LENGUAJE DE CONSULTA ESTRUCTURADO

Prof. De Bases de Datos:
Lcdo. Luis Peña

COMPONENTES DEL LENGUAJE DE CONSULTA ESTRUCTURADO

ESTRUCTURA DE UNA BASE DE DATOS

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Este lenguaje nos permite realizar consultas a nuestras bases de datos para mostrar, insertar, actualizar y borrar datos.

¿Qué es SQL?

- Lenguaje de consulta estructurado (SQL: Structured Query Languague) .
- Es un lenguaje de base de datos normalizado.
- Utilizado para consultar, modificar o eliminar datos en una Base de Datos.

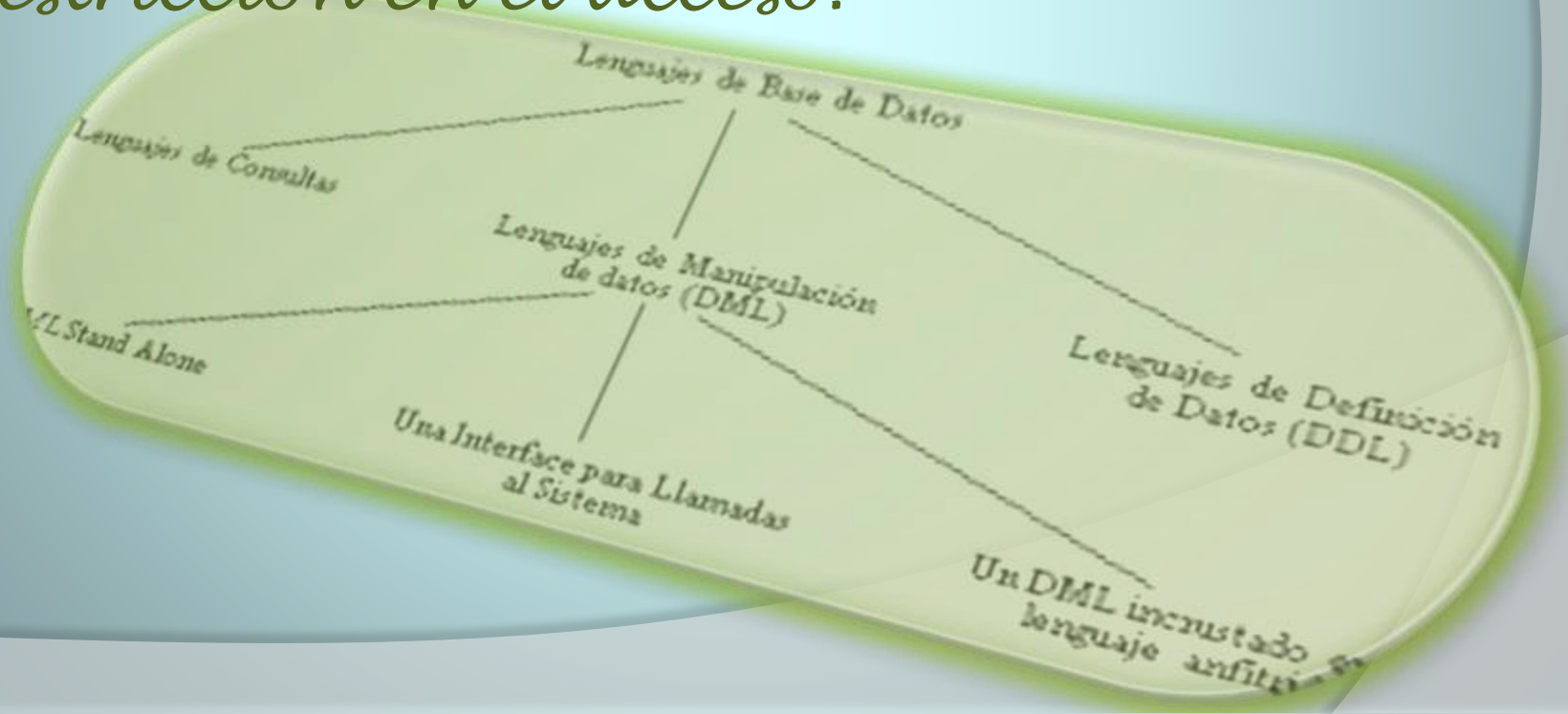
COMPONENTES DEL SQL

Esta compuesto por:

- Los comandos
- Las cláusulas
- Los operadores
- Funciones de agregados

COMANDOS

- DDL: Que permite crear y definir nuevas bases de datos, campos e índices.
- DML: Que permiten generar consultas para ordenar, recuperar, filtrar y extraer datos.
- DCL: Protección de los datos, tablas y restricción en el acceso.



