Universidade da Coruña
Departamento de Computación

Elementos de Oracle

Luis A. González Ares
lgares@udc.es

Elementos de Oracle – Planteamiento

Objetivos:

- Mostrar algunos elementos destacados de Oracle.

Contenido:

- Secuencias.
- Papelera de reciclaje.
- Funciones de interés.
- Variables de sustitución.
- Sinónimos.
- Índices.
- Tratamiento de datos temporales.
- Tratamiento de valores nulos.
- Consulta retrospectiva - Flashback query
- Columnas derivadas. Columnas virtuales.
- Índices invisibles.
- Tablas de solo lectura.
- Disparadores.

Oracle – Elementos

Secuencias

Cada tabla en un esquema relacional debe tener una clave primaria, esto es, que garantice que dos filas diferentes tienen valores diferentes en ella y que no toma valores nulos.

Normalmente las claves primarias toman valores numéricos, lo que redunda en ventajas en el rendimiento. Que esos valores sean o no significativos es una decisión de diseño, aunque si aportan alguna semántica siempre será beneficioso.

En situaciones en las que necesitamos garantizar que una columna toma valores diferentes sin importarnos dichos valores, como el caso de las claves subrogadas o sustitutas, Oracle posee el elemento secuencia, que asigna de forma automática valores a una columna.

Una secuencia se crea con `CREATE SEQUENCE`, que de forma abreviada es:

```
CREATE SEQUENCE <nombre_secuencia>
INCREMENT BY    <incremento>      START WITH <inicio>
```

Se accede a ella mediante:

```
<nombre_secuencia>.NEXTVAL    Incrementa el valor de la secuencia y
                              devuelve el nuevo resultado.

<nombre_secuencia>.CURRVAL    Devuelve el valor actual de la secuencia, una vez
                              que ya se ha incrementado su valor con NEXTVAL.
```

Se modifica con `ALTER SEQUENCE`. Por ejemplo el incremento se modifica con:

```
ALTER SEQUENCE <nombre_secuencia> INCREMENT BY <nuevo_incremento>
```

Oracle – Elementos

Secuencias (cont.)

Ejemplo.- Crear una secuencia que comience en 80 y se incremente de dos en dos.

Usarla luego para introducir `(80, Marketing)` en `dept9(deptno, dname)`:

```
CREATE SEQUENCE sqemp1  INCREMENT BY 2  START WITH 80;

INSERT INTO dept9 (deptno, dname) VALUES (sqemp1.NEXTVAL,'Marketing');
```

Una misma secuencia puede usarse para automatizar la asignación de valores de varias columnas de una o más tablas.

Ejemplo.- Usar la secuencia anterior para crear un nuevo empleado `(82,'GUTIERREZ')`:

```
INSERT INTO emp9 (empno, ename) VALUES (sqemp1.NEXTVAL,'GUTIERREZ');
```

La referencia de las secuencias creadas por el usuario se encuentra en la vista del catálogo `USER_SEQUENCES`.

Normalmente no se usan ni `NEXTVAL` ni `CURRVAL` en una consulta.

Además presentan limitaciones para su uso en una subconsulta, como columnas de una vista, con `DISTINCT`, `ORDER BY`, `GROUP BY`, `HAVING` y con los operadores conjuntistas: `UNION`, `INTERSECT` y `MINUS`.

Una secuencia se elimina como otro elemento cualquiera del esquema:

```
DROP SEQUENCE <nombre_secuencia>
```

Oracle – Elementos

Papelera de reciclaje

Oracle dispone de una implementación de una papelera de reciclaje en la que se almacenan las tablas eliminadas con `DROP TABLE` y los objetos asociados a las tablas eliminadas, como índices, vistas, etc.

Al realizar el `DROP TABLE` de una tabla, realmente se efectúa un cambio de nombre de la tabla y de sus objetos asociados, comenzando su nombre por `BIN`.

Los objetos eliminados pasan a la vista `USER_RECYCLEBIN` o `RECYCLEBIN` y pueden referenciarse indicando su nuevo nombre entre comillas. Podemos consultar el contenido de `RECYCLEBIN` con un `SELECT` o de forma abreviada con `SHOW RECYCLEBIN`

Sobre los objetos de la papelera no pueden realizarse sentencias DDL o DML diferentes a `SELECT`.

```
DROP TABLE  k5;

SELECT *
FROM   RECYCLEBIN;

SHOW   RECYCLEBIN
```

```
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
K5               BIN$YHGlepy+3mngQAoKLgdoNw==$0 TABLE        2009-01-14:20:40:36
```

```
SELECT * FROM "BIN$YHGlepy+3mngQAoKLgdoNw==$0";
```

Oracle – Elementos

Papelera de reciclaje (cont.)

