Bases de Datos

Structured Query Language (SQL)

Ciclo 2024

INTRODUCCIÓN

Introducción

- Los artículos de Cood (Modelo Relacional y Lenguajes Relacionales) provocaron un gran impacto en la comunidad científica y en la comercial.

- Lenguajes de Consulta Relacionales:

  - Álgebra Relacional: lenguaje procedural.

  - Cálculo Relacional (orientado a tuplas y orientado a dominios): lenguaje no procedural.

- Ambos lenguajes son equivalentes en poder expresivo.

- Objetivo de la comunidad de BD: desarrollar versiones implementadas de los lenguajes relacionales.

Introducción

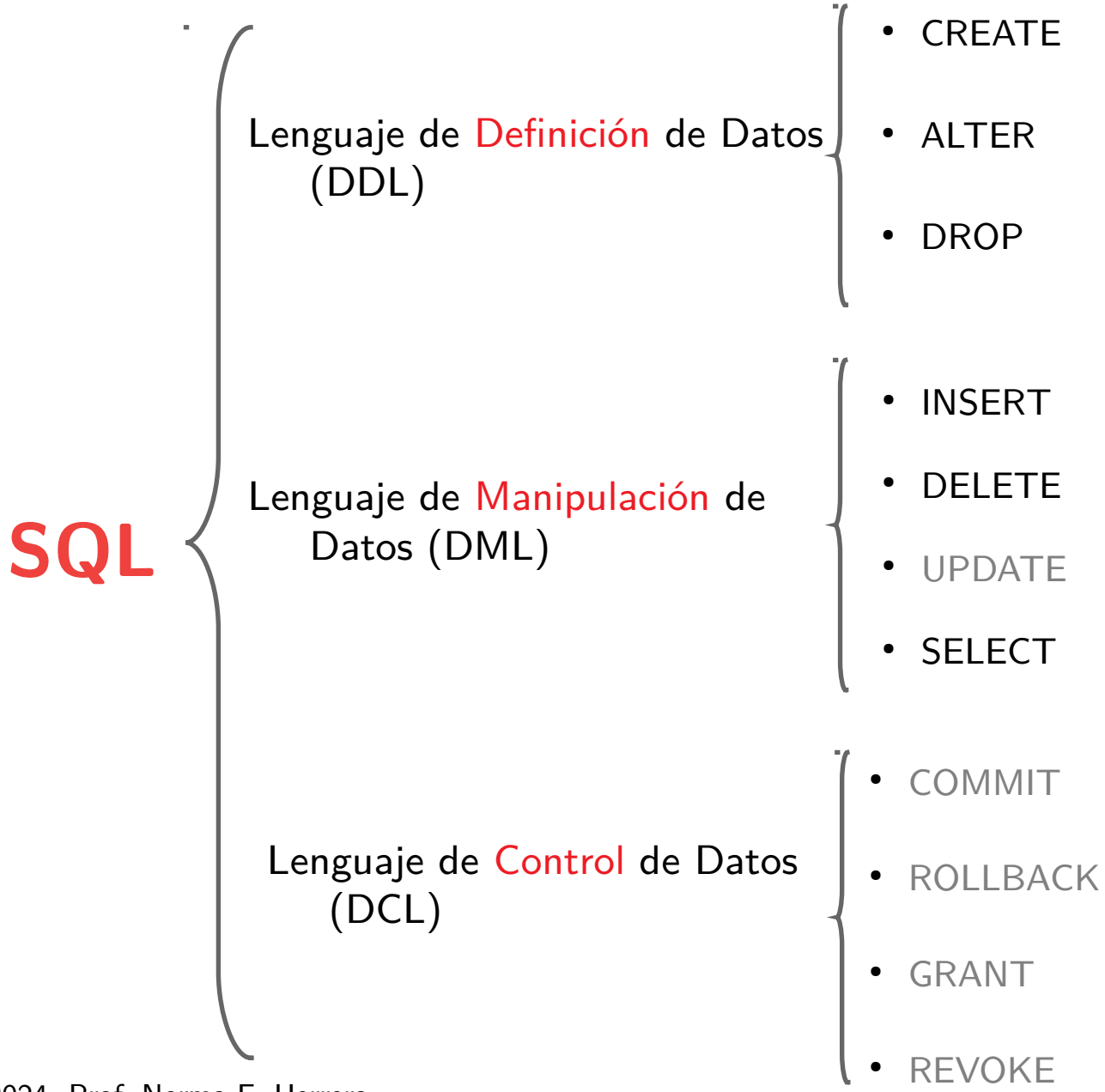
Los tres lenguajes mas importantes que surgieron fueron:

- SQL(Structured Query Language):
  - IBM, década 1970.
  - Lenguaje textual basado en Cálculo Relacional orientado a Dominios.
- QBE (Query By Example):
  - IBM, década 1970.
  - Lenguaje gráfico basado en Álgebra Relacional y construcciones del Cálculo Relacional orientado a dominios.
- QUEL(Query Language):
  - Lenguaje original de INGRES (SGBD desarrollado por la universidad Berkeley) década 1970
  - Basado en Cálculo Relacional orientado a tuplas.

Introducción

- SQL es un lenguaje de consulta estructurado para bases de datos relacionales.
- Originalmente: SEQUEL (Structured English QUEry Language) diseñado e implementado por IBM como interfaz para System R (sistema experimental de base de datos relacionales).
- En 1986, ANSI (American National Standards Institute) e ISO (International Standards Organization), publicaron una norma SQL, denominada SQL-86.
- Desde entonces, el estándar ha sido revisado para incluir más características: SQL-89, SQL-92, SQL-99, etc. La última revisión es de 2016: SQL-2016 (ISO/IEC 9075:2016).
- SQL es hoy en día el único lenguaje relacional de BD que es ANSI estándar.
- A pesar de la existencia de estándares, la mayoría de los códigos SQL no son completamente portables entre SGBD.

Introducción



SQL: DDL
(Data Definition Language)

SQL: DDL

- Sentencias que permiten crear y modificar esquemas:
  - Esquema de BD:

    CREATE DATABASE

  - Esquema de Relaciones:

    CREATE TABLE

    ALTER TABLE

    DROP TABLE

Creación de Esquemas: CREATE

CREATE DATABASE

- Sentencia que permiten crear un nuevo esquema de BD.
- Ejemplo: supongamos que tenemos el siguiente esquema relacional de BD que mantiene información sobre los cursos de formación que toman los docentes.

