

# SQL Básico



# Pre-requisitos

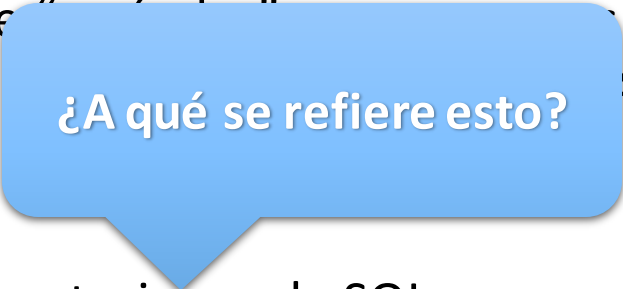
- Utilizaremos SQL Server Express para muchos ejemplos
  - Gratis
  - Muy completo
  - Tiene muy buenas herramientas de desarrollo
- Versión 2008 o 2012



# Introducción

- SQL se considera una de las razones del éxito de las base 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” de distintos fabricantes facilita la interoperabilidad (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 de SQL
  - PL/SQL
  - T-SQL
- Todas son muy similares

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 de SQL
  - PL/SQL
  - T-SQL
- Todas son muy similares

Ocurre muy a menudo en el mundo del software

# 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 es un inicialismo de Structured Query Language. Originalmente se llamaba SEQUEL (Structured English QUery Language).
- Diseñado e implementado en IBM Research
- Es un lenguaje comprensivo:
  - 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 es un inicialismo de Structured Query Language. Originalmente se llamaba SEQUEL (Structured English QUery Language).
- Diseñado e implementado en IBM Research
- Es un lenguaje con  Definición de datos
  - DDL
  - DML
  - Definición de vistas
  - Especificación de seguridad y autorización
  - Integridad referencial y control de transacciones

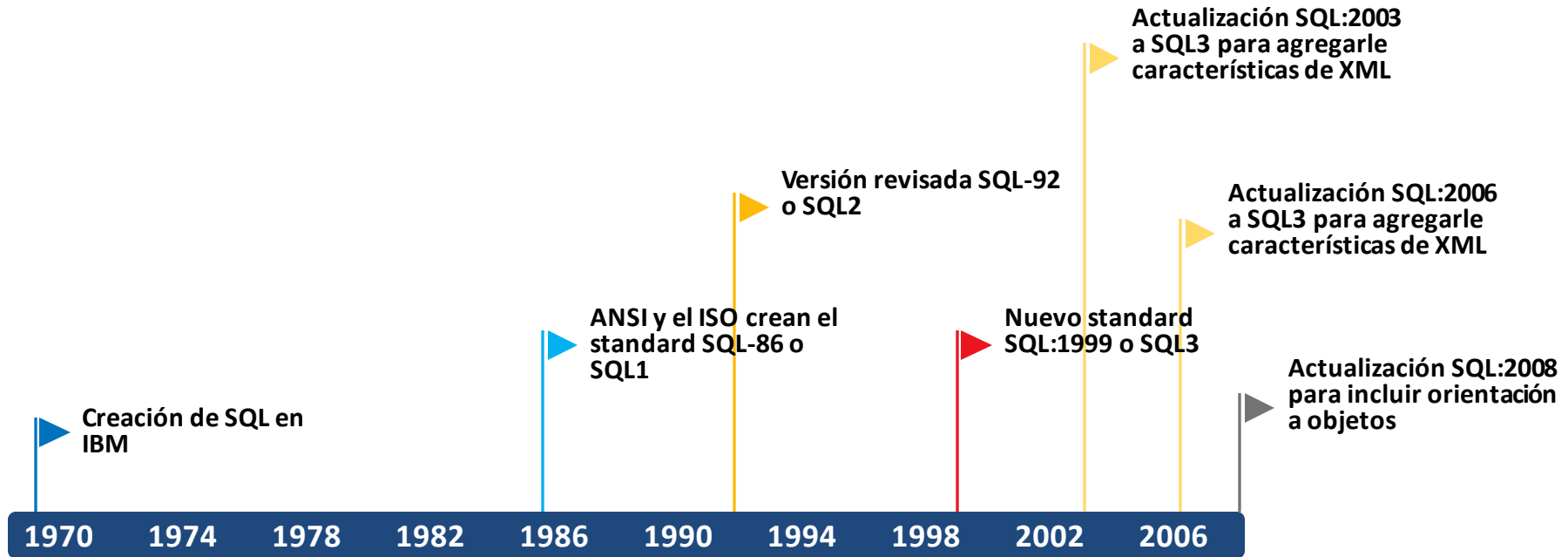


# El lenguaje SQL

- 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 es un inicialismo de Structured Query Language. Originalmente se llamaba SEQUEL (Structured English QUERY Language).
- Diseñado e implementado en IBM Research
- Es un lenguaje compuesto por:
  - DDL
  - DML
  - Definición de vistas
  - Especificación de seguridad y autorización
  - Integridad referencial y control de transacciones

