ÁLGEBRA Y CÁLCULO RELACIONAL

Bases de Datos

Álgebra Relacional

- Un *álgebra* es un sistema matemático constituido por
  - Operandos: objetos (valores o variables) desde los cuales nuevos objetos pueden ser construidos.
  - Operadores: símbolos que denotan procedimientos para construir nuevos objetos desde objetos dados.
- El *álgebra relacional* es un álgebra en la cual
  - Sus operandos son relaciones (instancias) o variables que representan relaciones.
  - Sus operadores están diseñados para hacer la tareas más comunes que se necesitan para manipular relaciones en una base de datos.
- El resultado es que el álgebra relacional se puede utilizar como un lenguaje de consulta.
- En la práctica el álgebra relacional debe ser *extendida* para abarcar la mayor parte de las tareas reales que se hacen con los datos.
- Estudiaremos en detalle los operadores clásicos.
- El álgebra relacional es un lenguaje de consulta procedimental. En este tipo de lenguaje,el usuario da instrucciones al sistema para que realice una serie de procedimientos u operaciones en la base de datos para calcular un resultado final.

  El SQL, es un lenguaje declarativo de acceso a base de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla información de bases de datos, así como hacer cambios en ellas.

Tablas

Estas serán las tablas que usaremos en la mayoría de ejemplos

Empleado

nombre	*sueldo*	*cod_dept*	*fecha_ing*
Torres	$ 1.200.000	A1	01/01/2004
Soto	$ 500.000	A2	01/01/2003
Pérez	$ 300.000	A2	01/10/2003
Figueroa	$ 600.000	A1	01/03/2002
Salas	$ 1.500.000	A1	01/01/2002
Ríos	$ 2.000.000	A3	01/06/2002
Campos	$ 800.000	A2	01/11/2003
Venegas	$ 600.000	A1	01/06/2002
Carcamo	$ 500.000	A2	01/04/2003
Gonzalez	$ 2.000.000	A3	01/10/2002

Departamento

nombre	*cod_dept*	*fecha_creac*
Informática	A1	01/03/2002
Marketing	A2	01/01/2002
Ventas	A3	01/01/2001
Recursos Humanos	A4	01/01/2003

Selección

- Operador de selección σ, selecciona un subconjunto de las tuplas de una relación.
- Tuplas seleccionadas son las que satisfacen cierto predicado lógico P. El predicado puede depender de los atributos de la relación y de valores constantes.
- El operador σ toma una relación como argumento y el resultado es una nueva relación.
- Sintaxis:

$$\sigma_P (r)$$

- Ejemplo: Seleccionar los datos del empleado Soto:

$$\sigma_{nombre=\text{Soto}}(empleado)$$

nombre	*sueldo*	*cod_dept*	*fecha_ing*
Soto	$ 500.000	A2	01/01/2003

- Ejemplo: Los datos de los empleados con sueldo $\geq$ \$500.000 que ingresaron durante el 2003:

$$\sigma_{\text{sueldo} \geq 500000 \wedge (\text{fech\_ing} \geq 01/01/2003 \wedge \text{fech\_ing} \leq 31/12/2003)} (\text{empleado})$$

nombre	*sueldo*	*cod_dept*	*fecha_ing*
Soto	$ 500.000	A2	01/01/2003
Campos	$ 800.000	A2	01/11/2003
Carcamo	$ 500.000	A2	01/04/2003

Proyección

- Operador de proyección π, proyecta una relación sobre un subconjunto de sus atributos.

- El operador π toma una relación como argumento y el resultado es una nueva relación.

- Sintaxis:

$$\pi_A(r)$$

donde A representa el conjunto de atributos sobre los que la relación r se proyectará.

  - Ejemplo: obtener los nombres de los distintos departamentos

$$\pi_{nombre}(departamento)$$

nombre
Informática
Marketing
Ventas
Recursos Humanos

- Obtener los montos de sueldo de los empleados:

$$\pi_{sueldo}(empleado)$$

sueldo
$ 1.200.000
$ 500.000
$ 300.000
$ 600.000
$ 1.500.000
$ 2.000.000
$ 800.000

se eliminan los repetidos! una relación es un conjunto.