COMANDOS LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

CREATE

- CREAR NUEVAS TABLAS.
- CREAR NUEVAS CAMPOS E ÍNDICES.

DROP

- ELIMINA TABLAS E ÍNDICES.

ALTER

- MODIFICAR LAS TABLAS AGREGANDO CAMPOS O CAMBIANDO LOS CAMPOS.

COMANDOS LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

Select	• Consultas registros de la base de datos que satisfagan un criterio.
Insert	• Cargar lotes de datos en la base de datos.
Update	• Modifica valores de los campos y registros.
Delete	• Elimina registros de una tabla de una base de datos.

COMANDOS LENGUAJE DE CONTROL DE DATOS (DCL)

REVOKE

- ESTE COMANDO CREA UN OBJETO DENTRO DE LA BASE DE DATOS.

GRANT

- ESTE COMANDO PERMITE MODIFICAR LA ESTRUCTURA DE UN OBJETO.

DANY

- ESTE COMANDO ELIMINA UN OBJETO DE LA BASE DE DATOS. SE PUEDE COMBINAR CON LA SENTENCIA ALTER.

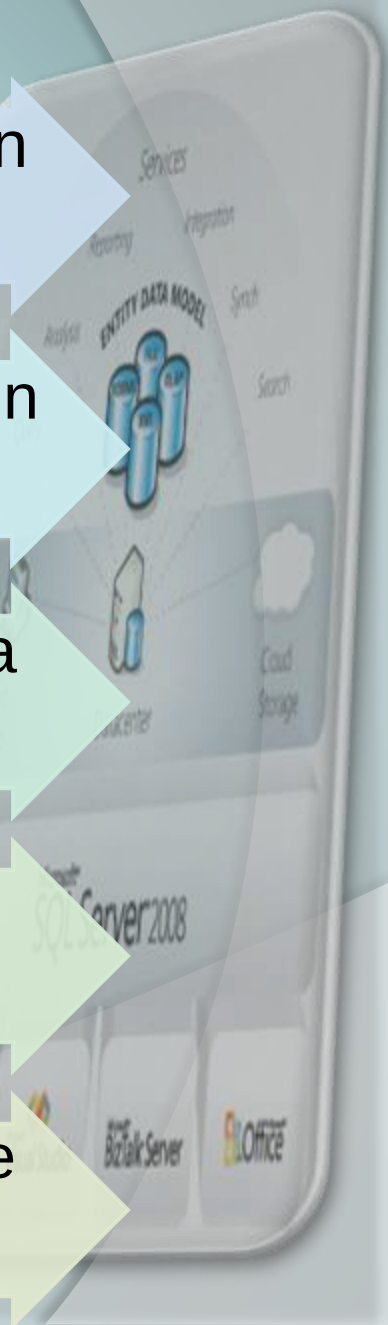
COMANDOS LENGUAJE DE CONTROL DE TRANSACCION (TCL)

SQL proporciona comandos para gestionar cada transacción siguiente:

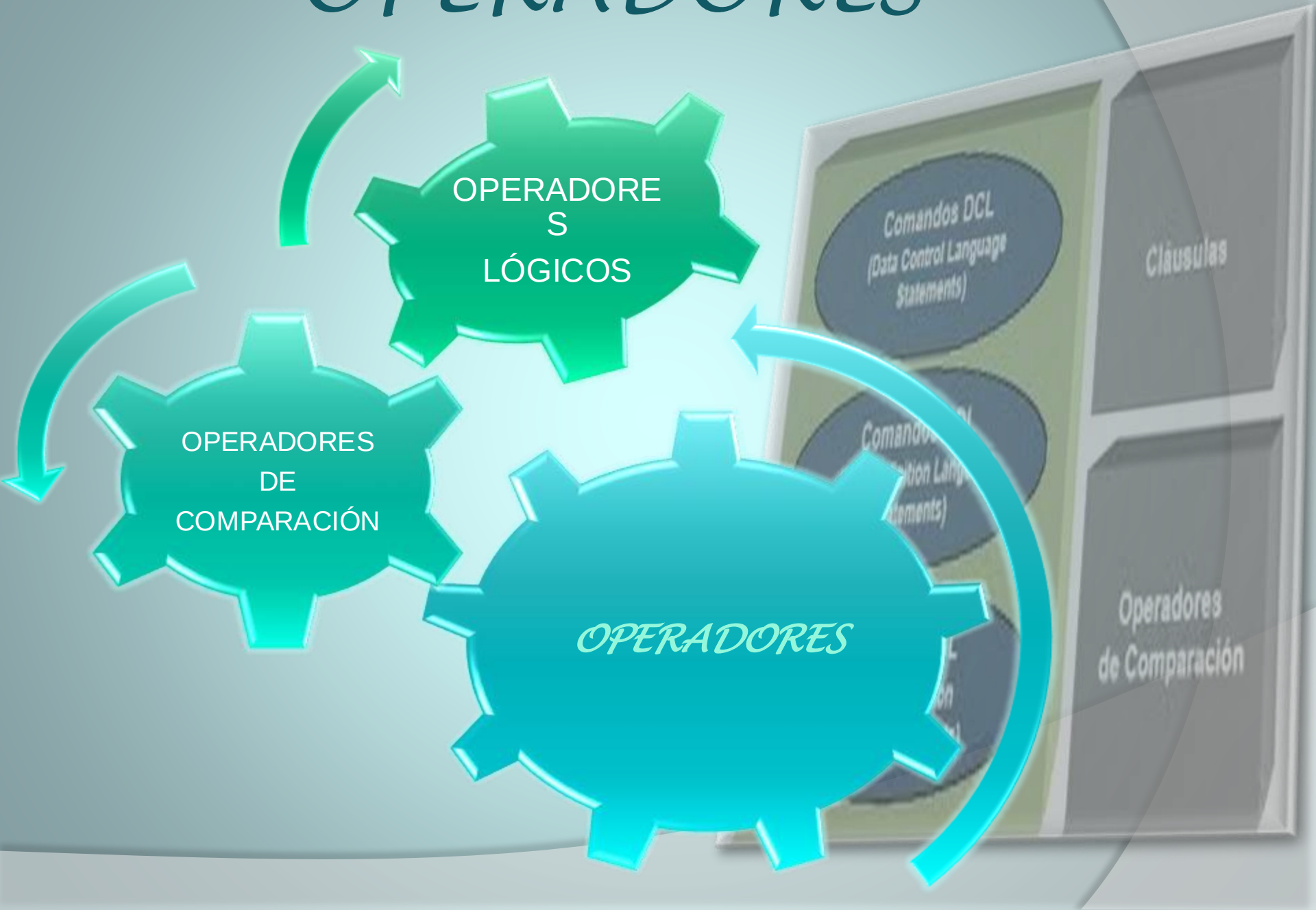
COMMIT	• Para guardar el estado de base de datos después de completar la transacción.
ROLLBACK	• Para restaurar el estado de base de datos, en un estado antes del inicio de la operación.

CLÁUSULA

FROM	• Especifica la tabla de la cual se van a seleccionar los registros.
WHERE	• Especifica las condiciones que deben de reunir los registros.
GROUP BY	• Separa los registros seleccionados a grupos específicos.
HAVING	• Expresa la condición que satisface cada grupo.
ORDER BY	• Ordena los registros seleccionados de acuerdo a un orden especifico.



OPERADORES


OPERADORES
LÓGICOS
OPERADORES
DE
COMPARACIÓN
OPERADORES
Comandos DCL
(Data Control Language
Statements)
Clausulas
Operadores
de Comparación