Manipulación de datos

# 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

```
> 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

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

No todos los usuarios están autorizados a crear SCHEMAS, el DBA es el que gestión estos permisos

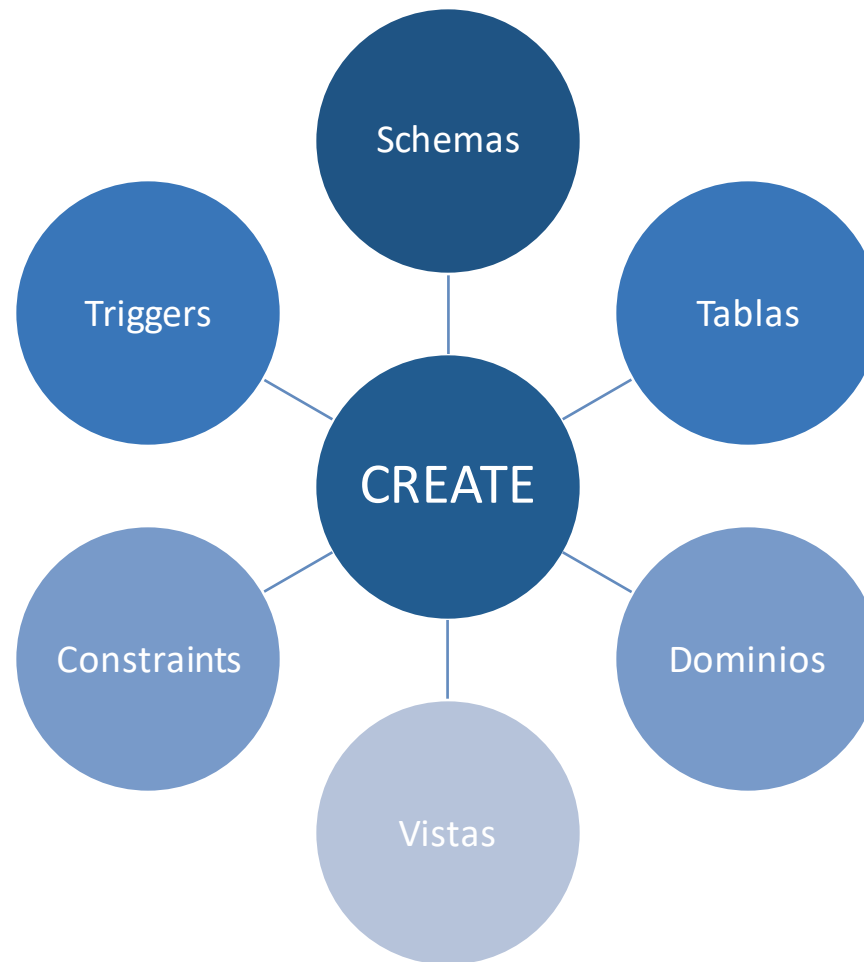
## ...Algunos conceptos en SQL

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

# Data Definition Language

DDL

# El comando CREATE



# El comando 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 table en SQL no es un conjunto de tuplas, es más bien un multi set o una bolsa de tuplas
- En una tabla SQL es possible que existan tuplas con valores iguales en todos sus atributos
- Si hay una llave establecida, la tabla será un conjunto de tuplas

# Ejemplos

## EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

## DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

## DEPT\_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

## PROJECT

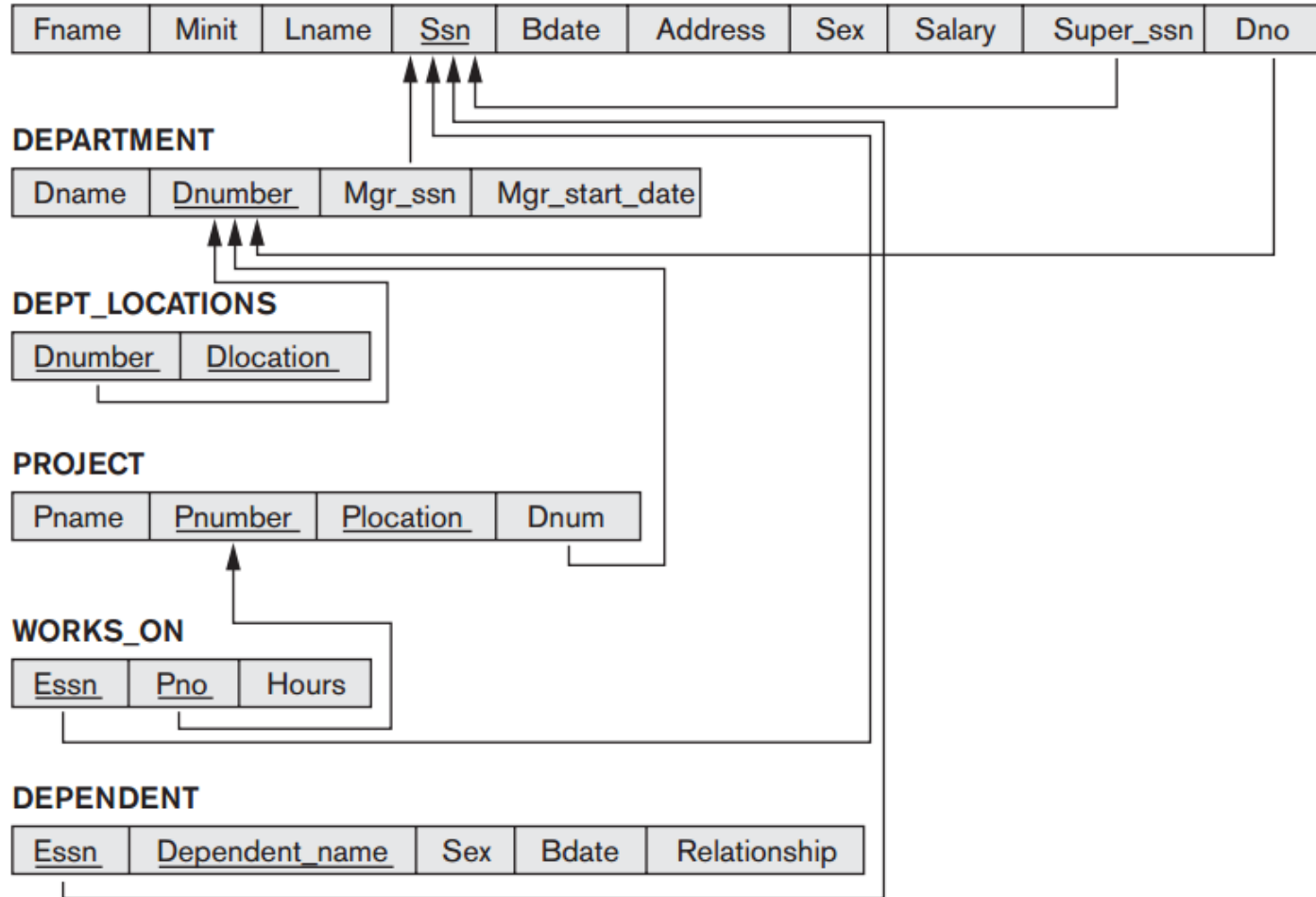
Pname	<u>Pnumber</u>	<u>Plocation</u>	Dnum
-------	----------------	------------------	------

## WORKS\_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

## DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	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)
);
```