Composición de Operaciones

- El resultado de cada operación es una nueva relación ⇒ se pueden aplicar operadores a los resultados de aplicaciones previas.

- Por ejemplo:

$$\pi_A(\sigma_P(r))$$

$$\sigma_P(\pi_A(r))$$

$$\sigma_{P_1}(\sigma_{P_2}(r))$$

- Ejemplo: Obtener los nombres de los empleados que ganan más de $1.000.000.

$$\pi_{nombre}(\sigma_{sueldo>1000000}(empleado))$$

nombre
Torres
Salas
Ríos
Gonzalez

- Ejemplo: Obtener el sueldo y la fecha de ingreso de Soto:

$$\pi_{sueldo,fech\_ing}(\sigma_{nombre=\text{Soto}}(empleado))$$

sueldo	*fecha_ing*
$ 500.000	01/01/2003

Unión

- Dado que las relaciones son conjuntos de tuplas, se pueden realizar las operaciones usuales de conjuntos como la unión.

- Sintaxis:

$$r_1 \cup r_2$$

- Se deben hacer ciertas restricciones para realizar la unión:
  - Ambas relaciones deben tener el mismo número de atributos.
  - El dominio del atributo *i*-ésimo de cada relación debe coincidir.
- Ejemplo: Obtener los nombres de los empleados que ganan más de $1.500.000 o que trabajan en el departamento con código A1.

$$\pi_{nombre}(\sigma_{sueldo>1500000}(empleado) \cup \sigma_{cod\_dept=A1}(empleado))$$

nombre
Torres
Figueroa
Salas
Ríos
Venegas
Gonzalez

Diferencia

- También se puede usar la diferencia de conjuntos, las tuplas que están en una relación pero no en la otra.

- Sintaxis:

$$r_1 - r_2$$

- Para poder realizar la diferencia se deben cumplir las mismas restricciones que para la unión
- Ejemplo:

  $\pi_{nombre}(alumno) - \pi_{nombre}(\sigma_{carrera=\text{Bioinformatica}}(alumno))$

  Resulta en una relación que contiene a todos los nombres de los alumnos excepto de los alumnos de la carrera de Bioinformática.

Producto Cartesiano

- Representa al producto cartesiano usual de conjuntos.
- Combina tuplas de cualquieras dos (o más) relaciones, hace la combinación de todos con todos.
- Si las relaciones a operar tienen N y M tuplas de n y m componentes respectivamente, la relación resultante del producto cartesiano tiene $N \times M$ tuplas de $n+m$ componentes.
- Sintaxis:

$$r_1 \times r_2$$

- No hay restricciones a los dominios de las relaciones similares a las anteriores operaciones.
- Nos permite *reunir* datos de dos relaciones distintas.
- Cuidado con los nombres repetidos! se deben renombrar ciertos atributos para no tener problemas.

departamento $\times$ *departamento*

nombre	*cd*	*fec creac*	*nombre*	*cd*	*fec creac*
Informática	A1	01/03/2002	Informática	A1	01/03/2002
Informática	A1	01/03/2002	Marketing	A2	01/01/2002
Informática	A1	01/03/2002	Ventas	A3	01/01/2001
Informática	A1	01/03/2002	Rec. Hum.	A4	01/01/2003
Marketing	A2	01/01/2002	Informática	A1	01/03/2002
Marketing	A2	01/01/2002	Marketing	A2	01/01/2002
Marketing	A2	01/01/2002	Ventas	A3	01/01/2001
Marketing	A2	01/01/2002	Rec. Hum.	A4	01/01/2003
Ventas	A3	01/01/2001	Informática	A1	01/03/2002
Ventas	A3	01/01/2001	Marketing	A2	01/01/2002
Ventas	A3	01/01/2001	Ventas	A3	01/01/2001
Ventas	A3	01/01/2001	Rec. Hum.	A4	01/01/2003
Rec. Hum.	A4	01/01/2003	Informática	A1	01/03/2002
Rec. Hum.	A4	01/01/2003	Marketing	A2	01/01/2002
Rec. Hum.	A4	01/01/2003	Ventas	A3	01/01/2001
Rec. Hum.	A4	01/01/2003	Rec. Hum.	A4	01/01/2003