Al realizar `DROP TABLE` realmente solo se efectúa un cambio de nombre, ya que todas las referencias físicas de la tabla se mantienen inalterables.

Los objetos de `RECYCLEBIN` se eliminan de ella al ejecutar alguna de las sentencias siguientes:

`PURGE TABLE <tabla>`	Elimina definitivamente la tabla
`PURGE INDEX <indice>`	Elimina definitivamente el índice
`PURGE RECYCLEBIN`	Elimina todos los objetos de `RECYCLEBIN`
`FLASHBACK TABLE <tabla> TO BEFORE DROP`	Restaura la tabla a su nombre original

Si la papelera almacena varias versiones de una misma tabla, o sea, si se han creado y eliminado varias tablas con el mismo nombre, la sentencia `FLASHBACK TABLE` restaura la que se ha eliminado más recientemente (LIFO).

Si despues de restaurar una tabla, posteriormente deseamos restaurar otra con el mismo nombre, debemos renombrarla:

```
FLASHBACK TABLE k0 TO BEFORE DROP;
FLASHBACK TABLE k0 TO BEFORE DROP RENAME TO kk0;
```

La papelera puede desactivarse de forma general poniendo a `OFF` el parámetro inicial `RECYCLEBIN`, o de forma puntual usando al eliminar una tabla una variante de `DROP`:

```
DROP TABLE <tabla> PURGE
```

Oracle – Elementos

Funciones de interés

Algunas funciones de interés son (véanse otras en el manual):

```
INITCAP(cadena)          Primer carácter de cada palabra a mayúsculas:
     INITCAP('MIs fIcherOs')  Mis Ficheros

UPPER/LOWER(cadena)      Pasa a mayúsculas/minúsculas:
     UPPER('MIs fIcherOs')  MIS FICHEROS  /  LOWER('MIs fIcherOs')  mis ficheros

SUBSTR('Cadena',n[,m])   Subcadena que empieza en n y tiene m caracteres:
     SUBSTR('Cadena',2,3)     ade

RTRIM(cadena)            Elimina los espacios en blanco a la derecha:
     RTRIM(' 12  45 ')        ' 12  45'

LTRIM(cadena)            Elimina los espacios en blanco a la izquierda:
     LTRIM(' 12  45 ')        '12  45 '

TRIM(cadena)             Elimina los espacios en blanco a ambos lados:
     TRIM(' 12  45 ')         '12  45'

LENGTH(cadena)           Longitud de la cadena:
     LENGTH(' 12  45 ')       8

LPAD(cad1,n,cad2)        cad1 con lg. n, ajustada a la dcha, rellenando a la izda con cad2
RPAD(cad1,n,cad2)        cad1 con lg. n, ajustada a la izda, rellenando a la dcha con cad2
     LPAD('123',5,'*')   **123            /    RPAD('123',5,'*')   123**
```

Oracle – Elementos

Funciones de interés (cont.)

```
POWER(n,m)                Eleva n a la m-ésima potencia:
     POWER(3,2)                 9

SQRT(n)                   Raíz cuadrada de n:
     SQRT(25)                   5

FLOOR(n)                  Mayor entero menor o igual a n
     FLOOR(11.2)                11

NVL(expresion,valor)      Sustitución de valor nulo:
     NVL(COMM,0)                Si COMM es no nulo = COMM. Si COMM es nulo = 0.

NVL2(expresion,v1,v2)     Sustitución condicional de valor nulo:
     NVL2(COMM,5,8)             Si COMM es no nulo = 5.    Si COMM es nulo = 8.

SYSDATE                   Fecha y hora actual:
     SYSDATE                    14/12/06  (depende del formato defectivo de la instalación)

USER                      Nombre del usuario.

De agrupamiento:

MIN(expresion)        MAX(expresion)        AVG(expresion)        STDDEV(expresion)
Mínimo                Máximo                Media                 Desviación estándar

VARIANCE(expresion)   COUNT(expresion)
Varianza              Número de elementos
```

Oracle – Elementos

Variables de sustitución

Las variables de sustitución permiten parametrizar una sentencia, pudiendo ejecutarla varias veces e introducir, en tiempo de ejecución, los datos que la hacen diferente.

Ejemplo.- Sentencia que obtiene las filas que cumplen una condición:

```
SQL> SELECT empno, ename, sal, deptno
     FROM   emp
     WHERE  sal > 1000
     AND    deptno = 10

EMPNO ENAME          SAL DEPTNO
----- ---------- ------- ------
 7782 CLARK        2,450     10
 7839 KING         5,000     10
 7934 MILLER       1,300     10
```

Objetivo: parametrizar la sentencia.

Método: usar variables para introducir los valores de los datos.