# Ejemplos

-- Creación de la tabla EMPLOYEE

CREATE TABLE EMPLOYEE

( Fname VARCHAR(15) NOT NULL,  
Mname CHAR,  
Lname VARCHAR(15) NOT NULL,  
NOT NULL,

Nombre de la columna

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(1),
   Lname      VARCHAR(15)      NOT NULL,
   Ssn        CHAR(9),
   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

-- 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)  
);
```

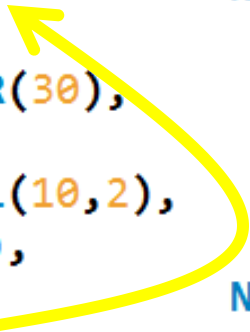
NOT NULL,

NOT NULL Constraint. Por defecto se permite NULL



# 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)
);
```

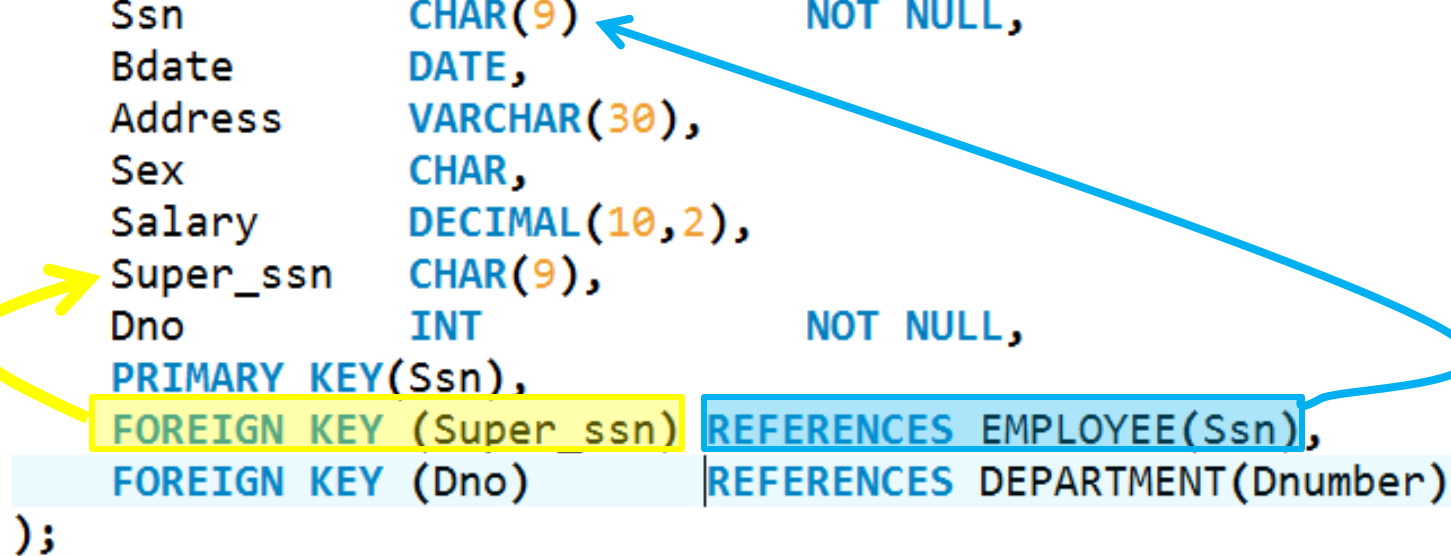


# Ejemplos

-- 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)
);
```



