Explicación sql

martes, 10 de octubre de 2023 11:04

DML

SQL- Manipulación de datos

- SELECT - Sentencia utilizada para listar contenido de una o varias tablas
- INSERT - Sentencia utilizada para insertar contenido en una tabla
- UPDATE - Sentencia utilizada para actualizar contenido de una tabla
- DELETE - Sentencia utilizada para eliminar contenido de una tabla

DDL

SQL- Definición de datos

- De base de datos: CREATE|DROP SCHEMA
- De dominios: CREATE|ALTER|DROP DOMAIN
- De tablas: CREATE|ALTER|DROP TABLE
- De vistas: CREATE|DROP VIEW
- De índices (algunos SGBD): CREATE|DROP INDEX

SELECT → si hay repetidos se los muestra tantas veces como aparezcan

Formato básico

distinct p. sacar repetidos

Select campo/s
From tabla/s

Ejemplo:

Alumno=(dni, nombre, apellido)
Select nombre
From alumno

En el select puedo utilizar * que me mostrará todos los campos de las tablas involucradas sin necesidad de escribir uno por uno

Alumno=(dni, nombre, apellido)
Select DISTINCT(nombre)
From alumno

Cosas pal where.

Operador	Significado	Ejemplo
=	es igual a	Select nombre From alumno Where (nombre="Luciana")
>	es mayor a	Select nombre From alumno Where (dni>24364321)
<	es menor a	Select nombre From alumno Where (dni<24364321)
>=	mayor o igual a	Select nombre From alumno Where (dni>=24364321)
<=	menor o igual a	Select nombre From alumno Where (dni<=24364321)
<>	distinto a	Select nombre From alumno Where (dni<>24364321)
BETWEEN	entre (se incluyen extremos)	Select nombre From alumno Where (dni between 24364321 and 27456789)
LIKE	como	ejemplo próxima diapositiva

- **LIKE**: brinda gran potencia para aquellas consultas que requieren manejo de Strings. Se puede combinar con:
  - %: representa cualquier cadena de caracteres, inclusive la cadena vacía.
  - _ **(guión bajo)**: sustituye solo el carácter del lugar donde aparece.

SELECT nombre, apellido
FROM alumno
WHERE (apellido **LIKE** '%ez')

SELECT nombre, apellido
FROM alumno
WHERE (nombre **LIKE** '__ciana')

Producto Cartesiano (,): para realizar un producto cartesiano, basta con poner en la cláusula FROM dos o más tablas separadas por coma.

SELECT *
FROM alumno,materia → todos vs todos

SELECT a.nombre as 'Nombre alumno' ,m.nombre as 'Nombre materia'   AS: Renombre de atributos.
FROM alumno a,materia m   Alias definido para una tabla.

|x| → no es necesario fijar
→ entiende cuando tiene campos con = so

Select *
From persona natural Join Localidad

Natural Join No lleva ON (entiende de una las relaciones)

Inner join si hay + de un campo con = nombre ej le puedo decir por q' campo relacionar. (o si se llaman distinto)

SELECT a.nombre,a.apellido, e.nota
FROM alumno a INNER JOIN examen e ON (a.dni=e.dni)   **¿Otra forma? ¿Cúal es mejor?**

= Select (mismas cols)
From Alumno A, Examen E
Where (A.DNI = E.DNI)

LEFT JOIN: contiene todos los registros de la tabla de la izquierda, aún cuando no exista un registro correspondiente en la tabla de la derecha, para uno de la izquierda. Retorna un valor nulo (NULL) en caso de no correspondencia.

RIGHT JOIN: es la inversa del LEFT JOIN.

SELECT *
FROM alumno a **LEFT JOIN** examen e **ON** (a.dni=e.dni)

SQL- Operadores

- **UNION:** misma interpretación que en AR. No retorna tuplas duplicadas.
- **UNION ALL:** misma interpretación que la UNION pero retorna las tuplas duplicadas.

IMPORTANTE: Las consultas a unir deben tener esquemas compatibles

SELECT nombre
FROM alumno
WHERE (dni > 25000000)
UNION
SELECT nombre
FROM materia

SQL- Operadores

- **ORDER BY**: permite ordenar las tuplas resultantes por el atributo que se le indique. Por defecto ordena de menor a mayor (operador **ASC**). Si se desea ordenar de mayor a menor, se utiliza el operador **DESC**.

```
SELECT nombre, apellido,dni
FROM alumno
WHERE (dni>23000000) and (nombre='Luciana')
ORDER BY apellido, dni DESC
```

Dentro de la cláusula ORDER BY se pueden indicar más de un criterio de ordenación. El segundo criterio se aplica en caso de empate en el primero y así sucesivamente.

OJO con los renombres.
no puedo hacer order by de algo q' renombré pq' order by se ejecuta 1ro.

- **GROUP BY**: agrupa las tuplas de una consulta por algún criterio con el objetivo de aplicar alguna función de agregación.

```
SELECT nombre, apellido, AVG(nota) as promedio
FROM alumno a INNER JOIN examen e ON
(a.dni=e.dni)
GROUP BY e.dni, nombre, apellido
```

1ro clave primaria → Cada columna incluida en el select debe ir en el group by.

Qué información se puede mostrar cuando se realizó un agrupamiento?
Porque es importante agrupar además por PK?