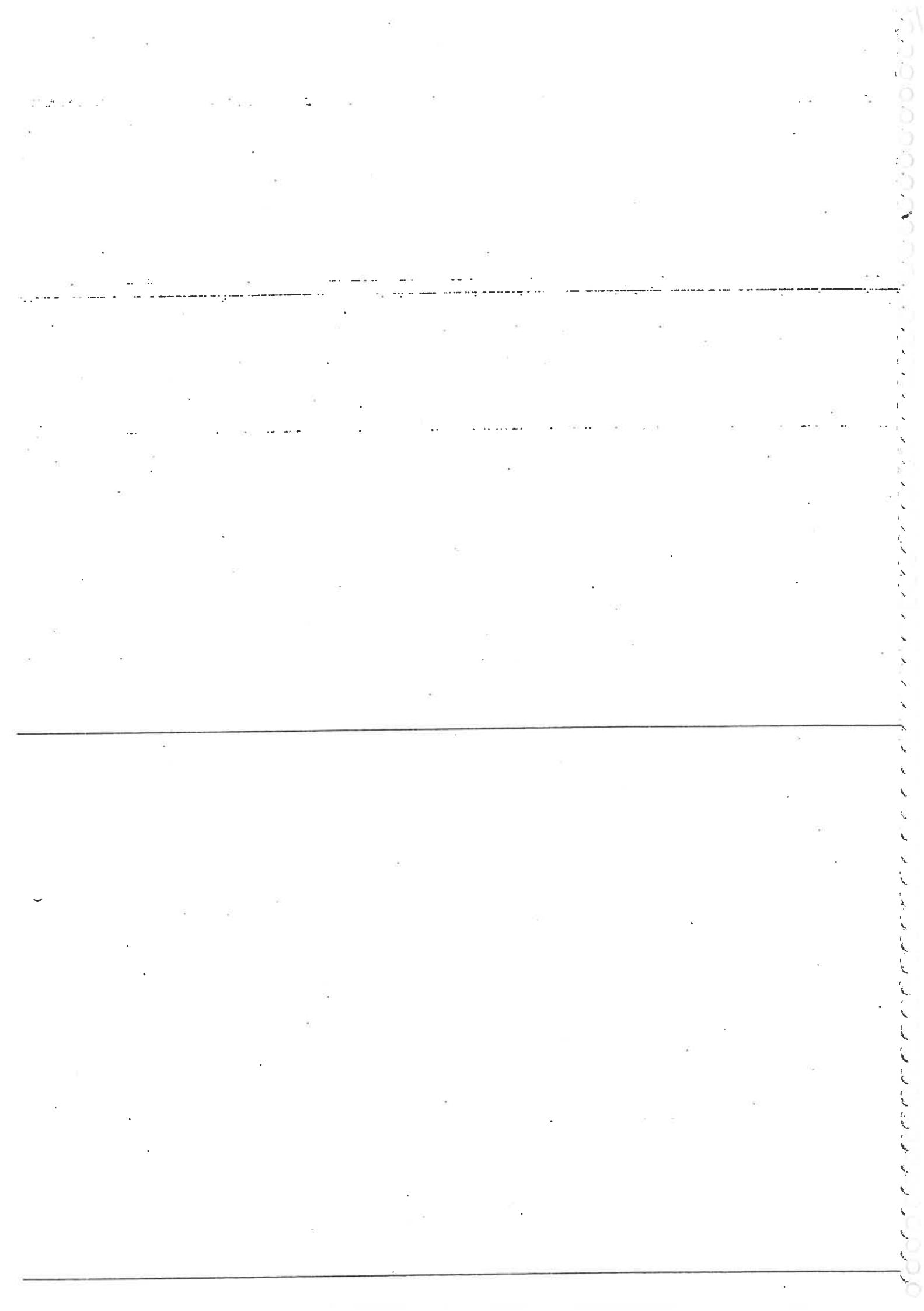


# **BASES DE DATOS**

---

**Grado Ingeniería Informática**



# **BASES DE DATOS**

**Curso:** 2012/13  
**Titulación:** Grado en Ingeniería Informática  
**Moodle:** <http://campusvirtual.ull.es/1213m2>  
**Profesor:** Jesús M. Jorge Santiso (<http://www.jjorge.es>)

**Descripción:** En esta asignatura se estudian los fundamentos de los sistemas de bases de datos basados en el modelo relacional. Se introducen los conceptos requeridos para diseñar una base de datos relacional, utilizar de modo interactivo el SQL y escribir programas de aplicaciones utilizando PL/SQL y SQL empotrado en C/C++.

## **Objetivos específicos:**

- 1.- Comprendión de las características del modelo relacional (Tablas, t-uplas, condiciones de integridad, ...).
- 2.- Aprendizaje y uso de los principales lenguajes de consulta (teóricos y comerciales) para el modelo relacional. (Algebra relacional, Cálculo relacional de t-uplas y de dominios, SQL-3 )
- 3.- Programación de aplicaciones en PL/SQL y SQL empotrado en C/C++.
- 4.- Aprender a diseñar una base de datos relacional.

## **Programa de Teoría**

1. INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS.
2. EL MODELO RELACIONAL.
3. SQL.
4. GESTIÓN DE TRANSACCIONES.
5. DISEÑO DE BASES DE DATOS.
6. DESARROLLO DE APLICACIONES DE BASES DE DATOS.

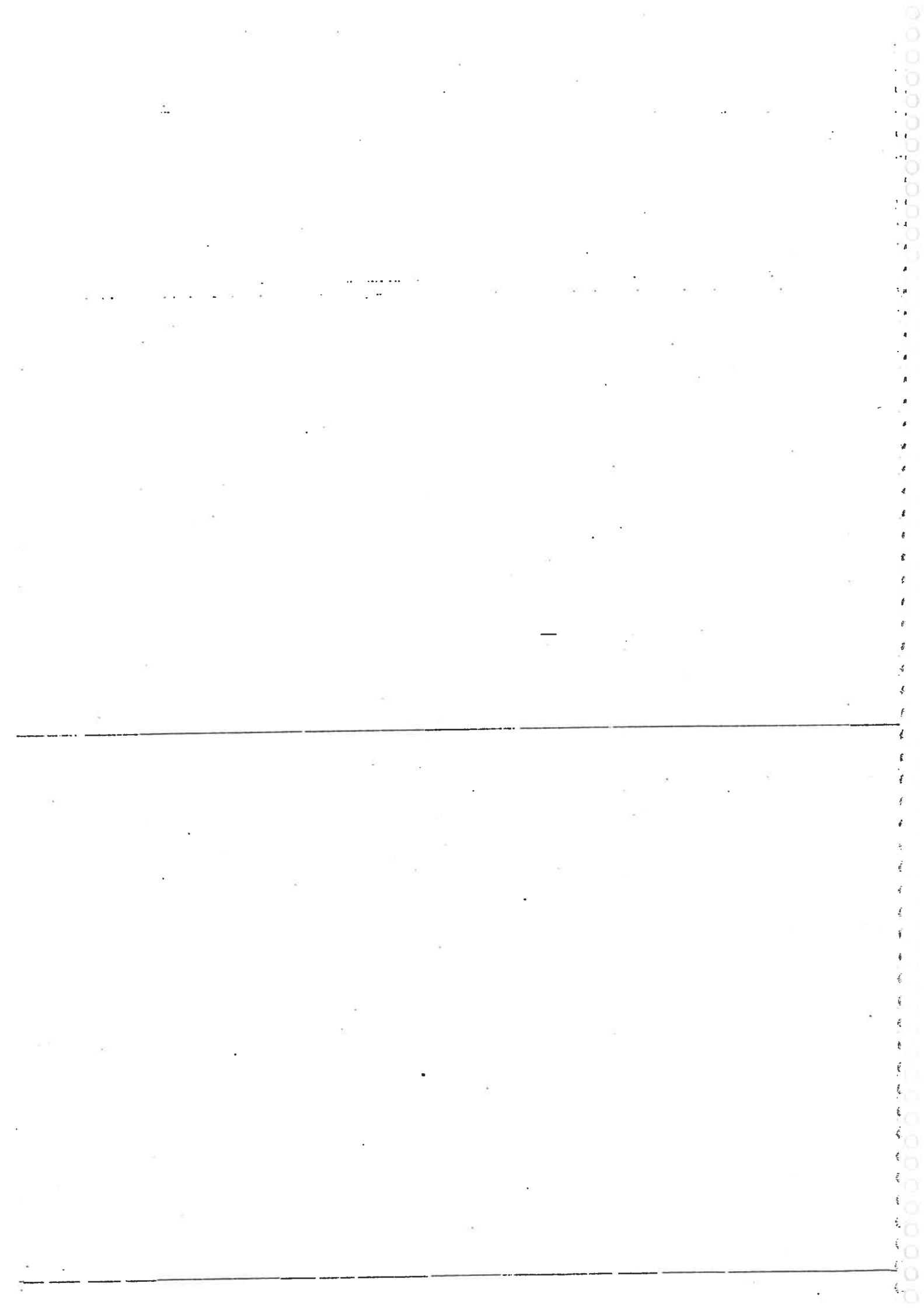
## **Programa de Prácticas**

1. PROCESADOR DE CONSULTAS INTERACTIVO SQLPLUS DE ORACLE.
2. ANÁLISIS DEL CASO PRÁCTICO GESTIÓN DOCENTE.
3. CREACIÓN Y CARGA INICIAL DE LA BASE DE DATOS.
4. CONSULTAS ELEMENTALES.
5. CONSULTAS USANDO MÚLTIPLES TABLAS.
6. CONSULTAS AVANZADAS.
7. CONTROL DE TRANSACCIONES.
8. DEFINICIÓN DE DATOS Y AUTORIZACIONES.
9. FUNDAMENTOS DE PL/SQL.
10. DISPARADORES.

## **Bibliografía:**

- [1] Elmasri, R. y Navathe, S. Fundamentos de Sistemas de Bases de Datos, Addison Wesley, Tercera edición, 2002.
- [2] Silberschatz, A., Korth, H. y Sudarshan, S. Fundamentos de Bases de Datos, McGraw-Hill, Quinta edición, 2006.
- [3] Rivero Cornelio; E. Bases de Datos Relacionales: Fundamentos y Diseño Lógico, Paraninfo, Universidad Pontificia Comillas, 2005.

**Método de evaluación:** Examen escrito (60%) + Prácticas (40%).



## TEMA 1: INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS

### 1.1. El procesamiento tradicional de datos. (sistemas de ficheros).

Proliferación de ficheros que contienen datos específicos para programas de aplicaciones, pero que tienen cierto desorden.

P. de aplicaciones (altas, bajas, ...) con un fichero asociado.

Se ponía demasiado énfasis a las aplicaciones y no a los datos.

Inconvenientes:

- Redundancia de datos. Datos repetidos (en contra de la idea del dato único actual). Se encarecen las actualizaciones y se necesita más almacenamiento.
- Inconsistencia. Las copias de los datos no concuerdan entre sí.
- Dificultad en el acceso a la información. Habría que crear un nuevo programa de aplicaciones.
- Dependencia de datos. Los datos dependían del formato en el que estaban almacenados y cada fichero tenía un formato distinto.
- Concurrencia. Por ejemplo si se actualizan los datos simultáneamente.
- Confidencialidad de la información. No toda la información debe de estar disponible para todo el mundo.
- Integridad. Condiciones que tienen que satisfacer los datos para que sean válidos. Ejemplo: una cuenta sólo puede tener dos titulares.
- Atomicidad. Las transacciones tienen que ser tratadas como un átomo, algo indivisible.
- Reestructurar la información muy compleja. Añadir un nuevo campo o modificar el formato de algún campo.  
Reconvertir los datos o reescribir programas.

Todos los inconvenientes son independientes de las aplicaciones.

## 1.2 Sistemas de Bases de Datos.

Lo importante son los datos, no las aplicaciones.

Los sistemas de bases de datos permiten trabajar con grandes cantidades de datos. Pretendían crear un trabajo cómodo, flexible, que permitiese un trabajo mejor con la base de datos.

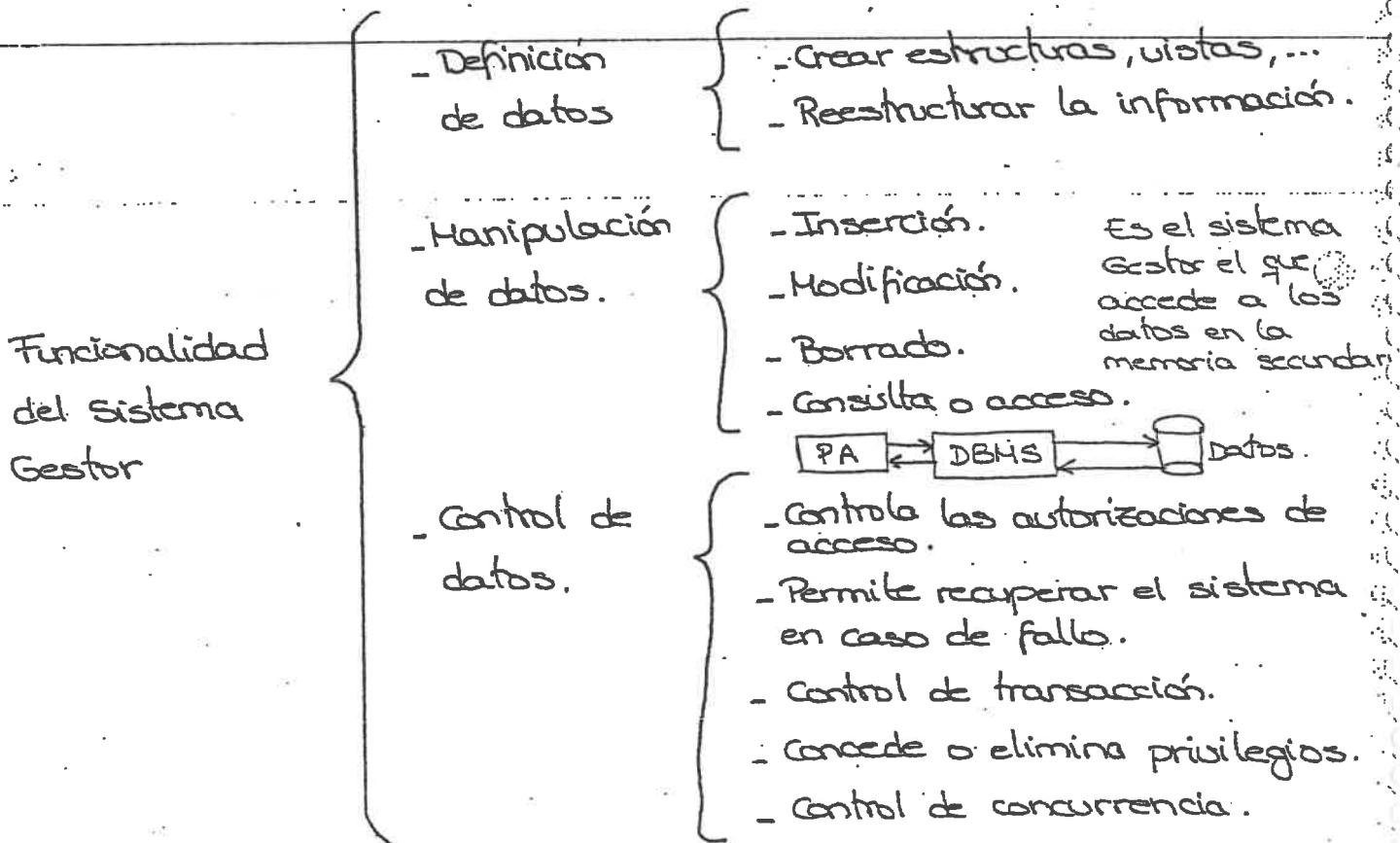
Las bases de datos son un conjunto de datos relacionados que dan soporte a un conjunto de aplicaciones.

Tienen que tener una redundancia controlada.

Ventajas: Son lo que antes eran las debilidades. Los datos son controlados por un sistema genérico llamado sistema gestor de la base de datos (SGBD). (DBMS) Data Base Management System.

Inconvenientes: Las licencias de uso son muy caras. Consumen muchos recursos, lo normal es tener servidores dedicados.

Instalar un sistema gestor es muy latoso y hay que tener unos conocimientos elevados.



### 1.3 - Abstracción de la información e independencia de datos.

Para que sea útil el sistema gestor debe almacenar la información de forma eficiente, ocupando el menor espacio posible.

La abstracción de la información oculta la complejidad del sistema gestor.

La independencia de los datos pretende no depender del formato en el que los datos están escritos, y es la capacidad del sistema gestor para hacer que las referencias a los datos en las aplicaciones no se vean afectadas por cambios en el diseño lógico de los datos.

Arquitectura ANSI / X3 / SPARC.

#### → Niveles de Abstracción:

1- Nivel físico o interno. → Esquema físico.

Cómo están almacenados los datos en memoria secundaria.

Si tengo uno o varios ficheros. De qué tipos son los índices. Cómo es el tipo de datos. Así no hay abstracción de la información.

2- Nivel conceptual: → Esquema conceptual.

Se describen todos los datos desde un punto de vista lógico.

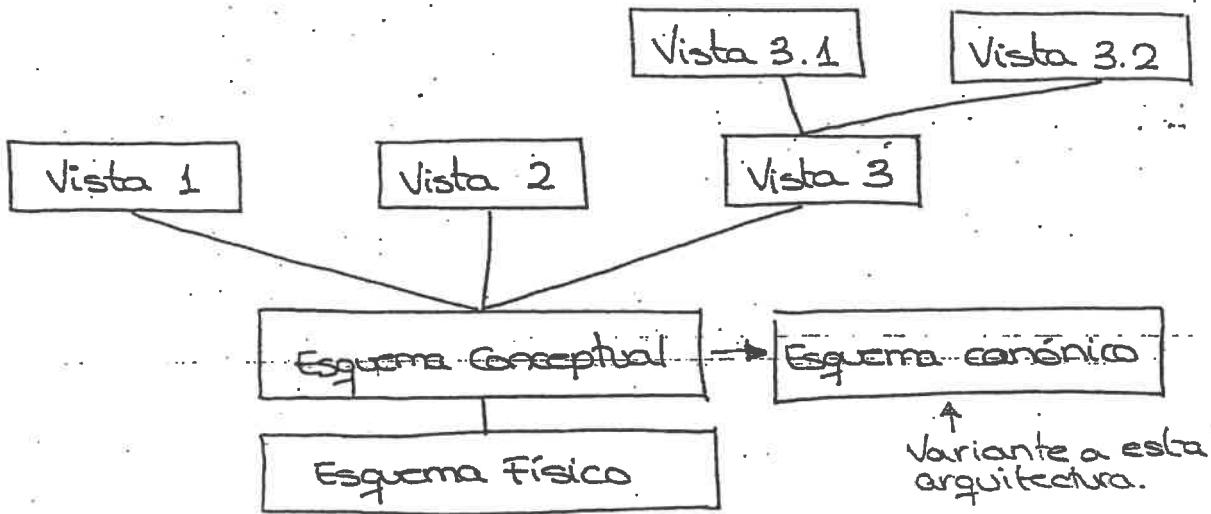
3- Nivel externo o de visión: → Subesquemas o vistas.

Se describen sólo una parte de los datos y de las relaciones entre los datos. Puede tener tantas vistas (descripciones) de la base de datos.

Una vista se puede crear a partir del nivel conceptual o a partir de otra vista.

Las vistas se utilizan para facilitar el trabajo al programador.

Esquema canónico: Utilizando el sistema gestor hablamos de lo que se implementa.



Cualquiera de las dos arquitecturas proporciona la independencia de datos.

La independencia de datos es la capacidad que ofrece el sistema gestor de modificar un esquema sin tener que hacerlo en los esquemas de niveles superiores.

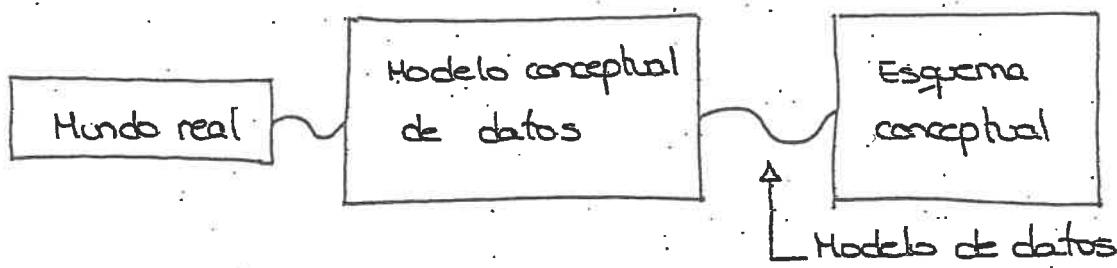
Hay dos tipos de independencia de datos:

- Física: Capacidad de cambiar el esquema físico sin tener que cambiar el esquema conceptual.

- Lógica: Capacidad de cambiar el esquema conceptual sin tener que cambiar los subesquemas o vistas.

La más sencilla de lograr para el sistema gestor es la independencia física. Lo más fácil es aislar los programas para los cambios de formatos.

## 1.4 Modelos de Datos.



Los modelos de datos sirven para representar el esquema de datos.

El modelo conceptual es un conjunto de ideas y conceptos interrelacionados en la mente del analista y que forman una imagen del mundo real.

El modelo de datos es el lenguaje en el que el analista describe el modelo conceptual. Son herramientas para describir los datos, relaciones de los datos. Sistemas formales porque hay un conjunto de estructuras dadas. Son abstractos porque el significado de los operadores independientes del sistema de datos.