Formacion={Docentes, Cursos, Tomo}

CREATE DATABASE Formacion
CREATE SCHEMA Formacion

Docentes= {D-Cod, D-DNI, D-Nbre, D-Dir }
dom(D-Cod)= dom (D-DNI)=$\mathbb{N}$
dom(D-Nbre)=Alfa+
dom(D-Dir)=AlfaNco*

Cursos= {C-Cod, C-Nbre, D-Cod-Supervisa }
dom(C-Cod)= dom(D-Cod-Supervisa) = $\mathbb{N}$
dom(C-Nbre)=Alfa+
FK(DCod-Supervisa)$\rightarrow$ Docentes(D-Cod)

Tomo= {D-Cod-Tomo, C-Cod-Tomo , Fecha-Tomo }
dom(D-Cod-Tomo)= dom(C-Cod-Tomo)=dom(Fecha-Tomo)=$\mathbb{N}$
FK(D-Cod-Tomo)$\rightarrow$ Docentes(D-Cod)
FK(C-Cod-Tomo) $\rightarrow$ Cursos (C-Cod)

CREATE TABLE

- Permite crear cada uno de los esquemas de relación que conforman el esquema de BD.

- Por cada relación hay que indicar:
  - Nombre de la relación.
  - Atributos que conforman el esquema (nombre y dominio).
  - Clave primaria.
  - Claves alternativas (si las hubiera).
  - Restricciones de integridad referencial (si las hubiera).

CREATE TABLE

- Modelo Relacional

Docentes= {D-Cod, D-DNI, D-Nbre, D-Dir }

dom(D-Cod)=ℕ

dom (D-DNI)=ℕ

dom(D-Nbre)=Alfa+

dom(D-Dir)=AlfaNcø*

- Implementación

```
CREATE TABLE Docentes (
        D-Cod    INT    NOT NULL,
        D-DNI    INT    NOT NULL,
        D-Nbre    VARCHAR (200) NOT NULL,
        D-Dir     VARCHAR (200),
   PRIMARY KEY (D-Cod )
) ;
```

- Los tipos de atributo y dominios se detectan en el proceso de modelado (MER , MR).
- Sólo puede existir una primary key por relación

CREATE TABLE: Tipos de Datos

- Tipos de Datos (SQL estándar):
  - Numérico enteros: **INT, SMALLINT**
  - Numéricos reales: **FLOAT, REAL, DOUBLE PRECISION**
  - Secuencias de caracteres de largo fijo: **CHAR (n)**
  - Secuencias de caracteres de largo variable: **VARCHAR (n)**
  - Fecha : **DATE**
  - Hora: **TIME**
- Siempre deben verificar en el manual del SGBD los tipos de datos permitidos.
- Intentar usar los tipos estándar (portabilidad).

CREATE TABLE: Clave Principal

- Modelo Relacional

Docentes= {D-Cod, D-DNI, D-Nbre, D-Dir }

dom(D-Cod)=ℕ

dom (D-DNI)=ℕ

dom(D-Nbre)=Alfa+

dom(D-Dir)=AlfaNco*

- Implementación

```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL,
        D-DNI   INT   NOT NULL,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir    VARCHAR (200),
   PRIMARY KEY (D-Cod )
) ;
```

```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL PRIMARY KEY ,
        D-DNI   INT   NOT NULL,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir    VARCHAR (200),
) ;
```

CREATE TABLE: Clave Principal

- Modelo Relacional

Tomo= {D-Cod-Tomo, C-Cod-Tomo , Fecha-Tomo }

dom(D-Cod-Tomo)= dom(C-Cod-Tomo)=dom(Fecha-Tomo)=$\mathbb{N}$

FK(D-Cod-Tomo)$\rightarrow$ Docentes(D-Cod)

FK(C-Cod-Tomo) $\rightarrow$ Cursos (C-Cod)

- Implementación

```
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo  DATE
   PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo )
) ;
```

CREATE TABLE: Clave Principal

- Modelo Relacional

Tomo= {D-Cod-Tomo, C-Cod-Tomo , Fecha-Tomo }

dom(D-Cod-Tomo)= *dom*(C-Cod-Tomo)=dom(Fecha-Tomo)=$\mathbb{N}$

FK(D-Cod-Tomo)→ Docentes(D-Cod)

FK(C-Cod-Tomo) → Cursos (C-Cod)

- Implementación

```
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo  DATE
  PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo)
) ;
```

```
CREATE TABLE Tomo (
    D-Cod-Tomo   INT   NOT NULL PRIMARY KEY
    C-Cod-Tomo   INT   NOT NULL PRIMARY KEY
    Fecha-Tomo  DATE
) ;
```

Es incorrecto:

- No pueden haber dos Primary Key
- Aunque se pudiera, esto estaría indicando dos claves simples y no una clave compuesta

CREATE TABLE: Clave Principal

- Si la clave es simple $\rightarrow$ la puedo indicar al declarar el atributo o al finalizar la declaración de atributos

- Si la clave es compuesta $\rightarrow$ solo la puedo declarar al finalizar la declaración de atributos

CREATE TABLE: Claves Alternativas

- Para las claves alternativas → UNIQUE + NOT NULL
- Ejemplo:
  - Modelo Relacional:

    Docentes= {D-Cod, D-DNI, D-Nbre, D-Dir }

    dom(D-Cod)=ℕ

    dom (D-DNI)=ℕ

    dom(D-Nbre)=Alfa+

    dom(D-Dir)=AlfaNco*

  - Implementación

```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL,
        D-DNI   INT   NOT NULL
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir    VARCHAR (200),
   PRIMARY KEY (D-Cod ),
   UNIQUE(D-DNI)
) ;
```

```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL,
        D-DNI   INT   NOT NULL UNIQUE,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir    VARCHAR (200),
  PRIMARY KEY (D-Cod )
) ;
```

CREATE TABLE: Claves Alternativas

- La palabra UNIQUE puede aparecer exactamente en los mismos lugares en los que puede aparecer la palabra PRIMARY KEY.

- En una relación pueden existir varias declaraciones UNIQUE pero sólo una declaración PRIMARY KEY.

- UNIQUE: admite valores nulos $\rightarrow$ agregar NOT NULL para que realmente sea una clave (analizar cada caso en aplicaciones reales).

