Bases de Datos I

Consultas de Datos
Parte 2

Facultad de Ciencias Exactas
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

INGENIERÍA DE SISTEMAS

Cursada 2019

Consultas SQL (cont)

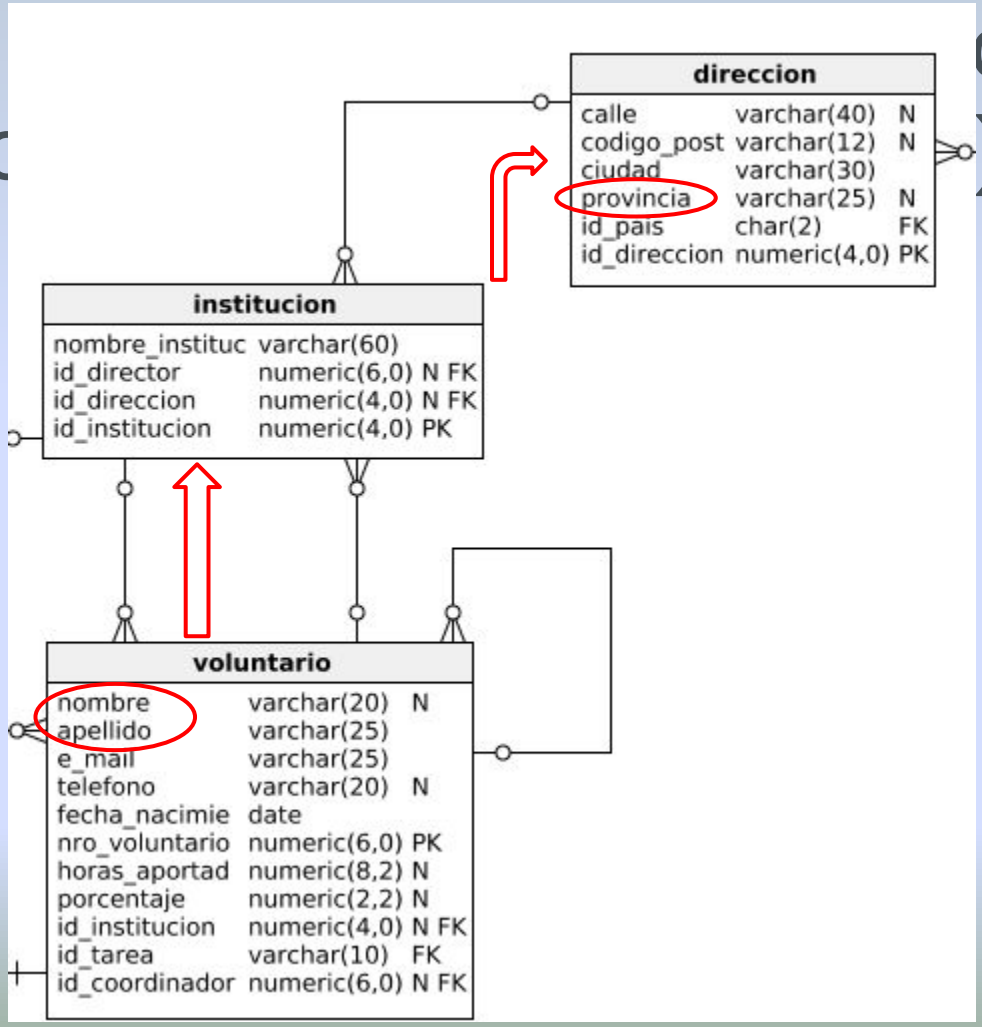
- ✓ La sentencia del lenguaje empleada para la recuperación de datos: **SELECT**.

- ✓ Sintáxis:

SELECT * | { [**DISTINCT**] columna | expresion [alias],...}
FROM lista de tablas
[WHERE condiciones**]**
[GROUP BY expresión de agrupamiento**]**
[HAVING condición de grupo**]**
[ORDER BY lista de columnas **[ASC|DESC]]**

Consultas de más de una tabla

Nos solicitan lo siguiente: seleccionar el nombre os de institucio) de Texas



Ejercicios

Liste los datos de contacto (nombre, apellido, e-mail y teléfono) de todos los voluntarios que hayan desarrollado tareas de hasta 5000 hs (max_horas-min_horas) y que las hayan finalizado antes del 24/07/1998.

