SQL

BASE DE DATOS II

SQL- Breve Historia

- **SQL (Structured Query Language)** es el lenguaje relacional de Base de Datos utilizado para trabajar con Bases de Datos Relacionales, y su historia está íntimamente ligada al desarrollo de éstas
- La mayoría de las DBMSs relacionales comerciales poseen un lenguaje declarativo de alto nivel, en el cual el usuario especifica qué quiere como resultado, dejando las decisiones de cómo ejecutar la consulta para el Sistema
- Entre esos lenguajes, el más conocido es el SQL, que se volvió el padrón norteamericano (ANSI 1986) e internacional (ISO 1989) para los DBMSs relacionales
- Originalmente, al SQL se lo llamó SEQUEL (Structured English QUEry Language) y fue proyectada e implementada en una DBMS experimental de IBM conocida como System R, que fue el prototipo de los RDBMSs comerciales de IBM: SQL/DS y DB2
- Actualmente, todos los RDBMS proveen de una variación del lenguaje SQL adaptado en mayor o menor grado a los estándares mencionados

Resumen de Componentes SQL

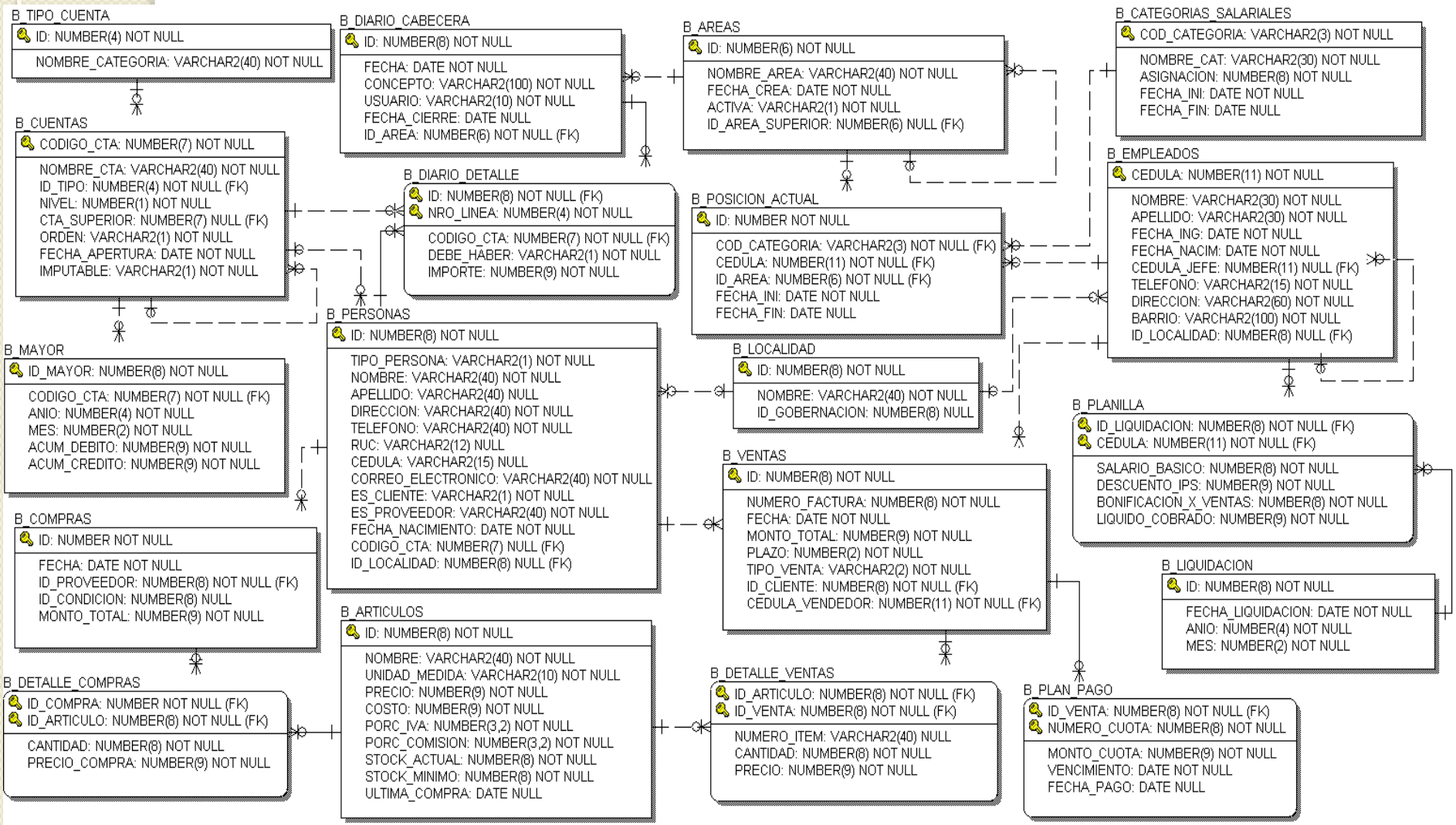
Lenguaje de Definición de datos (Data Definition Language o DDL): proporciona órdenes para la definición de esquemas de la tabla, borrado de tablas, creación de índices y modificación de esquemas de la tabla