Cuanto más capacidad expresiva son tanto mejores.

Los modelos de datos tienen 3 tipos de componentes:

- Estructuras de datos: Permiten crear objetos para almacenar información.
- Operadores: Permiten manipular estructuras de datos.
- Condiciones de Integridad: qué tienen que satisfacer los datos.

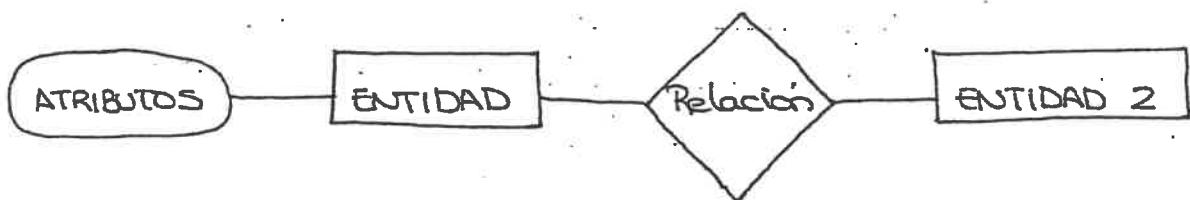
Los modelos de datos se clasifican en las siguientes categorías:

- Lógicos
  - Orientados a registros
    - Jerárquico.
    - En red.
    - Relacional.
  - Orientados a objetos → Entidad/Relación.
- Físicos: Representa los datos en el nivel físico.  
(están en desuso).

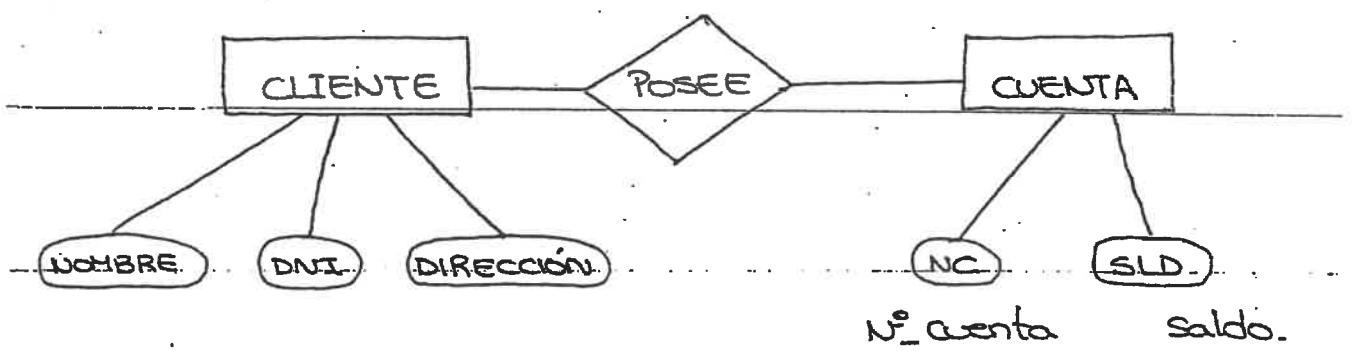
#### 14.1 - Modelos Entidad / Relación.

La realidad se percibe como a un conjunto de objetos llamados entidades de la misma o de distinta clase. Entre las entidades existen relaciones. Una se distingue de otras en función de las propiedades de los atributos.

Las entidades se representan gráficamente mediante rectángulos, los atributos mediante elipses y las relaciones con rombos.



Los atributos dentro de una entidad no se repiten. Pero de diferentes entidades si pueden tener un mismo atributo.



Las relaciones pueden ser de cuatro tipos de correspondencia:

- Uno a uno.
- Uno a muchos.
- Muchos a uno.
- Muchos a muchos.

100  
100  
100

#### 14.2. Modelo Relacional de datos.

El analista percibe la realidad como un conjunto de registros que se organizan en forma de tablas. Cada una tiene un nombre y tienen columnas que corresponden a los atributos. Al crear una tabla ponemos entre paréntesis los valores de los atributos.

Las filas son unas asociaciones de valores asociadas a esa tabla.

CLIENTE	DNI	NOMBRE	DIR	CUENTA	NC	SLD
1111	Juan	S/C			10	1.000
2222	Pedro	La legua			20	2.000
3333	Carlos	Tacoronte				

El contenido de la base de datos es variable y se denomina instancia, es volátil y cambiante.

Tiene que haber una tabla que relacione las dos tablas anteriores. La denominaremos Possee.

POSEE	DNI	NC	SLD	
1111	10	1.000		Hay un atributo más, ocasionaría redundancia.
2222	10	1.000		
1111	20	2.000		

Hay una redundancia mínima, pero es necesaria, pues nos permite relacionar unas tablas con otras.

Las tablas se representarían de la siguiente forma:

CLIENTE (DNI, NOMBRE, DIR)

CUENTA (NC, SLD)

POSEE (DNI, NC) → las claves primarias que reflejan las dos entidades

## 1.5 Lenguajes de Bases de Datos.

Se utilizan para comunicarse los usuarios con la base de datos.

Se clasifican atendiendo a la funcionalidad de la base de datos.

Hay tres tipos, según la sentencia.

### 1. Lenguajes de definición de datos (DDL). Data Definition Language.

Se utilizan para especificar el sistema conceptual de la base de datos. Permiten realizar tareas relativas a la estructura lógica de la base de datos (borrar, crear, modificar, ...).

Permite crear índices, campos y condiciones de integridad.

Crea un objeto vacío en el diccionario o directorio de la base de datos, que contiene metadatos (datos de los datos que almacena).

El sistema gestor de la base de datos es el encargado de actualizar el diccionario de la base de datos. El propio diccionario es una base de datos formada por sus respectivas tablas.

El esquema físico se crea de forma automática a partir del esquema conceptual.

### 2. Lenguajes de manipulación de datos (DML).

También son conocidos como Lenguajes de Gestión o de acceso.

Permiten tener acceso a los datos y hacer operaciones de: inserción, modificación, borrado, consulta.

La operación más importante es la de consultar.

Las bases de datos pueden ser estáticas o volátiles, dependiendo de qué operaciones sean más utilizadas.

Si predominan las operaciones de actualización, se dice que la base de datos es volátil (variante a lo largo del tiempo).

Si es predominante las consultas la base de datos es estática.

También se les conoce como lenguajes de consulta.

Se pueden clasificar atendiendo a diversos criterios:

- Declarativo: Indica qué datos quiere obtener, pero no la forma.  
*(Estructura relacional)*  
Analogía: Cambio automático de un coche.
- Procedimental: qué datos quiere y cómo los quiere obtener,  
*(Algoritmo relacional)*  
secuencia de operaciones.  
Analogía: cambio manual de un coche.

El más eficiente es el procedimental, que es más algebraico.

- Huesped:
    - Extender sintaxis del DML.
    - API
      - ODBC (Microsoft).
      - JDBC (Java).
      - OLEDB
- \* precompilador:  
para poder ejecutar  
en lenguaje.
- Interfaz de programas de aplicaciones.
- Independiente: Para generar programas no necesita lenguajes.  
Permite crear con sentencias DML: (PLSQL).

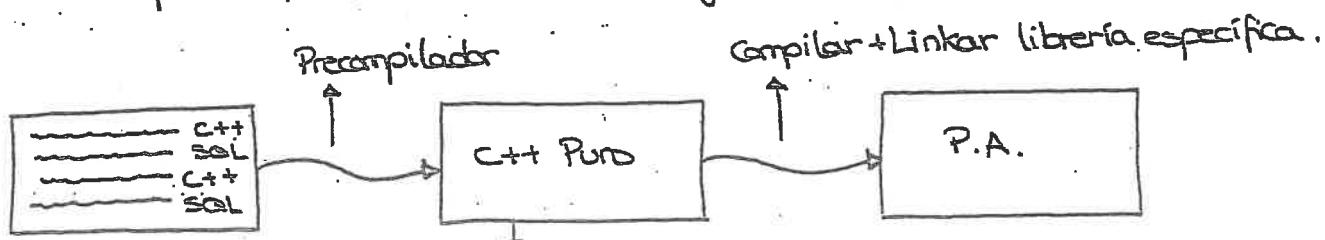
\* API: Conjunto estándar que te permite conectar con el sistema gestor de la base de datos.

Si las sentencias tienen que ser encontradas dentro de un lenguaje general, anfitrión (C++, Pascal), para generar programas de aplicaciones.

Hay dos formas de encontrar el DML, la primera definiendo la sintaxis del DML, habría que añadir delimitadores, nuevas sentencias.

Un API funcionaría con código C++ puro, controlado por el usuario, conociendo el api.

Un DML puede funcionar con los dos y se considera un DML dual.



Las sentencias SQL han sido sustituidas por llamadas al sistema gestor.

Según cómo sea la interacción del sistema:

- { - Interactivo / Conversacional: Se manda una sentencia y el sistema la ejecuta.
- Diferido: Se crea un programa y se ejecutan las sentencias.

### 3 - Lenguajes de control de datos:

Los lenguajes de control de datos permiten un control de acceso, control de transacciones, asignación de privilegios.

#### 1.6 - Usuarios de un sistema de una base de datos.

Se pueden clasificar atendiendo a la forma en que interactúan con el sistema.

- Los operadores de los programas de aplicaciones; sólo conocen cómo funcionan los programas.
- Administrador de la base de datos; persona con control centralizado. Se denomina DBA (Data Base Administrator). Se encarga de crear el esquema de la base de datos, lo diseñan e implementan, mantenimiento y modificaciones. Es la persona que da de alta en el sistema (control de acceso), concede o niega privilegios. Genera estadísticas sobre el uso de la base de datos. Controla el espacio en disco. Mantenimientos rutinarios, si el sistema falla, lo restaura. Formato conversacional.

- Programadores de Aplicaciones. Son los que desarrollan los programas: Usuarios especializados. Utilizan un DML en formato diferido. (Jugadores de Fútbol).

Sofisticados. Usan datos no habituales. Audio, video, información meteorológica, etc.

- Analistas de datos. Responden a peticiones no previstas en tiempo real. Crean informes de consultas no previstas. Usan herramientas más sofisticadas para hacer estudios de los datos. No se pierden equivocar. DML en modo conversacional.

### 1.7. Estructura de un sistema de bases de datos.

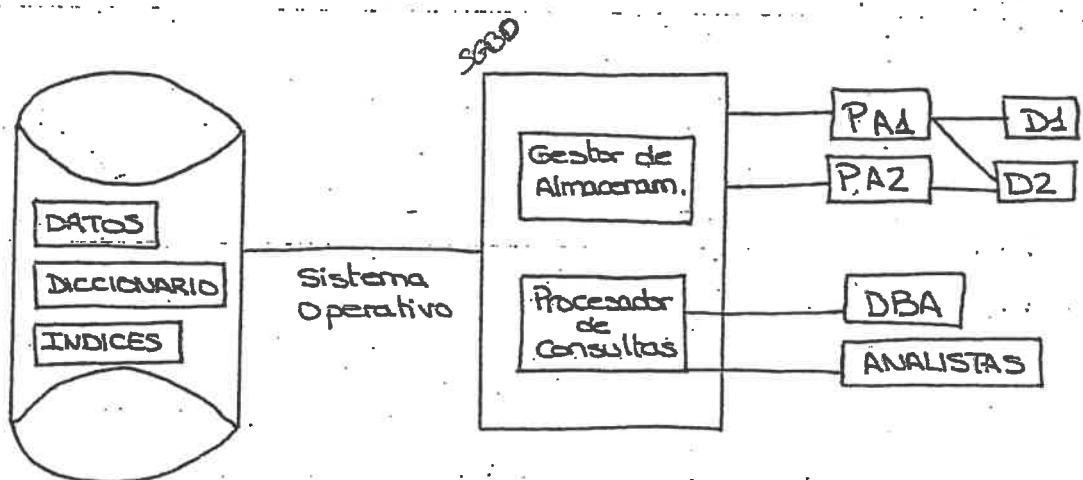
El sistema gestor se descompone en módulos, y éstos a su vez en submódulos. Se encargan de las distintas funcionalidades del sistema. Tiene dos módulos fundamentales:

- Gestor de Almacenamiento: Proporciona la interfaz entre los datos de bajo nivel y las peticiones de acceso. Para acceder a los datos tiene que comunicarse con el sistema operativo. Se encarga del almacenamiento y la modificación de los datos. Se descompone en submódulos:

- Gestor de Archivos: Todo lo que tiene que ver con los archivos.
- Gestor de Transacción: Control de transacciones.
- Gestor de Autorización: Comprueba los privilegios.
- Gestor de Memoria Intermedia (caché): Mejora del acceso.

También el Gestor de Almacenamiento se le considera que tiene datos: archivo de datos (contiene la base de datos); índices (mejoran el acceso, ocupan espacio y hay que mantenerlos); diccionario de la base de datos (contiene metadatos).

- Procesador de consultas: Tiene un intérprete, que es un programa de aplicación, el usuario escribe una consulta y el sistema responde. Compilador DML. Optimizador de consultas, que genera el mejor plan de acceso. Motor de evaluación de consultas, hace un análisis y ejecuta el mejor plan generado por el optimizador de consulta.



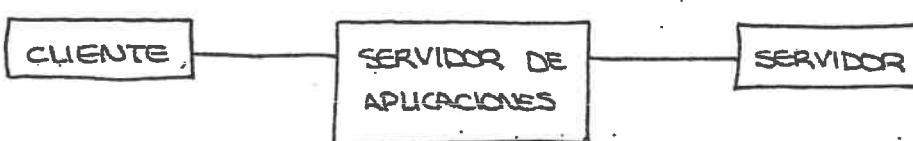
### 1.8 - Arquitectura de Aplicaciones.

Lo normal no es que el usuario interactúe con el sistema, interactúa a través de la red. La arquitectura puede venir servida por dos o por tres capas.

- Dos Capas. Servidor (Sistema Gestor) y cliente.



- Tres Capas. Se añade una capa adicional entre el servidor y el cliente. Los usuarios pasan los datos al servidor de aplicaciones. El cliente nunca instala nada.



Programas de aplicaciones (sólo se instalan una vez).

## TEMA 2: MODELO RELACIONAL DE DATOS

### 2.1 - Introducción

Sistema propuesto por Edgar Frank Codd (1923/2003).

Trabajó para IBM y obtuvo el premio Turing.

### 2.2 - Estructura de las bases de datos relacionales

Una base de datos relacional es un conjunto de tablas o relaciones.

Las tablas tienen un esquema y un cuerpo. Cada tabla tiene un nombre único. Una tabla tiene filas y columnas; en las columnas están los atributos, y en las filas hay valores asociados.

A las filas también se les llama t-uplas y representan una asociación de valores. Su significado depende del significado de la tabla.

Cada atributo tiene asociado un dominio. El dominio es el conjunto de valores permitido para ese atributo. Hay atributos discretos (sexo, edad, ...). Hay atributos continuos (saldo, ...). Cadenas de caracteres (nombres, apellidos, ...).

Los atributos son monolíticos, atómicos.

Una tabla se define como un subconjunto finito del producto cartesiano de los dominios de sus atributos...

$$R \leq \Delta_1 \times \Delta_2 \times \dots \times \Delta_n \quad (\text{No hay t-uplas repetidas}). \quad (\text{filas})$$

Dentro de una tabla no importan el orden de las filas. Aunque si importan el orden de los atributos.

④ Variable t en el atributo A  $\rightarrow t[A]$

R    A    B    C

a<sub>1</sub>   b<sub>1</sub>   c<sub>1</sub>

a<sub>2</sub>   b<sub>2</sub>   c<sub>1</sub>

a<sub>1</sub>   b<sub>2</sub>   c<sub>2</sub>

$$t = (t[A], t[B], t[C])$$

$$t = (t_1, t_2, t_3)$$

$$t = \langle t_1, t_2, t_3 \rangle$$

$$t = (\underbrace{t_1, t_2, \dots, t_n})$$

↓                                  ↓  
Variable de tipo tupla.      Variables de tipo dominio.

④ Las variables tienen asociado un dominio.

$$\text{dom}(t) = R \quad \text{dom}(t_i) = A \quad \text{dom}(t_n) = T_n$$

( $\exists t$ ) ( $t[A] > 5$ )  $\rightarrow$  Existe en t un valor en la columna A mayor que 5.

( $\forall t$ ) ( $t[A] > 5$ )  $\rightarrow$  Quantificativos.

( $\forall t$ ) ( $t[s \in D] \geq 100$ )

④ Se llama Grado de una T-upla al número de componentes que tiene la t-upla.

④ Se llama Grado de una Relación o Grado de una Tabla al número de atributos que tiene la tabla.

tablas  $\left\{ \begin{array}{l} \text{esquema} \\ \text{cuerpo} \end{array} \right. \leftarrow \text{instancia, contenido.}$

④ En el esquema de la tabla ponemos el nombre del atributo, un espacio en blanco y el dominio.

R (A<sub>1</sub> D<sub>1</sub>, A<sub>2</sub> D<sub>2</sub>, ..., A<sub>n</sub> D<sub>n</sub>)

No ponemos el dominio  
para no complicar la sintaxis.

R (A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>n</sub>)  $\rightarrow$  No ponemos el dominio.

R(r)  $\rightarrow$  r representa una lista.

R  $\rightarrow$  Hacemos referencia a la misma tabla, a todo el esquema de r.

R(r)   S(r)  $\rightarrow$  Tienen el mismo esquema pero son dos tablas distintas.

## \* BASE DE DATOS: EJERCICIOS DE CLASE:

CUENTE (DNI, NBC, CDC)	{ NBC → Nombre del cliente. CDC → Ciudad del cliente.
DEPOSITO (CS, NC, DNI, SLD)	{ CS → Código Sucursal. NC → Número de Cuenta. SLD → Saldo.
PRESTAMO (CS, NP, DNI, I)	{ CS → Código Sucursal. NP → Número de Préstamo. I → Importe.
SUCURSAL (CS, NBS, CDS)	{ CS → Código Sucursal. NBS → Nombre de la Sucursal. CDS → Ciudad de Sucursal.

### - CLAVE, SUPERCLAVE -

- Una superclave es un conjunto de atributos que nos permite identificar a las filas de una tabla.
- Todas las tablas tienen al menos una superclave.
- Necesitamos que la superclave sea minimal. Una clave es una superclave minimal.

CLIENTE (DNI, NBC, CDC) → Este conjunto tiene tres superclaves.  
clave.

Una tabla puede tener varias claves, no todas con el mismo cardinal. El administrador escoge una tabla primaria, el resto se llaman claves alternativas. (secundarias)

Hay cuatro métodos de selección de claves:

- 1 - Facilidad de uso: ( $\uparrow$  numérica,  $>$  alfanumérica).
- 2 - Estabilidad de la clave: El sistema crea un índice y habría que cambiarlo con una clave volátil, cambiante.
- 3 - Universalidad. Algo que sea clave y que sea conocido por todos.
- 4 - Fiabilidad. Comprobar si su valor es correcto o no. (NIF  $\rightarrow$  letra).

SUCURSAL (CS, NBS, COS)

Clave Primaria.

PRESTAMO (CS, NP, DNI, I)

Claves Primarias.

DEPOSITO (CS, NC, DNI, SLD)

Claves Primarias.

→ si no cambiaría el significado de la tabla.  
Una persona puede tener varias cuentas  
y en distintas sucursales.

Definición Superclave:

$\text{dom}(t^1) \cap \text{dom}(t^2)$

Sea  $R(r)$ .  $K \subseteq r$  es superclave de  $R \Leftrightarrow (\forall t^1, t^2)(t^1[H] = t^2[H]) \wedge (t^1[r] = t^2[r])$

$\Leftrightarrow (\forall t^1, t^2)(t^1 \neq t^2) \Rightarrow (t^1[H] \neq t^2[H])$

Para cualquier t-upla, si los valores son distintos, las t-uplas son distintas.

Definición Clave:

Sea  $R(r)$ .  $K \subseteq r$  es clave de  $R \Leftrightarrow$

- i)  $K$  es superclave.
- ii)  $K$  es  $\exists K' \subset K$ ,  $K'$  es superclave de  $R$ .  
No hay un subconjunto de  $K$ .

Definición Clave Ajena:

DNI es una clave ajena de la tabla DEPOSITO y hace referencia a la clave primaria en la tabla CUENTE.

1º Nivel →

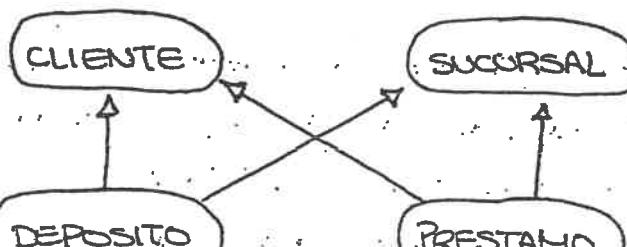
CLIENTE

SUCURSAL

2º Nivel →

DEPOSITO

PRESTAMO



- La clase ajena define a una tabla padre y a una tabla hijo.
- Tabla referenciada y Tabla referenciante.
- La clase ajena se define en la tabla hijo.

$R(r) \quad S(r) \quad K$  clase primaria de  $R \wedge A \in S$ .

A es clase ajena en S, referencia a la clase primaria K(r).