Oracle – Elementos

Variables de sustitución (cont.)

Parametrizar los valores de los datos

```
SQL> SELECT empno, ename, sal, deptno
     FROM   emp
     WHERE  sal > 1000
     AND    deptno = &dept

Introduzca un valor para dept: 20
antiguo   4: AND    deptno = &dept
nuevo   4: AND    deptno = 20

EMPNO ENAME          SAL DEPTNO
----- ---------- ------- ------
 7566 JONES        2,975     20
 7788 SCOTT        3,000     20
 7876 ADAMS        1,100     20
 7902 FORD         3,000     20
```

La aparición de los valores antiguo y nuevo se debe a la activación del `VERIFY`:

```
SQL> SET VERIFY OFF
```

Oracle – Elementos

Variables de sustitución (cont.)

Varias variables

```
SQL> SELECT empno, ename, sal, deptno
     FROM   emp
     WHERE  ename LIKE '&nome'                    -> Variable CHAR
     AND    deptno = &dept

Introduzca un valor para nome: S%                 -> Variable CHAR
Introduzca un valor para dept: 20

EMPNO ENAME          SAL DEPTNO
----- ---------- ------- ------
 7369 SMITH          800     20
 7788 SCOTT        3,000     20
```

Las variables de sustitución pueden ponerse en el lugar de:

- Un valor.
- Una columna.
- Una tabla.
- Expresiones en: ORDER BY y SELECT.
- Predicados en: WHERE y HAVING.

Oracle – Elementos

Variables de sustitución (cont.)

Sustitución de columnas y expresiones

```
SQL> SELECT empno, ename, sal, &cln1              -> Columna
     FROM   emp
     WHERE  &con                                  -> Predicado en WHERE

Introduzca un valor para cln1: job
Introduzca un valor para con: SAL > 3000

EMPNO ENAME          SAL JOB
----- ---------- ------- ---------
 7839 KING         5,000 PRESIDENT
```

Ha sustituido el nombre de una columna de un `SELECT` y un predicado en un `WHERE`.

```
SQL> INSERT INTO dept (deptno, dname, loc)
     VALUES (&dp,'&dn', &lc)                  -- Atención al tipo de dato

Introduzca un valor para dp: 50
Introduzca un valor para dn: VENTAS
Introduzca un valor para lc: 'BCN'            -- Dos alternativas para CHAR! (p.e. NULL)
```

Oracle – Elementos

Variables de sustitución (cont.)

Reutilizar el valor de una variable

Utilizando `&` siempre pide el valor para la variable.

Si se utiliza `&&` se puede reutilizar el valor en próximas ejecuciones.

```
SQL> SELECT empno, ename, sal, &&c1          -- &&c1
     FROM   emp
     WHERE  &con                             -- &con

Introduzca un valor para c1: mgr             -- Asignación c1
Introduzca un valor para con: SAL > 2000

EMPNO ENAME          SAL   MGR
----- ---------- ------- -----
 7566 JONES        2,975  7839
 ...

SQL> SELECT empno, ename, sal, &&c1
     FROM   emp
     WHERE  &con

Introduzca un valor para con: SAL > 2500     -- Sólo pide el valor de con

EMPNO ENAME          SAL   MGR
----- ---------- ------- -----
 7566 JONES        2,975  7839
 ...
```

Oracle – Elementos

Variables de sustitución (cont.)

Asignación de variables CHAR

```
     DEFINE   (permanente en la sesión).    Formato:  DEFINE [variable [= valor]]
     &&
SQL> DEFINE c2 = job
SQL> SELECT empno, ename, sal, &c2
  2  FROM   emp
  3  WHERE  sal > 2000;

EMPNO ENAME          SAL JOB                -- No pide valores
----- ---------- ------- ---------
 7566 JONES        2,975 MANAGER
 ...
```

La variable c2 se mantiene:

```
SQL> SELECT empno, ename, sal, &c2
     FROM   emp
     WHERE  sal > &val

Introduzca un valor para val: 2500

EMPNO ENAME          SAL JOB
----- ---------- ------- ---------
 7566 JONES        2,975 MANAGER
 ...
```

Oracle – Elementos

Variables de sustitución (cont.)

