# Diseño de Bases de Datos

# Structured Query Language(SQL)

Lenguaje de consultas de BD, compuesto por dos submódulos:

- Módulo para definición del modelo de datos, denominado **DDL** (Data Definition Language).
- Módulo para la operatoria normal de la BD, denominado **DML** (Data Manipulation Language).

# 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**

# SQL- Manipulación de datos

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

# SQL- Manipulación de datos

## SELECT

### Formato básico

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.

# SQL- Manipulación de datos

## SELECT

**DISTINCT**: se aplica a campos del select y elimina **tuplas** repetidas.

**Ejemplo:**

**Alumno** = (dni, nombre, apellido)

```
SELECT DISTINCT nombre
FROM alumno
```

| Operador | Significado | Ejemplo |
|---|---|---|
| = | es igual a | Select nombre, apellido<br>From alumno<br>Where (nombre=”Luciana”) |
| > | es mayor a | Select nombre<br>From alumno<br>Where (dni > 24564321) |
| < | es menor a | Select nombre<br>From alumno<br>Where (dni < 24564321) |
| >= | mayor o igual a | Select nombre<br>From alumno<br>Where (dni >= 24564321) |
| <= | menor o igual a | Select nombre<br>From alumno<br>Where (dni <= 24564321) |
| <> | distinto a | Select nombre<br>From alumno<br>Where (dni <> 24564321) |
| BETWEEN | entre (se incluyen extremos) | Select nombre<br>From alumno<br>Where (dni between 30000000 and 40000000) |
| LIKE | como | ejemplo próxima diapositiva |
| | | |

# SQL- Operadores

**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 sólo el carácter del lugar donde aparece.

```
SELECT nombre, apellido
FROM alumno
WHERE (apellido LIKE '%or%')
```

```
SELECT nombre, apellido
FROM alumno
WHERE (nombre LIKE '_ _ _')
```

# SQL- Operadores

Los atributos utilizados en el SELECT de una consulta SQL pueden tener asociados operaciones válidas para sus dominios.

**SELECT** apellido, dni + 5000

**FROM** alumno

**IS NULL** (su negación **IS NOT NULL**): verifica si un atributo contiene el valor de NULL, valor que se almacena por defecto si el usuario no define otro.

**SELECT** nombre, apellido

**FROM** alumno

**WHERE** (nombre **IS NOT NULL**)

# SQL- Operadores

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

**SELECT** *

**FROM** alumno, localidad

---

**SELECT** a.nombre as 'Nombre alumno' ,l.nombre as 'Nombre localidad'

**AS**: Renombre de atributos.

**FROM** alumno a,localidad l

Alias definido para una tabla.

# SQL- Producto natural

**INNER JOIN**: producto natural, reúne las tuplas de las relaciones que tienen sentido. El producto natural se realiza en la cláusula FROM indicando las tablas involucradas en dicho producto, y luego de la sentencia **ON** la condición que debe cumplirse.

**SELECT** a.nombre,a.apellido, e.nota
**FROM** alumno a **INNER JOIN** examen e **ON** (a.dni=e.dni)

**NATURAL JOIN:** análogo al producto natural de AR, trabaja por equicombinación.

# SQL- Producto Natural

**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

- **EXCEPT**: cláusula definida para la diferencia de conjuntos.

**IMPORTANTE: ambas consultas deben tener esquemas compatibles**

```
SELECT dni
FROM alumno
WHERE (dni > 25000000 )
EXCEPT
(SELECT  dni
FROM examen
)
```

- **INTERSECT**: cláusula para la operación de intersección

# 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.

# SQL- Funciones de agregación

Se utilizan en el SELECT y operan sobre un conjunto de tuplas de entrada produciendo un único valor de salida.

- **AVG**: promedio del atributo indicado para todas las tuplas del conjunto.
- **COUNT**: cantidad de tuplas involucradas en el conjunto de entrada.
- **MAX**: valor más grande dentro del conjunto de tuplas para el atributo indicado.
- **MIN**: valor más pequeño dentro del conjunto de tuplas para el atributo indicado.
- **SUM**: suma del valor del atributo indicado para todas las tuplas del conjunto.

# SQL- Agrupamiento

**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

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

# SQL- Agrupamiento clausula HAVING

La cláusula **HAVING** se usa con la cláusula GROUP BY para restringir los grupos que aparecen en la tabla de resultados mediante alguna condición que deben cumplir los grupos

```
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
HAVING AVG(nota) >= 6
```

# SQL- Subconsultas

Consiste en ubicar una consulta SQL dentro de otra. SQL define operadores de comparación para subconsultas:

- **= (igualdad)**: cuando una subconsulta retorna un único resultado, es posible compararlo contra un valor simple.
- **IN (pertenencia)**: comprueba si un elemento es parte o no de un conjunto. Negación (**NOT IN**).
- **=SOME**: igual a alguno.
- **>ALL**: mayor que todos.
- **<=SOME**: menor o igual que alguno

# SQL- Subconsultas

**SELECT** nombre, apellido

**FROM** alumno

**WHERE** dni **IN (SELECT** dni **FROM** examen **WHERE** nota < 4**)**

# SQL- Cláusula Exists

Permite comprobar si una subconsulta generó o no alguna tupla como respuesta. El resultado de la cláusula **EXISTS** es verdadero si la subconsulta tiene al menos una tupla, y falso en caso contrario. Negación (**NOT EXISTS**)

```
SELECT dni, nombre, apellido
FROM alumno a
WHERE EXISTS (SELECT * FROM examen e  WHERE
a.dni=e.dni  and  e.nota=10 )
```

Condición de la consulta principal

# SQL- Equivalencia AR

- T1|×|T2 ≡ SELECT * FROM T1 NATURAL JOIN T2 (equicombinación de tuplas de T1 y T2 sobre la foreign key de una que referencie a la primary key de la otra)
- T1|×|T2 ≡ SELECT * FROM T1 INNER JOIN T2 ON (...) (equicombinación de tablas sobre los campos que se especifiquen –cuando se quieren igualar campos que no coincidan con la *primary key* de una y la *foreign key* de la otra)
- T1|×|θT2 ≡ SELECT * FROM T1 NATURAL JOIN T2 WHERE
- T1 ×|T2 ≡ SELECT * FROM T1 LEFT JOIN T2 ON (...)
- T1 – T2 ≡ T1 EXCEPT T2

# SQL- ABM

- **INSERT INTO ... VALUES**: agrega tuplas a una tabla.

**INSERT INTO** alumno (dni, nombre, apellido) **VALUES** (2827893, ‘Raul’,'Perez');

- **DELETE FROM**: borra una tupla o un conjunto de tuplas de una tabla.

**DELETE FROM** alumno **WHERE** dni=2222222**;**

- **UPDATE … SET**: modifica el contenido de uno o varios atributos de una tabla.

**UPDATE** alumno **SET** nombre='Lorenzo' WHERE dni=22232425

# SQL

**¿Consultas?**