

Tema 4.

SQL

Tema 4. SQL

El estándar de las BD relacionales
Elmasri/Navathe 02

- Definición de datos (CREATE, ALTER, DROP) (8.1)
- Actualización (INSERT, DELETE, UPDATE) (8.4)
- Consultas (SELECT *) (8.2, 8.3)
- SQL Avanzado (entorno de programación y funciones incorporadas)

* Este curso no cubre todos los aspectos de SELECT.

No se estudian SELECT ANIDADOS (salvo IN), EXCEPT ni INTERSECT.

SQL

(Structured Query Language)

- Álgebra relacional → Orden de las operaciones
- SQL → Lenguaje declarativo → Optimización
- Evolución del estándar SQL:
 - SQL1 (1986), SQL2 ó SQL-92 (1992), SQL3 (1999),..., SQL:2011.
- SQL:
 - Definición (LDD).
 - Consulta y Actualización (LMD).
 - Vistas.
 - Funciones, procedimientos y triggers.
 - Inclusión en lenguajes (Java, VB, C, PASCAL,...).
- SGBD comerciales → Usan variantes de SQL. Nos centramos en SQL2.

SQL	Mod. Relacional
Tabla	Relación
Fila	Tupla
Columna	Atributo

SQL - LDD

(Structured Query Language – Data Definition Language)

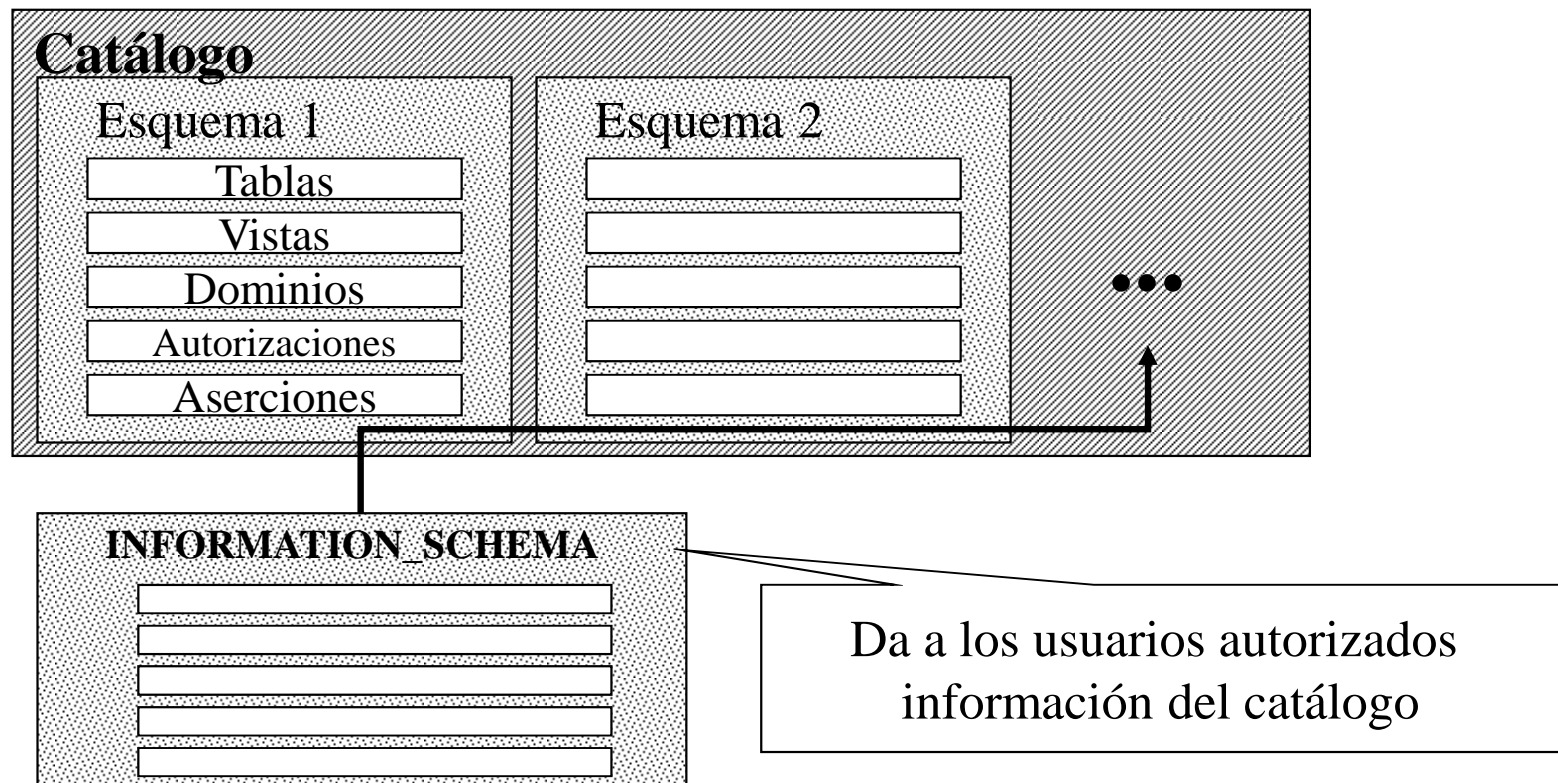
- Parte LDD de SQL: Lenguaje de Definición de Datos.
- Crea (CREATE), modifica la estructura (ALTER) o borra elementos de la base de datos (DROP).
- Elementos: la propia base de datos o tablas.
- Instrucciones básicas:
 - Crear el elemento
 - CREATE DATABASE ó CREATE SCHEMA
 - CREATE TABLE
 - Modificar estructura
 - ALTER TABLE
 - Borrar el elemento
 - DROP DATABASE ó DROP SCHEMA
 - DROP TABLE
- **Sintaxis de MySQL** correspondiente a estas instrucciones: **Práctica P5**

Crear base de datos en SQL: esquema y catálogo

- El término esquema de BD se incorpora en SQL2. Para crear el esquema de BD:

CREATE DATABASE Nombre; O bien **CREATE SCHEMA** Nombre;

- RI: Sólo entre relaciones del mismo catálogo.
- Esquemas de un catálogo: Pueden compartir elementos. (Ej. Dominios).



Crear tablas en SQL: Sintaxis de CREATE TABLE

La sintaxis que permite crear una tabla nueva en la BD es:

CREATE TABLE nombreDeLaTabla

(nombreCampo1Tabla tipoDatoCampo1 opcionesCampo1,
 nombreCampo2Tabla tipoDatoCampo2 opcionesCampo2,
 ...
 nombreCampoNTabla tipoDatoCampoN opcionesCampoN,

COLUMNAS O CAMPOS
DE LA TABLA

PRIMARY KEY(campo/sDeLaClavePrimaria),
UNIQUE(campo/sDeLaClaveAlternativa),
FOREIGN KEY(campo/sDeLaClaveExterna)
REFERENCES tablaRef(campo/sDeLaTablaRef)
ON UPDATE opcionAlActualizar
ON DELETE opcionAlBorrar);

RESTRICCIONES DE
INTEGRIDAD
DE LA TABLA

- En toda tabla bien diseñada debe haber al menos dos campos o columnas, obligatoriamente una sola clave primaria (PRIMARY KEY), de ninguna a varias claves alternativas (UNIQUE) y de ninguna a varias claves externas (FOREIGN KEY).

Crear tablas en SQL(2): Ejemplo

EJEMPLO: Creación de la tabla DEPARTAMENTO de la BD EMPRESA

- Por cada columna o campo de la tabla se debe indicar obligatoriamente: un *nombre* para el campo, y el *tipo de datos* que contendrá

CREATE TABLE DEPARTAMENTO

(

NOMBRED

VARCHAR(15)

NOT NULL,

Otras opciones para el campo

NUMEROD

INT

NOT NULL,

NSS_JEFE **CHAR(9),**

FECHA_INIC_JEFE **DATE,**

PRIMARY KEY(NUMEROD),

UNIQUE(NOMBRED),

FOREIGN KEY(NSS_JEFE) REFERENCES EMPLEADO(NSS)

ON UPDATE CASCADE

ON DELETE SET NULL

);

Crear tablas en SQL(3): Ejemplo

Se pueden poner *nombres a las distintas restricciones* y especificar distintas opciones para las columnas de la tabla (como *impedir valores nulos* e indicar *valores por defecto*).

```
CREATE TABLE DEPARTAMENTO
(
  ...,
  NSS_JEFE CHAR(9) NOT NULL DEFAULT '888665555',
  ...,
  CONSTRAINT CLPDEPTO PRIMARY KEY(NUMEROD),
  CONSTRAINT CLSDEPTO UNIQUE(NOMBRED),
  CONSTRAINT CLEXTDEPTO FOREIGN KEY (NSS_JEFE)
    REFERENCES EMPLEADO(NSS)
    ON UPDATE CASCADE
    ON DELETE SET DEFAULT
);
```

- La asignación de nombre a una restricción (CONSTRAINT) es opcional.
- El nombre dado a una restricción sirve para poderla sustituir o desechar.

Crear tablas en SQL(4): Tipos de datos

Tipos de datos NUMÉRICOS

- Enteros:
INTEGER / INT
SMALLINT
- Reales:
DECIMAL(i,j) / DEC(i,j)
NUMERIC(i,j)
 - Donde **i** es la cantidad total de dígitos y **j** la cantidad de dígitos decimales.
 - Por defecto: **i** según la implementación, **j**=0.**DOUBLE PRECISION**
FLOAT
REAL
- Ejemplos:
Código INT
Precio DECIMAL(7,2)

Crear tablas en SQL(5): Tipos de datos

Tipos de datos CADENA DE CARACTERES

- Longitud fija:
CHAR(n) ó **CHARACTER(n)**
- Longitud variable:
VARCHAR(n), **CHARACTER VARYING(n)** ó **CHAR VARYING(n)**
 - donde **n**=máximo definido. Por defecto **n**=1.

Representación nacional:

- Longitud fija:
NATIONAL CHARACTER(n), **NCHAR(n)** ó **NATIONAL CHAR(n)**
- Longitud variable:
NATIONAL CHARACTER VARYING(n), **NCHAR VARYING(n)** ó **NATIONAL CHAR VARYING(n)**
 - donde **n**=máximo definido. Por defecto **n**=1.
- Ejemplos:
 - Descripción **CHARACTER(20)**
 - Nombre **VARCHAR(12)**

Crear tablas en SQL(6): Tipos de datos

Tipos de datos CADENA DE BITS

- Longitud fija:
BIT(n) por defecto **n=1**.
- Longitud variable:
BIT VARYING(n) **n**=longitud máxima. Por defecto **n=1**.
- Ejemplos:
 - SiNo BIT UnFlag BIT(100)
 - OtroFlag BIT VARYING (125)

Crear tablas en SQL(7): Tipos de datos

Tipos de datos FECHA Y HORA

- Sólo fecha: **DATE**
 - Formado por Año(Year), Mes y Día. Generalmente YYYY-MM-DD.
- Sólo hora: **TIME** / **TIME(i)**
 - Formado por Hora, Minutos y Segundos. Normalmente HH:MM:SS.
 - **i** = posiciones de fracciones de segundo.
- Desplazamiento de huso horario: **TIME WITH TIME ZONE**
 - De -12:59 a +13:00. Por defecto: Uso local de la sesión.
- Marca de tiempo: **TIMESTAMP** [**WITH TIME ZONE**]
 - DATE, TIME y mínimo 6 posiciones de fracciones de segundo.
- Valor relativo (de tiempo): **INTERVAL**
 - Para incrementar/decrementar un valor fecha, hora o marca de tiempo.
 - AñosAMeses INTERVAL YEAR TO MONTH.

↑ ↑

Unidad mayor Unidad menor

Crear tablas en SQL(8): Tipos de datos

Tipos de datos personalizados: DOMINIOS

- Un dominio permite crear un nuevo tipo de datos a partir de los que ya están definidos en el estándar SQL.
- Instrucción básica para crear un nuevo dominio:

```
CREATE DOMAIN nombreNuevoTipoDato AS tipoDatoSQL;
```

- Ejemplo:

```
CREATE DOMAIN TIPO_NSS AS CHAR(9).
```

Luego se puede usar en CREATE TABLE como un tipo más. Por ejemplo:

```
NSS TIPO_NSS
```

```
NSS_JEFE TIPO_NSS
```

Crear tablas en SQL(9): Opciones para cada columna

El estándar SQL permite especificar algunas opciones dentro de cada campo de la tabla. Por ejemplo (existen más):

- **Se puede indicar un valor por defecto para ese campo de la tabla con DEFAULT**
 - Los SGBDs suelen tener un valor por defecto para cada tipo de dato.
 - DEFAULT valorPorDefecto permite personalizar ese valor.
 - Ejemplo: `NSS_JEFE CHAR(9) DEFAULT '888665555'`
- **Se puede indicar que un campo de la tabla nunca podrá estar vacío con NOT NULL**
 - Restricción.
 - Esta restricción la han de cumplir obligatoriamente todos los campos que formen parte de la clave primaria.
 - Ejemplo: `NUMEROD INT NOT NULL`

Crear tablas en SQL(10): Restricciones de integridad

- Las restricciones de integridad de la tabla se indican en **CREATE TABLE** detrás de la descripción de los atributos (columnas) de la tabla.
- **Claves:**
 - **PRIMARY KEY:** Clave primaria.
 - **UNIQUE:** Clave candidata.
 - **FOREIGN KEY:** Clave extranjera.
- **Integridad referencial** (para cada clave extranjera):
 - Se especifica lo que ocurrirá automáticamente al borrar y/o al modificar con:
 - **ON DELETE** **(al borrar)**
 - **ON UPDATE** **(al modificar)**
 - Opciones para cada una de ellas:
 - **SET NULL** **(dejar el campo vacío)**
 - **CASCADE** **(hacerlo en cascada, propagar)**
 - **SET DEFAULT** **(poner el valor por defecto de ese campo)**

Si no se pone no dejará
borrar y/o modificar
mientras haya filas relacionadas

Crear tablas en SQL(11): Ejemplo de FOREIGN KEY

- **CASCADE** es adecuado para relaciones de:
 - Vínculo (TRABAJA_EN).
 - Atributos multivaluados (LOCALIZACIONES_DEPT).
 - T. de entidad débiles (DEPENDIENTE).
- **Ejemplo de FOREIGN KEY:**
Codificación de la cláusula para la clave externa NSS_SUPERV en la creación de la tabla EMPLEADO de la BD EMPRESA

```
CREATE TABLE EMPLEADO ...  
CONSTRAINT CLESUPEREMP FOREIGN KEY (NSS_SUPERV) REFERENCES  
EMPLEADO(NSS) ON DELETE SET NULL ON UPDATE CASCADE;
```

Al borrar un empleado poner NULL en todos los empleados que lo tengan como jefe

Si se modifica el NSS, propagar el nuevo valor a todos los empleados que lo tienen como jefe

Modificar el diseño de una tabla: ALTER TABLE

- **ALTER TABLE** permite cambiar el diseño de la tabla que se indique, tanto el diseño de sus atributos como de sus restricciones de integridad.
- Su sintaxis es:
 - **ALTER TABLE** nombreTabla
 - Seguido de una de tres opciones:
 - **ALTER** para cambiar algo que ya existe.
 - **ADD** para añadir algo nuevo al diseño.
 - **DROP** para quitar algo.