Lenguaje interactivo de manipulación de Datos (Data Manipulation Language o DML): Incluye un lenguaje de consultas, basado en el álgebra y el cálculo relacional. Incluye sentencias para insertar, borrar y modificar tuplas o filas de las tablas de la BD

Comandos SQL

- **Manipulación de Datos: Data Manipulation Language (DML)**
  - SELECT, INSERT, UPDATE, DELETE, MERGE
- **Definición de Datos: Data Definition Language (DDL)**
  - CREATE, ALTER, DROP, RENAME, TRUNCATE
- **Control de Transacciones**
  - COMMIT, ROLLBACK, SAVEPOINT
- **Control de Acceso a los Datos: Data control language (DCL)**
  - GRANT, REVOKE

Modelo Utilizado para los ejercicios



Introducción al SQL: La sentencia básica

```
SELECT   * | {[DISTINCT],columna |
expresión [alias],....}
FROM  tabla;
```

Asterisco (*) indica todas las columnas

Comandos SQL

- Los comandos pueden tener varias líneas
- Se puede indentar para facilitar el entendimiento
- Los comandos pueden escribirse en mayúscula o minúscula
- Los comandos se ejecutan una vez ingresados el en buffer

Para seleccionar todas las columnas y filas

```
SQL> SELECT     *
  2  FROM   b_areas;
```

Selección de columnas y filas

```
SQL> SELECT id, nombre_area,
  2 fecha_crea, activa FROM b_areas a;
```

Principales Tipos de Datos Incorporados

VARCHAR2(tamaño)	Cadena de caracteres de longitud variable. El tamaño máximo es de 4.000 bytes y mínimo de 1
NVARCHAR2(tamaño)	Cadena de caracteres UNICODE de longitud variable. El tamaño máximo está determinado por la definición del juego de caracteres nacional, con un límite máximo de 4000 bytesbytes y mínimo de 1
NUMBER(p,s)	Numérico con 'p' posicions y escala 's'. La precisión 'p' varía de 1 a 38. La escala de -84 a 127. Un valor numérico requiere 1 a 22 bytes.
LONG	Datos de caracteres. Longitud de hasta 2 GB. Permanece para compatibilidad con versiones anteriores de Oracle.
DATE	Fecha. Desde el 1 de Enero de 4721 AC al 31 de Diciembre de 9999 DC. l formato por defecto se determina explícitamente por el parámetro NLS_DATE_FORMAT. El tamaño es de 7 bytes. Este tipo de datos contiene los campos de fecha y hora AÑO, MES, día, hora, minuto y segundo.
TIMESTAMP (precisión de fracción de seg.)	Fecha en años, meses, días, minutos, segundos. La precisión es el número de dígitos de la parte fraccional de segundos (0 a 9 y por defecto 6). El formato por defecto se determina explícitamente por el parámetro NLS_TIMESTAMP_FORMAT. El tamaño es de 7 o 11 bytes, dependiendo de la precisión.
CHAR(tamaño)	Representa caracteres de longitud fija. El tamaño máximo es de 2.000 bytes y mínimo 1.
BLOB	Binary Large Object (Objeto grande binario). El tamaño máximo es de (4 gigabytes - 1) * (tamaño del bloque de la base de datos).
CLOB	Objeto grande de caracteres. El tamaño máximo es de (4 gigabytes - 1) * (tamaño del bloque de la base de datos)
BFILE	Contiene la localización de un archivo binario almacenado fuera de la base de datos. Tamaño máximo de 4GB