Producto Cartesiano: Ejemplos

- Obtener el nombre del departamento en el que Soto trabaja: Primero hacemos la unión cartesiana igualando un atributo

$$\sigma_{depto.cod\_depto=emp.cod\_depto}(empleado \times departamento)$$

Ahora podemos hacer la selección y proyección

$$\pi_{depto.nombre}(\sigma_{emp.nombre=\text{Soto}}$$

$$(\sigma_{depto.cod\_depto=emp.cod\_depto}(empleado \times departamento)))$$

departamento.nombre
Marketing

Renombre

- A veces necesitamos obtener información uniendo datos de la misma tabla.
- Por ejemplo, obtener los nombres de todos los empleados que ingresaron después que Soto a la empresa.
- Primer intento:

$$empleado \times (\sigma_{nombre=\text{Soto}}(empleado))$$

- ¿Cómo nos referimos a una u otra instancia de la tabla *empleado*?
- El operador de renombre ρ soluciona el problema.
- El operador ρ toma una relación y entrega la misma relación pero con otro nombre, podemos referirnos a distintas instancias de la misma relación.
- Sintaxis:

$$\rho_X(r)$$

- Volviendo al ejemplo, obtener los nombres de todos los empleados que ingresaron después que Soto a la empresa, segundo intento:

$$empleado \times (\sigma_{nombre=\text{Soto}}(\rho_{empleado2}(empleado))$$

  - Ahora podemos hacer:

$$\pi_{empleado.nombre}(\sigma_{empleado.fecha\_ing>empleado2.fecha\_ing}(empleado \times (\sigma_{nombre=\text{Soto}}(\rho_{empleado2}(empleado)))))$$

Formalización de expresiones

- El álgebra relacional es un lenguaje de expresiones. Toda expresión se puede generar a partir de las siguientes reglas:
  - Toda relación de la base de datos es una expresión.
  - Si E_1 y E_2 son expresiones entonces las siguientes también son expresiones:
    - $E_1 \cup E_2$
    - $E_1 - E_2$
    - $E_1 \times E_2$
    - $\sigma_P(E_1)$ donde P es un predicado con atributos de E_1
    - $\pi_A(E_1)$ donde A es una lista de atributos de E_1
    - $\rho_X(E_1)$ donde x es el nuevo nombre de la relación E_1.
- Cada expresión generada por las reglas anteriores tiene como resultado una relación.

Operaciones Adicionales, Intersección

- Las operaciones anteriores son suficientes para definir toda el algebra relacional.
- Algunas consultas habituales son complejas de realizar con combinaciones de operaciones simples, usamos algunos operadores adicionales.

Intersección

- La intersección usual de conjuntos.
- Sintaxis:

$$r_1 \cap r_2$$

- Se deben cumplir las mismas restricciones que en la unión y diferencia, los atributos de la relaciones involucradas deben tener los mismos dominios.
- La intersección se puede crear a partir de la diferencia:

$$r_1 \cap r_2 = r_1 - (r_1 - r_2)$$

- Ejemplo:

$$\pi_{nombre}(alumno\ preg) \cap \pi_{nombre}(alumno\ posg)$$

resulta en una relación que contiene los nombres de todos los alumnos que son simultáneamente de pre y posgrado.

Reunión Natural (Natural Join)

- Hace un producto cartesiano de sus dos argumentos y realiza una selección forzando la igualdad de atributos que aparecen en ambas relaciones. Serían la clave primaria y foranea de ambas relaciones.
- Elimina repetidos (como toda operación de conjuntos).

- Sintaxis:

$$r_1 \bowtie r_2$$

Expresada en operaciones básicas quedaría

$$R \bowtie S = \Pi_{A1,A2...An}(\sigma_\theta(R \times S))$$

donde la condición θ es la igualdad Clave Primaria = Clave Externa (o Foránea), y la proyección elimina la columna duplicada (clave externa).

