

SQL Básico

CE3101 - Bases de Datos

Disclaimer / Descargo de Responsabilidad

Esta presentación corresponde a una guía usada por el profesor durante las clases. La misma ha sido modificada para ser utilizado en el modelo de cursos asistidos por tecnología. No es una versión final, por lo que la misma podría requerir todavía hacer algunos ajustes. Para aspectos de evaluación esta presentación es solo una guía, por lo que el estudiante debe profundizar con el material de lectura asignado y lo discutido en clases para aspectos de evaluación.

This presentation corresponds to a guide material used by the professor during classes. It has been modified to be used in the model of technology-assisted courses. It is not a final version, so it may still require some adjustments. For evaluation aspects, this presentation is only a guide, so the student should delve with the assigned reading material and what has been discussed in class.

Introducción

- SQL se considera una de las razones del éxito de las bases de datos relacionales.
- El uso de un lenguaje “estándar” entre motores de distintos fabricantes facilita la migración (se reduce el riesgo de “vendor lock-in”).
- Hay distintas implementaciones de SQL:
 - ◆ PL/SQL
 - ◆ T-SQL
- Todas son muy similares y sencillas de convertir.

Introducción

- SQL se considera una de las razones del éxito de las bases de datos relacionales.
- El uso de un lenguaje “estándar” entre motores de distintos fabricantes facilita la migración (se reduce el riesgo de “vendor lock-in”).
- Hay distintas implementaciones
 - ◆ PL/SQL
 - ◆ T-SQL
- Todas son muy similares y sencillas de convertir.



A qué se refiere esto?

Introducción

- SQL se considera una de las razones del éxito de las bases de datos relacionales.
- El uso de un lenguaje “estándar” entre motores de distintos fabricantes facilita la migración (se reduce el riesgo de “vendor lock-in”).
- Hay distintas implementaciones
 - ◆ PL/SQL
 - ◆ T-SQL



A qué se refiere esto?

→ Cada vendor adopta un estándar pero implementa lo que quiere.

Introducción

- SQL se considera una de las razones del éxito de las bases de datos relacionales.
- El uso de un lenguaje “estándar” entre motores de distintos fabricantes facilita la migración (se reduce el riesgo de “vendor lock-in”).

- Hay distintas implementaciones

- ◆ PL/SQL
- ◆ T-SQL

A qué se refiere esto?

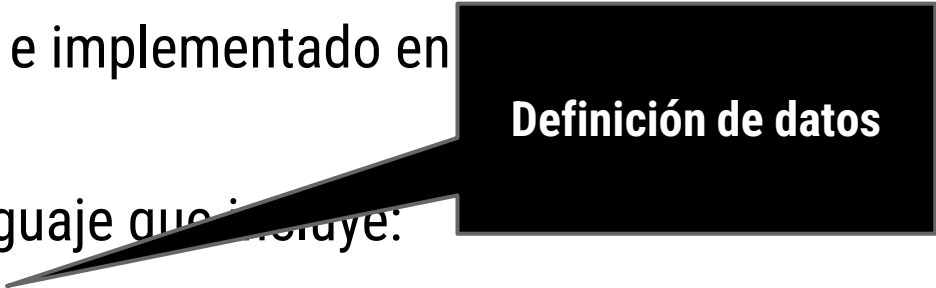
**Ocurre muy a menudo
en el contexto de
software**

- Cada vendor adopta un estándar pero implementa lo que quiere.

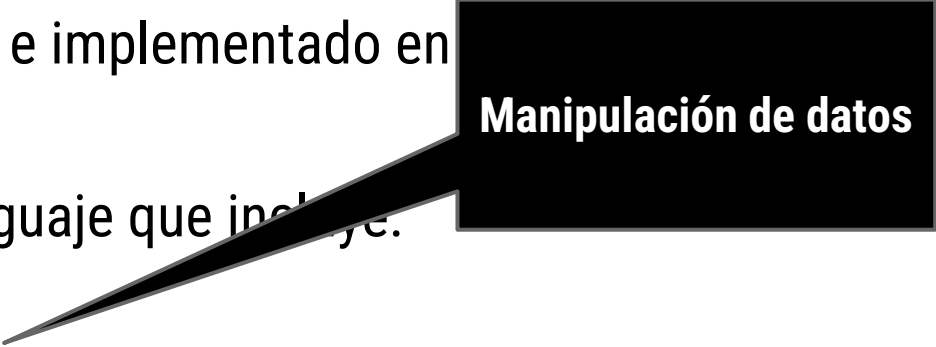
El lenguaje SQL

- SQL es un lenguaje declarativo de alto nivel. El usuario especifica lo que desea obtener y el DBMS se encarga de optimizarlo y ejecutarlo.
- SQL significa Structured Query Language. Originalmente se llamaba SEQUEL (Structured English QUery Language).
- Diseñado e implementado en IBM Research.
- Es un lenguaje que incluye:
 - ◆ DDL
 - ◆ DML
 - ◆ Definición de vistas
 - ◆ Especificación de seguridad y autorización
 - ◆ Integridad referencial y control de transacciones

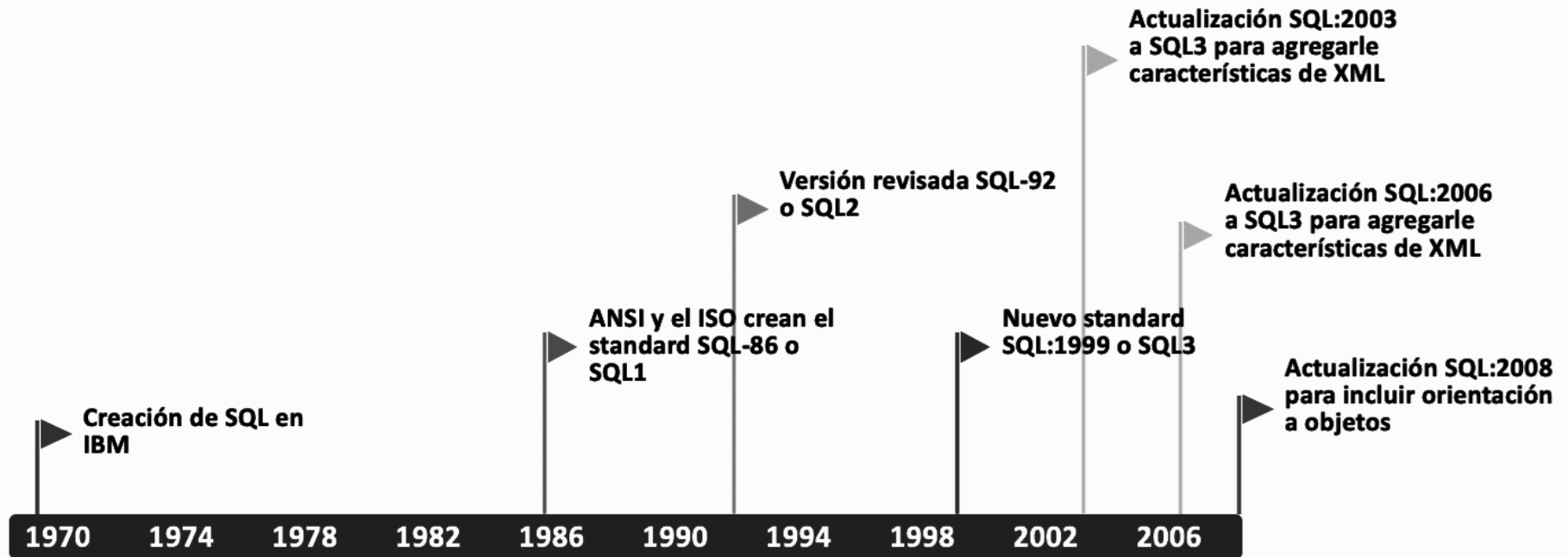
El lenguaje SQL

- SQL es un lenguaje declarativo de alto nivel. El usuario especifica lo que desea obtener y el DBMS se encarga de optimizarlo y ejecutarlo.
- SQL significa Structured Query Language. Originalmente se llamaba SEQUEL (Structured English QUery Language).
- Diseñado e implementado en **Definición de datos**
- Es un lenguaje que incluye:
 - ◆ DDL
 - ◆ DML
 - ◆ Definición de vistas
 - ◆ Especificación de seguridad y autorización
 - ◆ Integridad referencial y control de transacciones