Mensajes para las entradas de datos

```
     ACCEPT

SQL> ACCEPT val PROMPT 'Introduce o salario:'
Introduce o salario:1600                    -- Modifica el mensaje de petición
                                            -- y asigna un valor a la variable

SQL> SELECT empno, ename, sal, &c2
     FROM   emp
     WHERE  sal > &val                      -- El valor de val es el introducido
                                            -- en el ACCEPT, hasta una nueva asignación

EMPNO ENAME          SAL JOB
----- ---------- ------- ---------
 7566 JONES        2,975 MANAGER
 7698 BLAKE        2,850 MANAGER
 7782 CLARK        2,450 MANAGER
 7788 SCOTT        3,000 ANALYST
 7839 KING         5,000 PRESIDENT
 7902 FORD         3,000 ANALYST

SQL> DEFINE val
DEFINE VAL             = "1500" (CHAR)      -- El valor de val se mantiene
```

Oracle – Elementos

Sinónimos

El nombre de un objeto en Oracle es `usuario.objeto` con una correspondencia entre `usuario` y `esquema`.

La manera de referenciar un objeto de otro usuario tiene que ser de esa forma.

En muchas situaciones, como en los entornos de producción, se necesita referenciar un objeto, como por ejemplo una tabla, mediante un nombre único que puedan referenciar diversos usuarios.

Esto es lo que permiten los sinónimos: vincular un objeto a un nombre.

```
CREATE [PUBLIC] SYNONYM  [<esquema>.]<nombre_sinónimo>
FOR  [<esquema>.]<objeto>
```

Un sinónimo puede vincularse a una tabla, a una vista o a otro sinónimo.

Pueden crearse sinónimos públicos o privados, dependiendo que puedan utilizarlos todos los usuarios o sólo el creador.

Por ejemplo:

```
CREATE PUBLIC SYNONYM  emppro
FOR  emppro
```

Lo que hay tras un sinónimo se determina accediendo a las vistas del catálogo: `USER_SYNONYMS` y `ALL_SYNONYMS`.

Oracle – Elementos

Sinónimos (cont.)

```
DESC ALL_SYNONYMS
```

```
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 OWNER                                     NOT NULL VARCHAR2(30)
 SYNONYM_NAME                              NOT NULL VARCHAR2(30)
 TABLE_OWNER                                        VARCHAR2(30)
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 DB_LINK                                            VARCHAR2(128)
```

```
SELECT *
FROM   ALL_SYNONYMS
WHERE  SYNONYM_NAME LIKE 'EMPPRO';
```

```
OWNER                          SYNONYM_NAME
------------------------------ ------------------------------
TABLE_OWNER                    TABLE_NAME
------------------------------ ------------------------------
DB_LINK
--------------------------------------------------------------------------------
PUBLIC                         EMPPRO
LGARES                         EMPPRO
```

Oracle – Elementos

Índices

Los índices son estructuras auxiliares que almacenan los valores de una o más columnas o atributos de indexación. Se crean para facilitar las operaciones de consulta de los datos.

Oracle dispone de distintos tipos o variedades de índices, que siguen varias clasificaciones. En la terminología de Oracle, los más frecuentes son:

- De árbol-B (realmente B^+) o normales.

  Se trata de árboles B^+ que almacenan los valores de una o más columnas, cuyos nodos hoja apuntan al identificador interno de la fila de una tabla, o sea, al ROWID.

  Oracle los clasifica como `NORMAL` en la columna `INDEX_TYPE` de la vista `USER_INDEXES`.

  Dentro de ellos, todavía podemos considerar:

  – Índice no único.
  Si la columna o columnas sobre las que se define el índice no tienen la propiedad de unicidad, se dice que el índice es no único.
  Son los que se crean por defecto con la sentencia `CREATE INDEX`.

  ```
  CREATE INDEX in_emp_js ON emp(job,sal)        -- No único sobre dos columnas
  ```

  – Índice único.
  Si la columna o columnas sobre las que se define el índice tienen la propiedad de unicidad, se dice que el índice es único.
  Se crean con la sentencia `CREATE UNIQUE INDEX`. También de forma automática, al declarar una restricción de clave primaria.

  ```
  CREATE UNIQUE INDEX in_dept_dname ON dept(dname)  -- Único sobre una columna
  ```

Oracle – Elementos

Índices (cont.)

- De clave inversa.

  Invierten los bytes de los valores de cada columna incluida en el índice.

  Son útiles si se realizan modificaciones reiteradamente sobre los mismos nodos hoja.

  No permiten búsquedas por rango.

  Oracle los clasifica como `NORMAL/REV`.

```
CREATE INDEX in_ab ON t1(a,b) REVERSE   -- De clave inversa sobre dos columnas
```

Reconstrucción de un índice

Un índice puede reconstruirse con `ALTER INDEX`. La reconstrucción puede suponer un cambio de tipo, haciendo que pase de clave inversa a normal o viceversa.

```
ALTER INDEX in_ab REBUILD               -- Reconstruye el índice de clave inversa

ALTER INDEX in_ab REBUILD NOREVERSE     -- Reconstruye el índice y lo transforma
                                        -- al tipo NORMAL

ALTER INDEX in_ab REBUILD REVERSE       -- Reconstruye el índice y lo transforma
                                        -- al tipo NORMAL/REV
```

Oracle – Elementos

Índices (cont.)