Liste para cada voluntario su nro., nombre y apellido junto con el apellido de su coordinador.

Ensambles Internos

- Por medio del operador JOIN podemos combinar dos tablas según una condición para obtener tuplas compuestas por atributos de las dos relaciones combinadas. Existen diferentes maneras hacerlo:
- **INNER JOIN** (De equivalencia o Equi-join)**:** Es un caso particular de INNER JOIN en el que la condición que acota el resultado es una comparación de igualdad.
- **NATURAL JOIN:** Es un caso especial de equi-join en el que en el caso de existir columnas con el mismo nombre en las relaciones que se combinan, sólo se incluirá una de ellas en el resultado de la combinación.
- Si los nombres de columnas se repiten, hay que anteponer el nombre de la tabla para evitar ambigüedades.

Ensambles Internos

- Seleccionar el nombre y apellido de los voluntarios del estado (provincia) de Texas

```
SELECT nombre, apellido

FROM voluntario v INNER JOIN institucion i
        ON (v.id_institucion = i.id_institucion)

        NATURAL JOIN direccion d
WHERE d.provincia = 'Texas'
```

Mucho Cuidado con el Natural JOIN.
Listar todos los datos de los Departamentos de los Distribuidores Nacionales (Esquema de Películas)

Ejercicios utilizando join

Liste los datos de contacto (nombre, apellido, e-mail y teléfono) de todos los voluntarios que hayan desarrollado tareas de hasta 5000 hs (max_horas-min_horas) y que las hayan finalizado antes del 24/07/1998.

Liste para cada voluntario su nro., nombre y apellido junto con el apellido de su coordinador.

Consultas de más de una tabla

- Seleccionar el nombre y apellido de los voluntarios del estado (provincia) de Texas

```
SELECT nombre, apellido
FROM voluntario v, institucion i, direccion d

WHERE v.id_institucion = i.id_institucion
  AND i.id_direccion = d.id_direccion

  AND d.provincia = 'Texas'
```

Condiciones de ensamble (cantidad de tablas – 1)

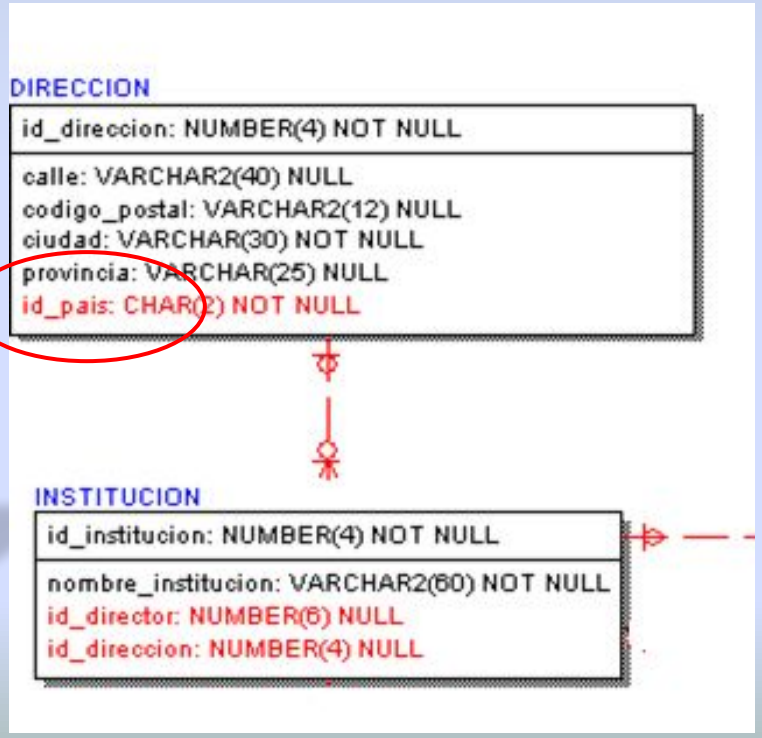
Consultas Anidadas

- La cláusula WHERE puede contener un SELECT anidado o subconsulta. Como una consulta en 2 pasos.
- Ejemplo: seleccionar el nombre de la/s instituciones del estado (provincia) de Texas.

```
SELECT id_direccion
FROM direccion d
WHERE d.provincia = 'Texas';

Resultado = 1400

SELECT nombre_institucion
FROM institucion i
WHERE i.id_direccion = 1400;
```