NombreDeLaBD.Nombretabla
Si solo trabajamos con esa BD
no hace falta ponerla
(solo el nombreTabla como antes)

- **Modificar diseño de la columna**

```
ALTER TABLE EMPRESA.DEPARTAMENTO ALTER NSS_JEFE  
DROP DEFAULT;
```

```
ALTER TABLE EMPRESA.DEPARTAMENTO ALTER NSS_JEFE  
SET DEFAULT '333445555';
```

Modificar el diseño de una tabla (2): ALTER TABLE...ADD

- Para añadir algo a una tabla ya creada, después de ADD se pone lo mismo que se indicaría en la instrucción CREATE TABLE.
- **Añadir columna**

```
ALTER TABLE EMPRESA.EMPLEADO ADD PUESTO VARCHAR(12);
```

- **Añadir restricción**

```
ALTER TABLE EMPRESA.EMPLEADO ADD  
CONSTRAINT CLEXTSUPERVISOR  
FOREIGN KEY (NSS_SUPERV) REFERENCES EMPLEADO(NSS)  
ON DELETE SET NULL  
ON UPDATE CASCADE;
```

Modificar diseño de una tabla (3): ALTER TABLE...DROP

- **Quitar columna**

```
ALTER TABLE EMPRESA.EMPLEADO DROP DIRECCION CASCADE;
```

- CASCADE borra restricciones y vistas que le hagan referencia.
- RESTRICT sólo se borra si no hay restricciones ni vistas que le hagan referencia.

- **Quitar restricción**

```
ALTER TABLE EMPRESA.EMPLEADO DROP PRIMARY KEY
```

```
ALTER TABLE EMPRESA.EMPLEADO DROP FOREIGN KEY  
CLEXTSUPERVISOR;
```

- En claves externas es preciso haber dado un nombre a la restricción a borrar.

Borrar base de datos y/o tablas en SQL: DROP

- Borrar base de datos

DROP DATABASE EMPRESA RESTRICT;

- **RESTRICT:** Borrar sólo si la BD no contiene elementos
- **CASCADE:** Borrar esquema y contenidos

– Para un esquema, lo mismo pero con SCHEMA en vez de DATABASE

- Borrar tabla

DROP TABLE DEPENDIENTE CASCADE;

- **RESTRICT:** Borrar sólo si no existen referencias a la tabla en clave externa de otra relación o en una vista.
- **CASCADE:** Borrar tabla y todas las restricciones y vistas donde haya referencias a ésta.

SQL - LMD

(Structured Query Language – Data Manipulation Language)

- Parte LMD de SQL: Lenguaje de Manipulación de Datos.
- Trabaja sobre los datos que hay en las tablas de la base de datos.
- Permite insertar (INSERT) , modificar (UPDATE) y borrar (DELETE) filas o simplemente consultar los datos sin cambiar los datos que haya (SELECT).
- Instrucciones básicas:
 - Insertar nuevas filas en una tabla
 - INSERT INTO
 - Modificar los datos de una o varias filas ya existentes
 - UPDATE
 - Borrar filas enteras de la tabla
 - DELETE
 - Consultar datos sin cambiarlos
 - SELECT
- **Sintaxis de MySQL** correspondiente a estas instrucciones: **Prácticas P5 y P7**

Insertar datos en una tabla: INSERT INTO

A1: INSERT INTO EMPLEADO

VALUES ('Ricardo', 'C', 'Martínez', '653298653', '1952-12-30', 'Olmo 98, Cedros, MX', 'H', 37000, '987654321', 4)

- Mismo orden en el que se especificaron los atributos al crear la tabla.

A1A: INSERT INTO EMPLEADO(NOMBRE, APELLIDO, ND, NSS)

VALUES ('Ricardo', 'Martínez', 4, '653298653')

- Así los atributos con valor NULL o DEFAULT se pueden omitir.
- Mismo orden en el que se especifican los atributos en la propia INSERT.

A2:INSERT INTO EMPLEADO (NOMBRE, APELLIDO ,NSS, ND)

VALUES ('Roberto', 'Huerta', '980760540', 2)

A2A: INSERT INTO EMPLEADO (NOMBRE, APELLIDO, ND)

VALUES ('Roberto', 'Huerta', 2)

- Rechazada por no proporcionar valor para NSS (clave primaria: NOT NULL).

Insertar datos en una tabla: INSERT INTO (2)

- INSERT INTO también permite insertar con una sola instrucción las tuplas que se obtienen como resultado de una consulta a la base de datos.

A3A: CREATE TEMPORARY TABLE INFO_DEPTOS (
 NOMBRE_DEPTO VARCHAR(15),
 NUM_DE_EMPS INTEGER,
 SAL_TOTAL INTEGER);

- Utilidad: Tabla temporal donde almacenar consultas.
 - Sus datos pueden *perder actualidad*.
 - Alternativa sin este problema: Vista.

*Consulta sobre la
base de datos*

A3B: INSERT INTO INFO_DEPTOS



```
SELECT NOMBRED, COUNT (*), SUM(SALARIO)  
FROM  DEPARTAMENTO INNER JOIN EMPLEADO ON NUMEROD=ND  
GROUP BY NOMBRED;
```

Modificar datos de una tabla: UPDATE

A4: UPDATE PROYECTO

SET LOCALIZACIONP='Bellaire', NUMD=5

WHERE NUMEROP=10;

- Una sola tabla.
- WHERE: Selección de tuplas a modificar.
- SET: Atributos a modificar y nuevos valores. El nuevo valor puede ser NULL o DEFAULT.
- Modificaciones de clave primaria pueden propagarse a clave/s extranjera/s (si se definió ON UPDATE CASCADE al crearlas).

A5: UPDATE EMPLEADO

SET SALARIO=SALARIO*1.1

WHERE ND = 5 ;

- En SALARIO=SALARIO*1.1 , el SALARIO de la izquierda se refiere al nuevo valor y el SALARIO de la derecha al valor antiguo.

Borrar datos de una tabla: DELETE

A6A: DELETE FROM EMPLEADO WHERE APELLIDO='Brown';

- Una sola tabla.
- WHERE: selección de tuplas a eliminar.
- El borrado se puede propagar (RI referencial).

A6B: DELETE FROM EMPLEADO WHERE NSS='123456789';

**A6C: DELETE FROM EMPLEADO
WHERE ND = 5 ;**

A6D: DELETE FROM EMPLEADO;

- Sin WHERE se borran todas las tuplas.
- DELETE borra los datos.
- DROP eliminaría la definición de la tabla, es decir, la propia tabla.

Consultar los datos de la base de datos: SELECT

- La instrucción SELECT permite recuperar datos almacenados en una o varias tablas de la base de datos.
- La sintaxis de una instrucción SELECT está formada por varias **cláusulas**:
 - SELECT datos a recuperar
 - FROM tablas
 - WHERE condiciones
 - GROUP BY atributos
 - HAVING condiciones sobre resultados asociados a group by
 - ORDER BY atributos de ordenación de los datos a recuperar
 - Solamente es obligatoria la cláusula SELECT. El uso o no de lo demás depende de la consulta a realizar.
 - Además, las cláusulas que se usen **deben indicarse en el orden descrito**.
- El **resultado** de una instrucción SELECT es **siempre otra tabla**.
- La instrucción SELECT se basa en el álgebra relacional, cuyas operaciones básicas se enuncian en las siguientes diapositivas.

Álgebra relacional: Operaciones

- El álgebra relacional consta de un conjunto de operaciones para manipular **relaciones (tablas) enteras**. El resultado de cada una de estas operaciones es otra relación.
- Clasificación:
 - **Operaciones básicas**
 - De teoría de conjuntos
 - UNIÓN, INTERSECCIÓN, DIFERENCIA.
 - PRODUCTO CARTESIANO.
 - Específicas para BD relacionales:
 - PROYECCIÓN.
 - SELECCIÓN.
 - REUNIÓN (JOIN)
 - » INTERNA, EXTERNA, NATURAL
 - **Otras**
 - Ordenación y eliminación de duplicados
 - Funciones agregadas
 - CUENTA, SUMA, PROMEDIO, MÁXIMO y MÍNIMO.

Álgebra relacional (2): Unión, intersección y diferencia

- Operaciones binarias: Intervienen dos relaciones R y S.
- Su resultado es otra relación.
- Las 2 relaciones han de tener igual tipo de tuplas: Compatibilidad de unión.

$R \cup S$

(En SQL: **UNION**)

La Unión entre dos relaciones R y S nos da todas las tuplas tanto de R como de S, eliminando las que están repetidas.

$R \cap S$

(En SQL: **INTERSECT**)

La Intersección entre dos relaciones R y S nos da todas las tuplas comunes a R y S, eliminando las que están repetidas.

$R - S$

(En SQL: **EXCEPT**)

La Diferencia entre dos relaciones R y S nos da todas las tuplas de R que no están en S.

Álgebra relacional (3): Ejemplos de unión, intersección y diferencia

ALUMNO	
NOM	APEL
Susana	Yáñez
Ramesh	Sánchez
Josué	Landa
Bárbara	Jaimes
Amanda	Flores
Jaime	Vélez
Ernesto	Gómez

PROFESOR	
NOM	APEL
John	Smith
Ricardo	Bueno
Susana	Yáñez
Francisco	Jiménez
Ramesh	Sánchez

Fig. 7.11 (a)
Dos relaciones
compatibles
con la unión.

NOM	APEL
Susana	Yáñez
Ramesh	Sánchez

Fig 7.11. (b)
 $\text{ALUMNO} \cap \text{PROFESOR}$

NOM	APEL
Susana	Yáñez
Ramesh	Sánchez
Josué	Landa
Bárbara	Jaimes
Amanda	Flores
Jaime	Vélez
Ernesto	Gómez
John	Smith
Ricardo	Bueno
Francisco	Jiménez

Fig 7.11. (c)
 $\text{ALUMNO} \cup \text{PROFESOR}$

NOM	APEL
Josué	Landa
Bárbara	Jaimes
Amanda	Flores
Jaime	Vélez
Ernesto	Gómez

Fig. 7.11. (d)
 $\text{ALUMNO} - \text{PROFESOR}$

NOM	APEL
John	Smith
Ricardo	Bueno
Francisco	Jiménez

Fig. 7.11. (e)
 $\text{PROFESOR} - \text{ALUMNO}$

Álgebra relacional (4): PROYECCIÓN

PI

$\pi_{\langle \text{lista de atributos} \rangle}(\mathbf{R})$ (En SQL: cláusula **SELECT**)

lista de atributos: Atributos de R separados por comas

- Operación unaria: interviene una sola relación R.
- Selecciona algunas **columnas** de una relación R.
- El resultado es otra relación que tiene sólo los atributos especificados:
 - Atributos en el mismo orden especificado en la lista.
 - Eliminación de duplicados, es decir, de los resultados repetidos.
- En SQL:
 - No se eliminan los duplicados (operación muy costosa computacionalmente, puesto que hay que ordenar las filas y después borrar lo repetido) si no se indica explícitamente con **DISTINCT** en la cláusula **SELECT**.

Álgebra relacional (5): Ejemplos de proyección

$\pi_{\text{APELLIDO, NOMBRE, SEXO, SALARIO}}(\text{EMPLEADO})$

En SQL:

SELECT APELLIDO, NOMBRE, SEXO, SALARIO FROM EMPLEADO

$\pi_{\text{SEXO, SALARIO}}(\text{EMPLEADO})$

En SQL:

*Eliminación de
duplicados*

SELECT DISTINCT SEXO, SALARIO FROM EMPLEADO

APELLIDO	NOMBRE	SEXO	SALARIO
Smith	John	H	30.000
Wong	Franklin	H	40.000
Zelaya	Alicia	M	25.000
Wallace	Jennifer	M	43.000
Narayan	Ramesh	H	38.000
English	Joyce	M	25.000
Jabbar	Ahmad	H	25.000
Borg	Jaime	H	55.000

SEXO	SALARIO
H	30.000
H	40.000
M	25.000
M	43.000
H	38.000
H	25.000
H	55.000

$\pi_{\text{APELLIDO, NOMBRE, SEXO, SALARIO}}(\text{EMPLEADO})$

$\pi_{\text{SEXO, SALARIO}}(\text{EMPLEADO})$

Álgebra relacional (6): SELECCIÓN

SIGMA

$\sigma_{\langle \text{condición de selección} \rangle}(\mathbf{R})$ (En SQL: cláusula **WHERE**)

- Operación unaria: interviene una sola relación R.
- Selecciona un subconjunto de **tuplas (filas)** de una relación R. Las que satisfacen la condición de selección.
- Importante sobre esta operación:
 - Comprueba que la condición se cumpla en cada tupla de R.
 - *Implica que **todos los datos que se comparan estén en la misma fila de R.***
- El resultado es otra relación que tiene los mismos atributos que R.
- En SQL:
 - Especificación de condiciones, que pueden incluir distintos operadores como los de comparación {=, <, <=, >, >=, < >} o los lógicos **AND, OR, NOT**.

Álgebra relacional (7): Ejemplos de selección

$\sigma_{(ND=4 \text{ Y SALARIO}>25000) \text{ O } (ND=5 \text{ Y SALARIO}>30000)}(\text{EMPLEADO})$

En SQL:

Todos los atributos

SELECT * FROM EMPLEADO

WHERE (ND=4 AND SALARIO>25000) OR (ND=5 AND SALARIO >30000);

NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIRECCIÓN
Franklin	T	Wong	333445555	1955-12-08	Valle 638, Houston, TX
Jennifer	S	Wallace	987654321	1941-06-20	Bravo 291, Bellaire, TX
Ramesh	K	Narayan	666884444	1962-09-15	Espiga 875, Heras, TX

SEXO	SALARIO	NSS_SUPERV	ND
H	40.000	888665555	5
M	43.000	888665555	4
H	38.000	333445555	5

$\sigma_{\text{SALARIO} \geq 38000 \text{ Y } ND \neq 5}(\text{EMPLEADO})$

En SQL:

SELECT * FROM EMPLEADO WHERE SALARIO >= 38000 AND ND < > 5;

Álgebra relacional (8): REUNIÓN (o JOIN)

REUNIÓN (INTERNA)

$R \bowtie_{\langle \text{condición de reunión} \rangle} S$ (En SQL: cláusula **FROM**)

El resultado de la Reunión interna consta de todas las combinaciones de cada tupla de **R** seguida de otra de **S**, **que satisfagan la condición de reunión.**

Es equivalente a un producto cartesiano (que tiene todas las combinaciones posibles entre las tuplas de R y S) seguido de una selección.

- Operación binaria: intervienen dos relaciones R y S
- Operación muy importante: Permite procesar vínculos entre relaciones y combinar los datos de varias tablas, tablas que no tienen por qué ser compatibles con la unión.
- Eliminación de duplicados.
- **Reunión externa:** Variante que ofrece en el resultado todas las tuplas de R y/o de S, aunque no cumplan la condición de reunión.

Álgebra relacional (9): Ejemplo de Reunión

- Obtener los datos de cada departamento junto con los datos del jefe de cada uno de ellos.