El lenguaje SQL

- SQL es un lenguaje declarativo de alto nivel. El usuario especifica lo que desea obtener y el DBMS se encarga de optimizarlo y ejecutarlo.
- SQL significa Structured Query Language. Originalmente se llamaba SEQUEL (Structured English QUery Language).
- Diseñado e implementado en **Manipulación de datos**
- Es un lenguaje que incluye:
 - ◆ DDL
 - ◆ DML
 - ◆ Definición de vistas
 - ◆ Especificación de seguridad y autorización
 - ◆ Integridad referencial y control de transacciones

Evolución de SQL



Algunos conceptos en SQL

SQL SCHEMA

→ Se identifica por un nombre e incluye autorización a un usuario propietario.

Incluye:

- ◆ Tablas
- ◆ Constraints
- ◆ Vistas
- ◆ Dominios
- ◆ Grants

Algunos conceptos en SQL

SQL SCHEMA

→ Se identifica por un nombre e incluye autorización a un usuario propietario.

Incluye:

- ◆ Tablas
- ◆ Constraints
- ◆ Vistas
- ◆ Dominios
- ◆ Grants

```
CREATE SCHEMA COMPANY AUTHORIZATION 'ISAAC';
```

Algunos conceptos en SQL

SQL SCHEMA

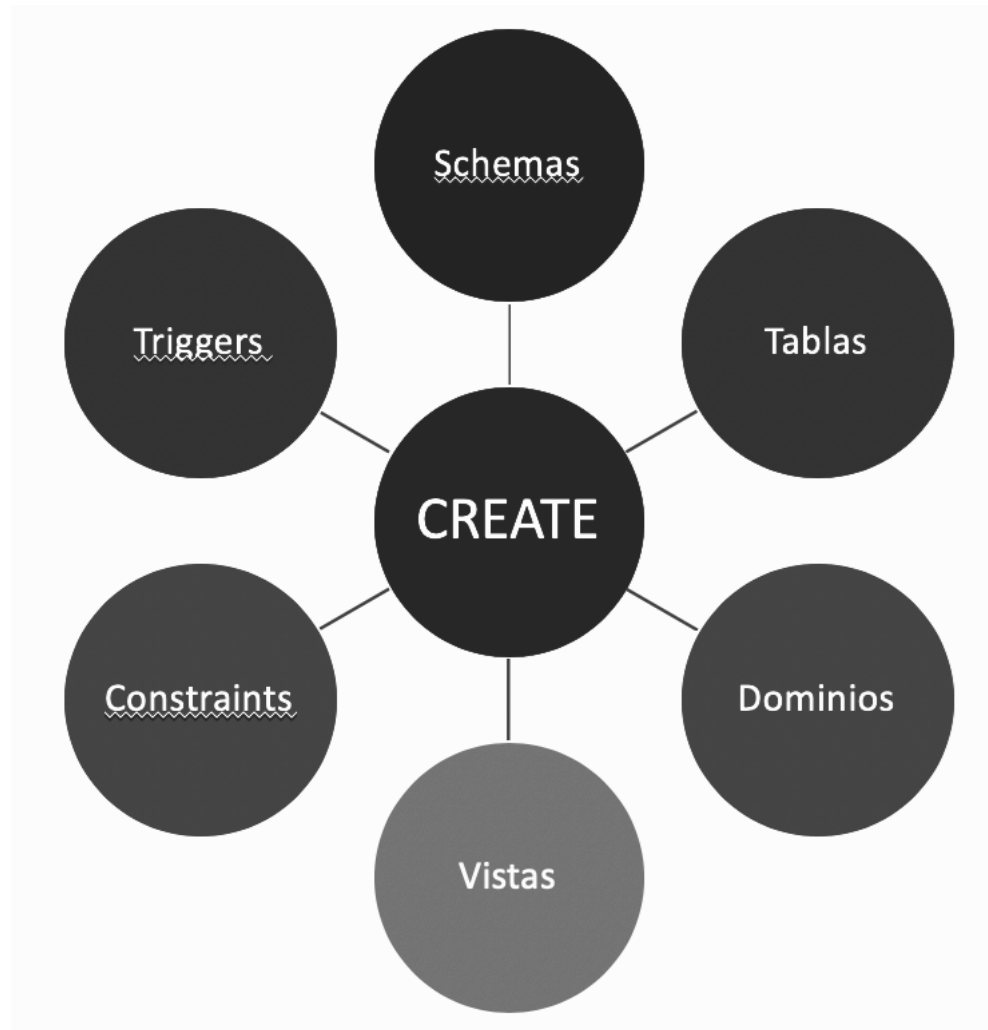
- Se identifica por un nombre e incluye autorización a un usuario propietario. Incluye:
 - ◆ Tablas
 - ◆ Constraints
 - ◆ Vistas
 - ◆ Dominios
 - ◆ Grants

CATALOG

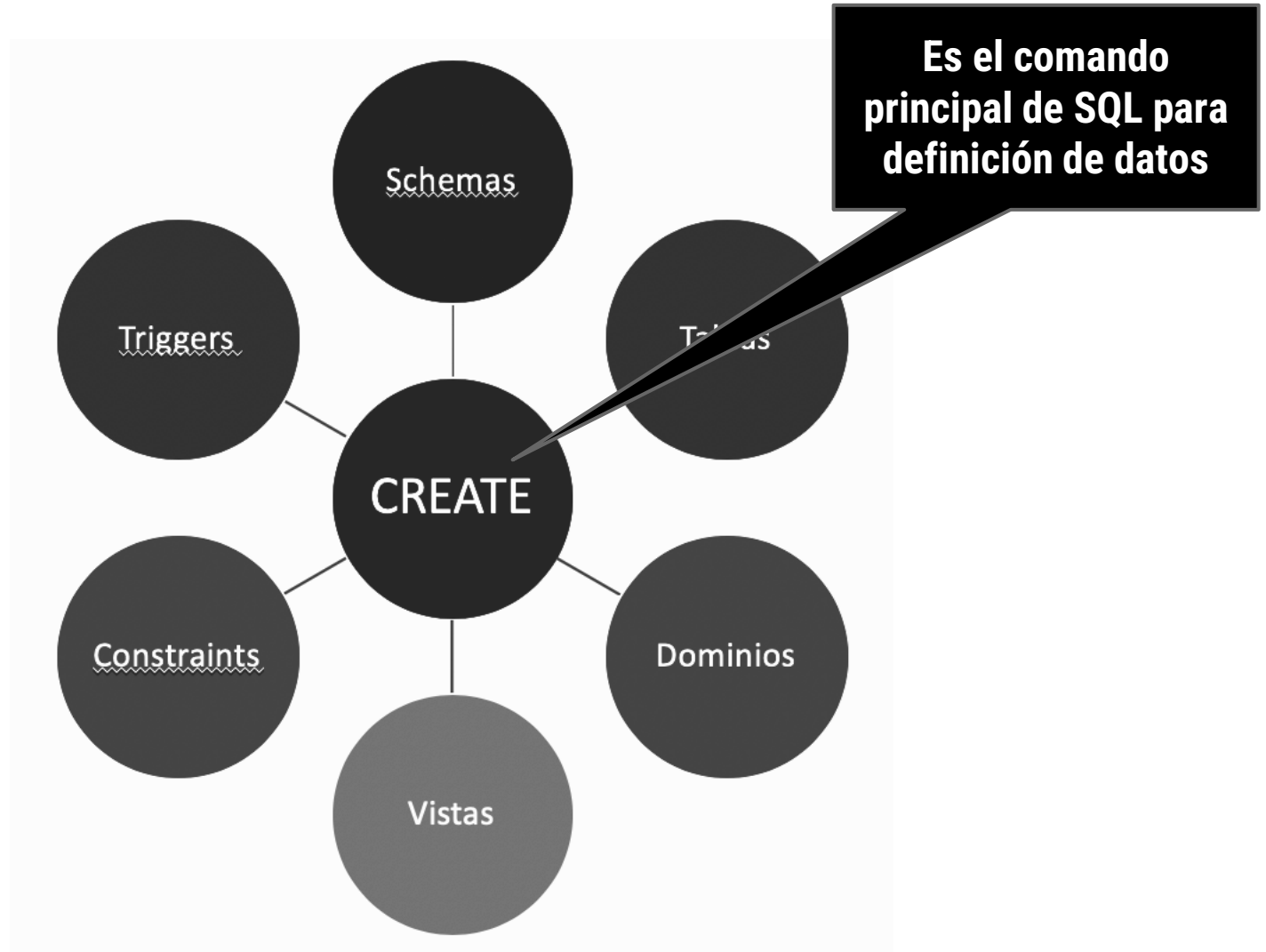
- Conjunto de esquemas SQL.
- Las restricciones de integridad referencial **solo se pueden establecer entre tablas que estén en el mismo catálogo**. Pueden ser de diferentes esquemas.

