

Anexo al tema 4. Repaso de álgebra relacional y SQL

El modelo de datos relacional, las restricciones relacionales y el álgebra relacional Elmasri/Navathe 02

El estándar de las Bases de Datos Relacionales Elmasri/Navathe 02

- Modelo relacional: conceptos, restricciones, operaciones de actualización y operaciones del álgebra
- Revisión de SQL

Conceptos

- BD: colección de relaciones
- **Relación:**

ALUMNO

Nombre	CódigoAlumno	Año	Especialidad
Smith	17	1	CS
Brown	8	2	CS

Valores columna mismo dominio
Fila= Tupla
Columna = Atributo

- **Relación = conjunto de tuplas** (no tienen sentido tuplas duplicadas)
- Dominios atómicos: ni compuestos ni multivaluados
- **Esquema de relación $R(A_1, \dots, A_n)$:** intensión
- **Relación (o estado de relación) r ó $r(R)$:** extensión
 $r = \{t_1, t_2, \dots, t_m\}$: conjunto de tuplas
 Cada valor v_i de un t_i $1 \leq i \leq n$ es:
 - un elemento de $\text{dom}(A_i)$
 - o un valor nulo
- **Orden entre las tuplas:** no se considera
- **Orden en los valores de una tupla:** es una lista ordenada de n valores. Lo importante es la correspondencia atributo-valor

Restricciones relacionales

- **De dominio:** valor atómico de un tipo
- **De clave:** atributo(s) que identifica(n) unívocamente a las tuplas.
 - Superclave y clave
 - Clave candidata y clave primaria
- **Integridad de entidades:** ninguna clave primaria puede contener el valor nulo
- **Integridad referencial:**
 - Una tupla que referencia a otra (de la misma u otra relación), debe referirse a una tupla **existente** en dicha relación
 - Se hace referencia a otra tupla mediante una **clave extranjera** (foránea, externa). Conjunto de atributos no vacío. Puede contener valor nulo.

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS	NSS_SUPERV	ND
John	B	Smith	123456789	333445555	5
Franklin	T	Wong	333445555	888665555	5
Alicia	J	Zelaya	999887777	987654321	4
Jennifer	S	Wallace	987654321	888665555	4
Ramesh	K	Narayan	666884444	333445555	5
Joyce	A	English	453453453	333445555	5
Ahmad	V	Jabbar	987987987	987654321	4
James	E	Borg	888665555	nulo	1

Clave extranjera
Clave extranjera

DEPARTAMENTO

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

Esquema de la BD "EMPRESA" con restricciones de integridad referencial

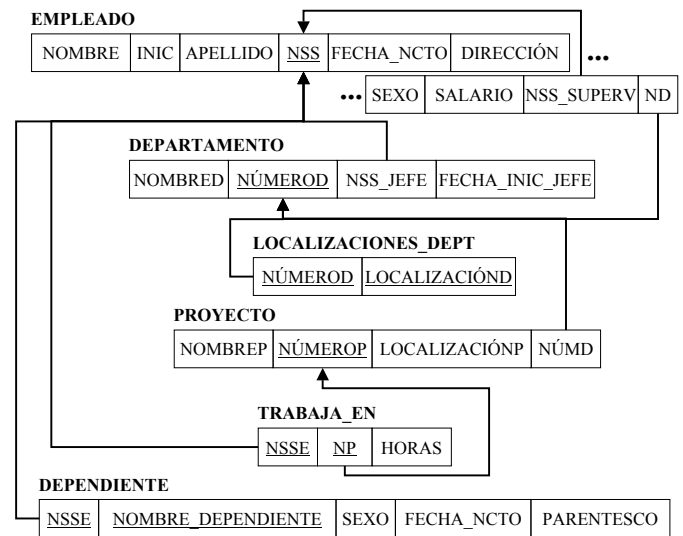


Figura 7.7 restricciones de integridad referencial representadas en el esquema de la base de datos relacional EMPRESA

Estado de la BD relacional “EMPRESA”

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS	FECHA_NCTO	DIRECCIÓN
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX

TRABAJA_EN

NSSE	NP	HORAS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	nulo

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

Figura 7.6 (1ª parte)
Un posible estado de la base de datos relacional del esquema EMPRESA

DEPARTAMENTO

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

Estado de la BD relacional “EMPRESA” (cont)

LOCALIZACIONES_DEPT

NÚMEROD	LOCALIZACIÓN
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

PROYECTO

NOMBREP	NÚMEROP	LOCALIZACIÓNP	NÚMD
ProductoX	1	Bellaire	5
ProductoY	2	Sugarland	5
ProductoZ	3	Houston	5
Automatización	10	Stafford	4
Reorganización	20	Houston	1
Nuevos beneficios	30	Stafford	4

DEPENDIENTE

NSSE	NOMBRE_DEPENDIENTE	SEXO	FECHA_NCTO	PARENTESCO
333445555	Alice	M	1986-04-05	HIJA
333445555	Theodore	H	1983-10-25	HIJO
333445555	Joy	M	1958-05-03	ESPOSA
987654321	Abner	H	1942-02-28	ESPOSO
123456789	Michael	H	1988-01-04	HIJO
123456789	Alice	M	1988-12-30	HIJA
123456789	Elizabeth	M	1967-05-05	ESPOSA

Figura 7.6 (2ª parte) Un posible estado de la base de datos relacional del esquema EMPRESA

Operaciones de actualización y violación de las RI

- Operaciones de actualización:
 - Insertar
 - Eliminar
 - Actualizar (modificar)
- Cuando se aplican no deben violar ninguna RI
- Insertar y actualizar pueden violar los 4 tipos de RI
- Eliminar sólo puede violar la I. Referencial
- En SQL se pueden definir acciones asociadas a la violación de RI (ejemplo. ON UPDATE CASCADE)

Álgebra relacional

- Operaciones para manipular **relaciones enteras**
- Permiten especificar consultas (recuperación de datos)
- El resultado de una consulta es otra relación
- Operaciones específicas del álgebra relacional:
 - SELECCIONAR
 - PROYECTAR
 - REUNIÓN (JOIN)

} Unarios

} Binario
- Operaciones de teoría de conjuntos:
 - UNIÓN
 - INTERSECCIÓN
 - DIFERENCIA
 - PRODUCTO CARTESIANO

} Binarios
- Otras operaciones:
 - DIVISIÓN
 - FUNCIONES AGREGADAS Y DE AGRUPACIÓN
 - Etc.

Seleccionar: σ

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS
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
James	E	Borg	888665555

...

NSS_SUPERV	ND
333445555	5
888665555	5
987654321	4
888665555	4
333445555	5
333445555	5
987654321	4
nulo	1

$\sigma = \text{sigma}$

$\sigma_{ND=4}(\text{EMPLEADO})$

NOMBRE	INIC	APELLIDO	NSS
Alicia	J	Zelaya	999887777
Jennifer	S	Wallace	987654321
Ahmad	V	Jabbar	987987987

...

NSS_SUPERV	ND
987654321	4
888665555	4
987654321	4

- Selecciona un subconjunto de **filas** (tuplas) de una relación
- Las que satisfacen una condición
- Condición: $\{=, <, \leq, >, \geq, \neq\}$ **Y, O, NO**
- El resultado es otra relación
- Conmutativa: $\sigma_{\langle \text{COND1} \rangle}(\sigma_{\langle \text{COND2} \rangle}(R)) = \sigma_{\langle \text{COND2} \rangle}(\sigma_{\langle \text{COND1} \rangle}(R))$
- $\sigma_{\langle \text{COND1} \rangle}(\sigma_{\langle \text{COND2} \rangle}(R)) = \sigma_{\langle \text{COND1} \rangle \text{ Y } \langle \text{COND2} \rangle}(R)$

Proyectar: π

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS
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
James	E	Borg	888665555

...

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

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

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

El (M, 25.000) duplicado se ha eliminado

Fig 7.8 (c)

- Selecciona las **columnas** especificadas de una relación (desechando el resto de columnas)
- El resultado es otra relación
- Eliminación de duplicados
- $\pi_{\langle \text{LISTA1} \rangle}(\pi_{\langle \text{LISTA2} \rangle}(R)) = \pi_{\langle \text{LISTA1} \rangle}(R)$
- No es conmutativa

Renombrar y resultados intermedios

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS
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
James	E	Borg	888665555

...

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

Resultado intermedio

$\text{TEMP} \leftarrow \sigma_{ND=5}(\text{EMPLEADO})$

NOMBRE	INIC	APELLIDO	NSS
John	B	Smith	123456789
Franklin	T	Wong	333445555
Ramesh	K	Narayan	666884444
Joyce	A	English	453453453

...

SALARIO	NSS_SUPERV	ND
30.000	333445555	5
40.000	888665555	5
38.000	333445555	5
25.000	333445555	5

Renombre de atributos

$R(\text{NOMBRE_PILA}, \text{PRIMER_APELL}, \text{SALARIO}) \leftarrow \pi_{\text{NOMBRE, APELLIDO, SALARIO}}(\text{TEMP})$

NOMBRE_PILA	PRIMER_APELL	SALARIO
John	Smith	30.000
Franklin	Wong	40.000
Ramesh	Narayan	38.000
Joyce	English	25.000

...