# 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 between columns in the EMPLOYEE table. A blue arrow points from the Ssn column to the REFERENCES EMPLOYEE(Ssn) part of the FOREIGN KEY (Super\_ssn) definition. A yellow arrow points from the Super\_ssn column to the FOREIGN KEY (Super\_ssn) definition. A light blue box highlights the FOREIGN KEY (Super\_ssn) REFERENCES EMPLOYEE(Ssn) line, and another light blue box highlights the FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) line.

# 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 posterior...

# 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);
```

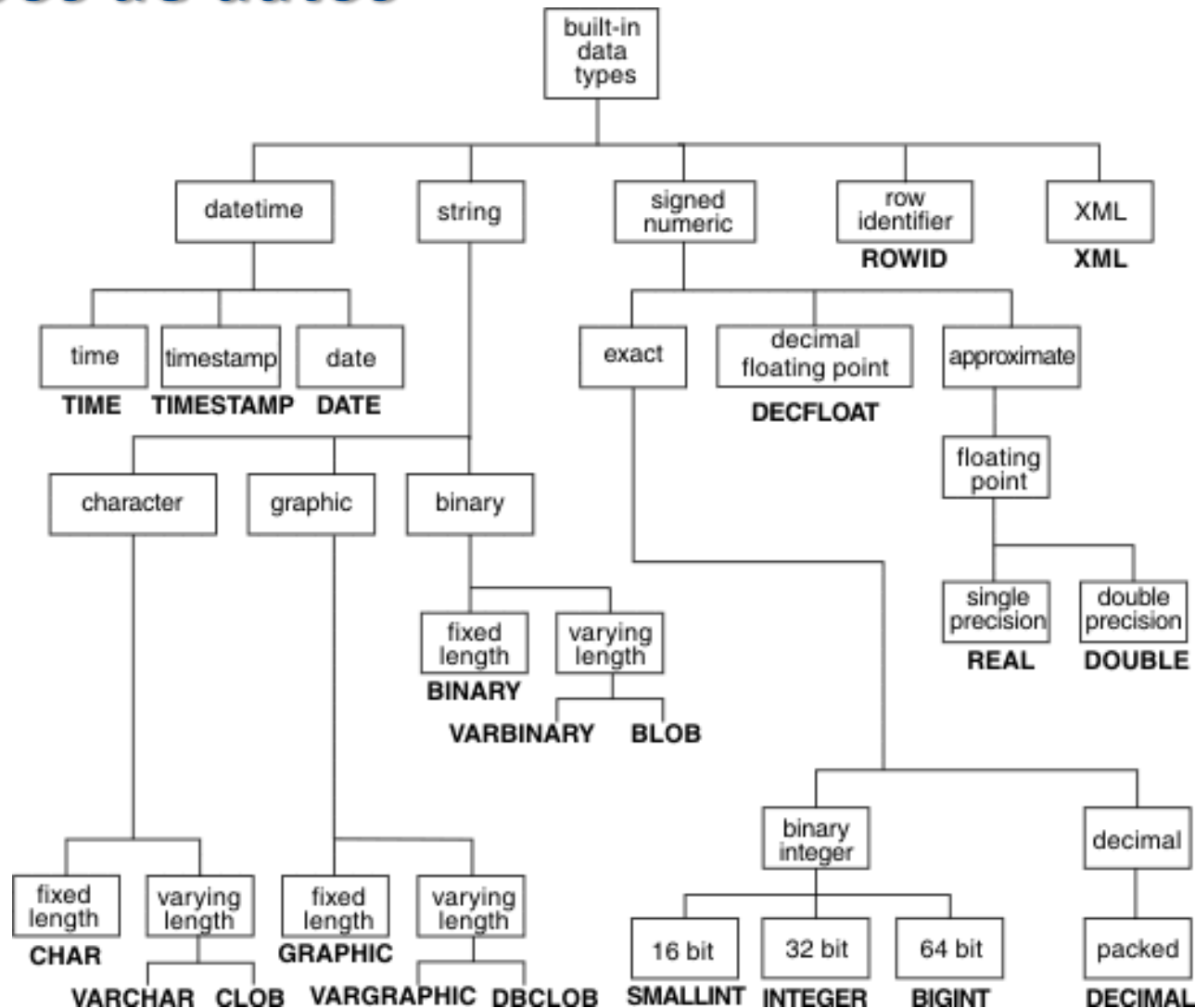
```
ALTER TABLE PROJECT  
ADD CONSTRAINT PROJECT_DEPARTMENT_FK FOREIGN KEY (Dnum)  
REFERENCES DEPARTMENT(Dnumber);
```

```
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);
```