DEPARTAMENTO \bowtie NSS_JEFE=NSS EMPLEADO

(se lee: reunión entre departamento y empleado con NSS_JEFE= NSS)

DEPARTAMENTO

NOMBRED	NUMEROD	NSS_JEFE	FECHA_INIC_JEFE
Investigación	5	333445555	1988-05-22
Administración	4	987654321	1995-01-01
Dirección	1	888665555	1981-06-19

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS	FECHA_NCTO	DIRECCIÓN
John	B	Smith	123456789	1965-01-09	Fresnos 731, Houston, TX
Franklin	T	Wong	333445555	1955-12-08	Valle 638, Houston, TX
Alicia	J	Zelaya	999887777	1968-07-19	Castillo 3321, Sucre, TX
Jennifer	S	Wallace	987654321	1941-06-20	Bravo 291, Bellaire, TX
Ramesh	K	Narayan	666884444	1962-09-15	Espiga 875, Heras, TX
Joyce	A	English	453453453	1972-07-31	Rosas 5631, Houston, TX
Ahmad	V	Jabbar	987987987	1969-03-29	Dalias 980, Houston, TX
Jaime	E	Borg	888665555	1937-11-10	Sorgo 450, Houston, TX

...

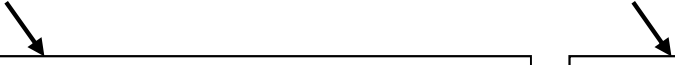
...

SEXO	SALARIO	NSS_SUPERV	ND
H	30.000	333445555	5
H	40.000	888665555	5
M	25.000	987654321	4
M	43.000	888665555	4
H	38.000	333445555	5
M	25.000	333445555	5
H	25.000	987654321	4
H	55.000	nulo	1

Álgebra relacional (10): Ejemplo de Reunión (2)

- (cont.): *Obtener los datos de cada departamento junto con los datos del jefe de cada uno de ellos.*

En SQL:

producto cartesiano *selección*

 SELECT * **FROM** DEPARTAMENTO, EMPLEADO **WHERE** NSS_JEFE=NSS ;

O bien:

SELECT * **FROM** DEPARTAMENTO **INNER JOIN** EMPLEADO **ON** NSS_JEFE=NSS;

RESULTADO

NOMBRED	NÚMEROD	NSS_JEFE	FECHA_INIC_JEFE	NOMBRE	INIC	APELLIDO
Investigación	5	333445555	1988-05-22	Franklin	T	Wong
Administración	4	987654321	1995-01-01	Jennifer	S	Wallace
Dirección	1	888665555	1981-06-19	Jaime	E	Borg

NSS	FECHA_NCTO	DIRECCIÓN	SEXO	SALARIO	NSS_SUPERV	ND
333445555	1955-12-08	Valle 638, Houston, TX	H	40.000	888665555	5
987654321	1941-06-20	Bravo 291, Bellaire, TX	M	43.000	888665555	4
888665555	1937-11-10	Sorgo 450, Houston, TX	H	55.000	nulo	1

Álgebra relacional (11): REUNIÓN NATURAL

- **Reunión natural (Natural join):**
 - Reunión en base a **todos los pares de atributos de igual nombre** en R y S.
 - Exige algún par de atributos que tengan el mismo nombre en las dos relaciones que intervienen.
 - Es una equirreunión seguida de la eliminación de atributos superfluos (se eliminan los atributos cuyo nombre se repite).

REUNIÓN NATURAL

R * S

(En SQL: cláusula **FROM**)

Una **Reunión Natural** es un tipo especial de Reunión en la que la **condición de reunión** está formada por tantas condiciones de igualdad unidas mediante el operador lógico **Y** como pares de atributos tengan el mismo nombre en R y S.

Álgebra relacional (12): Ejemplo de reunión natural

- *Juntar cada empleado con los datos del departamento para el que trabaja*
EMPLEADO * DEPTO

En SQL:

SELECT * FROM EMPLEADO NATURAL JOIN DEPTO

- La condición de reunión está implícita → ND del EMPLEADO = ND del DEPTO

EMPLEADO

NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIRECCIÓN
John	B	Smith	123456789	1965-01-09	Fresnos 731, Houston, TX
Franklin	T	Wong	333445555	1955-12-08	Valle 638, Houston, TX
Alicia	J	Zelaya	999887777	1968-07-19	Castillo 3321, Sucre, TX
Jennifer	S	Wallace	987654321	1941-06-20	Bravo 291, Bellaire, TX
Ramesh	K	Narayan	666884444	1962-09-15	Espiga 875, Heras, TX
Joyce	A	English	453453453	1972-07-31	Rosas 5631, Houston, TX
Ahmad	V	Jabbar	987987987	1969-03-29	Dalias 980, Houston, TX
Jaime	E	Borg	888665555	1937-11-10	Sorgo 450, Houston, TX

...

...

SEXO	SALARIO	NSS_SUPERV	ND
H	30.000	333445555	5
H	40.000	888665555	5
M	25.000	987654321	4
M	43.000	888665555	4
H	38.000	333445555	5
M	25.000	333445555	5
H	25.000	987654321	4
H	55.000	nulo	1

DEPTO

NOMBRED	<u>ND</u>	NSS_JEFE	FECHA_INIC_JEFE
Investigación	5	333445555	1988-05-22
Administración	4	987654321	1995-01-01
Dirección	1	888665555	1981-06-19

Álgebra relacional (13): Ejemplo de reunión natural (2)

DEPTO

NOMBRED	<u>ND</u>	NSS_JEFE	FECHA_INIC_JEFE
Investigación	5	333445555	1988-05-22
Administración	4	987654321	1995-01-01
Dirección	1	888665555	1981-06-19

RESULTADO DE LA REUNIÓN NATURAL

NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIRECCIÓN
John	B	Smith	123456789	1965-01-09	Fresnos 731, Houston, TX
Franklin	T	Wong	333445555	1955-12-08	Valle 638, Houston, TX
Alicia	J	Zelaya	999887777	1968-07-19	Castillo 3321, Sucre, TX
Jennifer	S	Wallace	987654321	1941-06-20	Bravo 291, Bellaire, TX
Ramesh	K	Narayan	666884444	1962-09-15	Espiga 875, Heras, TX
Joyce	A	English	453453453	1972-07-31	Rosas 5631, Houston, TX
Ahmad	V	Jabbar	987987987	1969-03-29	Dalias 980, Houston, TX
Jaime	E	Borg	888665555	1937-11-10	Sorgo 450, Houston, TX

SEXO	SALARIO	NSS_SUPERV	<u>ND</u>	NOMBRED	NSS_JEFE	FECHA_INIC_JEFE
H	30.000	333445555	5	Investigación	333445555	1988-05-22
H	40.000	888665555	5	Investigación	333445555	1988-05-22
M	25.000	987654321	4	Administración	987654321	1995-01-01
M	43.000	888665555	4	Administración	987654321	1995-01-01
H	38.000	333445555	5	Investigación	333445555	1988-05-22
M	25.000	333445555	5	Investigación	333445555	1988-05-22
H	25.000	987654321	4	Administración	987654321	1995-01-01
H	55.000	nulo	1	Dirección	888665555	1981-06-19

- Si hubiera varios pares de atributos con igual nombre se seleccionarían sólo las tuplas que igualen todas las parejas de atributos.

SELECT: Cláusulas básicas

SELECT datos a recuperar separados por comas

Atributos de las proyecciones, fórmulas y, en general, cualquier dato que tenga que aparecer en el resultado final. Un * indica todos los atributos de todas las tablas que intervienen.

FROM tablas

Aquí se indican las tablas que intervienen separadas por comas (1) o bien las tablas y sus tipos de reuniones – internas, externas o naturales – especificando los JOIN necesarios (2).

WHERE condiciones

Son las condiciones de selección de las tablas del FROM, a las que hay que unir las condiciones de reunión de las tablas mediante AND si se utilizó la sintaxis (1) en el FROM.

...

ORDER BY atributos de ordenación de los datos

En principio los datos recuperados salen desordenados. Esta opción sirve para ordenarlos, en ascendente (ASC) o descendente (DESC).

SELECT: Ejemplos de consultas básicas

- *Fecha de nacimiento y dirección de John Smith.*

EMPLEADO

NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIRECCIÓN	SEXO	...
--------	------	----------	------------	------------	-----------	------	-----

C1: SELECT FECHA_NCTO, DIRECCION
FROM EMPLEADO
WHERE NOMBRE='John' **AND** APELLIDO='Smith';

$\pi_{\text{FECHA_NCTO, DIRECCIÓN}}(\sigma_{\text{NOMBRE='John' AND APELLIDO='Smith'}}(\text{EMPLEADO}))$

- La cláusula **SELECT** tiene los atributos de π .
- La cláusula **FROM** tiene la relación que interviene.
- La cláusula **WHERE** tiene la condición de σ .

**El resultado de una
consulta SQL
puede contener ...**

tuplas repetidas

SELECT: Ejemplos de consultas básicas (2)

- *Nombre y dirección de los empleados del departamento de Investigación.*

EMPLEADO

NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIRECCIÓN	SEXO	...
				...	SALARIO	NSS_SUPERV	ND

DEPARTAMENTO

NOMBRED	<u>NÚMEROD</u>	NSS_JEFE	FECHA_INIC_JEFE
---------	----------------	----------	-----------------

C2: SELECT NOMBRE, APELLIDO, DIRECCION

FROM EMPLEADO, DEPARTAMENTO

WHERE NOMBRED='Investigacion' **AND** NUMEROD=ND;

- Secuencia de operaciones del álgebra relacional: $\sigma - \bowtie - \pi$

$DI \leftarrow \sigma_{NOMBRED='Investigación'}(DEPARTAMENTO)$

$EM_DI \leftarrow DI \bowtie_{NUMEROD=ND} EMPLEADO$

$RESUL \leftarrow \pi_{NOMBRED, APELLIDO, DIRECCIÓN}(EM_DI)$

- **WHERE:** Condiciones de σ y \bowtie

SELECT: Ejemplos de consultas básicas (3)

- *Nº de proyecto, nº de departamento que lo controla, apellido, dirección y fecha de nacimiento del jefe del depto. de todos los proyectos realizados en Stafford.*

EMPLEADO

NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIRECCIÓN	...
--------	------	----------	------------	------------	-----------	-----

DEPARTAMENTO

NOMBRED	<u>NÚMEROD</u>	NSS_JEFE	FECHA_INIC_JEFE
---------	----------------	----------	-----------------

PROYECTO

NOMBREP	<u>NÚMEROP</u>	LOCALIZACIÓNP	NÚMD
---------	----------------	---------------	------

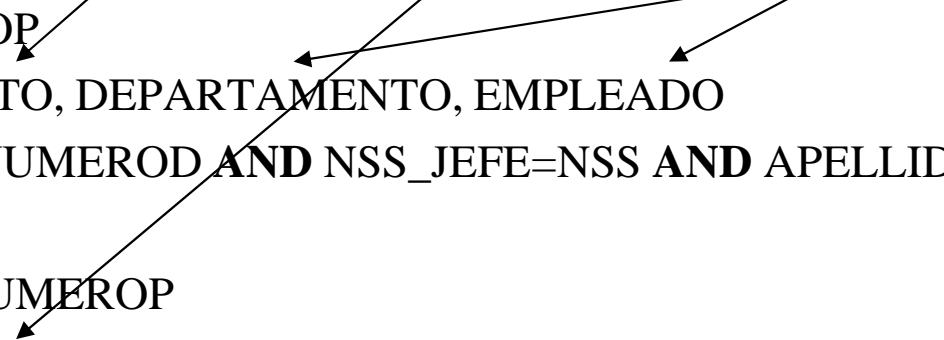
C3: SELECT NUMEROP, NUMD, APELLIDO, DIRECCION, FECHA_NCTO
FROM PROYECTO, DEPARTAMENTO, EMPLEADO
WHERE LOCALIZACIONP='Stafford' **AND** NUMD=NUMEROD **AND** NSS_JEFE=NSS;

- **WHERE:** Condiciones de σ y \bowtie
- Condiciones de selección (σ):
 - LOCALIZACIÓNP='Stafford'
- Condiciones de reunión (\bowtie) entre las tablas que intervienen:
 - NÚMD=NÚMEROD relaciona un proyecto con su departamento controlador.
 - NSS_JEFE=NSS relaciona ese departamento con el empleado que lo dirige.

SELECT: Ejemplos de consultas básicas (4)

- En SQL existen las operaciones de teoría de conjuntos **UNION**, **INTERSECT** y **EXCEPT** (diferencia) que involucran a dos **SELECT**.
- Se exige compatibilidad de unión: mismos atributos y mismo orden entre ellos. El operador **AS** sirve para que los atributos involucrados tengan los mismos nombres en los resultados de las dos **SELECT**.
- Por defecto las filas repetidas se eliminan del resultado. Incluyendo **ALL** se conservan las repeticiones.
- **Ejemplo de unión:** *Números de proyecto donde participa Wong como jefe del departamento controlador o como trabajador en el proyecto:*

C4: (**SELECT** NUMEROP
FROM PROYECTO, DEPARTAMENTO, EMPLEADO
WHERE NUMD=NUMEROD AND NSS_JEFE=NSS AND APELLIDO= 'Wong')
UNION
(**SELECT** NP AS NUMEROP
FROM TRABAJA_EN, EMPLEADO
WHERE NSSE=NSS AND APELLIDO='Wong');



SELECT: Calificación de atributos - El punto (.)

- La calificación de atributos se debe usar obligatoriamente cuando coinciden los nombres de los atributos en varias tablas.
- Nombre, apellido y dirección de los empleados del departamento de investigación.*

EMPLEADO					
NOMBRE	INIC	APELLIDO	<u>NSS</u>	FECHA_NCTO	DIR
DEPTO			SEXO	SALARIO	NSS_SUPERV
NOMBRE	<u>ND</u>	NSS_JEFE	FECHA_INIC_JEFE		



C5: SELECT EMPLEADO.NOMBRE, APELLIDO, DIR
FROM EMPLEADO, DEPTO
WHERE DEPTO.NOMBRE = 'Investigación' AND DEPTO.ND=EMPLEADO.ND ;



SELECT: Todos los atributos - El asterisco (*)

- El * selecciona todos los atributos de las relaciones de **FROM**.

- *Todos los datos de los empleados del departamento número 5.*

```
SELECT *  
FROM EMPLEADO  
WHERE ND=5;
```

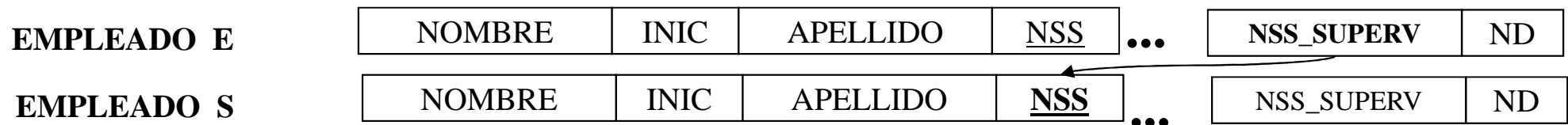
- *Todos los datos de los empleados del departamento de investigación.*

```
SELECT *  
FROM EMPLEADO, DEPARTAMENTO  
WHERE NOMBRED='Investigacion' AND ND=NUMEROD;
```

- En este último ejemplo saldrían todos los atributos de EMPLEADO y todos los atributos de DEPARTAMENTO.
- También se pueden calificar los atributos del *. Por ejemplo, si se pone EMPLEADO.* solo saldrían en la solución los atributos de la tabla EMPLEADO (y no las de DEPARTAMENTO).