Alternativa sin resultados intermedios o renombre de atributos

$\pi_{\text{NOMBRE, APELLIDO, SALARIO}}(\sigma_{ND=5}(\text{EMPLEADO}))$

Unión, intersección y diferencia

ALUMNO

NOM	APEL
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Bárbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

PROFESOR

NOMBRE	APELLIDO
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

ALUMNO \cup PROFESOR

NOM	APEL
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Bárbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

ALUMNO \cap PROFESOR

NOM	APEL
Susan	Yao
Ramesh	Shah

PROFESOR $-$ ALUMNO

NOMBRE	APELLIDO
John	Smith
Ricardo	Browne
Francis	Johnson

- **Compatibilidad con la unión:** ambas relaciones deben tener el mismo número de atributos y cada par de atributos correspondientes pertenecer al mismo dominio

$$\begin{array}{ccc}
 \text{conmutativas} & \text{asociativas} & \text{no conmutativa} \\
 R \cup S = S \cup R & (R \cup S) \cup T = R \cup (S \cup T) & R - S \neq S - R \\
 R \cap S = S \cap R & (R \cap S) \cap T = R \cap (S \cap T) &
 \end{array}$$

Producto Cartesiano: \times (CROSSJOIN)

PR_STAF	DP_STAF	
NOMBREP	NOMBRED	NÚMEROD
Automatización	Administración	4
Nuevos beneficios	Mantenimiento	8

$R \leftarrow PR_STAF \times DP_STAF$

NOMBREP	NOMBRED	NÚMEROD
Automatización	Administración	4
Automatización	Mantenimiento	8
Nuevos beneficios	Administración	4
Nuevos beneficios	Mantenimiento	8

- Relaciones: no han de ser compatibles con la unión
- $R(A_1, \dots, A_n) \times S(B_1, \dots, B_m) = Q(A_1, \dots, A_n, B_1, \dots, B_m)$ donde R tiene n' tuplas y S m' tuplas
 - Q tiene n' * m' tuplas
 - Q consta de todas las combinaciones de cada tupla de R seguida de otra de S

Reunión (JOIN): $| \times |$

$EMP \leftarrow \pi_{\text{APELLIDO, NSS, ND}}(EMPLEADO)$

APELLIDO	NSS	ND
Smith	123456789	5
Wong	333445555	5
Zelaya	999887777	4
Wallace	987654321	4
Narayan	666884444	5
English	453453453	5
Jabbar	987987987	4
Borg	888665555	1

$DPT \leftarrow \pi_{\text{NOMBRED, NSS_JEFE}}(\text{DEPARTAMENTO})$

NOMBRED	NSS_JEFE
Investigación	333445555
Administración	987654321
Dirección	888665555

$JEFE_DTO \leftarrow | \times |_{\text{NSS_JEFE=NSS}} EMP$

NOMBRED	NSS_JEFE	APELLIDO	NSS	ND
Investigación	333445555	Wong	333445555	5
Administración	987654321	Wallace	987654321	4
Dirección	888665555	Borg	888665555	1

- Combina *tuplas relacionadas* de 2 relaciones (o de la misma)
- Operación muy importante para cualquier BDR
- Permite procesar vínculos entre relaciones

Reunión (JOIN) (2)

- Relaciones: no han de ser compatibles con la unión
- $R(A_1, \dots, A_n) | \times |_{\text{COND}} S(B_1, \dots, B_m) = Q(A_1, \dots, A_n, B_1, \dots, B_m)$ donde R tiene n' tuplas y S m' tuplas
 - Q tiene un máximo de n' * m' tuplas
 - Q consta de todas las combinaciones de cada tupla de R seguida de otra de S, **que satisfagan la condición de reunión "COND"**
 - R y Q pueden ser el mismo conjunto
 - Si ninguna combinación cumple la condición "COND" el resultado es una relación vacía (sin tuplas)
- Condición "COND":
 - Se evalúa para cada combinación de tuplas
 - tiene la forma: <subcondición> Y <subcondición> Y ... Y <subcondición>
 - Cada subcondición tiene la forma: $A_i \theta B_j$ donde $A_i \in R$ y $B_j \in S$ y $\theta \in \{=, <, \leq, >, \geq, \neq\}$

Tipos de reunión (JOIN)

- **Reunión Theta:**
 - Cualquier reunión
 - Las tuplas cuyo atributo de reunión sea nulo NO aparecen en el resultado
- **Equirreunión (equijoin):**
 - sólo comparaciones de igualdad en "COND"
 - El resultado siempre tiene pares de atributos con valores idénticos en todas las tuplas
- **Reunión natural (join natural):**
 - equirreunión seguida de la eliminación de atributos superfluos.
 - "COND" implícita: igualdades de todos los pares de atributos de igual nombre
 - Exige algún par de atributos de igual nombre
 - Se identifica con *

Ejemplo de reunión natural: *

$PRY \leftarrow \pi_{\text{NOMBREP, NÚMD}}(\text{PROYECTO})$

NOMBREP	NÚMD
ProductoX	5
ProductoY	5
ProductoZ	5
Automatización	4
Reorganización	1
Nuevos beneficios	4

$DPT(\text{NOMBRED, NÚMD})$

$\leftarrow \pi_{\text{NOMBRED, NÚMEROD}}(\text{DEPARTAMENTO})$

NOMBRED	NÚMD
Investigación	5
Administración	4
Dirección	1

Tiene que haber al menos un par de atributos con el mismo nombre

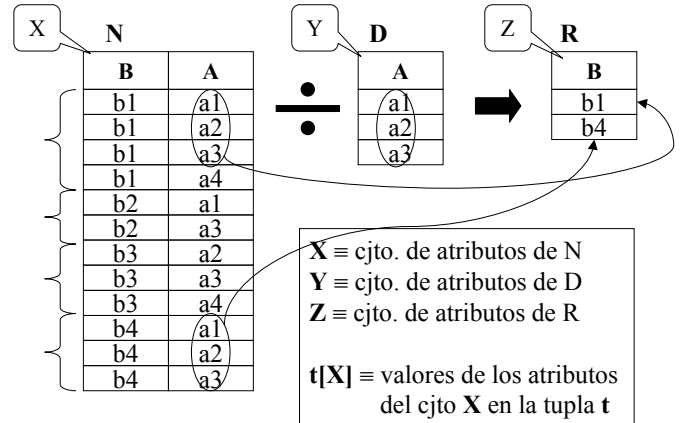
Desaparece un NÚMD

$\text{DEPTO_PROY} \leftarrow PRY * DPT$

NOMBREP	NÚMD	NOMBRED
ProductoX	5	Investigación
ProductoY	5	Investigación
ProductoZ	5	Investigación
Automatización	4	Administración
Reorganización	1	Dirección
Nuevos beneficios	4	Administración

División: ÷

Ejemplo: $R \leftarrow N \div D$



- Para cada atributo de D debe haber otro en N de igual nombre: $Y \subseteq X$
- El resultado tiene los atributos de N que NO están en D: $Z = X - Y$
- Son tuplas del resultado, $t \in R$, las que cumplen:
 - La combinación de valores de t está en alguna tupla de N
 - Esa combinación de valores de t se encuentra en tuplas de N junto todas y cada una de las combinaciones de valores de las tuplas de D
- Es decir, $t \in R$ si y solo si:

$$\forall t_D \in D \exists t_N \in N \text{ tal que } (t = t_N[Z] \wedge t_D = t_N[Y])$$

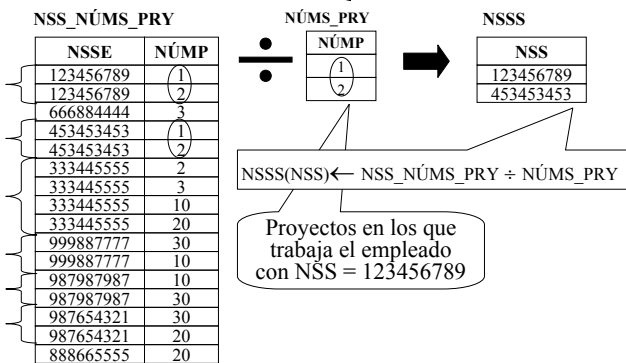
División: ÷ (2)

- La división sirve para construir consultas como la siguiente:

Obtener el NSS de los empleados que trabajan **en todos y cada uno** de los proyectos en los que trabaja el empleado con NSS=123456789.

$\text{NSS_NÚMS_PRY} \leftarrow \pi_{\text{NSS, NÚMP}}(\text{TRABAJA_EN})$

$\text{NÚMS_PRY} \leftarrow \pi_{\text{NÚMP}}(\sigma_{\text{NSS}=123456789}(\text{TRABAJA_EN}))$



Todas las parejas NSS de empleado con un proyecto en el que trabaja

Funciones agregadas (SUMA, PROMEDIO, MÁXIMO, MÍNIMO y CUENTA) y de agrupación

EMP

NSS	SALARIO
123456789	30.000
333445555	40.000
999887777	25.000
987654321	43.000
666884444	38.000
453453453	25.000
987987987	25.000
888665555	55.000

- En general, los valores duplicados también se consideran en los cálculos
- En SQL se puede poner para ello, por ejemplo: COUNT(DISTINCT SALARIO)

$R1 \leftarrow \int_{\text{CUENTA NSS, PROMEDIO SALARIO}}(\text{EMP})$

CUENTA_NSS	PROMEDIO_SALARIO
8	35125

$R2 \leftarrow \int_{\text{PROMEDIO SALARIO}}(\text{EMP})$

PROMEDIO_SALARIO
35125

El resultado es una relación, aunque se trate de una sola tupla con un solo atributo

$R3 \leftarrow \int_{\text{ND}} \int_{\text{CUENTA NSS, PROMEDIO SALARIO}}(\text{EMP})$

ND	CUENTA_NSS	PROMEDIO_SALARIO
5	4	33250
4	3	31000
1	1	55000

Funciones agregadas y de agrupación y valor nulo

EMP

NSS	NOM	DPTO
11	Alfredo	LSI
22	Ana	LSI
33	Juan	ATC
44	Federico	nulo
55	Ana	LSI

$R1 \leftarrow \text{DPTO } \mathcal{F} \text{ CUENTA NOM(EMP)}$

DPTO	CUENTA_NOM
LSI	3
ATC	1
nulo	1

NO cuenta los valores **diferentes** en el campo NOM para un mismo DPTO, sino cuántas tuplas tienen valor asignado en NOM para un mismo valor de DPTO

Si considera el **nulo** como valor de agrupación

$R2 \leftarrow \text{NOM } \mathcal{F} \text{ CUENTA DPTO(EMP)}$

NOM	CUENTA_DPTO
Alfredo	1
Ana	2
Juan	1
Federico	0

NO considera en la cuenta (o en el promedio, suma, ...) los valores **nu**los que pueda haber

Permite especificar una combinación de atributos de agrupación

$R3 \leftarrow \text{NOM, DPTO } \mathcal{F} \text{ CUENTA NSS(EMP)}$

NOM	DPTO	CUENTA_NSS
Alfredo	LSI	1
Ana	LSI	2
Juan	ATC	1
Federico	nulo	1

Reunión externa

$EMP \leftarrow \pi_{\text{APELLIDO, NSS}}(\text{EM- PLEADO})$

$DEP \leftarrow \pi_{\text{NOMBRED, NSS_JEFE}}(\text{DE- PARTAMENTO})$

APELLIDO	NSS
Smith	123456789
Wong	333445555
Zelaya	999887777
Wallace	987654321
Narayan	666884444
English	453453453
Jabbar	987987987
Borg	888665555

NOMBRED	NSS_JEFE
Investigación	333445555
Administración	987654321
Dirección	888665555

Reunión externa izquierda $R \bowtie S$

$R \leftarrow \leftarrow \pi_{\text{APELLIDO, NOMBRED}}(\text{EMP } \bowtie_{\text{NSS=NSS_JEFE}} \text{DEP})$

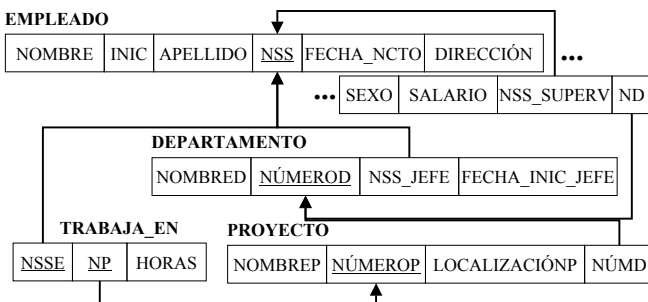
APELLIDO	NOMBRED
Smith	nulo
Wong	Investigación
Zelaya	nulo
Wallace	Administración
Narayan	nulo
English	nulo
Jabbar	nulo
Borg	Dirección

Conserva todas las tuplas de EMP aunque sea rellenando todos los campos correspondientes a DEP con valores nulos

Reunión externa derecha $R \bowtie S$

Reunión externa completa $R \bowtie S$

Ejemplos de consultas con álgebra relacional



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

$\text{PRYS_STAFF} \leftarrow \sigma_{\text{LOCALIZACIÓNP}='Stafford'}(\text{PROYECTO})$

$\text{DPT_CONTR} \leftarrow \text{PRYS_STAFF } \bowtie \text{ NÚMD=NÚMEROD DEPARTAMENTO}$

$\text{JEFE_DP_PRY} \leftarrow \text{DPT_CONTR } \bowtie \text{ NSS_JEFE=NSS EMPLEADO}$

$\text{RESULTADO} \leftarrow \pi_{\text{NÚMEROP, NÚMD, APELLIDO, DIRECCIÓN, FECHA_NCTO}}(\text{JEFE_DP_PRY})$

- Nombre de los empleados que trabajan en todos los proyectos del departamento 5.