CREATE TABLE: Claves Alternativas

- Modelo Relacional

Empleados= {E-Legajo, E-TipoDoc, E-NroDoc, E-Nbre, E-email }

dom(E-Legajo)= dom (E-NroDoc) =$\mathbb{N}$

dom (E-TipoDoc)={Pasaporte, DNI, otro}

dom(E-Nbre)=Alfa+

dom(E-email)=AlfaNco+

- Implementación (suponiendo que E-Legajo es la clave principal)

```
CREATE TABLE Empleados (
    E-Legajo    INT NOT NULL,
    E-TipoDoc VARCHAR(9) INT NOT NULL
    E-NroDoc  INT NOT NULL,
    E-Nbre      VARCHAR (200) NOT NULL,
    E-Email     VARCHAR (200) NOT NULL
  PRIMARY KEY (E-Legajo),
  UNIQUE(E-TipoDoc, E-NroDoc)
  UNIQUE (E-email)
) ;
```

```
CREATE TABLE Empleados (
    E-Legajo    INT   NOT NULL,
    E-TipoDoc VARCHAR(9) INT NOT NULL UNIQUE,
    E-NroDoc  INT NOT NULL UNIQUE,
    E-Nbre      VARCHAR (200) NOT NULL
    E-Email     VARCHAR (200) NOT NULL
  PRIMARY KEY (E-Legajo),
  UNIQUE (E-email)
) ;
```

CREATE TABLE: Claves Alternativas

- Modelo Relacional

Empleados= {E-Legajo, E-TipoDoc, E-NroDoc, E-Nbre, E-email }

dom(E-Legajo)= dom (E-NroDoc) =ℕ

dom (E-TipoDoc)={Pasaporte, DNI, otro}

dom(E-Nbre)=Alfa+

dom(E-email)=AlfaNco+

- Implementación (suponiendo que E-Legajo es la clave principal)

```
CREATE TABLE Empleados (
    E-Legajo    INT NOT NULL PRIMARY KEY,
    E-TipoDoc VARCHAR(9) INT NOT NULL
    E-NroDoc  INT NOT NULL
    E-Nbre      VARCHAR (200) NOT NULL,
    E-Email     VARCHAR (200)  UNIQUE
   UNIQUE (E-TipoDoc, E-NroDoc)
) ;
```

```
CREATE TABLE Empleados (
    E-Legajo   INT NOT NULL PRIMARY KEY,
    E-TipoDoc VARCHAR(9) INT NOT NULL,
    E-NroDoc  INT NOT NULL
    E-Nbre      VARCHAR (200) NOT NULL,
    E-Email     VARCHAR (200) NOT NULL UNIQUE,
  UNIQUE (E-TipoDoc, E-NroDoc)
) ;
```

CREATE TABLE: Atributos Opcionales

- Para los atributos opcionales se puede dar un valor por DEFAULT.
- Ejemplo: supongamos que en docentes tenemos el atributo *teléfono* que es opcional

Docentes= {D-Cod, D-DNI, D-Nbre, D-Dir, D-TE }

dom(D-Cod)=ℕ
dom (D-DNI)=ℕ
dom(D-Nbre)=Alfa+
dom(D-Dir)=AlfaNco*
dom (D-TE)=ℕ

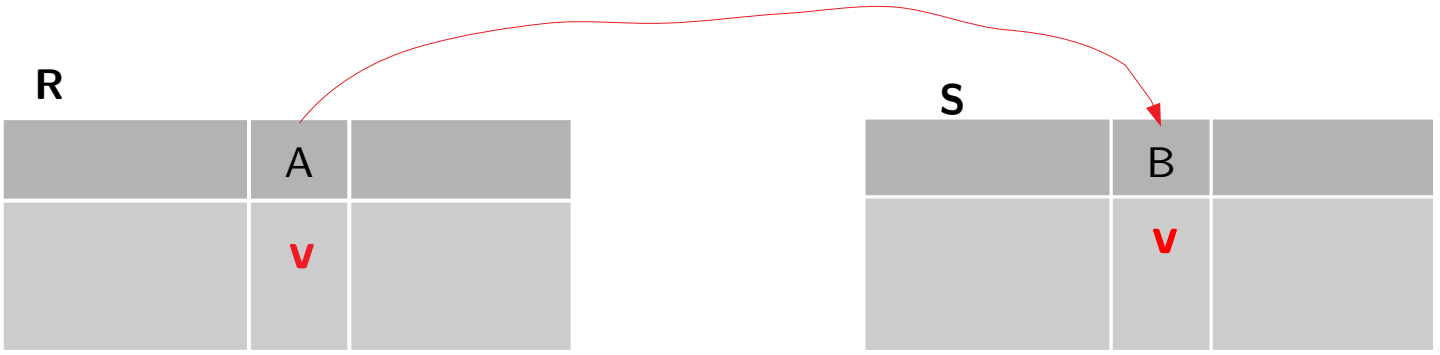
```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL PRIMARY KEY,
        D-DNI   INT   NOT NULL UNIQUE,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir    VARCHAR (200),
        D-TE     INT DEFAULT 0
) ;
```

CREATE TABLE: Atributos Opcionales

- Durante la inserción de nuplas:
  - Si no se da un valor para el atributo opcional y no hay valor por DEFAULT $\rightarrow$ NULL.
  - Si no se da un valor para el atributo opcional y hay valor por DEFAULT $\rightarrow$ se guardará el valor por DEFAULT.

CREATE TABLE: Restricciones de Integridad Referencial

- Ya vimos en el MR lo que significaban las restricciones de integridad referencial:
  - Integridad Referencial entre A(R) y B(S): si un valor v aparece en el atributo A de una nupla de R, entonces **v debe aparecer** en el atributo B de alguna nupla de S.



- En el MR las indicamos con FK (sigla de Foreign Key)

CREATE TABLE: Restricciones de Integridad Referencial

- Durante la creación del esquema de una relación se especifican las restricciones de integridad mediante la cláusula FOREIGN KEY.
- Ejemplo:
  - Modelo relacional

    Tomo= {D-Cod-Tomo, C-Cod-Tomo , Fecha-Tomo }

    dom(D-Cod-Tomo)= dom(C-Cod-Tomo)=dom(Fecha-Tomo)=$\mathbb{N}$

    FK(D-Cod-Tomo)$\rightarrow$ Docentes(D-Cod)

    FK(C-Cod-Tomo) $\rightarrow$ Cursos (C-Cod)

  - Implementación:

```
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo  DATE
    PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo),
    FOREIGN KEY (D-Cod-Tomo) REFERENCES Docentes (D-Cod)
    FOREIGN KEY (C-Cod-Tomo) REFERENCES Cursos (C-Cod)
) ;
```

CREATE TABLE: Restricciones de Integridad Referencial

- Ejemplo:

Docentes= {D-Cod, D-DNI, D-Nbre, D-Dir }
dom(D-Cod)= dom (D-DNI)=$\mathbb{N}$
dom(D-Nbre)=Alfa+
dom(D-Dir)=AlfaNco*

Cursos= {C-Cod, C-Nbre, C-Duracion, D-Cod-Supervisa }
dom(C-Cod)= dom (D-Cod-Supervisa)=$\mathbb{N}$
dom(C-Duracion) ={10,...,80}
dom(C-Nbre)=Alfa+
FK(D-Cod-Supervisa)$\rightarrow$ Docentes(D-Cod)

