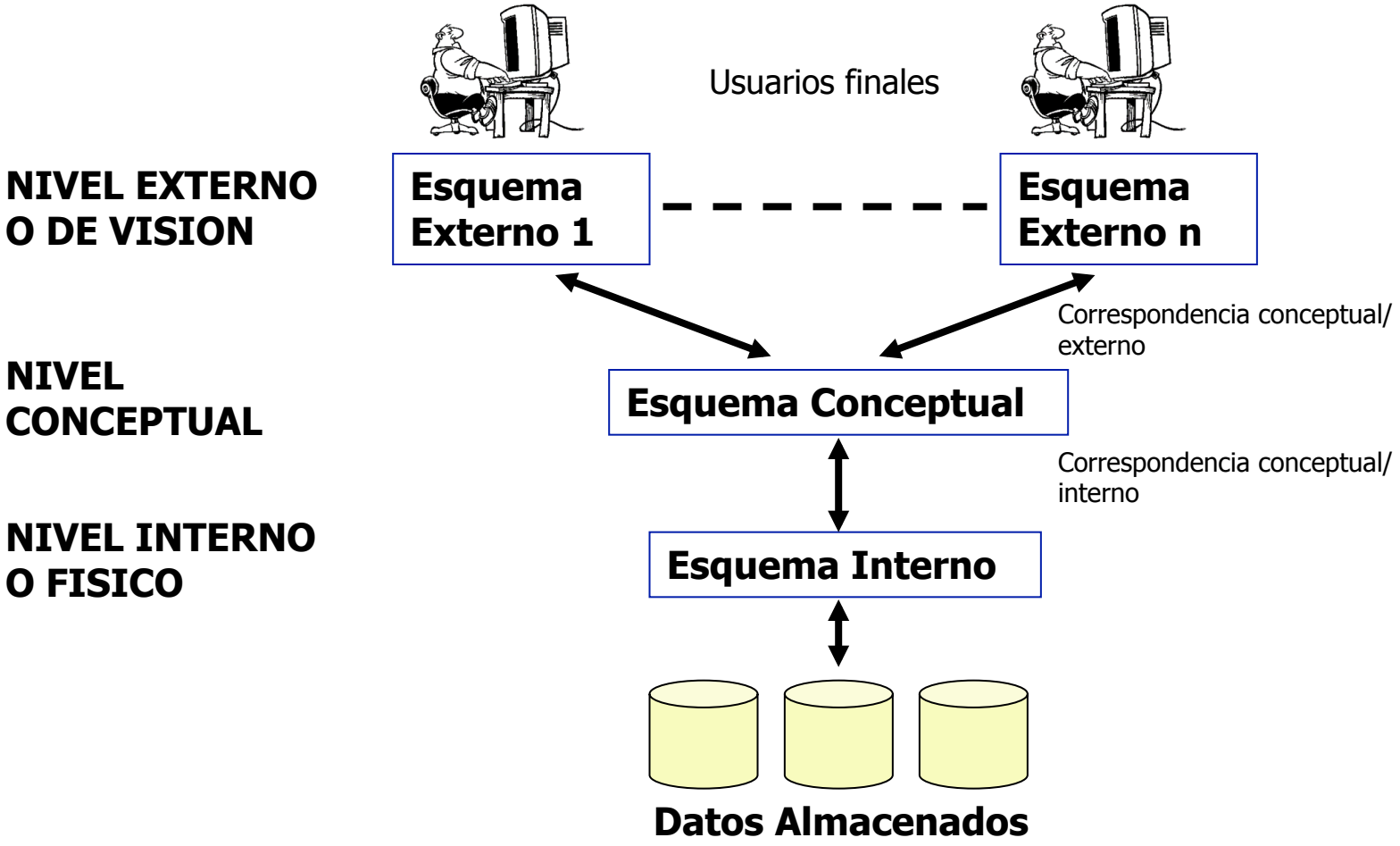
Vistas (Tablas virtuales) en SQL

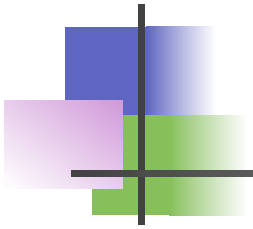
- Concepto de VISTA



Vistas (Tablas virtuales) en SQL

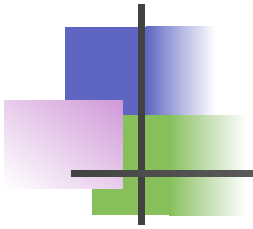
- Arquitectura y Niveles de abstracción





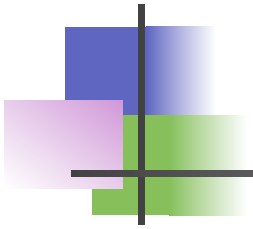
Vistas (Tablas virtuales) en SQL

- Concepto de VISTA
  - Una vista (VIEW) es una tabla lógica (no física) que se basa en otras tablas o vistas.
  - Funciona como una ventana a través de la cual pueden visualizarse o modificarse datos de las tablas.
  - Una vista no contiene datos en si misma.
  - Las tablas sobre las que se define una vista se llaman tablas base.
  - La vista se almacena como una sentencia SELECT nominada en el diccionario de datos.



Vistas (Tablas virtuales) en SQL

- ¿Por qué utilizar vistas?
  - Conseguir un acceso restringido a la base de datos, ya que la vista puede mostrar sólo una porción específica.
  - Sustituir consultas complejas por varias consultas más simples que emplean vistas.
    - Por ejemplo, una selección sobre una vista definida sobre varias tablas será más fácil de realizar que si tuviéramos que utilizar la combinación (JOIN) de esas tablas para realizarla.
  - Proveer independencia de datos.
  - Proveer distintas maneras de ver los mismos datos, adaptados a cada usuario o aplicación.
  - Permitir el acceso a grupos de usuarios de acuerdo con unos criterios concretos.



Vistas (Tablas virtuales) en SQL

- Vistas Simples vs Complejas
  - En función de las operaciones de manipulación de datos (DML), es decir, INSERT, UPDATE y DELETE, que se pueden realizar a través de ellas, existen dos categorías de vistas:
  - **Simples**:
    - Extraen los datos de una sola tabla
    - No contienen funciones ni grupos de datos (GROUP BY)
    - Siempre pueden realizarse operaciones DML a través de ellas.