Data Definition Language (DDL)

CREATE



CREATE



Cómo crear una tabla en SQL?

- Se utiliza el comando `CREATE TABLE`.
- Se especifican los atributos asignándole a cada uno: nombre, tipo de datos (dominio) y constraints por cada atributo.
- La llave e integridad referencial se puede especificar después de los atributos con el mismo comando `CREATE TABLE` o especificar posteriormente con el comando `ALTER TABLE`.

Cómo crear una tabla en SQL?

- Se utiliza el comando CREATE TABLE.
- Se especifican los atributos asignándole a cada uno: nombre, tipo de datos (dominio) y constraints por cada atributo.
- La llave e integridad referencial se puede especificar después de los atributos con el mismo comando CREATE TABLE o especificar posteriormente con el comando ALTER TABLE.



Forma acostumbrada

Qué es una tabla en el contexto de SQL?

- Una tabla en SQL no es exactamente una relación.
- Una tabla en SQL no es un conjunto de tuplas, es más bien un multiset o una bolsa de tuplas.
- En una tabla SQL es posible que existan tuplas con valores iguales en todos sus atributos.
- Si hay una llave establecida, la tabla será un conjunto de tuplas.

Ejemplos

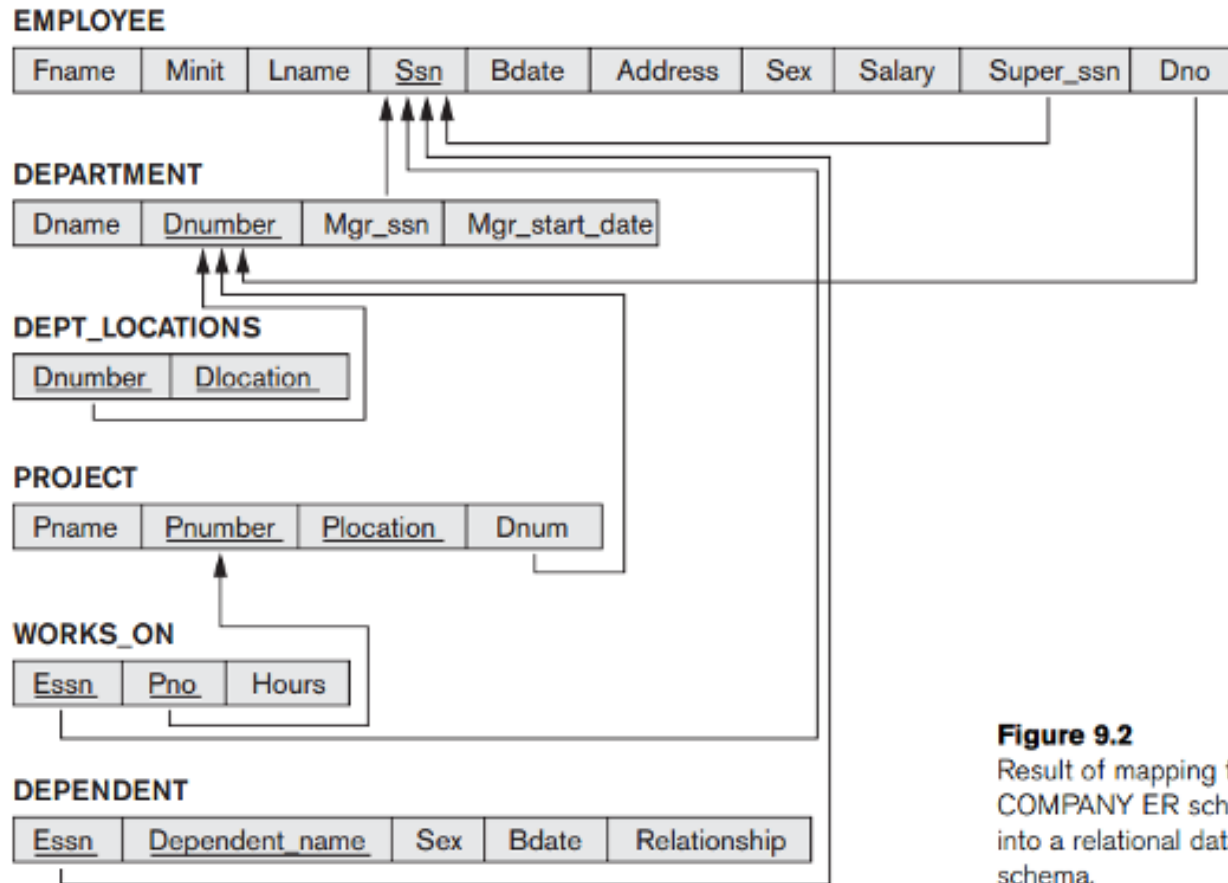


Figure 9.2
Result of mapping the
COMPANY ER schema
into a relational database
schema.

Ejemplos

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        CHAR(9)          NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary      DECIMAL(10,2),
    Super_ssn  CHAR(9),
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno)      REFERENCES DEPARTMENT(Dnumber)
);
```


Ejemplos

Nombre de la columna

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(  Fname      VARCHAR(15)      NOT NULL,
   Minit      CHAR,
   Lname      VARCHAR(15)      NOT NULL,
   Ssn        CHAR(9)          NOT NULL,
   Bdate      DATE,
   Address    VARCHAR(30),
   Sex        CHAR,
   Salary     DECIMAL(10,2),
   Super_ssn  CHAR(9),
   Dno        INT              NOT NULL,
   PRIMARY KEY(Ssn),
   FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
   FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
);
```

Ejemplos

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(  Fname      VARCHAR(15) NOT NULL,
   Minit      CHAR,
   Lname      VARCHAR(15) NOT NULL,
   Ssn        CHAR(9) NOT NULL,
   Bdate      DATE,
   Address     VARCHAR(30),
   Sex        CHAR,
   Salary      DECIMAL(10,2),
   Super_ssn   CHAR(9),
   Dno         INT NOT NULL,
   PRIMARY KEY(Ssn),
   FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
   FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
);
```



Tipo de dato de la columna

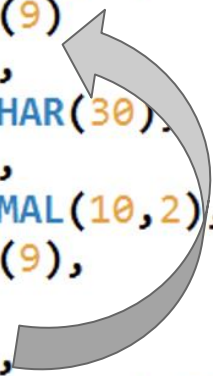
Ejemplos

NOT NULL Constraint.
Por defecto se permite
NULL

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        CHAR(9)          NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary      DECIMAL(10,2),
    Super_ssn  CHAR(9),
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
);
```


Ejemplos

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        CHAR(9)          NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary      DECIMAL(10,2),
    Super_ssn  CHAR(9),
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
);
```



Ejemplos

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        CHAR(9)          NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary     DECIMAL(10,2),
    Super_ssn  CHAR(9),
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
);
```

The diagram illustrates the relationships defined in the SQL code. A thick grey curved arrow originates from the `Super_ssn` column and points to the `Ssn` column, indicating a self-referencing relationship (transitive dependency). A thinner grey curved arrow originates from the `Dno` column and points to the `DEPARTMENT(Dnumber)` table, indicating a foreign key relationship.