- Ejemplo: listar todos los empleados y el nombre del departamento en el que trabajan

$$\pi_{emp.nombre,dept.nombre}(empleado \bowtie departamento)$$

empleado.nombre	*departamento.nombre*
Torres	Informática
Soto	Marketing
Pérez	Marketing
Figueroa	Informática
Salas	Informática
Ríos	Ventas
Campos	Marketing
Venegas	Informática
Carcamo	Marketing
Gonzalez	Ventas

Join en General

- Un forma más general de hacer Join es especificando una *propiedad de reunión*.
- Se hace entonces un producto cartesiano de las dos relaciones y se realiza una selección forzando una propiedad más general que la igualdad de atributos que aparecen en ambas relaciones.

- Sintaxis:

$$r_1 \bowtie_P r_2$$

  donde P es la propiedad de reunión.

- Ejemplo: listar todos los pares de nombres de empleados y departamentos tales que el empleado ingresó a la empresa en una fecha anterior a la de creación del departamento

$\pi_{emp.nombre,depto.nombre}$(empleado $\bowtie$(fecha_ing<fecha_creac) departamento)

Relaciones Temporales

- A veces las consultas se hacen muy extensas.
- Una forma de simplificarlas es usando relaciones temporales y asignándoles expresiones para usar después:
- $r \leftarrow E$ asigna la expresión de álgebra relaciones E a la nueva relación r.
- Ejemplo:

$$temp \leftarrow \sigma_{sueldo>500000}(empleado \bowtie departamento)$$

$$\pi_{empleado.nombre,departamento.nombre}(temp)$$

- El último resultado son los nombres de empleados y el departamento en el que trabajan tales que el sueldo del empleado es mayor a $500.000.

Cálculo Relacional de Tuplas

- El cálculo relacional de tuplas es un lenguaje no procedural, es de tipo declarativo.
- Con el álgebra relacional damos un procedimiento para una expresión.

$$\pi_{A_1,B_1}(\sigma_{A_1=v}(r_A \bowtie r_B))$$

- En el cálculo relacional de tuplas especificamos la información deseada sin dar un procedimiento para obtenerla.

$$\{t \mid P(t)\}$$

  - Este último conjunto representa a la relación de todas las tuplas *t* que cumplen la propiedad lógica *P*.

- Supongamos que necesitamos sólo los nombres de los empleados que tienen sueldo mayor a $500.000.

  En álgebra relacional usamos el operador π para obtener sólo los nombres. En el cálculo relacional de tuplas debemos usar la construcción *existe* "∃":

$$\exists\, t \in r(Q(t))$$

  que significa: "existe una tupla *t* en la relación *r* que cumple el predicado *Q*(*t*)".

- Ejemplo: obtener los nombres de los empleados que tienen sueldo mayor a $500.000.

$$\{t \mid \exists s \in \textit{empleado}\ (t[\textit{nombre}] = s[\textit{nombre}] \wedge s[\textit{sueldo}] > 500000\}$$

- Se lee: "el conjunto de todas las tuplas tales que existe una tupla *s* en la relación *empleado* para la cual los valores de *t* y *s* son iguales en el atributo *nombre* y el valor de *s* en el atributo *sueldo* es mayor que 500000".
- La variable de tupla *t* se define sólo en el atributo *nombre*, es el único atributo *cuantificado*. El resultado es una relación con sólo un atributo.

Cálculo Relacional de Tuplas: Ejemplos

- Supongamos que necesitamos los datos de todos los empleados del departamento de Marketing y no sabemos el código del departamento.
- En álgebra relacional usamos el operador $\bowtie$ para obtener los datos de ambas tablas.
- En el cálculo relacional de tuplas debemos usar también la construcción *existe*:

$$\{t \mid t \in empleado \ \wedge \ \exists s \in departamento\ (s[cod\_depto] = t[cod\_depto] \wedge\ s[nombre] = marketing\}$$