Tomo= {D-Cod-Tomo, C-Cod-Tomo , Fecha-Tomo }
dom(D-Cod-Tomo)= dom(C-Cod-Tomo)=dom(Fecha-Tomo)=$\mathbb{N}$
FK(D-Cod-Tomo)$\rightarrow$ Docentes(D-Cod)
FK(C-Cod-Tomo) $\rightarrow$ Cursos (C-Cod)

CREATE TABLE: Restricciones de Integridad Referencial

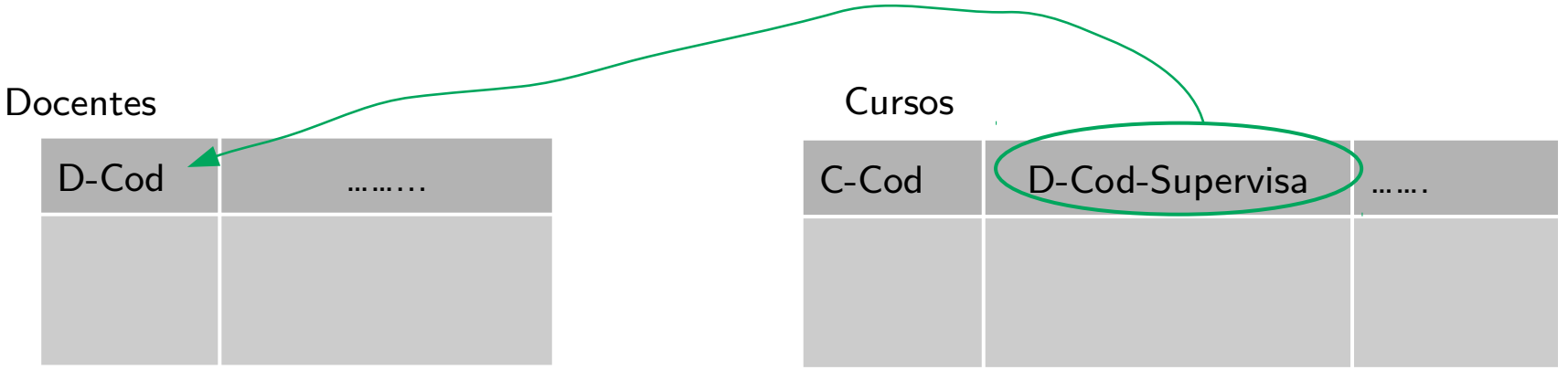
```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL PRIMARY KEY,
        D-DNI   INT   NOT NULL UNIQUE,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir    VARCHAR (200) DEFAULT 'no declara'
) ;

CREATE TABLE Cursos (
        C-Cod    INT NOT NULL PRIMARY KEY,
        C-Nbre   VARCHAR (200) NOT NULL,
        C-Duracion   INT DEFAULT 10,
        D-Cod-Supervisa   INT NOT NULL,
    FOREIGN KEY (D-Cod-Supervisa) REFERENCES Docentes (D-Cod)
) ;
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo  DATE
  PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo),
  FOREIGN KEY (D-Cod-Tomo) REFERENCES Docentes (D-Cod)
  FOREIGN KEY (C-Cod-Tomo) REFERENCES Cursos (C-Cod)
) ;
```

El DBMS controla que en las instancias se mantengan las restricciones establecidas por PRIMARY KEY, UNIQUE y FOREING KEY

CREATE TABLE: Restricciones de Integridad Referencial

```
CREATE TABLE Cursos (
        C-Cod    INT NOT NULL PRIMARY KEY,
        C-Nbre   VARCHAR (200) NOT NULL,
        C-Duracion   INT NOT NULL,
        D-Cod-Supervisa   INT NOT NULL,
    FOREIGN KEY (D-Cod-Supervisa) REFERENCES Docentes (D-Cod)
) ;
```