Consultas con subconsultas de una fila

Seleccionar el nombre de la/s instituciones del estado (provincia) de Texas.

```
SELECT nombre_institucion
FROM institucion i
WHERE i.id_direccion = (SELECT id_direccion
                        FROM direccion d
                        WHERE d.provincia = 'Texas');
```

- **Muy importante…..**que ocurriría si hay más de un resultado para Texas?
  - Usar operadores de fila única o **single-rows (=, >, <, <>, >=, <=)** para subconsultas que retornan una fila
  - Usar operadores de múltiples filas o **multiple-rows** (**IN, ANY, ALL**) para subqueries que retornan varias filas

Consultas con subconsultas de una fila

```
SELECT nombre, apellido
FROM   voluntario
WHERE horas_aportadas = (SELECT MIN(horas_aportadas)
                         FROM voluntario);
```

```
SELECT id_tarea, MIN(horas_aportadas)
  FROM voluntario
GROUP BY id_tarea
HAVING MIN(horas_aportadas) > (SELECT AVG(horas_aportadas)
                               FROM voluntario);
```

- Es responsabilidad de quien escribe el SELECT asegurar que la subconsulta devolverá una sola fila. Si el subquery devuelve más de una fila, dará error de ejecución

Consultas con subconsultas de varias filas

OPERADOR	SIGNIFICADO
IN	Retorna TRUE si el valor está incluido en los valores retornados por la subconsulta
EXISTS	Retorna TRUE si el subquery devuelve al menos una fila. FALSE si devuelve 0 filas
ANY	Retorna TRUE si la comparación es TRUE para al menos un valor retornado por el subquery
ALL	Retorna TRUE si la comparación es TRUE para todos los valores retornados por el subquery

Consultas con subconsultas de varias filas

Operador [NOT] IN

Permite determinar si los valores de una columna, o conjunto de ellas, están contenidos (o no) en una lista definida o dentro de otra tabla(subconsulta).

Seleccionar el nro., nombre y apellido de los voluntarios que realizan tareas cuyo nombre comienza con ORG

```
SELECT nro_voluntario, nombre, apellido
FROM voluntario
WHERE id_tarea IN
        (SELECT id_tarea
        FROM tarea
        WHERE nombre_tarea LIKE 'ORG%');
```

De igual manera se pueden seleccionar los voluntarios que no realizan dicha tarea con NOT IN (si nombre_tarea no permite nulos)

Operador IN

- La condición del operador IN puede plantearse indicando la lista de valores dada por extensión, entonces

  IN (v1, v2, …., vn) es equivalente a

  ((x=v1) or (x=v2) …or (x=vn))

- De igual manera el operador

  NOT IN (v1, v2, …., vn) es equivalente a

  NOT ((x=v1) or (x=v2) …or (x=vn))

Ejercicios

Determinar los nros., nombres y apellidos de los voluntarios que son coordinadores y cuyas instituciones donde se desempeñan se encuentren en ‘Estados Unidos’. Ordene los datos por los ids.

Consultas con subconsultas de varias filas

Operador [NOT] EXISTS

Se refiere únicamente a la determinación de si la subconsulta devuelve alguna fila.

Si devuelve una o más filas, el predicado se evalúa como verdadero; de lo contrario, el predicado se evalúa como falso.

Tiene la forma de una **consulta correlacionada**:

```
SELECT lista de atributos
FROM tabla_externa E
WHERE EXISTS
        (SELECT 1
        FROM tabla_interna I
        WHERE E.atributo1 = I.atributo1
            …..
            AND E.atributoN = I.atributoN);
```

Operador EXISTS

El operador EXISTS comprueba la existencia de filas en el conjunto de filas del resultado de la subconsulta.

- Si se encuentra un valor de fila de la subconsulta entonces:
  - Se coloca la condición como verdadera
  - La búsqueda no continúa en la consulta interna
- Si no se encuentra un valor de fila en la subconsulta entonces:
  - Se coloca la condición como falsa
  - la búsqueda continúa en la consulta interna para el siguiente valor de la tabla externa.

Seleccionar el nro., nombre y apellido de los voluntarios que realizan tareas cuyo código comienza con ORG

```
SELECT nro_voluntario, nombre, apellido
FROM voluntario V
WHERE EXISTS
        (SELECT 1
        FROM tarea T
        WHERE V.id_tarea = T.id_tarea
            AND nombre_tarea LIKE 'ORG%');
```

Operador NOT EXISTS

De igual forma el para NOT EXISTS comprueba que no existan filas en el conjunto de filas del resultado de la subconsulta interna.