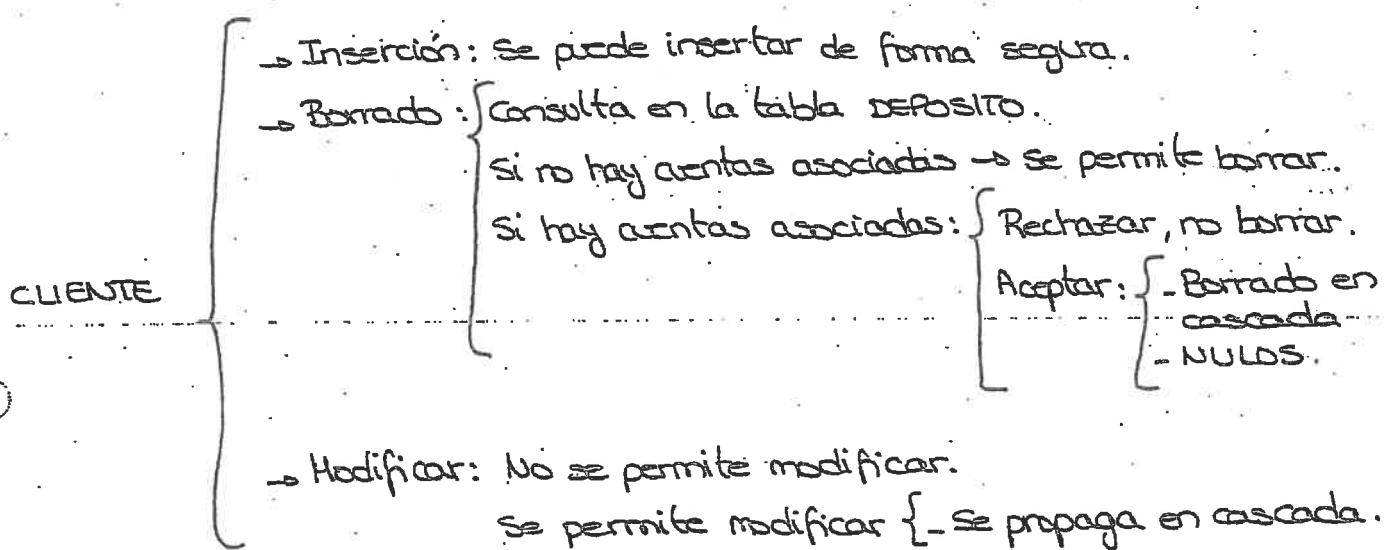
$$\Leftrightarrow (\forall t)(\exists t)(t[A] = t[K])$$

### Acciones compensatorias

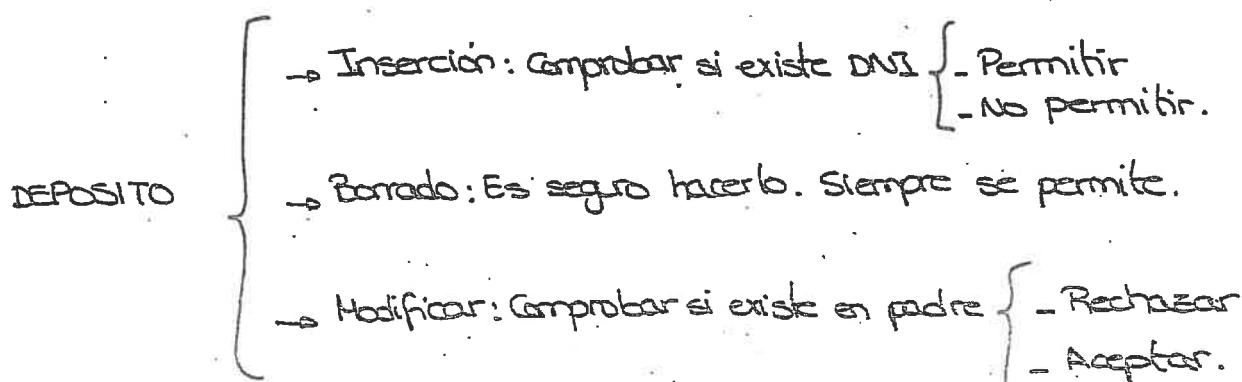
También denominadas acciones de restauración de la consistencia.

Se consideran dos casos: tabla padre y tabla hijo.

Primero con la tabla padre, (CLIENTE):



Ahora con la tabla hija:



## 2.3 - Valores Nulos.

Nulos o desconocidos. Sirven para representar información no conocida en la base de datos. Se representan con: ?.

La información desconocida puede ser por varios motivos: que no existe el dato, que existe pero no se sepa, que existe pero no se quiere dar el dato.

Sirven para darle flexibilidad a la base de datos.

El valor nulo es un elemento más del dominio del atributo.

Para que dos filas sean iguales, todos sus atributos tienen que ser iguales y sus nulos tienen que ser los mismos.

Si a un valor null le sumas algo, el valor sería null.

$$? + 3 = ?$$

En comparaciones sería una lógica ternaria:

$$x > 2 \rightarrow ? > 2 \quad \left\{ \begin{array}{l} V \\ F \\ d? \end{array} \right.$$

Tabla de verdad asociada a la conjunción lógica:

$\wedge$	V	F	?	$\vee$	V	F	?	$>$	V	F	(Negación).
V	V	F	?	V	V	V	V	V	F		
F	F	F	F	F	V	F	?	F	V		
?	?	F	?	?	V	?	?	?	?		

Comprobar que un atributo toma el valor NULL.

IS NULL (A)  $\rightarrow$  función lógica que devuelva V o F.

## 2.4 - Integridad de claves primarias.

Regla de integridad. Los atributos de una clave primaria no pueden contener valores nulos.

## 2.5 - Integridad referencial.

Regla de integridad. Los valores no nulos de la clave ajena tienen que existir en la clave primaria referencial.

### EMPLEADO (DNI\_E, NBE, DNI\_J)

Actúa como padre y como hijo.

Para dar de alta a un empleado tiene que estar dado de alta su jefe. Salvo el más alto nivel que no tiene jefe.

## 2.6 - Componentes del modelo relacional de datos.

Todos los modelos de datos tienen tres componentes:

- Estructuras de Datos: Tablas, índices, vistas, t-uplas, ...

- Operadores: Operadores de álgebra relacional { - Unarios  
- Binarios

Producen otra tabla, un sistema cerrado. Al aplicar un operando en una tabla, obtengo otra tabla. Permite anidar unos dentro de otros.

{ - R.I. Claves Primarias.

- Condiciones de Integridad: { - R.I. Referencial.

Se definen de manera declarativa, sin programarlas.

## 2.7 - Álgebra Relacional

### 2.7.1 - Introducción:

Lenguaje de consultas procedimentales. Indicar qué se quiere y cómo se quiere obtener.

Operadores

-	Esenciales. No se puede suprimir sin perder potencia expresiva del lenguaje.
-	Derivadas. Se pueden suprimir, pero son más eficientes.

### 2.7.2 - Operadores esenciales:

No se pueden eliminar sin quitar potencia expresiva al lenguaje.

#### \* SELECCIÓN:

La selección de t-uplas de una relación R, es otra relación con la misma cabecera que la de R y cuyo cuerpo está formado por las t-uplas de R que verifican una condición impuesta a los atributos. En la "condición" pueden aparecer operadores de comparación ( $=, <, >$ ) y booleanos (AND, OR, NOT). La selección extrae las t-uplas especificadas de una relación dada, es decir, restringe la relación sólo a las t-uplas que satisfacen la condición. Los atributos que aparecen en la "condición" deben estar definidos sobre el mismo dominio. Es un operador de filtrado.

$S(F)(R) \rightarrow$  Selección de la tabla R, las filas F.

$S(F)(R)(r) \rightarrow$  Detallamos filas.

- Listar los clientes del banco que viven en S/C.

Consulta de lenguaje natural  
S (CDC = "Santa Cruz") (CUENTE). Utilizamos esta tabla  
 $|S(F)(R)(r)| \leq |IR(r)| \rightarrow$  Menor o igual que el origen.  
 $S((SLD > 10000) \wedge (S=1)) (DEPOSITO)$ ; Es importante el operador  $\wedge$  y !

- Listar los clientes que viven en S/C y se llaman Juan.

$S(CDC = 'Santa Cruz') \wedge (NOMBRE = 'Juan') (CUENTE)$ .

Notación de otros autores:  $O_F(R)$

Hago una consulta en DEPOSITO, de quién tiene un saldo superior a 10000

### Definición: Proyección:

Es una relación que tiene como cabecera la formada por los atributos de  $R$  y en cuyo cuerpo aparecen todas las tuplas de  $R$  restringidas a dichos atributos, eliminando las tuplas repetidas. Concentra la información, reduce la tabla en anchura y posiblemente también en profundidad.

$R$	A	B	C	$P(A, B)(R)$	A	B
	$a_1$	$b_1$	$c_1$		$a_1$	$b_1$
	$a_1$	$b_1$	$c_2$		$a_2$	$b_1$
	$a_2$	$b_1$	$c_3$			

$P(s)(R)$

proyección de la tabla  $R$  sobre los atributos  $s$

Otra anotación sería:  $\Pi_A(R)$ . (otros autores)

Proyección de clientes que viven en La Laguna.

$P(DNI) S (CDC = 'La Laguna')(CLIENTE)$

### \*PRODUCTO CARTESIANO:

Relación cuya cabecera es la combinación de las cabeceras de  $R_1$  y  $R_2$ , y cuyo cuerpo está formado por el conjunto de todas las tuplas tales que  $t$  es la combinación de una tupla  $t_1$  perteneciente a  $R_1$  y una tupla  $t_2$  perteneciente a  $R_2$ . El producto cartesiano construye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones.

Resultado de una nueva tabla que concatena las dos anteriores.

Per cada fila de la tabla  $R$ , le asignamos una fila de la tabla  $S$ . Tiene el inconveniente de que el sistema se wedge, ya que ocuparía una cantidad de espacio muy grande.

$$|R \times S| = |R| * |S|$$

$$(R \times S) \quad (r+s)$$

m+n filas.

R	A B C	S	B D	R x S	A R.B C S.B D
	a <sub>1</sub> b <sub>1</sub> c <sub>1</sub>		c <sub>1</sub> d <sub>1</sub>		a <sub>1</sub> b <sub>1</sub> c <sub>1</sub> d <sub>1</sub>
	a <sub>2</sub> b <sub>2</sub> c <sub>2</sub>		c <sub>2</sub> d <sub>2</sub>		a <sub>2</sub> b <sub>2</sub> c <sub>2</sub> d <sub>2</sub>
			c <sub>3</sub> d <sub>3</sub>		a <sub>1</sub> b <sub>1</sub> c <sub>2</sub> d <sub>3</sub>

- Nombres de los clientes que tienen al menos una cuenta en la sucursal con código 1.

$$P(NBC) \quad S(CS=1) \quad (\text{CLIENTE} \times \text{DEPOSITO}) = P(NBC) \quad (\text{CLIENTE}).$$

Esta consulta sólo nos muestra todos los clientes, pero no nos especifica al cliente con su respectiva cuenta.

$$P(NBC) \quad S((CS=1)^*(\text{CLIENTE.DNI} = \text{DEPOSITO.DNI})) \quad (\text{CLIENTE} \times \text{DEPOSITO})$$

De forma más eficiente:

$$P(NBC) \quad S(\text{CLIENTE.DNI} = \text{DEPOSITO.DNI}) \quad (S(CS=1)(\text{DEPOSITO}) \times \text{CLIENTE}).$$

Reducimos el tamaño de la tabla resultante:

$$P(DNI) \quad (S(CS=1)(\text{DEPOSITO})) \times \text{CLIENTE}.$$

• No se pueden multiplicar las tablas por sí mismas. Para hacerlo se asignan trucos como por ejemplo las "VISTAS" (dar un alias).

- Sucursales ubicadas en La Laguna.

Esquema CS.

$$A = P(CS) \quad \underbrace{S(CDS='La Laguna') \quad (\text{SUCURSAL})}_{\text{Esquema = atributos de sucursal.}}$$

$$\left. \begin{array}{l} A = R \\ B = R \end{array} \right\} A \times B.$$

\*UNION: "RUS"  $\leq |R| + |S|$  (equivale a una suma).

Es una relación cuya cabecera es idéntica a  $R_1$  o  $R_2$  (uniendo dos relaciones  $R_1$  y  $R_2$ ), y cuyo cuerpo está formado por todas las tuplas pertenecientes a  $R_1$ , o a  $R_2$ , o a las dos. La unión construye una relación formada por todas las tuplas que aparecen en algunas de las dos relaciones especificadas. Omite aquellos atributos que estén repetidos. Tienen que tener ambas relaciones el mismo número de columnas. Los dominios de los atributos i-ésimos coinciden, de cada columna (primera con primera, ...).

• mismo grado

R	A	B	C
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>2</sub>	c <sub>2</sub>	

S	A	B	C
c <sub>1</sub>	d <sub>1</sub>	e <sub>1</sub>	
c <sub>2</sub>	d <sub>2</sub>	e <sub>2</sub>	

RUS	A	B	C
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>2</sub>	c <sub>2</sub>	
c <sub>1</sub>	d <sub>1</sub>	e <sub>1</sub>	

Ejemplo:

- DNI de los clientes que tienen al menos una cuenta en la sucursal con código 1 y al menos un préstamo en la sucursal con código 1.

$A \Rightarrow P(\text{DNI}) \quad S(\text{CS}=1) \quad (\text{DEPÓSITO})$

$B \Rightarrow P(\text{DNI}) \quad S(\text{CS}=1) \quad (\text{PRESTAMO})$

} AUB.

RUS (rus)

\*DIFERENCIA:  $|R-S| \leq |R|$

Es una relación cuya cabecera es idéntica a la de  $R_1$  (o  $R_2$ ) y cuyo cuerpo está formado por todos las tuplas pertenecientes a  $R_1$  pero no a  $R_2$ . La diferencia construye una relación formada por todas las tuplas de la primera relación que no aparecen en la segunda. Las tablas tienen que tener el mismo grado y que los dominios de los atributos i-ésimos coincidan.

$$0 \leq |R-S| \leq |R|$$

cuando R y S  
son iguales

cuando R y S son distintas

### Ejemplo:

- Clientes que tienen al menos una cuenta en la sucursal con código 1, pero no tienen un préstamo en dicha sucursal.

$A \Rightarrow P(\text{DNI}) \ S (\text{CS} = 1) \ (\text{DEPOSITO})$

$B \Rightarrow P(\text{DNI}) \ S (\text{CS} = 1) \ (\text{PRESTAMO})$

$$\left. \begin{array}{l} \\ \end{array} \right\} A - B$$

### 2.7.3. Operadores derivados:

#### \* INTERSECCIÓN: RNS

Es una relación cuya cabecera es idéntica a  $R_1$  (o a  $R_2$ ) y cuyo cuerpo está formado por todas las tuplas pertenecientes tanto a  $R_1$  como a  $R_2$ . La intersección construye una relación formada por todas las tuplas que aparecen en las dos relaciones especificadas. Sólo representa dominios repetidos.

$$|RNS| \leq \min \{|R_1|, |S_1|\}$$

Ambas tablas tienen que tener el mismo grado. Los dominios tienen que coincidir.

R   A B		S   C D		RNS
$a_1$	$b_1$	$a_1$	$b_1$	$a_1$
$a_2$	$b_2$	$a_3$	$b_3$	

- DNI de los clientes con cuenta en la sucursal 1 con préstamo en la sucursal con código 1.

$A \Rightarrow P(\text{DNI}) \ S (\text{CS} = 1) \ (\text{DEPOSITO})$

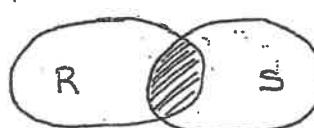
$B \Rightarrow P(\text{DNI}) \ S (\text{CS} = 1) \ (\text{PRESTAMO})$

$$\left. \begin{array}{l} \\ \end{array} \right\} \underline{\underline{A \cap B}}$$

- En función de los operadores esenciales  $\rightarrow$  Diferencia.

$$R - (R - S) = S - (S - R)$$

$\hookrightarrow$  Operador conmutativo.



No se usa mucho

\* YUNCIÓN:  $R \text{ Y}_{(F)} S \rightarrow$  Unión entre R y S según el predicado de F.  
 También es conocido como reunión, concatenación, ..., combinación, ...  
 Combina el producto cartesiano y la selección.

$$\underbrace{R \text{ Y}_{(F)} S}_{(r+s)} = S(F) (R \times S)$$

- Nombres de clientes con cuenta en la sucursal con código 1.  
 $P(\text{NBC})(\text{CS} = 1) \wedge (\text{CLIENTE.DNI} = \text{DEPOSITO.DNI})$  (CLIENTE X DEPOSITO)

$P(\text{NBC})(\text{CLIENTE} \vee ((\text{CS} = 1) \wedge (\text{CLIENTE.DNI} = \text{DEPOSITO.DNI}))) \text{DEPOSITO}$

Es más eficiente que de la manera anterior, pero no se reduce la cantidad escrita.

\* YUNCIÓN NATURAL:  $|R * S|$

Es una relación que contiene todas las posibles combinaciones de t-uplas una de cada una de las dos relaciones, tales que las dos t-uplas participantes en la combinación tengan los mismos valores en los atributos comunes. Une atributos y elimina una de cada columna repetida.  
Selecciona igualdad de valores en los atributos homónimos.

$$R * S \equiv P(r \cup s) | S(R.rns = S.rns) (R \times S)$$

R	A	B	C	F	B	D	E	R * F	A	B	C	D	E
1	0	1			2	1	0		1	0	1	3	0
1	0	0			1	1	1		1	0	0	3	0
0	2	3			2	1	3		0	2	3	1	0
1	2	3			0	3	0		0	2	3	1	3
0	2	0			3	0	0		1	2	3	1	0

Hace una unión por el campo común.

Unión y producto cartesiano al mismo tiempo.

$$\underline{R * S} = \underline{R \times S} \quad rns = \emptyset$$

→ Si no hay atributos comunes la función natural equivale al producto cartesiano.

→ Si el esquema de R coincide exactamente con el esquema de S.  
 $r = s$ . Sería la intersección.  $\underline{R * S} = \underline{RNS}$ .

- Nombres de cliente con cuenta en la sucursal 1.

$P(NBC) \ S(CS=1) \ (CLIENTE * DEPOSITO)$ .

\* COCIENTE:

Hay que verificar que el esquema de S tiene que estar contenido en el esquema de R.  $R(r) \ S(s) \rightarrow S \subset r$ .

Se aplica al primer conjunto, elimina todas las filas que no coinciden con la segunda columna.

Da como resultado una tabla que contiene:  $(r - s)$

El contenido se extrae de:  $P(r-s)(R)$

$(R/S) \times S \subseteq R$  (cociente  $\times$  divisor  $\leq$  dividendo).

$(r - s + \emptyset) = r$  el esquema se verifica.

El cociente en función de los operadores esenciales:

$$(R/S) = P(r-s)(R) - P(r-s)((P(r-s)(R) \times S) - R)$$

- DNI de los clientes que tienen cuentas en todas las sucursales de La Laguna.

$$S \Rightarrow P(CS) \ S(CDS = 'La Laguna') \ (\text{SUCURSAL}) \quad \left. \right\} R/S.$$

$$R \Rightarrow P(DNI, CS) \ (\text{DEPOSITO}).$$

S está contenido en R.

→ Parejas de datos, los DNI de los clientes de dichas sucursales.

$$R/S(DNI) \times S$$

### - Actualización -

#### 2.7.4 Modificación de la Base de Datos.

El álgebra relacional permite hacer inserciones, borrado y modificaciones. Permite hacer operaciones de actualización de la base de datos.

	operador esencial	Masivamente:
Inserción	$R \cup \{t^1\}$ (1 fila)	$R \cup S$ (Múltiples filas).
Borrado	$R - \{t^1\}$	$R - S$
Modificación	$(R - \{t^1\}) \cup \{t^2\}$	$(R \cup \{t^2\}) - \{t^1\} \rightarrow$ No se hace. Podemos borrar un atributo que no forma parte de la clave.

#### 2.7.5 Operación de complementos.

El complemento de la tabla es lo que contiene la tabla.

$$R(A_1, A_2, \dots, A_n)$$

$\downarrow \quad \downarrow$

$$\text{dominio} = D_1 \times D_2 \times \dots \times D_n$$

El operador complemento no se puede definir en álgebra relacional.

$$\bar{R} = D_1 \times D_2 \times \dots \times D_n - R \rightarrow \text{Podría ser } \infty$$

Por eso no podemos hallar el complemento con respecto a cualquier cosa.

Lo que falta, que no está en  $R$ .  
Pueden haber dominios infinitos.

#### 2.7.6 Potencia expresiva del álgebra relacional.

Conjunto de frases gramaticalmente correctas (idioma). En el álgebra relacional son los (operadores esenciales) + operadores + potencia expresiva.

El cálculo relacional de t-uplas es relacionalmente completo con respecto al álgebra relacional, misma potencia expresiva.

SOL es relationalmente completo, pero en álgebra no se puede hacer toda las consultas de SOL. Algebra relacional es un lenguaje pobre.

#### 2.7.7 Propiedades de los operadores.

Las consultas son compatibles y equivalentes cuando dan los mismos resultados ambas consultas. Si son equivalentes dan lo mismo. Lo normal es que una consulta tenga diversas formas de hacerla. Las propiedades de los operadores se utilizan para optimizar las consultas.

- (i) Calcular el conjunto de consultas equivalentes.
- (ii) Seleccionar la "mejor". (optimizar)

## Propiedades:

- Dos consultas son equivalentes cuando devuelven el mismo resultado cualesquiera sea sus valores de operando.