Ejemplos

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        CHAR(9)          NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary      DECIMAL(10,2),
    Super_ssn  CHAR(9),
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
);
```

Antes de poder crear la
tabla EMPLOYEE, tiene
que existir la tabla
DEPARTMENT

Ejemplos

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        CHAR(9)          NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary      DECIMAL(10,2),
    Super_ssn  CHAR(9),
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno)      REFERENCES DEPARTMENT(Dnumber)
);
```

Utilizar un ALTER
después

Crear las tablas sin integridad referencial

```
-- Creación de la tabla EMPLOYEE
CREATE TABLE EMPLOYEE
(  Fname      VARCHAR(15)      NOT NULL,
   Minit      CHAR,
   Lname      VARCHAR(15)      NOT NULL,
   Ssn        CHAR(9)          NOT NULL,
   Bdate      DATE,
   Address    VARCHAR(30),
   Sex        CHAR,
   Salary     DECIMAL(10,2),
   Super_ssn  CHAR(9),
   Dno        INT              NOT NULL,
   PRIMARY KEY(Ssn),
   FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
);

-- Creación de la tabla DEPARTMENT
CREATE TABLE DEPARTMENT
(  Dname      VARCHAR(15) NOT NULL,
   Dnumber    INT         NOT NULL,
   Mgr_ssn    CHAR(9)     NOT NULL,
   Mgr_start_date DATE,
   PRIMARY KEY (Dnumber),
   UNIQUE (Dname)
);
```

Crear las tablas sin integridad referencial

```
-- Crear la tabla DEPT_LOCATIONS
CREATE TABLE DEPT_LOCATIONS
(   Dnumber      INT          NOT NULL,
    Dlocation    VARCHAR(15)  NOT NULL,
    PRIMARY KEY (Dnumber, Dlocation)
);

-- Crear la tabla PROJECT
CREATE TABLE PROJECT
(   Pname        VARCHAR(15)  NOT NULL,
    Pnumber      INT          NOT NULL,
    Plocation    VARCHAR(15),
    Dnum         INT          NOT NULL,
    PRIMARY KEY (Pnumber),
    UNIQUE (Pname)
);

-- Crear la tabla WORKS_ON
CREATE TABLE WORKS_ON
(   Essn        CHAR(9)       NOT NULL,
    Pno         INT          NOT NULL,
    Hours       DECIMAL(3,1)  NOT NULL,
    PRIMARY KEY (Essn, Pno)
);
```

Crear las tablas sin integridad referencial

```
-- Crear la tabla DEPENDENT
CREATE TABLE DEPENDENT
(
    Essn          CHAR(9)      NOT NULL,
    Dependent_name VARCHAR(15) NOT NULL,
    Sex           CHAR,
    Bdate         DATE,
    Relationship   VARCHAR(8),
    PRIMARY KEY (Essn, Dependent_name)
);
```

Modificar las tablas para agregar integridad referencial

```
ALTER TABLE EMPLOYEE
ADD CONSTRAINT EMPLOYEE_DEPARTMENT_FK FOREIGN KEY (Dno)
REFERENCES DEPARTMENT(Dnumber);
```

```
ALTER TABLE DEPARTMENT
ADD CONSTRAINT DEPARTMENT_EMPLOYEE_FK FOREIGN KEY (Mgr_ssn)
REFERENCES EMPLOYEE(Ssn);
```

```
ALTER TABLE DEPT_LOCATIONS
ADD CONSTRAINT DEPT_LOCATIONS_DEPARMENT_FK FOREIGN KEY (Dnumber)
REFERENCES DEPARTMENT(Dnumber);
```

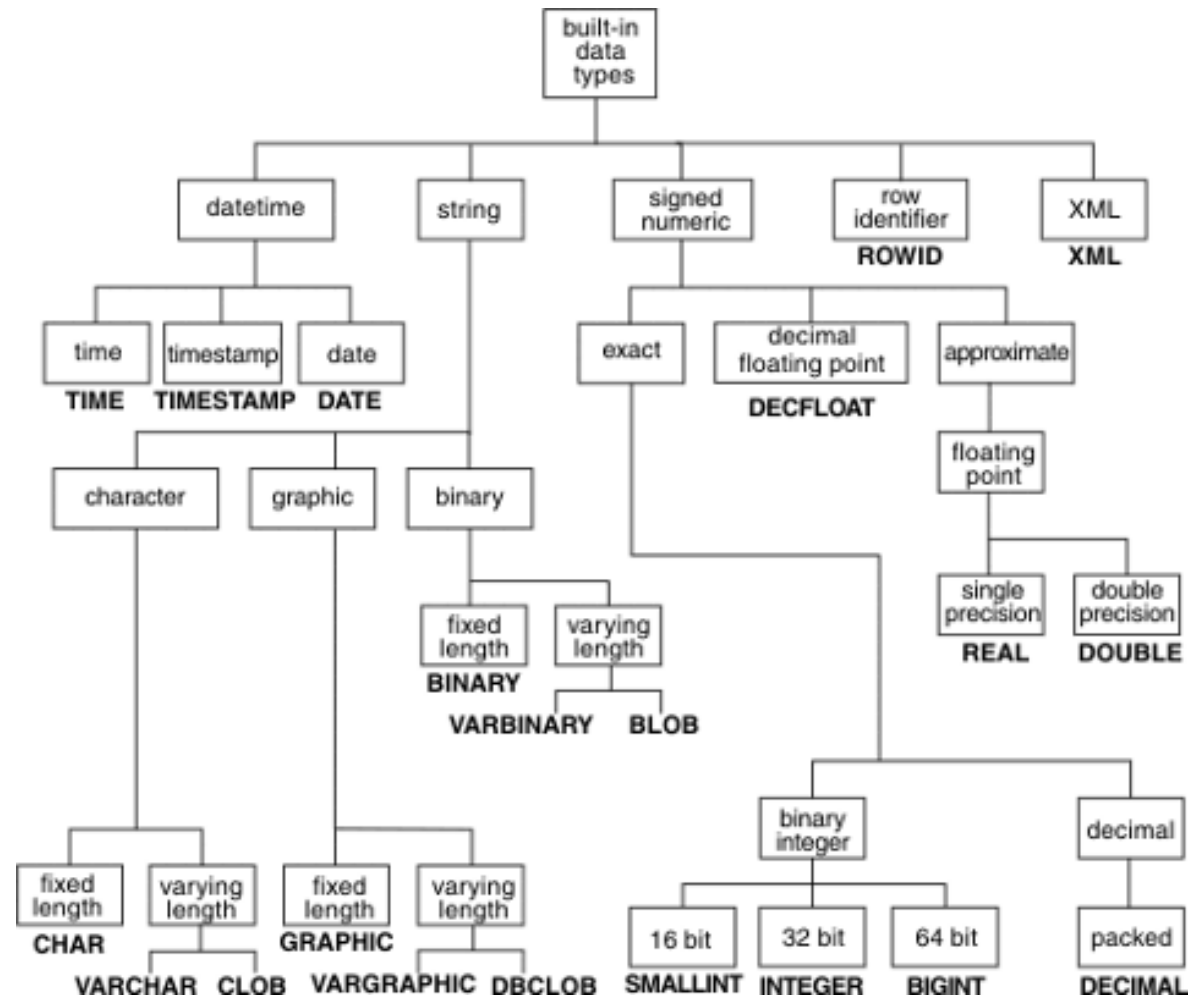
```
ALTER TABLE PROJECT
ADD CONSTRAINT PROJECT_DEPARTMENT_FK FOREIGN KEY (Dnum)
REFERENCES DEPARTMENT(Dnumber);
```

```
ALTER TABLE WORKS_ON
ADD CONSTRAINT WORKS_ON_EMPLOYEE_FK FOREIGN KEY (Essn)
REFERENCES EMPLOYEE(Ssn);
```

```
ALTER TABLE WORKS_ON
ADD CONSTRAINT WORKS_ON_PROJECT_FK FOREIGN KEY (Pno)
REFERENCES PROJECT(Pnumber);
```