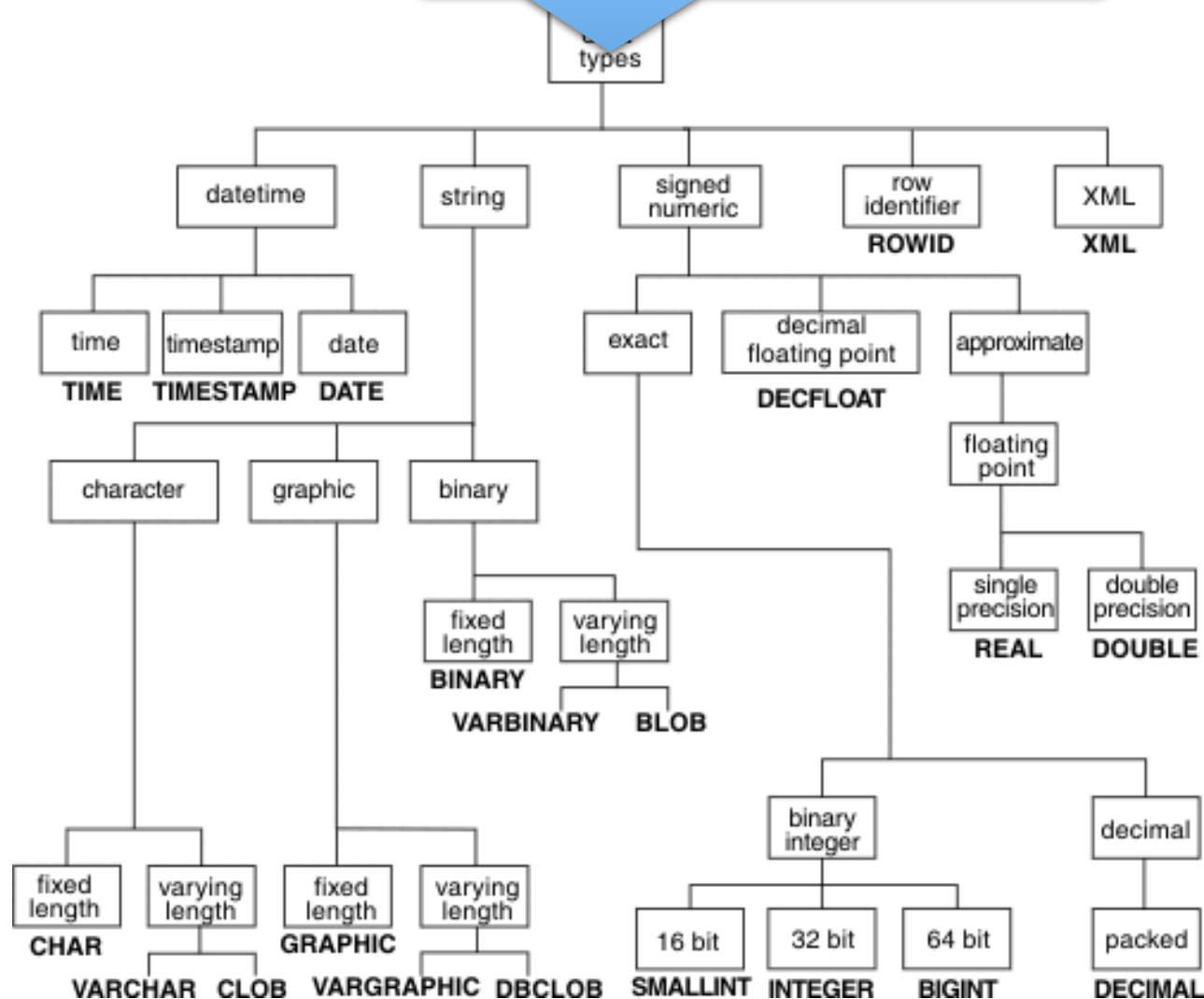
```
ALTER TABLE "DEPENDENT"  
ADD CONSTRAINT DEPEDENDT_EMPLOYEE_FK FOREIGN KEY (Essn)  
REFERENCES EMPLOYEE(Ssn);
```