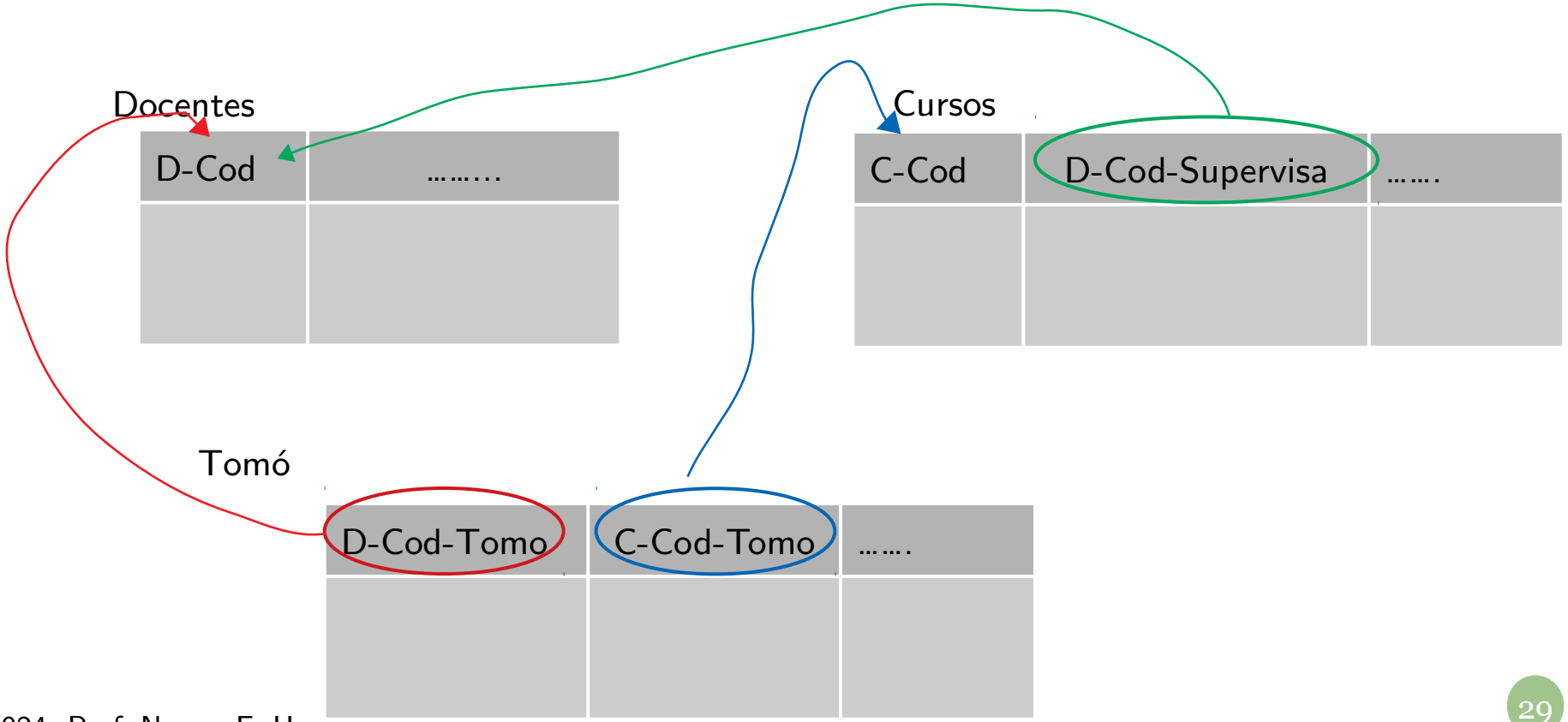


Tomó

D-Cod-Tomo	C-Cod-Tomo	……

CREATE TABLE: Restricciones de Integridad Referencial

```
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo   DATE
    PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo),
    FOREIGN KEY (D-Cod-Tomo) REFERENCES Docentes (D-Cod)
    FOREIGN KEY (C-Cod-Tomo) REFERENCES Cursos (C-Cod)
) ;
```



Modificación de Esquemas: DROP y ALTER

DROP TABLE

- Permite modificar el esquema de la BD eliminando alguna de sus relaciones .
- Sintaxis:

  DROP <Nombre de relación>

- Ejemplo:

  DROP Docentes

- Si hay referencias externas a la relación (FK) no se permitirá la eliminación.
- En estos casos hay una opción para realizar eliminación en cascada.

ALTER TABLE

- Permite modificar el esquema de una relación.
- Sintaxis:

  ALTER TABLE <Nombre de Relación > <OPERACION> [,<OPERACION>, ...]

  Donde la operación puede ser agregar una columna (atributo)
  ADD <Columna> <Tipo> [Restricciones]

  o eliminar una columna (atributo)
  DROP <Columna>

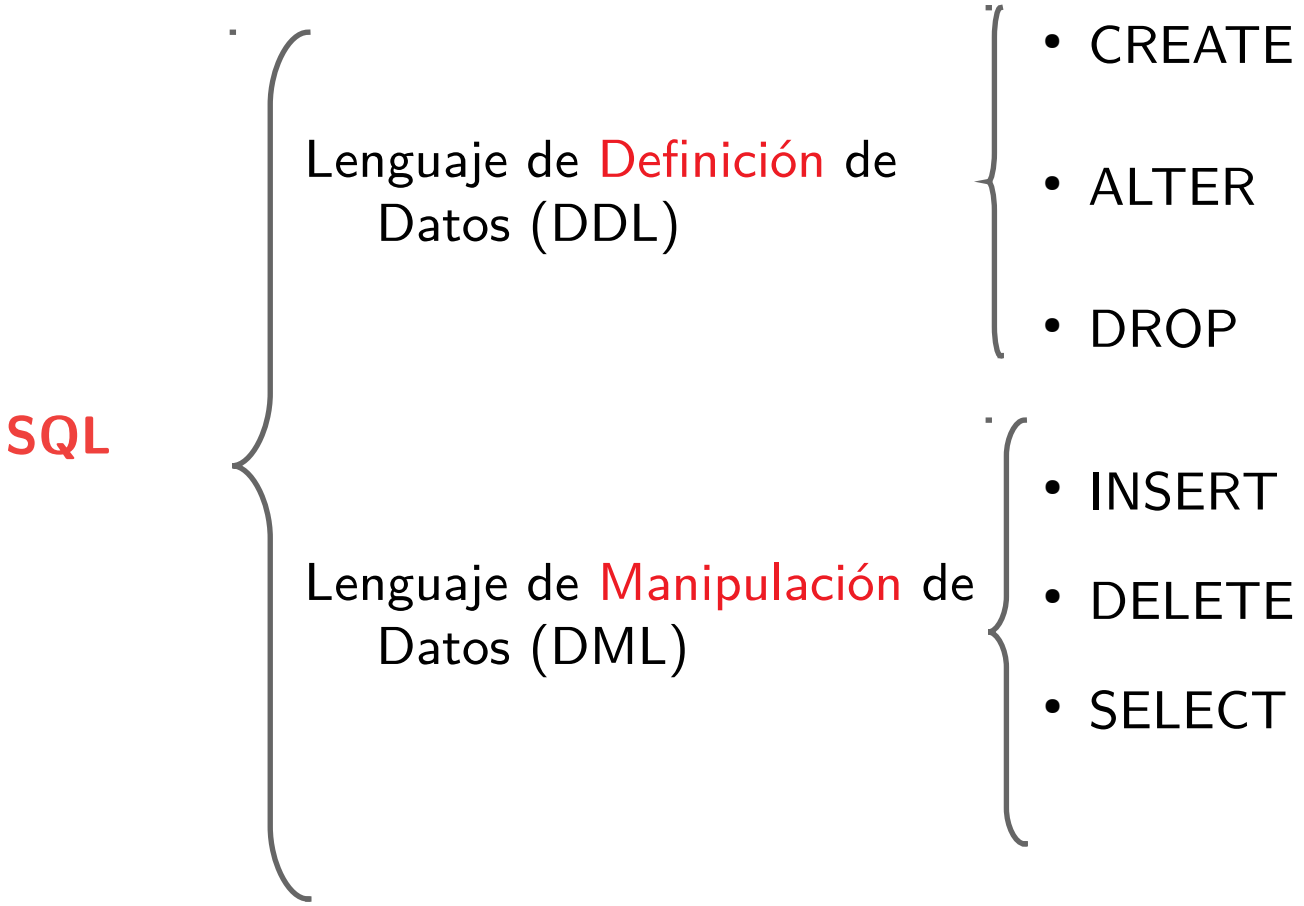
- En la sintaxis: corchetes → opcional, corchete angulares → obligatorio
- Ejemplos:

  ALTER TABLE Docentes ADD D-Titulo VARCHAR(60);

  ALTER TABLE Docentes DROP D-Dir;


SQL: DML
(Data Manipulation Language)

SQL: DDL y DML


SQL
Lenguaje de Definición de Datos (DDL)
• CREATE
• ALTER
• DROP
Lenguaje de Manipulación de Datos (DML)
• INSERT
• DELETE
• SELECT

Modificación de Instancias: INSERT y DELETE

INSERT: Sintaxis

- CREATE TABLE: crea esquema; en ese punto la instancia es vacía.
- INSERT: permite modificar la instancia de una relación insertando nuevas nuplas.
- En este proceso de inserción el SGBD controla las restricciones especificadas durante la creación de la relación (claves y restricciones de integridad).
- Sintaxis:

```
INSERT
INTO  <Nbre Relación>
VALUES <lista de valores>
```

Se da la nupla completa, con los valores dados según el orden usado en el CREATE TABLE

```
INSERT
INTO  <Nbre Relación> [Nbre Atributo,..., Nbre Atributo]
VALUES <lista de valores>
```

Los atributos opcionales pueden no ir; los valores se deben dar según el orden dado en la lista de atributos

INSERT: Ejemplos

```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL PRIMARY KEY,
        D-DNI   INT   NOT NULL UNIQUE,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir     VARCHAR (200) DEFAULT 'no declara'
) ;
```

```
INSERT
INTO  Docentes
VALUES (10,  18111222, 'Juan Pérez', 'Sucre 234');
```

```
INSERT
INTO  Docentes
VALUES ( 20,  18111222, 'Ana Salina', 'España  123');
```

- Si hacemos estos dos INSERT, el segundo nos dará error, ¿por qué?