SELECT: Seudónimos de tablas y atributos - AS

- Los pseudónimos permiten hacer **varias referencias a una misma relación**.
- *Nombre, apellido de cada empleado junto al nombre y apellido de su supervisor.*



C6: `SELECT E.NOMBRE, E.APELLIDO, S.NOMBRE, S.APELLIDO
FROM EMPLEADO AS E, EMPLEADO AS S
WHERE E.NSS_SUPERV=S.NSS;`

- Se pueden usar pseudónimos aunque no haya varias referencias a una relación. Por ejemplo, C5 se puede reescribir para usar nombres más cortos en sus tablas:

C5A: `SELECT E.NOMBRE, E.APELLIDO, E.DIR
FROM EMPLEADO AS E, DEPTO AS D
WHERE D.NOMBRE = 'Investigación' AND D.ND=E.ND;`

SELECT: Seudónimos de tablas y atributos - AS (2)

- La construcción **AS** sirve para declarar los seudónimos de las tablas.
- El uso de **AS** es opcional. Así, EMPLEADO **AS** E es lo mismo que EMPLEADO E.
- **AS** también permite cambiar el nombre de cualquier columna (atributo) que aparezca en el resultado.

C6A: SELECT E.NOMBRE **AS** NOMBRE_EMPLEADO,
E.APELLIDO **AS** APEL_EMPLEADO,
S.NOMBRE **AS** NOMBRE_SUPERVISOR,
S.APELLIDO **AS** APEL_SUPERVISOR
FROM EMPLEADO **AS** E, EMPLEADO **AS** S
WHERE E.NSS_SUPERV = S.NSS;

En MYSQL NO

- Algunos SGBD permiten además cambiar los nombres de atributos al poner un nuevo nombre a la tabla:

FROM EMPLEADO **AS** E(NP, INI, AP, NSS, FN, DIR, SEX, SAL, NSSS, ND), ...

SELECT: Resultados repetidos o distintos - ALL y DISTINCT

- SQL no elimina automáticamente las tuplas repetidas.
- Si sólo nos interesan los resultados distintos hay que indicarlo con DISTINCT.
- *Obtener todos los salarios de la empresa aunque estén repetidos.*

C7: SELECT SALARIO FROM EMPLEADO;

Es lo mismo que:

C7A: SELECT ALL SALARIO FROM EMPLEADO;

**Poner ALL o no poner nada
implica que en el resultado
pueden salir filas repetidas**

- *Obtener los salarios diferentes que hay en la empresa.*

C7B: SELECT DISTINCT SALARIO FROM EMPLEADO;

- *Obtener las combinaciones (salario, apellido) diferentes que hay en la empresa.*

C7C: SELECT DISTINCT SALARIO, APELLIDO FROM EMPLEADO;

- En este ejemplo no puede haber dos filas con igual salario y apellido...
- ..., pero sí puede haber dos filas con el mismo salario.

SELECT: Operaciones aritméticas y de concatenación

- La instrucción SELECT admite operaciones aritméticas y de concatenación en sus cláusulas.
- Los **operadores aritméticos** son '+', '-', '*' y '/'
 - '+', '-' también sirve para sumar/restar a fecha, hora o marca de tiempo, un intervalo compatible.
 - Se puede calcular un intervalo como diferencia entre fechas, horas o marcas de tiempo.
- El operador '||' **concatena** cadenas de caracteres.
- *Nombre y salario de los empleados que trabajan en 'ProductoX' al aumentarles el sueldo un 10% :*

C8: SELECT NOMBRE, APELLIDO, 1.1*SALARIO
FROM EMPLEADO, TRABAJA_EN, PROYECTO
WHERE NSS=NSSE **AND** NP=NUMEROP **AND** NOMBREP='ProductoX';

SELECT: La cláusula WHERE y su omisión

- **WHERE** indica las condiciones que deben cumplir las filas que saldrán en el resultado. Operadores para las condiciones:
 - Valores nulos: IS NULL / IS NOT NULL
 - De comparación y lógicos
 - Comparación de subcadenas: LIKE
 - Conjuntos de valores: IN / NOT IN y BETWEEN
- Estos operadores se pueden usar en cualquier sitio que haya una condición
- La **omisión de WHERE** indica una selección de tuplas incondicional (es decir, se cogen todas las filas). Equivale a **WHERE TRUE**. Sin WHERE y con más de una relación en FROM, se hace un producto cartesiano.
 - Ejemplos:
 - *NSS de todos los empleados:*
SELECT NSS FROM EMPLEADO;
 - *Combinaciones posibles de NSS de empleados con nombres de departamento:*
SELECT NSS, NOMBRED FROM EMPLEADO , DEPARTAMENTO;

SELECT: Valores nulos - IS NULL / IS NOT NULL

- SQL considera el **NULL** (valor nulo) de diferente forma que al resto de datos.
 - Las tuplas con valores nulos en el atributo de reunión no se incluyen en el resultado (salvo si es una reunión externa).
 - Cualquier cosa operada con NULL devuelve NULL (UNKNOWN = desconocido).
- Para ver si un dato es nulo o no, en lugar de = y ≠ se usa **IS NULL** e **IS NOT NULL**.

- *Nombre y apellido de los empleados sin supervisor conocido.*

C9: SELECT NOMBRE, APELLIDO FROM EMPLEADO WHERE NSS_SUPERV IS NULL ;

- Si se pusiera **WHERE NSS_SUPERV = NULL**
- Para las filas con NSS_SUPERV nulo se estaría comparando si NULL = NULL.
- Esta comparación NO devuelve ni cierto ni falso. Devuelve NULL, que es desconocido, con lo que no se cumpliría la condición y no saldría en la solución.

- *Nombre y apellido de los empleados con supervisor.*

C9A: SELECT NOMBRE, APELLIDO FROM EMPLEADO WHERE NSS_SUPERV IS NOT NULL ;

SELECT: Operadores de comparación y lógicos

- **Operadores de comparación** para una condición simple:
 - Igual =
 - Menor que <
 - Menor o igual que <=
 - Mayor que >
 - Mayor o igual que >=
 - Distinto de < >
 - **Operadores lógicos** para unir varias condiciones simples:
 - Y lógico **AND**
 - O lógico **OR**
 - No lógico **NOT**
 - *NSS de los empleados que trabajan en los proyectos 1, 10, 20 ó 30.*
- C10: SELECT DISTINCT NSSE FROM TRABAJA_EN
WHERE NP=1 OR NP=10 OR NP=20 OR NP=30;**

SELECT: Comparación de subcadenas - LIKE

- El operador **LIKE** permite comparar el dato con una subcadena de caracteres.
 - `_` sustituye a un carácter arbitrario.
 - `%` a un n° indeterminado de caracteres.
- *Empleados que viven en Houston, estado de Texas:*

```
SELECT NOMBRE, APELLIDO FROM EMPLEADO  
WHERE DIRECCION LIKE '%Houston, TX%';
```

- *Empleados que nacieron en la década de 1950:*

```
SELECT NOMBRE, APELLIDO FROM EMPLEADO  
WHERE FECHA_NCTO LIKE ' __ 5%';
```

SELECT: Conjuntos de valores - BETWEEN , IN/NOT IN

- **BETWEEN** sirve para indicar un rango de valores para el dato.
- *NSS y salario de los empleados del departamento 5 cuyo salario está entre 30.000 y 40.000 \$.*

```
SELECT NSS, SALARIO FROM EMPLEADO  
WHERE ND=5 AND SALARIO BETWEEN 30000 AND 40000;
```

Entre 30000 y 40000
ambos incluidos

- El operador **IN** sirve para indicar un conjunto explícito de valores para el dato. Se puede traducir como “está en”.
- El conjunto explícito se encierra entre paréntesis.
- *NSS de los empleados que trabajan en los proyectos 1, 10, 20 ó 30.*

```
C10A: SELECT DISTINCT NSSE FROM TRABAJA_EN  
WHERE NP IN (1, 10, 20,30);
```

SELECT: Conjuntos de valores - BETWEEN , IN/NOT IN (2)

- El operador **IN** admite utilizar el resultado de una **SELECT** como conjunto de valores, en vez de tener una lista fija de valores.
- *Nombre, apellido y salarios de los empleados que trabajan en el departamento de Investigación.*

**C11: SELECT NOMBRE, APELLIDO, SALARIO FROM EMPLEADO
WHERE ND IN (SELECT NUMEROD FROM DEPARTAMENTO
WHERE NOMBRED= 'Investigacion');**

- También existe el operador **NOT IN** (se puede interpretar como “no está en”)
- *Nombre, apellido y salarios de los empleados que NO trabajan en el departamento de Investigación.*

**C11A: SELECT NOMBRE, APELLIDO, SALARIO FROM EMPLEADO
WHERE ND NOT IN (SELECT NUMEROD FROM DEPARTAMENTO
WHERE NOMBRED= 'Investigacion');**

SELECT: Ordenación de los resultados - ORDER BY

- *Empleados y proyectos donde trabajan ordenados en descendente por nombre de departamento y en cada departamento alfabéticamente por apellido y nombre del proyecto:*

**C12: SELECT NOMBRED, APELLIDO, NOMBRE, NOMBREP
FROM DEPARTAMENTO, EMPLEADO, TRABAJA_EN, PROYECTO
WHERE NUMEROD=ND AND NSS=NSSE AND NP=NUMEROP
ORDER BY NOMBRED DESC, APELLIDO ASC, NOMBREP;**

- Por defecto (si no se pone nada), el orden es ascendente.
- Se puede indicar un orden para cada campo.
 - **DESC** indica orden descendente.
 - **ASC** indica orden ascendente.

SELECT: Tablas reunidas - JOIN EN FROM

- El lenguaje SQL permite dos tipos de sintaxis para indicar una reunión:
 - (1) Poner en el FROM las tablas separadas por comas y en el WHERE las condiciones de reunión unidas con AND.
 - No sirve para cualquier tipo de reunión, ya que no permite indicar qué tipo de reunión es.
 - (2) Poner todo en el FROM, tanto las tablas como las condiciones de reunión y su tipo de reunión.

SELECT: Tablas reunidas - JOIN EN FROM (2)

- En la cláusula FROM se pueden especificar los diferentes **tipos de reunión** entre cada dos tablas que intervengan en la consulta:
 - Reunión interna: **INNER JOIN** (o solo **JOIN**)
 - Reunión natural: **NATURAL JOIN**
 - Reunión externa:
 - Izquierda: **LEFT [OUTER] JOIN**
 - Derecha: **RIGHT [OUTER] JOIN**
 - Ambas: **FULL [OUTER] JOIN**
- NO se pueden definir seudónimos de tablas reunidas:

FROM (EMPLEADO INNER JOIN DEPARTAMENTO ON ND=NÚMEROD) ~~AS ED ...~~

SELECT: Tablas reunidas - JOIN EN FROM (3)

- **Ejemplo de reunión interna:** *Nombre y dirección de los empleados del departamento de Investigación.*

C2: SELECT NOMBRE, APELLIDO, DIRECCION

FROM EMPLEADO, DEPARTAMENTO

WHERE NOMBRED='Investigacion' **AND** ND=NUMEROD ;

Es lo mismo que

C2A: SELECT NOMBRE, APELLIDO, DIRECCION

FROM EMPLEADO **INNER JOIN** DEPARTAMENTO **ON** ND=NUMEROD

WHERE NOMBRED='Investigacion';

SELECT: Tablas reunidas - JOIN EN FROM (4)

- Ejemplo de reunión externa:** *Obtener el apellido de todos los empleados y al lado el de su supervisor, si es que lo tiene. Si no lo tiene, el apellido del supervisor debe salir en blanco.*

EMPLEADO

NOMBRE	INIC	APELLIDO	<u>NSS</u>
John	B	Smith	123456789
Franklin	T	Wong	333445555
Alicia	J	Zelaya	999887777
Jennifer	S	Wallace	987654321
Ramesh	K	Narayan	666884444
Joyce	A	English	453453453
Ahmad	V	Jabbar	987987987
Jaime	E	Borg	888665555

...

<u>NSS_SUPERV</u>
333445555
888665555
987654321
888665555
333445555
333445555
987654321
nulo

...

C6B: SELECT E.APELLIDO AS APEL_EMPLEADO,
S.APELLIDO AS APEL_SUPERVISOR
FROM EMPLEADO E **LEFT OUTER JOIN** EMPLEADO S **ON** E.NSS_SUPERV=S.NSS;

RESULTADO

APEL_EMPLEADO	APEL_SUPERVISOR
Smith	Wong
Wong	Borg
Zelaya	Wallace
Wallace	Borg
Narayan	Wong
English	Wong
Jabbar	Wallace
Borg	

- Con **INNER JOIN** la última fila no saldría en el resultado, por no cumplir la condición de reunión.

SELECT: Tablas reunidas - JOIN EN FROM (5)

- Ejemplo para ver la diferencia entre reunión interna y externa:

Obtener los datos de cada empleado junto con los de sus familiares

- Si se resuelve mediante **reunión interna**:

```
SELECT * FROM EMPLEADO, DEPENDIENTE WHERE NSS=NSSE;
```

O bien

```
SELECT * FROM EMPLEADO INNER JOIN DEPENDIENTE ON NSS=NSSE;
```

– *No salen los empleados que no tengan familiares.*

- Si se resuelve mediante **reunión externa**:

```
SELECT * FROM EMPLEADO LEFT OUTER JOIN DEPENDIENTE ON NSS=NSSE;
```

– *Se obtienen TODOS los empleados aunque no tengan familiares.*

– *En los empleados sin familiares, la parte de los familiares aparecerá vacía (con nulos).*

SELECT: Tablas reunidas - JOIN EN FROM (6)

- **Ejemplo de reunión natural:** *Añadir a cada departamento los lugares de ubicación de cada departamento*

SELECT * FROM DEPARTAMENTO NATURAL JOIN LOCALIZACIONES_DEPT;

DEPARTAMENTO

NOMBRED	<u>NÚMEROD</u>	NSS_JEFE	FECHA_INIC_JEFE
Investigación	5	333445555	1988-05-22
Administración	4	987654321	1995-01-01
Dirección	1	888665555	1981-06-19

LOCALIZACIONES_DEPT

<u>NÚMEROD</u>	<u>LOCALIZACIÓN</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

RESULTADO DE LA REUNIÓN

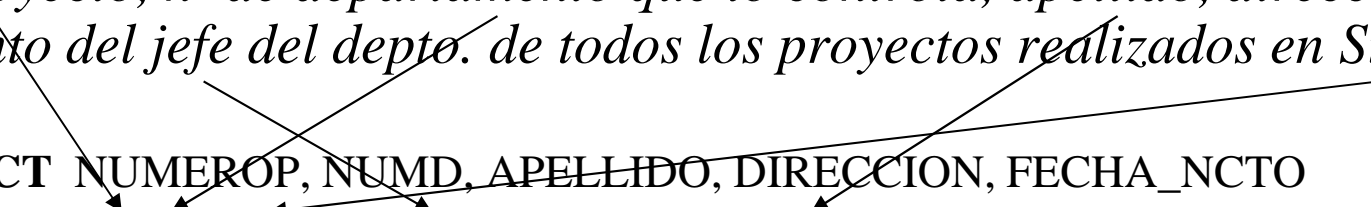
NOMBRED	<u>NÚMEROD</u>	NSS_JEFE	FECHA_INIC_JEFE	<u>LOCALIZACIÓN</u>
Dirección	1	888665555	1981-06-19	Houston
Administración	4	987654321	1995-01-01	Stafford
Investigación	5	333445555	1988-05-22	Bellaire
Investigación	5	333445555	1988-05-22	Sugarland
Investigación	5	333445555	1988-05-22	Houston

- Si hubiera varios pares de atributos con igual nombre se seleccionarían sólo las filas que igualen todas las parejas de atributos.

