TEMA 4. EL MODELO RELACIONAL

1. El modelo relacional
2. Estructura de datos relacional
3. Reglas de integridad
4. Lenguajes relacionales
5. Vistas

1. El Modelo Relacional

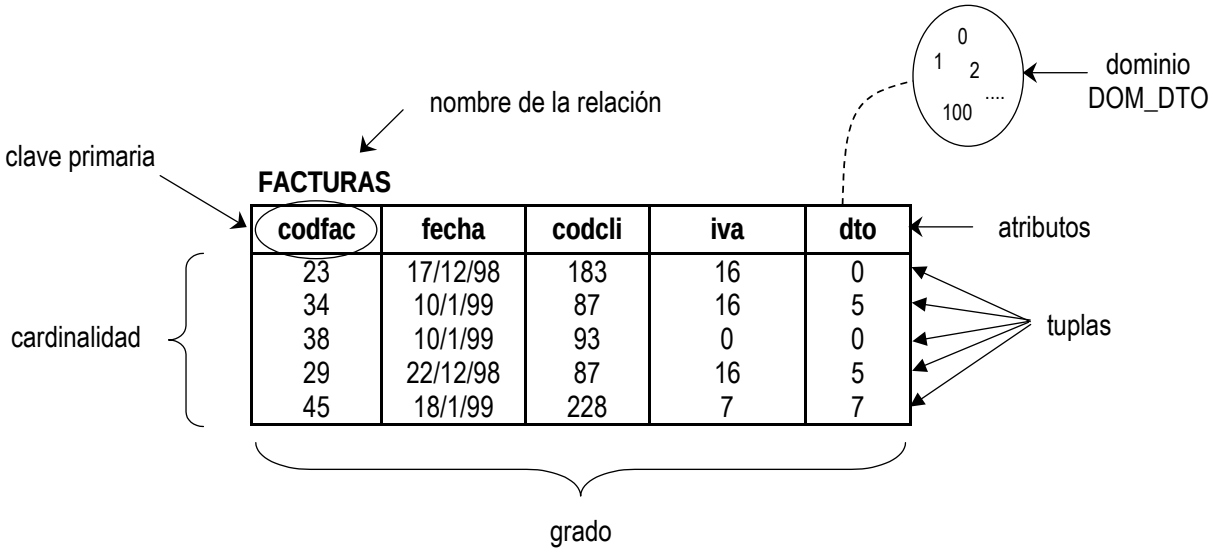
Se debe a E.F. Codd y data de 1970.

Está basado en dos ramas de las matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden. Su base matemática hace que el modelo sea predecible, fiable y seguro ¿qué más se puede pedir?

Todo modelo de datos tiene que ver con tres aspectos de los datos:

- *Estructura de datos.*
- *Integridad de datos.*
- *Manejo de datos.*

2. Estructura de Datos Relacional



FACTURAS(codfac, fecha, codcli, iva, dto)

Una **relación R** definida sobre un **conjunto de dominios D_1, D_2, ... D_n** consta de:

- **Cabecera**: conjunto fijo de pares **atributo:dominio**.

  $\{(A_1:D_1), (A_2:D_2), \dots (A_n:D_n)\}$

  n: **grado** de R (relación n-ária)

  - Cada A_j corresponde a un único D_j.
  - Los A_j son todos distintos.

- **Cuerpo**: conjunto variable de **tuplas**.

  Tupla: conjunto de pares **atributo:valor**.

  $\{(A_1:v_{i1}), (A_2: v_{i2}), \dots (A_n: v_{in})\}$ con i = 1, 2, ... m

  m: **cardinalidad** de R (número de tuplas)

  - En cada $(A_j:v_{ij})$ se tiene que $v_{ij} \in D_j$.

Relación **FACTURAS** definida sobre el conjunto de dominios:

Dominio	*Definición*
DOM_CODFAC :	Números enteros positivos.
DOM_FECHA :	Fechas válidas.
DOM_CODCLI :	Números enteros positivos.
DOM_IVA :	Números enteros positivos.
DOM_DTO :	Números enteros entre 0 y 100.

Cabecera de **FACTURAS**:

{(codfac: DOM_CODFAC), (fecha: DOM_FECHA),
(codcli: DOM_CODCLI), (iva: DOM_IVA), (dto: DOM_DTO }

Es una relación de **grado 5**.