Otros Tipos de Datos

TIMESTAMP [(precisión de fracción de seg.)] WITH TIME ZONE	Tiene las mismas características que el TIMESTAMP, pero el el tamaño se fija en 13 bytes, ya que este tipo de datos contiene los campos de fecha y hora: AÑO, MES, DIA, HORA, MINUTO, SEGUNDO, TIMEZONE_HOUR y TIMEZONE_MINUTE. Cuenta con las fracciones de segundo y **una zona horaria explícita.**
INTERVAL YEAR [(precisión del año)] TO MONTH	Almacena un período de tiempo en años y meses, donde precisión del año es el número de dígitos en el campo datetime AÑO. Los valores aceptados son del 0 al 9. El valor predeterminado es 2. El tamaño se fija en **5 bytes**. INTERVAL YEAR usa el símbolo '-' (guión), entre el año y el mes.
INTERVAL DAY [(precisión de días)] TO SECOND [(fracción de segundos)]	Almacena un período de tiempo en días, horas, minutos y segundos, donde * day_precision es el número máximo de dígitos en el campo datetime DÍA. Los valores aceptados son del 0 al 9. El valor predeterminado es 2. * fractional_seconds_precision es el número de dígitos en la parte fraccionaria del campo SEGUNDO. Los valores aceptados son del 0 al 9. El valor por defecto es 6. El tamaño se fija en **11 bytes**. Este tipo de intérvalo utiliza espacio entre el número de días y la hora.

Expresiones Aritméticas

- Operadores utilizados
  - Add +
  - Subtract -
  - Multiply *
  - Divide /

Expresiones Aritméticas

Ejemplo: Ver el importe total de cada detalle de ventas

```
SQL> SELECT id_articulo, cantidad,
  2  precio, cantidad*precio
  3  FROM b_detalle_ventas;
```

```
ID_ARTICULO  CANTIDAD     PRECIO CANTIDAD*PRECIO
----------- --------- ---------- ---------------
     646237        10     208182         2081820
     725410        10     499091         4990910
     566179        10       4455           44550
     567052        10       4455           44550
       …
```

Precedencia de Operadores

- Multiplicación y división tienen precedencia por sobre resta y suma
- Los operadores se evalúan de derecha a izquierda
- Los paréntesis ayudan a cambiar el orden de precedencia

Orden de precedencia	Operaciones
1	Operadores aritméticos
2	Operadores de Concatenación
3	Operadores de Condición
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT EQUAL TO
7	NOT <condición lógica>
8	AND <condición lógica>
9	OR <condición lógica>

Precedencia de Operadores

```
SQL> SELECT precio, precio * cantidad + 100
  2  FROM b_detalle_ventas;

    PRECIO PRECIO*CANTIDAD+100
---------- --------------------
    208182              2081920
    499091              4991010
```

```
SQL> SELECT precio, precio * (cantidad + 100)
  2  FROM b_detalle_ventas;

    PRECIO PRECIO*(CANTIDAD+100)
---------- ----------------------
    208182              22900020
    499091              54900010
```

Utilización de un "alias" para las columnas

La operación de "renombramiento" se da a través de la utilización de un alias o sobrenombre, que aparece en la cabecera de las columnas al seleccionarlas