$\text{PRY_DP5 (NÚMP)} \leftarrow \pi_{\text{NÚMEROP}}(\sigma_{\text{NÚMD}=5}(\text{PROYECTO}))$

$\text{EMP_PRY(NSS, NÚMP)} \leftarrow \pi_{\text{NSSE, NP}}(\text{TRABAJA_EN})$

$\text{NSSS_EMP} \leftarrow \text{EMP_PRY } \div \text{ PRY_DP5}$

$\text{RESULTADO} \leftarrow \pi_{\text{APELLIDO, NOMBRE}}(\text{NSSS_EMP } * \text{ EMPLEADO})$

Revisión de SQL (Structured Query Language)



- Álgebra relacional → orden de las operaciones
- SQL: lenguaje **declarativo** →
 - Se indica cuál es el resultado esperado
 - Permite que el SGBD seleccione las operaciones y el orden más adecuados para obtener el resultado: **optimización**
- SGBD comerciales → son variantes de SQL
- Estándares SQL: **SQL1** 1986, **SQL2** 1992 y **SQL3** (extenderá SQL2 con conceptos recientes de BD y OO)
- SQL:
 - Lenguaje de Definición de Datos (LDD)
 - Lenguaje de Manipulación de Datos (LMD): consulta y actualización
 - Definición de vistas
 - Especificación de seguridad y autorización
 - Definición de restricciones de integridad
 - Especificación de control de transacciones
 - Reglas para inclusión en lenguajes (C, PASCAL,...)

Esquema y catálogo en SQL2

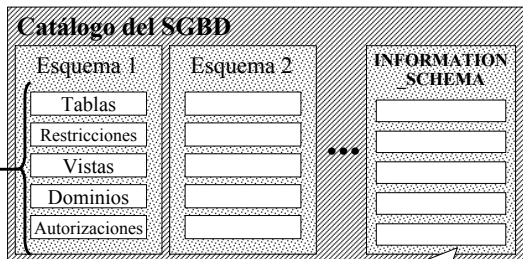


- **Esquema de BD:** el término se incorporó en SQL2

```
CREATE SCHEMA Nombre [AUTHORIZATION Usuario];
```

Cuenta propietaria del esquema

- **Catálogo del SGBD:** colección de esquemas en un entorno SQL

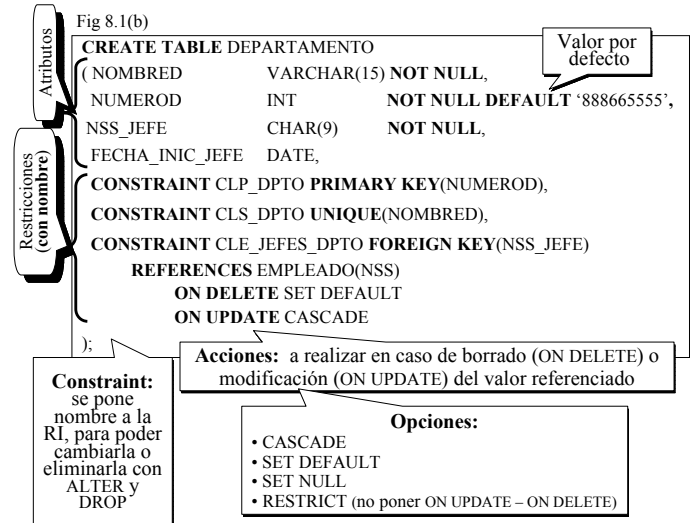
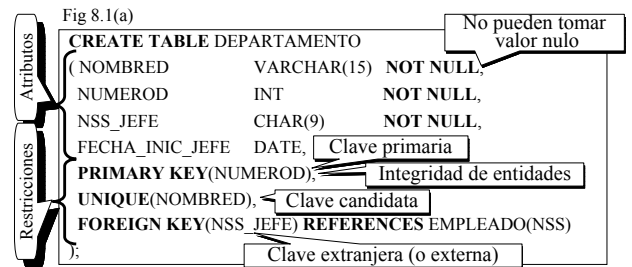


Elementos de un esquema

Esquema especial que da a los usuarios autorizados información de todos los esquemas del catálogo del SGBD

- Restricciones de integridad (RI): sólo entre relaciones del mismo catálogo del SGBD
- Esquemas del catálogo del SGBD: pueden compartir elementos (por ejemplo dominios)

Instrucción CREATE TABLE



CREATE TABLE: especificación de restricciones y valores por omisión



- Definición de atributos:
 - **NOT NULL:** no se permite que el atributo tome valor nulo
 - **DEFAULT un-valor:** se indica qué valor tomará el atributo si no se le asigna nada. Si no se indica DEFAULT, el valor por defecto es el valor nulo
- Especificación de restricciones (tras las definiciones de atributo):
 - **PRIMARY KEY:** clave primaria
 - **UNIQUE:** clave candidata
 - **FOREIGN KEY:** clave extranjera
 - Se puede **calificar** con:
 - **ON DELETE:** en caso de borrarse la tupla a la que se hace referencia con un valor de la clave extranjera
 - **ON UPDATE:** en caso de modificarse el valor de clave primaria a la que se hace referencia con el valor de clave extranjera

```
CREATE TABLE DEPARTAMENTO
(NOMBRED VARCHAR(15) NOT NULL,
NUMEROD INT NOT NULL DEFAULT 1,
NSS_JEFE CHAR(9) NOT NULL,
FECHA_INIC_JEFE DATE,
PRIMARY KEY(NUMEROD),
UNIQUE(NOMBRED),
FOREIGN KEY(NSS_JEFE) REFERENCES EMPLEADO(NSS)
ON UPDATE CASCADE ON DELETE SET NULL);
```

CREATE TABLE: especificación de restricciones y valores por omisión (2)



FOREIGN KEY (clave extranjera):

- **Acciones** posibles (en ON DELETE / ON UPDATE):

- **SET NULL:** el valor de clave extranjera en cuestión se sustituye por el valor nulo
- **SET DEFAULT:** el valor de clave extranjera en cuestión se sustituye por el valor por defecto del atributo clave extranjera
- **CASCADE:** en caso de ON DELETE se borran las tuplas que hacen referencia a la tupla que desaparece.

En caso de ON UPDATE se aplica la misma modificación en las claves extranjeras que la realizada sobre la clave primaria a la que hacen referencia.

CASCADE es adecuada para:

- vínculos (TRABAJA_EN)
- atributos multivaluados (LOCALIZACIONES_DEPT)
- tipos de entidad débiles (DEPENDIENTE)

- **RESTRICT** (cuando NO se pone ON DELETE / ON UPDATE): **impide** el borrado (si falta ON DELETE) o la modificación (si falta ON UPDATE) de cualquier tupla referenciada desde un valor de la clave extranjera en cuestión.

```
CREATE TABLE DEPARTAMENTO
(NOMBRED VARCHAR(15) NOT NULL,
NUMEROD INT NOT NULL DEFAULT 1,
NSS_JEFE CHAR(9) NOT NULL,
FECHA_INIC_JEFE DATE,
PRIMARY KEY(NUMEROD),
UNIQUE(NOMBRED),
FOREIGN KEY(NSS_JEFE) REFERENCES EMPLEADO(NSS)
ON UPDATE CASCADE ON DELETE SET NULL);
```

Borrar esquemas (DROP SCHEMA) y borrar tablas (DROP TABLE)



Borrar un esquema completo:

```
DROP SCHEMA EMPRESA CASCADE
```

- **RESTRICT**: borra el esquema sólo si NO contiene ningún elemento
- **CASCADE**: borra el esquema y todos sus contenidos

Borrar una tabla de un esquema:

```
DROP TABLE DEPENDIENTE CASCADE
```

- **RESTRICT**: borra la tabla sólo si NO existen referencias a la misma:
 - Desde claves externa de otra tabla
 - Desde alguna vista
- **CASCADE**: borra tabla y todas las restricciones (*constraints*) y vistas donde haya referencias a ésta

Evolución del esquema: columnas (ALTER TABLE)



Añadir columnas:

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

- En las tuplas existentes se asignan valores NULL. Alternativas: definir DEFAULT o introducir valores con la orden UPDATE (se estudiará más adelante)
- **NOT NULL** no está permitido

Borrar columnas:

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

- **CASCADE** borra también las restricciones (*constraints*) y vistas que hagan referencia a la columna (DIRECCIÓN)
- **RESTRICT** sólo borra la columna (DIRECCIÓN) si no hay restricciones ni vistas que le hagan referencia

Modificar la definición de columnas:

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

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

- Elimina la definición DEFAULT '888665555'
- Inserta una nueva definición de DEFAULT para el atributo NSS_JEFE

Evolución del esquema: restricciones (constraints) (ALTER TABLE)



Borrar restricciones:

```
ALTER TABLE EMPRESA.EMPLEADO DROP CONSTRAINT CLE_SUPERV_EMP;
```

- Es preciso haberle dado un nombre con CONSTRAINT en la definición (por ejemplo en CREATE TABLE)

Añadir restricciones:

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

Consultas básicas

```
SELECT columnas FROM tablas [WHERE condición]
```

La omisión de WHERE equivale a WHERE TRUE

- Condiciones en **WHERE**: {=, <, >, <=, >=}, AND, OR, NOT
- El resultado puede contener **TUPLAS REPETIDAS**