Una de las tuplas de **FACTURAS** es:

{(codfac: 38), (fecha: 10/1/99), (codcli: 93), (iva: 0), (dto: 0) }

que es la misma tupla que esta otra:

{(fecha: 10/1/99), (iva: 0), (dto: 0), (codfac: 38), (codcli: 93) }

Propiedades de las Relaciones

- Cada relación tiene un nombre distinto.
- Los valores de los atributos son atómicos (relaciones normalizadas).

Antes (grado 2)

Codfac	Detalle		
	Línea	Codart	Cant
23	1	L8763	300
	2	TNF23	200
	3	UCM8	400
28	1	ND43	300
	2	UCM8	400
32	1	TNF23	200

Relación no normalizada (con grupos repetitivos)

Después (grado 3)

Codfac	Línea	Codart	Cant
23	1	L8763	300
23	2	TNF23	200
23	3	UCM8	400
28	1	ND43	300
28	2	UCM8	400
32	1	TNF23	200

Relación normalizada (1ª forma normal)

los valores del atributo DETALLE no son atómicos, son relaciones

- Cada atributo tiene un nombre distinto.
- Los atributos no están ordenados.
- No hay tuplas duplicadas.
- Las tuplas no están ordenadas.

Tipos de relaciones

- ***Relaciones base***: con nombre, reales, autónomas (parte directa de la base de datos).

```
CREATE TABLE PROVINCIAS
( CODPRO    VARCHAR2(2),
  NOMBRE    VARCHAR2(30),
CONSTRAINT CP_PROVINCIAS PRIMARY KEY (CODPRO));
```

- ***Vistas***: con nombre, derivadas, virtuales.

```
CREATE VIEW COM_VAL
    AS SELECT PU.CODPUE,PU.NOMBRE,PR.CODPRO,PR.NOMBRE PROVINCIA
        FROM  PUEBLOS PU, PROVINCIAS PR
        WHERE PU.CODPRO=PR.CODPRO
        AND   PR.CODPRO IN ('03','12','46');
```

- ***Instantáneas***: con nombre, derivadas, reales (sólo lectura), refresco periódico.

```
CREATE SNAPSHOT FAC_VLC
STORAGE INITIAL 50K NEXT 50K
REFRESH FAST NEXT NEXT_DAY(TRUNC(SYSDATE),'MONDAY')
AS
   SELECT * FROM VLC.FACTURAS;
```

- ***Resultados de consultas***: con o sin nombre, no persisten en la base de datos.
- ***Resultados intermedios***: sin nombre, no persisten en la base de datos.
- ***Resultados temporales***: con nombre, se destruyen automáticamente.

Claves

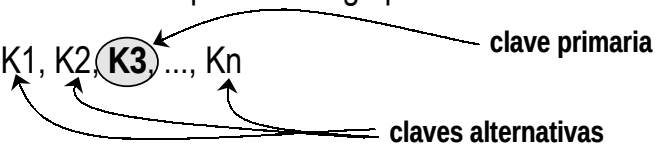
Superclave: identifica de modo único las tuplas de una relación.

Clave candidata: superclave en la que ninguno de sus subconjuntos es una superclave de la relación. Debe satisfacer:

1. ***Unicidad***
2. ***Irreducibilidad (minimalidad)***

Clave primaria: clave candidata que se escoge para identificar las tuplas de modo único.

Claves candidatas: K1, K2, **K3**, ..., Kn



Importancia de las claves primarias: constituyen el mecanismo de direccionamiento de tuplas de un sistema relacional

Claves

Clave ajena: sus valores deben coincidir con los de la clave primaria de otra relación
→ representa una ***relación*** entre datos a modo de ***referencia***.

Ddiagramas referenciales:

FACTURAS —codcli→ CLIENTES **"cliente al que corresponde la factura"**

Observaciones

- Clave ajena y clave primaria a la que referencia : mismo dominio.
- Camino referencial de Rn a R1 : Rn → .... → R2 → R1
- Auto-referencia : R1 → R1
- Ciclo referencial sobre Rn : Rn → .... → R2 → R1 → Rn
- Las claves ajenas de las relaciones base necesitan, a veces, contener nulos.

nulo: ausencia de valor

El requisito de que los valores de clave ajena coincidan con los valores de una determinada clave primaria es el "pegamento" que mantiene unida la base de datos.