- Se lee: "el conjunto de todas las tuplas *t* de la relación *empleado* tales que existe una tupla *s* en la relación *departamento* para la cual los valores de *t* y *s* son iguales en el atributo *cod_depto* y el valor de *s* en el atributo *nombre* es *marketing*".
- ¿Cómo expresamos consultas del tipo "los nombres de todos los alumnos que no son de bioinformática, suponiendo que contamos con las relaciones *alumno* y *alumno_bioinformatica*?
- Necesitamos el operador lógico de negación "¬".

$$\{t \mid \exists s \in alumno\ (s[nombre] = t[nombre]) \wedge \neg\exists u \in alumno\_bioinformatica\ (u[nombre] = t[nombre])\}$$

- Se lee: "el conjunto de todas las tuplas *t* tales que existe una tupla *s* en la relación *alumno* que comparte el atributo *nombre* con *t* y NO existe una tupla *u* en la relación *alumno_bioinformatica* que comparta el atributo nombre con *t*".

Cálculo Relacional de Tuplas: Ejemplos

- En general podemos usar "casi" cualquier expresión lógica para seleccionar tuplas.
- Podemos usar las construcciones *para todo* "$\forall$", *implica que "*$\Rightarrow$*",* etc.

$$\{t \mid \exists s \in \textit{depatamento}\ (s[\textit{nombre}] = t[\textit{nombre}] \wedge \forall u \in \textit{empleado}\ (u[\textit{cod\_depto}] = s[\textit{cod\_depto}] \Rightarrow u[\textit{sueldo}] > 500000))\}$$

- Se lee: "el conjunto de todas las tuplas *t* tales que, existe una tupla *s* en la relación departamento con la que comparte el atributo *nombre* y que para toda tupla *u* en la relación empleado, si *u* comparte el atributo *cod_depto* con *s* entonces el valor del atributo *sueldo* de *u* es mayor que 500000.
- En este conjunto están los nombres de los departamentos cuyos empleados tienen todos un sueldo mayor a $500.000.
  ¿Cómo hacemos esta consulta en álgebra relacional?
- ¿Por qué "casi" todas las expresiones lógicas? ¿Qué resulta de la siguiente consulta?

$$\{t \mid \neg(t \in \textit{empleado})\}$$

- ¡Resulta una relación con un número infinito de tuplas!
  ¡Tuplas que ni siquiera se encuentran en nuestra base de datos!
- No aceptamos entonces este tipo de consultas, aceptamos sólo *consultas seguras*.
- Una consulta $\{t \mid P(t)\}$ es segura si el resultado son valores que pertenecen a $dom(P)$, donde $dom(P)$ es la unión de los dominios de todas las relaciones que aparecen en P.

$\{t \mid \neg(t \in \textit{empleado})\}$ no es segura

$\{t \mid t \in \textit{alumno} \wedge \neg(t \in \textit{alumno bioinformatica})\}$ si es segura.

Cálculo Relacional de Dominios

- Es un lenguaje de consulta sobre el modelo relacional no procedural.
- A diferencia del cálculo relacional de tuplas, consulta acerca valores del dominio de los atributos de las relaciones, no de las tuplas de éstas.
- Una consulta en cálculo relacional de dominios es de la siguiente forma:

$$\{< x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$$

  dónde $x_1, x_2, \ldots, x_n$ se llaman variables de dominio y P es una fórmula proposicional sobre las variables del dominio.

- Ejemplo: Obtener el nombre, el código y la fecha de creación de los departamentos creados después del 2003

$$\{< a, b, c > \mid < a, b, c > \in \textit{departamento} \ \wedge \ c > 1/1/2003\}$$

- Ejemplo: Obtener el nombre de los empleados que ganan $800.000

$$\{< n > \mid \exists a, b, c\,(< n, a, b, c > \in \textit{empleado} \ \wedge \ c = 800000)\}$$

- Ejemplo: Obtener los nombres de los empleados y de los departamentos en los que trabajan

$$\{< e, d > \mid \exists a, b, c, f\,(< e, a, b, c > \in \textit{empleado} \wedge < d, a, f > \in \textit{departamento}\}$$

- A veces es más simple representar consultas para atributos específicos en el cálculo de dominios que en el cálculo de tuplas.
- También existe una noción muy similar de *consultas seguras* en el cálculo de tuplas. Las consultas que se salen del dominio no son seguras.