# Tipos de datos

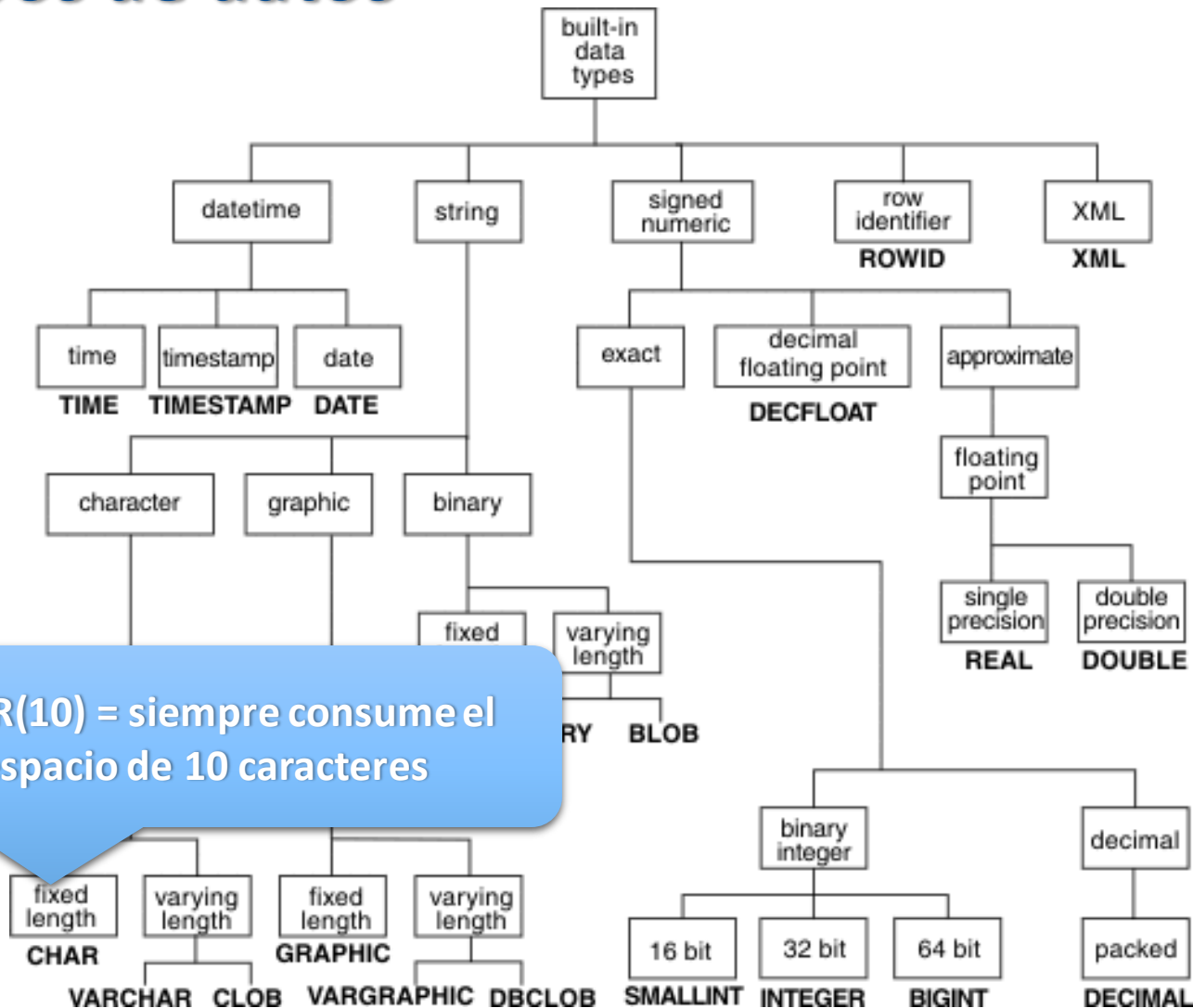


# Tipos de datos

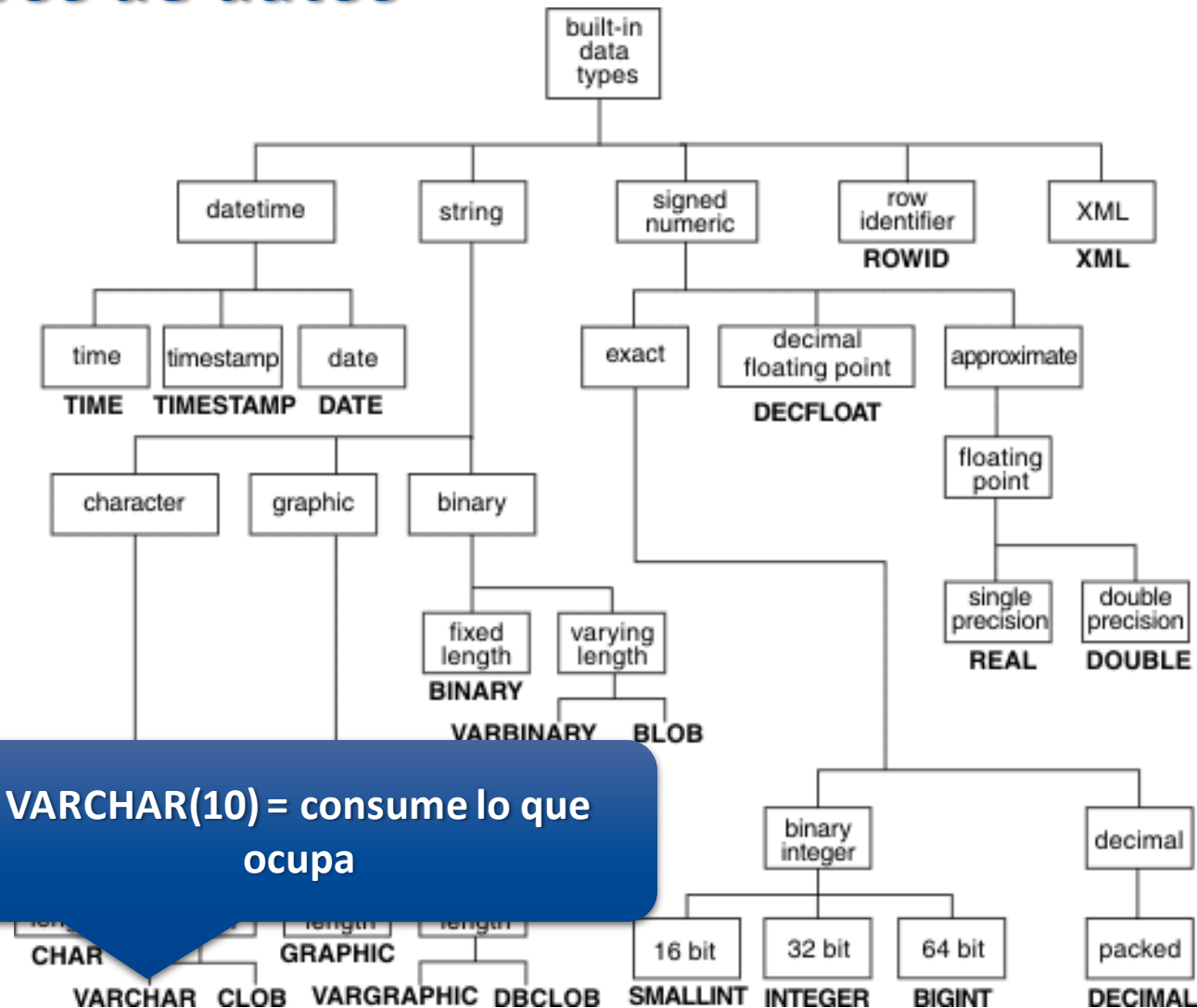
Escoger los tipos cuidadosamente



# Tipos de datos



# Tipos de datos



# ¿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)
```

- 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)
);
```