1. La unión y la intersección son comutativos y asociativos.

$$R \cup S = S \cup R$$

Comutativos

$$(R \cup S) \cup T = R \cup (S \cup T)$$

Asociativos

2. Si sólo importa la información de las tablas, se puede decir que el producto cartesiano ( $\times$ ) y la unión natural ( $*$ ) son cuantitativos y asociativos.

$$R * (S * T) = (R * S) * T \rightarrow \text{Si no importa el orden de los atributos:}$$

3. La unión y la intersección verifican la distributiva de la una en cuanto a la otra.

$$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$$

$$R \cap (S \cup T) = (R \cap S) \cup (R \cap T) \rightarrow \text{(más eficiente!).}$$

4. Commutativa entre la selección y el producto cartesiano ( $\times$ ).

Commutativa entre la selección y la unión natural ( $*$ ).

$$S(F_1 \wedge F_2)(R * S) = (S(F_1)(R)) * (S(F_2)(S))$$

Sólo se cumplirá si:  $\left\{ \begin{array}{l} \text{solo atributos de } R \\ \text{solo atributos de } S \end{array} \right.$  más eficiente, reduce antes de multiplicar.

$$S(F_1)(R * S) = S(F_1)(R) * S \rightarrow \text{Generalizada, selecciona toda la tabla. } (F_1 \wedge \text{TRUE})$$

5. Cascada de selecciones.

$$S(F_1)(S(F_2)(S(F_3)(\dots(S(F_n)))))) = S(F_1 \wedge F_2 \wedge \dots \wedge F_n)(R)$$

$$S(F_2)(R \times S) = (R) \times S(F_2)(S)$$

→ Intersección - OR

Aplica la selección una sola vez. Eficiente

6. Cascada de proyecciones.

$$P(A_1)(P(A_2)(P(A_3)(\dots(P(A_n)))))) = P(A_1)(R)$$

Operaciones superfluas.

(R)

Tiene que darse que estén contenidos...  
A<sub>2</sub> está contenido en A<sub>1</sub>!

7. Commutación de la proyección y la selección.

$$P(A)(S(F)(R)) = S(F)(P(A)(R)) \rightarrow \text{Depende de la consulta será una u otra.}$$

Cuando en el predicado F sólo aparecen atributos que están en A.

8. Commutativa de la proyección y el producto cartesiano. / Unión Natural  
 $P(A)(R \times S) = (P(A \cap r)(R)) \times (P(A \cap s)(S))$

9. Commutativa entre la selección y cualquier operador del conjunto. {U, ∩, -  
 $S(F)(R \cup S) = (S(F)(R)) \cup (S(F)(S)) \rightarrow$  Eficiencia depende del filtro

10. Commutación entre la proyección y la unión.

$$P(A)(R \cup S) = (P(A)(R)) \cup (P(A)(S))$$

2.7.3 Vistas: También se les llama relaciones virtuales

- Es cuando asignamos un nombre a una consulta. El sistema guarda la definición y no se calcula. La vista se puede usar, "llamarla". Se calculan cuando las usamos. El contenido no se almacena.

- Al hacer una actualización, la vista se actualiza permanentemente.

- Por otro lado, las vistas son lentas, hay que recalcularlas.

- Las vistas materializadas son vistas para las que se almacena el contenido. ocupan memoria y cada actualización hay que actualizar la vista en que se basa en esa tabla.

• Deben ser actualizadas siempre que no incumplan ninguna regla de integridad y que las operaciones sean suficientemente sencillas.

• Usos de las vistas:

1- Simplificación de las consultas, reutilización del código, aumento de legibilidad.  $A \cup B$ ;  $A \cap B$ ;  $A - B$   
y la productividad.

2- Proteger la confidencialidad de los datos. Seguridad. Da privilegio por los atributos no sensibles, los sensibles los ocultan.

$A = P(CS, NC, DNI) \cup (DEPOSITO)$   $\rightarrow$  se dan permisos a la vista  
( $s(CS=1)$ ) no a la tabla..

3- Protección frente a cambios en el diseño. Hace que la base de datos sea robusta. Por ejemplo: almacenamos información en una única tabla y con el tiempo el administrador cambia las tablas.

\* Noticia: las vistas son sencillables

• Vista materializada = vista con contenido

### 2.7.9. Extensión de operadores.

Operadores binarios y se usan con NULL.

#### \* UNION EXTERNA. (RUE S) (rus)

R y S no tienen porque coincidir en grado. Los dominios de los atributos homónimos tienen que coincidir.

El esquema de la tabla resultado es la unión de ambos esquemas.

Se completan las tablas adecuadamente con valores NULL.

Los valores repetidos se eliminan.

Permite mezclar fuentes heterogéneas en un único repositorio.

R	A B C	S	B D	(SUE R)	A B C D
	a <sub>1</sub> b <sub>1</sub> c <sub>1</sub>		b <sub>1</sub> d <sub>1</sub>		a <sub>1</sub> b <sub>1</sub> c <sub>1</sub> ?
	a <sub>2</sub> b <sub>2</sub> c <sub>2</sub>		b <sub>2</sub> d <sub>2</sub>		a <sub>2</sub> b <sub>2</sub> c <sub>2</sub> ?
	a <sub>3</sub> b <sub>3</sub> c <sub>3</sub>				a <sub>3</sub> b <sub>3</sub> c <sub>3</sub> ?

#### \* UNION NATURAL EXTERNA. (R YE S)

Generaliza a la unión natural. El esquema de la tabla resultado es la unión de los dos esquemas (rus).

Unión ← Producto cartesiano. Selecciona filas. Añade valor que no existe en otra tabla. Proyecta. O un valor nulo. Las nuevas t-uplas se completan con nulo.

R	A B	S	A C	R YE S	A B C
	a 1		a 6		a 1 6
	b 2		e 7		b 2 ?
	? 3		? 8		? 3 ?
	c 4		c 9		c 4 9
	d 5		f A		d 5 ?

Natural outer join

full → completa  
 left → + los gr nul emp. ig.  
 right.

e	?	7
?	?	8
?	?	A

↳ Unión natural (normal)

-- tachados → right

### \* Unión Posible: (R Y P. S) (r+s)

Da todas las posibilidades. El esquema es la concatenación de ambos esquemas, no la unión.

La tabla de resultados, (r+s) es el producto cartesiano de r y s, seleccionar filas que tienen igualdad de valores en los atributos homónimos considerando dos valores nulos, iguales entre sí.

R.	A	B	S	B	C	R Y P S	A	R.B	S.B	C
a	1		1	A		a	1	1	A	
b	2		?	B		b	2	?	B	
c	?		2	?		c	2	2	?	
?	3		4	C		c	?	1	A	
						c	?	2	B	
						c	?	2	?	
						?	3	?	B	

Tiene todas las opciones posibles. Los atributos homónimos no son iguales por eso no los elimina.

### 2.8 Cálculo Relacional.

Cálculo Relacional  $\left\{ \begin{array}{l} \text{T-uples: } \{t_n\} / p(t) \\ \text{Dominios: } \{x_1, x_2, \dots, x_n\} / P(x_1, x_2, \dots, x_n) \end{array} \right\}$

Una proposición lógica, tipo declarativa. Verdadero o falso, pero no ambas cosas.

El Cálculo Relacional es el área de la lógica que estudia las proposiciones.

La lógica de predicados es más específica, enuncia la condición y el objeto en el que se aplica, el quién. Qué se afirma y sobre quién se afirma.

## 2.8.1. Nociones básicas sobre lógicas de predicado.

OPERADORES lógicos:

- |                                     |                                      |
|-------------------------------------|--------------------------------------|
| 1). $\neg$ Negación.                | 4). $\rightarrow$ Implicación.       |
| 2). $\wedge$ (y lógico) Conjunción. | 5). $\leftrightarrow$ Bicondicional. |
| 3). $\vee$ (o lógico) Disyunción.   | 6). or exclusivo                     |
- $\left. \begin{array}{l} \\ \\ \end{array} \right\}$  No se van a usar.

$$P \rightarrow q \Rightarrow p \text{ entonces } q.$$

Antecedente  $\rightarrow$  Conclusion  
consecuente.

→ Ejemplo: Si llueve, entonces las calles se mojan.

- Recíproco de la implicación:  $q \rightarrow p \equiv \neg p \rightarrow \neg q$
- Inversa de la implicación:  $\neg p \rightarrow \neg q$ . Contrario.

Dos lógicas son equivalentes cuando tienen la misma tabla de verdad.

$$P \rightarrow q \equiv \neg q \rightarrow \neg p \text{ (equivalentes). } (p \vee (q \wedge r)) \rightarrow s.$$

Tautología: Algo que siempre es verdad. (llueve o no llueve).

$$P \vee \neg P$$

Contradicción: Decir una cosa y mantener también la postura contraria.

$$P \wedge \neg P$$

Contingencia: Devolver una cosa u otra. Verdadero o falso.

P y q son equivalentes, si el bicondicional es una tautología.

$$P \equiv q \Leftrightarrow P \leftrightarrow q \text{ tautología} \quad P \leftrightarrow q \left\{ \begin{array}{l} P \rightarrow q \\ q \rightarrow P \end{array} \right.$$

### \* PROPIEDADES O LEYES.

- Identidad:  $P \wedge V \equiv P$

$$P \vee F \equiv P$$

- Dominación:  $P \wedge F \equiv F$ .

$$P \vee V \equiv V$$

- Idempotentes:  $p \vee p \equiv p$       - Doble negación:  $\neg(\neg p) \equiv p$   
 $p \wedge p \equiv p$

- Comutativas:  $p \wedge q \equiv q \wedge p$       - Asociativas:  $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$   
 $p \vee q \equiv q \vee p$        $(p \vee q) \vee r \equiv p \vee (q \vee r)$

- Distributivas:  $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$       - Negación:  $p \wedge \neg p \equiv F$   
 $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$        $p \vee \neg p \equiv V$

- Morgan:  $\neg(p \wedge q) \equiv \neg p \vee \neg q$       - Absorción:  $p \vee (p \wedge q) \equiv p$   
 $\neg(p \vee q) \equiv \neg p \wedge \neg q$        $p \wedge (p \vee q) \equiv p$

- Equivocencia Condicional:  $p \rightarrow q \equiv \neg p \vee q$   
 $p \rightarrow q \equiv \neg q \rightarrow \neg p$   
 $\neg(p \rightarrow q) \equiv p \wedge \neg q$   
 $\neg(p \rightarrow \neg q) \equiv p \wedge q$   
 $p \rightarrow q \wedge p \rightarrow r \equiv p \rightarrow (q \wedge r)$   
 $p \rightarrow r \wedge q \rightarrow r \equiv (p \vee q) \rightarrow r$   
 $p \rightarrow q \vee p \rightarrow r \equiv p \rightarrow (q \vee r)$   
 $p \rightarrow r \wedge q \rightarrow r \equiv (p \wedge q) \rightarrow r$ .

→ Todas estas propiedades se utilizan para simplificar predicados.

$(\exists x)(P_x) \rightarrow$  Existe un  $x$  que se verifica en el resto del dominio.

- Equivocencias. Permiten transformar un predicado lógico en otro.

$$(\exists x)(P_x) \equiv \neg(\forall x)\neg(P_x)$$

$$(\forall x)(P(x) \rightarrow q(x))$$

$$(\forall x)(P_x) \equiv \neg(\exists x)\neg(P_x)$$

$$(\forall x)(\neg P(x) \vee q(x))$$

$$\neg(\exists x)(P_x) \equiv (\forall x)\neg(P_x)$$

$$(\exists x)(\exists y)(\exists z) P(x,y,z) \equiv (\exists x,y,z) P(x,y,z).$$

$$(\forall x)(\forall y)(\forall z) P(x,y,z) \equiv (\forall x,y,z) P(x,y,z).$$

$$(\exists x)(\forall y)(\exists z) P(x,y,z) \not\equiv (\forall y)(\exists x)(\exists z) P(x,y,z) \rightarrow \text{No se pueden intercambiar los quantificadores.}$$

## 2.8.2 - Definición informal de Cálculo Relacional de T-uplas (CRT).

Conjunto de t-uplas  $t$  de grado  $n$  que satisface el predicado lógico  $p(t)$ .

$$\{t_n / p(t)\}$$

$\downarrow n$ : de atributos.

Variable libre Variable ligada. Tiene que estar cuantificada.

Puede estar sin cuantificar.

- DNI de los clientes que tienen al menos una cuenta en el banco.

Álgebra relacional :  $P(DNI) \cap DEPOSITO$

Cálculo : Determinamos el dominio :  $dom(d) = DEPOSITO$ .

$$\{t_n / (\exists d) (t[DNI] = d[CDNI])\}.$$

- DNI de clientes con al menos una cuenta en la sucursal con código 1.

$P(DNI) \cap S(CS=1) \cap DEPOSITO$

$$\{t_n / (\exists d) ((t[DNI] = d[CDNI]) \wedge (d[CS] = 1)) \quad dom(d) = DEPOSITO\}.$$

- Nombres de los clientes que tienen al menos una cuenta en la sucursal con código 1.

$P(NBC) \cap S(CS=1) \cap (CLIENTE * DEPOSITO)$ .

$dom(d) = DEPOSITO \quad dom(c) = CLIENTE$

$$\begin{aligned} & \{t_n / (\exists c) ((t[NBC] = c[NBC]) \wedge \\ & (\exists d) ((d[DNI] = c[CDNI]) \wedge (d[CS] = 1)))\}. \end{aligned}$$

$$\begin{aligned} & \{t_n / (\exists c, d) (((t[NBC] = c[NBC]) \wedge (d[DNI] = c[CDNI])) \wedge \\ & \quad \wedge (d[CS] = 1))\}. \end{aligned}$$

- Listar los t-uplos de la tabla DEPOSITO.

$$\{t_d / t \in \text{DEPOSITO}\} \rightarrow \{t_d / t \in \text{DEPOSITO}\}$$

Más simplificado

- DNI de los clientes que tienen un préstamo o una cuenta en la sucursal con código 1.

$$P(\text{DNI}) \wedge S(\text{CS} = 1) \wedge (\text{DEPOSITO}) = R$$

$$P(\text{DNI}) \wedge S(\text{CS} = 1) \wedge (\text{PRESTAMO}) = S$$

RUS → Une ambas tablas y omite las repeticiones.

$$\text{dom}(d) = \text{DEPOSITO} \quad \text{dom}(p) = \text{PRESTAMO}$$

$$\{t_d / (\exists d)((t[\text{DNI}] = d[\text{DNI}]) \wedge (d[\text{CS}] = 1)) \vee \\ (\exists p)((t[\text{DNI}] = p[\text{DNI}]) \wedge (p[\text{CS}] = 1))\}$$

- DNI de los clientes que tienen un préstamo y una cuenta en la sucursal con código 1.

$$R = P(\text{DNI}) \wedge S(\text{CS} = 1) \wedge (\text{DEPOSITO}) \quad S = P(\text{DNI}) \wedge S(\text{CS} = 1) \wedge (\text{PRESTAMO}) \quad \left. \begin{array}{l} \text{R} \cap \text{S} \\ \text{(La intersección se queda con los atributos comunes en ambas tablas)} \end{array} \right.$$

$$\text{dom}(d) = \text{DEPOSITO} \quad \text{dom}(p) = \text{PRESTAMO}$$

$$\{t_d / (\exists d)((t[\text{DNI}] = d[\text{DNI}]) \wedge (d[\text{CS}] = 1)) \wedge \\ (\exists p)((t[\text{DNI}] = p[\text{DNI}]) \wedge (p[\text{CS}] = 1))\}$$

- DNI de los clientes con cuenta en la sucursal con código 1 pero que no tienen préstamo.

$$P(\text{DNI}) \wedge S(\text{CS} = 1) \wedge (\text{DEPOSITO}) = R \quad P(\text{DNI}) \wedge S(\text{CS} = 1) \wedge (\text{PRESTAMO}) = S \quad \left. \begin{array}{l} R - S \end{array} \right.$$

$$\text{dom}(d) = \text{DEPOSITO} \quad \text{dom}(p) = \text{PRESTAMO}$$

$$\{t_d / (\exists d)((t[\text{DNI}] = d[\text{DNI}]) \wedge (d[\text{CS}] = 1)) \wedge \\ (\exists p)((t[\text{DNI}] = p[\text{DNI}]) \wedge (p[\text{CS}] = 1))\}$$

Quantificador universal  $\forall$  → Relacionado con el cociente en álgebra relacional.

- DNI de los clientes que tienen cuenta en todas las sucursales de 'La Laguna'.

Equivalente: 'Para cualquier sucursal, el cliente tiene cuenta'.

$$R = P(DNI, CS) \text{ (DEPOSITO)}$$

$$S = P(CS) \text{ s } (CDS = 'La Laguna') \text{ (SUCURSAL)}$$

R/S

Elimina todas las filas que no coinciden con la segunda columna.

$$\{ t_{11} / (\forall s) ((s[CCDC] \neq 'La Laguna')) \vee \dots \}$$

$$\{ t_{11} / (\exists d) ((t[DNI] = d[DNI]) \wedge (d[CS] = s[CS])) \}$$

Cociente  $\Leftrightarrow$  "Para todo" ( $\forall$ ).

$$\begin{cases} P \rightarrow q \not\equiv P \wedge q \\ P \rightarrow q \equiv \neg P \vee q \end{cases}$$

- DNI de los clientes con préstamo de más de 6.000 €.

$$P(DNI) \text{ s } (I > 6000) \text{ (PRESTAMO)}.$$

$$\text{dom}(p) = \text{PRESTAMO}$$

$$\{ t_{12} / (\exists p) ((p[DNI] = t[DNI]) \wedge (p(I) > 6000)) \}.$$

- DNI y ciudad donde residen los clientes con préstamo en la sucursal con código 1.

$$P(DNI, CDC) \text{ s } (CS = 1) \text{ (PRESTAMO * CLIENTE)} \rightarrow \text{Y.N.} \Rightarrow \text{Unión} \times \text{el campo comù.}$$

$$\text{dom}(p) = \text{PRESTAMO} \quad \text{dom}(c) = \text{CLIENTE}$$

$$\{ t_{12} / (\exists p, c) (p[CS] = 1) \wedge (p[DNI] = c[DNI]) \wedge$$

$$(t[DNI] = c[DNI]) \wedge (t[CDC] = c[CDC]) \}.$$

$$(\forall x)(P_x) = \neg(\exists x)\neg(P_x) \quad \text{"Para todo" o "Ninguno".}$$

Sobre el conjunto vacío se puede afirmar lo que sea, que será cierto.  
Ejemplo: Disparate → Los pelícanos de la clase son rubios.

$$\{ t_{13} / \neg(\exists s) (s[CDC] = 'La Laguna') \wedge$$

$$\neg(\exists d) ((t[DNI] = d[DNI]) \wedge (d[CS] = s[CS])) \}$$

- DNI del cliente de la sucursal con código 1 que tiene menor saldo.

### \* Algebra Relacional

$\text{dom}(d^1) = \text{dom}(d^2) = \text{DEPOSITO}$

$$\left\{ t_d / (\exists d^1) ((t[\text{DNI}] = d^1[\text{DNI}]) \wedge (d^1[\text{CS}] = 1)) \wedge \right. \\ \left. (\forall d^2) ((d^2[\text{CS}] \neq 1) \wedge (d^2[\text{SLD}] \geq d^1[\text{SLD}])) \right\}.$$

El mínimo es un elemento del conjunto y no existe otro elemento menor que él.

$$\left\{ t_d / (\exists d^1) ((t[\text{DNI}] = d^1[\text{DNI}]) \wedge (d^1[\text{CS}] = 1)) \wedge \right. \\ \left. (\exists d^2) ((d^2[\text{CS}] = 1) \wedge (d^2[\text{SLD}] \geq d^1[\text{SLD}])) \right\}.$$

### 2.8.3 - Cálculo Restringido de T-uplas.

Predicado Sano  
seguro

El cálculo se restringe para que sea sano y que se pueda evaluar en un número finito de operaciones. Un predicado es sano si tiene equivalente en álgebra relacional. El cálculo restringido obliga a que un predicado sea sano.

El álgebra es al menos tan potente como el cálculo relacional.

A.R.  $\Rightarrow$  C.R.T.  $\rightarrow$  Misma potencia expresiva.

### 2.8.4 - Potencia expresiva del C.R.T.

d) ¿Alquier cosa que se puede escribir en álgebra, también se puede escribir en cálculo? Para hacerlo, tenemos que comprobar que los operadores tengan equivalentes.

$$P(A)(R) = \{t_{(A)} / (\exists x) (t[A] = x[A])\}$$

$$S(F)(R) = \{t / t \in R \wedge (F)\}$$

$$R \times S = \{ t_{r+s} / ( \exists x, y ) ((t[r] = x[r]) \wedge (t[s] = y[s])) \}$$

$$RUS = \{ t / t \in R \wedge t \in S \}$$

$$R - S = \{ t / t \in R \wedge t \notin S \}$$

Por tanto,  $CRT \succcurlyeq AR \} \leftrightarrow AR \equiv CRT$ . Son equipotentes.

$$AR \succcurlyeq CRT$$

Cualquier cosa que escriba en álgebra se puede escribir en cálculo restringido de t-uplas.

### 2.8.5. Definición formal de CRT.

Todas las consultas satisfacen esta forma:  $\{ t_a / p(t) \}$

Está formado por átomos y conectores lógicos.

Los átomos son de tres tipos:

$$1) s \in R$$

$$2) s[x] @ t[y]$$

$$3) s[x] @ c \text{ (constante)}$$

$$@ \in \{ <, >, <=, >=, =, <> \}$$

los predicados satisfacen las siguientes reglas:

a) Un átomo es un predicado. (Listar toda R).

b) Si  $p^1, p^2$  son predicados  $\Rightarrow p^1 \wedge p^2, p^1 \vee p^2, \neg p^1, (p^1) =$  son <sup>fórmulas</sup> predicados.

c) Si  $P(x)$  es un predicado  $\Rightarrow (\exists x)(P_x), (\forall x)(P_x) =$  son predicados.

