FICHEROS y BB.DD.

Práctica 2

Dinámica Relacional

uc3m | Universidad Carlos III de Madrid

Prácticas de la asignatura: hoja de ruta

sesión 1

- Modelado Relacional (*esquema relacional*)
- Implementación: entorno SQL+ (consola interacción)
- Estática Relacional: creación de tablas en SQL (LDD)

sesión 2

- Dinámica Relacional
  - consultas básicas en SQL y gestión transaccional
  - carga de datos (ejecución de scripts + volcado)
  - del álgebra relacional al SQL

sesión 3

- Mecanismos de SQL avanzados
  - vistas y disparadores

examen de prácticas

sesión 4

- Diseño Físico
  - Parametrización de la base
  - Organizaciones base y auxiliares
  - Hints

práctica 1

práctica 2

práctica 3

Pasos a seguir

1. Carga de Datos
    1. Identificación de Proveniencia de los datos de cada tabla
    2. Diseño de la sub-consulta
    3. Ejecución de la carga
    4. [análisis y resolución de problemas encontrados]
2. Resolución de Consultas
    1. Diseño de la consulta en Álgebra Relacional
    2. Traducción de cada consulta a SQL
    3. Pruebas
3. Documentación

INSERCIÓN en el Lenguaje SQL

Inserción directa de valores:

```
INSERT INTO <nombre_de_la_tabla>
      [<nombre_columna1>, ...., <nombre_columnaN>]
      VALUES (<valor1>, ....,<valorN>) [,(row2…) […]];
```

Tipos:

- Inserción total (todas las columnas):
  - los valores deben ser compatibles con la definición de las columnas en el esquema de inserción.
  - si algún valor es nulo se debe especificar con la constante NULL.

- Inserción parcial (algunas columnas):
  - se deben especificar las columnas (redefiniendo el esquema de inserción).

Ejercicio: Inserción de Tuplas

- Insertad tuplas en las tablas del ej. de clase (BD prácticas).

1. Si no lo tenéis creado ya, ejecutad el script para crear las tablas.
2. Insertad vuestra titulación y esta asignatura.
3. Insertad vuestros datos (2 alumnos), vuestra matrícula, vuestro grupo, y haceos miembros de él. Para especificar un valor de fecha, usad to_date('01-01-2000', 'DD-MM-YYYY')
4. Insertad los datos sobre las prácticas 1 y 2, y vuestra entrega de cada una.

Sintaxis de la QUERY

```
[WITH
      <símbolo> AS <subquery>
      [, <símbolo> AS <subquery> … ] ]

SELECT [ALL|DISTINCT] <lista de selección>

      FROM <cláusula de origen>

      [WHERE <condición> ]

      [GROUP BY <expresión> [HAVING <condcn>]]

      [{UNION|UNION ALL|MINUS|INTERSECT} <query>]

      [ORDER BY <expresión> [ASC|DESC]] ;
```

La proyección en la query

➢ la proyección se refleja en la <lista de selección>

Lista (no vacía) de datos a recuperar separados por comas.

- Ha de ser posible obtenerlas del área de trabajo (from).
- Puede ser todo el área de trabajo (*) o bien incluir:
  - **atributos** del esquema de relación del área de trabajo
  - **pseudo-columnas**, como ROWNUM y table.ROWID,...
  - **constantes** (como por ejemplo 'X'), **variables** ligadas (como :NEW)
  - **operaciones:** aritméticas (+, -, …), alfanuméricas (||, substr, …), conversión (TO_CHAR, …), codificación (NVL, CASE, …), funciones del sistema (USER, SYSDATE, …), etc,. aplicadas sobre lo anterior.
  - funciones de **agregación** (si área de trabajo agrupada)

Operaciones Matemáticas

- **aritmética básica** : sumas, restas, productos, divisiones, mod, …
- se pueden **operar fechas** (en días):
  - sysdate-1 es la fecha de ayer
  - sysdate-fecha_ini es el número de días transcurridos
- existen otras funciones como
  - redondeos: round, floor, ceil, trunc
  - signo: sign, abs
  - sqrt, log, ln, power
  - greatest, least
  - funciones trigonométricas: sin, cos, tan, asin, sinh, …

Operaciones con cadenas de caracteres

- Concat, **||**
- **lenght** (longitud)
- lower, upper, initcap (minúsculas, mayúsculas, iniciales)
- **substr** (substring)
- **replace** (reemplaza las ocurrencias de un substring por otro)
- **instr** (posición de la ocurrencia iésima de un substring)
- lpad, ltrim, rpad, rtrim,trim (poner o quitar espacios, o cualquier carácter o subcadena, a la izquierda o derecha o ambas)
- convert (cambia el juego de caracteres)
- otras como ascii, chr, …

Ejercicio: Consultas

- Consultad las tablas del ej. de clase (BD prácticas).

  ¿Sabríais decirme…?