- Basados en funciones.

  En ellos los valores del índice no son valores de columnas, sino el resultado de expresiones sobre los valores de las columnas, pudiendo tratarse de operadores, funciones, etc.

  Oracle los clasifica como `FUNCTION-BASED NORMAL`.

```
CREATE INDEX in_emp_sal_comm  ON emp(sal + comm)

CREATE INDEX in_emp_job       ON emp(job DESC)   -- Por tener orden descendente !
```

- De mapa de bits o bitmap.

  Se trata de arrays de bits que indican si las diferentes filas de la tabla, toman o no cada posible valor de la columna, usando para ello los bit 1 y 0.

  Usan una estructura arbórea, almacenando en los nodos hoja los arrays de bits.

  Las operaciones se reducen así a realizarlas sobre los arrays de bits.

  Oracle los clasifica como `BITMAP`.

```
CREATE BITMAP INDEX inbm_emp_deptno ON emp(deptno)
```

Oracle – Elementos

Índices (cont.)

- De mapa de bits o bitmap (cont).

  Ejemplo:

```
Tablas:                                        Bitmaps:

Nombre   | Sexo | Puesto      | Nivel          Sexo          Nivel
-------------------------------------
Ana      |  M   | Desarrollo  | 14              M  1010       12  0001
Juan     |  H   | Sistemas    | 15              H  0101       13  0000
Eva      |  M   | Preventa    | 14                            14  1010
Carlos   |  H   | Desarrollo  | 12                            15  0100

SELECT ...
WHERE  sexo = 'H'
AND    nivel IN (14,15)


sexo = 'H'      nivel = 14    nivel = 15     sexo = 'H'   niv=14 o 15

  0               /   1           0  \          0            1       0
  1              |    0           1   |         1            1       1
  0       AND    |    1     OR    0   |    =    0     AND    1   =   0
  1               \   0           0  /          1            0       0
```

Oracle – Elementos

Índices (cont.)

Oracle usa el término `cardinalidad` para indicar la proporción entre los distintos valores de las columnas que forman un índice y el número de filas diferentes a nulo en esas columnas.

Se dice que la cardinalidad de las columnas que forman el índice es alta, si tienen muchos valores diferentes, y que es baja, si tienen pocos.

Los índices basados en árboles-B tienen las siguientes características:

- Son adecuados si las columnas tienen alta cardinalidad.
- La modificación de los valores de sus columnas no supone problemas.
- No resultan muy adecuados para consultas con predicados `OR`.
- Son habituales en los entornos OLTP.

Y los índices bitmap:

- Son adecuados si las columnas tienen baja cardinalidad.
- La modificación de los valores de sus columnas suele originar problemas.
- Resultan muy adecuados para consultas con predicados `OR`.
- Son habituales en los entornos OLAP o de DSS.

Oracle – Elementos

Índices (cont.)

Otros tipos de índices:

Oracle dispone de otros tipos de índices, que realmente se corresponden con diferentes organizaciones de almacenamiento de los datos de una tabla, más que con la idea habitual de índices.

Entre ellos citamos:

- Tablas organizadas mediante índices (IOT)
  Los datos de la tabla están almacenados en la estructura de un índice de árbol-B, siguiendo el orden de la clave primaria.

- Índice árbol-B cluster
  Un cluster o agrupamiento (en Oracle) es un método de almacenar tablas que consiste en que varias tablas comparten los mismos bloques físicos, según los valores de alguna columna común, que será la denominada clave de agrupamiento.
  Sobre la clave de agrupamiento se genera un índice de árbol-B, denominado índice de agrupamiento.

- Índice hash cluster
  Si en un agrupamiento sustituimos el índice de árbol-B por una función de hash, se dice que tenemos un índice hash cluster.
  Lo que se obtiene es una forma alternativa de acceder a los datos de las tablas, sin realmente utilizar un índice, lo que implica que no hay que realizar un acceso a disco para obtener la posición del bloque físico de los datos, sino que se calcula mediante el cómputo de la función de hash.

Oracle – Elementos

Índices (cont.)