4. Reglas de Integridad

Regla de integridad : restricción que debe cumplirse sobre una BD en todos sus estados.

Reglas de negocio : reglas de integridad específicas de cada base de datos

Reglas de integridad generales :

- ***Regla de integridad de entidades*** (amiga de las claves primarias).
- ***Regla de integridad referencial*** (amiga de las claves ajenas).

Además existen las ***restricciones de dominios*** : al definir cada atributo sobre un dominio, se impone una restricción sobre el conjunto de valores permitidos para cada atributo.

Regla de Integridad de Entidades

"Ninguno de los atributos que componen la clave primaria puede ser nulo."

¡¡En una base de datos relacional nunca se almacena información de algo que no se puede identificar!!

Observaciones:

- La regla se aplica a las relaciones base (parte directa de la base de datos).
- La regla se aplica sólo a la clave primaria (no a las claves alternativas).

Regla de Integridad Referencial

"Si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser todos nulos."

La regla de integridad referencial se enmarca en términos de estados de la base de datos: nos dice lo que es un estado ilegal ¡¡pero no nos dice cómo podemos evitarlo!!

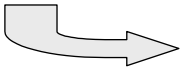
¿Qué hacer si estando en un estado legal, llega una operación que conduce a un estado ilegal? Existen dos opciones:

- **Rechazar** la operación.
- **Aceptar** la operación y realizar **operaciones adicionales** compensatorias que conduzcan a un estado legal.

Reglas para las claves ajenas

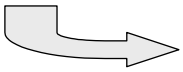
Regla de los nulos: ¿Tiene sentido que la clave ajena acepte nulos?

Regla de borrado: ¿Qué hacer si se intenta borrar la tupla referenciada por la clave ajena?



- **Restringir**
- **Propagar**
- **Anular**

Regla de modificación: ¿Qué hacer si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?



- **Restringir**
- **Propagar**
- **Anular**

Ejemplos

			NULOS	BORRADO	MODIFICACION
PUEBLOS	codpro →	PROVINCIAS	no	propagar	Propagar
CLIENTES	codpue →	PUEBLOS	no	restringir	Propagar
FACTURAS	codcli →	CLIENTES	sí	anular	Propagar
LINEAS_FAC	codfac →	FACTURAS	no	propagar	Propagar
LINEAS_FAC	codart →	ARTICULOS	sí	anular	Propagar

Pregunta: ***¿se puede borrar la provincia de Castellón ?***

PIEZA(codpieza,descrip)
COMPONENTE(codpieza,codcomponente,cant)

			NULOS	BORRADO	MODIFICACION
COMPONENTE	codpieza →	PIEZA			
COMPONENTE	codcomponente →	PIEZA			

4. Lenguajes Relacionales

Álgebra relacional y ***cálculo relacional*** : la base de los lenguajes relacionales (Codd).

Álgebra relacional → lenguaje ***procedural***

Cálculo relacional → lenguaje ***no procedural***

} ¡¡pero ambos lenguajes son ***equivalentes***!!

AVISO: son lenguajes formales no muy "amigables", pero se deben estudiar porque sirven para comprender las operaciones básicas de los lenguajes de manejo de datos.

Un lenguaje es ***relacionalmente completo*** si permite obtener cualquier relación que se pueda derivar mediante el álgebra relacional (o el cálculo relacional).

Algebra Relacional

Restricción (selección): `R WHERE condición`

Obtiene las tuplas de R que cumplen la condición especificada. Esta condición es una comparación en la que aparece al menos un atributo de R; puede ser una combinación booleana de varias de estas comparaciones.

Piezas rojas con peso mayor de 15: `P WHERE COLOR='rojo' AND PESO>15`

P#	PNOMBRE	COLOR	PESO	CIUDAD
P2	perno	rojo	17	Londres
P6	engrane	rojo	19	París

Proyección: R [a_i, ..., a_k]

Obtiene una relación que contiene un subconjunto vertical de R, extrayendo los valores de los atributos especificados y eliminando tuplas duplicadas.