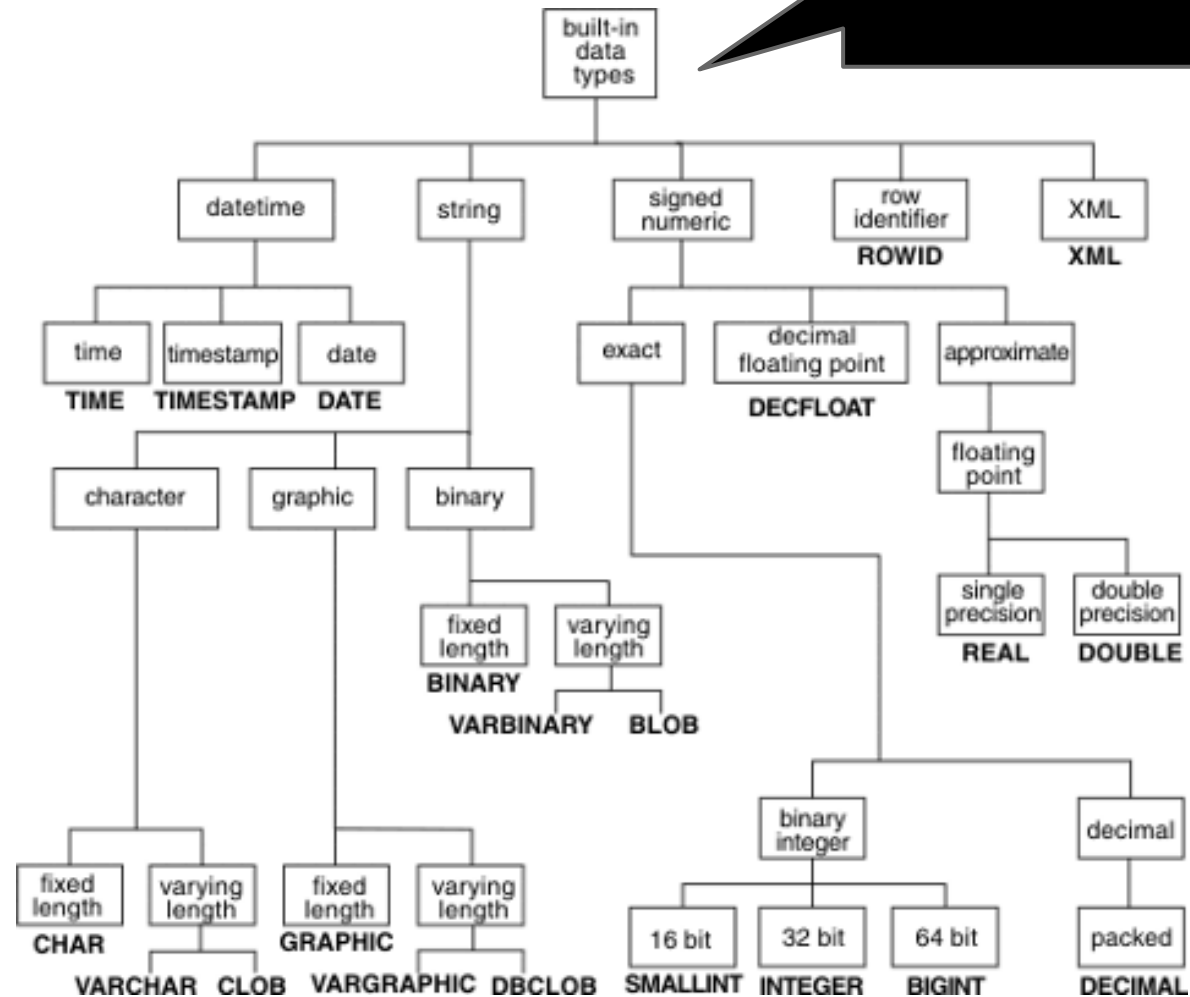
```
ALTER TABLE "DEPENDENT"
ADD CONSTRAINT DEPEDENDT_EMPLOYEE_FK FOREIGN KEY (Essn)
REFERENCES EMPLOYEE(Ssn);
```