Seleccionar el nro., nombre y apellido de los voluntarios que NO realizan tareas cuyo nombre comienza con ORG

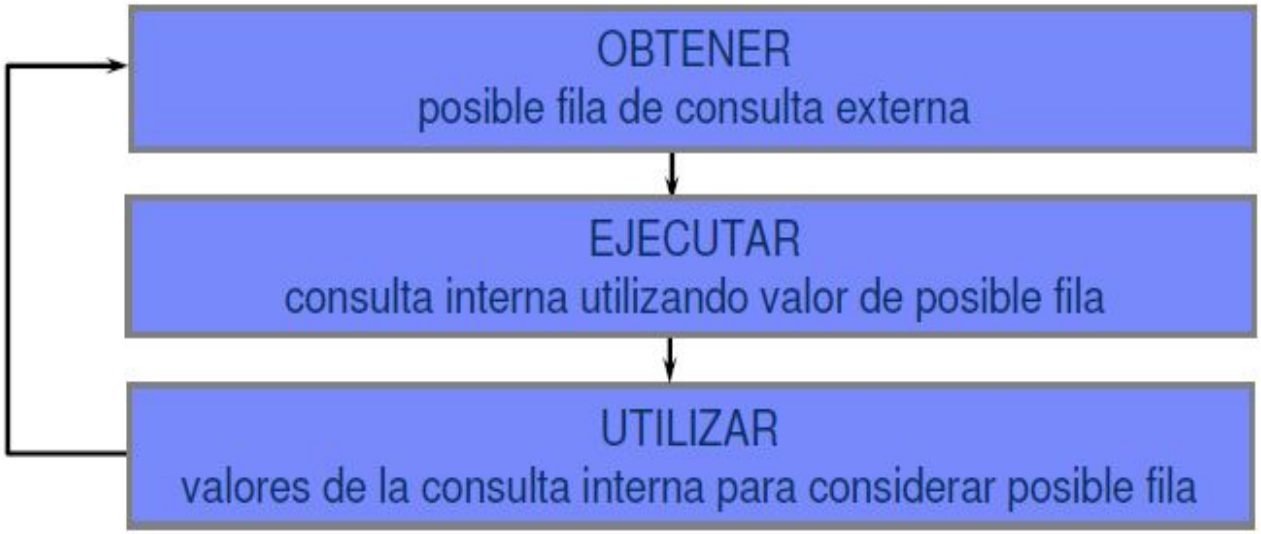
```
SELECT nro_voluntario, nombre, apellido
FROM voluntario V
WHERE NOT EXISTS
        (SELECT 1
        FROM esq_vol_tarea T
        WHERE V.id_tarea = T.id_tarea
            AND nombre_tarea LIKE 'ORG%');
```

La sintaxis de IN y EXISTS deberían dar los mismos registros si la consulta está bien construida, aunque la ejecución sean diferente.

NOT IN y NOT EXISTS no son sinónimos. El valor NULL determina la diferencia.

Subconsultas Correlacionadas

Se utilizan para el procesamiento fila a fila. Cada subconsulta se ejecuta una vez para cada fila de la consulta externa.




```
SELECT columna1, columna2, ...
FROM   tabla1  externa
WHERE  columna1 <operador>
                      (SELECT  columa1, columna2
                       FROM    tabla2 intera
                       WHERE   expr1 =
                                 externa.expr2);
```

Uso del operador EXISTS

Seleccionar el nro., nombre y apellido de los voluntarios que son [que no son] coordinadores

```
SELECT nro_voluntario, nombre, apellido
FROM voluntario V1
WHERE [NOT] EXISTS
        (SELECT 1
        FROM voluntario V2
        WHERE V1.nro_voluntario = V2.id_coordinador);
```

Ensamble Externo

El ensamble interno (*inner join*) sólo se queda con las filas que tienen valores idénticos en las columnas de las tablas que compara.

Pero….puede suceder que perdamos alguna fila interesante de alguna de las dos tablas; por ejemplo, porque poseen valores NULL…..

Por esto se SQL dispone del ensamble externo (*outer join*), que nos permite obtener todos los valores de la tabla que hemos puesto a la derecha, los de la tabla que hemos puesto a la izquierda o los valores de ambas tablas.