  1. Cuántos días de vida tiene cada alumno
  2. Si un crédito son 25 horas, asignaturas con las horas que implican.
  3. Las iniciales de cada alumno

- La validez sintáctica no es suficiente. Debe comprobarse la validez del resultado.
- ¿Cómo comprobar si el resultado obtenido es correcto? Diseño de ***pruebas***
  1. Obtén un muestreo de los datos, y somételo a la consulta (restringe la consulta al muestreo)
  2. Comprueba que el resultado es correcto

Funciones de conversión (de tipo de datos)

- to_char(number[, template])
- to_number(CHAR [, template])
- to_date(CHAR, template)
- to_char(DATE, template)

Template: cadena (entre 'comillas') que especifica el formato literal del valor proporcionado/devuelto (según proceda).

Ejemplos de máscaras numéricas para el número 1119.95:

- '9999.99' → "1119.95"
- '9,999.99' → "1,119.95"
- '€9,999.99' → "€1,119.95"

Referencia: http://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements004.htm#i345102

Funciones de conversión (II)

Las máscaras pueden contener caracteres (constantes) y ***abreviaturas***:

- Y, YY, YYY, YYYY: últimos 1, 2, 3 ó 4 dígitos del año (respectivamente)
- D, DD, DDD, DAY: día en la semana, mes, año, y literal
- MM, MON, MONTH: mes en número, abreviatura, y literal
- HH, MI, SS: horas, minutos, segundos
- … y algunas más (como 'J', día juliano desde 1/1/-4712)

Ejemplos de máscaras de fecha:

- 'DD-MON-YY' → "19-jun-13"
- 'DAY, DD de MONTH de YYYY' → "miércoles, 19 de junio de 2013"
- 'DD/MM/YYYY HH:MI:SS' → "19/06/2013 11:49:45"

Referencia: http://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements004.htm#i345102

Funciones de Codificación/Decodificación

- CASE: elige valor a devolver según casos
- DECODE: sustituye valores por otros valores
- NVL: sustituye null por otro valor
- COALESCE: de una serie de atributos, devuelve el primer no nulo

Ejemplos:

```
SELECT nombre,  CASE WHEN edad<18 THEN 'menor'
                     WHEN edad>65 THEN 'jubilado'
                     ELSE 'joven' END AS categoria
       FROM clientes;

SELECT decode(turno, 'M','mañana', 'T','tarde', 'No matriculado')
       from clientes;

SELECT fullname, nvl(apellido2, ''),
       nvl2(email, 'tiene email', 'no tiene email')
       from clientes;

SELECT fullname, coalesce(email, movil, tlf) from clientes;
```

El área de trabajo (workspace)

➢ **workspace**: es una tabla temporal (vinculada a las tablas de origen)

- Se define por la cláusula FROM, indicando una tabla o varias combinadas
- Las tablas pueden ser relaciones base, vistas, consultas (*subqueries*) o *tablas constantes*: … `VALUES((row1…),(row2…),…) AS alias(col1,col2,…)`
- Existen diversas combinaciones en Oracle:
  - Producto Cartesiano (`CROSS JOIN` alias `,`)
  - Combinación Natural (`NATURAL JOIN`): para Oracle, la keyword NATURAL solo implica eliminar las claúsulas USING y ON
  - Combinación Genérica (`[INNER] JOIN`): será *equijoin* si se aplica USING o bien ON (…=…), y *nonequijoin* en otro caso.
  - Combinación Externa (`left/right/full OUTER JOIN`)
  - Combinación por Unión (`UNION JOIN`)

La selección de tuplas

- la **expresión condicional** (**WHERE**) puede ser…
  - una comparación `(=, !=, <, >, <=, >=)` de expresiones
    - pueden ser listas de expresiones o una subquery
    - la segunda puede ir cuantificada `(SOME, ANY, ALL)`
  - test de inclusión (en conjunto o subquery):
    `<expr> [NOT] IN {<expr_list>|subquery}`
  - test de inclusión (en rango):
    `<expr> [NOT] BETWEEN <expr> AND <expr>`
  - test de valor nulo:
    `<expr> IS [NOT] NULL`
  - test de semejanza:
    `<expr_caracteres> [NOT] LIKE <patrón>`
  - test de existencia:
    `EXISTS subquery`
  - operación lógica `(NOT, AND, OR)` sobre otras condiciones

Ejercicio: Consultas

- Consultad las tablas del ej. de clase (BD prácticas).

  ¿Sabríais decirme…?

  1. Qué asignaturas tienen más de 3 créditos.
  2. Qué alumnos tienen las prácticas que empiezan por 'P'
  3. Prácticas sin entregar aún
  4. Entregas en el último mes en el 'Grado en Ingeniería Informática'

Nota: es recomendable siempre resolver primero las consultas algebraicamente

- ¿Cómo comprobar si el resultado obtenido es correcto?
  1. Analiza las expresiones condicionales y define casos de prueba
  2. Identifica información que deba estar incluida/excluida en cada caso.
  3. Si alguno de los casos no puede ser probado, inserta/borra información.

