Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas

Departamento de Informática

Bases de Datos

Guía de Trabajo Nro. 3

SQL: Consultas avanzadas

Msc. Lic. Hugo Minni

2009

Consultas que involucran más de una tabla
INNER JOIN

Cuando necesitamos una consulta que retorne datos de más de una tabla, debemos enlazar las mismas a través de las columnas que las asocian en el modelo físico (*Primary Keys* y *Foreign Keys*). A esto le denominamos *unión equidistante* (`INNER JOIN`). A la columna o columnas en común las denominamos *columnas de unión*.

Supongamos que deseamos obtener el título de las publicaciones junto al nombre de la editorial que las publicó. Estos datos residen en dos tablas diferentes, por lo tanto es necesario establecer un `INNER JOIN`.

Las columnas en común que establecen la asociación se especifican a través de la cláusula `ON`:

```
SELECT title, pub_name
   FROM titles INNER JOIN publishers
                  ON titles.pub_id = publishers.pub_id
```

Alias de tabla

En la *Guía de Trabajo Nro. 1* aprendimos a utilizar *alias de columna*. También es posible especificar *alias de tabla*.
Los alias de tabla tiene por objeto generalmente abreviar la referencia a una tabla que debemos repetir varias veces en una sentencia `SELECT`.
Especificamos un alias de tabla en las cláusulas `FROM` e `INNER JOIN` directamente después del nombre de la tabla. Por ejemplo:

```
SELECT title, T.pub_id, pub_name
   FROM titles T INNER JOIN publishers P
                    ON T.pub_id = P.pub_id
```

1. Obtenga un listado de apellidos y nombres de autores junto a los títulos de las publicaciones de su autoría. Ordene el listado por apellido del autor.

2. Obtenga un listado que incluya los nombres de las editoriales (tabla `publishers`) y los nombres y apellidos de sus empleados (tabla `employee`) pero sólo para los empleados con `job level` de 200 o más.

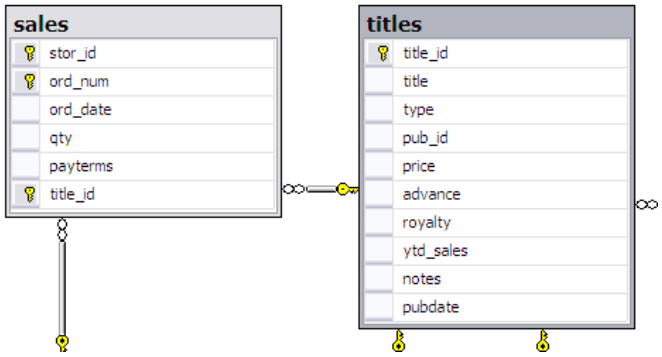
La cláusula `GROUP BY`

Recordemos que la cláusula `GROUP BY` nos permite agrupar filas en base a los valores de una o más columnas. `GROUP BY` divide el resultado de la sentencia `SELECT` en un conjunto de grupos, de manera que para cada grupo de más de una fila los valores de la o las columnas que definen el grupo son idénticos. `GROUP BY` se ubica luego de la cláusula `WHERE`:

```
SELECT columna1
   FROM tabla
   GROUP BY columna1
```

Debemos tener en cuenta que, una vez definidos grupos en SQL, solo podemos especificar *atributos de grupo* en la *lista de salida* del `SELECT`.
Por ejemplo, si definimos grupos a partir de los tipos de publicación (columna `type`) en la tabla `titles`, *atributos de grupo* serán la cantidad de publicaciones de cada tipo, la suma de los precios de cada tipo de publicación, etc.

3. La tabla `sales` contiene información de ventas de publicaciones. Cada venta es identificada unívocamente por un número de orden (`ord_num`), el almacén donde se produjo (`stor_id`) y la publicación vendida (`title_id`).
La venta posee también la fecha de venta (`ord_date`) y la cantidad vendida de la publicación (`qty`).