OPERADORES LÓGICOS

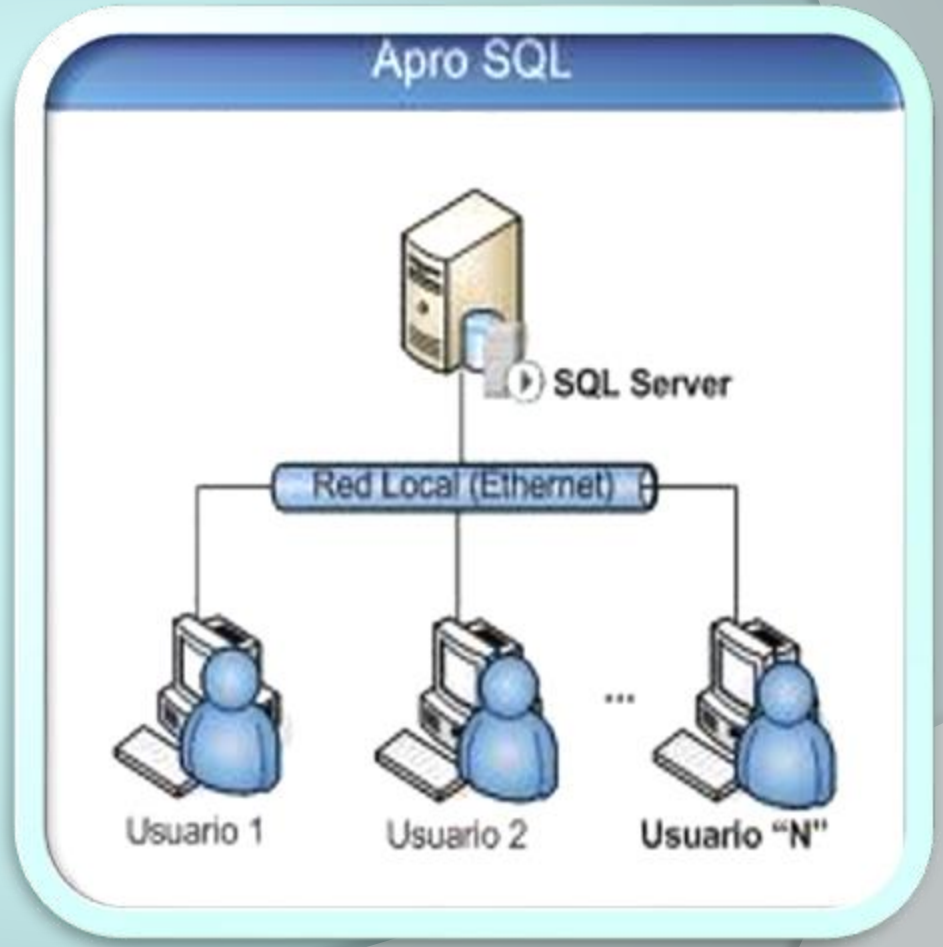
OPERADORES	USO
AND	ES EL "Y" LÓGICO. EVALÚA DOS CONDICONES Y DEVUELVE UN VALOR DE VERDAD SÓLO SI AMBAS SON CIERTAS.
OR	ES EL "O" LÓGICO. EVALÚA DOS CONDICIONES Y DEVUELVE UN VALOR SI ALGUNA DE LAS DOS ES CIERTA.
NOT	NEGACIÓN LÓGICA. DEVUELVE EL VALOR CONTRARIO DE LA EXPRESIÓN.

OPERADORES DE COMPARACIÓN

OPERADOR	USO
<	MENOR QUE
>	MAYOR QUE
<>	DISTINTO QUE
<=	MENOR O IGUAL QUE
>=	MAYOR O IGUAL QUE
BETWEEN	UTILIZADO PARA ESPECIFICAR UN INTERVALO DE VALORES
LIKE	UTILIZADO EN LA COMPARACIÓN DE UN MODELO
IN	UTILIZADO PARA ESPECIFICAR REGISTROS DE UNA BASE DE DATOS.

FUNCIONES DE AGREGADOS

- Se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.



FUNCIONES DE AGREGADOS

COMANDOS	USOS
AVG	UTILIZADO PARA CALCULAR EL PROMEDIO DE LOS VALORES DE UN CAMPO DETERMINADO.
COUNT	UTILIZADO PARA DEVOLVER EL NUMERO DE REGISTROS DE SELECCIÓN.
SUM	UTILIZADO PARA DEVOLVER LA SUMA DE TODOS LOS VALORES DE UN CAMPO.
MAX	UTILIZADO PARA DEVOLVER EL VALOR MÁS ALTO DE UN CAMPO.
MIN	UTILIZADO PARA DEVOLVER EL VALOR MÁS BAJO DE UN CAMPO.