Tipos de datos

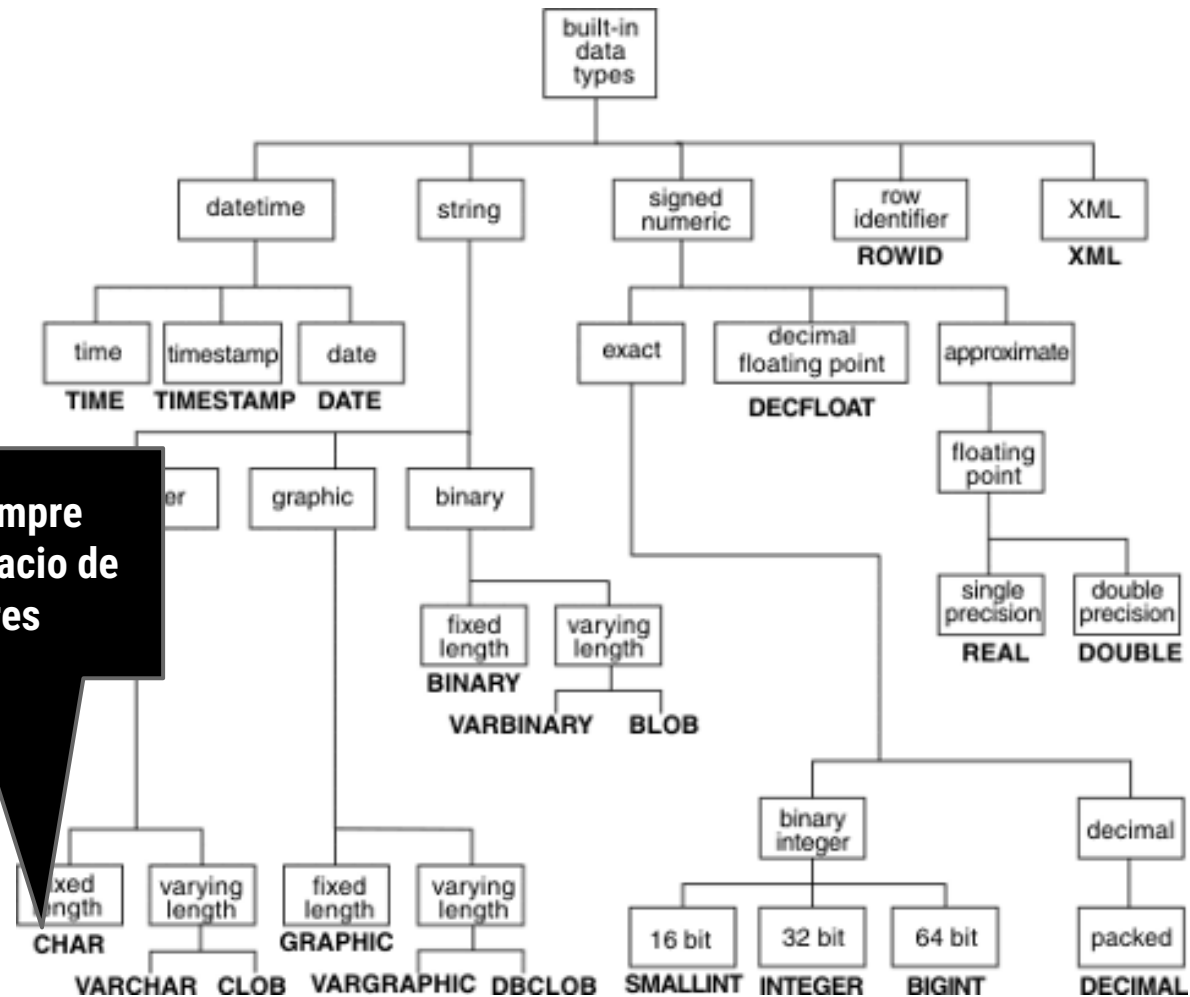


Tipos de datos

Escoger los tipos cuidadosamente

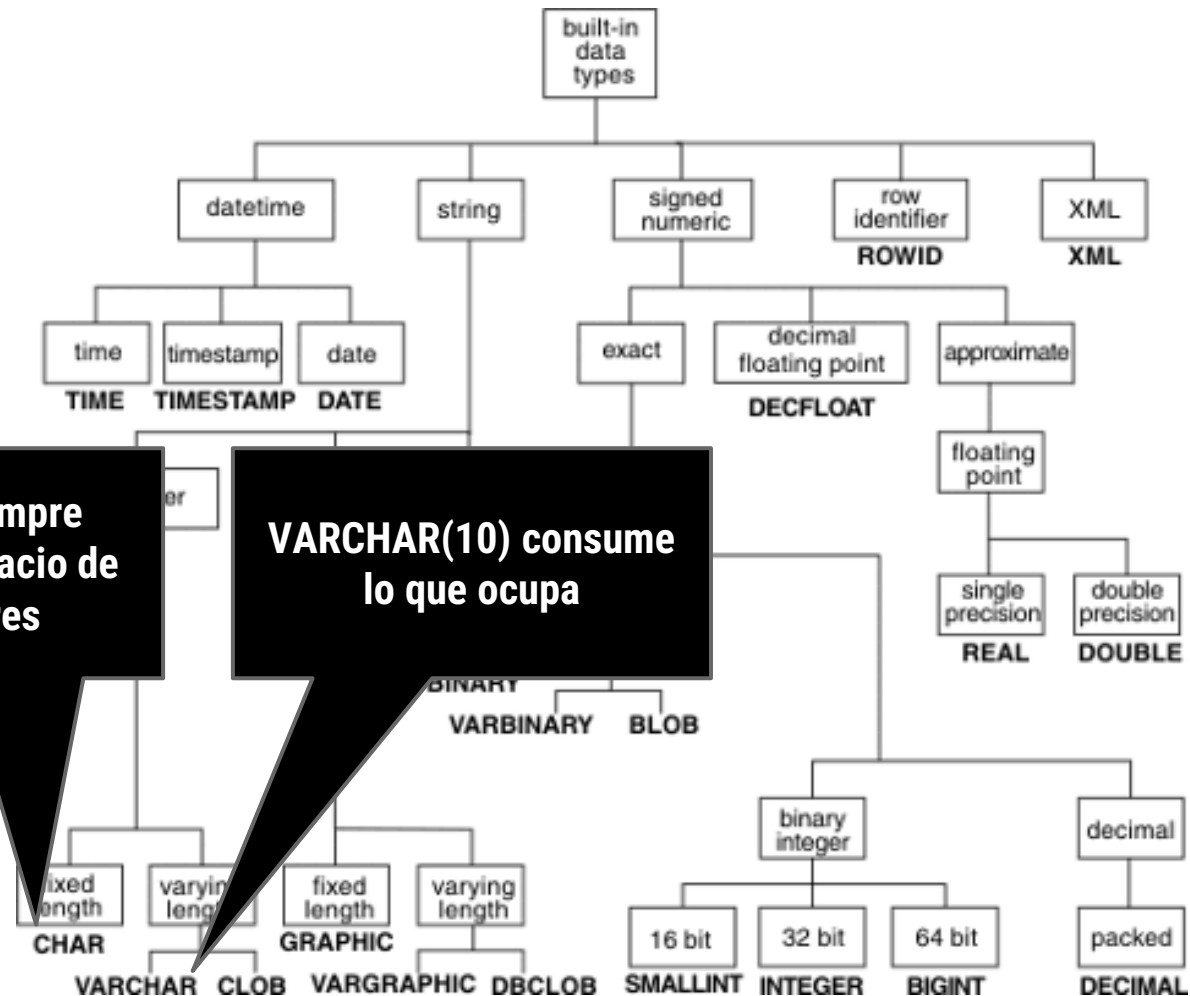


Tipos de datos



CHAR(10) siempre consume el espacio de 10 caracteres

Tipos de datos



CHAR(10) siempre consume el espacio de 10 caracteres

VARCHAR(10) consume lo que ocupa

Cómo se crea un dominio en SQL?

→ No es soportado en todos los DBMS

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```

→ En SQL Server:

```
CREATE TYPE SSN_TYPE FROM CHAR(9) NOT NULL;
```

→ Con el dominio creado, se puede utilizar como si fuera un tipo de datos más.

→ El dominio puede incluir validaciones

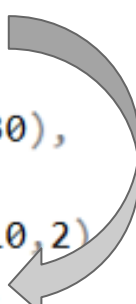
```
CREATE DOMAIN D_NUM AS INTEGER  
CHECK (D_NUM > 0 AND D_NUM < 21);
```

Cómo se asigna un dominio en SQL?

```
CREATE TABLE EMPLOYEE_2
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        SSN_TYPE         NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary     DECIMAL(10,2),
    Super_ssn  SSN_TYPE,
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE_2(Ssn)
);
```

Cómo se asigna un dominio en SQL?

```
CREATE TABLE EMPLOYEE_2
(   Fname      VARCHAR(15)      NOT NULL,
    Minit      CHAR,
    Lname      VARCHAR(15)      NOT NULL,
    Ssn        SSN_TYPE         NOT NULL,
    Bdate      DATE,
    Address    VARCHAR(30),
    Sex        CHAR,
    Salary     DECIMAL(10,2),
    Super_ssn  SSN_TYPE,
    Dno        INT              NOT NULL,
    PRIMARY KEY(Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE_2(Ssn)
);
```

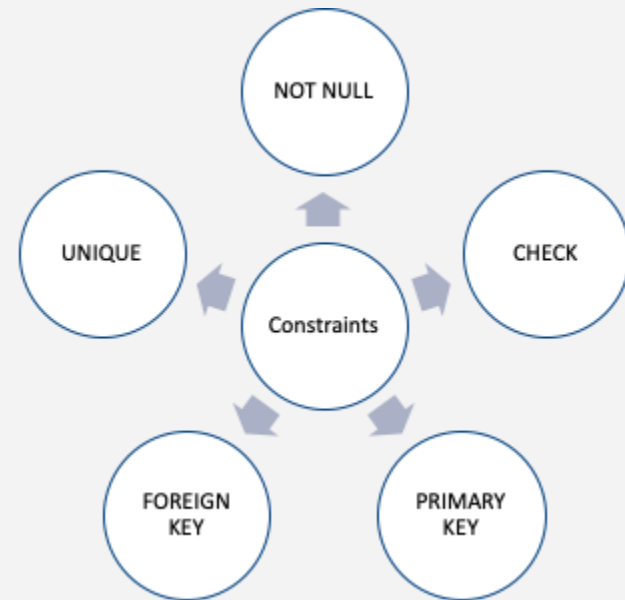


Qué es un constraint?

