SQL

SQL (Structured Query Language: Lenguaje Estructurado de Consultas) es el lenguaje que usas para emitir las instrucciones al Servidor Oracle. Es, de hecho, el lenguaje estándar usado por todos los vendedores de bases de datos relacionales importantes, y Oracle cumple a Nivel de Entrada con SQL92, el mas reciente estándar internacional ISO (International Standards Organization: Organización de los Estándares Internacionales). Hay varios aspectos de SQL que pueden diferir de los lenguajes de computación con los que estás familiarizados, tales como los siguientes:

- SQL no es procedimental. En SQL, le dices al Servidor que hay que hacer pero no cómo se debe hacer eso. Esto te libera de lidiar con un montón de detalles.
- Las sentencias SQL son independientes unas de otras. Aunque PL/SQL se dirige hacia esto, SQL en sí mismo no tiene sentencias condicionales u otras de control de flujo
- SQL emplea un set operaciones a la vez. Opera sobre colecciones de datos arbitrariamente grandes en un solo paso.
- SQL usa Null y una Lógica de Tres Valores. En la mayoría de los Lenguajes, las expresiones Booleanas son evaluadas TRUE o de lo contrario FALSE. En SQL, existen TRUE, FALSE, o NULL. Esto será explicado en breve.

Recuperando los Datos

Suponga que necesita sacar de la tabla Customers la información sobre el cliente llamado "Peel". Esto se llama hacer una consulta. Para hacerlo, podría expedir la siguiente sentencia:

```
SELECT *
    FROM Customers
    WHERE LNAME = 'Peel';
```

Esto produce lo siguiente:

CNUM	FNAME	LNAME	ADDRESS
4005	Julia	Peel	197 Myrtle Court, Brisbane, CA

Oracle interpreta la sentencia como sigue: Cualquier número de espacios en blanco y/o saltos de líneas son equivalentes a un espacio en blanco o salto de línea. Estos son delimitadores, y los saltos de líneas y espacios extra están por legibilidad: todos son equivalentes a un “espacio en blanco”. Igualmente, no es significante si las sentencias están en letras mayúsculas o minúsculas, excepto en las cadenas literales como la que se está buscando ('Peel').

SELECT es una palabra clave que le dice a la base de datos que esto es una consulta. Todas las sentencias SQL comienzan con palabras claves. El asterisco significa que se recuperen todas las columnas; altemativamente podrías haber listado las columnas deseadas por sus nombres, separados por comas. La cláusula FROM Customers identifica a la tabla desde la cuál quieres sacar los datos.

WHERE LNAME = `Peel' es el predicado. Cuando una sentencia SQL contiene un predicado, Oracle testea el predicado contra cada registro de la tabla y realiza la acción (en este caso, SELECT) sobre todos los registros que hacen al predicado verdadero (TRUE). Este es un ejemplo de un set operaciones realizadas a la vez. El predicado es opcional, pero en su ausencia la operación se realiza sobre la tabla entera, así que, en este caso, debería haber sido recuperada toda la tabla. El punto y coma es el terminador de la sentencia.

Nulls y Lógica de Tres Valores

Con los predicados, deberías ser consciente de la Lógica de Tres Valores. En SQL, los valores Booleanos básicos de TRUE y FALSE están suplementados con otro: NULL, también llamado UNKNOWN. Esto es porque SQL reconoce que los datos pueden estar incompletos o inaplicables y que el valor de veracidad de un predicado puede por lo tanto no ser conocido. Específicamente, una columna puede contener un null, lo cuál significa que no hay un valor aplicable conocido. Una comparación entre dos valores usando operadores relacionales – por ejemplo, a = 5 – normalmente es TRUE o FALSE. Cuando los nulls son comparados con otros valores, no obstante, incluyendo a otros nulls, el valor Booleano no es ni TRUE ni FALSE pero en si mismo es NULL.
Algo mas al respecto, NULL tiene el mismo efecto que FALSE. La excepción mas importante es que, mientras NOT FALSE = TRUE, NOT NULL = NULL. En otras palabras, si conoces que una expresión es FALSE, y la niegas (tomas lo contrario de) esta, entonces sabes que eso es TRUE. Si no conoces si la expresión es TRUE o FALSE, y la niegas, todavía no sabes su valor. En ciertos casos, la lógica de tres valores puede crear problemas con tu lógica de programación si no la tomas en cuenta.