Orden de Ejecución de los Comandos

- Dada una sentencia SQL de selección que incluye todas las posibles clausulas, el orden de ejecución de las mismas es el Siguiente:
  1. Clausula FROM
  2. Clausula WHERE
  3. Cláusula GROUP BY
  4. Cláusula HAVING
  5. Cláusula SELECT
  6. Cláusula ORDEN BY

Consulta básica

SELECT A

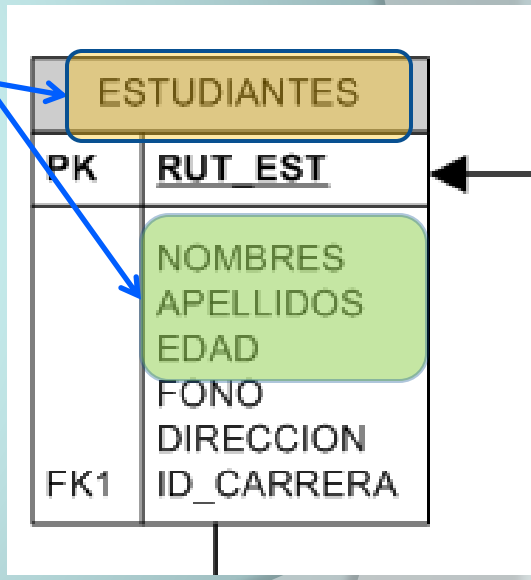
FROM B

Donde A son los datos que requiero (columnas) y B es de donde obtengo esos datos.

Ejemplo

SELECT nombre, apellidos, edad

FROM estudiantes



	NOMBRES	APELLIDOS	EDAD
1	Esteban	Riquelme	22
2	Ernesto	Cardenas	25
3	Jenni	Gonzalez	21
4	Boris	Mella	20

WHERE

- A veces NO se necesitan obtener datos tan generales, y es cuando se aplican filtros, con la clausula WHERE.

```
SELECT   A
FROM   B
WHERE   C
```

Donde C, es una o más condiciones.

Ejemplo

```
SELECT  nombres, apellidos, edad
FROM  estudiantes
WHERE  edad > 21
```

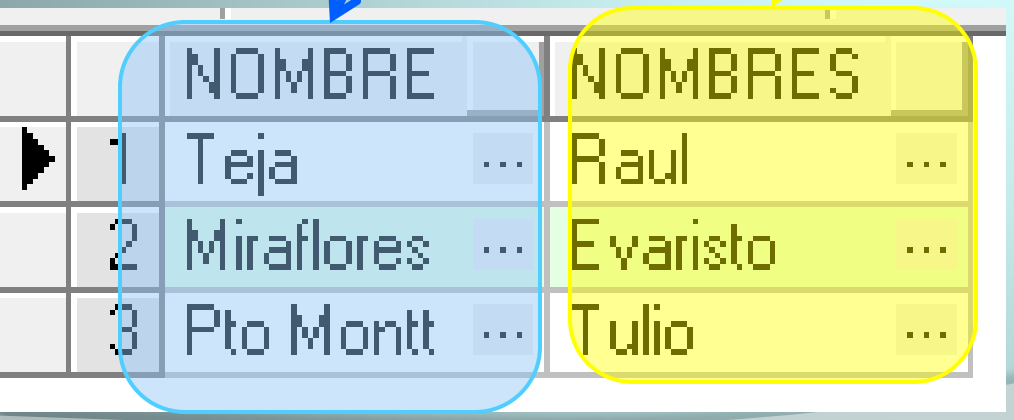
	NOMBRES	APELLIDOS	EDAD
▶ 1	Esteban	Riquelme	22
2	Ernesto	Cardenas	25

Alias

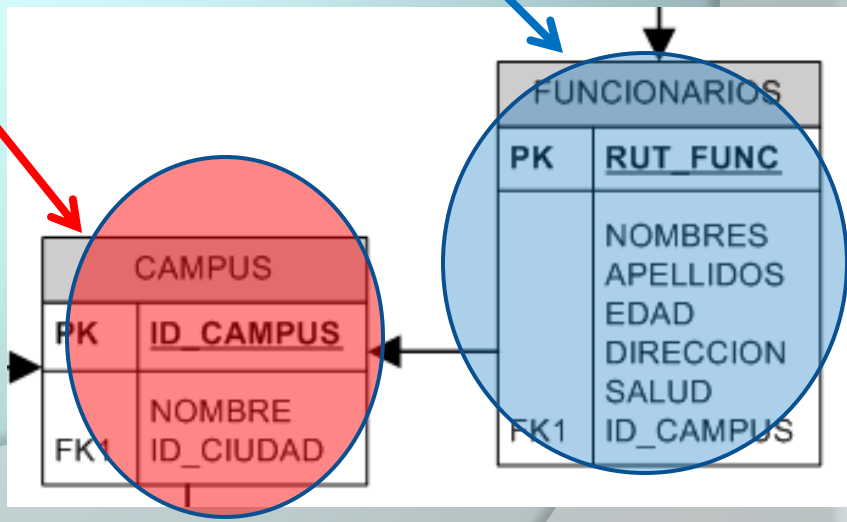
- Los alias son un nombre de asignación que se le dan a los recursos, en este caso las tablas.
- Luego se pueden llamar a sus atributos desde ese alias, continuados con un punto (“.”) .