SELECT: Tablas reunidas - JOIN EN FROM (7)

- La reunión en una SELECT puede necesitar cualquier número de tablas, es decir, los **JOIN** pueden ser **entre más de dos tablas**.
- Para reunir más de dos tablas, se indica mediante paréntesis la reunión de las dos primeras tablas con la siguiente, la reunión resultante de estas tres tablas con la siguiente, etc.
- *Nº de proyecto, nº de departamento que lo controla, apellido, dirección y fecha de nacimiento del jefe del depto. de todos los proyectos realizados en Stafford.*

C3: **SELECT** NUMEROP, NUMD, APELLIDO, DIRECCION, FECHA_NCTO
 FROM PROYECTO, DEPARTAMENTO, EMPLEADO
 WHERE NUMD=NUMEROD **AND** NSS_JEFE=NSS **AND** LOCALIZACIONP='Stafford';



C3A: **SELECT** NUMEROP, NUMD, APELLIDO, DIRECCION, FECHA_NCTO
 FROM (PROYECTO **INNER JOIN** DEPARTAMENTO **ON** NUMD=NUMEROD)
 INNER JOIN EMPLEADO **ON** NSS_JEFE=NSS
 WHERE LOCALIZACIONP='Stafford';

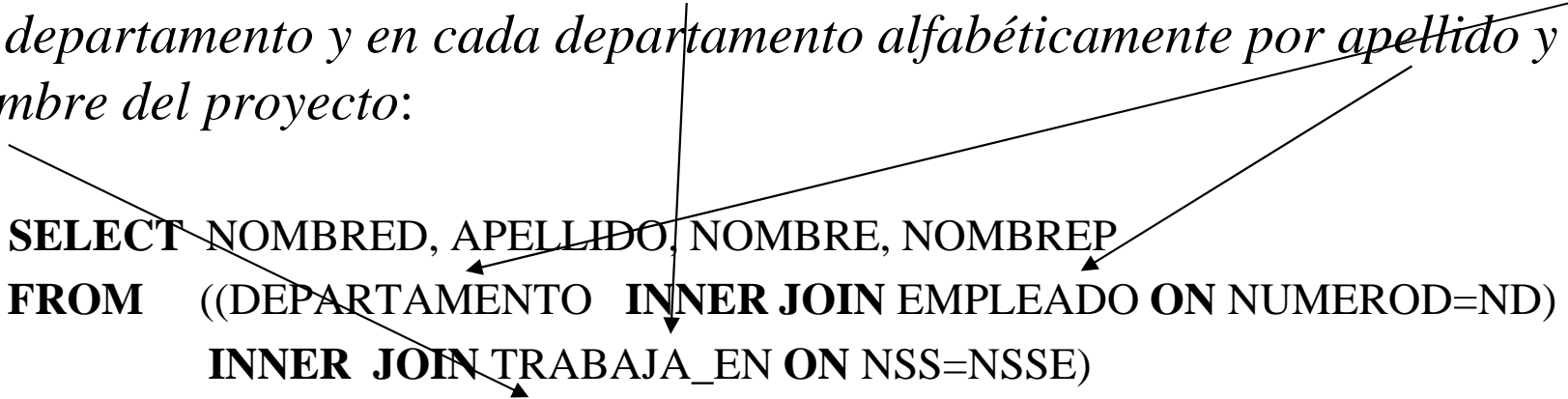
SELECT: Tablas reunidas - JOIN EN FROM (8)

- *Nombre y salario de los empleados que trabajan en 'ProductoX' al aumentarles el sueldo un 10% :*

C8A: SELECT NOMBRE, APELLIDO, 1.1*SALARIO
FROM (EMPLEADO **INNER JOIN** TRABAJA_EN **ON** NSS=NSSE)
INNER JOIN PROYECTO **ON** NP=NUMEROP
WHERE NOMBREP='ProductoX';

- *Empleados y proyectos donde trabajan ordenados en descendente por nombre de departamento y en cada departamento alfabéticamente por apellido y nombre del proyecto:*

C12A: SELECT NOMBRED, APELLIDO, NOMBRE, NOMBREP
FROM ((DEPARTAMENTO **INNER JOIN** EMPLEADO **ON** NUMEROD=ND)
INNER JOIN TRABAJA_EN **ON** NSS=NSSE)
INNER JOIN PROYECTO **ON** NP=NUMEROP
ORDER BY NOMBRED **DESC**, APELLIDO **ASC**, NOMBREP;



SELECT: Funciones agregadas

- Las **funciones agregadas** (\mathfrak{F}) son otras operaciones no básicas del álgebra relacional, incluidas en el lenguaje SQL de los SGBD.

F GÓTICA

<atribs agrupación> \mathfrak{F} <lista funciones> (**R**)

Devuelve una relación con los resultados de la lista de funciones, agrupados por los atributos indicados.

<atribs agrupación>: Uno o varios atributos separados por comas. Son los atributos por los que queremos que aparezca cada resultado.

<lista funciones>: Son pares <función agregada> <atributo> separados por comas.

<función agregada>: Las más comunes son:

CUENTA : Contabiliza el número de tuplas no nulas del atributo indicado.

SUMA: Suma el contenido de las tuplas del atributo indicado.

PROMEDIO: Lo mismo para la media.

MÁXIMO: Calcula el valor máximo contenido en las tuplas del atributo indicado.

MÍNIMO: Lo mismo para el valor mínimo.

SELECT: Funciones agregadas (2)

- En SQL, existen varias **funciones agregadas** que se pueden incluir en las distintas cláusulas de la instrucción SELECT.
- Se aplican a una columna de una tabla y son:
 - Cuenta: **COUNT**(columna) → cuenta filas no vacías
 - Suma: **SUM**(columna) → devuelve la suma de los datos de esa columna.
 - Promedio: **AVG**(columna) → devuelve la media de esa columna.
 - Máximo: **MAX**(columna) → devuelve el mayor valor de esa columna.
 - Mínimo: **MIN**(columna) → devuelve el menor valor de esa columna
- Y los **atributos de agrupación** se indican en la cláusula **GROUP BY**.

SELECT: Cláusulas con funciones agregadas

SELECT [**DISTINCT**] datos a recuperar separados por comas

Atributos de las proyecciones, fórmulas, funciones agregadas, en general cualquier dato que haya que recuperar.

FROM tablas

Aquí se indican las tablas que intervienen separadas por comas (1) o bien las tablas y sus reuniones anidadas mediante paréntesis sucesivos (2). En este último caso se indica el tipo y condiciones de cada reunión, que podrá ser INNER JOIN, NATURAL JOIN u OUTER JOIN.

WHERE condiciones

Son las condiciones de selección de las tablas del FROM, a las que hay que unir las condiciones de reunión de las tablas mediante AND si se utilizó la sintaxis (1) en el FROM.

GROUP BY atributos de agrupación de las funciones agregadas, separados por comas

HAVING condiciones sobre resultados asociados a GROUP BY

Solo aparecen si se utilizan funciones agregadas (COUNT, SUM, AVG, MAX o MIN).

GROUP BY indica los atributos por los que aparecerán desglosados los resultados obtenidos por las funciones agregadas. HAVING se usa para poner condiciones sobre los resultados de las funciones agregadas.

ORDER BY atributos de ordenación de los datos

En principio los datos recuperados salen desordenados. Esta opción sirve para ordenarlos, en ascendente (ASC) o descendente (DESC).

SELECT: Ejemplos de funciones agregadas

- Ejemplo 1: *Obtener el número de empleados de la empresa y la media de sus salarios*

Σ CUENTA NSS, PROMEDIO SALARIO(EMPLEADO)

En SQL:

C13: SELECT COUNT(NSS), AVG(SALARIO)
FROM EMPLEADO;

COUNT(NSS)	AVG(SALARIO)
8	35125

- Ejemplo 2: *Obtener, para cada departamento, el número de empleados y la media de sus salarios*

ND Σ CUENTA NSS, PROMEDIO SALARIO(EMPLEADO)

En SQL:

C13A: SELECT ND, COUNT(NSS), AVG(SALARIO)
FROM EMPLEADO
GROUP BY ND;

ND	COUNT(NSS)	AVG(SALARIO)
5	4	33250
4	3	31000
1	1	55000

SELECT: Ejemplos de funciones agregadas (2)

- Suma de salarios de los empleados, junto a salarios máximo, mínimo y medio:

C14: SELECT SUM(SALARIO), MAX(SALARIO), MIN(SALARIO), AVG(SALARIO)
FROM EMPLEADO;

- Lo mismo para empleados del departamento 'Investigación':

C15: SELECT SUM(SALARIO), MAX(SALARIO), MIN(SALARIO), AVG(SALARIO)
FROM EMPLEADO INNER JOIN DEPARTAMENTO ON ND=NUMEROD
WHERE NOMBRED='Investigacion';

EMPLEADO		
NOMBRE	SALARIO	ND
John	nulo	5
Franklin	15.000	5
Ramesh	10.000	5
Joyce	10.000	5
Alicia	10.000	4
Jennifer	20.000	4
Ahmad	20.000	4
Jaime	20.000	1

DEPARTAMENTO

NOMBRED	NÚMEROD
Investigación	5
Administración	4
Dirección	1

C14:

SUM(SALARIO)	MAX(SALARIO)	MIN(SALARIO)	AVG(SALARIO)
105.000	20.000	10.000	15.000

C15:

SUM(SALARIO)	MAX(SALARIO)	MIN(SALARIO)	AVG(SALARIO)
35.000	15.000	10.000	11.666

35.000 / 3

SELECT: Variantes de la función COUNT

- Cuántos empleados hay en la EMPRESA y cuántos en el departamento 'Investigación':*

C16: SELECT COUNT(*) FROM EMPLEADO;

El * se refiere a filas.
Cuenta todas las filas de la tabla

**C16A: SELECT COUNT(*)
FROM EMPLEADO INNER JOIN DEPARTAMENTO ON ND=NUMEROD
WHERE NOMBRED='Investigacion';**

- Cuántos salarios hay:*

Cuenta los salarios distintos que hay. No cuenta el valor nulo

C17: SELECT COUNT(DISTINCT SALARIO) FROM EMPLEADO;

C17A: SELECT COUNT(SALARIO) FROM EMPLEADO;

Cuenta las filas con salario no nulo

EMPLEADO

NOMBRE	SALARIO	ND
John	nulo	5
Franklin	15.000	nulo
Ramesh	10.000	5
Joyce	10.000	5
Alicia	10.000	4
Jennifer	20.000	4
Ahmad	20.000	4
Jaime	20.000	1

C17:

COUNT(DISTINCT SALARIO)
3

C17A:

COUNT(SALARIO)
7

C16:

COUNT(*)
8

C16A:

COUNT(*)
3

SELECT: Atributos de agrupación - GROUP BY

- En la cláusula **GROUP BY** se indica el/los atributo/s de agrupación.
 - Todo atributo de la cláusula **SELECT** ha de estar en el **GROUP BY**.
 - En la cláusula **SELECT** se ponen algunos/todos los de **GROUP BY**.
- Obtener, para cada departamento con salario medio menor de 35000, el número de empleados y la media de sus salarios :*

**C13B: SELECT ND, COUNT(*), AVG(SALARIO) FROM EMPLEADO
GROUP BY ND HAVING AVG(SALARIO)<35000;**

EMPLEADO

NOMBRE	SALARIO	ND
John	30.000	5
Franklin	40.000	5
Ramesh	38.000	5
Joyce	25.000	5
Alicia	25.000	4
Jennifer	43.000	4
Ahmad	25.000	4
Jaime	55.000	1

C13B:

ND	COUNT(*)	AVG(SALARIO)
5	4	33250
4	3	31000

- Los valores nulos forman su propio grupo.

EMPLEADO

NOMBRE	SALARIO	ND
Jon	null	null
Juan	10	null
Rosa	10	5
Ana	10	5

ND	COUNT(*)	AVG(SALARIO)
null	2	10
5	2	10

SELECT: Atributos de agrupación - GROUP BY (2)

- *Obtener por cada proyecto su número y nombre junto al número de empleados que trabajan en él.*

C18: SELECT NUMEROP, NOMBREP, COUNT(*)

FROM PROYECTO INNER JOIN TRABAJA_EN ON NUMEROP=NP

GROUP BY NUMEROP, NOMBREP;

Agrupación por varios atributos

PROYECTO
INNER JOIN
TRABAJA_EN:

NOMBREP	NÚMEROP
ProductoX	1
ProductoX	1
ProductoY	2
ProductoY	2
ProductoY	2
ProductoZ	3
ProductoZ	3
Automatización	10
Automatización	10
Automatización	10
Reorganización	20
Reorganización	20
Reorganización	20
Nuevas prestaciones	30
Nuevas prestaciones	30
Nuevas prestaciones	30

NP	HORAS
1	32.5
1	20.0
2	7.5
2	20.0
2	10.0
3	40.0
3	10.0
10	10.0
10	10.0
10	35.0
20	10.0
20	15.0
20	nulo
30	5.0
30	20.0
30	30.0

C18:

NÚMEROP	NOMBREP	COUNT(*)
1	ProductoX	2
2	ProductoY	3
3	ProductoZ	2
10	Automatización	3
20	Reorganización	3
30	Nuevas prestaciones	3

SELECT: Condición para cada grupo - HAVING

- Lo mismo que antes pero sólo para aquellos proyectos donde trabajen más de 2 personas.

C18A: SELECT NUMEROP, NOMBREP, COUNT(*)
FROM PROYECTO INNER JOIN TRABAJA_EN ON NUMEROP=NP
GROUP BY NUMEROP, NOMBREP HAVING COUNT(*)>2;

PROYECTO
INNER JOIN
TRABAJA_EN:

NOMBREP	NÚMEROP
ProductoX	1
ProductoX	1
ProductoY	2
ProductoY	2
ProductoY	2
ProductoZ	3
ProductoZ	3
Automatización	10
Automatización	10
Automatización	10
Reorganización	20
Reorganización	20
Reorganización	20
Nuevas prestaciones	30
Nuevas prestaciones	30
Nuevas prestaciones	30

NP	HORAS	
1	32.5	}
1	20.0	
2	7.5	}
2	20.0	
2	10.0	
3	40.0	}
3	10.0	
10	10.0	}
10	10.0	
10	35.0	
20	10.0	}
20	15.0	
20	nulo	
30	5.0	}
30	20.0	
30	30.0	

C18A:

NÚMEROP	NOMBREP	COUNT(*)
2	ProductoY	3
10	Automatización	3
20	Reorganización	3
30	Nuevas prestaciones	3

SELECT: HAVING vs. WHERE

- Obtener por cada proyecto su número y nombre junto al número de empleados del departamento 5 que trabajan en él.