Álgebra vs Cálculo Relacional

- El cálculo relacional nos entrega dos lenguajes de consulta en una base de datos bajo el modelo relacional.
- A veces se hace mucho más “natural” expresar consultas en un lenguaje de cálculo relacional que con el álgebra relacional.
- Ventaja: se expresa “lo que se quiere” sin necesitar explicar cómo obtenerlo (sin especificar un procedimiento).
- Resultado muy importante:

  El álgebra relacional, el cálculo relacional de tuplas y el cálculo relacional de dominios tienen el mismo poder expresivo ⇒ pueden responder el mismo tipo de consultas en una base de datos relacional.

Modificación de la Base de Datos

- Hasta ahora hemos visto como obtener información desde la Base de Datos.
- Es necesario entregar herramientas para poder alterar el estado de la Base de Datos.
- Eliminación, Inserción, Actualización son algunas de las tareas necesarias, usaremos el álgebra relacional.

Eliminación

- Usamos el operador de diferencia ($-$) y asignación ($\leftarrow$)

$$r \leftarrow r - E$$

donde r es una relación y E una expresión del álgebra relacional.

- Eliminar los datos de Soto

$$empleado \leftarrow empleado - \sigma_{nombre=\text{Soto}}(empleado)$$

- Eliminar todos los empleados del departamento de Marketing

$$r1 \leftarrow \sigma_{dep.nombre=\text{Marketing}}(empleado \bowtie departamento)$$

$$r2 \leftarrow \pi_{emp.nombre,sueldo,emp.cod\_depto,fech\ ing}(r1)$$

$$empleado \leftarrow empleado - r2$$

<u>nombre</u>	sueldo	cod_dept	fecha_ing
Torres	$ 1.200.000	A1	01/01/2004
Figueroa	$ 600.000	A1	01/03/2002
Salas	$ 1.500.000	A1	01/01/2002
Ríos	$ 2.000.000	A3	01/06/2002
Venegas	$ 600.000	A1	01/06/2002
Gonzalez	$ 2.000.000	A3	01/10/2002

Modificación de la Base de Datos (cont.)

Inserción

- Usamos el operador de unión

$$r \leftarrow r \cup E$$

donde r es una relación y E una expresión del álgebra relacional.

- Agregar el departamento de Finanzas con código A5 y fecha de creación 5/8/2004

$$departamento \leftarrow departamento \cup \{(\text{Finanzas, A5, 5/8/2004})\}$$

nombre	*cod_dept*	*fecha_crea*
Informática	A1	01/03/2002
Marketing	A2	01/01/2002
Ventas	A3	01/01/2001
Recursos Humanos	A4	01/01/2003
Finanzas	A5	05/08/2004

Actualización

- A veces queremos cambiar el valor de algún atributo de una tupla sin cambiarla entera. Podría hacerse una eliminación e inserción, pero resulta engorrosos muchas veces.
- Usamos un nuevo operador δ (no reasignamos!):

$$\delta_{A \leftarrow E}(r)$$

que representa el hecho de cambiar el atributo A por la expresión matemática E en la relación r.

- Subir el sueldo de todos los empleados en %10 si su sueldo actual es mayor que $1.000.000 y en %20 si es menor

$$\delta_{sueldo \leftarrow sueldo * 1,1}(\sigma_{sueldo \geq 1000000}(empleado))$$
$$\delta_{sueldo \leftarrow sueldo * 1,2}(\sigma_{sueldo < 1000000}(empleado))$$

- Cuidado con el orden!!!

nombre	*sueldo*	*cod_dept*	*fecha_ing*
Torres	$ 1.320.000	A1	01/01/2004
Soto	$ 600.000	A2	01/01/2003
Pérez	$ 360.000	A2	01/10/2003
Figueroa	$ 720.000	A1	01/03/2002
Salas	$ 1.650.000	A1	01/01/2002
Ríos	$ 2.200.000	A3	01/06/2002
Campos	$ 960.000	A2	01/11/2003
Venegas	$ 720.000	A1	01/06/2002
Carcamo	$ 600.000	A2	01/04/2003
Gonzalez	$ 2.200.000	A3	01/10/2002