Ejemplo

```
SELECT  c.nombres, f.nombres
FROM  campus c, funcionarios f
WHERE  c.id_campus =
   f.id_campus
```



	NOMBRE	NOMBRES
1	Teja	Raul
2	Miraflores	Evaristo
3	Pto Montt	Tulio



Alias 2

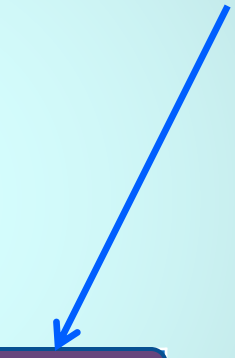
- Es posible dar un alias (nombre) al titulo de las columnas de una tabla, que no es el mismo que posee en la Base de Datos.
- Ejemplo sin Alias:

```
Select  l.titulo, l.agno
From  libros l
```

TITULO	AGNO
Matemáticas	2007
Lenguaje y Comunicaciones	1998
Cs. Biológicas	2003

Ejemplo con Alias

```
Select l.titulo, l.agno As AÑO
From libros l
```



TITULO	AÑO
Matemáticas	2007
Lenguaje y Comunicaciones	1998
Cs. Biológicas	2003

GROUP BY

La cláusula GROUP BY se usa para generar valores de agregado para cada fila del conjunto de resultados. Cuando se usan sin una cláusula GROUP BY, las funciones de agregado sólo devuelven un valor de agregado para una instrucción SELECT.

Ejemplo:

SELECT ***nombre_columna1, nombre_columna2***
FROM ***nombre_tabla***
GROUP BY ***nombre_columna1***

GROUP BY: Ejemplo

tienda_info

nombre_tienda	ventas	fecha
Valdivia	1500	05-jan-2010
Temuco	250	07-jan-2010
Valdivia	300	08-jan-2010
Osorno	700	08-jan-2010

CONSULTA

```
SELECT nombre_tienda, SUM(ventas)
FROM tienda_info
GROUP BY nombre_tienda
```

RESULTADO

Valdivia	1800
Temuco	250
Osorno	700

HAVING

Especifica una condición de búsqueda para un grupo o agregado. HAVING sólo se puede utilizar con la instrucción SELECT. Normalmente, HAVING se utiliza en una cláusula GROUP BY. Cuando no se utiliza GROUP BY, HAVING se comporta como una cláusula WHERE.

Ejemplo:

SELECT *nombre_columna1, SUM(nombre_columna2)*
FROM *nombre_tabla*
[GROUP BY nombre_columna1]
HAVING *(condición de función aritmética)*

HAVING: Ejemplo

tienda_info

nombre_tienda	ventas	fecha
Valdivia	1500	05-jan-2010
Temuco	250	07-jan-2010
Valdivia	300	08-jan-2010
Osorno	700	08-jan-2010

CONSULTA

```
SELECT nombre_tienda, SUM(ventas)
FROM tienda_info
GROUP BY nombre_tienda
HAVING SUM(ventas) > 1500
```

RESULTADO

Valdivia	1800

ORDER BY

Especifica el orden utilizado en las columnas devueltas en una instrucción SELECT. La cláusula ORDER BY no es válida en vistas, funciones insertadas, tablas derivadas ni subconsultas.