En función de una variable, también lo son exponencialmente el resto de las variables.

d) La única variable libre que puede haber es t, el resto de variables tienen que estar cuantificadas.

e) Nada más es un predicado.

fórmula

## 2.8.6. Cláusulas de Dominios (CRD)

Las variables tienen valores del dominio de algún atributo.  
Conjunto de tuplas se denota así:  $\{x_1, x_2, \dots, x_n\} / P(x_1, x_2, \dots, x_n)\}$

Átomos son:

- 1)  $\langle s_1, \dots, s_n \rangle \in R$ .
- 2)  $x @ y$ .
- 3)  $x @ c$ . (Ejemplo:  $c_s = 1$ ).

Predicados son:

- a) Un átomo es un predicado.
- b) Si  $p^1, p^2$  son predicados  $\Rightarrow p^1 \wedge p^2; p^1 \vee p^2; \neg p^1; (p^1) =$  son predicados.
- c) Si  $P(x)$  es un predicado  $\Rightarrow (\exists x)(P_x); (\forall x)(P_x) =$  son predicados.
- d) Las únicas variables libres son:  $\langle x_1, x_2, \dots, x_n \rangle$ .
- e) Nada más es un predicado.

- DNI de las personas que tienen al menos un préstamo en el banco con un importe superior a 100.000 €.

$$P(DNI) \leq (I > 100.000) \text{ (PRESTAMO)}$$

$$\{ \langle dñi \rangle / (\exists cs, np, i) ((\langle cs, np, dñi, i \rangle \in \text{PRESTAMO}) \wedge (i > 100.000)) \}$$

- DNI de las personas que tienen una cuenta o un préstamo en la sucursal con código 1.

$$\{ \langle dñi \rangle / (\exists nc, sd) (\langle 1, nc, sd, dñi \rangle \in \text{DEPOSITO}) \vee (\exists np, i) (\langle 1, np, i, dñi \rangle \in \text{PRESTAMO}) \}$$

usamos la variable  
cs, como una constante  
y nos ahorramos una  
condición ( $cs = 1$ ).

↓  
otra forma de hacerlo:  
 $\{ \wedge \neg (\exists \dots) \mid \text{Negando una de las dos partes} \}$

→ Reglas de estilo:

- Variables en minúsculas.
- Consultas significativas.
- consultas lo más sencillas posible.

- DNI de los titulares de préstamos en la sucursal con código 1 y con un préstamo superior a 100.000€.

$$P(DNI) \cdot S((CS=1) \wedge (I > 100.000)) \text{ (PRESTAMO).}$$

$$\{ t_{\alpha} / (\exists p) (t[DN] = p[DN]) \wedge (p[I] > 100.000) \wedge (p[CS] = 1) \} \quad \text{dom}(p) = \text{PRESTAMO}.$$

$$\{ \langle dñi \rangle / (\exists np, i) (\langle 1, np, i, dñi \rangle \in \text{PRESTAMO}) \wedge (i > 100.000) \}$$

- DNI de los clientes que tienen cuenta en todas las sucursales de La Laguna. ↔ Para cualquiera que sea la sucursal, tendrá cuenta.

$$\{ \langle dñi \rangle / (\forall cs, nbs) \underbrace{(\langle cs, nbs, 'La Laguna' \rangle \in \text{SUCURSAL})}_{cd's} \wedge \underbrace{\exists nc, sld} \quad \begin{array}{l} \text{Para cualquier sucursal} \\ \text{de La Laguna.} \end{array} \\ (\langle cs, nc, dñi, sld \rangle \in \text{DEPOSITO}) \}$$

Con un saldo superior a 1000€: en cada una de ellas.

$$\wedge (sld > 1000) \quad // \text{En alguna con un saldo} > 10.000. \quad *$$

- DNI del titular de la cuenta con menor saldo en la sucursal con código 1.

$$\{ \langle dñi \rangle / (\exists nc, sld) (\langle 1, nc, dñi, sld \rangle \in \text{DEPOSITO}) \wedge \underbrace{\exists nc', sld', dñi'}_{\text{Elemento del conjunto.}} (\langle 1, nc', dñi', sld' \rangle \in \text{DEPOSITO}) \wedge (sld' \geq sld) \}$$

$$\{ \langle dñi \rangle / (\exists nc, sld) (\langle 1, nc, dñi, sld \rangle \in \text{DEPOSITO}) \wedge \neg (\exists nc', sld', dñi') (\langle 1, nc', dñi', sld' \rangle \in \text{DEPOSITO}) \wedge (sld' < sld) \}.$$

## TEMA 3: SQL

### 3.1 - Introducción: (Query Language).

Lenguaje para acceder a la base de datos.

DML, DDL y DCL (concede privilegios, controla transacciones, etc...).

Es un lenguaje de tipo declarativo y tiene un planificador de consulta.

Hay otros lenguajes, pero ninguno es tan importante.

Desarrollo:

SEQUEL → SQUARE → SEQUEL → SQL (ORACLE) (1976).

### 3.2 - Conceptos fundamentales:

Las sentencias van a estar formadas por cláusulas, y éstas tienen un orden.

Reglas de estilo:

- las palabras reservadas van en mayúscula.

- cada cláusula va en una línea distinta.

### 3.3 - DML.

#### 3.3.1 - Consultas. Recuperación de datos.

SELECT CS  
FROM DEPOSITO; } P(CS) (DEPOSITO) → Realmente es una proyección.

DISTINCT:

SQL no elimina filas repetidas; para hacerlo hay que quitar duplicados.

Definición: unique

SELECT [DISTINCT | ALL] \* {lista expresiones [AS] "alias"} }.

FROM — ;

↓  
Por defecto.

- Proyectar la tabla DEPOSITO.   

```
SELECT *
FROM DEPOSITO;
```
- Muestra el 110% de los saldos de las cuentas.   

```
SELECT SLD * 1.1
FROM DEPOSITO;
```
- Se puede poner un alias:   

```
SELECT SLD * 1.1 AS "110% Saldo"
FROM DEPOSITO;
```

### FROM ; WHERE

Equivale al producto cartesiano. Despues de él se pueden poner una lista de tablas.

FROM lista\_tablas

[WHERE predicado] → operador selección.

Pueden aparecer operadores comparacionales, operadores lógicos; operadores aritméticos (AND, OR, NOT, (), ); (=, >, <, <=, >=).

- DNI de clientes con saldo superior a 1000 y sucursal con código 1.

```
SELECT DNI
FROM DEPOSITO
WHERE (CS = 1) AND (SLD > 1000);
```

- Sucursales situadas en la laguna.

```
SELECT *
FROM SUCURSAL
```

```
WHERE (CDS = 'la laguna');
```

```
P(CS) S(CDS = 'la laguna'); (SUCURSAL)
```

- DNI de los clientes con alguna cuenta en alguna sucursal de la laguna.

```
SELECT DNI
FROM SUCURSAL, DEPOSITO
WHERE (CDS = 'la laguna')
AND (SUCURSAL.CS = DEPOSITO.CS);
```

$P(DNI) S(CDS = 'la laguna') (DEPOSITO * SUCURSAL)$

Se puede añadir un alias a las tablas, es como una vista pero declarada localmente.

```
SELECT DNI  
FROM SUCURSAL S, DEPOSITO D  
..... AND (S.CS = D.CS);
```

BETWEEN → Comprueba si la expresión pertenece al intervalo.

Establece una condición entre dos límites de valores.

[NOT] BETWEEN lim\_inf AND lim\_sup;

- Listar los datos de las sucursales cuyo código esté comprendido entre 1 y 100

```
SELECT *  
FROM SUCURSAL  
WHERE (CS >= 1) AND (CS <= 100);
```

↔

```
SELECT *  
FROM SUCURSAL  
WHERE CS BETWEEN 1 AND 100.
```

IN ↔ Pertenece.

Crea una condición en un conjunto de valores específico.

exp [NOT] IN conjunto\_valores.

④ Listar las sucursales cuyo código sea 1 o 3, o 5, o 7.

```
SELECT *  
FROM SUCURSAL  
WHERE CS IN (1,3,5,7);
```

⑤ DNI de las personas que tienen alguna cuenta en alguna sucursal de 'la legua'.

```
SELECT DNI  
FROM DEPOSITO  
WHERE CS IN (SELECT CS  
..... FROM SUCURSAL  
..... WHERE (CDS = 'La legua'));
```

↔ ANY forma implícita, resultado de una consulta.

→ Subconsulta  
consulta anidada.  
Estilo sgl.

```
SELECT DNI  
FROM DEPOSITO NATURAL JOIN SUCURSAL
```

} Estilo álgebra relacional.

ANY → Si se verifica el op. comp. para algún elem. del conjunto.

ANY ; ALL

Operadores comparacionales cualificados.

exp op-comparacional ANY / ALL conjunto\_valores. (explícita o implícita).

$A \leq \text{ANY/ALL} (\text{conjunto}) \rightarrow$  Verdadero si A es  $\leq$  que "algun", "todos", elementos del conjunto.

ALL → Op. comp. satisface para

④ Operadores que no se usan: todos los elementos del conjunto.

→ = ANY ↔ IN (Algun elemento del conjunto).

→ = ALL ↔ FALSO (Nunca se usa).

→ <> ANY ↔ No se usa, pues siempre es falso. verdadero

→ <> ALL ↔ NOT IN, (distinto de todo, no está en el conjunto).

↳ Siempre verdadero. No se usa.

- DNI del cliente de la sucursal con código 1 y con mayor saldo.

SELECT DNI

FROM DEPOSITO

WHERE (CS=1) AND (SLD) >= ALL (SELECT SLD

$>= \text{ALL} \rightarrow \text{Maximos}$

$<= \text{ALL} \rightarrow \text{Minimos}$

FROM DEPOSITO

WHERE (CS=1);

Si queremos → ANY  
Le cambia jerar.  
el resultado sera el que sea  
tener mas saldo

si la subconsulta devuelve verdadero.

EXISTS

Operador lógico. Devuelve verdadero si la consulta tiene alguna fila y falso en caso contrario. Quantificador existencial del álgebra relacional.

[NOT] EXISTS <consulta>

- Personas que tienen cuenta en todos los sucursales.

dom(s) = SUCURSAL : dom(d) = DEPOSITO

{ $\exists t / (\forall s)(\exists d)((t[CDNI] = d[CDNI]) \wedge (d[CS] = t[CS]))$ }

Para cualquier sucursal, el cliente tiene cuenta en esa sucursal. Para SQL:

{ $\exists t / (\forall s)(\exists d)((t[CDNI] = d[CDNI]) \wedge (d[CS] = t[CS]))$   
 $\exists s \exists d ((t[CDNI] = d[CDNI]) \wedge (d[CS] = s[CS]))$ }

```

SELECT DNI
FROM DEPOSITO T
WHERE NOT EXISTS (SELECT *
                   FROM SUCURSAL S
                   WHERE NOT EXISTS (SELECT *
                                      FROM DEPOSITO D
                                      WHERE (T.DNI = D.DNI)
                                         AND (D.CS = S.CS)));

```

} No existe que el cliente tenga cuenta.  
 } El cliente tiene cuenta.

- clientes con cuenta en todos los sucursales de 'la laguna'.

$$\begin{aligned}
 & \{ \exists d : \text{dom}(s) = \text{SUCURSAL} \wedge \text{dom}(d) = \text{DEPOSITO} \rightarrow (\exists d) \\
 & \{ \exists d : ((\exists s)((s.CS) = 'la laguna') \wedge (t.DNI) = d.DNI) \wedge (d.CS = s.CS)) \vee \\
 & \quad (\exists d)(\exists s)((s.CS) = 'la laguna') \wedge (t.DNI) = d.DNI) \wedge (d.CS = s.CS)) \}.
 \end{aligned}$$

```

SELECT DNI
FROM DEPOSITO T
WHERE NOT EXISTS (SELECT *

```

} No existe una sucursal en la laguna donde el cliente no tiene cuenta.

- clientes para los cuales no puede ocurrir que exista una sucursal en la laguna en la que el cliente no tenga cuenta.

```

                   FROM SUCURSAL S
                   WHERE (CS = 'la laguna')

```

} No existe una sucursal en la laguna

```

                   AND NOT EXISTS (SELECT *

```

```

                   FROM DEPOSITO D

```

```

                   WHERE (T.DNI = D.DNI)

```

```

                   AND (D.CS = S.CS));

```

} El cliente tiene cuenta.

## LIKE

Operador lógico. Compara una cadena de caracteres con un patrón.

En el patrón se pueden introducir los caracteres salvajes:

wild cards

- % y -

--> cualquier carácter.

- %> cualquier cadena. (incluida la cadena NULL).

[NOT] LIKE 'patrón' → <cadena> LIKE 'patrón'.

- Nombres de clientes cuyos nombres empiezan por 'J'.

```
SELECT NBC  
FROM CLIENTE WHERE NBC LIKE 'J%';
```

(LENGTH)

- Nombres de clientes con a lo sumo cuatro caracteres.

```
SELECT NBC  
FROM CLIENTE  
WHERE NBC NOT LIKE ('-----%'); → No tiene cinco caracteres o más.
```

IS NULL

Operador lógico. Verdadero si la expresión toma valor nulo y falso en caso contrario. IS [NOT] NULL <expresión>.

- DNI de los profesores que dan clase actualmente.

```
SELECT DNI  
FROM PLAN_DOCENTE  
WHERE FF IS NULL;
```

```
SELECT DNI  
FROM CLIENTE  
WHERE CDC IS NULL;
```

- DNI de los clientes para los que no se conoce su ciudad de residencia ⇒

```
SELECT DNI  
FROM CLIENTE C  
WHERE NOT EXISTS (SELECT *
```

```
      FROM DEPOSITO D  
      WHERE (C.DNI = D.DNI) AND (CS = 1));
```

} Cliente para el cual no ocurre:

} Tiene cuenta en la sucursal 1.

```
SELECT DNI  
FROM DEPOSITO  
MINUS (SELECT DNI  
       ↓  
       FROM DEPOSITO  
       ↓  
       WHERE (CS = 1));
```

[WHERE DNI NOT IN]

## FUNCIONES DE GRUPO:

La tabla de resultados devuelve un único grupo. Son cosas que resumen del grupo (media, desviación típica, mínimo, máximo, etc.).

. Media: AVG ([DISTINCT ALL] exp).

. Suma: SUM ( " " )

. Mínimo: MIN ( " " ). Variancia: VARIANCE ( " " ).

. Máximo: MAX ( " " ). Desviación Típica: STDEV ( " " ).

. Contar: COUNT ; COUNT ([DISTINCT ALL] exp | \*).

Cuenta el número de filas que tiene la consulta. ↳ "todas", incluyendo los null y los duplicados.

→ las funciones de grupo van después del select: No se mezclan con atributos.

- Saldo medio de las cuentas de la sucursal con código 1.

```
SELECT AVG (SLD)  
FROM DEPOSITO  
WHERE (CS=1);
```

} las cuentas pueden tener varios titulares, por tanto se cuentan varias veces. Dos cuentas distintas con el mismo saldo.

```
SELECT AVG (SLD)
```

```
FROM (SELECT DISTINCT CS, NC, SLD FROM DEPOSITO) A
```

① → Vista implícita.  
WHERE CS=1;

Alias

## GROUP BY

Operador lógico. Crea grupos atendiendo a igualdad de valores para las expresiones.

- Para cada sucursal, el número de cuentas corrientes que tiene.

```
SELECT CS, COUNT (DISTINCT NC)  
FROM DEPOSITO  
GROUP BY CS;
```

} N° de cuentas distintas que tiene.  
cuantos más atributos hay en el select, más pequeños son los grupos.

## HAVING <predicado>

Selecciona aquellos grupos que satisfacen el predicado.

WHERE → filas ; HAVING → Grupos.

↳ - solo grupos que cumplen el predicado

- DNI de los clientes que tienen más de dos cuentas.

SELECT DNI

FROM DEPOSITO

GROUP BY DNI, CS

HAVING COUNT(\*) > 2;

} Más de dos cuentas en una misma sucursal.

↳ Podría ser una subconsulta.

- DNI de los clientes que tienen cuenta en todas las sucursales. (saber SQL)

SELECT DNI

FROM DEPOSITO

GROUP BY DNI

HAVING COUNT(DISTINCT CS) = (SELECT COUNT(\*)

FROM SUCURSAL);

Nº de sucursales distintas en  
las que tiene cuenta el cliente.

Nº de sucursales que tiene el banco.

## ORDER BY

Ordena la salida de una consulta. No se puede poner en una subconsulta,  
sólo en la consulta principal.

[ ORDER BY exp: [ASC | DESC] [, ...] ].  
por defecto.

Última cláusula del select. Sólo una por consulta.

Ordena de izquierda a derecha.

- Listar en orden inverso los nombres de los empleados de la  
sucursal 1.

SELECT NBC

FROM CLIENTE NATURAL JOIN PRESTAMO

## OPERADORES DE CONJUNTO:

- 1) UNION <consulta 1>.  $\Rightarrow$  consulta1 UNION consulta2.
- 2) INTERSECT <op.conjunto>.
- 3) MINUS <consulta 2>.
  - 1) une filas de la consulta 1 con las de la consulta 2. Los repetidos sólo los une una vez. UNION ALL  $\Rightarrow$  une todos los filas.
  - 2) Intersecta las filas de la consulta 1 con la consulta 2.
  - 3) MINUS  $\leftrightarrow$  EXCEPT

② - Clientes que tienen cuenta en las mismas sucursales que el cliente con dñi = 1111.

SELECT DNI

FROM DEPOSITO D1

WHERE NOT EXISTS (SELECT CS FROM DEPOSITO } Sucursales de dni 1111.  
WHERE (DNI = 1111)  
MINUS } Resta.

SELECT CS FROM DEPOSITO D2 } Persona que queremos  
WHERE (D1.DNI = D2.DNI); } proyectar.

Tiene que dar vacío, puesto que tendría que tener cuenta en las mismas sucursales que la persona con DNI = 1111.

- Listar los códigos de aquellas sucursales situadas en la misma ciudad que la sucursal con código 1.

SELECT CS

FROM SUCURSAL

WHERE CDS = (SELECT CDS

FROM SUCURSAL

WHERE CS = 1);

} Código de sucursales.

} Ciudades.

Retorna un único valor, ya sea un número o un escalar.

La clave primaria de la tabla SUCURSAL es CS, por ello sabemos que retorna un único valor. Es el nivel más fácil.

La subconsulta puede devolver un vector de valores, una columna. Es el siguiente grado de dificultad.

- DNI del cliente con menor saldo en la sucursal con código 1.

```
SELECT DNI  
FROM DEPOSITO  
WHERE (CS=1) AND (SLD <= ALL (SELECT SLD  
                                FROM DEPOSITO  
                                WHERE (CS=1)));
```

El operador tendrá que estar cuantificado.

Subconsultas que devuelven una matriz. Siguiente grado de dificultad.

- Cuentas titulares de los clientes del cliente con DNI 1111.

```
SELECT DNI  
FROM DEPOSITO  
WHERE (CS,NC) IN (SELECT CS,NC  
                    FROM DEPOSITO  
                    WHERE (DNI = 1111));
```

### FROM

FROM A CROSS JOIN B : Producto cartesiano entre A y B.

FROM A NATURAL [INNER] JOIN B : Unión Natural entre A y B.

FROM A INNER JOIN B : Unión entre A y B: on predicado. RY(F)S = S(F) (R x S)

FROM A INNER JOIN B USING (Columnas) (de las que queremos la igualdad).

FROM A LEFT [OUTER] JOIN B : Unión Natural externa a la izquierda.

FROM A RIGHT [OUTER] JOIN B : " " " " " derecha.

FROM FULL [OUTER] JOIN : " " " " " completa.

- Nombres de los clientes que tienen cuenta en la sucursal con código 1.

```
SELECT NBC  
FROM CLIENTE NATURAL INNER JOIN DEPOSITO  
WHERE CS=1;
```

### 3.3.2 Actualización.

- Inserción
- Borrado
- Modificación.

\* Inserción: Insertar en una tabla el resultado de una consulta.

INSERT INTO table [ (col1, col2, ..., coln) ] [opcional]

[VALUES (...,...,...); | \_consulta\_ ] ↳ El resto de columnas se llenan con NULL o con el valor por defecto al crear la tabla.

- Insertar en la tabla 'cliente1', las filas a los clientes con cuenta en la sucursal con código 1.

INSERT INTO CLIENTE1

SELECT \*

FROM CLIENTE

WHERE DNI IN (SELECT DNI FROM DEPOSITO  
WHERE CS=1);

- Crear una base de datos a partir de otra, no se suele hacer.

\* Borrado:

DELETE FROM table; → Borrado total de la tabla, el objeto existe, pero vacío.

DELETE FROM table [alias]

[WHERE predicado]; → Borra las filas que cumplen el predicado.

También se puede borrar con una consulta.

- Borrar de la tabla DEPOSITO la cuenta con código 1 y número de cuenta 100.

DELETE FROM DEPOSITO

WHERE (CS=1) AND (NC=100); → Se produce borrado en cascada.

Hay peligroso, se puede corromper la base de datos.

Para hacer los cambios permanentes, hay que confirmarlos.

### \* Modificación:

Modificaremos los valores de los atributos en las filas de la base de datos.