- Es una restricción impuesta sobre el valor de un atributo.
- Hay varios constraints.
- Se les puede asignar un nombre único.

Qué es un constraint?

- Es una restricción impuesta sobre el valor de un atributo.
- Hay varios constraints.
- Se les puede asignar un nombre único.



Qué es un constraint?

```
CREATE TABLE DEPARTMENT
(   Dname          VARCHAR(15) NOT NULL,
    Dnumber        INT          NOT NULL CHECK(DNumber > 0 AND DNumber < 21),
    Mgr_ssn        CHAR(9)      NOT NULL,
    Mgr_start_date DATE,
    PRIMARY KEY (Dnumber),
    UNIQUE (Dname)
);
```

Qué es un constraint?

```
CREATE TABLE DEPARTMENT
(   Dname          VARCHAR(15) NOT NULL,
    Dnumber        INT          NOT NULL CHECK(DNumber > 0 AND DNumber < 21),
    Mgr_ssn        CHAR(9)      NOT NULL,
    Mgr_start_date  DATE,
    PRIMARY KEY (Dnumber),
    UNIQUE (Dname)
);
```

Cómo se especifica una llave primaria?

→ Se puede establecer en el CREATE o ALTER

→ Si la llave primaria es solo un atributo se puede especificar:

```
Dnumber INT PRIMARY KEY;
```

→ Si es una llave compuesta debe declararse al final del create:

```
PRIMARY KEY(Ssn)
```

→ Normalmente, es más fácil establecerla mediante un ALTER

Referential Triggered Action

```
CREATE TABLE EMPLOYEE
(
    ...,
    Dno      INT          NOT NULL      DEFAULT 1,
    CONSTRAINT EMPCHK
    PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
        ON DELETE SET NULL      ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
    FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
        ON DELETE SET DEFAULT    ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
(
    ...,
    Mgr_ssn  CHAR(9)      NOT NULL      DEFAULT '888665555',
    ...,
    CONSTRAINT DEPTPK
    PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
    UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
    FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
        ON DELETE SET DEFAULT    ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
(
    ...,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
        ON DELETE CASCADE        ON UPDATE CASCADE);
```

Data Manipulation Language (DML)

Sentencia SELECT

DESCRIPCIÓN

- Sentencia para recuperar datos de la base de datos.
- No es igual que el operador SELECT (σ) del álgebra relacional. Combina varias operaciones relacionadas.

SINTAXIS BASICA

```
SELECT    <attribute list>  
FROM      <table list>  
WHERE     <condition>;
```

Sentencia SELECT

DESCRIPCIÓN

- Sentencia para recuperar datos de la base de datos.
- No es igual que el operador SELECT (σ) del álgebra relacional. Combina varias operaciones relacionadas.

SINTAXIS BASICA

**Lista de atributos que se
desean obtener
(PROJECT)**

```
SELECT    <attribute list>  
FROM      <table list>  
WHERE     <condition>;
```


Sentencia SELECT

DESCRIPCIÓN

- Sentencia para recuperar datos de la base de datos.
- No es igual que el operador SELECT (σ) del álgebra relacional. Combina varias operaciones relacionadas.

SINTAXIS BASICA

Lista de atributos que se
desean obtener
(PROJECT)
Puede ser *

```
SELECT    <attribute list>  
FROM      <table list>  
WHERE     <condition>;
```

Sentencia SELECT

DESCRIPCIÓN

- Sentencia para recuperar datos de la base de datos.
- No es igual que el operador SELECT (σ) del álgebra relacional. Combina varias operaciones relacionadas.

SINTAXIS BASICA

SELECT
FROM
WHERE

<attribute>
<table>
<condition>

Lista de atributos que se
desean obtener
(PROJECT)
Puede ser *



Debe evitarse

Sentencia SELECT

DESCRIPCIÓN

- Sentencia para recuperar datos de la base de datos.
- No es igual que el operador SELECT (σ) del álgebra relacional. Combina varias operaciones relacionadas.

SINTAXIS BASICA

```
SELECT    <attribute list>  
FROM      <table list>  
WHERE     <condition>;
```

Condición de selección

Sentencia SELECT

DESCRIPCIÓN

- Sentencia para recuperar datos de la base de datos.
- No es igual que el operador SELECT (σ) del álgebra relacional. Combina varias operaciones relacionadas

SINTAXIS BASICA

```
SELECT    <attribute list>  
FROM      <table list>  
WHERE     <condition>;
```

Hay un iterador implícito que recorre cada tupla y evalúa el where para cada tupla.

Sentencia SELECT (Ejemplos)



```
SELECT  Bdate, Address  
FROM    EMPLOYEE  
WHERE   Fname='John' AND Minit='B' AND Lname='Smith';
```

```
SELECT  Fname, Lname, Address  
FROM    EMPLOYEE, DEPARTMENT  
WHERE   Dname='Research' AND Dnumber=Dno;
```

```
SELECT  Pnumber, Dnum, Lname, Address, Bdate  
FROM    PROJECT, DEPARTMENT, EMPLOYEE  
WHERE   Dnum=Dnumber AND Mgr_ssn=Ssn AND  
        Plocation='Stafford';
```

```
SELECT  Fname, EMPLOYEE.Name, Address  
FROM    EMPLOYEE, DEPARTMENT  
WHERE   DEPARTMENT.Name='Research' AND  
        DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Sentencia SELECT (Ejemplos)

```
SELECT    Bdate, Address
FROM      EMPLOYEE
WHERE     Fname='John' AND Minit='B' AND Lname='Smith';
```



```
SELECT    Fname, Lname, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dname='Research' AND Dnumber=Dno;
```

```
SELECT    Pnumber, Dnum, Lname, Address, Bdate
FROM      PROJECT, DEPARTMENT, EMPLOYEE
WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn AND
          Plocation='Stafford';
```

```
SELECT    Fname, EMPLOYEE.Name, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     DEPARTMENT.Name='Research' AND
          DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Sentencia SELECT (Ejemplos)

```
SELECT    Bdate, Address
FROM      EMPLOYEE
WHERE     Fname='John' AND Minit='B' AND Lname='Smith';
```

```
SELECT    Fname, Lname, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dname='Research' AND Dnumber=Dno;
```



```
SELECT    Pnumber, Dnum, Lname, Address, Bdate
FROM      PROJECT, DEPARTMENT, EMPLOYEE
WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn AND
          Plocation='Stafford';
```

```
SELECT    Fname, EMPLOYEE.Name, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     DEPARTMENT.Name='Research' AND
          DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```


Sentencia SELECT (Ejemplos)

```
SELECT    Bdate, Address
FROM      EMPLOYEE
WHERE     Fname='John' AND Minit='B' AND Lname='Smith';
```

```
SELECT    Fname, Lname, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dname='Research' AND Dnumber=Dno;
```

```
SELECT    Pnumber, Dnum, Lname, Address, Bdate
FROM      PROJECT, DEPARTMENT, EMPLOYEE
WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn AND
          Plocation='Stafford';
```




```
SELECT    Fname, EMPLOYEE.Name, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     DEPARTMENT.Name='Research' AND
          DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```


Sentencia SELECT (Ejemplos)

```
SELECT  Bdate, Address
FROM    EMPLOYEE
WHERE   Fname='John' AND Minit='B' AND Lname='Smith';
```

```
SELECT  Fname, Lname, Address
FROM    EMPLOYEE, DEPARTMENT
WHERE   Dname='Research' AND Dnumber=Dno;
```

```
SELECT  Pnumber, Dnum, Lname, Address, Bdate
FROM    PROJECT, DEPARTMENT, EMPLOYEE
WHERE   Dnum=Dnumber AND Mgr_ssn=Ssn AND
        Plocation='Stafford';
```



```
SELECT  Fname, EMPLOYEE.Name, Address
FROM    EMPLOYEE, DEPARTMENT
WHERE   DEPARTMENT.Name='Research' AND
        DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Previene la ambigüedad

Sentencia SELECT (Ejemplos)