Vistas del catálogo

Los datos fundamentales de los índices se encuentran en `USER_INDEXES` y en `USER_IND_COLUMNS`.

```
DESCRIBE  USER_INDEXES

Nombre                                    Nulo?    Tipo
----------------------------------------- -------- ----------------------------
INDEX_NAME                                NOT NULL VARCHAR2(30)
INDEX_TYPE                                         VARCHAR2(27)
TABLE_OWNER                               NOT NULL VARCHAR2(30)
TABLE_NAME                                NOT NULL VARCHAR2(30)
. . .
```

```
DESCRIBE  USER_IND_COLUMNS

Nombre                                    Nulo?    Tipo
----------------------------------------- -------- ----------------------------
INDEX_NAME                                         VARCHAR2(30)
TABLE_NAME                                         VARCHAR2(30)
COLUMN_NAME                                        VARCHAR2(4000)
COLUMN_POSITION                                    NUMBER
COLUMN_LENGTH                                      NUMBER
CHAR_LENGTH                                        NUMBER
DESCEND                                            VARCHAR2(4)
```

Oracle – Elementos

Tratamiento de datos temporales

Tipo de datos DATE

Permite almacenar fechas, horas y minutos, o combinaciones de ambos.

Debemos comprobar los parámetros relativos a la configuración nacional o NLS (National Language Support), entre ellos `NLS_DATE_FORMAT`, comprobando:

```
SQL> SELECT *  FROM NLS_SESSION_PARAMETERS
```

El formato defectivo puede ser: `DD/MM/RR`. `RR` es similar a `YY` salvo que facilita el trabajo con fechas de distintos siglos.

Pueden utilizarse las funciones `TO_DATE` y `TO_CHAR` para realizar las conversiones de los datos de tipo DATE cuando no sigan el formato defectivo:

```
INSERT INTO kk1 VALUES (10,'nome',TO_DATE('03:45','HH24:MI'))

SELECT cd, nmb, TO_CHAR(tmp,'HH24:MI')  FROM ...
```

Oracle – Elementos

Tratamiento de valores nulos

Ordenación con nulos

Una ordenación ascendente coloca los nulos al final y una descendente los coloca al principio:

```
SQL> SELECT empno, comm
     FROM   emp
     WHERE  deptno IN (10,30)
     ORDER BY comm
```

```
EMPNO    COMM
----- -------
 7844       0
 7499     300
 7521     500
 7654   1,400
 7698
 7934
 7782
 7900
 7839
```

```
SQL> SELECT empno, comm
     FROM   emp
     WHERE  deptno IN (10,30)
     ORDER BY comm DESC
```

```
EMPNO    COMM
----- -------
 7698
 7782
 7900
 7934
 7839
 7654   1,400
 7521     500
 7499     300
 7844       0
```

De alguna forma considera que los valores nulos son "más altos" que los nulos.

Oracle – Elementos

Tratamiento de valores nulos (cont.)

Ordenación con nulos (cont.)

Aunque con SQL podría obtenerse una modificación en el resultado de la ordenación, haciendo que los nulos apareciesen donde se desease, Oracle permite hacer lo mismo de un modo muy sencillo, añadiendo `NULLS FIRST` o `NULLS LAST` a los atributos a ordenar:

```
SQL> SELECT empno, comm
     FROM   emp
     WHERE  deptno IN (10,30)
     ORDER BY comm NULLS FIRST
```

```
EMPNO    COMM
----- -------
 7698
 7782
 7900
 7934
 7839
 7844       0
 7499     300
 7521     500
 7654   1,400
```

```
SQL> SELECT empno, comm
     FROM   emp
     WHERE  deptno IN (10,30)
     ORDER BY comm DESC NULLS LAST
```

```
EMPNO    COMM
----- -------
 7654   1,400
 7521     500
 7499     300
 7844       0
 7698
 7934
 7782
 7900
 7839
```

Oracle – Elementos

Consulta retrospectiva - Flashback query

No se trata de un elemento, sino de una funcionalidad.

Oracle permite realizar una consulta cualquiera de forma que nos devuelve el resultado, no en el momento actual, sino el que se obtenía en un instante del tiempo anterior al actual.

```
SQL> SELECT * FROM emp;              -- Devuelve el resultado actual

     SELECT * FROM emp
     AS OF TIMESTAMP TO_TIMESTAMP('11-04-07 20:03','DD-MM-YY HH24:MI')
     -- Devuelve el resultado que se obtenía en el instante indicado
```

Elementos que la hacen posible

```
     En el init{SID}.ora:

         ##########################################
         # System Managed Undo and Rollback Segments
         ##########################################
         undo_management=AUTO
         undo_retention=10800
         undo_tablespace=UNDOTBS1

     En la tabla de rendimiento dinámico: V$PARAMETER

         Parámetro: undo_retention
```

Oracle – Elementos

Columnas derivadas. Columnas virtuales.

Un dato derivado es el que se obtiene a partir del valor de algún otro. Por ejemplo, la edad de una persona.

Siempre se han podido definir columnas que tuviesen un dato derivado, usando diversas alternativas para mantener el valor del dato correcto, fundamentalmente los disparadores.