Se desea obtener un listado como el siguiente, que muestre los ingresos (precio de publicación * cantidad vendida) que ha proporcionado cada autor a partir de las ventas de sus publicaciones. Ordene el listado por orden descendente de ingresos.
Cuales son los atributos de grupo?.

au_lname	au_fname	Ingresos
Ringer	Albert	1357.6000
Ringer	Anne	1302.2000
Dull	Ann	1000.0000
Hunter	Sheryl	1000.0000
Panteley	Sylvia	838.0000
MacFeather	Stearns	730.5500

Condiciones de grupo
La cláusula HAVING

De manera análoga a como WHERE nos permite especificar una condición que deben satisfacer las filas que formarán parte de la lista de salida, la cláusula HAVING nos permite especificar una condición que deben cumplir los grupos para formar parte de la lista de salida:

```
SELECT columna1
   FROM tabla
   GROUP BY columna1
   HAVING condicion-de-grupo
```

Debemos tener en cuenta que, la condición de una cláusula HAVING sólo pueden incluir comparaciones contra *atributos de grupo*.

4. Obtenga los tipos de publicaciones cuya media de precio sea mayor a $12.

Subconsultas

Recordemos que una subconsulta es una sentencia SELECT anidada que es necesaria para completar la consulta principal. En el procesamiento de la sentencia completa, se evalúa primero la subconsulta y luego se aplican los resultados a la consulta principal

A la sentencia SQL principal le llamamos *outer query*, y a la sentencia SQL anidada le llamamos *inner query*.

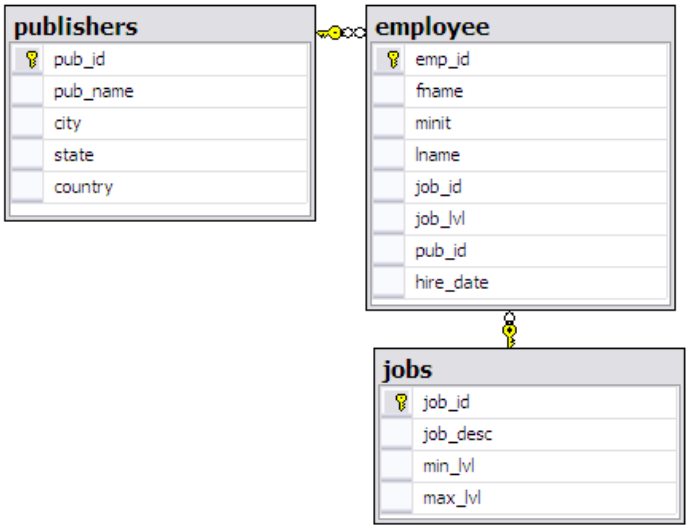
Predicado de comparación
Subconsultas de valor único

La forma más sencilla de subconsulta consiste en una sentencia SQL que retorna un *único valor escalar* que es necesario para evaluar una condición en el *outer query*. Una subconsulta de valor único posee la siguiente forma:

```
SELECT col1, col2
   FROM tabla
   WHERE col1 [=, <>, <, <=, >, >=] (inner join de valor único)
```

Lo que tenemos en la cláusula WHERE del *outer query* es un **predicado de comparación** entre una *columna* o *expresión de valor* y una *subconsulta de valor único*. Recordemos que los *tipos de datos* de la expresión de valor y la subconsulta de valor único deben ser comparables.

5. La tabla `employee` posee información de los empleados de cada editorial. Por cada empleado tenemos su identificación (`emp_id`), su nombre (`fname`) y apellido (`lname`). Cada empleado pertenece a una editorial (`pub_id`) y posee una fecha de contratación (`hire_date`). Las funciones de los empleados se describen en la tabla `jobs`. Cada empleado posee una función (`job_id`).



Obtenga el apellido y nombre del empleado contratado más recientemente.