EMPLEADO

NOMBRE	INIC	APELLIDO	NSS	FECHA_NCTO	DIRECCIÓN	...
...	SEXO	SALARIO	NSS_SUPERV	ND		

DEPARTAMENTO

NOMBRED	NÚMEROD	NSS_JEFE	FECHA_INIC_JEFE
---------	---------	----------	-----------------

- Nombre y dirección de los empleados del departamento de Investigación

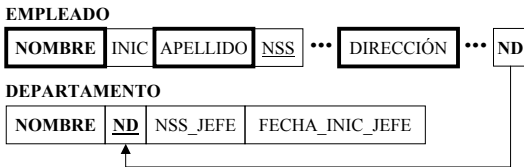
$\pi_{\text{NOMBRE, APELLIDO, DIRECCIÓN}}(2)$

C1: SELECT NOMBRE, APELLIDO, DIRECCIÓN FROM EMPLEADO, DEPARTAMENTO WHERE NOMBRED='Investigación' AND NÚMEROD=ND

(2) = $\sigma_{\text{NOMBRED}='Investigación'}(1)$

(1) = DEPARTAMENTO \bowtie NÚMEROD=ND EMPLEADO

Calificar atributos y alias



- Nombre, apellido y dirección de los empleados del departamento de investigación

Calificando atributos:

```
C1A: SELECT EMPLEADO.NOMBRE, APELLIDO, DIRECCIÓN
FROM EMPLEADO, DEPARTAMENTO
WHERE DEPARTAMENTO.NOMBRE = 'Investigación' AND
DEPARTAMENTO.ND=EMPLEADO.ND
```

Utilizando alias:

```
C1A': SELECT E.NOMBRE, APELLIDO, DIRECCIÓN
FROM EMPLEADO AS E, DEPARTAMENTO AS D
WHERE D.NOMBRE = 'Investigación' AND D.ND=E.ND
```

Declaración de alias (pointing to AS E, AS D)
Uso de alias (pointing to D.NOMBRE, D.ND, E.ND)

- Para cambiar los nombres de atributo:

```
FROM ..., DEPARTAMENTO AS D(NOM, ND, NSS, FECHA)
```

SELECT * ALL y DISTINCT

- Seleccionar todos los atributos de las tablas de FROM

```
C1A'': SELECT *
FROM EMPLEADO, DEPARTAMENTO
WHERE NOMBRED='Investigación' AND ND=NÚMEROD
```

- Seleccionar todos los atributos de EMPLEADO

```
C1A'': SELECT EMPLEADO.*
FROM EMPLEADO, DEPARTAMENTO
WHERE NOMBRED='Investigación' AND ND=NÚMEROD
```

- Seleccionar todos los valores (incluidos los repetidos) de salario de EMPLEADO

```
C11: SELECT ALL SALARIO
FROM EMPLEADO
```

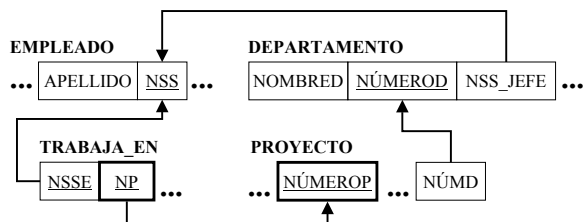
Opción por defecto (pointing to ALL)

- Seleccionar todos los *diferentes* valores de salario de EMPLEADO

```
C11A: SELECT DISTINCT SALARIO
FROM EMPLEADO
```

UNION [ALL], INTERSECT [ALL] y EXCEPT [ALL]

- Por defecto las tuplas repetidas se eliminan del resultado
- Con UNION ALL se conservan las repeticiones
- Se exige *compatibilidad de unión*



- Números de proyecto donde participa Smith como trabajador o como jefe del departamento controlador:

```
C4: SELECT NUMEROP
FROM PROYECTO, DEPARTAMENTO, EMPLEADO
WHERE NÚMD=NÚMEROD AND NSS JEFE=NSS AND
APELLIDO='Smith'
```

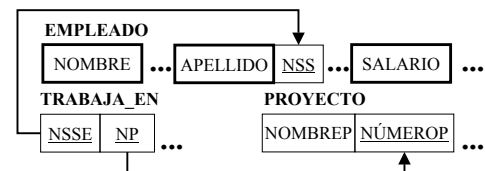
```
➔ UNION
SELECT NP
FROM TRABAJA_EN, EMPLEADO
WHERE NSSE=NSS AND APELLIDO='Smith'
```

LIKE '+', '-', '*', '/', y '|'

- Empleados que viven en Houston, Texas:

```
C12: SELECT NOMBRE, APELLIDO
FROM EMPLEADO
WHERE DIRECCIÓN LIKE '%Houston, TX%'
```

- % sustituye a un n° arbitrario de caracteres
- _ sustituye a un solo carácter

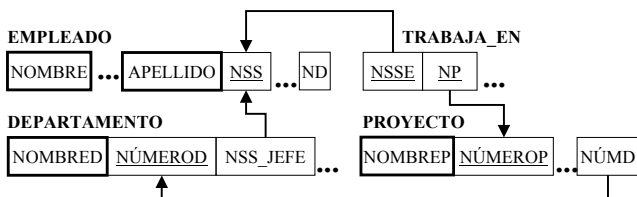


- Nombre y salario de los empleados que trabajan en 'ProductoX' tras aumentarles el sueldo un 10% :

```
C13: SELECT NOMBRE, APELLIDO, 1.1*SALARIO
FROM EMPLEADO, TRABAJA_EN, PROYECTO
WHERE NSS=NSSE AND NP=NÚMEROP AND
NOMBREP='ProductoX'
```

- ¿Qué produce 1.1 * SALARIO cuando SALARIO vale NULL?

Ordenación de tuplas (ORDER BY)



- Empleados y proyectos donde trabajan, ordenados por departamento y, dentro de cada departamento, ordenados alfabéticamente por apellido y nombre :

```
C15: SELECT NOMBRED, APELLIDO, NOMBRE, NOMBREP
FROM DEPARTAMENTO, EMPLEADO, TRABAJA_EN,
PROYECTO
WHERE NÚMEROD=ND, NSS=NSSE, NP=NÚMEROP
ORDER BY NOMBRED, APELLIDO, NOMBRE
```

- Por defecto, el orden es ascendente
- **DESC** indica orden descendente
- **ASC** indica orden ascendente
- El valor null también ocupa un orden entre los demás valores

```
ORDER BY NOMBRED DESC, APELLIDO ASC,
NOMBRE ASC
```

BETWEEN y conjuntos explícitos de valores

BETWEEN:

- Información de los empleados cuyo salario está entre 30.000 y 40.000

```
SELECT *
FROM EMPLEADO
WHERE (SALARIO BETWEEN 30000 AND 40000)
```

Equivale a SALARIO >= 30000 AND SALARIO <= 40000

Conjuntos explícitos de valores:

- NSS de los empleados que trabajan en los proyectos 1, 2 o 3

```
C17: SELECT DISTINCT NSSE
FROM TRABAJA_EN
WHERE NP IN (1,2,3)
```

Consultas anidadas

- **SELECT** en la cláusula **WHERE** de otra **SELECT**

TRABAJA_EN		
NSSE	NP	HORAS

- Información de los empleados que trabajan en algún proyecto en el que trabaje más de 10 horas a la semana el empleado '123456789'

```
SELECT NSSE
FROM TRABAJA_EN
WHERE NP IN (SELECT NP FROM TRABAJA_EN
WHERE NSS='123456789' AND HORAS>10)
```

Puede haber más niveles de anidamiento

- Algunas anidadas (como las que usan '=' e **IN**) se pueden escribir sin anidamientos:

```
SELECT T.NSSE
FROM TRABAJA_EN AS T INNER JOIN TRABAJA_EN AS
T_EMP ON T.NP=T_EMP.NP
WHERE T_EMP.NSS='123456789' AND T_EMP.HORAS>10
```

- Se admite el uso de conjuntos explícitos de valores: **WHERE (NP,HORAS) IN (SELECT NP, HORAS FROM ...)**
- Se compara **un valor** de atributo (o conjunto de atributos entre paréntesis) con el **conjunto de tuplas** devueltas por la subconsulta

Proceso de consultas anidadas

T1				T2			
A	B	C	D	E	F	G	A
1	bb	5	aaa	1	aaa	2	1
2	ab	3	aba	2	aeb	3	4
3	cb	4	bbe	3	aeb	8	2
4	ec	2	aeb	4	aeb	5	3
				5	aba	8	1

Atributo de T1

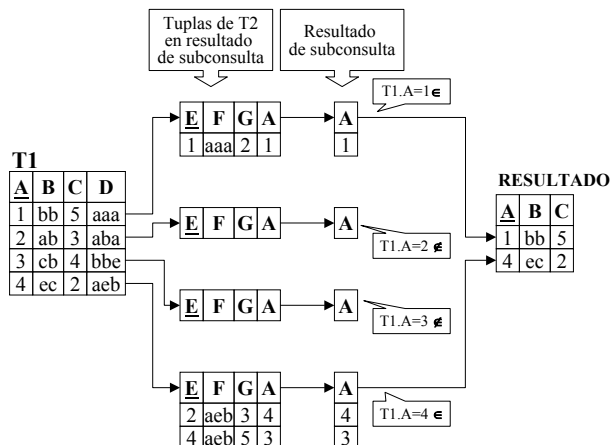
```
SELECT (A) B, C
```

```
FROM T1
```

```
WHERE A IN (SELECT (A)
FROM T2
WHERE G<6 AND F=T1.D)
```

Atributo de T2

Atributo del SELECT externo



IN, ALL y ANY (o SOME)

- WHERE atributo(s) IN subconsulta**
WHERE atributo(s) = ANY subconsulta
 - Cierto si y sólo si el valor del/de los atributo/s coincide con alguna tupla de la subconsulta
- WHERE atributo(s) < ANY subconsulta**
 - Cierto si y sólo si el valor del/de los atributo/s es menor que alguna tupla de la subconsulta
 - =, <, >, <=, >=, <>
- WHERE atributo(s) < ALL subconsulta**
 - Cierto si y sólo si el valor del/de los atributo/s es menor que todas las tuplas de la subconsulta
 - <, >, <=, >=, <>
- WHERE atributo(s) < subconsulta**
 - Cierto si y sólo si el valor del/de los atributo/s es menor que la única tupla de la subconsulta
 - =, <, >, <=, >=, <>

DNI	SALARIO
1	100
2	150
3	175
4	200
5	160

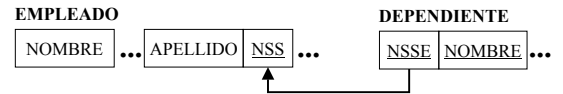
```

SELECT DNI
FROM EMPLEADO
WHERE SALARIO (*) ( SELECT SALARIO
                    FROM EMPLEADO)

(*)          Resultado
< ALL       =>  Ø
<=ALL      =>  1
< ANY       =>  1, 2, 3, 5
<= ANY     =>  1, 2, 3, 4, 5
    
```

EXISTS

- Devuelve cierto si y sólo si la subconsulta devuelve alguna tupla.