- Es útil cuando la columna es el resultado de un cálculo
- Debe seguir inmediatamente a la columna u operación que representará una columna
- Se puede utilizar la palabra clave "AS", pero no es obligatoria
- Si el alias tiene caracteres especiales o espacios, deberá utilizarse comillas

```
SQL> SELECT precio, precio * cantidad as Importe
  2  FROM b_detalle_ventas;

    PRECIO    IMPORTE
---------- ----------
    208182    2081820
```

Operador de Concatenación

- Se representa por 2 barras verticales (||), y tiene por función enlazar columnas o cadenas de caracteres a otras columna, resultando en una nueva columna de caracteres. Ejemplo:

```
SQL> SELECT nombre || ' ' ||apellido
  2    FROM b_empleados;

NOMBRE||''||APELLIDO
----------------------------------------
Olaf Brandenstein
Jose Caniza Livieres
```

Valores NULOS (NULL)

- NULL es un valor no disponible, no asignado, no conocido o inaplicable; por tanto no tiene igual significado que CERO (0) o espacio
- Si una expresión aritmética contiene valores nulos, toda la expresión se evalúa como NULL

Filas duplicadas

- La sentencia SELECT despliega todas las filas, incluyendo duplicadas (a diferencia de la operación algebraica)

```
SQL> SELECT nombre_area
  2    FROM b_areas;
```

- Para eliminar las filas duplicadas se utiliza la cláusula DISTINCT

```
SQL> SELECT DISTINCT nombre_area
  2    FROM b_areas;
```

DISTINCT para múltiples columnas

- DISTINCT se aplica a todas las columnas en la sentencia SELECT

```
SQL> SELECT DISTINCT cedula, nombre, apellido
  2    FROM b_empleados;
```

- Cuando DISTINCT es aplicado a múltiples columnas, el resultado representa combinaciones de las mismas

Comando WHERE

- Despliega los datos bajo cierta condición

```
SQL> SELECT cedula, nombre, apellido
  2  FROM b_empleados
  3  WHERE cedula_jefe = 952160;

      CEDULA NOMBRE    APELLIDO
------------ --------- -------------
     1098169 Carmen    Ferreira
     1309873 Martin    Moreno
     3008180 Eva       Gonzalez
      800909 Carmelo   Caballero
     1333394 Jose      Balmaceda
```

Comando ORDER BY

- Permite ordenar los datos

```
SQL> SELECT cedula, nombre, apellido
  2  FROM b_empleados
  3  WHERE cedula_jefe = 952160
  4  ORDER by apellido;

     CEDULA NOMBRE    APELLIDO
----------- --------- --------------
    1333394 Jose      Balmaceda
     800909 Carmelo   Caballero
    1098169 Carmen    Ferreira
    3008180 Eva       Gonzalez
    1309873 Martin    Moreno
```

Comando ORDER BY

El orden de aparición de las filas lo da la cláusula ORDER BY

- **ASC** indica orden ascendente, es el valor por defecto
- **DESC** indica orden descendente

```
SQL> SELECT cedula, nombre, apellido
  2  FROM b_empleados
  3  WHERE cedula_jefe = 952160
  4  ORDER by apellido DESC;

      CEDULA NOMBRE   APELLIDO
------------ -------- ------------
     1309873 Martin   Moreno
     3008180 Eva      Gonzalez
     1098169 Carmen   Ferreira
      800909 Carmelo  Caballero
     1333394 Jose     Balmaceda
```

Comando ORDER BY

- La cláusula ORDER BY es la última que aparece en la sentencia SELECT
- No solamente puede ordenarse por campos de la tabla, sino también por expresiones o aliases.
- Los valores NULL se despliegan:
  - En primera posición en secuencias descendentes
  - En última posición en secuencias ascendentes

Ordenando Múltiples Columnas

- Se puede indicar la "posición" del campo a ordenar, para no volver a escribir la columna

```
SQL> SELECT id, nombre_area
  2  FROM b_areas
  3  ORDER BY 2;
```

