TALLER 3

SELECT BÁSICO

SELECT Básico

- Para obtener datos de la base utilizamos la sentencia **SELECT**
  - SQL es declarativo, especifica qué devolver sin decir cómo
- Comenzaremos con consultas simples
  - Sin agrupamiento
  - Una sola tabla
  - Filtrado simple

Sintaxis básica

- **SELECT** [**DISTINCT**] [* | lista-expresiones]
  FROM nombre_tabla
  [**WHERE** condición]
  [**ORDER BY** expresiones_orden]
  [**OFFSET** cant_filas_a_descartar **ROWS**]
  [**FETCH FIRST** cant_max_resultados **ROWS ONLY**];

- **SELECT** y **FROM** lo único obligatorio
- Opción **DISTINCT** inmediatamente después de **SELECT** evita devolver filas repetidas

Columnas a devolver

- * : devuelve todas las columnas de la/las tabla/s participante/s
  - No recomendado en programas!
- No está limitado a devolver valores de columna
  - Lista de expresiones, separadas por coma
    - Se puede renombrar expresiones a devolver (expresión AS “alias”)
  - Una expresión puede ser:
    - Un nombre de columna (se devuelve el valor para cada fila)
    - Un valor fijo (se devuelve dicho valor)
    - Operaciones entre expresiones (ej.: sumas, multiplicaciones, concatenaciones)
    - Funciones, (ej: upper(p) pasa a mayúsculas el valor de p)
      - Pueden recibir expresiones como parámetro

Orden

- **ORDER BY** expresion_1 [**DESC**]
  [, expresion2 [**DESC**]]
  (…) [, expresionN [**DESC**]]
- Ordena las filas a devolver según cada expresión
  - Si la expresión es un número X, se toma como ordenar por la columna a devolver X
    - la primer columna es la 1
  - Se puede usar el alias de la columna para ordenar por ella
- Cada tipo de dato tiene un orden
  - Carácter: alfabético (definido por el set de caracteres)
  - Fechas: de anterior a posterior
- Para orden descendente, usar la opción **DESC**

Primer ejemplo sencillo

- Obtener para cada alumno las siguientes dos columnas:
  - El apellido, una coma y el nombre, todo en mayúsculas. Llamar a esta columna “nombre_completo”
  - El padrón
- Ordenar por la primer columna

```
SELECT upper(apellido || ', ' || nombre)
       AS "nombre_completo"
       , padron
FROM  alumnos
ORDER BY "nombre_completo";
```

Paginado

- Permite devolver sólo algunas filas en vez de todas
  - [**OFFSET** cant_filas_a_descartar **ROWS**]
    FETCH FIRST cant_max_resultados **ROWS ONLY**
  - Se devuelven únicamente “cant_max_resultados” filas
  - Si no se quiere comenzar desde la primera fila, indicar la cantidad a saltear en “cant_filas_a_descartar”
- Siempre se debe utilizar junto con **ORDER BY**
  - Por más que no sea obligatorio, para evitar resultados inciertos
  - El orden debe estar totalmente definido (no admitir empates)
- Evitar utilizar para otras cosas que no sean paginado
  - Ejemplo de uso incorrecto: Obtener el padrón del alumno que tiene la mayor nota (no se sabe cuantos son!)

Paginado – valores a usar

Offset	Resultados
0	Fila 1
1	Fila 2
2	Fila 3
3	Fila 4
4	Fila 5
5	Fila 6
6	Fila 7
7	Fila 8
8	Fila 9
9	Fila 10
10	Fila 11
11	Fila 12
12	Fila 13
13	Fila 14
14	Fila 15
15	Fila 16
16	Fila 17
17	Fila 18

FETCH FIRST 5 ROWS ONLY

OFFSET 5
FETCH FIRST 5 ROWS ONLY

OFFSET 10
FETCH FIRST 5 ROWS ONLY

Paginado - Compatibilidad

- Muchos SGBD no siguen el standard:
  - **SELECT TOP** N para MSSQL
  - **LIMIT** cantidad [**OFFSET** inicio] en PostgreSQL
  - **LIMIT** [inicio,] cantidad en MySql y otros
- Revisar documentación
  - Revisar si para no comenzar desde el primer registro utilizan offset o fila de inicio

Segundo ejemplo sencillo

- Listar de modo paginado todas las notas registradas, indicando únicamente padrón, fecha y nota
- Mostrar las consultas para obtener la primer y tercera página, con páginas de 7 filas cada una

```
SELECT padron, fecha, nota FROM  notas
ORDER BY padron, fecha
FETCH FIRST 7 ROWS ONLY;
```

```
SELECT padron, fecha, nota FROM  notas
ORDER BY padron, fecha
OFFSET 14 ROWS
FETCH FIRST 7 ROWS ONLY;
```

Filtrado de resultados

- Se utiliza la cláusula WHERE condición
  - La condición debe evaluarse como verdadero o falso
  - Devuelve las filas para las que la condición evaluada devuelve verdadero
- Condiciones más comunes:
  - Comparaciones: =, <>, >, <, >, =, <=
  - Concatenadores lógicos: AND, OR, NOT
  - Exp BETWEEN X AND Y
    - equivalente a Exp >= X AND Exp <= Y
- Recomendación: Utilizar paréntesis en condiciones complejas

Expresiones de tipo caracter

- Operador LIKE: Compara por patrones
  - Columna LIKE ‘A%’
- Problemas de CASE-SENSITIVE
  - Definir un modo de almacenar los datos (ej: siempre en mayúsculas)
  - Utilizar ILIKE
    - No aprovecha índices
  - Utilizar funciones
    - upper(columna) = ‘VALOR’
    - Pueden aprovechar índices que utilicen dicha función

Tercer ejemplo sencillo

- Listar el número y nombre de todas las materias del departamento de código 75

```
SELECT numero, nombre FROM  materias
WHERE codigo = 75;
```

Ejercicios

- Resolver los Ejercicios del taller