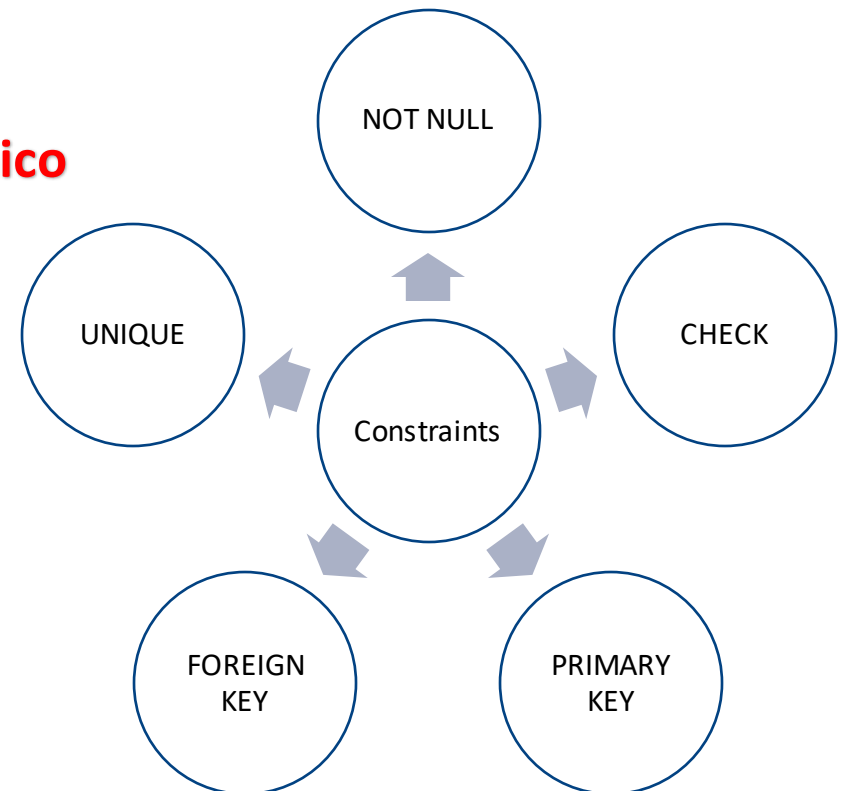
# ¿Cómo se asigna un dominio en SQL?

```
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 tipos de 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 un solo 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 EMPPK
        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

# La sentencia SELECT

- Sentencia para recuperar datos de la base de datos
- No es igual que el operador SELECT ( $\sigma$ ) del álgebra relacional. Combina varias operaciones relacionales
- El bloque SELECT básico es:

<b>SELECT</b>	<attribute list>
<b>FROM</b>	<table list>
<b>WHERE</b>	<condition>;

# La sentencia SELECT

- Sentencia para recuperar datos de la base de datos
- No es igual que el operador SELECT ( $\sigma$ ) del álgebra relacional. Combina varias operaciones relacionales
- El bloque SELECT básico es:

Lista de atributos que se desean obtener. PROJECT

**SELECT**  
**FROM**  
**WHERE**

<attribute list>  
<table list>  
<condition>;



# La sentencia SELECT

- Sentencia para recuperar datos de la base de datos
- No es igual que el operador SELECT ( $\sigma$ ) del álgebra relacional. Combina varias operaciones relacionales
- El bloque SELECT básico es:

**SELECT**  
**FROM**  
**WHERE**

Puede ser \*

<attribute list>  
<table list>  
<condition>;

# La sentencia SELECT

- Sentencia para recuperar datos de la base de datos
- No es igual que el operador SELECT ( $\sigma$ ) del álgebra relacional. Combina varias operaciones relacionales
- El bloque SELECT básico es:

**SELECT**  
**FROM**  
**WHERE**

Puede ser \*

Debe evitarse

<attribute list>  
<table list>  
<condition>;

# La sentencia SELECT

- Sentencia para recuperar datos de la base de datos
- No es igual que el operador SELECT ( $\sigma$ ) del álgebra relacional. Combina varias operaciones relacionales
- El bloque SELECT básico es:

**SELECT**  
**FROM**  
**WHERE**

Condición de selección

<table list>  
<condition>;


# La sentencia SELECT

- Sentencia para recuperar datos de la base de datos
- No es igual que el operador SELECT ( $\sigma$ ) del álgebra relacional. Combina varias operaciones relacionales
- El bloque SELECT básico es:

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

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

# La sentencia **SELECT**



```
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;
```

# La sentencia **SELECT**

```
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;
```

# La sentencia **SELECT**

```
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;
```

# La sentencia **SELECT**

```
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;
```



# La sentencia SELECT

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


Previene la ambigüedad

# La sentencia SELECT

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

**'Calificar' la columna**

# La sentencia SELECT




```
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;
```

# La sentencia SELECT




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

Renombra la tabla

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

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

# La sentencia SELECT




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

La referencias a la tabla  
cambian  
consecuentemente

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

# La sentencia SELECT

```
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
WHERE     EMPLOYEE
          Dno=5;
```

**Producto cartesiano**

# La sentencia SELECT

```
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;
```

# La sentencia SELECT

```
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;
```



# La sentencia **SELECT**




```
( 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' );
```

# La sentencia SELECT




```
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;
```

# La sentencia **SELECT**

```
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;
```

# La sentencia SELECT

```
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
P.Pname='ProductX';
```

Se puede renombrar  
columnas

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


# La sentencia SELECT

```
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;
```

# La sentencia **SELECT**



```
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;
```

# La sentencia SELECT




```
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

# La sentencia SELECT




```
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




# La sentencia SELECT



```
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

# La sentencia **SELECT**



```
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**

# INSERT, DELETE y UPDATE

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



**INSERT INTO  
VALUES**

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


**INSERT INTO  
VALUES**

**EMPLOYEE (Fname, Lname, Dno, Ssn)**  
( 'Richard', 'Marini', 4, '653298653' );

# INSERT, DELETE y UPDATE

- El comando INSERT se utiliza para agregar una sola tupla a una relación
- 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');
```

# INSERT, DELETE y UPDATE

- 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 cuales filas serán eliminadas. Si dicha condición no se establece, todas las filas serán eliminadas.

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

```
DELETE FROM      EMPLOYEE;
```

# INSERT, DELETE y 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 cuales atributos serán modificados

<b>UPDATE</b>	PROJECT
<b>SET</b>	Plocation = 'Bellaire', Dnum = 5
<b>WHERE</b>	Pnumber=10;

# SQL Básico