Los SGBD presentan diferentes alternativas, por ejemplo definir una columna derivada mediante una expresión, pudiendo o no permitir después su modificación. Normalmente estas columnas derivadas se almacenan como cualquier otra.

Desde la versión 11, Oracle permite crear columnas derivadas especiales a las que denomina **columnas virtuales**, que:

- Mantienen su valor de forma automática mediante un "cálculo al vuelo".
- Sus valores no se almacenan, por lo que no consumen espacio.
- No se permite la modificación de sus valores.

Formato:

```
<columna> <tipo de dato> [GENERATED ALWAYS] AS ( <expresión> ) [VIRTUAL]
```

Oracle – Elementos

Columnas derivadas. Columnas virtuales. (cont.)

Ejemplo: Creamos una tabla con una columna virtual C3:

```
     CREATE TABLE kv1 (c1 NUMBER(5), c2 NUMBER(5), c3 AS (c1 + c2) )

     INSERT INTO kv1 (c1, c2) VALUES (1, 2)
     INSERT INTO kv1 (c1, c2) VALUES (5, 7)

     SELECT * FROM kv1
        C1         C2         C3
---------- ---------- ----------
         1          2          3
         5          7         12

     INSERT INTO kv1  VALUES (10, 20)

ERROR en linea 1:
ORA-00947: no hay suficientes valores

     INSERT INTO kv1  VALUES (10, 20, 5)

ERROR en linea 1:
ORA-54013: No se permite la operacion INSERT en columnas virtuales
```

Oracle – Elementos

Índices invisibles

La creación y eliminación de índices tiene un efecto importante sobre las acciones que toma el optimizador.

Oracle permite crear índices y que el optimizador los tenga en cuenta -hacerlos visibles- o no -hacerlos invisibles-.

```
CREATE INDEX in_kv1_c2 ON kv1(c2) INVISIBLE

ALTER INDEX in_kv1_c2 VISIBLE

ALTER INDEX in_kv1_c2 INVISIBLE
```

Tablas de solo lectura

Puede hacerse que una tabla sea de solo lectura:

```
ALTER TABLE kv1 READ ONLY
```

Luego puede eliminarse esa cualidad, haciendo que pueda modificarse de nuevo:

```
ALTER TABLE kv1 READ WRITE
```

Oracle – Elementos

Disparadores

Diferencias con el estándar

Ejemplo de disparador estándar ejecutado sobre Oracle:

```
SQL> CREATE TRIGGER sal_pos_ins
       AFTER INSERT ON Emp
       REFERENCING NEW ROW AS nova
       FOR EACH ROW
       WHEN (nova.sal <=0)
       DELETE FROM Emp
         WHERE empno = nova.empno

  REFERENCING NEW ROW AS nova
                 *
ERROR en línea 3:
ORA-04074: nombre de REFERENCING no válido
```

- El error está en el término `ROW` (no aceptado en Oracle).
- El ejemplo es de SQL estándar, no de Oracle.

Oracle – Elementos

Disparadores (cont.)

Comprobación de la existencia de los disparadores

Explorar la vista USER_TRIGGERS.

```
SQL> desc user_triggers
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 TRIGGER_NAME                                       VARCHAR2(30)
 TRIGGER_TYPE                                       VARCHAR2(16)
 TRIGGERING_EVENT                                   VARCHAR2(227)
 TABLE_OWNER                                        VARCHAR2(30)
 BASE_OBJECT_TYPE                                   VARCHAR2(16)
 TABLE_NAME                                         VARCHAR2(30)
 COLUMN_NAME                                        VARCHAR2(4000)
 REFERENCING_NAMES                                  VARCHAR2(128)
 WHEN_CLAUSE                                        VARCHAR2(4000)
 STATUS                                             VARCHAR2(8)
 DESCRIPTION                                        VARCHAR2(4000)
 ACTION_TYPE                                        VARCHAR2(11)
 TRIGGER_BODY                                       LONG
```

Oracle – Elementos

Disparadores (cont.)

Ejemplo de creación en Oracle

```
SQL> CREATE TABLE wemp AS SELECT * FROM emp

SQL> SELECT *
     FROM wemp

EMPNO ENAME      JOB          MGR HIREDATE     SAL    COMM DEPTNO
----- ---------- --------- ----- -------- ------- ------- ------
 7369 SMITH      CLERK      7902 17/12/80     800              20
 ...


SQL> CREATE OR REPLACE TRIGGER sal_pos_ins
       AFTER INSERT ON wemp
       REFERENCING NEW AS nova
       FOR EACH ROW
       WHEN (nova.sal <=0)
       DELETE FROM wemp
         WHERE empno = nova.empno

Advertencia: Disparador creado con errores de compilación.
```

El SGBD indica que el disparador genera errores de compilación!