```
SELECT      E.Fname, E.Lname, S.Fname, S.Lname  
FROM        EMPLOYEE AS E, EMPLOYEE AS S  
WHERE       E.Super_ssn=S.Ssn;
```

```
SELECT      Ssn, Dname  
FROM        EMPLOYEE, DEPARTMENT;
```

```
SELECT      *  
FROM        EMPLOYEE  
WHERE       Dno=5;
```

Sentencia SELECT (Ejemplos)

Renombra la tabla



```
SELECT      E.Fname, E.Lname, S.Fname, S.Lname
FROM        EMPLOYEE AS E, EMPLOYEE AS S
WHERE       E.Super_ssn=S.Ssn;
```

```
SELECT      Ssn, Dname
FROM        EMPLOYEE, DEPARTMENT;
```

```
SELECT      *
FROM        EMPLOYEE
WHERE       Dno=5;
```

Sentencia SELECT (Ejemplos)



```
SELECT      E.Fname, E.Lname, S.Fname, S.Lname  
FROM        EMPLOYEE AS E, EMPLOYEE AS S  
WHERE       E.Super_ssn=S.Ssn;
```

La referencia a la tabla
cambian
consecuentemente

Ssn, Dname
EMPLOYEE, DEPARTMENT;

```
SELECT      *  
FROM        EMPLOYEE  
WHERE       Dno=5;
```

Sentencia SELECT (Ejemplos)

```
SELECT  E.Fname, E.Lname, S.Fname, S.Lname
FROM    EMPLOYEE AS E, EMPLOYEE AS S
WHERE   E.Super_ssn=S.Ssn;
```



```
SELECT  Ssn, Dname
FROM    EMPLOYEE, DEPARTMENT;
```

```
SELECT  *
FROM    EMPLOYEE
WHERE   Dno=5;
```

Sentencia SELECT (Ejemplos)

```
SELECT    E.Fname, E.Lname, S.Fname, S.Lname
FROM      EMPLOYEE AS E, EMPLOYEE AS S
WHERE     E.Super_ssn=S.Ssn;
```



```
SELECT    Ssn, Dname
FROM      EMPLOYEE, DEPARTMENT;
```

```
SELECT    *
FROM      EMPLOYEE
WHERE     Dno=5;
```

Producto Cartesiano

Sentencia SELECT (Ejemplos)

```
SELECT      E.Fname, E.Lname, S.Fname, S.Lname
FROM        EMPLOYEE AS E, EMPLOYEE AS S
WHERE       E.Super_ssn=S.Ssn;
```

```
SELECT      Ssn, Dname
FROM        EMPLOYEE, DEPARTMENT;
```



```
SELECT      *
FROM        EMPLOYEE
WHERE       Dno=5;
```

Sentencia SELECT (Ejemplos)

```
( SELECT      DISTINCT Pnumber
  FROM        PROJECT, DEPARTMENT, EMPLOYEE
  WHERE       Dnum=Dnumber AND Mgr_ssn=Ssn
              AND Lname='Smith' )

UNION

( SELECT      DISTINCT Pnumber
  FROM        PROJECT, WORKS_ON, EMPLOYEE
  WHERE       Pnumber=Pno AND Essn=Ssn
              AND Lname='Smith' );
```


Sentencia SELECT (Ejemplos)




```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   Address LIKE '%Houston,TX%';
```

```
SELECT  E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM    EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE   E.Ssn=W.Essn AND W.Pno=P.Pnumber AND
        P.Pname='ProductX';
```

```
SELECT  *
FROM    EMPLOYEE
WHERE   (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Sentencia SELECT (Ejemplos)

```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   Address LIKE '%Houston,TX%';
```




```
SELECT  E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM    EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE   E.Ssn=W.Essn AND W.Pno=P.Pnumber AND
        P.Pname='ProductX';
```

```
SELECT  *
FROM    EMPLOYEE
WHERE   (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Sentencia SELECT (Ejemplos)

SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston,TX%';

Se puede renombrar
columnas



SELECT E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE E.Ssn=W.Essn AND W.Pno=P.Pnumber AND
P.Pname='ProductX';

SELECT *
FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000) AND Dno = 5;

Sentencia SELECT (Ejemplos)

```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   Address LIKE '%Houston,TX%';
```

```
SELECT  E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM    EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE   E.Ssn=W.Essn AND W.Pno=P.Pnumber AND
        P.Pname='ProductX';
```



```
SELECT  *
FROM    EMPLOYEE
WHERE   (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Sentencia SELECT (Ejemplos)

```
SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT D, EMPLOYEE E, WORKS_ON W,
          PROJECT P
WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND
          W.Pno= P.Pnumber
ORDER BY  D.Dname, E.Lname, E.Fname;
```

Sentencia SELECT (Ejemplos)

```
SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT D, EMPLOYEE E, WORKS_ON W,
          PROJECT P
WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND
          W.Pno= P.Pnumber
ORDER BY  D.Dname, E.Lname, E.Fname;
```



Primero ordena por

Sentencia SELECT (Ejemplos)

```
SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT D, EMPLOYEE E, WORKS_ON W,
          PROJECT P
WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND
          W.Pno= P.Pnumber
ORDER BY  D.Dname, E.Lname, E.Fname;
```

... después por

Sentencia SELECT (Ejemplos)

```
SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT D, EMPLOYEE E, WORKS_ON W,
          PROJECT P
WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND
          W.Pno= P.Pnumber
ORDER BY  D.Dname, E.Lname, E.Fname;
```

... y después por

Sentencia SELECT (Ejemplos)

```
SELECT  D.Dname, E.Lname, E.Fname, P.Pname
FROM    DEPARTMENT D, EMPLOYEE E, WORKS_ON W,
        PROJECT P
WHERE   D.Dnumber= E.Dno AND E.Ssn= W.Essn AND
        W.Pno= P.Pnumber
ORDER BY D.Dname, E.Lname, E.Fname;
```

ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC

Sentencia INSERT

- El comando INSERT se utiliza para agregar una sola tupla a una relación.
- Hay varias formas de especificar el comando INSERT.

- Hay varias formas de especificar el comando INSERT:

```
INSERT INTO EMPLOYEE  
VALUES      ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98  
              Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

```
INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)  
VALUES      ('Richard', 'Marini', 4, '653298653');
```

Sentencia DELETE

DESCRIPCIÓN

- El comando DELETE elimina tuplas de una tabla.
- Un DELETE se aplica sobre una única tabla a la vez. Dependiendo de las acciones establecidas para la integridad referencial, se puede propagar a otras.
- La condición WHERE establece cuáles filas serán eliminadas. Si dicha condición no se establece, todas las filas serán eliminadas.

SINTAXIS

```
DELETE FROM      EMPLOYEE  
WHERE            Ssn='123456789';
```

```
DELETE FROM      EMPLOYEE;
```

Sentencia UPDATE

- El comando UPDATE se utiliza para modificar los valores de una o más tuplas de una tabla.
- La condición WHERE limita el ámbito de acción del UPDATE. Si no se establece el WHERE, se actualizan todas las tuplas.
- La cláusula SET establece cuáles atributos serán modificados.

```
UPDATE    PROJECT  
SET       Plocation = 'Bellaire', Dnum = 5  
WHERE     Pnumber=10;
```

Taller A

→ Cree una base de datos que refleje el estado de la página 72.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

SQL Básico

CE3101 - Bases de Datos