INSERT: Ejemplos

```
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo  DATE
   PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo),
   FOREIGN KEY (D-Cod-Tomo) REFERENCES Docentes (D-Cod)
   FOREIGN KEY (C-Cod-Tomo) REFERENCES Cursos (C-Cod)
) ;
```

```
INSERT
INTO  Tomó
VALUES ( 10, 20, 10/10/2004);
```

El SGBD controlará que este valor esté en la columna C-Cod de la instancia de Cursos.

El SGBD controlará que este valor esté en la columna D-Cod de la instancia de Docentes

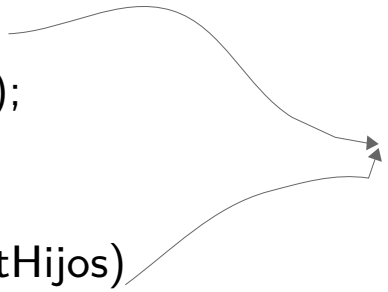
El SGBD controlará que no exista otra nupla en Tomó que tenga esta combinación de valores

INSERT: Ejemplos

```
CREATE TABLE Docentes (
        D-Cod   INT   NOT NULL PRIMARY KEY,
        D-Nbre   VARCHAR (200) NOT NULL,
        D-Dir     VARCHAR (200),
        D-CantHijos     INT DEFAULT 0
) ;
```

```
INSERT
INTO  Docentes (D-Cod, D-Dir, D-Nbre)
VALUES ( 10, 'Sucre 1234', 'Juan Pérez');

INSERT
INTO  Docentes (D-Nbre, D-Cod, D-CantHijos)
VALUES ( 'Ana García', 20, 2);
```



No es necesario respetar el orden en que fueron declarados porque damos la lista de atributos.

D-Cod	D-Nbre	D-Dir	D-CantHijos
10	Juan Pérez	Sucre 123	0
20	Ana García	NULL	2

DELETE

- Permite eliminar nuplas de una instancia de relación.
- Sintaxis:

```
DELETE  FROM <Nbre Relacion>
[ WHERE <condicion> ]
```

Elimina todas las nuplas de la instancia (tabla) que cumplan la condición

- Ejemplos:

```
DELETE  FROM Docentes
WHERE D-Cod = 20
```

```
DELETE  FROM Tomo
WHERE D-Cod-Tomo = 20
```

- El SGBD controla que se respeten las FK: no se pueden eliminar nuplas que están siendo referenciadas desde otras tablas (instancias).

DELETE

```
CREATE TABLE Tomo (
        D-Cod-Tomo   INT   NOT NULL,
        C-Cod-Tomo   INT   NOT NULL,
        Fecha-Tomo   DATE
   PRIMARY KEY (D-Cod-Tomo, C-Cod-Tomo),
   FOREIGN KEY (D-Cod-Tomo) REFERENCES Docentes (D-Cod)
   FOREIGN KEY (C-Cod-Tomo) REFERENCES Cursos (C-Cod)
) ;
```

Docentes

D-Cod	D-Nbre	D-Dir	D-CantHijos
10	n1	d1	0
20	n2	d2	2
30	n1	d3	1

DELETE → controla que no exista en Tomo

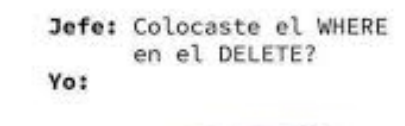
Tomo

D-Cod-Tomo	C-Cod-Tomo	Fecha-Tomo
10	100	f1
30	200	f2

INSERT → controla que exista en Docentes

DELETE

- Cuidado! Si la cláusula WHERE se omite, se eliminan todas las nuplas de la relación → instancia vacía



- Algunos SGBD dan la posibilidad de inhabilitar la opcion de hacer un DELETE sin el WHERE.

DELETE

DELETE FROM Docentes ⟶ Se eliminan todas las nuplas de la instancia, pero la relación sigue existiendo: Modifica instancia, no el esquema

DROP TABLE Docentes ⟶ Se elimina la relación Docentes del esquema de la BD: Modifica Esquema

Realizando Consultas: SELECT

SELECT

- Sintaxis

SELECT <lista de atributos a seleccionar>

FROM < lista de relaciones(tablas) >

[WHERE < condición>]

[GROUP BY < atributo de agrupación>]

[HAVING < condición de agrupacíon>]

[ORDER BY < lista de atributos>]

SELECT

- La consulta más sencilla que podemos tener es:

  SELECT <lista de atributos a seleccionar>

  FROM < lista de relaciones(tablas) >

- En esta clase estudiaremos consultas sobre una única relación(tabla) de la base de datos:

  SELECT <lista de atributos a seleccionar>

  FROM R

SELECT: Consultas Sencillas

```
CREATE TABLE Docentes (
        D-Cod    INT    NOT NULL PRIMARY KEY,
        D-DNI    INT    NOT NULL UNIQUE,
        D-NbreApellido  VARCHAR (200) NOT NULL,
        D-Titulo    VARCHAR (200) NOT NULL
) ;
```

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Titulo
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

SELECT: Consultas Sencillas

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

SELECT <lista de atributos a seleccionar>
FROM R

Consulta :

SELECT D-NbreApellido, D-Título
FROM Docentes

D-NbreApellido	D-Título
María Sosa	Licenciada
Juan Solís	Ingeniero
Ana Pérez	Ingeniero
Jorge Celi	Profesor

- Semántica: obtiene nombre y título de todos los docentes.
- ¿Costo (tiempo)?

SELECT: Uso del Asterisco(*)

- El asterisco se usa para indicar que queremos recuperar todos los atributos.
- Ejemplo:

```
SELECT   *
FROM     Docentes
```

Devolverá toda la instancia de Docentes.

SELECT: Consultas Sencillas

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

SELECT D-Título
FROM Docentes

D-Título
Licenciada
Ingeniero
Ingeniero
Profesor

Las nuplas repetidas no se eliminan automáticamente.