Vistas

- Hasta ahora hemos operado al nivel conceptual, directamente con las tablas de la base de datos. Muchas veces es mejor que ciertos usuarios tenga acceso a ciertos datos o vistas de los datos.
- Por ejemplo en una aplicación para fijar reuniones, los empleados deben poder ver los nombres de los empleados y los departamentos a los que pertenecen pero no sus sueldos.

$$\pi_{emp.nombre,emp.cod\quad dept,dept.nombre}(empleado \bowtie departamento)$$

- Para crear una vista usaremos conceptualmente la sentencia create view de la siguiente forma:

create view *v* as *E*

dónde *v* es el nombre de la vista a crear y *E* es una expresión de consulta.

- Una vista puede usarse como cualquier relación efectiva de la base de datos, pero cuidado una vista NO es una nueva relación, es simplemente una forma abreviada de referirse a una consulta. La vista se *instancia* cada vez que nos referimos a ella.
- Ejemplo:

create view *importantes* as

$$\pi_{nombre,cod\_dept}(\sigma_{sueldo \geq 1000000}(empleado))$$

importantes

nombre	*cod_dept*
Torres	A1
Salas	A1
Ríos	A3
Gonzalez	A3

Actualización de Vistas y Valores Nulos

- Las vistas son útiles para consultar datos, pero presentan problemas para actualizaciones, inserciones y borrados.

  La dificultad radica en que las modificaciones de la base de datos expresadas en términos de vistas deben traducirse en modificaciones de las relaciones reales en el modelo lógico de la base de datos.

- Por ejemplo, si escribimos

  $$importantes \leftarrow importantes \cup \{(\text{Gomez}, \text{A4})\}$$

  implica una actualización de la relación *empleado* pero sin especificar ni sueldo ni fecha de ingreso. Existen entonces dos posibilidades:

  - Rechazar la inserción y lanzar un error.
  - Insertar la tupla (Gomez, null, A4, null) en la relación *empleado.*

- El valor 'null' representa un valor nulo, desconocido o que no existe. Todas las comparaciones que implican 'null' son por definición falsas (esto último tendrá implicancias prácticas importantes).

- Un problema importante de la anterior actualización es que, a pesar de que la relación empleado tiene una nueva tupla

nombre	*sueldo*	*cod_dept*	*fecha_ing*
Torres	$ 1.200.000	A1	01/01/2004
Soto	$ 500.000	A2	01/01/2003
Pérez	$ 300.000	A2	01/10/2003
Figueroa	$ 600.000	A1	01/03/2002
Salas	$ 1.500.000	A1	01/01/2002
Ríos	$ 2.000.000	A3	01/06/2002
Campos	$ 800.000	A2	01/11/2003
Venegas	$ 600.000	A1	01/06/2002
Carcamo	$ 500.000	A2	01/04/2003
Gonzalez	$ 2.000.000	A3	01/10/2002
Gomez	null	A4	null

La vista *importantes* no cambia

Actualización de Vistas (cont.)

- El siguiente es el reultado de la vista *importantes* antes y después de la inserción si se ingresan valores nulos para los atributos no especificados.

importantes

nombre	*cod_dept*
Torres	A1
Salas	A1
Ríos	A3
Gonzalez	A3

importantes ← *importantes* ∪ {(Gomez, A4)}

importantes

nombre	*cod_dept*
Torres	A1
Salas	A1
Ríos	A3
Gonzalez	A3

- Una posible solución al problema sería haber insertado datos (mínimos) adicionales al ingresar la nueva tupla en la vista de manera de obtener el resultado deseado, o sea, haber insertado la tupla con atributo (Gomez, 1000000, A4, null) pero ¿cómo podemos tomar esta decisión automáticamente? Un tema complejo que vendrá determinado por cada relación concreta.