- Se puede ordenar por múltiples columnas, en cuyo caso, el orden colocado en la lista del "Order By" indica el orden de clasificación. Se puede ordenar también utilizando el alias o una expresión

```
SQL> SELECT id, nombre_area
  2  FROM b_areas
  3  ORDER BY id, nombre_area;
```

Operadores Lógicos y de Comparación

- Operadores de comparación lógicos

  = > >= < <=

- Operadores de comparación propios de SQL
  - BETWEEN ... AND...
  - IN(lista de valores)
  - LIKE
  - IS NULL
- Operadores Lógicos
  - AND
  - OR
  - NOT

Expresiones Negativas

- A veces es más fácil limitar las filas resultantes, excluyendo las filas que uno no quiere que aparezcan
- Operadores lógicos

  != <> ^=
- Operadores SQL
  - NOT BETWEEN
  - NOT IN
  - NOT LIKE
  - IS NOT NULL

Operadores IN y BETWEEN

- Use el operador BETWEEN para probar valores que se encuentran "entre" o "fuera de" un rango de valores.

```
SQL> SELECT DISTINCT cedula, nombre, apellido
  2    FROM b_empleados
  3    WHERE apellido BETWEEN 'Aguayo' and 'Florentin';
```


- Use el operador BETWEEN para probar valores que se encuentran "entre" o "fuera de" un rango de valores.

```
SQL> SELECT DISTINCT cedula, nombre, apellido
  2    FROM b_empleados
  3    WHERE apellido BETWEEN 'Aguayo' and 'Florentin';
```

Operador LIKE

- Se puede usar el operador LIKE para realizar búsquedas utilizando cadenas de caracteres
- Las condiciones de la búsqueda pueden contener caracteres literales o números
- "%" denota ninguno o muchos caracteres
- "_" denota un carácter

```
SQL> SELECT nombre ||' '|| apellido as Empleado
  2  FROM b_empleados
  3  WHERE apellido like 'B%';
```

```
SQL> SELECT nombre ||' '|| apellido as Empleado
  2  FROM b_empleados
  3  WHERE nombre like 'Cint_ia';
```

Operador LIKE

- El operador LIKE puede usarse como un opción para algunas comparaciones BETWEEN

```
SQL> SELECT nombre ||' '|| apellido as Empleado
  2  FROM b_empleados
  3  WHERE apellido like 'B%';
```

- Para buscar los propios caracteres especiales, use la cláusula de ESCAPE

```
SQL> SELECT nombre ||' '|| apellido Cliente, correo_electronico
  2  FROM b_personas
  3  WHERE correo_electronico like  '%\_%' ESCAPE '\';
```

Operador SQL IS NULL

- Para comparar con valores nulos, use el operador IS NULL.
- No use el operador "=".

```
SQL> SELECT cedula, nombre ||' '|| apellido Cliente
  2  FROM b_personas
  3  WHERE ruc is null;
```

Condiciones Múltiples

- Se pueden usar criterios complejos, combinando condiciones con operadores AND u OR
- AND exige ambas condiciones para ser VERDADERO

```
SQL> SELECT codigo_cta, nombre_cta
  2  FROM b_cuentas
  3  WHERE id_tipo=1
  4  AND imputable='S';
```

- OR requiere que cualquiera de las condiciones sea Verdadera para ser VERDADERO

```
SQL> SELECT codigo_cta, nombre_cta
  2  FROM b_cuentas
  3  WHERE id_tipo=1
  4  OR imputable='S';
```

Reglas de precedencia

- Se puede cambiar las reglas de precedencia utilizando paréntesis

Orden de Eval.	Operador
1	Todos los operadores de comparación
2	AND
3	OR

Resumen de la sintaxis

```
SELECT    *| { [DISTINCT] columna | expresión [alias], ...}
FROM      tabla
[WHERE    condición(nes)]
[ORDER BY  {columna, expresión, alias, ...} [ASC|DESC]];
```