C19: SELECT NUMEROP, NOMBREP, COUNT(*)

FROM (PROYECTO INNER JOIN TRABAJA_EN ON NUMEROP=NP) INNER JOIN EMPLEADO ON NSS=NSSE

WHERE ND=5 GROUP BY NUMEROP, NOMBREP;

PROYECTO
INNER JOIN
TRABAJA_EN
INNER JOIN
EMPLEADO:

NOMBREP	NÚMEROP	NP	ND	
ProductoX	1	1	5	}
ProductoX	1	1	5	
ProductoY	2	2	5	
ProductoY	2	2	5	}
ProductoY	2	2	5	
ProductoZ	3	3	5	
ProductoZ	3	3	5	}
Automatización	10	10	5	
Automatización	10	10	5	
Automatización	10	10	4	}
Reorganización	20	20	4	
Reorganización	20	20	4	
Reorganización	20	20	5	}
Nuevas prestaciones	30	30	1	
Nuevas prestaciones	30	30	4	
Nuevas prestaciones	30	30	4	}
Nuevas prestaciones	30	30	4	

C19:

NÚMEROP	NOMBREP	COUNT(*)
1	ProductoY	2
2	ProductoY	3
3	ProductoZ	2
10	Automatización	1
20	Reorganización	1

SELECT: Primero WHERE, después HAVING

- *Cuántos empleados hay con salario mayor que 25.000 por departamento. Sólo departamentos con más de 2 empleados con tal sueldo.*

C20: SELECT NOMBRED, COUNT(*)

FROM EMPLEADO INNER JOIN DEPARTAMENTO ON ND = NUMEROD

WHERE SALARIO > 25000

GROUP BY NOMBRED HAVING COUNT(*) > 2;

EMPLEADO
INNER JOIN
DEPARTAMENTO:

NOMBREP
John
Franklin
Ramesh
Joyce
Alicia
Jennifer
Ahmad
Jaime

SALARIO
30.000
40.000
38.000
25.000
25.000
43.000
25.000
55.000

ND
5
5
5
5
4
4
4
1

NOMBRED
Investigación
Investigación
Investigación
Investigación
Administración
Administración
Administración
Dirección

Primero se ejecuta
WHERE

...
INNER JOIN
... WHERE
SALARIO
>25000:

NOMBREP
John
Franklin
Ramesh
Jennifer
Jaime

SALARIO
30.000
40.000
38.000
43.000
55.000

ND
5
5
5
4
1

NOMBRED
Investigación
Investigación
Investigación
Administración
Dirección

NO
NO

C20:

NOMBRED	COUNT(*)
Investigación	3

SQL Avanzado

- **SQL básico** (el visto hasta ahora):
 - LDD Objetos: Base de datos (o esquema) y Tablas
 - LDD Diseño: tipos de datos, CREATE, ALTER y DROP
 - LMD actualización datos: INSERT, UPDATE y DELETE
 - LMD operadores básicos: LIKE, BETWEEN, AND, OR, etc.
 - LMD consulta datos: SELECT y todas sus cláusulas (JOIN, HAVING, etc.)
 - LMD funciones agregadas: COUNT, SUM, AVG, MIN y MAX
- **SQL avanzado:**
 - Más elementos NO obligatorios, que se pueden utilizar en el lenguaje SQL básico.
 - Facilitados por la mayoría de SGBDs: MySQL, SQL Server, etc.
 - Dos tipos:
 - El entorno de programación del SGBD en SQL.
 - Otras funciones incorporadas por el SGBD.

SQL Avanzado - Entorno de programación en SQL

- Además de todo lo visto, algunos SGBD disponen de otros objetos del lenguaje SQL, que enriquecen su entorno de programación. Objetos:
 - Vistas
 - Funciones y Procedimientos almacenados
 - Triggers

- **Vistas***

- Almacenan una SELECT
- Restringen los campos que puede manipular un usuario
- Sintaxis creación:

`CREATE VIEW nombreVista AS SELECT...`

- Borrado:

`DROP VIEW nombreVista;`

*** En MySQL se puede ver la select con `SHOW CREATE TABLE nombreVista;`**

SQL Avanzado - Entorno de programación en SQL (2)

- **Funciones y Procedimientos almacenados**

- Almacenan rutinas (programas del usuario de la BD) en las que, además de las instrucciones SQL, se pueden declarar variables, instrucciones de control, etc. Funcionan de forma similar a cualquier lenguaje de programación.

- Sintaxis creación:

CREATE {FUNCTION | PROCEDURE} nombre (argumentos)
cuerpoDeLaFuncionOProc

- Llamada: CALL nombre(argumentos)

- **Triggers** (disparadores o desencadenadores)

- Es un tipo de procedimiento almacenado que se ejecuta automáticamente cuando se intentan cambiar (entendido como insertar, modificar o borrar) datos de una tabla.
- Sintaxis creación y borrado: con CREATE TRIGGER y DROP TRIGGER

SQL Avanzado - Funciones incorporadas al lenguaje SQL

- Las **funciones incorporadas** son funciones adicionales que el SGBD pone a disposición del usuario en SQL. Tipos más comunes:
 - Funciones matemáticas
 - Funciones de fecha/hora
 - Funciones de conversión entre tipos de datos
 - Funciones avanzadas
- Se pueden utilizar tanto en las instrucciones SQL básicas como dentro del entorno de programación que proporcione el propio SGBD de la BD.
- No todas siguen un estándar:
 - Cada SGBD las puede implementar a su manera.
 - Habrá que revisar su sintaxis concreta en el propio SGBD.
 - A continuación, ejemplos de funciones de fecha y hora en MySQL y sus correspondientes en MS Access.

SQL Avanzado – Funciones de fecha y hora en MySQL

Idioma para los nombres

- Se puede elegir el idioma con la variable local **lc_time_names**
- Ver el contenido de la variable: `SELECT @@lc_time_names ;`
- Cambiar el valor: `SET lc_time_names='código del idioma';`

Algunos códigos (<https://dev.mysql.com/doc/refman/8.0/en/locale-support.html>):

Inglés de EEUU: `en_US` (por defecto)

Castellano de España: `es_ES`

Euskera: `eu_ES`

Formato para las fechas y horas

- El formato por defecto para fechas y horas es `aaaa-mm-dd y hh:mm:ss`
- Se puede cambiar a cualquier otro con la instrucción **date_format** (http://mysql.conclase.net/curso/?sqlfun=DATE_FORMAT)
- Sintaxis: `date_format(fecha , formato);`
- `SELECT DATE_FORMAT('2019-03-05 15:40:22', '%d/%m/%Y');`
daría como resultado solo la fecha y en formato `05/03/2019`

SQL Avanzado - Funciones de Fecha y Hora en MySQL (2)

```
-----  
select now() as Fecha_Hora, curdate() as soloFecha, curtime() as soloHora  
-----
```

```
+-----+-----+-----+  
| Fecha_Hora          | soloFecha | soloHora |  
+-----+-----+-----+  
| 2016-04-14 01:35:01 | 2016-04-14 | 01:35:01 |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
-----  
select '2016-3-2 12:58:07' as Fecha_Hora,  
year('2016-3-2 12:58:07') as Anio,  
month('2016-3-2 12:58:07') as Mes,  
day('2016-3-2 12:58:07') as Dia,  
hour('2016-3-2 12:58:07') as Hora,  
minute('2016-3-2 12:58:07') as Minuto,  
second('2016-3-2 12:58:07') as Segundo  
-----
```

```
+-----+-----+-----+-----+-----+-----+  
| Fecha_Hora          | Anio | Mes | Dia | Hora | Minuto | Segundo |  
+-----+-----+-----+-----+-----+-----+  
| 2016-3-2 12:58:07 | 2016 | 3 | 2 | 12 | 58 | 7 |  
+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
-----  
select '2016-9-13 12:58:07' as Fecha_Hora,  
month('2016-9-13 12:58:07') as NumeroMes,  
monthName('2016-9-13 12:58:07') as NombreMes,  
date_Format('2016-9-13 12:42:59', "%b") as NombreMesAbreviado  
-----
```

```
+-----+-----+-----+-----+  
| Fecha_Hora          | NumeroMes | NombreMes | NombreMesAbreviado |  
+-----+-----+-----+-----+  
| 2016-9-13 12:58:07 | 9 | September | Sep |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

SQL Avanzado - Funciones de Fecha y Hora en MySQL (3)

```
-----
select '2016-9-13 12:58:07' as Fecha_Hora,
weekday('2016-09-13 12:42:59') as NumeroDiaSemana,
dayName('2016-09-13 12:42:59') as NombreDiaSemana,
date_Format('2016-09-13',"%a") as NombreDiaSemanaAbreviado
-----
```

Fecha_Hora	NumeroDiaSemana	NombreDiaSemana	NombreDiaSemanaAbreviado
2016-9-13 12:58:07	1	Tuesday	Tue

1 row in set (0.00 sec)

```
-----
select '2016-04-05 00:00:00' as Fecha1,'2016-04-13 12:42:59' as Fecha2,
datediff('2016-04-13 12:42:59','2016-04-05 00:00:00') as DiasEntreLasDosFechas
-----
```

Fecha1	Fecha2	DiasEntreLasDosFechas
2016-04-05 00:00:00	2016-04-13 12:42:59	8

1 row in set (0.00 sec)

```
-----
select '2016-7-22 20:35:44' as Fecha_Hora,
date_add('2016-7-22 20:35:44', interval 3 year) as Mas3Anios,
date_add('2016-7-22 20:35:44', interval -5 month) as Menos5Meses,
date_add('2016-7-22 20:35:44', interval 16 day) as Mas16Dias
-----
```

Fecha_Hora	Mas3Anios	Menos5Meses	Mas16Dias
2016-7-22 20:35:44	2019-07-22 20:35:44	2016-02-22 20:35:44	2016-08-07 20:35:44

1 row in set (0.00 sec)

SQL Avanzado - Funciones de Fecha y Hora en MS Access

Fecha y Hora actual		
Fecha_Hora	SoloFecha	SoloHora
13/04/2016 22:20:25	13/04/2016	22:20:25

Now, Date y Time

Partes de una fecha u hora						
Fecha_Hora	Año	Mes	Día	Hora	Minuto	Segundo
2/3/2016 12:58:07	2016	3	2	12	58	7

Year, Month y Day
Hour, Minute y Second

Dia del Mes num y letra			
Fecha_Hora	NumeroMe:	NombreMes	NombreMesAbreviado
13/9/2016 12:58:07	9	septiembre	sep

MonthName,
Weekday ,
WeekdayName y

Dia de la Semana num y letra			
Fecha_Hora	NumeroDiaSemana	NombreDiaSemana	NombreDiaSemanaAbreviado
13/9/2016 12:58:07	3	martes	mar

Format

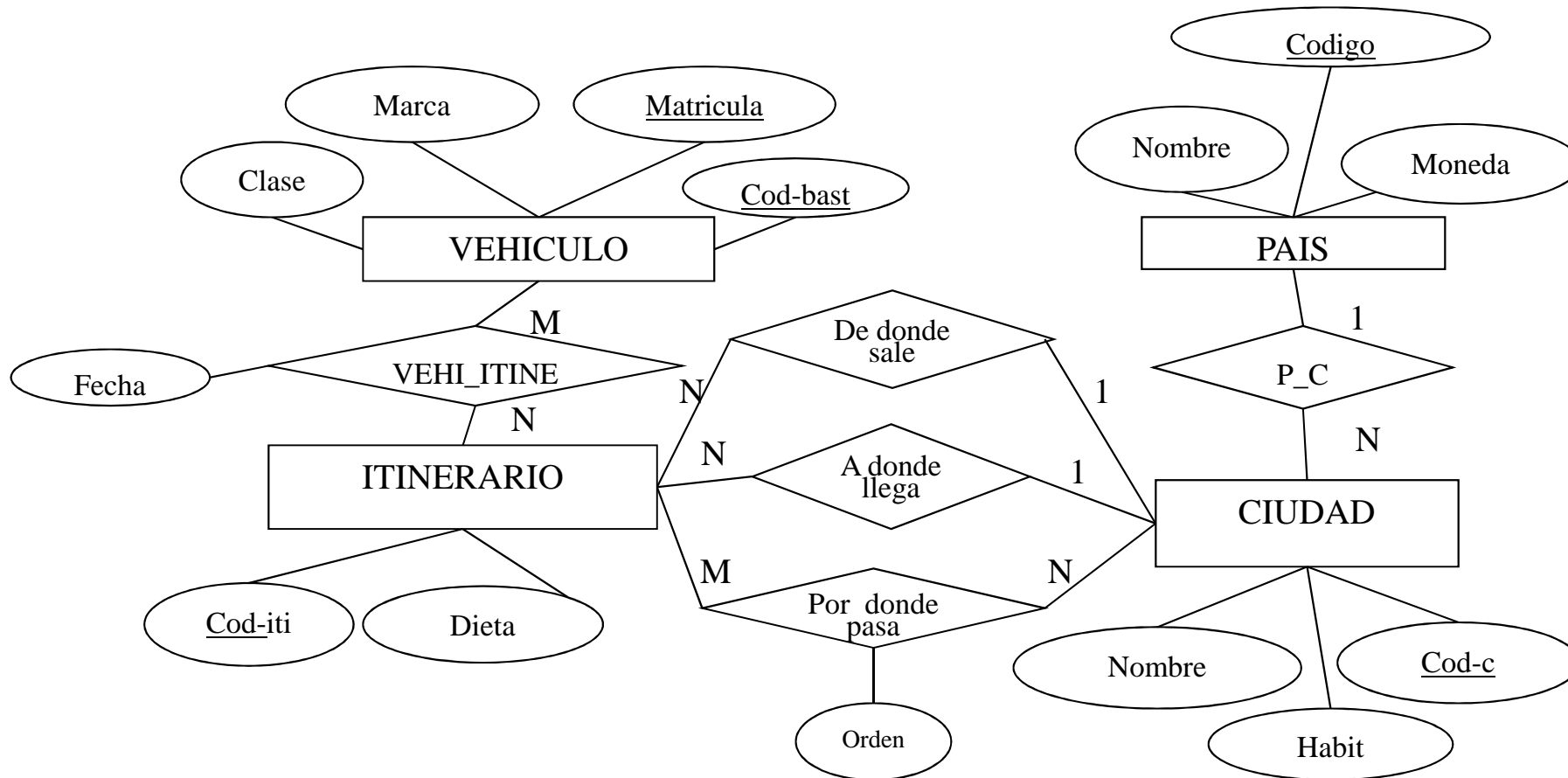
Ejemplo DateDiff		
Fecha1	Fecha2	DiasEntreLasDosFechas
05/04/2016-04-05 00:00:00	13/04/2016 12:42:59	8

DateDiff y
DateAdd

Ejemplos DateAdd			
Fecha_Hora	Mas3Años	Menos5Meses	Mas16Dias
22/7/2003 20:35:44	22/07/2006 20:35:44	22/02/2003 20:35:44	07/08/2003 20:35:44

Ejercicios SQL

Itinerarios



- Hacer el esquema relacional (paso a tablas)


Itinerarios (2)

1. Completa las instrucciones de creación de la BD para el esquema ER anterior.
2. Incluir el itinerario (111, 50000, 95, 141) que sale de Sevilla (95, 'Sevilla', 500000, 34) y llega a Friburgo (141, 'Friburgo', 300000, 49), pasando por Zaragoza (93, 'Zaragoza', 450000, 34), París (16, 'París', 6000000, 33) y Karlstadt (148, 'Karlstadt', 200000, 49). Los países correspondientes son España (34, 'España', 'Euro'), Francia (33, 'Francia', 'Euro') y Alemania (49, 'Alemania', 'Euro'). Ninguno de los datos está en la BD.
3. Todos los itinerarios que pasan por Sevilla (solo pasar, no salir o llegar) se han desviado por Córdoba. Los datos de Córdoba (957, 'Córdoba', 134000, 34) no están en la BD.
4. El vehículo con nº de bastidor 3 ha tenido un accidente y lo han llevado a la chatarra. Se ha cambiado el vehículo a los itinerarios donde figuraba por (345, 'Renault', 'BI-9999-XX', 'Express'), que no está aún en la BD.

Itinerarios (3)