SELECT: Eliminación de Duplicados

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

```
SELECT DISTINCT D-Título
FROM   Docentes
```

Elimina repetidos. Pero cuidado! insume tiempo adicional

D-Título
Licenciada
Ingeniero
Profesor

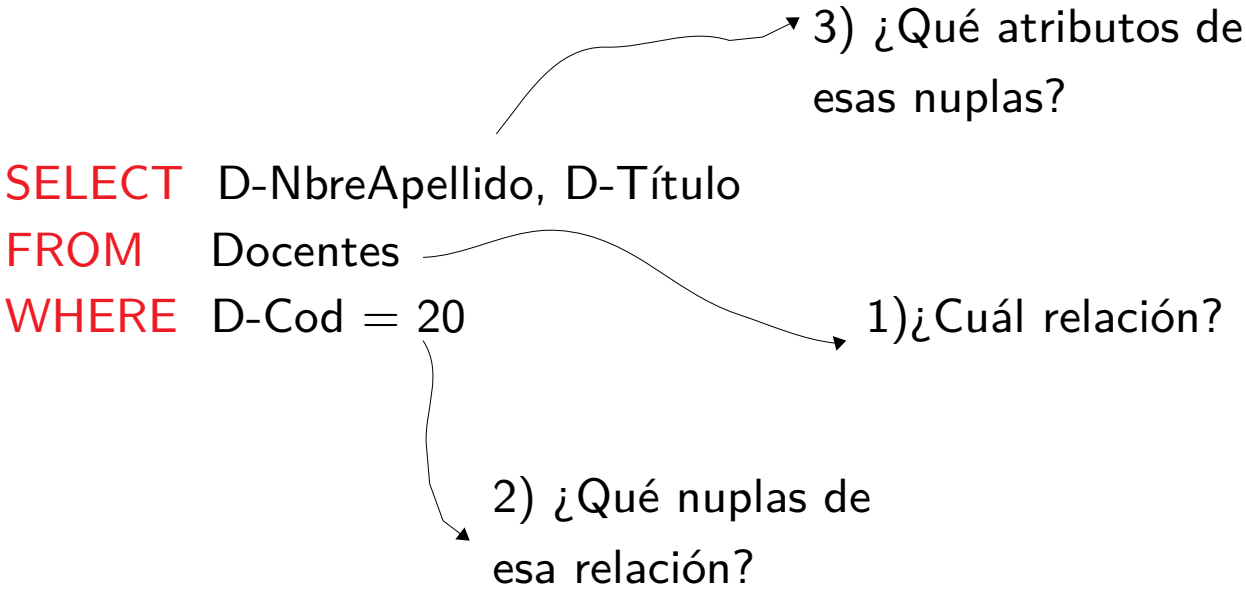
SELECT: Cláusula WHERE

SELECT <lista de atributos a seleccionar>

FROM <R >

[WHERE < condición>]

- En el resultado sólo intervienen las nuplas de R que hacen verdadera la condición especificada en el WHERE.
- Ejemplo:



SELECT: Cláusula WHERE

- En las condiciones podemos usar:
  - Operadores Lógicos:

    AND

    OR

    NOT

  - Operadores de Comparación

    <

    >

    <>

    =

    BETWEEN

    LIKE

    IN

SELECT: Cláusula WHERE

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

```
SELECT   D-NbreApellido, D-Título
FROM     Docentes
WHERE    D-Cod =20
```

D-NbreApellido	D-Título
Juan Solís	Ingeniero

Consulta :

```
SELECT   *
FROM     Docentes
WHERE    D-Cod =20
```

D-Cod	D-DNI	D-NbreApellido	D-Título
20	2222	Juan Solís	Ingeniero

SELECT: Cláusula WHERE

Tomo:

D-Cod-Tomo	C-Cod-Tomo	Fecha-Tomó
10	100	fecha1
10	200	fecha2
20	300	fecha1
30	100	fecha2

Consulta :

SELECT C-Cod-Tomo
FROM Tomo
WHERE D-Cod-Tomo=10

C-Cod-Tomo
100
200

Consulta :

SELECT D-Cod-Tomo
FROM Tomo
WHERE C-Cod-Tomo=100

D-Cod-Tomo
10
30

¿Semántica de las consultas?

SELECT: Cláusula WHERE

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

SELECT D-NbreApellido, D-Título
FROM Docentes
WHERE D-Cod $>$ 20

D-NbreApellido	D-Título
Ana Pérez	Ingeniero
Jorge Celi	Profesor

SELECT: Cláusula WHERE

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

SELECT D-NbreApellido, D-Título
FROM Docentes
WHERE D-Cod =20 OR D-Cod=40

D-NbreApellido	D-Título
Juan Solís	Ingeniero
Jorge Celi	Profesor

SELECT: Cláusula WHERE

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

SELECT D-NbreApellido, D-Título
FROM Docentes
WHERE D-Cod =20 AND D-Cod=40

D-NbreApellido	D-Título

- Es una consulta sintácticamente bien escrita pero que carece de sentido semántico: siempre da como resultado una relación vacía.

SELECT: Cláusula WHERE

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

SELECT D-NbreApellido, D-Título
FROM Docentes
WHERE D-Cod $>$ 20 AND D-Titulo $=$ Profesor

D-NbreApellido	D-Título
Jorge Celi	Profesor

SELECT: Cláusula WHERE

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

- Nombre de los docentes que tengan título de profesor y de los docentes que tengan título de Ingeniero

```
SELECT   D-NbreApellido
FROM     Docentes
WHERE    D-Titulo = Profesor  AND  D-Titulo = Ingeniero
```

SELECT: Funciones Agregadas

- Las funciones agregadas (operadores de agregación) permiten resumir el contendido de una columna en un único valor.
- En el estándar existen 5 operadores de agregación:
    - COUNT: cuenta la cantidad de tuplas en una consulta
    - SUM: regresa la suma de algún atributo numérico.
    - AVG: calcula la media aritmética
    - MAX: obtiene el máximo
    - MIN: obtiene el mínimo

SELECT: Funciones Agregadas

Cursos:

C-Cod	C-Nombre	C-Duracion	D-Cod-Supervisa
1000	Lógica	50	10
2000	Bases de Datos	40	20
3000	Álgebra	90	30
4000	Sistemas Operativos	60	20

```
SELECT  AVG(C-Duracion)
FROM    Cursos
```

AVG (C-Duracion)
60

```
SELECT  MIN(C-Duracion)
FROM    Cursos
```

MIN (C-Duracion)
40

```
SELECT  MAX(C-Duracion)
FROM    Cursos
```

MAX (C-Duracion)
90

¿Semántica de las consultas?