  - **Complejas**:
    - Extraen los datos de múltiples tablas
    - Contienen funciones o grupos de datos (GROUP BY)
    - Es frecuente que no permitan operaciones DML.

Vistas (Tablas virtuales) en SQL

- Creando una Vista – sintaxis (i):

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW <vista>
   [(<alias>[, <alias>] … )]
   AS <subconsulta>
   [WITH CHECK OPTION [CONSTRAINT <restricción>]]
   [WITH READ ONLY [CONSTRAINT <restricción>]];
```

Vistas (Tablas virtuales) en SQL

- Creando una Vista – sintaxis (ii):

OR REPLACE	Se utiliza por si la vista ya estuviera creada anteriormente. En ese caso, la sustituye por la nueva definición.
FORCE	Crea la vista sin comprobar si las tablas base existen.
NO FORCE	Crea la vista sólo si las tablas base de donde se extraen los datos existen realmente (es la opción por defecto).
<vista>	Es el nombre de la vista.
<alias>	Especifica alias para las expresiones/columnas seleccionadas por la subconsulta. El número de alias debe coincidir con el número de expresiones seleccionadas por la vista.

Vistas (Tablas virtuales) en SQL

- Creando una Vista – sintaxis (iii):

<subconsulta>	Es una sentencia SELECT completa. Se pueden emplear alias para las columnas en la lista que sigue al SELECT.
WITH CHECK OPTION	Especifica si la vista pueden actualizarse. El SGBD comprobará si la vista puede actualizarse.
<restricción>	Nombre asignado a la restricción CHECK OPTION o a la restricción READ ONLY.
WITH READ ONLY	Asegura que no podrán ejecutarse operaciones de DML a través de la vista. La vista sólo permite consultas.

Vistas (Tablas virtuales) en SQL

- Creando una Vista – Ejemplo:
  - Crear una vista que contiene los apellidos y nombre de los empleados del departamento "Ventas". Asegurarse que a través de ella sólo pueden modificarse, eliminarse o añadirse los empleados de dicho departamento.