Agrupación y Orden

- La cláusula **GROUP BY** define el *criterio de agrupación*
- El área de trabajo agrupada sólo tiene, a priori, las columnas incluidas en el criterio de agrupación.
- Se le pueden añadir columnas aplicando funciones de agregación sobre las columnas excluidas del criterio de agrupación.
- Las funciones de agregación predefinidas son: COUNT, AVG, SUM, MIN, MAX, FIRST, y LAST
- Existen otras: MEDIAN, VARIANCE, STDDEV, CORR, COVAR, etc.
- La clausula WHERE se ejecuta antes de agrupar; si se desea realizar una selección del área de trabajo ya agrupada, se usa la clausula HAVING
- La clausula **ORDER BY** define el criterio de ordenación (asc / desc)

Ejercicio: Consultas

- Consultad las tablas del ej. de clase (BD prácticas).

  ¿Sabríais decirme…?

  1. Las iniciales de cada alumno, por orden alfabético de apellido
  2. Cuántos días de media habéis tardado en entregar las prácticas
  3. Cuál es el alumno más joven
  4. Grupos de varios miembros que no difieran en edad más de 100 días.

- ¿Cómo comprobar si el resultado obtenido es correcto?
  1. Analiza criterio de agrupación y condiciones, y define casos de prueba
  2. Identifica información referente a un grupo (muestreo), y altera su composición (inserta/borra) observando cómo se modifica el resultado.

INSERCIÓN en el Lenguaje SQL (II)

Inserción mediante una subconsulta:

```
INSERT INTO <nombre_de_la_tabla>
      [<nombre_columna1>, ...., <nombre_columnaN>]
      SELECT … ;
```

- El resultado de la consulta será insertado en la tabla.
- El orden de las columnas en la *subquery* debe coincidir con el definido en el esquema de inserción (y en su defecto, con el esquema de la tabla destino).
- Para realizar una carga masiva, lo habitual es la inserción desde otra BD.
- Hay que prestar atención al **orden** en que se realiza la **carga de cada tabla.**

Problemas en la INSERCIÓN MASIVA

➢ Restricción de superclave violada (*primary key* o *unique*)

- Se debe localizar las tuplas en conflicto y compararlas para identificar el problema

- Si la instrucción es INSERT INTO tabla (*subquery*), seguir estos pasos:
  - recuperar los valores en conflicto

```
SELECT <superclave>
   FROM (subquery)
   GROUP BY <superclave> HAVING COUNT('x')>1;
```

  - escoger uno de esos valores (*valor1*) y recuperar las filas con ese valor

```
SELECT * FROM (subquery) WHERE <superclave> = valor1;
```

  - Buscar atributos que toman valores distintos para la misma clave primaria
  - Analizar la causa del problema en base a lo anterior (ver casos)

Problemas en la INSERCIÓN MASIVA

- Restricción de superclave violada (continuación).
  Casos y posibles causas del problema:

  1. **Error de Diseño**: las tuplas difieren en un atributo, y éste debería formar parte de la superclave. Solución: corregir el diseño.

  2. **Error en los Datos**: las tuplas difieren en un atributo que no forma parte de la superclave, y no debería tener dos valores.
     Solución: adoptar una asunción semántica implícita para resolver el problema
     Ejemplo: *"si el mismo producto aparece con dos precios, el correcto es el mayor"*.

  3. **Error en el Diseño de la Subquery**: las tuplas difieren en un atributo que no forma parte de la superclave, y sí tiene sentido que tenga dos valores.
     Solución: reescribir la subquery.
     Ejemplo: el atributo adopta valores nulo y no nulo, y debe ignorarse el nulo.

  4. Otros errores: Ambas tuplas son idénticas (no debería ocurrir).

BORRADO Y MODIFICACIÓN en SQL

```
DELETE [FROM] <tabla> [<alias>]
      [WHERE <condicion>]
```

```
DELETE stocks;
/* ¿peligroso? ¡No tanto como DROP o TRUNCATE! */
DELETE FROM stocks WHERE f_caducidad<SYSDATE;
```

```
UPDATE <tabla> SET
{ <columna> = {<expresion>|<subquery>}
| (<columna> [{,<columna>}])=<subquery>}
      [WHERE <condicion>]
```

```
UPDATE persons SET age=25 WHERE name='Javi';
UPDATE stock a SET price = (SELECT a.price*b.change
                              FROM euro_quote b
                              WHERE b.name='peseta');
```

TAKE IT EASY… Learning is FUN !

- **Apps (en Android / IOS)**
  - Learn SQL (SoloLearn)
  - SQL Practice PRO (exercises)
  - Pocket PLSQL, SQL Tutorial, Learn SQL queries, …
- **En página web**:
  - sqlzoo.net