Creando Tablas

Aquí está cómo creas las tablas en SQL. Puedes usar la siguiente sentencia SQL para crear la tabla Customers:

```
CREATE TABLE Customers
        (cnum       integer NOT NULL PRIMARY KEY,
        FNAME       char(15) NOT NULL,
        LNAME       char(15) NOT NULL,
        ADDRESS     varchar2);
```

Después de las palabras clave CREATE TABLE sigue el nombre de la tabla y entre paréntesis la lista de sus columnas con la definición de cada una. Integer, char, y varchar2 son los tipos de datos: todos los datos de una columna dada tienen siempre el mismo tipo de dato (varchar2 significa una cadena de caracteres de longitud variable y char de tamaño fijo).
NOT NULL y PRIMARY KEY son “constraints”, es decir, restricciones sobre las columnas en las que aparecen. Estas restringen los valores que puedes ingresar en aquellas columnas. Específicamente, NOT NULL te prohíbe la entrada de nulls en la columna. PRIMARY KEY te previene de la entrada duplicada de valores dentro de la columna y hace que la columna sea apta para ser un padre para alguna clave foránea.

Propiedad y Convenciones de Nombrado

Note que cuando creas una tabla en SQL, te apropias de ésta. Esto significa que generalmente tienes el control sobre quién tiene acceso a ella, y que es parte de un “schema”, es decir, un esquema que tiene tu nombre de usuario en Oracle. Un esquema es una colección de objetos de la base de datos nombrado bajo el control de un único usuario de Oracle. Los esquemas heredan los nombres de sus dueños. Cuando otros usuarios se refieren a un objeto que tu has creado, ellos tienen que preceder su nombre por el nombre del esquema seguido de un punto (sin espacios). SQL utiliza una convención de nombrado jerárquico con los niveles de la jerarquía separados por puntos. De Hecho, a veces tienes que preceder los nombres de las columnas por los nombres de las tablas para evitar ambigüedades, en tales casos también usas el punto. El siguiente es un ejemplo de la forma nombreEsquema.nombreTabla.nombreColumna:

scott.Customers.LNAME

Puedes simplificar las referencias como ésta usando sinónimos, que son alias para las tablas u otros objetos de la base de datos. Los sinónimos pueden ser privados, lo que significa que son parte de tu esquema y controlas su uso, o públicos, significa que todo los usuarios pueden tener acceso al mismo. Por ejemplo, puedes crear un sinónimo “Cust” para scott.Customers como sigue:

CREATE SYNONYM Cust FOR scott.Customers;

Este podría ser un sinónimo privado, que es como se crea por omisión. Ahora podrías reescribir el ejemplo de arriba como esto:

Cust.LNAME

Aún tienes que referirte a las columnas directamente. Los sinónimos solamente se pueden hacer para las tablas, no para los componentes de las tablas como las columnas.

Insertando y Manipulando los Datos

Cuáles son las sentencias SQL que determinan el contenido actual de los datos?

Principalmente, tres – la sentencia INSERT, la sentencia UPDATE, y la sentencia DELETE. INSERT ubica un registro en una tabla, UPDATE cambia los valores que contiene un registro, y DELETE elimina un registro de una tabla.

La Sentencia INSERT

Para INSERT, simplemente identificas la tabla y sus columnas y listas los valores, como sigue a continuación:

```
INSERT INTO Customers (cnum, FNAME, LNAME)
    VALUES (2004, 'Harry', 'Brighton');
```

Esta sentencia inserta un registro (fila) con un valor para cada columna excepto ADDRESS. Ya que, en tu sentencia CREATE TABLE no pusiste la restricción NOT NULL sobre la columna ADDRESS, y puesto que no diste aquí un valor para esa columna, Oracle pone en esta columna null. Si estás insertando un valor dentro de cada columna de la tabla, y tienes los valores ordenados como están las columnas en la tabla, puedes omitir la lista de columnas. Opcionalmente puedes poner una sentencia SELECT en lugar de la cláusula VALUES de la sentencia INSERT para recuperar los datos desde otro lugar en la base de datos y duplicarlos aquí.

La Sentencia UPDATE

UPDATE es similar a SELECT en que toma un predicado y opera sobre todos los registros que hacen verdadero, TRUE, al predicado. Por ejemplo:

```
UPDATE Customers
    SET ADDRESS = null
    WHERE LNAME = 'Subchak';
```

Esto pone en null a todas las “addresses”, es decir, direcciones de los clientes llamados `Subchak'. La cláusula SET del comando UPDATE puede referirse a los valores actuales de la columna. En este caso "Current" significa los valores en la columna antes de que haya ocurrido cualquier cambio por esta sentencia.