Proveedores y piezas que envían:

SP[S#,P#]

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

Ciudades en las que se almacena alguna pieza:

P[CIUDAD]

CIUDAD
París
Londres
Roma

Producto cartesiano: R TIMES S

Obtiene una relación cuyas tuplas están formadas por la concatenación de todas las tuplas de R con todas las tuplas de S.

Datos de los proveedores que envían la pieza P1 y cantidad que envían:

SP WHERE P#='P1'

S#	P#	CANT
S1	P1	300
S2	P1	300

S TIMES (SP WHERE P#='P1')

S.S#	SNOMBRE	ESTADO	CIUDAD	SP.S#	P#	CANT
S1	Salazar	20	Londres	S1	P1	300
S2	Jaimes	10	París	S1	P1	300
S3	Bernal	30	París	S1	P1	300
S4	Corona	20	Londres	S1	P1	300
S5	Aldana	30	Atenas	S1	P1	300
S1	Salazar	20	Londres	S2	P1	300
S2	Jaimes	10	París	S2	P1	300
S3	Bernal	30	París	S2	P1	300
S4	Corona	20	Londres	S2	P1	300
S5	Aldana	30	Atenas	S2	P1	300

¿era esto lo que queríamos?

Para quedarse sólo con los datos de los proveedores que envían la pieza P1, hay que hacer una restricción:

```
(S TIMES (SP WHERE P#='P1')) WHERE S.S#=SP.S#
```

S.S#	SNOMBRE	ESTADO	CIUDAD	SP.S#	P#	CANT
S1	Salazar	20	Londres	S1	P1	300
S2	Jaimes	10	París	S2	P1	300

Mediante una proyección podemos quedarnos solamente con los atributos que nos interesan:

```
((S TIMES (SP WHERE P#='P1')) WHERE S.S#=SP.S#) [S.S#,SNOMBRE,ESTADO,CIUDAD,CANT]
```

S#	SNOMBRE	ESTADO	CIUDAD	CANT
S1	Salazar	20	Londres	300
S2	Jaimes	10	París	300

Unión: `R UNION S`

Obtiene una relación cuyas tuplas son las que se encuentran en R, o en S, o en ambas relaciones a la vez. Para poder realizar esta operación, R y S deben ser compatibles para la unión.

Dos relaciones son **compatibles para la unión** si ambas tienen la misma cabecera, es decir, si tienen el mismo número de atributos, se llaman igual y se encuentran definidos sobre los mismos dominios.

Ciudades en las que hay proveedores o se almacenan piezas:

`S[CIUDAD]`

CIUDAD
Londres
París
Atenas

`P[CIUDAD]`

CIUDAD
París
Londres
Roma

`S[CIUDAD] UNION P[CIUDAD]`

CIUDAD
París
Londres
Roma
Atenas

Diferencia: `R MINUS S`

Obtiene una relación que tiene las tuplas que se encuentran en R y no se encuentran en S. Para poder realizar esta operación, R y S deben ser compatibles para la unión.

Ciudades en las que hay proveedores y no se almacenan piezas:

`S[CIUDAD]`

CIUDAD
Londres
París
Atenas

`P[CIUDAD]`

CIUDAD
París
Londres
Roma

`S[CIUDAD] MINUS P[CIUDAD]`

CIUDAD
Atenas

Ciudades en las que se almacenan piezas y no hay proveedores:

`S[CIUDAD]`

CIUDAD
Londres
París
Atenas

`P[CIUDAD]`

CIUDAD
París
Londres
Roma

`P[CIUDAD] MINUS S[CIUDAD]`

CIUDAD
Roma

Join (concatenación): `R JOIN S`

Obtiene una relación cuyas tuplas son todas las tuplas de R concatenadas con todas las tuplas de S que en los atributos comunes (los que se llaman igual) tienen los mismos valores. Estos atributos comunes aparecen una sola vez en el resultado.