```
CREATE TABLE VEHICULO (  
    Cod-bast          INTEGER NOT NULL,  
    Marca             VARCHAR (15),  
    Matricula         VARCHAR (8) NOT NULL,  
    Clase             VARCHAR (8),  
    PRIMARY KEY (Cod-bast),  
    UNIQUE (Matrícula));
```

```
CREATE TABLE PAIS (  
    Codigo            INTEGER NOT NULL,  
    Nombre            VARCHAR (15) NOT NULL,  
    Moneda            VARCHAR (10),
```



Itinerarios (4)

```
CREATE TABLE ITINERARIO (  
    Cod-iti          INTEGER NOT NULL,  
    Dieta            INTEGER,  
    Cod-dedonde      INTEGER,  
    Cod-adonde       INTEGER,  
    PRIMARY KEY (Cod-iti),  
    FOREIGN KEY (Cod-dedonde) REFERENCES CIUDAD (Cod-c)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (Cod-adonde) REFERENCES CIUDAD (Cod-c)  
        ON DELETE CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE CIUDAD (  
    Cod-c            INTEGER NOT NULL,  
    Nombre           VARCHAR (15) NOT NULL,  
    Habit            INTEGER,  
    Codigo           INTEGER,
```



Itinerarios (5)

```
CREATE TABLE VEHI_ITINE (  
    Cod-bast          INTEGER NOT NULL,  
    Cod-iti           INTEGER NOT NULL,  
    Fecha             DATE DEFAULT CURRENT_DATE,  
    PRIMARY KEY (Cod-bast, Cod-iti),  
    FOREIGN KEY (Cod-bast) REFERENCES VEHICULO (Cod-bast)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (Cod-iti) REFERENCES ITINERARIO (Cod-iti)  
        ON DELETE CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE POR_DONDE_PASA (  
    Cod-iti           INTEGER NOT NULL,  
    Cod-c             INTEGER NOT NULL,  
    Orden             INTEGER,
```

--

```
    FOREIGN KEY (Cod-c) REFERENCES CIUDAD (Cod-c)  
        ON DELETE CASCADE ON UPDATE CASCADE);
```

Ejercicio sobre R.I.

Analizar todas las R.I. violadas por cada operación cuando se ejecuta sobre la BD de la figura siguiente:

- a. INSERT INTO CUENTA VALUES (7, 7, 20000, null, null)
- b. INSERT INTO CLIENTE VALUES (123, Pepi, null, null)
- c. INSERT INTO CUENTA-CLIENTE VALUES (4, null)
- d. INSERT INTO CUENTA VALUES (10, 4, 100000, 2, 3)
- e. DELETE FROM CLIENTE WHERE DNI=999
- f. DELETE FROM CUENTA-CLIENTE WHERE NCta=0 and DNI>000
- g. UPDATE CUENTA-CLIENTE SET NCta=5, DNI=111 WHERE NCta=0 and DNI=111
- h. UPDATE CUENTA-CLIENTE SET NCta=3, DNI=222 WHERE NCta=0
- i. UPDATE CLIENTE SET Nombre='Ataulfo A.' WHERE DNI>888
- j. UPDATE CUENTA SET Banco=2, NSuc=2 WHERE NCta=0
- k. UPDATE CUENTA SET Banco=2 WHERE Banco=1
- l. UPDATE CUENTA SET Banco=null, NSuc=null WHERE Banco=1
- m. UPDATE CUENTA SET Banco=null WHERE NCta=3

Ejercicio sobre R.I. (2)

BANCO

Código	Nombre	Dir
1	BBVA	Gran Vía, 17
2	Santander	Espolón, 1
2101	Kutxa	Garibai, 6
2102	BBK	Salaberria, 3

SUCURSAL

Banco	NSuc	Ciudad
1	1	Bilbao
1	2	S. Sebastián
1	3	Vitoria
2	1	Santander
2	2	S. Sebastián
2101	1	S. Sebastián
2101	2	Pasajes
2102	1	Bilbao
2102	2	Sestao
2102	3	Portugalete

CE

CUENTA-CLIENTE

NCta	DNI
0	111
0	222
0	555
0	000
1	111
2	111
2	222
3	111
3	333
4	111
4	222
4	444
5	111
5	555
6	111
6	222

...

...

6	333
6	666
7	111
7	777
8	111
8	222
8	444
8	888
9	111
9	333
9	999

CE

CE

PRÉSTAMO-CLIENTE

NPres	DNI
0	111
0	000
1	111
2	111
2	222
3	111
3	333
4	111
4	444
5	111
5	555
6	111
6	666
7	111
7	777
8	111
8	888
9	111
9	999

CE

CE

CLIENTE

DNI	Nombre	Dir	Tfno
111	Juan	Legazpi, 1	111111
222	Pedro	Guridi, 5	222222
333	Gaizka	Lardizabal, 1	333333
444	Luisa	Nagusia, 2	444444
555	Ceferina	Elkano, 3	555555
666	Yolanda	Av. Madrid, 9	666666
777	Segismundo	H. Cortés, 9	777777
888	Ataulfo	Cervantes, 1	888888
999	Teodorico	Velázquez, 7	999999
000	Atenea	Sorolla, 13	101010

CUENTA

NCta	Interés	Saldo	Banco	NSuc
1	1	20000	1	1
2	1	30000	1	2
3	1	40000	1	3
4	2	30000	2	1
5	2	40000	2	2
6	3	50000	2101	1
7	3	60000	2101	2
8	3	60000	2102	1
9	4	70000	2102	2
0	4	80000	2102	3

CE

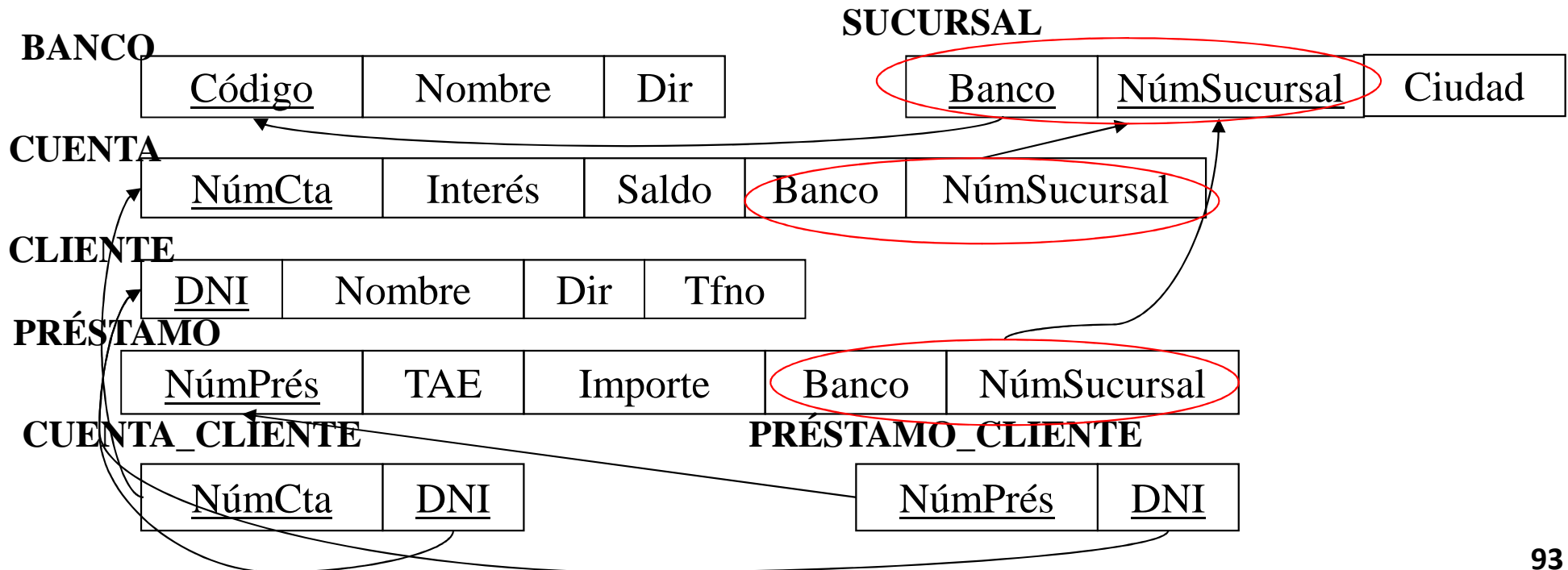
PRÉSTAMO

NPres	TAE	Importe	Banco	NSuc
1	10	200000	1	1
2	10	300000	1	2
3	10	400000	1	3
4	20	300000	2	1
5	20	400000	2	2
6	30	500000	2101	1
7	30	600000	2101	2
8	30	600000	2102	1
9	40	700000	2102	2
0	40	800000	2102	3

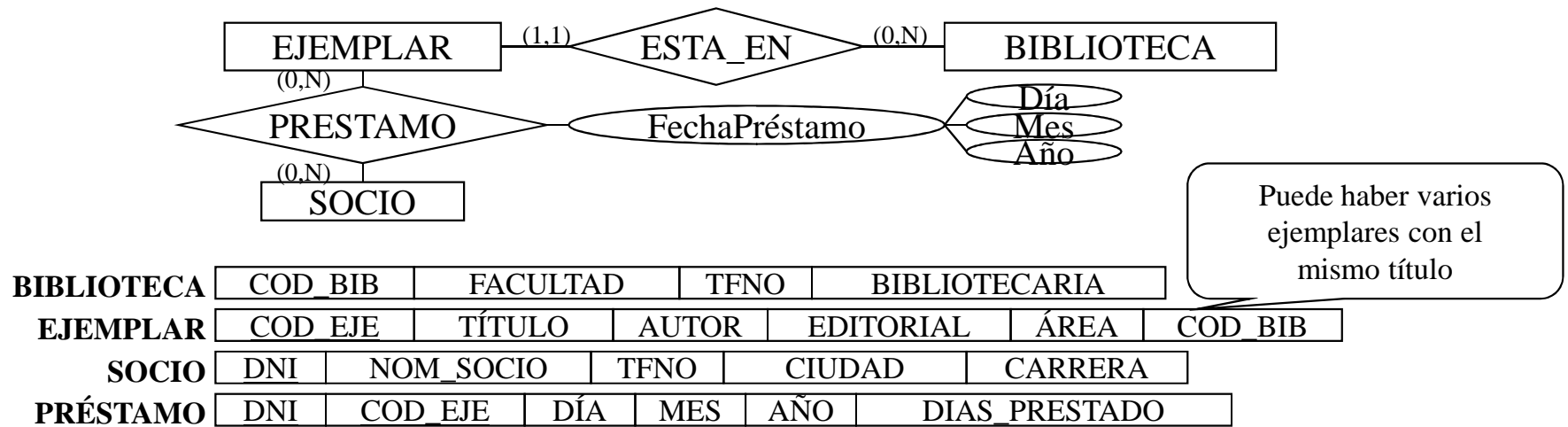
CE

Ejercicios sobre el esquema ER de Elmasri & Navathe Pág. 66 (salvo SUCURSAL.Ciudad)

- a. Datos de todos los bancos.
- b. Direcciones de todos los bancos.
- c. DNI de los clientes que viven en la calle Urbietta.
- d. Diferentes TAE ofrecidos por el banco sito en la Plaza Circular 1.
- e. Datos de los clientes junto a los datos de todas sus cuentas.
- f. Datos de los clientes con cuenta en un banco que tenga sucursal en Andoain.
- g. Insertar la tupla (12, 123) en CUENTA_CLIENTE.
- h. El banco BBK ha cambiado su dirección a 'Arenal 18'. Introducir la modificación en la BD.



Ejercicio de consultas en SQL



Escribe las siguientes consultas en SQL sobre la BD anterior:

- a. Pares de títulos del área de informática sacados en préstamo por el mismo socio.

Posible salida:

título	título
"Fundamentos de BD"	"BD:, ¡Qué gozada!"
"Fundamentos de BD"	"SQL para novatos"
"BD:, ¡Qué gozada!"	"Fundamentos de BD"

- b. Títulos junto al número total de préstamos en cada año entre 1990 y 2000. Sólo aparecerán si se han prestado al menos a diez socios distintos en el mismo año (evitando que un socio sesgue la estadística al tomar un título varias veces).

título	año	total de préstamos
"Fundamentos de BD"	1997	43
"Fundamentos de BD"	2000	59
"BD:, ¡Qué gozada!"	1987	57
"BD:, ¡Qué gozada!"	1993	34

Ejercicio de consultas en SQL (2)

Escribe las siguientes consultas en SQL sobre la BD anterior:

- c. Lista de pares ordenados de bibliotecas que tengan ejemplares de libros de la misma área. Por ejemplo, si las bibliotecas A, B y C tienen libros de informática y las bibliotecas A y C de filosofía, deberán aparecer: <A,B,Informática>, <A,C,Informática>, <A,C,Filosofía>, <B,A,Informática>, <B,C,Informática>, ...

Elmasri & Navathe 8.13 (BD EMPRESA)

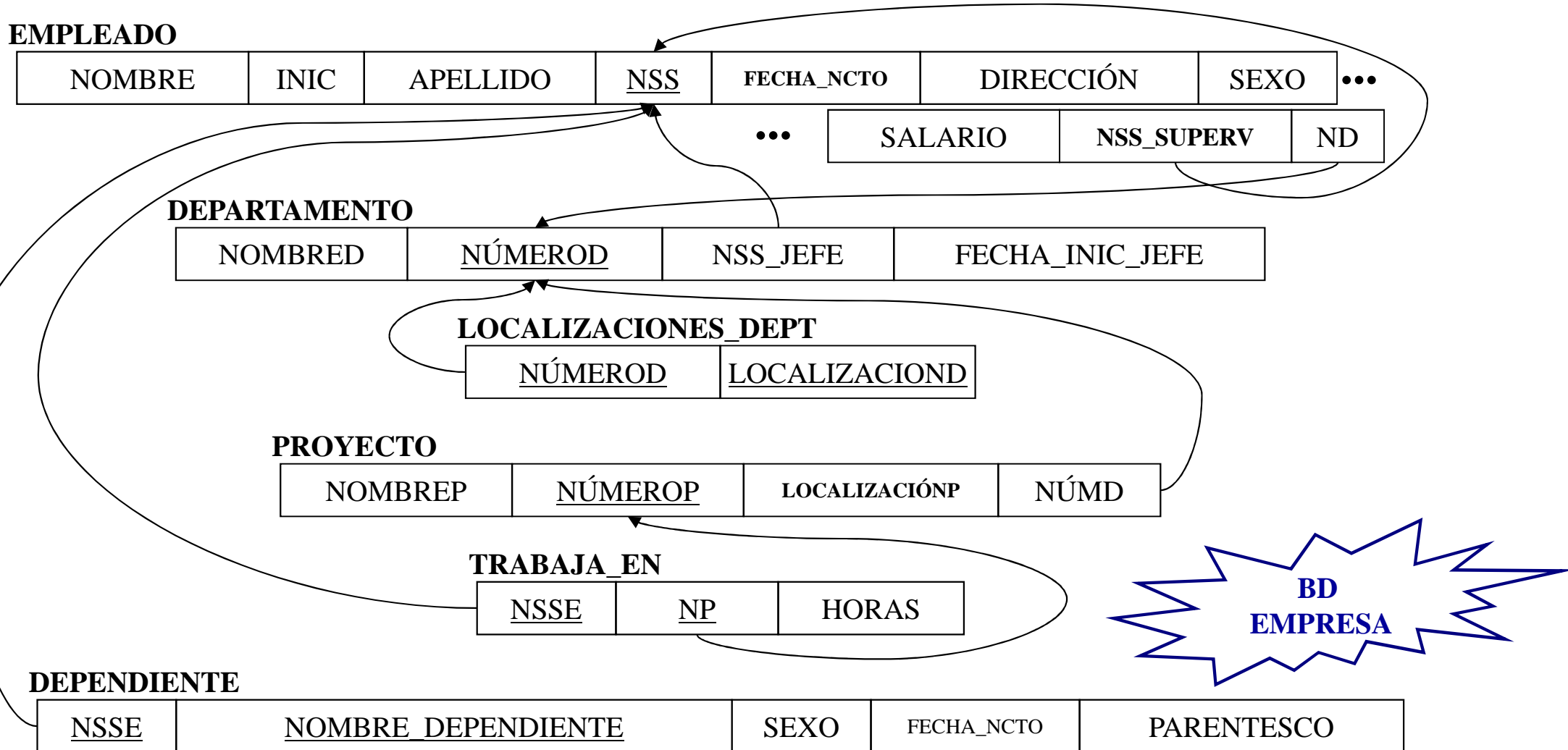







Figura 7.7 Restricciones de integridad referencial representadas en el esquema de la base de datos relacional EMPRESA.

Elmasri & Navathe 8.13 (BD EMPRESA) (2)

Sobre el esquema de la BD EMPRESA, obtener:






- a. Empleados del departamento 5 que trabajan más de 10 horas/semana en el proyecto 'ProductoX'.  INNER JOIN CON MÁS DE 2 TABLAS
- b. Empleados con un dependiente con su mismo nombre de pila.  INNER JOIN CON 2 TABLAS
- c. Empleados cuyo jefe directo es Franklin Wong.  DATOS EN DISTINTAS FILAS DE LA MISMA TABLA
- d. Nombre de cada proyecto junto al número total de horas trabajadas por los empleados en él.  GROUP BY

Modificar la consulta para que solamente salgan aquellos proyectos en los que el total de horas sea mayor que 40.  HAVING



- e. Apellidos y nombres de todos los empleados que trabajan en cada uno de los proyectos, junto al nombre del proyecto en el que trabajan. Los resultados deberán salir ordenados por apellido y dentro de cada apellido por nombre de proyecto.  ORDER BY

Nota: Si lo entendemos como empleados que trabajan en todos los proyectos sería una operación de división y no lo podríamos hacer.

Elmasri & Navathe 8.13 (BD EMPRESA) (3)

- f. Empleados que no trabajan en ningún proyecto.  NOT IN
- g. Nombre de cada departamento junto al salario medio de los empleados asignados al mismo.  GROUP BY
- h. Salario medio de las empleadas de la EMPRESA.  FUNCIÓN AGREGADA SIN GROUP BY
- i1. Nombre y dirección de los empleados que trabajan en algún proyecto situado en Houston pero cuyo departamento controlador no está situado allí.
Nota: la condición para que un departamento no esté en Houston NO es
LOCALIZACIOND < > 'Houston'.
Si usamos esta condición, con los datos que hay en la BD nos saldría el departamento 5, que estaría mal, porque es un departamento que SÍ está en Houston, pero que también está en otros sitios distintos, por eso sale.  NOT IN
- i2. Nombre y dirección de los empleados que trabajan en algún proyecto situado en Houston, pero que el departamento para el que trabajan no está situado allí.
- j. Jefes de departamento sin dependientes.  IN / NOT IN

Elmasri & Navathe 8.16 (BD UNIVERSIDAD)

- a. Nombres de los estudiantes de 1° de la carrera CS
- b. Nombre de los cursos impartidos por el profesor Anderson en los años 1998 y 99.
- c. Para cada sección impartida por el profesor Anderson, obtener el código de curso, semestre, año y número de estudiantes que tomaron la sección.
- d. Nombre y boletín de notas de los estudiantes de 1° de CS. El boletín incluye el nombre y código del curso, los créditos, el semestre, el año y las notas de los cursos aprobados (nota A, B ó C) por el estudiante.
- e. Nombres y departamentos de carrera (especialidad) de los estudiantes calificados con nota 'A' en todos sus cursos. 
- f. Nombres y departamentos de carrera (especialidad) de los estudiantes que NO tengan nota 'A' en ningún curso. 

ALUMNO

Nombre	Codigo Alumno	Año	Especialidad
--------	---------------	-----	--------------

CURSO

Nombre Curso	Codigo Curso	Creditos	Departamento
--------------	--------------	----------	--------------

REQUISITO

Codigo Curso	Codigo Requisito
--------------	------------------

SECCION

Identificador Seccion	Codigo_Curso	Semestre	Año	Profesor
-----------------------	--------------	----------	-----	----------

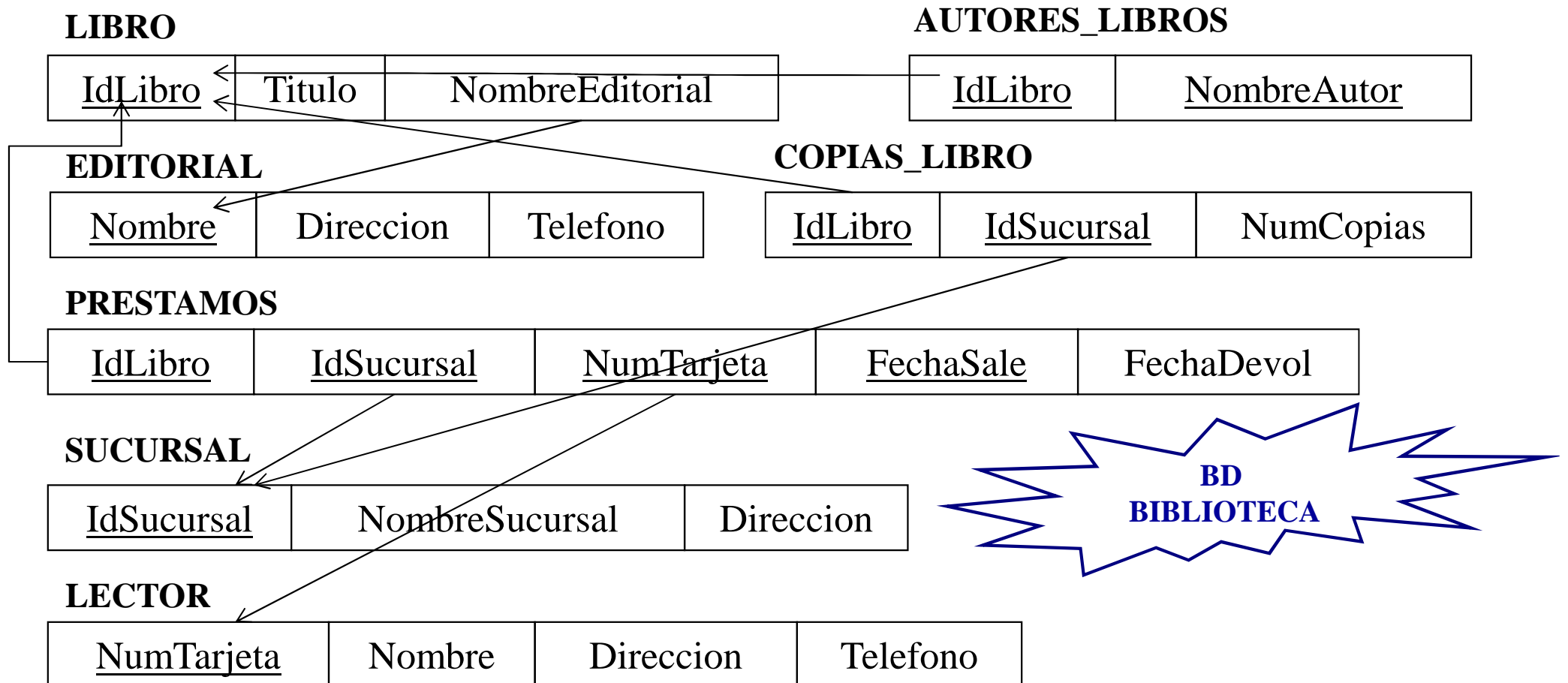
INFORME_CALIFICACIONES

Codigo Alumno	Identificador Seccion	Calificacion
---------------	-----------------------	--------------




Elmasri & Navathe 8.11 (BD BIBLIOTECA)

Dado el esquema de la Base de Datos BIBLIOTECA



Elmasri & Navathe 8.11 (BD BIBLIOTECA) (2)

Obtener los siguientes datos:

- a. Número de copias de *La tribu perdida* en la sucursal de *Sharpstown*.
- b. Identificador de sucursal y núm. de copias del libro *La tribu perdida* en cada una de las sucursales.
- c. Nombres de lectores sin libros en préstamo. 
- d. Título del libro, nombre y dirección del lector de los préstamos de la sucursal de *Sharpstown* con fecha de devolución la de hoy. Insertar previamente los datos que sean necesarios para que haya algún resultado.
- e. Por cada sucursal: Nombre y total de ejemplares de libro en préstamo.
- f. Para los lectores con más de 5 libros en préstamo, obtener su nombre, dirección y nº de libros.
- g. Por cada libro escrito total o parcialmente por Stephen King, obtener su título y el nº de copias en la sucursal 'Central'.

Más ejercicios sobre Elmasri & Navathe 8.11 (BD BIBLIOTECA)

- a. Tiempo medio que duran los préstamos de libros (en conjunto).
- b. Tiempo medio que duraron los préstamos de libros (en conjunto) que se prestaron el año pasado.
- c. Número total de préstamos, número de socios de la biblioteca que han tomado libros en préstamo y número medio de libros que han tomado en préstamo los socios.

Nº medio de libros = Nº total préstamos / Nº lectores distintos

CONSULTA A