- Empleados **sin** familiares dependientes:

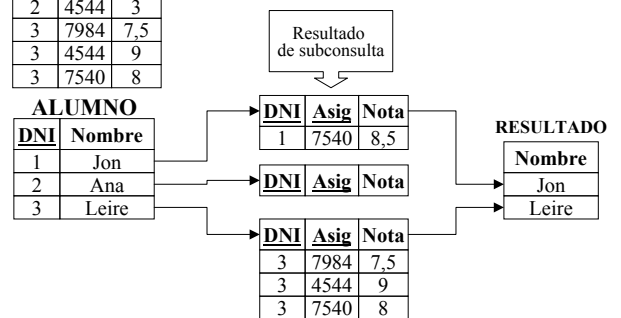
```

SELECT NOMBRE, APELLIDO
FROM EMPLEADO
WHERE NOT EXISTS ( SELECT * FROM DEPENDIENTE
                  WHERE NSS=NSSE)
    
```

DNI	Asig	Nota
1	7984	5
1	7450	4,5
1	7540	8,5
2	7984	6
2	4544	3
3	7984	7,5
3	4544	9
3	7540	8

```

SELECT Nombre
FROM ALUMNO AS A
WHERE EXISTS ( SELECT * FROM NOTA
              WHERE Nota >= 7 AND DNI=A.DNI)
    
```

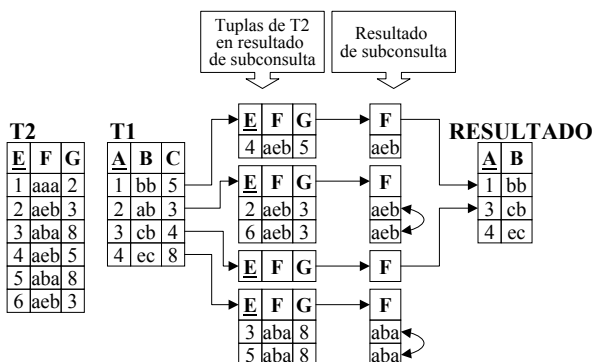


UNIQUE

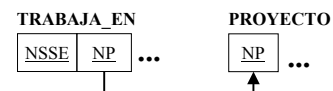
- Devuelve cierto si y sólo si la subconsulta no devuelve tuplas duplicadas.
- Ejemplo:

```

SELECT A, B
FROM T1
WHERE UNIQUE ( SELECT F FROM T2 WHERE G=T1.C)
    
```



División en SQL



- NSS de aquellos empleados que trabajan en **todos** los proyectos de la empresa

Resultado ← $(\pi_{NSSE, NP} TRABAJA_EN) \div (\pi_{NP} PROYECTO)$

```

SELECT DISTINCT T1.NSSE
FROM TRABAJA_EN AS T1
WHERE NOT EXISTS (
    ( SELECT NP FROM PROYECTO)
EXCEPT
    ( SELECT NP FROM TRABAJA_EN AS T2
      WHERE T1.NSSE = T2.NSSE)
)
    
```

Transformación de teoría de conjuntos

Todos los proyectos de la empresa

Todos los proyectos del empleado T1

```

SELECT DISTINCT T1.NSSE
FROM TRABAJA_EN AS T1
WHERE NOT EXISTS (
    ( SELECT NP FROM PROYECTO AS P
      WHERE NOT EXISTS
        ( SELECT *
          FROM TRABAJA_EN
          WHERE T1.NSSE = T2.NSSE AND
                P.NP=T2.NP)
    )
)
    
```

Transformación de cálculo relacional

Proyectos en los que no trabaja el empleado T1

Tuplas del proyecto P y del empleado T1

IS NULL e IS NOT NULL

- Nombre y apellido de empleados sin supervisores

```
C18: SELECT NOMBRE, APELLIDO
      FROM EMPLEADO
      WHERE NSS_SUPERV IS NULL
```

- Si pusiera **WHERE NSS_SUPERV = NULL**
 - Para las filas con NSS_SUPERV nulo se estaría comparando si NULL = NULL
 - Esta comparación **NO** devuelve cierto ni falso
 - La comparación devuelve NULL (UNKNOWN)

Una condición puede NO verificarse por FALSE o por NULL

- Cualquier cosa operada con NULL devuelve NULL (UNKNOWN)**

Ejemplos con valor null

La "tabla" del null

A	B	AND	OR	A	NOT
True	Null	Null	True	Null	Null
False	Null	Null	Null		
Null	Null	Null	Null		

Código de vendedor

Ventas

Cod	Vendedor	Vendido	Cuota
1	Juan	1000	1000
2	María	1500	1000
3	Jesús	500	Null
4	Ana	1000	1200
5	Aitor	1100	1000
6	Leire	2000	Null

```
SELECT Vendedor FROM Ventas WHERE Vendido > Cuota
```

Vendedor
María
Aitor

```
SELECT Vendedor FROM Ventas WHERE Vendido <= Cuota
```

Vendedor
Juan
Ana

```
SELECT Vendedor, Cuota+500 AS X FROM Ventas
```

Vendedor	Cuota
Juan	1500
María	1500
Jesús	Null
Ana	1700
Aitor	1500
Leire	Null

```
SELECT SUM(VD) AS V1, SUM(CU) AS V2, (SUM(VD)-SUM(CU)) AS V3, SUM(VD-CU) AS V4
```

```
FROM Ventas AS V(CD,VR,VD,CU)
```

V1	V2	V3	V4
7100	4200	2900	400

Ejemplos con valor null (2)

Código de vendedor

Cod	Vendedor	Vendido	Cuota
1	Juan	1000	1000
2	María	1500	1000
3	Jesús	500	Null
4	Ana	1000	1200
5	Aitor	1100	1000
6	Leire	2000	Null

```
SELECT SUM(CU) AS C1, COUNT(CU) AS C2, COUNT(*) AS C3
      FROM Ventas AS V(CD,VR,VD,CU)
      WHERE Vendido=500 OR Vendido=2000
```

```
SELECT Cuota, COUNT(*) AS C1
      FROM Ventas
      GROUP BY Cuota
```

```
SELECT VD, SUM(CU) AS V, COUNT(CU) AS C1, COUNT(*) AS C2
      FROM Ventas AS V(CD,VR,VD,CU)
      GROUP BY Vendido
```

Renombrar atributos del resultado con AS

```
C8A: SELECT E.APELLIDO AS NOMBRE_EMPLEADO,
           S.APELLIDO AS NOMBRE_SUPERVISOR
      FROM EMPLEADO AS E, EMPLEADO AS S
      WHERE E.NSS_SUPERV = S.NSS
```

- Cambia el nombre de cualquier columna (atributo) que aparezca en el resultado
- Antes hemos visto que la construcción **AS** también sirve para declarar alias de tablas:

```
FROM EMPLEADO E, ...
```

SQL1

```
FROM EMPLEADO AS E, ...
```

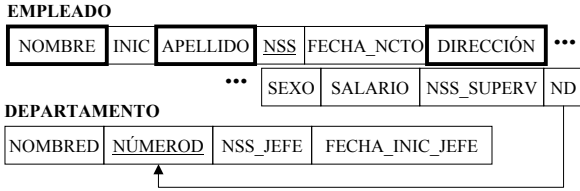
SQL2

SQL2

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

Tablas combinadas

(INNER, NATURAL y OUTER JOIN en FROM)



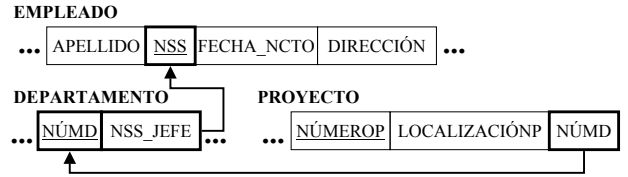
- Nombre y dirección de los empleados del departamento de Investigación

C1: C1A: **SELECT** NOMBRE, APELLIDO, DIRECCIÓN
FROM (EMPLEADO **INNER JOIN** DEPARTAMENTO
ON ND=NÚMEROD)
WHERE NOMBRED='Investigación'

- Este concepto se incorporó a SQL2
- La consulta se entiende más fácilmente, al evitar la mezcla de condiciones de \Join y σ en el WHERE
- Se pueden especificar diferentes tipos de reunión:
INNER JOIN (o **JOIN**), **NATURAL JOIN**, **LEFT [OUTER] JOIN**, **RIGHT [OUTER] JOIN** y **FULL [OUTER] JOIN**
- NO** se pueden definir **alias de tablas combinadas**:

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

Anidamiento de tablas combinadas



SELECT NÚMEROP, NÚMD, APELLIDO, DIRECCIÓN
 FECHA_NCTO
FROM (PROYECTO **NATURAL JOIN** DEPARTAMENTO)
INNER JOIN EMPLEADO **ON** NSS_JEFE=NSS
WHERE LOCALIZACIÓNP='Stafford'

OUTER JOIN

T1		T2	
A	B	E	F
@	null	000	null
aa	1	100	1
bb	2	101	1
cc	3	200	2
		400	4

SELECT *
FROM T1 **INNER JOIN** T2
ON B=F

A	B	E	F
aa	1	100	1
aa	1	101	1
bb	2	200	2

SELECT *
FROM T1 **LEFT JOIN** T2
ON B=F

A	B	E	F
@	null	null	null
aa	1	100	1
aa	1	101	1
bb	2	200	2
cc	3	null	null

Oracle: **SELECT** * **FROM** T1,T2 **WHERE** B=(+)F
SQLServer: **SELECT** * **FROM** T1,T2 **WHERE** B*=F

Oracle: ...**WHERE** B(+)=F
SQLServer: ...**WHERE** B*=F

SELECT *
FROM T1 **RIGHT JOIN** T2
ON B=F

A	B	E	F
null	null	000	null
aa	1	100	1
aa	1	101	1
bb	2	200	2
null	null	400	4

Oracle: no lo implementa
SQLServer: ...**WHERE** B*=*F

SELECT *
FROM T1 **FULL JOIN** T2
ON B=F

A	B	E	F
@	null	null	null
null	null	000	null
aa	1	100	1
aa	1	101	1
bb	2	200	2
cc	3	null	null
null	null	400	4

UNION JOIN

- Se puede considerar como una mezcla de FULL OUTER JOIN y UNION.
- Los atributos correspondientes con dominio compatible se tratan como en la unión. Los incompatibles, se tratan como en FULL OUTER JOIN.
- Utilidad limitada.