UPDATE tabla [alias]

SET col1 = exp<sup>1</sup> [col2 = exp<sup>2</sup>, ...] → Todas las filas de la tabla.

[WHERE predicado];

exp<sup>i</sup> → Puede ser una constante, el contenido de un campo sumado o multiplicado con algo y/o una consulta.

- Hacemos un ingreso de 100€ en la cuenta número 100 de la sucursal 1.

UPDATE DEPOSITO

SET SLD = SLD + 100

WHERE (CS = 1) AND (NC = 100);

→ Se hará tantas veces como titulares tenga la cuenta.

- Incrementar el saldo de la cuenta 100 de la sucursal con código 1 con el 50% del máximo saldo de la sucursal 1.

UPDATE DEPOSITO

SET SLD = SLD + 0.5 \* (SELECT DISTINCT MAX(SLD) FROM DEPOSITO  
WHERE (CS = 1))

WHERE (CS = 1) AND (NC = 100);

Cuando se utiliza una aplicación con código, dentro de las consultas utilizamos variables.

UPDATE DEPOSITO

SET SLD = SLD + :int

WHERE (CS = :cs) AND (NC = :nc);

→ lo normal es hacer la modificación a mano.

### 3.4 DDL

Sentencias de definición de datos. Actúa directamente sobre el contenedor y no sobre el contenido. Todas las sentencias DDL hacen un "commit work" antes y después de ejecutar la sentencia.

#### 3.4.1 - Tipos de dominios

SQL Oracle: (nativo)

CHAR [(n)] → Cadena de caracteres de longitud fija. Si no especificas  $n=1$ .

VARCHAR2 [(n)] → Cadena de longitud variable. ( $2 \rightarrow 70$ ).  $n \geq 1$ .

NUMBER [p,[s]] → p(precisión) N° de dígitos totales, no incluye el punto.  $p \geq 1$ .

s(escala) N° de dígitos decimales.  $0 \leq s \leq p$ . Entero. Defecto  $s=0$

DATE → Almacena fechas y horas. Como un string de caracteres. Distingue sus campos (siglo, año, mes, hora,...). Cuando sólo se introduce el día, la hora se pone en 12:00:00, y al contrario mes: 01-mes.

Se pueden restar dos fechas.  $d_1 - d_2 = n$ , da un número, que es el número de días transcurridos. A una fecha también se le puede sumar un número entero o decimal, y da una fecha:  $d_1 = d_2 + n$ .

Range de valores de fechas: [-4712 ... 4712].

ANSII SQL	SQL Oracle
CHAR (n)	CHAR (n)
CHARACTER (n)	
CHAR VARYING (n)	VARCHAR2 (n)
NUMERIC (p,s)	NUMBER (p,s)
DECIMAL (p,s)	
INT	
INTEGER	NUMBER (p)
SMALLINT	
FLOAT (b)	
DOUBLE PRECISION	NUMBER Max Precision;
REAL	
DATE	DATE

### 3.4.2. Tablas:

#### \* Crear tablas:

CREATE TABLE nombre\_table

(def col1,

def col2, ...

def coln,) → Podemos acabar aquí, o añadir condiciones de integridad:

[cond-int-table1,

cond-int-table n];

def.col = Nombre.Tipo de dato [DEFAULT] [cond]

Ejemplo: CREATE TABLE CLIENTE

(DNI NUMBER (4) PRIMARY KEY

NOMBRE VARCHAR2 (15)

CEDULA VARCHAR2 (20);

Se almacena en el diccionario el contenido del objeto.

Se puede crear una tabla y la carga con el resultado de una consulta.

CREATE TABLE nombre\_table [(col1, col2, ..., coln)]

AS consulta;

Crea el objeto y lo carga en la misma operación.

- Clientes con cuentas:

CREATE TABLE CLIENTE1

AS (SELECT \* FROM CLIENTE

WHERE DNI IN (SELECT DNI FROM DEPOSITO));

\* Borrar tablas: Elimina contenido del objeto, la definición en el diccionario, los índices, las vistas, ... Borra el contenido y el contenedor.

DROP DATE.

DROP DATE nombre\_table [CASCADE CONSTRAINTS].

\* Cambiar nombre a una tabla: Tablas, vistas o sinónimos.

RENAME nombre\_actual TO nombre\_nuevo;

\* Crear alias a una tabla: Ya sabemos hacerlo de forma local, en el ámbito de una sentencia. De forma global sería:

CREATE SYNONYM nombre FOR tabla;

Para las COLUMNAS podemos hacer las siguientes actualizaciones:

① \* Modificar la estructura de una tabla: Añadir nuevas columnas, borrarlas, renombrarlas. Modificar la definición de las columnas.

ALTER TABLE tabla

RENAME COLUMN col\_vieja TO col\_nueva

ADD (def\_col); → Añade columnas en el extremo derecho con todos los valores a NULL o valor por defecto.

No añadir condición integridad antes de rellenar la columna.

- Añadir teléfonos a la tabla CLIENTE.

ALTER TABLE CLIENTE

ADD (TELEFONO NUMBER (9));

\* Borrar una columna: Puede generar problemas. Puede haber referencias a la columna que estamos eliminando, aunque se puede borrar en cascada.

DROP COLUMN [CASCADE CONSTRAINTS]

\* Modificar una columna: Incrementar el ancho, seguro. Decrementar el ancho, inseguro, perdida de datos.

NUCLEO → TRUNCATE TABLE NOMBRE  
limpiar.

MODIFY (def\_col); → columna existente.

SI SE RECUPERAN LOS DATOS DE LA TABLA;  
DELET FROM TABLA;

### 3.4.3. VISTAS.

\* Crear Vistas: Podemos crear vistas y luego realizar una consulta. Es igual que una vista en álgebra relacional. También se puede basar en otra vista, o por defecto, en tablas. Al crear la vista se almacena la definición en el diccionario. No ocupa espacio. Son lentas.

```
CREATE VIEW nombre_vista  
AS consulta;
```

- DNI de clientes con cuenta en la sucursal con código 1.

```
CREATE VIEW D1  
AS (SELECT DNI  
    FROM DEPOSITO  
    WHERE (CS=1));
```

La vista se calcula cuando:

```
SELECT * FROM D1;
```

Vistas materializadas: CREATE [MATERIALIZED] VIEW nombre\_vista;

son rápidas. El contenido está almacenado implícitamente. ocupan espacio.

Podemos crear vistas sobre vistas.

\* Borrar Vistas: DROP [MATERIALIZED] VIEW

\* Los vistos se basan en tablas. No se pueden modificar. Lo que se modifica son las tablas en las que se basa la vista.

### 3.4.4. INDICES.

Estructura de datos auxiliar que permite acelerar las búsquedas. Se suelen indexar los atributos que más frecuentemente aparecen en el WHERE o en el HAVING. Funcionan igual que el índice de un libro. Si se añaden nuevas páginas, hay que actualizarlo.

Aceleran las búsquedas, pero consumen espacio y hacen que las actualizaciones sean más lentas. Un índice se crea cuando hay un número considerable de filas.

Lunes Martes Miércoles Jueves  
 Pechos. Otros. Cerdas Pecho/Pizmaz  
 bíceps. Biceps/biceps Cerdas Cerdas

En la vida real no se actualizan, por tanto hay que reconstruirlos.

Se pueden comprimir indices si tienen atributos que permiten valores duplicados.

\* Crear un indice: No se puede comprimir, no acepta valores repetidos.

`CREATE [UNIQUE] INDEX nombre_index`

`ON tabla (col1,col2,...);` → Índice único (1 columna única).  
 → Compuesto (2 columnas).

Podemos crear indices con el mismo nombre sobre tablas distintas.

\* Borrar un indice: `DROP INDEX nombre_index;`

[ON TABLA]; → si hay del mismo nombre para tablas distintas.

\* Modificar un indice:

`ALTER INDEX nombre_index`

`RENAME TO nombre_nuevo;`

[ENABLE | DISABLE] → Activarlo o desactivarlo. Por defecto está activado.  
 Se desactiva para una inserción masiva.

\* Reconstruir un indice:

`REBUILD;`

[COMPRESS | NO COMPRESS], compificados ocupan menos espacio pero son más lentos.

### 3.4.5. Control de integridad.

Condiciones que tienen que satisfacer los datos para que sean formalmente válidos.

[CONSTRAINT nombre] condición [DISABLE]

Si satisface la condición de integridad, deja activa la sentencia.

Si no ponemos nombre, automáticamente asigna uno: SYS\_C\_\_\_\_\_.

Una condición de integridad se define al crear la tabla. Se pueden definir a nivel de columna o a nivel de tabla.

`CREATE TABLE tabla`

`(def_col def_cond_int);` // A nivel de columna.

`(def_cond_tabla)` // A nivel de tabla.

- intrarrelación  
 - interrelación

- estado  
 - transición

Podemos querer una condición de integridad con la tabla ya creada.

### SINTAXIS

ALTER TABLE tabla

- Activar o desactivar una previamente creada.

{ENABLE | DISABLE | CONSTRAINT nombre;

- Borrar una cond.i. que ya existe.

DROP CONSTRAINT nombre [CASCADE];

RENAME CONSTRAINT nombre actual TO nombre nuevo;

MODIFY CONSTRAINT nombre nueva-def;

- Añadir una nueva condición. Tiene que tener la sintaxis de una condición de integridad a nivel de tabla.

CONSTRAINT nombre :

[CONSTRAINT nombre] NULL

NOT NULL

PRIMARY KEY → No permite null ni duplicados.

UNIQUE → Si null. No duplicados.

REFERENCES tabla [columna] [ON DELETE {CASCADE | SET NULL}]

CHECK predicado; → Comprueba si es verdadero.

- Definir la tabla CLIENTE:

CREATE TABLE CLIENTE

(DNI NUMBER(4) PRIMARY KEY → Condición de integridad a nivel de columna.

NOMBRE VARCHAR2(15)

CED\_VARCHAR2(15);

PRIMARY KEY (DNI); → Cond. integridad a nivel de tabla.

Condición de integridad a nivel de tabla:

[CONSTRAINT nombre]

PRIMARY KEY (col1, col2, ...)

UNIQUE (col1, col2, ...)

FOREIGN KEY (col1, col2, ...) REFERENCES tabla [(col1, col2, ...)] [ON DELETE {CASCADE | SET NULL}]

CHECK predicado;

- Nombres abreviados:

PK → Primary Key

FK → Foreign Key

N → NULL

UQ → Unique

CH → CHECK

NN → NOT NULL

- Definir la tabla DEPOSITO.

```
CREATE TABLE DEPOSITO
```

```
(CS NUMBER(2) CONSTRAINT FK_DEPOSITO_SUCURSAL REFERENCES SUCURSAL(CS) ON  
DELETE CASCADE,  
NC NUMBER(6)
```

```
DNI NUMBER(4)
```

```
SLD NUMBER
```

```
CONSTRAINT PK_DEPOSITO PRIMARY KEY (CS, NC, DNI));
```

Nivel de columna

```
ALTER TABLE DEPOSITO
```

```
ADD CONSTRAINT FK_DEPOSITO_CLIENTE
```

```
FOREIGN KEY(DNI) REFERENCES CLIENTE ON DELETE SET NULL;
```

→ Modificamos una condición de integridad de la tabla DEPOSITO.

- Que el saldo sea mayor o igual que 1000 €.

```
SLD NUMBER CHECK (SLD >= 1000)
```

A nivel de columna

- Tabla SUCURSAL (CS, NBS, CDS) ya creada. Modificar el tipo de CS a un rango entre 1 y 100; y no entre [-99, 99].

```
ALTER TABLE SUCURSAL
```

```
ADD CONSTRAINT CH_CS_SUCURSAL CHECK CS BETWEEN 1 AND 99;
```

- El nombre de la sucursal tiene que estar en mayúsculas.

```
ALTER TABLE SUCURSAL
```

```
ADD CONSTRAINT CH_NBS_SUCURSAL CHECK NBS = UPPER(NBS);
```

- las sucursales tienen que estar en La Laguna, La Orotava y Santa Cruz.

```
ALTER TABLE SUCURSAL
```

```
ADD CONSTRAINT CH_CS_SUCURSAL
```

```
CHECK CDS IN ('Santa Cruz', 'La Laguna', 'La Orotava');
```

→ Barrer condiciones de integridad Primary Key

```
ALTER TABLE
```

```
OR REPLACE CONSTRAINT nombre [PRIMARY KEY] [CASCADE]
```

## ASERCIÓNES:

Algo que se afirma como algo cierto. Afectan a varias tablas.

\* Crear un aserto: CREATE ASSERTION nombre\_aserto  
CHECK predicado [ENABLE];

ALTER ASSERTION nombre\_aserto [RENAME TO nombre\_nuevo] ;  
[ENABLE | DISABLE];

\* Borrar un aserto: DROP ASSERTION nombre;

Los asertos ralentizan las actualizaciones, consumen muchos recursos.

- Los titulares de un préstamo tienen que tener, al menos, una cuenta con un saldo superior a 1.000 €.

( $\forall p$ ) ( $\exists d$ ) (( $p[DNI] = d[DNI]$ )  $\wedge$  ( $d[SLD] > 1000$ ))

$\neg(\exists p) \neg(\exists d) ((p[DNI] = d[DNI]) \wedge (d[SLD] > 1000))$

CREATE ASSERTION AS1

CHECK NOT EXISTS (SELECT \* ...)

FROM PRESTAMO P

WHERE NOT EXISTS (SELECT \* FROM DEPOSITO D  
WHERE (P.DNI = D.DNI)  
AND (SLD > 1000));

- Todas las cuentas SLD > 1000. ( $\forall d$ ) ( $d[SLD] <= 1000$ ).

- Ninguna cuenta tiene un saldo superior a 1000€.

$\neg(\exists d) (d[SLD] > 1000)$

En los asertos no se usa el cálculo relacional porque intervienen funciones de grupo.

los saldos por cruce de cuenta tienen que ser el mismo

CREATE ASSERTION SLO\_UNICO\_mismo\_westa  
CHECK

$\neg((\exists a) (\exists b) (\exists c) (a \neq b) \wedge (a[SLD] \neq c[SLD]))$

- Una sucursal no puede prestar más dinero del que tiene en efectivo, la suma de todos los saldos de sus cuentas. No prestan lo que no tienen.

CREATE ASSERTION AS2

CHECK NOT EXISTS (SELECT \* FROM SUCURSAL S

WHERE (SELECT SUM (I)

FROM PRESTAMO P

WHERE (P.CS = S.CS) > (SELECT SUM (SLD)

FROM DEPOSITO D

WHERE (D.CS = S.CS));

(SELECT DISTINCT CS, NC, SLD  
FROM DEPOSITO)

Para contarlos una sola vez.

Realmente la consulta está mal porque no tiene en cuenta los importes y saldos repetidos, por los titulares.

### DEPENDENCIA FUNCIONAL

Cuando el valor del primer atributo define inequívocamente al segundo atributo. No puede ocurrir que en R haya dos filas con el mismo valor en A y distinto valor en B. El DNI depende funcionalmente de la fecha de nacimiento de la persona, siempre tienen que coincidir.

- Implementar condición de integridad con la fecha de nacimiento y el DNI.

$$\text{dom}(r^1) = \text{dom}(r^2) = R$$

$$\exists r^1, r^2 ((r^1[A] = r^2[A]) \wedge (r^1[B] \neq r^2[B]))$$

$$\forall r^1, r^2 ((r^1[A] \neq r^2[A]) \vee (r^1[B] = r^2[B]))$$

CREATE ASSERTION AS3

→ Saber Cálculo Relacional.

CHECK NOT EXISTS (SELECT \* FROM RR1, RR2

WHERE (R1.A = R2.A) AND (R1.B <> R2.B));

CREATE ASSERTION AS4

→ Saber SQL.

CHECK ① = NOT EXISTS (SELECT \* FROM R

Daría problemas al  
violar la condición de  
integridad.

GROUP BY A

HAVING COUNT (DISTINCT B) > 1);

### 3.4.6. Disparadores. TRIGGERS

Sinónimos: desencadenador, iniciador, gatillo,...

Conjunto de instrucciones que el sistema ejecuta de forma autónoma cuando ocurre algún evento y se verifica en una determinada condición.

Se basan en el modelo de: evento / condición / acción.

Se dice que es una base de datos activa si tiene disparadores.

Ejemplo: El banco automáticamente abre un préstamo a un cliente cuando el saldo de su cuenta sea negativo. Su cuenta se pondrá a cero.

Evento → Actualización en la tabla REGISTRO.

Condición → Si el saldo es negativo.

Acción → Inserta en la tabla PRESTAMO y actualiza la tabla DEPOSITO.

#### \* Crear un Disparador:

CREATE TRIGGERS "descubierto"

AFTER UPDATE OF SLD ON DEPOSITO

REFERENCING NEW AS nfila → Referenciado. Fila nueva después de la actualización

FOR EACH ROW → Hace cosas por cada fila que actualizo.

Variáble de transacción

WHEN (nfila.SLD < 0)

BEGIN → Conjunto de instrucciones. Acciones =.

INSERT INTO PRESTAMO VALUES (nfila.cs, nfila.nc, nfila.DNI, nfila.SLD).

UPDATE DEPOSITO SET SLD = 0

WHERE (CS=nfila.cs) AND (NC=nfila.nc) AND (DNI=nfila.DNI)

END;

Sintaxis: CREATE [OR REPLACE] TRIGGER nombre

{AFTER | BEFORE} {DELETE | INSERT | UPDATE [OF col]} ON tabla

[REFERENCING {NEW | OLD} [AS] nfila]

[FOR EACH ROW]

[WHEN :predicado]

BEGIN

END;

→ Si no se pone, se trata de un disparador por sentencia.

Hay que evitar que sea recursivo, que salte de nuevo el disparador. La base de datos se bloquee.

\* Borrar un disparador: DROP TRIGGER nombre;

\* Modificar un disparador: ALTER TRIGGER nombre {ENABLE | DISABLE}

- Cuando se borre una cuenta en DEPOSITO, comprobar si el titular tiene más cuenta y/o préstamos. Si no tiene, le damos de baja en la tabla CLIENTE.

Evento / Condición / Acción.

CREATE TRIGGER T1

AFTER DELETE ON DEPOSITO

REFERENCING OLD AS ofila

FOR EACH ROW

WHEN NOT EXISTS (SELECT \* FROM DEPOSITO

WHERE DNI = ofila.DNI

AND NOT EXISTS (SELECT \* FROM PRESTAMO  
WHERE DNI = ofila.DNI))

BEGIN

DELETE FROM CLIENTE WHERE (DNI = ofila.DNI)

END;

- Base de datos de un supermercado. Reponedor de estanterías.

STOCK (CP, CNT) → N° de productos que hay en el estante.

NIVEL-MINIMO (CP,NIVEL) → Umbral para repasar.

PEDIDOS (CP,CNT') → Por cada producto, unidades a repasar.

UNIDADES A PEDIR (CP,CNT') → Unidades que pedir.

Evento → Actualizar el stock de mercancía en las estanterías.

Condición → Comprobar el nivel del umbral. N° de productos que quedan.

Acción → Repasar el número correcto de productos.

### 3.5. DCL. Control de Datos.

#### 3.5.1. Gestión de transacciones:

T = Conjunto de operaciones que tienen que hacerse de forma única. Por ejemplo: transacción de dinero.

Una transacción está compuesta por muchas operaciones. Para oracle, todas es una transacción. Se pueden confirmar mientras las hacemos o las deshacemos. Se pueden establecer puntos de control y así deshacer parcialmente una transacción.

Confirmamos la transacción con: COMMIT [WORK];

También podemos deshacer una transacción por completo, o parcialmente:

ROLLBACK [WORK];

ROLLBACK [WORK] [TO [SAVEPOINT] pto\_control];

Establecer puntos de control: SAVEPOINT punto; (Etiqueta).

#### 3.5.2. Introducción a la seguridad:

Hay tres tipos de acceso mal intencionados a la base de datos:

- Robo de información.
- Actualizaciones de la base de datos para dejarla en estado inconsistente.
- Borrado no autorizado de la base de datos.

Hay varios niveles de seguridad:

- Nivel Base de Datos: concede privilegios a los usuarios. Tienen su propio nivel de seguridad. Cuentas de usuario.
- Nivel sistema operativo: si se accede al sistema operativo también se puede acceder a la base de datos.
- Nivel de Red.
- Nivel Físico: Ubicación del sistema.
- Nivel Humano.

### 3.5.3. Control de acceso:

El acceso se realiza a través de cuentas de usuario. Para crearlos utilizamos:

`CREATE USER nombre`

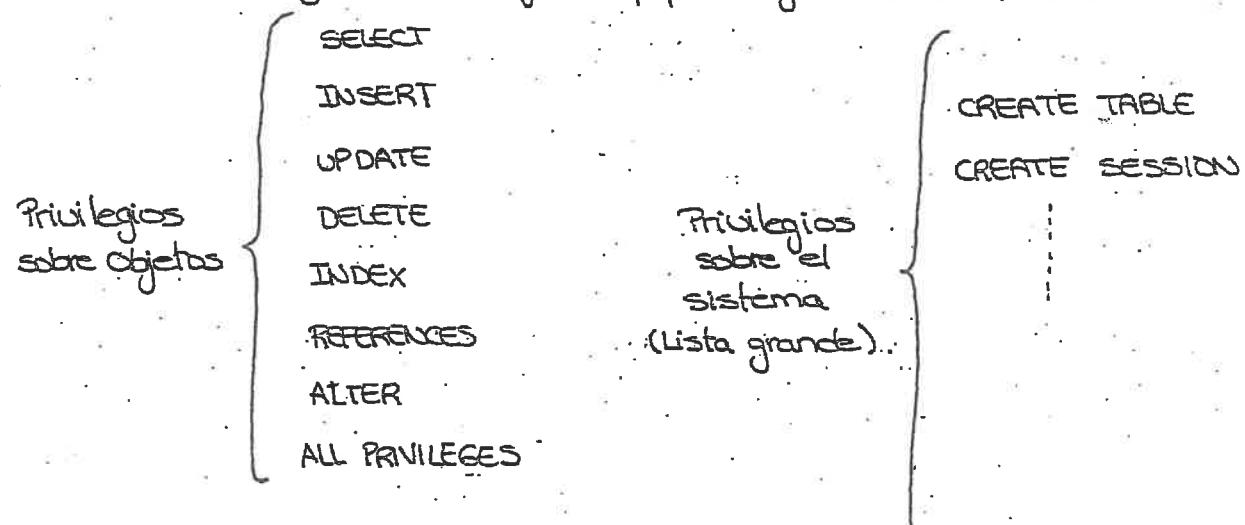
`IDENTIFIED password;`

El administrador modifica y borra al usuario.