Su formato es:

```
SELECT nombre_columnas_a_seleccionar
FROM t1 [NATURAL] [LEFT|RIGHT|FULL] [OUTER] JOIN t2
{ON condiciones| [USING (columna [,columna...])}
[WHERE condiciones];
```

Ensamble Externo

Ensamble externo: Operador RIGHT JOIN

Listar las entregas de peliculas y todos los videos.

```
SELECT *
FROM entrega RIGHT JOIN video;
```

Ensamble externo: Operador LEFT JOIN

Listar todas las empresas productoras junto con las películas que producen:

```
SELECT *
FROM empresa_productora LEFT JOIN pelicula;
```

Ensamble Externo

Ensamble externo: Operador FULL JOIN

Listar todos los distribuidores nacionales e internacionales

```
SELECT *
FROM nacional N FULL JOIN internacional I
        ON (N.id_distrib_mayorista = I.id_distribuidor) ;
```

Listara todos los distribuidores nacionales tengan o no distribuidor mayorista y tambien se incluiran los distribuidores internacionales que no estaban asociados a ningun distribuidor nacional.

Nota: la cláusula ON la debo especificar cuando las columnas de ensamble (join) no coinsiden en nombre

Información faltante

- Información faltante fue un gran problema desde los comienzos de las BD.
- Faltante, en su sentido más amplio significa ausente.
- Las razones de la ausencia pueden ser diversas.
- La información ausente en un sistema 'manual', en papel, se manejaría, por ejemplo:

  1) dejando el espacio en blanco …

  2) dibujando una línea de tachado en ese espacio …

  3) colocando un N/A (no aplicable)… o NS/NC (no sabe/no contesta) ….

  4) Otras formas apropiadas para el caso …

Los NULOS!!!!

null no es un valor, es la ausencia de valor! Esta es la noción aceptada por SQL.

null significa valor desconocido o no existente… pero en realidad hay varias interpretaciones más, hay diferentes tipos de información faltante:

1. Atributos inaplicables (ej. ÁreaDirigida para un empleado que no es director de área)
2. Aplicable pero desconocido (ej. Comisión, para un alumno que aún no la tiene asignada)
3. Valor no existente (ej. NroPasaporte para quien no lo tiene)
4. Indefinido (ej. NotaPromedio para quien no ha rendido finales)
5. Valor no provisto (ej. NoSabe/NoContesta)
6. Valor resultante de una operación algebraica (ej. Caracteres de relleno en el ensambles externos)

‘Valores’ nulos?

Cada tipo de nulo tiene su propia semántica, características y operatoria

→ antinatural utilizar el mismo tratamiento para todos los casos

→ mal uso potencial (ej. la suma Comisión + Sueldo para un empleado que no vende dará un valor desconocido)

→ si la ausencia de un valor depende de la ausencia de otro
→ hay que definir un null nuevo!

Ej.: Si el CodSalarioFliar es null (no aplicable), entonces el ValorSalarioFliar es null (no aplicable en consecuencia del anterior)

Si depende de dos nulls → otro null con semántica más elaborada!

Tratamiento de los nulos

Surgen complicaciones cuando los valores nulos participan en operaciones aritméticas o de comparación (deberían evitarse siempre que sea posible).
Un nulo indica «*valor desconocido o no existente*», →

- cualquier operación aritmética (+, –, * , /) que los incluya debe devolver un valor nulo.

- cualquier comparación (como <, <=, >, >= y !=) que los incluya se evalúa al valor especial desconocido; no se puede decir si el resultado es verdadero o falso, es el nuevo valor lógico **desconocido**.

Lógica trivaluada

Las comparaciones que incluyen nulos pueden aparecer dentro de expresiones booleanas que incluyan operaciones AND, OR, NOT, entonces se debe definir la forma en que estas operaciones tratan el valor lógico desconocido.

LÓGICA DE TRES VALORES

VERDADERO (TRUE)
FALSO (FALSE)
DESCONOCIDO (UNKNOWN)

& (AND)	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

I (OR)	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F

~(NOT)	
T	F
U	U
F	T

Lógica trivaluada y SQL

Qué efecto tienen sobre las operaciones relacionales?

Selección → se requiere que la condición evalúe en VERDADERO, no en FALSO ni DESCONOCIDO.