T1	T2	T3	T4
A	B	C	B D
aa	aa	1	a 1
bb	ab	2	aa 2
cc	bc		b 3

SELECT *
FROM T1 **UNION JOIN** T2

A
aa
ab
bb
bc
cc

SELECT *
FROM T2 **UNION JOIN** T3

B	C
aa	null
ab	null
bc	null
null	1
null	2

SELECT *
FROM T2 **UNION JOIN** T4

B	D
a	1
aa	2
ab	null
b	3
bc	null

Funciones agregadas y de agrupación

- **COUNT** (cuenta), **SUM** (suma), **MAX** (máximo), **MIN** (mínimo), **AVG** (media)

- Con expresiones : **AVG**(1.1*SALARIO) dominio con orden total
- **MAX** y **MIN** también con atributos **NO** numéricos

- Suma de salarios del dpto. 'Investigación', junto a los salarios máximo, mínimo y medio:

C20: `SELECT SUM(SALARIO), MAX(SALARIO), MIN(SALARIO), AVG(SALARIO)
FROM EMPLEADO INNER JOIN DEPARTAMENTO ON
ND=NÚMEROD
WHERE NOMBRED='Investigación'`

- N° de empleados en el departamento 'Investigación':

C22: `SELECT COUNT(*)
FROM EMPLEADO INNER JOIN DEPARTAMENTO ON
ND=NÚMEROD
WHERE NOMBRED='Investigación'`

EMPLEADO			DEPARTAMENTO			
NOMBREP	SALARIO	ND	NOMBRED	NÚMEROD		
John	NULL	5	Investigación	5		
Franklin	15.000	5	Administración	4		
Ramesh	10.000	5	Dirección	1		
Joyce	10.000	5				
Alicia	10.000	4				
Jennifer	20.000	4				
Ahmad	20.000	4				
James	20.000	1				

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

COUNT(*)
4

35.000 / 3

© A. Jaime 2003

DBD Tema 4

53

Funciones agregadas y de agrupación (2)

- Cuántos valores de salario **diferentes** hay:

C23: `SELECT COUNT(DISTINCT SALARIO)
FROM EMPLEADO`

NO cuenta los valores nulos

- Cuántos valores de salario hay (con repeticiones):

C23': `SELECT COUNT(SALARIO)
FROM EMPLEADO`

Cuenta las filas con salario NO nulo

- Cuántos empleados hay:

C23'': `SELECT COUNT(*)
FROM EMPLEADO`

Cuenta todas las filas de la tabla

EMPLEADO	SALARIO	ND
John	NULL	5
Franklin	15.000	NULL
Ramesh	10.000	5
Joyce	10.000	5
Alicia	10.000	4
Jennifer	20.000	4
Ahmad	20.000	4
James	20.000	1

COUNT(DISTINCT SALARIO)
3

COUNT(SALARIO)
7

COUNT(*)
8

- ¿Qué ocurre cuando todos los salarios valen NULL?
- ¿Qué ocurre cuando la tabla está vacía?

© A. Jaime 2003

DBD Tema 4

54

Atributos de agrupación: GROUP BY

- Obtener por cada dpto. su nombre y número junto al número de empleados del mismo y el salario medio del dpto.:

C24': `SELECT NOMBRED, ND, COUNT(*) AS N_EMP,
AVG(SALARIO) AS SAL_MED
FROM EMPLEADO LEFT JOIN DEPARTAMENTO ON
ND=NÚMEROD
GROUP BY NOMBRED, ND`

- Todos los atributos de **SELECT** (que no son atributos de funciones) deben estar en **GROUP BY**
- Los atributos de **GROUP BY** no es obligatorio que estén en **SELECT**

SALARIO	ND	NOMBRED
30.000	5	Investigación
15.000	5	Investigación
10.000	5	Investigación
10.000	5	Investigación
10.000	4	Administración
20.000	4	Administración
20.000	4	Administración
20.000	1	Dirección

NOMBRED	ND	N_EMP	SAL_MED
Investigación	5	4	16.250
Administración	4	3	16.666
Dirección	1	1	20.000

SALARIO	ND	NOMBRED
NULL	NULL	NULL
10	NULL	NULL
10	5	DMC
10	5	DMC

NOMBRED	ND	N_EMP	SAL_MED
NULL	NULL	2	10
DMC	5	2	10

Los valores nulos forman su propio grupo

© A. Jaime 2003

DBD Tema 4

55

HAVING

- Para especificar una condición en términos del grupo de tuplas asociado a cada valor de los atributos de agrupación



- Para cada proyecto con más de 2 empleados, obtener su número, nombre y n° de empleados

SELECT NP, NOMBREP, COUNT(*)
FROM PROYECTO NATURAL JOIN TRABAJA_EN
GROUP BY NP, NOMBREP
HAVING COUNT(*)>2

© A. Jaime 2003

DBD Tema 4

56

Primero WHERE, después HAVING

- Nº de empleados con salario > 30.000 en cada dpto. Sólo para dptos. con más de 2 empleados con ese sueldo

```
SELECT NOMBRED, COUNT(*)
FROM EMPLEADO INNER JOIN DEPARTAMENTO ON
ND = NÚMEROD
WHERE SALARIO > 30.000
GROUP BY NOMBRED
HAVING COUNT(*) > 2
```

EMPLEADO	NOMBREP	SALARIO	ND	NOMBRED
INNER JOIN	José	35.000	5	Investigación
DEPARTAMENTO:	Federico	40.000	5	Investigación
	Ramón	38.000	5	Investigación
	Josefa	25.000	5	Investigación
	Alicia	25.000	4	Administración
	Jazmín	43.000	4	Administración
	Ahmed	25.000	4	Administración
	Jaime	55.000	1	Dirección

Primero se ejecuta WHERE

EMPLEADO	NOMBREP	SALARIO	ND	NOMBRED
INNER JOIN ... WHERE SALARIO > 30.000:	José	35.000	5	Investigación
	Federico	40.000	5	Investigación
	Ramón	38.000	5	Investigación
	Jazmín	43.000	4	Administración
	Jaime	55.000	1	Dirección

Después se ejecuta HAVING

NOMBRED	COUNT(*)
Investigación	3

GROUP BY con ORDER BY

- Obtener por cada proyecto su número y nombre junto al número de empleados que trabajan en él, ordenado ascendentemente por el número de empleados

```
C25': SELECT NÚMEROP, NOMBREP, COUNT(*)
FROM PROYECTO INNER JOIN TRABAJA_EN
ON NÚMEROP=NP
GROUP BY NÚMEROP, NOMBREP
ORDER BY COUNT(*) ASC
```

```
C25'': SELECT NÚMEROP, NOMBREP, COUNT(*) AS
NUM_EMP
FROM PROYECTO INNER JOIN TRABAJA_EN
ON NÚMEROP=NP
GROUP BY NÚMEROP, NOMBREP
ORDER BY NUM_EMP ASC
```

NO se puede usar el nuevo nombre (AS) del atributo del resultado en ORDER BY

CASE

```
SELECT NOMBRE,
CASE
WHEN ESTADOCIVIL='S' THEN 'SOLTERO/A'
WHEN ESTADOCIVIL='C' THEN 'CASADA/O'
WHEN ESTADOCIVIL='D' THEN 'DIVORCIADO/A'
ELSE 'VIUDA/O'
END, EDAD, FECHA_NACIMIENTO
FROM PERSONAS;
```

- En los WHEN cualquier condición (AND, OR, ...)
- Ahorro espacio almacenamiento: S/C/D/V frente a Soltero/Casado ...

```
SELECT NOMBRE,
CASE ESTADO_CIVIL
WHEN 'S' THEN 'SOLTERO/A'
WHEN 'C' THEN 'CASADA/O'
WHEN 'D' THEN 'DIVORCIADO/A'
ELSE 'VIUDA/O'
END, EDAD, FECHA_NACIMIENTO
FROM PERSONAS;
```

- En los WHEN un valor posible del atributo

```
UPDATE EMPLEADO
SET SUELDO = CASE DEPTO
WHEN 'VIDEO' THEN SUELDO*1.1
WHEN 'MÚSICA' THEN SUELDO*1.2
ELSE 0
END;
```

NULLIF y COALESCE

Operador NULLIF:

```
SELECT ...
FROM ...
WHERE BENEFICIO / NULLIF(COSTO, -1) > 100
```

- Cuando COSTO vale -1 NULLIF devuelve NULL
- La división tiene un comportamiento predefinido cuando el denominador sea NULL (el resultado es NULL)

```
WHERE BENEFICIO / CASE WHEN COSTO = -1 THEN NULL
ELSE COSTO END > 100
```

- NULLIF es una abreviatura de CASE

Operador COALESCE:

```
K1: SELECT NOMBRE, COALESCE(SUELDO, PARO,
SALARIO_SOCIAL) AS SALARIO
FROM DATOS_HACIENDA;
```

- Devuelve el primer valor NO nulo de la lista que sigue a la palabra COALESCE

DATOS_HACIENDA

NOMBRE	SUELDO	PARO	SALARIO_SOCIAL
Matias	NULL	NULL	20.000
Marta	NULL	30.000	NULL
Maidar	40.000	NULL	NULL

K1:

NOMBRE	SALARIO
Matias	20.000
Marta	30.000
Maidar	40.000

CAST (conversión de tipos) y constructor de valor de tupla

```
WHERE DIRECTOR.FECHA_INICIO >  
    CAST(EMPLEADO.FECHA_ALTA_SS  
    AS DATE);
```

- Convierte el campo FECHA_ALTA_SS a tipo DATE

- Una sola comparación incluye todos los valores de dos tuplas:

```
WHERE (NOMBRE, EDAD, ESTADO_CIVIL) =  
    ("José María", 18, 'S')
```

equivale a:

```
WHERE NOMBRE= "José María" AND EDAD=18  
    AND ESTADO_CIVIL="S"
```

```
WHERE (C1, C2, C3) < (T1, T2, T3)
```

equivale a:

```
WHERE C1<T1 OR  
    (C1=T1 AND C2<T2) OR  
    (C1=T1 AND C2=T2 AND C3<T3)
```

Análisis de consultas SQL

```
SELECT <atributos y funciones>  
FROM <tablas>  
[WHERE <condición>]  
[GROUP BY <atributos agrupación>]  
[ORDER BY <atributos ordenación>]
```

También hay
HAVING, que
este curso no
estudiaremos

- Una consulta SQL se *evalúa conceptualmente* así:

- Primero **FROM**, seguido de **WHERE**, en tercer lugar **GROUP BY** y por último **ORDER BY**
- Si no hay **GROUP BY** ni **ORDER BY**, para cada combinación de tuplas (una de cada tabla de **FROM**), se evalúa la condición de **WHERE**. Si es cierta se colocan en el resultado los valores correspondientes a los atributos del **SELECT**.
- Esta **NO** es una forma eficiente de implementar una consulta SQL. Así pues, cada SGBD tiene rutinas para **optimizar** la evaluación.