`DROP USER nombre_usuario;`

### 3.5.4. Tipos de actualización:

Los usuarios pueden tener varios tipos de privilegios. Principalmente son de dos tipos: privilegios sobre objetos y privilegios sobre el sistema.



Al conceder el privilegio de conceder privilegios se crea un grafo. Si le quitamos el privilegio a alguien, se eliminan en cascada.

El propietario de un objeto creado por él, puede compartirlo.

### 3.5.5. Autorizaciones y vistas.

Si tenemos el privilegio de crear una vista, a través de ella podemos ver la tabla.

Para crear una vista tenemos que tener el privilegio de crearla y de consulta. Idem para las modificaciones o borrado a través de la vista.

### 3.5.6 - Concesión de privilegios:

- Conceder privilegios sobre objetos:

GRANT { lista\_privilegios | ALL PRIVILEGES }

ON objeto

TO { lista\_usuarios | PUBLIC }

[WITH GRANT OPTION] → Para transmitirlo a otros usuarios.

Se puede especificar en la lista de privilegios.

- Conceder privilegios sobre el sistema:

GRANT lista\_privilegios;

TO { lista\_usuarios | PUBLIC }

[WITH ADMIN OPTION].

### 3.5.7 - Eliminación de privilegios:

- Sobre objetos: REVOKE { lista\_privilegios | ALL PRIVILEGES }

ON objeto

FROM { lista\_usuarios | PUBLIC }.

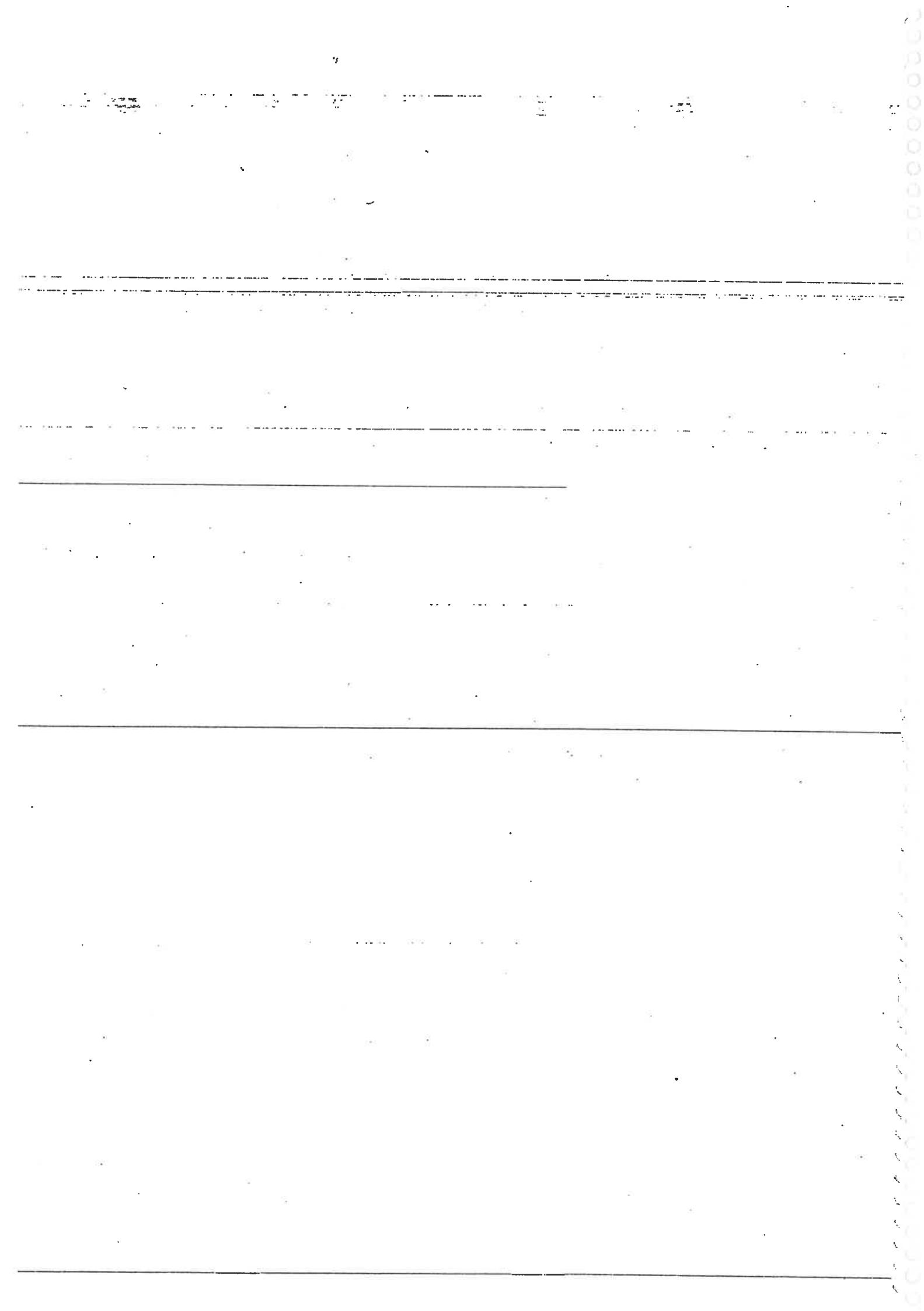
- Sobre el sistema: REVOKE lista\_privilegios

FROM { lista\_usuarios | PUBLIC }.

### 3.6.8. El concepto de rol o papel.

- Crear un rol :  
(Administrador)      CREATE ROLE nombre  
[NOT IDENTIFIED || IDENTIFIED BY password]  
Por defecto.
- Eliminar un rol :  
(Administrador)      DROP ROLE nombre;
- Modificar un rol :  
ALTER ROLE nombre  
[NOT IDENTIFIED || IDENTIFIED BY ...]
- Activar un rol :  
SET ROLE { nombre [IDENTIFIED BY password] , ... || →  
→ ALL [EXCEPT] lista-roles || NONE }.
- Conceder privilegios a los roles : Igual que a los usuarios.
- Conceder roles a los usuarios : Como privilegios en el sistema.  
GRANT → ROLE

DBA → Rol del administrador de la base de datos.



## TEMA 4 : GESTIÓN DE TRANSACCIONES

### 4.1 Concepto de transacción.

Conjunto de operaciones que desde un punto de vista lógico constituye una unidad. Operaciones de consulta y modificación.

Tienen que verificar unas propiedades. (ACID) (Iniciales de las 4 propiedades).

### 4.2 Propiedades ACID.

#### 1. Atomicidad (Atomicity):

O se ejecutan todas o no se ejecuta ninguna.

#### 2. Consistencia (Consistency):

La ejecución aislada de la transacción. Sin que se ejecute otra de forma concurrente con ella. Está bien programada, es lógica.

#### 3. Aislamiento (Isolation):

Aunque se ejecutan dos transacciones concurrentemente ( $T_i$  y  $T_j$ ) ocurre que a los efectos de  $T_i$ , es como si  $T_j$  hubiese acabado, antes que  $T_i$  comience, o viceversa.

No existe la concurrencia para el usuario.

#### 4. Persistencia (Durability):

Tras la finalización con éxito de una transacción, los cambios realizados perduran en el sistema, aunque este falle.

OLTP → Online Transactional Processing.

Hay dos tipos de acceso:

- Leer (x)
- Escribir (x) → No se hacen de forma inmediata (Lo normal).

→ Transacción 1, T<sub>1</sub>:

\* Pasar 50 € de una cuenta A, a una cuenta B.

Leer (A)

A = A - 50

// Supongo que A es mayor que 50.

Escribir (A)

Leer (B)

// Se pueden mover ciertas operaciones, otras no.

B = B + 50

Escribir (B)

La suma de los saldos de A y B es la misma antes y después de ejecutar la transacción. Por eso esa transacción es consistente.

El sistema garantiza la atomicidad.

Las inconsistencias temporales no se pueden mostrar.

Para asegurar la persistencia se puede hacer guardando todos los datos, o de otra forma, guardando en un registro "log" la información de las actualizaciones de las modificaciones. Con esto se puede recuperar la información si el sistema falla, al reiniciarse.

El aislamiento tiene que ver con la concurrencia.

#### 4.3.1. Estados de una transacción.

Una transacción arbitraria debe estar en alguno de los siguientes estados:

- Activa: Estado inicial. Mientras se está ejecutando.

- Abortada: Cuando se han deshecho los cambios realizados en la base de datos. Deja la base de datos en el estado inicial.

- Fallida: cuando se da cuenta que no puede continuar con la ejecución normal.

- Confirmado o comprometido: cuando se completa con éxito. Deja la base de datos en un estado consistente, no se puede deshacer.

Para deshacerlo hay que hacerlo de forma manual, con una acción compensatoria, programándola.

- Parcialmente confirmado o comprometida: Después de que se ejecuta la última operación de la transacción.

Aunque se hubiesen completado todas las operaciones, se puede abortar la transacción.

Cuando una transacción llega al estado fallido, lo normal es que aborte; una vez que se aborta, el sistema tiene dos opciones:

- 1- Reiniciar la transacción. Si el error es de hardware o software, no tiene que ver con la lógica de la transacción.
- 2- Cancelar la transacción. Cuando hay errores lógicos en la transacción.

#### 4.4 - Acceso concurrente:

Se necesitan dos transacciones,  $T_1$  definida anteriormente.

→ Transacción 2,  $T_2$ :

\* Pasar el 10% del saldo de la cuenta A a la cuenta B.

Leer (A)

$$\text{temp} = A * 0.1$$

$$A = A - \text{temp}$$

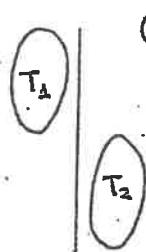
Escribir (A)

Leer (B)

$$B = B + \text{temp}$$

Escribir (B)

Se podrían ejecutar  $T_1$  y  $T_2$  de forma concurrente; aunque no lo harímos así.

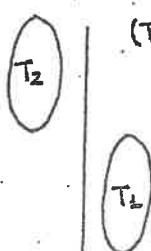


$(T_1, T_2)$

A	B
1000	2000
855	2145

→ La base de datos sigue estando en un estado consistente.

El orden en que se ejecutan las transacciones no importa.



$(T_2, T_1)$

A	B
1000	2000
850	2150

→ El estado del sistema final difiere.

Planificaciones secuenciales válidas.

En general se tendrán n transacciones.

Se denomina planificación de las n transacciones a cualquier ordenación de las instrucciones de las transacciones que consigue el orden en que aparecen dichas transacciones de forma individual.

Una planificación es válida si conserva la consistencia de la base de datos al terminar.

Una planificación es secuencial cuando mantienen agrupadas todas las operaciones de una transacción. No están mezcladas unas con otras.

Todas las planificaciones secuenciales son válidas. Empiezan en un estado consistente y acaban en un estado consistente. No hay concurrencia.

! posibles planificaciones secuenciales:

### Cambios de contexto:

$$P_1 = (T_1, T_2)$$

$$P_2 = (T_2, T_1)$$

$P_3:$	$T_1$	$T_2$													
	Leer (A)		→ Planificación válida. No secuencial.												
	$A = A - 50$														
	Escribir (A)		Equivalentes a $P_1$ .												
		Leer (A)	Deja la base de datos en un estado consistente.												
		$temp = A \neq 0'1$													
		$A = A - temp$													
		Escribir (A)													
Leer (B)			<table border="1"><thead><tr><th>A</th><th>B</th></tr></thead><tbody><tr><td>1000</td><td>2000</td></tr><tr><td>950</td><td>2000</td></tr><tr><td>855</td><td>2000</td></tr><tr><td>855</td><td>2050</td></tr><tr><td>865</td><td>2145</td></tr></tbody></table>	A	B	1000	2000	950	2000	855	2000	855	2050	865	2145
A	B														
1000	2000														
950	2000														
855	2000														
855	2050														
865	2145														
$B = B + 50$															
Escribir (B)															
	Leer (B)														
	$B = B + temp$														
	Escribir (B)														

Si el sistema operativo tiene el control de la concurrencia, produciría cambios de contexto.

#### 4.5. Secuencialidad (seriabilidad) de una planificación en cuanto a conflictos.

- Se pueden definir diferentes formas de equivalencias.
- En función de la forma que le hayamos dado tendremos diferentes formas de secuencialidad.
- Hay una equivalencia en cuanto a conflicto - - - ?

Dos instrucciones consecutivas están en conflicto si ocurre:

- Cada una de las operaciones pertenece a una transacción distinta.
- Ambas acceden al mismo elemento x.
- Alguna de las dos operaciones es escritura.

Cuando dos operaciones consecutivas están en conflicto se pueden intercambiar su orden relativo sin que eso afecte al resultado de la operación.

$P_3:$	$T_1$	$T_2$	$P_3:$	$T_1$	$T_2$
	Leer (A)			Leer (A)	
	Escribir (A)			Escribir (A)	
	Leer (B)			Leer (B)	
		Escribir (B)		Escribir (B)	
		Leer (A)	→		Leer (A)
		Escribir (A)	→		Escribir (A)
	Leer (B)		Mediante conflictos	Leer (B)	
	Escribir (B)			Escribir (B)	

Cuando intercambio ibas operaciones consecutivas de transacciones distintas en cuanto a conflicto, obtengo una transacción equivalente a la original en cuanto a conflicto.

Una planificación P es secuencial en cuanto a conflicto si es equivalente a alguna planificación secuencial.

### TEOREMA:

Cualquier planificación secuenciable en cuanto a conflicto es válida.

No todas las planificaciones en cuanto a conflicto es secuenciable.

Por ejemplo:

Ps:	T <sub>3</sub>	T <sub>4</sub>	→ Esta planificación no se puede tocar.
	Leer(A)		• No es secuencial en cuanto a conflicto.
		Escribir(A)	• <T <sub>3</sub> , T <sub>4</sub> >
	Escribir(A)		• <T <sub>4</sub> , T <sub>3</sub> >

Hay algoritmos para comprobar si una planificación es secuenciable en cuanto a conflicto.

- Bloqueo de 2 fases.
  - Esquemas multiversión (ORACLE)
  - Marcas temporales
- } Protocolos de gestión de la concurrencia.

### 4.6 Recuperabilidad.

Si una transacción falla hay que deshacerla, pero al hacerlo quizás hay que deshacer otras transacciones. Esto ocurre si una transacción depende de la otra.

Se dice que T<sub>j</sub> depende de T<sub>i</sub>, si T<sub>j</sub> lee datos que T<sub>i</sub> ha escrito.

Ejemplo:	T <sub>5</sub>	T <sub>6</sub>
	Leer(A).	
	Escribir(A)	Leer(A)
		Escribir(A)
	Leer(B)	
	Escribir(B)	

Planificación secuencial.

Si T<sub>5</sub> falla, también hay que deshacer junto con T<sub>5</sub>, T<sub>6</sub>, porque T<sub>6</sub> depende de T<sub>5</sub>.

### Def:

Una planificación P se dice que es recuperable si para cada par de transacciones,  $T_i$  y  $T_j$ , tales que  $T_j$  depende de  $T_i$ , ocurre que la operación confirmar  $T_i$  aparece antes que la de confirmar  $T_j$ .

En el ejemplo anterior, para que sea recuperable hay que confirmar  $T_5$  y luego  $T_6$ . La operación de confirmar es inherente a cada transacción.

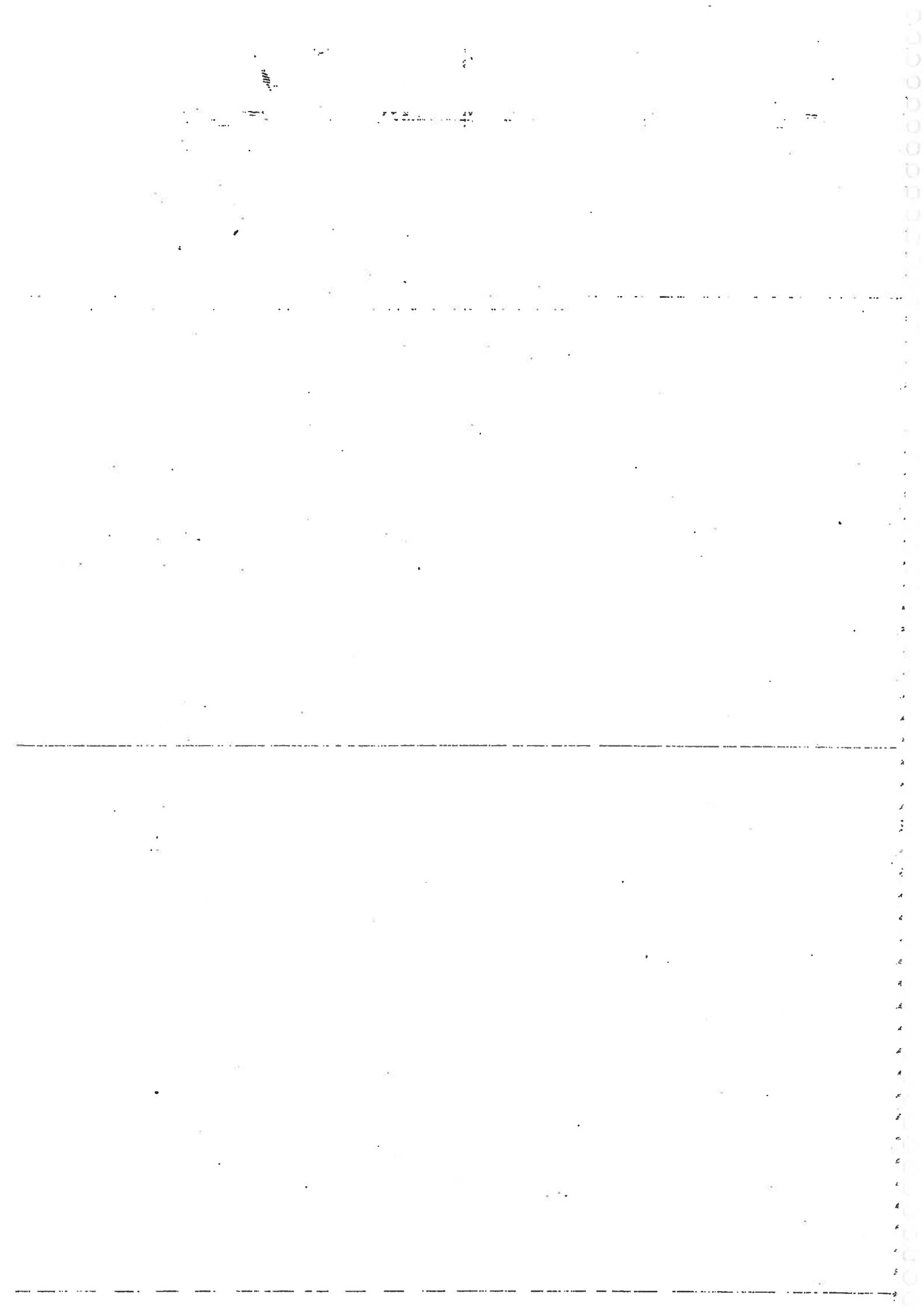
Cualquier planificación secuencial es recuperable.

Si la planificación es recuperable, el sistema jamás tiene que deshacer una transacción que ya estaba confirmada. Persistencia.

### Def: Planificación sin cascada.

Para todo par de transacciones  $T_i$  y  $T_j$  de la planificación, aunque  $T_j$  dependa de  $T_i$ , ocurre que la operación confirmar  $T_i$  aparece antes que la correspondiente operación de lectura de  $T_j$ .

Cualquier planificación sin cascada es recuperable. El recíproco no es cierto. Las planificaciones secuenciales son planificaciones sin cascada.



## TEMA 5 : DISEÑO DE BASES DE DATOS RELACIONALES

### 5.1 - Introducción :

Tendremos un esquema en la base de datos,  $B = (R_1, R_2, \dots, R_n)$   
los datos tienen asociados una serie de condiciones de integridad, condiciones lógicas que tienen que satisfacer los datos (CINT).

Objetivos del diseño de la base de datos :

- Eliminación de redundancias.
- Simplificación de verificación de las C.I.
- Recuperación de la información de manera eficaz (consultas sencillas).

El diseño de la base de datos pretende diseñar utilizando algoritmos matemáticos, para hacerlo se basa en unas condiciones de integridad.

C.I.      {  
              - Dependencias funcionales.  
              - Dependencias plurales.

- Intrrelación: Una condición de integridad sólo afecta a una tabla.

- Interrrelación: Una condición de integridad afecta a varias tablas.