SELECT: Funciones Agregadas

Cursos:

C-Cod	C-Nombre	C-Duracion	D-Cod-Supervisa
1000	Lógica	50	10
2000	Bases de Datos	40	20
3000	Álgebra	90	30
4000	Sistemas Operativos	60	20

```
SELECT  COUNT(C-Duracion)
FROM    Cursos
```

COUNT(C-Duracion)
4

```
SELECT  SUM(C-Duracion)
FROM    Cursos
```

SUM (C-Duracion)
240

¿Semántica de las consultas?

- Hay que ser cuidadosos en elegir el operador acorde a la semántica de la consulta que vamos a resolver.

SELECT: Funciones Agregadas

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

SELECT D-Título
FROM Docentes

D-Título
Licenciada
Ingeniero
Ingeniero
Profesor

SELECT DISTINCT D-Título
FROM Docentes

D-Título
Licenciada
Ingeniero
Profesor

SELECT COUNT(DISTINCT D-Título)
FROM Docentes

COUNT (DISTINCT D-Título)
3

¿Semántica?

SELECT: Funciones Agregadas

Docentes

D-Cod	D-DNI	D-NbreApellido	D-Título
10	1111	María Sosa	Licenciada
20	2222	Juan Solís	Ingeniero
30	3333	Ana Pérez	Ingeniero
40	4444	Jorge Celi	Profesor

Consulta :

```
SELECT  COUNT(D-Cod)
FROM    Docentes
```

¿Semántica?

Consulta :

```
SELECT  COUNT(D-Cod)
FROM    Docentes
WHERE   D-Titulo='Ingeniero'
```

SELECT: Funciones Agregadas

Tomo:

D-Cod-Tomo	C-Cod-Tomo	Fecha-Tomó
10	100	fecha1
10	200	fecha2
20	300	fecha1
30	100	fecha2
10	400	fecha3
30	400	fecha1

Consulta :

```
SELECT  COUNT(C-Cod-Tomo)
FROM    Tomo
WHERE   D-Cod-Tomo=10  OR  D-Cod-Tomo=30
```

Consulta :

```
SELECT  COUNT(DISTINCT C-Cod-Tomo)
FROM    Tomo
WHERE   D-Cod-Tomo=10  OR  D-Cod-Tomo=30
```

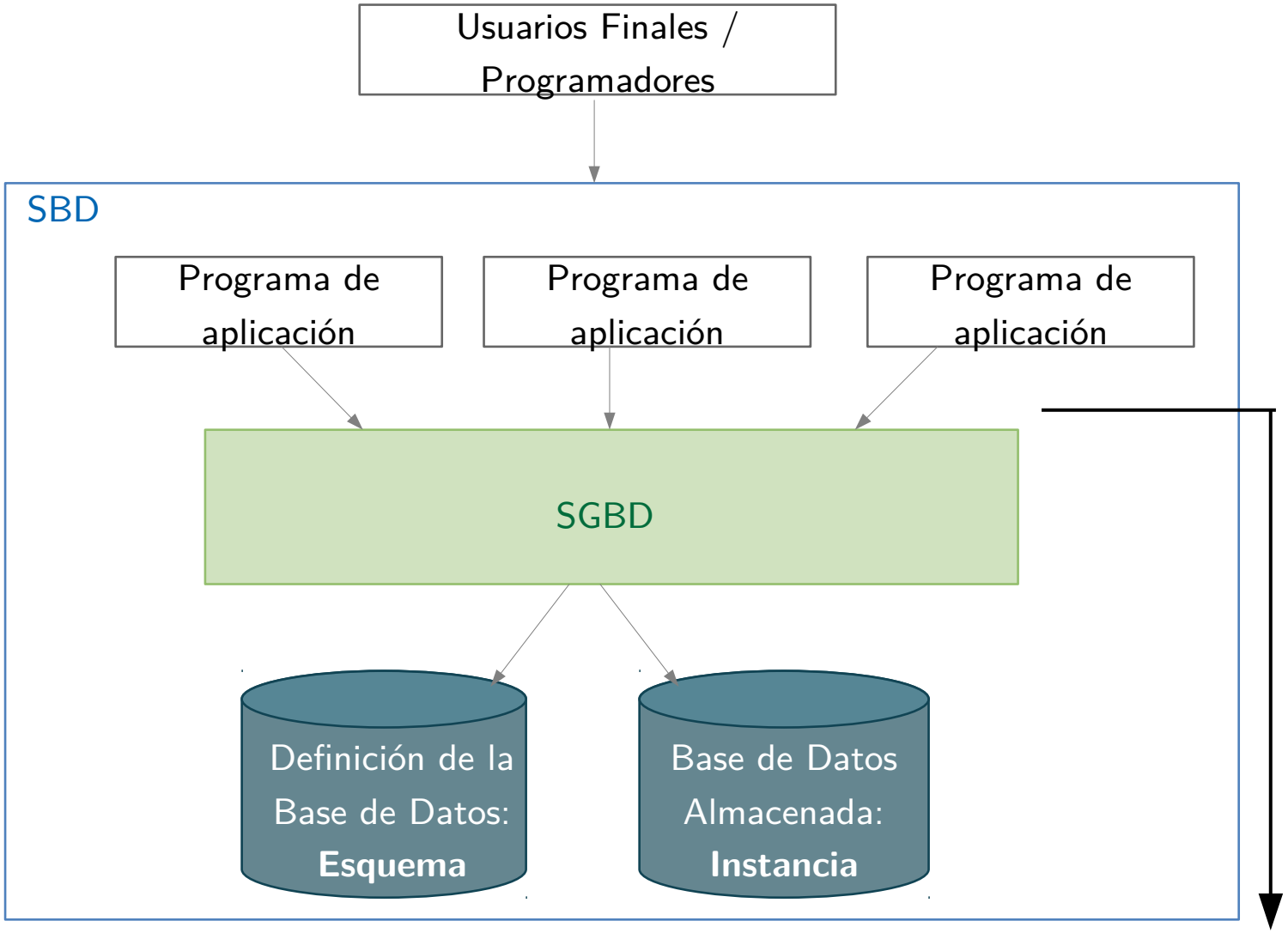
SELECT: Asignación de Alias

- También es posible definir ALIAS para nombres de relaciones y/o atributos usando la palabra clave AS

```
SELECT  C-Cod-Tomo D-Cod-Tomo AS D
FROM    Tomo
WHERE   D=10  OR D=30
```

- Los alias no modifican el esquema.
- Utilidad:
  - Economizar escritura.
  - Mostrar resultados de manera mas amigable al usuario
- Cuando veamos ensambles, veremos la utilidad de usar alias para nombres de relaciones.

RESUMIENDO



CREATE
ALTER
DROP

INSERT
DELETE
SELECT