Datos de los proveedores que envían la pieza P1 y cantidad que envían:

```
S JOIN (SP WHERE P#='P1')
```

S#	SNOMBRE	ESTADO	CIUDAD	P#	CANT
S1	Salazar	20	Londres	P1	300
S2	Jaimes	10	París	P1	300

Mediante una proyección podemos quedarnos solamente con los atributos que nos interesan:

```
(S JOIN (SP WHERE P#='P1')) [S#,SNOMBRE,ESTADO,CIUDAD,CANT]
```

Nótese que esta expresión obtiene el mismo resultado que la expresión:

```
((S TIMES (SP WHERE P#='P1')) WHERE S.S#=SP.S#) [S.S#,SNOMBRE,ESTADO,CIUDAD,CANT]
```

Porque el *join* es básicamente un producto cartesiano y una restricción de igualdad sobre los atributos comunes.

Outer—Join: `R JOIN S (+)`

El outer—join es un *join* en el que las tuplas de R que no tienen valores en común con ninguna tupla de S, también aparecen en el resultado.

Datos de todos los proveedores con las piezas que envían: `S JOIN SP(+)`

S#	SNOMBRE	ESTADO	CIUDAD	P#	CANT
S1	Salazar	20	Londres	P1	300
S1	Salazar	20	Londres	P2	200
S1	Salazar	20	Londres	P3	400
S1	Salazar	20	Londres	P4	200
S1	Salazar	20	Londres	P5	100
S1	Salazar	20	Londres	P6	100
S2	Jaimes	10	París	P1	300
S2	Jaimes	10	París	P2	400
S3	Bernal	30	París	P2	200
S4	Corona	20	Londres	P2	200
S4	Corona	20	Londres	P4	300
S4	Corona	20	Londres	P5	400
S5	Aldana	30	Atenas		

Cuando en ambas relaciones hay tuplas que no se pueden concatenar y se desea que en el resultado aparezcan también todas estas tuplas, tanto las de una relación como las de la otra, se utiliza el **outer—join completo** : `R(+) JOIN S(+)`

Parejas de piezas azules y proveedores de la misma ciudad. Las piezas y proveedores que queden desparejados también se deben obtener:

```
(P WHERE COLOR='azul')(+) JOIN S(+)
```

P#	PNOMBRE	COLOR	PESO	CIUDAD	S#	SNOMBRE	ESTADO
P3	birlo	Azul	17	Roma			
P5	leva	Azul	12	París	S2	Jaimes	10
P5	leva	Azul	12	París	S3	Bernal	30
				Londres	S1	Salazar	20
				Londres	S4	Corona	20
				Atenas	S5	Aldana	30

Intersección: `R INTERSECT S`

Obtiene una relación que contiene las tuplas de R que también se encuentran en S. Para poder realizar esta operación, R y S deben ser compatibles para la unión.

Ciudades en las que hay proveedores y se almacenan piezas:

`S[CIUDAD]`

CIUDAD
Londres
París
Atenas

`P[CIUDAD]`

CIUDAD
París
Londres
Roma

`S[CIUDAD] INTERSECT P[CIUDAD]`

CIUDAD
Londres
París

`R INTERSECT S` obtiene el mismo resultado que `R MINUS (R MINUS S)`

División: `R DIVIDEBY S`

Suponiendo que la cabecera de R es el conjunto de atributos A y la cabecera de S es el conjunto de atributos B, tales que B es un subconjunto de A. Si consideramos C como el subconjunto de los atributos de A que no están en B (A - B), la división obtiene una relación cuya cabecera es el conjunto de atributos C y que contiene las tuplas de R que están acompañadas de todas las tuplas de S.

Proveedores que suministran todas las piezas que se almacenan en Londres:

`(P WHERE CIUDAD='Londres')[P#]`

P#
P2
P4

`SP[S#,P#] DIVIDEBY (P WHERE CIUDAD='Londres')[P#]`

S#
S1
S4

Agrupación (resumen):

SUMMARIZE R GROUPBY(a_i,...,a_k) ADD cálculo AS atributo

Esta operación agrupa las tuplas de R que tienen los mismos valores en los atributos especificados y realiza un cálculo sobre los grupos obtenidos (SUM, AVG, MAX, MIN, COUNT). La relación resultado tiene como cabecera los atributos por los que se ha agrupado y el cálculo realizado con el nombre especificado en atributo.