Análisis de consultas SQL (2)

- En SQL hay varias alternativas para especificar la misma consulta :
- **Ventaja:** el programador elige la técnica que le resulte más cómoda.
- Desde el punto de vista de **optimización de consultas**, conviene que las consultas tengan el **menor anidamiento** y el menor **ordenamiento implícito** posible.
- **Desventaja:** el programador puede desconocer cuál es la técnica más eficiente en cada caso
- Idealmente, el SGBD debería procesar la consulta de la misma manera sin importar cómo se haya escrito.
- En la **práctica** esto resulta muy difícil, y es **conveniente que el usuario sea consciente** de qué construcciones tienen un costo más elevado que otras.

INSERT



A1: INSERT INTO EMPLEADO

```
VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98  
Oak Forest, Katy, TX', 'H', 37000, '987654321', 4)
```

- Mismo orden en el que se especificaron los atributos en **CREATE TABLE**

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

```
VALUES ('Richard', 'Marini', '653298653')
```

- Así los atributos con valor NULL o DEFAULT se pueden omitir
- Los valores de **VALUES** en el mismo orden que se especifican los atributos en **INSERT INTO**
- También se pueden incluir varias tuplas en la misma instrucción: **INSERT ... INTO ... VALUES (tupla1), (tupla2), ... (tuplaN)**

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

```
VALUES ('Robert', 'Hatcher', '980760540', 2)
```

- Rechazada por la inexistencia del departamento número 2

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

```
VALUES ('Robert', 'Hatcher', 5)
```

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



INSERT (2)

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

A3B: INSERT INTO INFO_DEPTOS (NOMBRE_DEPTO,
 NÚM_DE_EMPS, SAL_TOTAL)
SELECT NOMBRED, COUNT (*), SUM(SALARIO)
FROM DEPARTAMENTO **INNER JOIN** EMPLEADO
ON NÚMEROD=ND
GROUP BY NOMBRED;

- Inserta varias tuplas (el resultado de la consulta)
- Utilidad: tabla temporal donde realizar consultas
 - Sus datos pueden *perder actualidad*
 - Alternativa sin este problema: vista



DELETE

A4A: DELETE FROM EMPLEADO
WHERE APELLIDO='Brown'

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

A4B: DELETE FROM EMPLEADO
WHERE NSS='123456789'

A4C: DELETE FROM EMPLEADO
WHERE ND IN
 (SELECT NÚMEROD
FROM DEPARTAMENTO
WHERE NOMBRED='Investigación')

A4D: DELETE FROM EMPLEADO

- Sin WHERE se borran todas las tuplas (quedaría la tabla vacía)
- Usando **DROP TABLE** se hubiera eliminado además la definición de la tabla



UPDATE

A5: UPDATE PROYECTO
SET LOCALIZACIÓNP='Bellaire', NÚMD=5
WHERE NÚMEROP=10

- Una sola tabla
- WHERE: selección de tuplas a modificar
- SET: atributos a modificar y nuevos valores
- SET: el nuevo valor puede ser NULL o DEFAULT
- Modificaciones de clave primaria pueden propagarse a clave/s extranjera/s (debido a las acciones declaradas en la RI, como CASCADE)

A6: UPDATE EMPLEADO
SET SALARIO=SALARIO*1.1
WHERE ND IN (SELECT NÚMEROD
FROM DEPARTAMENTO
WHERE NOMBRED='Investigación')

- A la izda se refiere al nuevo valor de SALARIO
- A la dcha al valor antiguo

Vistas en SQL

- Una vista es una **tabla derivada** de otras tablas (que pueden ser tablas de base u otras vistas).
- Tipos de vista:
 - **Tabla virtual:** se calcula, no se almacena en la BD. Siempre está *al día*: sus "tuplas" se crean cuando se realiza una consulta sobre la vista.
 - **Vista materializada:** se crea una físicamente una tabla cuando se consulta por primera vez. Se pueden hacer sucesivas consultas sobre esa tabla. Hay técnicas para mantener la vista actualizada de forma incremental.
- Ejemplo de creación de la vista TRABAJA_EN_1:
CREATE VIEW TRABAJA_EN_1
AS SELECT NOMBRE, APELLIDO, NOMBREP, HORAS
FROM (EMPLEADO **INNER JOIN** TRABAJA_EN **ON**
 NSS=NSSE) **INNER JOIN** PROYECTO **ON** NP=NUMEROP
- La vista se puede usar en consultas:
SELECT NOMBRE, APELLIDO
FROM TRABAJA_EN_1
WHERE NOMBREP='ProyectoX'
- Para borrar una vista: **DROP VIEW** TRABAJA_EN_1
- Una vista tiene limitaciones para actualizar sus tuplas como si fuese una tabla, ya que puede corresponder a varias actualizaciones de las tablas de base.

Ejercicios

Ejercicio: Operaciones del álgebra

EMP			DEP		PROY		TRAB		
NOMBRE	NSS	ND	NUMD	NOMD	NP	NOMBRE	NSSE	NP	HORAS
Iker	11	1	1	LSI	1	.NET	11	1	5
Ana	22	1	2	ATC	2	XML	11	2	2
Jon	33	2			3	EJB	22	1	3
Karmele	44	2			4	UML	22	3	1
							44	3	4

Dibujar las relaciones resultantes de realizar las siguientes operaciones del álgebra relacional:

- Nombre y NSS de cada empleado junto al número de cada proyecto en el que trabaja:

$$PERS \leftarrow \pi_{NOMBRE, NSS}(EMP)$$

$$TRB(NSS, NP) \leftarrow \pi_{NSS, NP}(TRAB)$$

$$R6 \leftarrow PERS * TRB$$
- Nombre de cada empleado junto al nombre de cada proyecto en el que trabaja:

$$PRY(NP, NOMP) \leftarrow PROJ$$

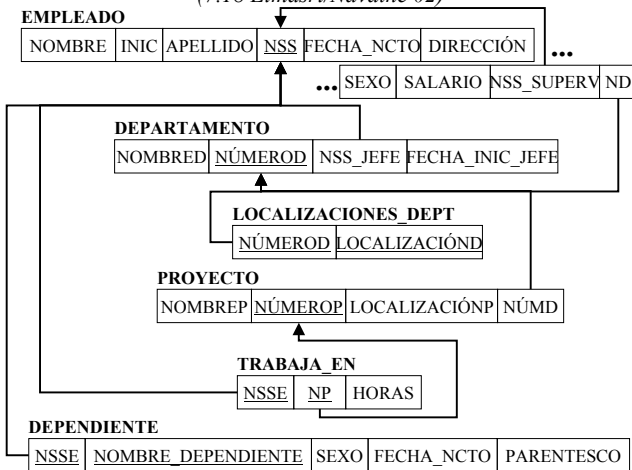
$$PERS_PRY \leftarrow R6 * PRY$$

$$R7 \leftarrow \pi_{NOMBRE, NOMP}(PERS_PRY)$$

– ¿Qué cambia en PERS_PRY si se usa PROJ en lugar de PRY?

Ejercicio: consultas álgebra relacional EMPRESA

(7.18 Elmasri/Navathe 02)



- Empleados del departamento 5 que trabajan más de 10 horas/semana en el proyecto 'Producto X'
- Nombre de cada proyecto junto al número total de horas invertidas por los empleados en él.
- Nombres de todos los empleados que trabajan en todos y cada uno de los proyectos
- Empleados que no trabajan en ningún proyecto
- Nombre y dirección de los empleados que trabajan en algún proyecto que, por una parte, está situado en Houston y por otra el proyecto pertenece a un departamento que no está situado en Houston.

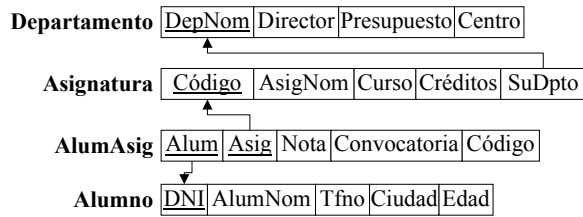
Ejercicio: consultas SQL EMPRESA

(8.13 (7.18) Elmasri/Navathe 02)

Sobre el esquema de BD de la figura 7.7 (pg. 4): (donde pide empleados se refiere a su nombre y apellido)

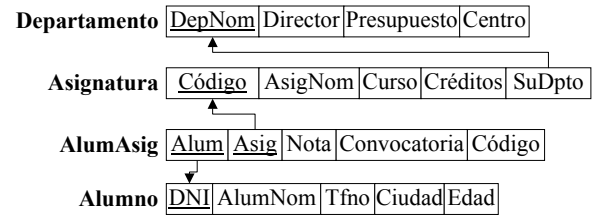
- Empleados del departamento 5 que trabajan más de 10 horas/semana en el proyecto 'Producto X'
- Empleados con un dependiente con su mismo nombre de pila
- Empleados cuyo jefe directo es Franklin Wong
- Nombre de cada proyecto junto al número total de horas trabajadas por los empleados en él.
- Empleados que trabajan en todos los proyectos de la empresa
- Empleados que no trabajan en ningún proyecto
- Nombre de cada departamento junto al salario medio de los empleados asignados al mismo
- Salario medio de las empleadas de la compañía
- Nombre y dirección de los empleados que trabajan en algún proyecto situado en Houston pero departamento no está situado allí
- Jefes de departamento sin dependientes

Ejercicio: consultas SQL UNIVERSIDAD (1)



- Por cada curso, obtener el alumno con mejor nota media en las asignaturas de este curso.
- Obtener el DNI de los alumnos que están “limpios” en 3º curso, es decir, que están matriculados en 3º y tienen aprobadas todas las asignaturas de cursos inferiores.
- DNI de los alumnos que están matriculados de ALGUNA asignatura de tercero y tienen aprobadas TODAS las asignaturas de segundo.
- Obtener el DNI de los alumnos matriculados en al menos una asignatura de cada uno de los departamentos de Informática.
- Obtener el nombre de los alumnos que están matriculados en sexta convocatoria en al menos dos asignaturas (sin utilizar funciones agregadas).
- Obtener el DNI de los alumnos que están matriculados de todas las asignaturas de 3º y de alguna otra asignatura de cursos inferiores.