Oracle – Elementos

Disparadores (cont.)

Ejemplo de creación en Oracle (cont.)

Comprobamos los errores de compilación del disparador (válida cualquier opción):

```
SQL> SHOW ERRORS TRIGGER sal_pos_ins

SQL> SHO ERR
```

Seguimos con el ejemplo:

```
SQL> SHO ERR
Errores para TRIGGER SAL_POS_INS:

LINE/COL ERROR
-------- -----------------------------------------------------------------
1/7      PL/SQL: SQL Statement ignored
2/19     PL/SQL: ORA-00904: "NOVA"."EMPNO": identificador no válido
```

El problema está en `"NOVA"."EMPNO"`.

Oracle – Elementos

Disparadores (cont.)

Ejemplo de creación en Oracle (cont.)

Aunque se hubiera creado el bloque PL/SQL ...

```
SQL> CREATE OR REPLACE TRIGGER sal_pos_ins
       AFTER INSERT ON wemp
       REFERENCING NEW AS nova
       FOR EACH ROW
       WHEN (nova.sal <=0)
       BEGIN
         DELETE FROM wemp
           WHERE empno = nova.empno
       END

Advertencia: Disparador creado con errores de compilación.
```

vemos que el error se mantiene.

Origen del problema:

Los valores de las columnas en la acción deben precederse con los dos puntos!.

Oracle – Elementos

Disparadores (cont.)

Ejemplo de creación en Oracle (cont.)

Creamos el disparador:

```
SQL> CREATE OR REPLACE TRIGGER sal_pos_ins
       AFTER INSERT ON wemp
       REFERENCING NEW AS nova
       FOR EACH ROW
       WHEN (nova.sal <=0)
       DELETE FROM wemp
         WHERE empno = :nova.empno

Disparador creado.
```

También podría crearse con:

```
SQL> CREATE OR REPLACE TRIGGER sal_pos_ins
       AFTER INSERT ON wemp
       REFERENCING NEW AS nova
       FOR EACH ROW
       WHEN (nova.sal <=0)
       BEGIN                                -> Bloque PL/SQL
         DELETE FROM wemp
         WHERE empno = :nova.empno;         -> ;
       END;                                 -> ;

Disparador creado.
```

Oracle – Elementos

Disparadores (cont.)

Ejecución del disparador

Se realiza un `INSERT` en la tabla `wemp` para que el disparador se ejecute:

```
SQL> INSERT INTO wemp (empno, sal) VALUES (9999, -100)
           *
ERROR en línea 1:
ORA-04091: la tabla OPS$LGARES.WEMP está mutando, puede que el disparador/la
función no puedan verla
ORA-06512: en "OPS$LGARES.SAL_POS_INS", línea 2
ORA-04088: error durante la ejecución del disparador 'OPS$LGARES.SAL_POS_INS'
```

En oracle la ejecución de un disparador que en la acción o cuerpo del disparador intente modificar la misma tabla que se modifica en el evento del disparador, da un error.

Aquí intentamos modificar `wemp` en la acción, siendo esa misma la tabla cuya modificación supone el evento.

Oracle – Elementos

Disparadores (cont.)

Ejecución del disparador (cont.)

Creamos otra tabla y otro disparador, para que sea otra la tabla que se modifica en la acción del disparador:

```
SQL>  CREATE TABLE xemp AS SELECT empno, sal FROM emp WHERE 1=2

SQL>  CREATE OR REPLACE TRIGGER sal_pos_ins1
        AFTER INSERT ON wemp
        REFERENCING NEW AS nova
        FOR EACH ROW
        WHEN (nova.sal <=0)
        BEGIN
          INSERT INTO xemp VALUES (:nova.empno, :nova.sal);
        END;

Disparador creado.
```

Oracle – Elementos

Disparadores (cont.)

Ejecución del disparador (cont.)

Introducimos datos en `wemp` para que se ejecute el disparador:

```
SQL> INSERT INTO wemp (empno, sal) VALUES (9999, -100);
insert into wemp (empno, sal) values (9999, -100)
            *
ERROR en línea 1:
ORA-04091: la tabla OPS$LGARES.WEMP está mutando, puede que el disparador/la
función no puedan verla
ORA-06512: en "OPS$LGARES.SAL_POS_INS", línea 2
ORA-04088: error durante la ejecución del disparador 'OPS$LGARES.SAL_POS_INS'
```

El error está en el primer disparador, no en el segundo!.

```
SQL> DROP TRIGGER SAL_POS_INS;

SQL> INSERT INTO wemp (empno, sal) VALUES (9999, -100);

1 fila creada.

SQL> select * from xemp;

EMPNO     SAL
----- -------
 9999    -100
```