Proyección → implica la eliminación de duplicados, pero null<>null, entonces dos tuplas aparentemente iguales no podrían ser eliminadas! (la misma tupla no debería ser igual a sí misma!) **pero eso NO ocurre en SQL.**

Lógica trivaluada y SQL

- En las funciones de agregado SUM(), AVG(), MIN(), MAX(), se eliminan implícitamente los valores null del cálculo.
- Para COUNT(…) no cuenta las filas en las cuales el campo entre () es null, a menos que éste sea `*´ ya que COUNT(*) devuelve el número total de filas sin depender de los valores de una columna.
- Para funciones de agregado, si cada ítem de datos en una columna es null, SUM(), AVG(), MIN(), MAX() devuelven null
- Para ORDER BY, null es el ‘mayor’
- Para DISTINCT, null es duplicado de null

Lógica trivaluada y SQL

- IN (v1, v2, …., vn) dado que se evalúa como
  ((x=v1) or (x=v2) …or (x=vn))

Entonces x=NULL evalúa a desconocido, verdadero para cualquier valor v1…vn y falso para el resto.

- NOT IN (v1, v2, …., vn) dado que se evalúa como
  ((x<>v1) and (x<>v2) …and (x<>vn))

Entonces evalúa a desconocido para null, verdadero para cualquier valor distinto de v1…vn y falso para el resto.

Que problema puede haber ?

Outer Join

- El ensamble común (interno) de 2 relaciones combina tuplas que tienen elementos comparables.

Las tuplas sin pareja no se muestran en el resultado!!

- El ensamble externo (outer join) incluye las tuplas sin pareja en el resultado y rellena con NULL los valores inexistentes
  - Left outer join
  - Right outer join
  - Full outer join

Valores por defecto

- Son una alternativa al uso de NULLs
  - Si un valor es desconocido, se rellena con un valor en particular
  - Son valores, por lo tanto no es necesario aplicar la Lógica Trivaluada
- Suelen tener más significado que los NULLs
  - 'ninguno', 'desconocido', 'ns/nc', 'no aplicable'
- Ejemplos
  - ZZZ para los nombres
  - -1 para peso, precio, cantidad, etc. (Observar que -1 no sería considerado accidentalmente como válido, sin embargo que pasaría con una operación UPDATE Producto SET Peso = Peso + 3 ??)

Problemas con valores por defecto

- Ya que son valores 'reales'…
  - Pueden actualizarse como cualquier otro valor.
  - Se necesita encontrar un valor que no pudiese ser utilizado en ninguna circunstancia (distinguible)
  - Se corre el riesgo de una interpretación incorrecta.
  - En SQL los valores por defecto deben ser del mismo tipo que los valores de la columna (por ej. No se podría usar 'desconocido' en una columna de enteros!!)

Partición de tablas

- NULLs y valores por defecto tratan de rellenar datos faltantes con elementos dato distinguibles de los reales.
  - NULLs indican que el dato no está, utilizando el mismo elemento de relleno para cualquier tipo de datos.
  - Los valores por defecto indican que la información falta colocando un elemento dato de la misma clase del faltante pero distinguible.

- A menudo pueden removerse tuplas que tienen datos faltantes.
  - Se pueden separar las columnas que tienen nulos de manera que queden en otra tabla.
  - Las subtuplas que contienen nulos no se incluyen en las tablas generadas.

Problemas con partición de tablas

- Se introducen nuevas tablas
  - La información se disemina entre estas tablas
  - Las consultas se vuelven más complejas y se requieren más ensambles
- Se puede recuperar la tabla original, pero…
  - Se necesita aplicar outer joins …
  - Esto introduce NULLs, que retrotrae a todos los problemas ya analizados, nuevamente !!

Qué método usar?

- Puede ser una decisión personal .. pero debería tenerse en cuenta que:
  - Los valores por defecto **NO** deberían usarse cuando pueden 'confundirse' con valores reales
  - Partir tablas **NO** debería utilizarse en demasía pues aumentaría la cantidad de tablas en el esquema lleva a tener más ensambles a la hora de las consultas …
  - Los NULLs pueden ser (y frecuentemente son) usados cuando las otras alternativas parecen inapropiadas.
  - No es necesario utilizar siempre el mismo recurso…se pueden combinar de forma inteligente!!

Siempre tener presente donde se pueden encontrar nulos!!!
Revisar en el planteo de sentencias las sentencias SELECT