El predicado `[NOT] IN`

Vimos el Predicado `IN` en la *Guía de Estudio Nro. 1*. Recordemos que `IN` toma un valor y observa si el mismo se encuentra en un conjunto de valores comparables, que expresábamos en una lista de valores encerrados entre paréntesis:

```
SELECT address
   FROM authors
   WHERE au_id IN ('172-32-1176', '238-95-7766')
```

SQL permite que el conjunto de valores comparables sea el resultado de la ejecución de una *subconsulta*:

```
WHERE columna-o-expresion-de-valor [NOT] IN (SELECT expresion-de-columna
                                             FROM tabla)
```

En el caso del ejemplo:

```
SELECT address
   FROM authors
   WHERE au_id IN (SELECT au_id
                   FROM authors
                   WHERE au_id = '172-32-1176' OR
                         au_id = '238-95-7766')
```

6. Obtenga un listado de nombres de editoriales que han editado publicaciones de tipo `business`. Esta consulta se puede resolver con un `INNER JOIN` o con un predicado `IN`. Resuélvala de ambas maneras.

7. Obtenga un listado de títulos de las publicaciones que no se vendieron ni en `1993` ni en `1994`. (Columna `ord_date` en tabla `Sales`)

`INNER JOIN` o predicado `IN`

Podemos vernos tentados a resolver esta consulta a través de un `INNER JOIN` entre las tablas de títulos (para obtener el título de la publicación) y la de ventas (a fin de evaluar la condición de fecha de la venta). La consulta sería la siguiente:

```
SELECT titles.title_id, title
   FROM titles INNER JOIN sales
                  ON titles.title_id = sales.title_id
   WHERE YEAR(sales.ord_date) NOT IN (1993, 1994)
```

Sin embargo, esta solución no retorna los datos esperados, ya que obtenemos el listado de las publicaciones que tuvieron ventas en años diferentes a `1993` y `1994`, pero no obtenemos las publicaciones que no tuvieron ventas en absoluto. Ante requerimientos de este tipo, la consulta se debe resolver a través de un predicado `IN`.

Subconsultas correlacionadas

Recordemos que una subconsulta correlacionada es una subconsulta (*inner query*) que contiene referencias a una consulta externa (*outer query*), y que los resultados que arroje están correlacionados con –o dependen de- **cada fila individual** de la consulta principal.

8. Obtenga un listado como el siguiente con las publicaciones que poseen un precio menor que el promedio de precio de la editorial a la que pertenecen.

title	pub_name	price
You Can Combat Computer Stress!	New Moon Books	2.9900
Life Without Fear	New Moon Books	7.0000
Emotional Security: A New Algorithm	New Moon Books	7.9900
The Gourmet Microwave	Binnet & Hardley	2.9900
Fifty Years in Buckingham Palace Kitchen	Binnet & Hardley	11.9500
Sushi, Anyone?	Binnet & Hardley	14.9900
Cooking with Computers: Surreptitious Ba	Algodata Infosystems	11.9500

El predicado `[NOT] EXISTS`

Recordemos que el predicado `EXISTS` es un cuantificador existencial que nos permite evaluar si el resultado de una subconsulta existe o por el contrario la misma proporciona un conjunto vacío:

```
SELECT columna
   FROM tabla
   WHERE [NOT] EXISTS (subconsulta)
```

`EXISTS (no-vacío)` evalúa verdadero.
El predicado `EXISTS` siempre precede a una subconsulta, y esta subconsulta es siempre una *subconsulta correlacionada*.