```
+-----+
| tiempo_medio |
+-----+
|      52.9231 |
+-----+
1 row in set (0.00 sec)
```

CONSULTA B

```
+-----+
| tiempo_medio_anio_pasado |
+-----+
|              4.5000 |
+-----+
1 row in set (0.00 sec)
```

CONSULTA C

```
+-----+-----+-----+
| total | num_socios | media |
+-----+-----+-----+
|    15 |          2 | 7.5000 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Más ejercicios sobre Elmasri & Navathe 8.11 (BD BIBLIOTECA)

- d. Número medio de copias por libro de la biblioteca “central” (en conjunto).
- e. Número de préstamos de cada socio (basta indicar su número de tarjeta) que ha llevado préstamos.
- f. Número de préstamos por socio para todos los socios (incluyendo los que no han llevado préstamos).

CONSULTA D

```
+-----+
| media_central |
+-----+
|          10.6250 |
+-----+
1 row in set (0.00 sec)
```

CONSULTA E

```
+-----+-----+
| NumTarjeta | numero_de_prestamos |
+-----+-----+
|          10000 |                3 |
|          10001 |                12 |
+-----+-----+
2 rows in set (0.00 sec)
```

CONSULTA F

```
+-----+-----+
| NumTarjeta | numero_de_prestamos |
+-----+-----+
|          10000 |                3 |
|          10001 |                12 |
|          10002 |                0 |
+-----+-----+
3 rows in set (0.00 sec)
```

Más ejercicios sobre Elmasri & Navathe 8.11 (BD BIBLIOTECA)

- g. Identificador de libro junto al número de veces que ha sido prestado.
- h. Identificadores de cada libro y sucursal junto al número de veces que ha sido prestado el libro en la sucursal.

CONSULTA G

IdLibro	num_veces_prestado
101	1
102	3
103	6
104	1
105	1
106	1
108	2

7 rows in set (0.02 sec)

CONSULTA H

IdLibro	IdSucursal	num_veces_prestado
101	2	1
102	2	3
103	2	3
103	3	3
104	2	1
105	2	1
106	3	1
108	2	2

8 rows in set (0.00 sec)

Más ejercicios sobre Elmasri & Navathe 8.11 (BD BIBLIOTECA)

- i. Para cada libro: identificador de libro y número de autores.
- j. Para cada libro: título y número de autores.

CONSULTA I

IdLibro	num_autores
100	2
101	1
102	1
103	1
104	1
105	1
106	1
107	1
108	1
109	1

10 rows in set (0.00 sec)

CONSULTA J

Titulo	num_autores
Sistemas de Bases de Datos	2
El Resplandor	1
It	1
Carrie	1
La Torre Oscura I	1
La Torre Oscura II	1
La Torre Oscura III	1
La Torre Oscura IV	1
El Codigo Da Vinci	1
Angeles y Demonios	1

10 rows in set (0.00 sec)

Más ejercicios sobre Elmasri & Navathe 8.11 (BD BIBLIOTECA)

- k. Para cada día del mes pasado en el que hubiera préstamos, número de libros distintos que se prestaron ese día.

```
CONSULTA K
+-----+-----+
| FechaSale          | libros_distintos_prestados_mes_pasado |
+-----+-----+
| 2020-01-23 12:51:09 | 1 |
| 2020-01-25 12:51:09 | 1 |
+-----+-----+
2 rows in set (0.00 sec)
```

(Nota: En este ejemplo de resultado, se supone que estamos en febrero de 2020.)

Consultas sobre la BD BUQUES

Resolver las siguientes consultas sobre la BD BUQUES

- 1.1. Nombres de todos los buques ordenados alfabéticamente .
 - 1.2. Nombres de todos los buques cuyo nombre empieza por A ordenados alfabéticamente.
 - 1.3. Nombres de todos los buques de tipo 20 y cuyo nombre empieza por A ordenados alfabéticamente.
 - 1.4. Nombres de todos los buques cuyo tipo es 10, 20 ó 40 ordenados alfabéticamente.
 - 1.5. Nombres de todos los buques cuyo nombre comience por las letras que están entre la A y la M ordenados alfabéticamente en descendente.
-
- 2.1. Visitas realizadas después del 20/11/2000.
 - 2.2. Nombres de los buques que han hecho alguna visita.
 - 2.3. Número total de visitas.
 - 2.4. Nombres de los buques que han hecho más de dos visitas, junto con el número de visitas realizadas.

Consultas sobre la BD BUQUES (2)

1. Número de países visitados por cada buque que haya visitado algún puerto.
2. N° de veces que han ido a Cádiz cada uno de los buques cuyo tonelaje es mayor que 1000.

Difíciles

3. Nombres de aquellos buques que, en el mismo día, hayan salido de un puerto y llegado a otro puerto situado en un mar distinto al del puerto de partida, junto al nombre de los mares visitados.
4. Nombres de aquellos buques que hayan ido, en menos de 5 días, desde el puerto de Santander al de Cádiz, habiendo hecho escala en el puerto de Vigo.
5. Información de todas las visitas de buques con puerto de origen Detroit junto a la información de su visita anterior al mismo puerto de esa visita.