```
CREATE VIEW EmpDepVentas
   AS SELECT apellidos, nombre
          FROM Empleados
          WHERE dep="Ventas"
          WITH CHECK OPTION;
```

Vistas (Tablas virtuales) en SQL

- Creando una Vista – Ejemplo:
  - "Los empleados del departamento de 'Investigación' "

```
CREATE VIEW INVESTIGACION AS
SELECT E.DNI, E.NOMBRE, E.APELLIDO1, E.SUELDO,E.DIRECCION
FROM (EMPLEADO E JOIN DEPARTAMENTO
        ON E.DNO = NUMERODPTO)
WHERE NOMBREDPTO = 'Investigacion'
ORDER BY E.APELLIDO1,E.NOMBRE;
```

  - Ahora podemos consultar directamente la vista INVESTIGACION

```
SELECT * FROM INVESTIGACION;
```

Vistas (Tablas virtuales) en SQL

- Creando una Vista – Ejemplo:
  - Por cada departamento queremos mantener un resumen con el total de empleados y el sueldo total

```
CREATE VIEW RESUMEN (NOMBRE_DEPTO, TOTAL_EMPLEADOS,
TOTAL_SUELDO) AS
SELECT NOMBREDPTO, COUNT(*), SUM(SUELDO)
FROM (EMPLEADO JOIN DEPARTAMENTO ON DNO=NUMERODPTO)
GROUP BY NOMBREDPTO
ORDER BY NOMBREDPTO;
```

  - Ahora

```
SELECT * FROM RESUMEN;
```

Vistas (Tablas virtuales) en SQL

- Creando una Vista – Empleo de Alias
  - Los alias se pueden definir dentro de la subconsulta o fuera de ella, justo a continuación del nombre la vista.
  - Crear una vista que contiene los apellidos y nombre de los empleados del departamento "Ventas" renombrando dichas propiedades como "Last_name" y "First_name".
    - a) CREATE VIEW EmpDepVentas
      AS SELECT apellidos Last_name, nombre First_name
      FROM Empleados
      WHERE dep="Ventas";
    - b) CREATE VIEW EmpDepVentas (Last_name, First_name)
      AS SELECT apellidos, nombre
      FROM Empleados
      WHERE dep="Ventas";

Vistas (Tablas virtuales) en SQL

- Otros operadores:
  - Visualizar la estructura de una vista:

    DESCRIBE <vista>;
    donde:
    <vista> Es el nombre de la vista.

  - Listar las vistas existentes:

    SELECT * FROM USER_VIEWS;

  - Eliminación de una vista:
    - DROP VIEW <nombre de la vista>;

Vistas (Tablas virtuales) en SQL

- Recuperando datos desde una Vista

  - Una Vista se puede emplear exactamente igual que una tabla base para recuperar datos de ella mediante consultas (SELECT).
  - Cuando se ejecuta una vista, el DBMS recupera la definición de la vista y ejecuta la subconsulta correspondiente.

  - Ejemplo: Listar los apellidos y nombre de los empleados del departamento "Ventas" ordenados alfabéticamente.

```
SELECT apellidos, nombre
   FROM EmpDepVentas
   ORDER BY apellidos, nombre;
```

Vistas (Tablas virtuales) en SQL

- Implementación de Vistas
  - Modificación de consulta. Convertir la consulta sobre las tablas de base. Su principal desventaja es el tiempo de ejecución.
  - Materialización de vistas. Crear físicamente la tabla de vista cuando se consulta la vista por primera vez y mantener la tabla para aprovecharla en consultas posteriores. Esto implica contar con estrategias para mantener actualizada la vista siendo una de ella la actualización incremental

Vistas (Tablas virtuales) en SQL

- Actualización de Vistas
  - Muy complicada y ambigua.
  - Una vista con una sola tabla base es actualizable si los atributos de la vista contienen la clave primaria o alguna otra clave candidata de la relación base.
  - En general las vistas definidas sobre múltiples tablas por medio de join no son actualizables.
  - Las vistas definidas mediante agrupación y funciones de agregación no son actualizables.