Retomando el ejercicio 7 (títulos de las publicaciones que no se vendieron ni en `1993` ni en `1994`), podemos también resolverlo a través del uso del predicado `EXISTS`. El resultado es idéntico al que obtendríamos con un `INNER JOIN`, es decir, NO obtenemos las publicaciones que no tuvieron ventas en absoluto:

```
SELECT title, titles.title_id
   FROM titles
   WHERE EXISTS (SELECT *
                     FROM sales
                     WHERE sales.title_id = titles.Title_id AND
                           YEAR(ord_date) NOT IN (1993, 1994)
                 )
```

9. Como modificaría esta última consulta a fin de incluir también las publicaciones que no se vendieron nunca?.

La expresión `CASE` simple

La expresión `CASE` simple compara un valor contra una lista de valores y retorna un resultado asociado al primer valor coincidente:

```
CASE expresion0
   WHEN expresion1 THEN ResultadoExpresion11
   [ [WHEN expresion2 THEN ResultadoExpresion2]
     [...]
   [ ELSE ResultadoExpresionN]
End
```

La `expresion0` se compara contra `expresion1`, `expresion2`, etc. hasta que se encuentra una coincidencia o se encuentra el final de la lista de expresiones. Si `CASE` encuentra coincidencia, retorna el `ResultadoExpresion` correspondiente. Si no encuentra coincidencia alguna, retorna `ResultadoExpresionN`.

10. La tabla `authors` posee una columna llamada `contract` con valores `0` ó `1` indicando si el autor posee o no contrato con la editora. Se desea obtener un listado como el siguiente para los autores de California (columna `state` con valor `CA`).

Nombre	Apellido	Posee contrato?
Johnson	White	Si
Marjorie	Green	Si
Cheryl	Carson	Si
Michael	O'Leary	Si
Dean	Straight	Si
Abraham	Bennet	Si

La expresión `CASE` searched

Esta forma de expresión `CASE` permite el análisis de varias condiciones lógicas y retorna un resultado asociado a la primera que evalúa a verdadero:

```
CASE
   WHEN condicion1 THEN expresion11
   [ [WHEN condicion2 THEN expresion2]
     [...]]
   [ ELSE expresionN]
End
```

Si se encuentra una condición verdadera, el resultado es la expresión correspondiente. Si todas las condiciones son falsas, el resultado es `expresionN`.

11. La columna `job_lvl` indica el puntaje del empleado dentro de su área de especialización. Se desea obtener un reporte como el siguiente, ordenado por puntaje y apellido del empleado:

lname	Nivel
Ashworth	Puntaje entre 100 y 200
Brown	Puntaje entre 100 y 200
Domingues	Puntaje entre 100 y 200
Chang	Puntaje mayor que 200
Cramer	Puntaje mayor que 200
Devon	Puntaje mayor que 200
Schmitt	Puntaje menor que 100
Smith	Puntaje menor que 100
Tonini	Puntaje menor que 100

Vistas

Una vista es un objeto que permite visualizar datos en una tabla normal, pero de forma diferente. Proporcionan un método para visualizar los datos en las tablas subyacentes sin duplicarlos.
En operaciones `SELECT`, `INSERT`, `UPDATE` o `DELETE`, las vistas se comportan como tablas, con algunas restricciones.
Si bien una vista es un objeto físico independiente, no almacena datos como una tabla, sino que se puede pensar como un filtro a través del cual se puede visualizar un conjunto de datos en la base de datos.

La sintaxis para crear una vista es:

```
CREATE VIEW <Nombre-Vista> AS
   <SELECT...>
```

Las vistas pueden ser seleccionadas sin restricción. Cualquier sintaxis `SELECT` que es legal contra una tabla es legal contra una vista.
Cuando se crea una vista no es posible utilizar cláusulas `ORDER BY`. Se debe ordenar o calcular cuando se selecciona de la vista.

12. Cree una vista que muestre los títulos escritos por autores de *California* (`state = CA`). Se deben mostrar apellido, nombres y teléfono del autor, título, tipo y precio de la publicación.

13. Obtenga un listado de apellido y nombres (concatenados en una única columna) y títulos de las publicaciones de la editorial `0736` (`New Moon Books`). Las publicaciones de `New Moon Books` deben ser extraídos de una vista que posea todas las columnas de la tabla de títulos pero únicamente para las publicaciones de esta editorial.