Ejemplo:

SELECT *nombre_columna1, nombre_columna2*
FROM *nombre_tabla*
[WHERE *condicion*]
ORDER BY *nombre_columna1* [*ASC, DESC*]

ORDER BY: Ejemplo

tienda_info

nombre_tienda	ventas	fecha
Valdivia	1500	05-jan-2010
Temuco	250	07-jan-2010
Valdivia	300	08-jan-2010
Osorno	700	08-jan-2010

CONSULTA

```
SELECT nombre_tienda, ventas, fecha
FROM tienda_info
ORDER BY ventas DESC
```

RESULTADO

Valdivia	1500	05-jan-2010
Osorno	700	08-jan-2010
Valdivia	300	08-jan-2010
Temuco	250	07-jan-2010

OPERADORES LOGICOS (AND-OR)

C1	C2	C1 AND C2
V	V	V
V	F	F
F	V	F
F	F	F

C1	C2	C1 OR C2
V	V	V
V	F	V
F	V	V
F	F	F

OPERADORES LOGICOS: Ejemplo AND

tienda_info

nombre_tienda	ventas	fecha
Valdivia	1500	05-jan-2010
Temuco	250	07-jan-2010
Valdivia	300	08-jan-2010
Osorno	700	08-jan-2010

CONSULTA

```
SELECT *
FROM tienda_info
WHERE ventas > 500
AND nombre_tienda = 'Valdivia'
```

RESULTADO

Valdivia 1500 05-jan-2010

OPERADORES LOGICOS: Ejemplo OR

tienda_info

nombre_tienda	ventas	fecha
Valdivia	1500	05-jan-2010
Temuco	250	07-jan-2010
Valdivia	300	08-jan-2010
Osorno	700	08-jan-2010

CONSULTA

```
SELECT *
FROM tienda_info
WHERE ventas > 500
OR nombre_tienda = 'Valdivia'
```