La Sentencia DELETE

DELETE es bastante similar a UPDATE. La siguiente sentencia elimina todos los registros (filas) para los clientes llamados `Subchak':

```
DELETE FROM Customers
    WHERE LNAME = 'Subchak';
```

Puedes eliminar solamente registros enteros, no valores individuales. Para hacer esto, use UPDATE para poner los valores en null. Sea muy cuidadoso con DELETE en no omitir el predicado; esto vacía la tabla.

Consultando Múltiples Tablas Mediante Joins, es decir Junturas

Aunque solo recupera datos, SELECT es la sentencia mas compleja en SQL. Una razón de esto es que puedes usarla para consultar cualquier número de tablas en una sentencia, correlacionando los datos de varias maneras. Una forma de hacer esto es con un “join”, es decir una juntura, que es una sentencia SELECT que correlaciona los datos a partir de más de una tabla. Una juntura encuentra cada combinación posible de registros, de manera tal que un registro es tomado desde cada tabla juntada. Esto significa que tres tablas de 10 registros cada una pueden producir un millar de registros de salida (10 * 10 * 10) cuando son juntados. Típicamente, usas el predicado para filtrar la salida en términos de

alguna relación. El tipo mas común de juntura, llamado juntura natural, filtra la salida en términos de la relación entre "foreign key", es decir la clave foránea/"parent key", es decir la clave padre explicadas previamente en este apéndice. Por ejemplo, para ver las personas en la tabla Customers junto con sus números de teléfono varios desde la tabla Customers_Phone, podrías ingresar lo siguiente:

```
SELECT a.CNUM, LNAME, FNAME, PHONE, TYPE
    FROM Customers a, Customer_Phone b
    WHERE a.CNUM = b.CNUM;
```

En la sentencia de arriba, a y b son variables de rango, también denominadas variables de correlación. Ellas son sencillamente nombres alternos para las tablas cuyos nombres siguen a la cláusula FROM, así que a = Customers y b = Customers_Phone. Puedes ver que aquí necesitas las variables de rango para distinguir a Customers.CNUM de Customers_Phone.CNUM en las cláusulas SELECT y WHERE. Aún cuando no sean necesarias, las variables de rango son a menudo convenientes.

Aquí está la salida de la juntura natural:

CNUM	LNAME	FNAME	PHONE	TYPE
4005	Peel	Julia	375-296-8226	home
4005	Peel	Julia	375-855-3778	beeper
4008	Lopez	Emilio	488-255-9011	home
4008	Lopez	Emilio	488-633-8591	work
4011	Lim	Kerry	577-936-8554	home

Esta salida representa cada combinación de registros a partir de las dos tablas donde ambos registros tienen el mismo valor de CNUM .

Junturas Externas

Observe en el ejemplo anterior que las personas de la tabla Customers que no tienen números de teléfono (a saber, CNUM 4007) no fueron seleccionadas. Si un registro no tiene uno correspondiente en la otra tabla, el predicado nunca es verdadero para aquel registro. A veces, no deseas este efecto, y puedes anularlo usando un "outer join", es decir, una juntura externa. Una juntura externa es una juntura que incluye a todos los registros de una de las tablas juntadas, a pesar de que no hayan correspondencias en la otra tabla. Semejante juntura inserta nulls en la salida en cualesquiera de las columnas que fueron tomadas desde la tabla en la que se falló para proporcionar la combinación de la tabla juntada externamente. Aquí está la misma consulta hecha como una juntura externa:

```
SELECT a.CNUM, LNAME, FNAME, PHONE, TYPE
    FROM Customers a, Customer_Phone b
     WHERE a.CNUM = b.CNUM (+);
```

Esta es la salida de la sentencia de arriba:

CNUM	LNAME	FNAME	PHONE	TYPE
4005	Peel	Julia	375-296-8226	home
4005	Peel	Julia	375-855-3778	beeper
4007	Subchak	Terry	NULL	NULL
4008	Lopez	Emilio	488-255-9011	home
4008	Lopez	Emilio	488-633-8591	work
4011	Lim	Kerry	577-936-8554	home

Note que la única diferencia en la consulta es la adición del (+) a la cláusula WHERE. Este sigue la tabla para que los nulls sean insertados. La salida de la consulta, entonces, incluye al menos un registro para cada registro de la tabla que no tiene añadido el (+) en el predicado.
También puedes usar las sentencias SELECT para producir valores para procesarlos dentro de las consultas (estos son llamados subconsultas), y puedes realizar operaciones estándar sobre conjuntos (UNION, INTERSECTION) en la salida de la sentencia SELECT.