Número total de unidades de piezas suministradas por cada proveedor:

SUMMARIZE SP GROUPBY (S#) ADD SUM(CANT) AS CANT_TOTAL

S#	CANT_TOTAL
S1	1300
S2	700
S3	200
S4	900

Número medio de unidades suministradas por envío:

SUMMARIZE SP GROUPBY () ADD AVG(CANT) AS MEDIA

MEDIA
258

Cálculo Relacional

El cálculo relacional se basa en una rama de la lógica que es el ***cálculo de predicados***.

Una característica del cálculo relacional es que se utilizan ***variables***, que toman valores de tuplas de una relación, o bien toman valores en el dominio de un atributo.

cálculo relacional ***orientado a tuplas*** → ***variables tupla***

```
RANGE OF SX IS S
SX.S# WHERE SX.CIUDAD='PARIS'
```

cálculo relacional ***orientado a dominios*** → ***variables dominio***

```
RANGE OF SX IS S#
SX WHERE S(S#:SX,CIUDAD:'PARIS')
```

Variables tupla

```
RANGE OF T IS X1, X2, ..., Xn
```

Cada **Xi** es una relación con nombre o una expresión del cálculo entre paréntesis, de modo que cada **Xi** se evalúa a una relación **Ri**.

La ***variable tupla*** **T** toma valores en la unión de las relaciones **Ri**, **i=1:n** (las relaciones **Ri** deben ser compatibles para la unión).

```
RANGE OF SX IS S
RANGE OF PX IS P
RANGE OF SPX IS SP
RANGE OF CIUDADX IS (SX.CIUDAD),(PX.CIUDAD)
RANGE OF SP4X IS (SX WHERE ∃SPX (SPX.S#=SX.S# AND SPX.P#='P4'))
```


Expresiones del cálculo relacional

lista_de_objetivos [**WHERE** fbf]

lista_de_objetivos: objetivos separados por comas
objetivo: [**X =**] **T.A** **T** : variable tupla
A : atributo de la relación asociada a **T**
X : nuevo nombre del atributo
fbf: fórmula bien formada (comparación o combinación booleana de comparaciones)

Evaluación

Sea **T**, **U**, ..., **V** el conjunto de variables tupla especificadas en la lista de objetivos.
Sean **X1**, **X2**, ... , **Xn** los nombres de los atributos a obtener en el resultado.

(1) Se forma el producto cartesiano **T** × **U** × ... × **V** (**T**, **U**, ..., **V** toman todos los valores posibles de sus rangos).
(2) Se eliminan (restricción) las tuplas del producto anterior que no satisfacen la fbf del **WHERE**, si lo hay.
(3) Se proyecta el resultado anterior sobre **X1**, **X2**, ..., **Xn**.

```
SX.SNOMBRE,SPX.P# WHERE (SX.S#=SPX.S# AND SPX.CANT>100)
```

Evaluación de las fbf

E y G son dos fbf

E	G	NOT E	F AND G	E OR G	IF E THEN G (= NOT E OR G)
V	V	F	V	V	V
V	F	F	F	V	F
F	V	V	F	V	V
F	F	V	F	F	V

∃**x** (E) : se evalúa a verdadero si al sustituir en E la variable tupla **x** por alguna tupla de su rango, se tiene que E se evalúa a verdadero.

```
PX.P#,PX.PNOMBRE WHERE ∃SPX (SPX.P#=PX.P# AND SPX.CANT>100)
```

∀**x** (E) : se evalúa a verdadero si al sustituir en E la variable tupla **x** por cada tupla de su rango, se tiene que E se evalúa a verdadero para cada una de ellas.

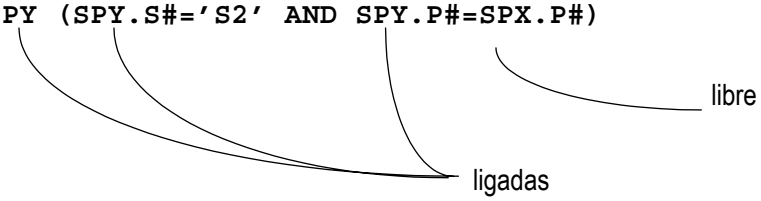
```
PX.P#,PX.PNOMBRE WHERE ∀SPX (IF SPX.P#=PX.P# THEN SPX.CANT>100)
```


Algo más sobre variables tupla

Una ocurrencia de una variable tupla es ***ligada*** si sobre ella actúa un cuantificador (∃,∀), o está dentro del alcance de un cuantificador que actúa sobre ella misma.