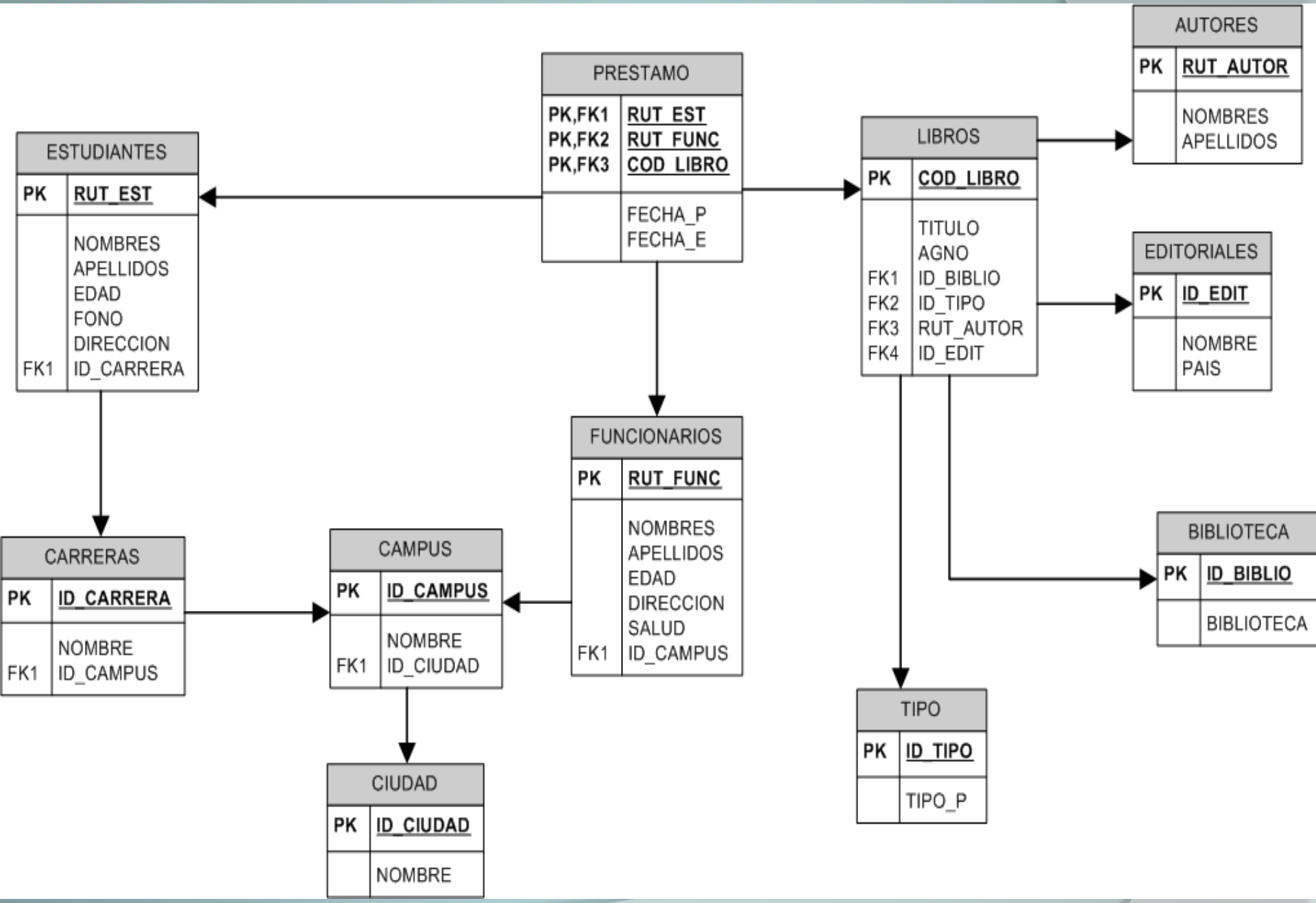
RESULTADO

Valdivia	1500	05-jan-2010
Valdivia	300	08-jan-2010
Osorno	700	08-jan-2010

Modelo de Datos

Para este laboratorio usaremos la Base de Datos Biblioteca.

Para ello use la imagen que se encuentra en siveduc, “Biblioteca.png” y cargue el archivo “Biblioteca.sql” en PLSQL como se enseño en la clase anterior.


AUTORES
PK RUT_AUTOR
NOMBRES
APELLIDOS
PRESTAMO
PK,FK1 RUT_EST
PK,FK2 RUT_FUNC
PK,FK3 COD_LIBRO
FECHA_P
FECHA_E
LIBROS
PK COD_LIBRO
TITULO
AGNO
FK1 ID_BIBLIO
FK2 ID_TIPO
FK3 RUT_AUTOR
FK4 ID_EDIT
ESTUDIANTES
PK RUT_EST
NOMBRES
APELLIDOS
EDAD
FONO
DIRECCION
FK1 ID_CARRERA
EDITORIALES
PK ID_EDIT
NOMBRE
PAIS
FUNCIONARIOS
PK RUT_FUNC
NOMBRES
APELLIDOS
EDAD
DIRECCION
SALUD
FK1 ID_CAMPUS
BIBLIOTECA
PK ID_BIBLIO
BIBLIOTECA
CARRERAS
PK ID_CARRERA
NOMBRE
FK1 ID_CAMPUS
CAMPUS
PK ID_CAMPUS
NOMBRE
FK1 ID_CIUDAD
TIPO
PK ID_TIPO
TIPO_P
CIUDAD
PK ID_CIUDAD
NOMBRE

EJEMPLOS 1

- **Ingrese a la BBDD, y ejecute la siguiente instrucción:**

```
create table CAMIONES
(
  COD_CAMION INTEGER not null,
  MARCA      VARCHAR2(150),
  MODELO     VARCHAR2(300),
  ANNO       NUMBER,
  PESO_CARGA INTEGER,
  PATENTE    VARCHAR2(300)
)
```

- **Ahora ejecute la siguiente instrucción:**

```
alter table CAMIONES
add constraint PK_CAMIONES primary key (COD_CAMION);
```

- **Y ahora ejecute la siguiente instrucción**:

```
drop table CAMIONES
```

EJEMPLOS 2

- **Ejecute las siguientes instrucciones**:

```
create table CAMIONES
   (
     COD_CAMION INTEGER not null,
     MARCA      VARCHAR2(150),
     MODELO     VARCHAR2(300),
     ANNO       NUMBER,
     PESO_CARGA INTEGER,
     PATENTE    VARCHAR2(300)
   )
alter table CAMIONES
     add constraint PK_CAMIONES primary key (COD_CAMION);
```

- **Y ahora ejecute la siguiente instrucción**:

```
insert into CAMIONES (COD_CAMION, MARCA, MODELO, ANNO,
PESO_CARGA, PATENTE)
values (1000, 'susuki', 'baleno', 1995, 45, 'pi-1516');
```

EJEMPLOS 2.1

- **Ejecute:**

```
update camiones c set c.modelo='probando'
where c.cod_camion=1000
```

- **Y luego ejecute:**

```
delete from camiones c
where c.cod_camion=1000
```

RECUERDE QUE PARA QUE LOS CAMBIOS SEAN VISIBLES Y EFECTIVAMENTE SE REALICEN DEBE PRESIONAR:

F10 correspondiente al comando **Commit**