Ejercicio: consultas SQL UNIVERSIDAD (2)



- Nombre de los alumnos que han obtenido como nota SOBRESALIENTE en al menos dos asignaturas de cualquiera de los cursos en los que hayan estado matriculados (sin utilizar funciones agregadas).
- Obtener, sin usar funciones agregadas, el nombre de los alumnos que están matriculados sólo de asignaturas del departamento DMC.
- Obtener, sin usar funciones agregadas, el nombre de los alumnos matriculados en alguna asignatura de tercer curso que no tengan ninguna matrícula en asignaturas de segundo curso.
- Para cada asignatura del departamento DMC con más de 10 alumnos matriculados, obtener el código de la asignatura y el número de convocatoria media en la que se encuentran los alumnos de Arnedo.
- Obtener el nombre de los alumnos de Nájera mayores de 20 años que se encuentran matriculados en alguna asignatura de tercero.

Soluciones

Solución Operaciones del álgebra (1)

EMP			DEP		PROY		TRAB		
NOMBRE	NSS	ND	NUMD	NOMD	NP	NOMBRE	NSSE	NP	HORAS
Iker	11	1	1	LSI	1	.NET	11	1	5
Ana	22	1	2	ATC	2	XML	11	2	2
Jon	33	2			3	EJB	22	1	3
Karmele	44	2			4	UML	22	3	1
							44	3	4

PRY1_MASIHORA			R1	R2	TODOS_PRY	R3
NSSE	NP	HORAS	NSSE	NP	NP	NP
11	1	5	11	1	1	4
22	1	3	22	2	2	
				3	3	
					4	

$PRY1_MASIHORA \leftarrow \sigma_{NP=1 \text{ Y } HORAS > 1}(TRAB)$
 $R1 \leftarrow \pi_{NSS}(PRY1_MASIHORA)$
 $R2 \leftarrow \pi_{NP}(TRAB)$
 $TODOS_PRY \leftarrow \pi_{NP}(PROY)$
 $R3 \leftarrow TODOS_PRY - R2$

R4					R5	
NOMBRE	NSS	ND	NUMD	NOMD	NOMBRE	NOMD
Iker	11	1	1	LSI	Iker	LSI
Ana	22	1	1	LSI	Ana	LSI
Jon	33	2	2	ATC	Jon	ATC
Karmele	44	2	2	ATC	Karmele	ATC

$R4 \leftarrow EMP \bowtie_{ND=NUMD} DEP$
 $R5 \leftarrow \pi_{NOMBRE, NOMD}(R4)$

Solución Operaciones del álgebra (2)

EMP			DEP		PROY		TRAB		
NOMBRE	NSS	ND	NUMD	NOMD	NP	NOMBRE	NSSE	NP	HORAS
Iker	11	1	1	LSI	1	.NET	11	1	5
Ana	22	1	2	ATC	2	XML	11	2	2
Jon	33	2			3	EJB	22	1	3
Karmele	44	2			4	UML	22	3	1
							44	3	4

PERS			TRB		R6		
NOMBRE	NSS		NSS	NP	NOMBRE	NSS	NP
Iker	11		11	1	Iker	11	1
Ana	22		11	2	Iker	11	2
Jon	33		22	1	Ana	22	1
Karmele	44		22	3	Ana	22	3
			44	3	Karmele	44	3

$PERS \leftarrow \pi_{NOMBRE, NSS}(EMP)$
 $TRB(NSS, NP) \leftarrow \pi_{NSS, NP}(TRAB)$
 $R6 \leftarrow PERS * TRB$

PRY		PERS_PRY				R7	
NP	NOMP	NOMBRE	NSS	NP	NOMP	NOMBRE	NOMP
1	.NET	Iker	11	1	.NET	Iker	.NET
2	XML	Iker	11	2	XML	Iker	XML
3	EJB	Ana	22	1	.NET	Ana	.NET
4	UML	Ana	22	3	EJB	Ana	EJB
		Karmele	44	3	EJB	Karmele	EJB

$PRY(NP, NOMP) \leftarrow \pi_{NP, NOMBRE}(PROY)$
 $PERS_PRY \leftarrow R6 * PRY$
 $R7 \leftarrow \pi_{NOMBRE, NOMP}(PERS_PRY)$

PERS_PRY		
NOMBRE	NSS	NP
Iker	11	1
Iker	11	2
Ana	22	1
Ana	22	3
Karmele	44	3

Usando PROY en lugar de PRY en PERS_PRY

Solución consultas álgebra relacional EMPRESA

(Elmasri/Navathe 7.18)

- a) Empleados del departamento 5 que trabajan más de 10 horas/semana en el proyecto 'Producto X'
- $$EMP_DP5(NSSE) \leftarrow \pi_{NSSE}(\sigma_{ND=5}(EMPLEADO))$$
- $$NUM_PROY(NP) \leftarrow \pi_{NUMEROP}(\sigma_{NOMBREP='ProductoX'}(PROYECTO))$$
- $$EMP_PR \leftarrow \sigma_{HORAS > 10}(TRABAJA_EN * NUM_PROY)$$
- $$R1 \leftarrow \pi_{NOMBRE, APELLIDO}(EMP_PR * EMP_DP5)$$
- b) Empleados con un dependiente con su mismo nombre de pila
- $$DEP(NSSE, NOMBRE) \leftarrow \pi_{NSSE, NOMBRE-DEPENDIENTE}(DEPENDIENTE)$$
- $$EMP_CON_DEP_IGUAL \leftarrow EMPLEADO * DEP$$
- $$R2 \leftarrow \pi_{NOMBRE, APELLIDO}(EMP_CON_DEP_IGUAL)$$
- c) Empleados cuyo jefe directo es Franklin Wong
- $$F_WONG(NSSE_SUPERV) \leftarrow \pi_{NSSE}(\sigma_{NOMBRE='Franklin' \wedge APELLIDO='Wong'}(EMPLEADO))$$
- $$SUPERVISADOS \leftarrow F_WONG * EMPLEADO$$
- $$R3 \leftarrow \pi_{NOMBRE, APELLIDO}(SUPERVISADOS)$$
- d) Nombre de cada proyecto junto al número total de horas trabajadas por los empleados en él.
- $$PROY_NOM \leftarrow PROYECTO \times_{|NUMEROP=NP} TRABAJA_EN$$
- $$R4(NOMBREP, HORAS) \leftarrow \pi_{NOMBREP} \sum_{SUMA\ HORAS}(PROY_NOM)$$

Solución consultas EMPRESA (2)

(Elmasri/Navathe 7.18)

- e) Nombres de todos los empleados que trabajan en cada uno de los proyectos

Todos los proyectos junto a los empleados que trabajan en cada uno

$$PRY_EMPS(NOMP, NSS) \leftarrow \pi_{NOMBREP, NSSE}(\sigma_{PROYECTO \times_{|NUMEROP=NP} TRABAJA_EN})$$

$$R5 \leftarrow \pi_{NOMP, NOMBRE, APELLIDO}(PRY_EMPS * EMPLEADO)$$

Los que trabajan en todos y cada uno de los proyectos

$$PROYS(NP) \leftarrow \pi_{NUMEROP}(PROYECTO)$$

$$EMPS_TODOS_PRYS(NSS) \leftarrow \pi_{NP, NSSE}(TRABAJA_EN) \div PROYS$$

$$R5' \leftarrow \pi_{NOMBRE, APELLIDO}(EMPLEADO * EMPS_TODOS_PRYS)$$

- f) Empleados que no trabajan en ningún proyecto

$$EMP_PROY(NSSE) \leftarrow \pi_{NSSE}(TRABAJA_EN)$$

$$EMPS \leftarrow \pi_{NSSE}(EMPLEADO)$$

$$EMPS_SIN_PRY \leftarrow EMPS - EMP_PROY$$

$$R6 \leftarrow \pi_{NOMBRE, APELLIDO}(EMPLEADO * EMPS_SIN_PRY)$$

- g) Nombre de cada departamento junto al salario medio de los empleados asignados al mismo

$$DPTO_EMP \leftarrow DEPARTAMENTO \times_{|NUMEROD=ND} EMPLEADO$$

$$R7 \leftarrow NOMBRED \sum_{PROMEDIO\ SALARIO}(DPTO_EMP)$$

- h) Salario medio de las empleadas de la empresa

$$EMP_FEM \leftarrow \sigma_{SEXO='M'}(EMPLEADO)$$

$$R8 \leftarrow \sum_{PROMEDIO\ SALARIO}(EMP_FEM)$$

Solución consultas EMPRESA (3)

(Elmasri/Navathe 7.18)

- i) Nombre y dirección de los empleados que trabajan en algún proyecto situado en Houston pero cuyo departamento no está situado allí

$$PRY_HOUSTON \leftarrow \pi_{NUMEROP, NUMD}(\sigma_{LOCALIZACIONP='Houston'}(PROYECTO))$$

$$DPT_HOUSTON \leftarrow \pi_{NUMEROD}(\sigma_{LOCALIZACIOND='Houston'}(LOCALIZACIONES_DEPTOS))$$

$$TODOS_DPTOS \leftarrow \pi_{NUMEROD}(DEPARTAMENTO)$$

$$DPT_NO_HOUSTON \leftarrow TODOS_DPTOS - DPT_HOUSTON$$

$$PRY_BUSCADOS(NP) \leftarrow \pi_{NUMEROP}(PRY_HOUSTON \times_{|NUMD=NUMEROD} DPT_NO_HOUSTON)$$

$$EMP_BUSCADOS(NSSE) \leftarrow \pi_{NSSE}(TRABAJA_EN * PRY_BUSCADOS)$$

$$R9 \leftarrow \pi_{NOMBRE, APELLIDO, DIRECCION}(EMPLEADO * EMP_BUSCADOS)$$

(1) NO se puede calcular así:

$$DPT_NO_HOUSTON \leftarrow \pi_{NUMEROD}(\sigma_{LOCALIZACIOND \neq 'Houston'}(LOCALIZACIONES_DEPTOS))$$

- j) Jefes de departamento sin dependientes

$$JEFES(NSSE) \leftarrow \pi_{NSSE_JEFE}(DEPARTAMENTO)$$

$$JEFES_CON_DEP(NSSE) \leftarrow \pi_{NSSE}(JEFES * DEPENDIENTE)$$

$$JEFES_SIN_DEP(NSSE) \leftarrow JEFES - JEFES_CON_DEP$$

$$R10 \leftarrow \pi_{NOMBRE, APELLIDO}(JEFES_SIN_DEP * EMPLEADO)$$

$$JEFES_SIN_DEP(NSSE) \leftarrow JEFES - (\pi_{NSSE}(EMPLEADO \times_{|NSS=NSSE} DEPENDIENTE))$$

Alternativa válida