Cualquier otra ocurrencia que no cumpla lo anterior es ***libre***.

```
SPX.S# WHERE ∃SPY (SPY.S#='S2' AND SPY.P#=SPX.P#)
```



La expresión anterior es equivalente a:

```
SPX.S# WHERE ∃SPZ (SPZ.S#='S2' AND SPZ.P#=SPX.P#)
```

Cálculo relacional VS álgebra relacional

El **algoritmo de reducción de Codd** convierte cualquier expresión del cálculo relacional en una expresión del álgebra relacional.

(1) Obtener el rango de cada variable tupla, aplicando las restricciones del **`WHERE`** que sea posible (rangos restringidos).

(2) Construir el producto cartesiano de los rangos obtenidos en el paso (1).

(3) Restringir el producto cartesiano del paso (2) usando las restricciones de join del **`WHERE`**.

(4) Aplicar los cuantificadores de derecha a izquierda del siguiente modo:

- ♦ ∃**`RX`** : proyectar el resultado actual para eliminar todos los atributos de la relación asociada a **`RX`**.
- ♦ ∀**`RX`** : dividir el resultado actual entre la relación que contiene el rango restringido asociado a **`RX`**.

(5) Proyectar el resultado del paso (4) de acuerdo con las especificaciones de la lista de objetivos.

5. Vistas

Arquitectura de tres niveles: una ***vista externa*** corresponde a la estructura de la base de datos tal y como la ve un usuario en particular.

Modelo relacional: una ***vista*** es una ***relación virtual*** (una relación que en realidad no existe como tal) que se construye realizando operaciones como las del álgebra relacional: restricciones, proyecciones, concatenaciones, etc. a partir de las relaciones base de la base de datos.

Esquema externo : puede tener tanto relaciones base como vistas derivadas de las relaciones base de la base de datos.

Una vista es una relación virtual que se produce cuando un usuario la consulta.

Cuando un usuario realiza un cambio sobre la vista (no todo tipo de cambios están permitidos), este cambio se realiza sobre las relaciones de las que se deriva.

Utilidad de las vistas:

- Proporcionan seguridad ocultando partes de la base de datos a ciertos usuarios.
- Permiten que los usuarios accedan a los datos en el formato que ellos desean o necesitan.
- Se pueden simplificar operaciones sobre las relaciones base que son complejas.
- Se puede utilizar una vista para ofrecer un esquema externo a un usuario de modo que éste lo encuentre "familiar"
    - Los atributos se pueden renombrar.
    - Se puede cambiar el orden en que se visualizan las columnas.
    - Se pueden hacer restricciones para que sólo se vea un subconjunto horizontal de la relación.
    - Etc.

Las vistas proporcionan ***independencia de datos a nivel lógico***:

- Si se añade un atributo a una relación, los usuarios no se percatan de su existencia si sus vistas no lo incluyen.
- Si una relación existente se reorganiza o se divide en varias relaciones, se pueden crear vistas para que los usuarios la sigan viendo como al principio.

Condiciones bajo las cuales se determina si se permite realizar una actualización sobre una vista:

- Se permiten las actualizaciones de vistas que se definen mediante una consulta simple sobre una sola relación base y que contienen la clave primaria de la relación base.
- No se permiten las actualizaciones de vistas que se definen sobre varias relaciones base.
- No se permiten las actualizaciones de vistas definidas con operaciones de agrupamiento (GROUPBY).

Base de datos de proveedores y piezas

S (proveedores)

S#	SNOMBRE	ESTADO	CIUDAD
S1	Salazar	20	Londres
S2	Jaimes	10	París
S3	Bernal	30	París
S4	Corona	20	Londres
S5	Aldana	30	Atenas

P (piezas)

P#	PNOMBRE	COLOR	PESO	CIUDAD
P1	tuerca	verde	12	París
P2	perno	rojo	17	Londres
P3	birlo	azul	17	Roma
P4	birlo	rojo	14	Londres
P5	leva	azul	12	París
P6	engrane	rojo	19	París

SP (envíos)

S#	P#	CANT
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400