Las condiciones de integridad pueden ser de estado o de transición.

- Estado: Para comprobar si se cumple la condición de integridad sólo hay que ver el estado de la tabla de la base de datos.

- Transición: Necesitamos el estado actual y el estado anterior o el estado actual y el estado siguiente.

Ejemplo: Número de convocatorias a una asignatura.

### 5.2 - Diseños equivalentes :

Tenemos un diseño  $B$  y nos preguntamos si es posible encontrar un diseño "equivalente" a  $B$ ,  $B'$ .

Un objetivo irrenunciable es que  $B'$  contenga tanta información como  $B$ .

Def:

$B'$  contiene al menos tanta información como  $B$  si existe un algoritmo en álgebra relacional que nos permite obtener una extensión cualquiera de  $B$  a partir de una de  $B'$ .

Diseño de la base de datos:

$B$ : BANCO (DNI, N, CDC, CS, S, CDS, NC, SLD);

$B'$ : CLIENTE (DNI, N, CDC)

DEPÓSITO (CS, NC, DNI, SLD)

SUCURSAL (CS, S, CDS)

BANCO = CLIENTE \* DEPÓSITO \* SUCURSAL

$B$  tiene tanta información como  $B'$ , y viceversa. Son equivalentes.

CLIENTE = P (DNI, N, CDC) (BANCO)

DEPÓSITO = P (CS, NC, DNI) (BANCO)

SUCURSAL = P (CS, S, CDS) (BANCO)

El diseño  $B'$  permite almacenar más cosas que el diseño de  $B$ .

Las tablas pequeñas son más manejables que las grandes, tienen que cumplir las condiciones de integridad de las tablas grandes.

\* Condiciones de integridad:  $B, CINT : CP = (cs, NC; DNI)$

1-  $(\forall b^1, b^2) (b^1[cs] \neq b^2[cs]) \vee ((b^1[N] = b^2[N]) \wedge (b^1[CDC] = b^2[CDC]))$

Si tengo localizada a la sucursal, sus datos no varían.

2-  $(\forall b^1, b^2) (b^1[DNI] \neq b^2[DNI]) \vee ((b^1[N] = b^2[N]) \wedge (b^1[CDC] = b^2[CDC]))$

Si tengo identificada a la persona, sus datos no varían.

3-  $(\forall b^1, b^2) ((b^1[cs] \neq b^2[cs]) \vee (b^1[NC] \neq b^2[NC]) \vee (b^1[SLD] = b^2[SLD]))$

El saldo de cada persona es único.

$B'$  verifica la simplificación de las condiciones de integridad.

$B'$  = 3 claves primarias.

Con cada diseño hay unas condiciones de integridad asociadas distintas.  
 $(B, CINT), (B', CINT)$ .

### 5.3 - Descomposición reversible por yunción.

Si tenemos una tabla R con n atributos, el proceso de descomposición va a sustituir R por m tablas. Cada una de las tablas se obtiene mediante proyecciones de R, de tal manera que la yunción natural de las m tablas tiene el mismo esquema que R.

$$R(A_1, A_2, \dots, A_n) ; R_1, R_2, \dots, R_n ; R_1 \times R_2 \times \dots \times R_n.$$

Ejemplo:  $R = (\text{DNI}, N, \text{CDC}, \text{NC}, \text{SLD})$

- Descomposición 1:

$$R_1(\text{DNI}, N, \text{CDC})$$

$$R_2(\text{NC}, \text{SLD})$$

$$R_3(\text{DNI}, \text{NC})$$

- Descomposición 2:

$$S_1(\text{DNI}, N, \text{CDC}, \text{NC})$$

$$S_2(\text{NC}, \text{SLD})$$

- Descomposición 3:

$$T_1(\text{DNI}, N, \text{CDC})$$

$$T_2(\text{DNI}, \text{NC}, \text{SLD})$$

- Descomposición 4:  $\rightarrow$  No válida.

$$V_1(\text{DNI}, N, \text{CDC}, \text{SLD})$$

$$V_2(\text{NC}, \text{SLD})$$

Las descomposiciones 1, 2 y 3 son válidas, están todos los atributos de R. Si se hace una yunción natural se obtiene R. Se pueden obtener mediante proyecciones.

En la descomposición 4, la yunción de V<sub>1</sub> y V<sub>2</sub> es distinto de R. Al hacerla aparece basura. Estamos relacionando mal los saldos.

La M1 es una descomposición reversible por yunción sin pérdida de información.

Ejemplo: Descomponemos R en dos tablas:

<u>R</u>	<u>A B C</u>	<u><math>S = P(B, C)(R)</math></u>	<u>B C</u>	<u><math>T = P(A, C)(R)</math></u>	<u>A C</u>
	a <sub>1</sub> b <sub>1</sub> c <sub>1</sub>		b <sub>1</sub> c <sub>1</sub>		a <sub>1</sub> c <sub>1</sub>
	a <sub>2</sub> b <sub>1</sub> c <sub>1</sub>		b <sub>2</sub> c <sub>1</sub>		a <sub>2</sub> c <sub>1</sub>
	a <sub>3</sub> b <sub>2</sub> c <sub>1</sub>		b <sub>3</sub> c <sub>2</sub>		a <sub>3</sub> c <sub>1</sub>
	a <sub>4</sub> b <sub>3</sub> c <sub>2</sub>				a <sub>4</sub> c <sub>2</sub>

S y T están compuestas por proyecciones de R, la unión natural de ambas tablas dan como resultado a R.

S*T.	A	B	C
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>3</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>1</sub>	b <sub>2</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>2</sub>	c <sub>1</sub>	
a <sub>3</sub>	b <sub>2</sub>	c <sub>1</sub>	
a <sub>4</sub>	b <sub>3</sub>	c <sub>2</sub>	

→ No es función reversible por unión, no hay las mismas filas que en R.

→ Tuplas espirales: No están en R: Basura.

No todas las descomposiciones son válidas.

Parece ser que R está contenido en S\*T y que S\*T no está en R.

$$R \subseteq S*T \text{ y } S*T \not\subseteq R$$

### Def.

La descomposición de R en R<sub>1</sub>, R<sub>2</sub>, ... R<sub>n</sub>, se dice que es reversible por unión si se verifica que: R = R<sub>1</sub> \* R<sub>2</sub> \* ... \* R<sub>m</sub>.

Si la descomposición es reversible por unión, el nuevo esquema formado contiene al menos tanta información como el original. Y viceversa.  
Los dos diseños son equivalentes.

$$B = \{R\} ; B' = \{R_1, \dots, R_m\}$$

Suponemos que tenemos R con U atributos: R(U)

A, B, C (Particiones de U).

$$\begin{cases} - A \cap B = A \cap C = B \cap C = \emptyset \\ - A \cup B \cup C = U \end{cases}$$

### Descomposición Binaria:

Se descompone la tabla R en dos tablas.

$$S = P(A, B) (R)$$

$$T = P(A, C) (R)$$

### Proposición:

Siempre ocurre que  $R$  está contenido en la unión natural de  $S*T$ .

$$R \subseteq S*T$$

### Demostración:

$$\langle a, b, c \rangle \in R \quad \text{d} \Rightarrow \langle a, b, c \rangle \in S*T?$$

$$\rightarrow \left\{ \begin{array}{l} \langle a, b \rangle \in S \\ \langle a, c \rangle \in T \end{array} \right\} \quad \langle a, b, c \rangle \in S*T$$

En general:  $S*T \not\subseteq R$ .

Para comprobarlo usamos un contraejemplo. Supongamos que:

$$\langle a, b, c \rangle \notin R$$

$$\left. \begin{array}{l} \langle a, b, d \rangle \in R \Rightarrow \langle a, b \rangle \in S \\ \langle a, e, c \rangle \in R \Rightarrow \langle a, c \rangle \in T \end{array} \right\} \Rightarrow \langle a, b, c \rangle \in S*T.$$

$\langle a, b, c \rangle \notin R \rightarrow$  T-upla espuria. Se ha generado y no estaba en  $R$ .

### Def: Dependencia plural o Multivaluada.

Sólo afecta a los atributos de una tabla. Se dice que  $A$  depende pluralmente de  $B$ , o que  $B$  depende pluralmente de  $A$  y se representa:

$$A \rightarrow\!\!\!> B$$

Si y sólo si para cualesquiera dos tramas de  $R$ , con el mismo valor en  $A$ , también existen en  $R$  las que resultan al intercambiar los valores de  $B$ .

R	A	B	C
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>1</sub>	b <sub>2</sub>	c <sub>2</sub>	
a <sub>2</sub>	b <sub>2</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>1</sub>	c <sub>2</sub>	

$$dA \rightarrow\!\!\!> B?$$

cuando intercambiamos los valores, no me da una fila que contenga  $R$ .

Al añadir las dos últimas filas si se verifica la dependencia plural. Contempla todas las posibilidades.

Si se verifica que  $A \rightarrow\!\!\!> B$ , también se da que  $A \rightarrow\!\!\!> C$ .

Existen una dependencia plural entre las personas que viven en una casa y el teléfono fijo, es el mismo para todos.

Si hay una dependencia plural se puede hacer una descomposición reversible por yuxtaposición.

### TEOREMA:

La descomposición de  $R$  en  $S \circ T$ , siendo  $S = P(A, B)(R)$  y  $T = P(A, C)(R)$  es reversible por yuxtaposición, si y sólo si  $A$  define pluralmente a  $B$ .

$$R = S \circ T \Leftrightarrow A \rightarrow\!\!\!> B \quad (A \rightarrow\!\!\!> C)$$

### Demostración:

" $\Leftarrow$ ". Probamos hacia la izquierda, que la condición es suficiente.

•  $d R \subseteq S \circ T ? \rightarrow$  Siempre se da.

•  $d S \circ T \subseteq R ? \rightarrow d ?$

$$\text{Sea } \langle a, b, c \rangle \in S \circ T \Rightarrow \left\{ \begin{array}{l} \langle a, b \rangle \in S \Rightarrow \exists x \langle a, b, x \rangle \in R \\ \langle a, c \rangle \in T \Rightarrow \exists y \langle a, c, y \rangle \in R \end{array} \right\} \underline{\langle a, b, c \rangle \in R}$$

Por hipótesis:

$$A \rightarrow\!\!\!> B.$$

" $\Rightarrow$ ".  $\forall \langle a, b, x \rangle, \langle a, c, y \rangle \in R \Rightarrow d \langle a, c, x \rangle, \langle a, b, y \rangle \in R ?$

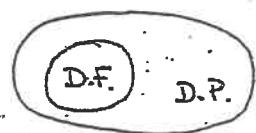
$$\begin{aligned} \text{Si: } & \left\{ \begin{array}{l} \langle a, b, x \rangle \in R = S \circ T \Rightarrow \langle a, b \rangle \in S, \langle a, x \rangle \in T \Rightarrow \\ \langle a, y, c \rangle \in R = S \circ T \Rightarrow \langle a, c \rangle \in S, \langle a, y \rangle \in T \Rightarrow \end{array} \right. \textcircled{X} \\ & \left. \begin{array}{l} \Rightarrow \langle a, c, x \rangle \in S \circ T = R \\ \Rightarrow \langle a, b, y \rangle \in S \circ T = R \end{array} \right\} \text{c.q.d.} \end{aligned}$$

Probamos que se cumple cuando intercambiamos los valores.

## Def: Dependencia funcional

Tipo particular de dependencia plural:

- Se dice que hay una dependencia funcional ( $A \rightarrow B$ ) o q $\in$  B depende funcionalmente de A, si para cualesquiera dos t-uplas de R, que tienen el mismo valor en A, los valores de B coinciden.



### Corolario:

- \* Si  $A \rightarrow B \Rightarrow A \rightarrow\rightarrow B$ . Si A define funcionalmente a B, también lo hace pluralmente.
- \* Si  $A \rightarrow B \Rightarrow R = S * T$ . Al revés no se cumple.  
La descomposición es reversible por unión si A define pluralmente a B.

## Def:

Una condición de integridad es trivial si se verifica siempre, en cualquier extensión posible en la base de datos.

### Ejemplo:

Tenemos este diseño:

$$B = (R(A, B, C), \text{CINT} = \{A \rightarrow B\})$$

$$B' = ((S = P(A, B)(R), T = P(A, C)(R)), \text{CINT} = \emptyset)$$

B' es el nuevo diseño con tablas más pequeñas. No tiene condiciones de integridad.

## 5.4 - Dependencias funcionales.

Son condiciones de integridad interrelacionales, sólo afectan a una tabla.  
 Si tienes una tabla R, se dice que hay dependencia funcional ( $A \rightarrow B$ ) si y sólo si para cualesquiera dos tuplas de R que tengan el mismo valor en A, entonces los valores t coinciden.

$$A \rightarrow B \Leftrightarrow \forall t^1, t^2 \in R, t^1[A] = t^2[A] \Rightarrow t^1[B] = t^2[B]$$

↳ consecuente.

↳ Antecedente (determinante, descriptor).

### Proposición:

Si B está contenido en A, entonces A define a B, es trivial la dependencia funcional.

$$B \subseteq A \Rightarrow A \rightarrow B$$

$(R, DF) \Rightarrow$  Conjunto de todas las dependencias funcionales que afectan a R

### Ejemplo:

$$R = \{ \begin{matrix} A & B & C \\ \textcircled{A}, \textcircled{B}, \textcircled{C}, \textcircled{D}, \textcircled{E}, \textcircled{F} \end{matrix} \}$$

$$DF = \{ \begin{matrix} A \rightarrow BC \\ E \rightarrow AF \end{matrix} \}$$

$$C = \{B, C, D\}$$

Nos dan este diseño 1, reversible por yunción:

$$R_1(A, B, C) \quad R_2(A, D, E, F) \quad DF_1 = \{A \rightarrow BC\} \quad DF_2 = \{E \rightarrow AF\}$$

Diseño 2, también reversible por yunción:

$$S_1(A, E, F) \quad S_2(B, C, D, E) \quad DF_1 = \{E \rightarrow AF\} \quad DF_2 = \cancel{\{A \rightarrow BC\}}$$

$$\cancel{\{A \rightarrow BC\}}$$

→ Pierde la dep. funcional.

Hay que programarlo.

No todas las descomposiciones reversibles por yunción son buenas.

Programar la dependencia funcional:  $(R, DF) = \{A \rightarrow B\}$

SELECT MAX(COUNT(DISTINCT A))  
 FROM R  
 GROUP BY A;

tienen que ser 1

→ Para comprobar que se da la dependencia funcional.

- Tenemos  $R(U)$ ,  $K$  superclave.

$K \Leftrightarrow$  superclave en  $R \Leftrightarrow \forall t^1, t^2 \in R$

$t^1[K] = t^2[K] \Rightarrow t^1 = t^2$ ; si  $t^1 \neq t^2 \Rightarrow t^1[K] \neq t^2[K]$

$$t^1[U.K] = t^2[U.K] \Leftrightarrow K \rightarrow U.$$

Si y sólo si  $K$  define funcionalmente a  $U$ , y  $K$  es minimal.

Ejemplo:  $R = (A, B, C, D, E, F)$   $DF = \{A \rightarrow BC; E \rightarrow AF\}$

$R1(A, B, C)$

$R2(A, D, E)$

clave: (A)

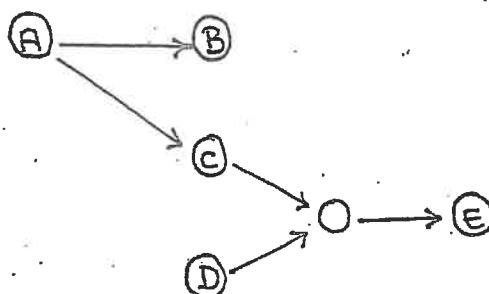
$DF1 = \{A \rightarrow BC\}$

$DF2 = \{E \rightarrow AF\}$

No hay filas con el mismo valor en A, ahora es clave. No hay que programar la dependencia funcional. Gracias a la dependencia funcional A es clave.

Def: Grafo de dependencia funcional

Sea:  $R(A, B, C, D, E, F)$ ,  $DF = \{A \rightarrow BC; CD \rightarrow E, B \rightarrow E\}$



La clave es AFD. Está formada por nodos aislados y por los que no incide ningún arco.

### 5.4.1 - Sistema de inferencia para dependencias funcionales.

Dada una tabla R y un conjunto de DF que satisfacen la tabla, una dependencia funcional es lógicamente inferible a partir del conjunto DF si se cumple en todas las extensiones posibles en las que se amplia DF.

#### \* Axiomas de Armstrong:

##### - Axioma 1 : Reflexiva.

$$\text{Si } B \subseteq A \Rightarrow A \rightarrow B$$

##### - Axioma 2 : Aumentativa.

$$\text{Si } A \rightarrow B \Rightarrow AC \rightarrow BC$$

##### - Axioma 3 : Transitiva.

$$\text{Si } A \rightarrow B \text{ y } B \rightarrow C \Rightarrow A \rightarrow C.$$

los tres axiomas constituyen un sistema de inferencia completo y consistente.

Es completo porque cualquier dependencia funcional que sea inferible a partir de DF lo puedo obtener aplicando los axiomas.

Es consistente porque cualquiera se puede obtener aplicando los axiomas es inferible a partir de DF.

A partir de los axiomas obtenemos una serie de propiedades.

#### - Propiedad 1 : Descomposición:

$$\boxed{\text{Si } X \rightarrow Y, Z \subseteq Y \Rightarrow X \rightarrow Z}$$

Demostración:  $X \rightarrow Y$

$$Z \subseteq Y \Rightarrow Y \rightarrow Z \quad \left. \begin{array}{c} \\ \end{array} \right\} \Rightarrow X \rightarrow Z$$

↑                      ↓  
Axioma 1          Axioma 3.

$$\begin{array}{l} A \rightarrow B \in E \\ \Downarrow \\ A \rightarrow B \\ A \rightarrow E \\ A \rightarrow C \end{array}$$

#### - Propiedad 2 : Unión: (Lo contrario a descomposición).

$$\boxed{\text{Si } X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ}$$

Demostración:

$$\begin{array}{c} X \rightarrow Y \Rightarrow X \rightarrow YX \\ \downarrow \\ X \rightarrow Z \Rightarrow YX \rightarrow ZY \end{array} \quad \left. \begin{array}{c} \\ \end{array} \right\} \Rightarrow X \rightarrow YZ$$

- Propiedad 3: Pseudotransitiva:

$$\boxed{\text{Si } X \rightarrow Y, WY \rightarrow Z \Rightarrow WX \rightarrow Z}$$

Demostración:

$$\left. \begin{array}{l} X \rightarrow Y \Rightarrow WX \rightarrow WY \\ \downarrow WY \rightarrow Z \end{array} \right\} \Rightarrow WX \rightarrow Z$$

↓  
Axioma 2.

↓  
Axioma 3.

Ejemplo:  $R(A, B, C, D, E, F)$ ,  $DF = \{ A \rightarrow BC, CD \rightarrow E, B \rightarrow E \}$

$A \rightarrow E$ , es lógicamente inferible a partir de  $DF$ .

Comprobación:

$$\left. \begin{array}{l} A \rightarrow BC \Rightarrow A \rightarrow B \\ A \rightarrow C \\ B \rightarrow E \end{array} \right\} \Rightarrow A \rightarrow E$$

↓  
Transitiva.

### 5.4.2 - Tipos de dependencias funcionales.

⇒ Dependencia funcional plena total o completa:

Se dice que  $Y$  tiene una dependencia funcional completa respecto a  $X$ , si  $X$  define funcionalmente a  $Y$  y ningún subconjunto propio de  $X$  depende funcionalmente a  $Y$ .

$$X \rightarrow Y \wedge \forall z \subset X, z \not\rightarrow Y$$

⇒ Dependencia funcional parcial:

Se dice que  $Y$  tiene una dependencia funcional parcial respecto a  $X$  si  $X$  define funcionalmente a algún subconjunto propio de  $Y$ .

$$X \rightarrow Y \wedge \exists z \subset Y, z \rightarrow Y$$

⇒ Dependencia funcional elemental:

Una dependencia funcional es elemental si es completa no trivial y el consecuente está formado por un único atributo.

Dos conjuntos de atributos son descriptores equivalentes si  $X \rightarrow Y$  e  $Y \rightarrow X \Rightarrow \boxed{X \leftrightarrow Y}$

### → Dependencia funcional transitiva:

Se dice que  $z$  tiene una dependencia funcional transitiva respecto a  $x$  a través de  $y$ , si  $x$  define a  $y$ ,  $y$  define a  $z$  e  $y$  no define a  $x$ .

$$x \rightarrow y; y \rightarrow z; \quad \underline{y \not\rightarrow x}.$$

se tiene que cumplir para ser transitiva.

Si además  $z$  no define a  $y$ , se dice que la dependencia es transitiva estricta.

### 5.4.3 - Cierre de un conjunto de dependencias funcionales.

Se llama cierre o clausura de  $DF$  y se denota por  $DF^+$ , al conjunto de todas las dependencias funcionales que se pueden inferir a partir de  $DF$ .  
Dos conjuntos de dependencias funcionales  $DF_1$  y  $DF_2$  son equivalentes si su cierre coincide. Por eso la importancia del cierre.

Se puede calcular el cierre de varias formas:

- Usando la definición de la dependencia funcional. (No se usa nunca)
- Usando los axiomas de Armstrong. Lo obtendríamos de forma algebraica. (Prácticamente no se usa).
- De forma gráfica, usando el diagrama de dependencias funcionales.

No se usa salvo que el diagrama sea pequeño.

Por tanto, calcular de forma explícita el conjunto de dependencias funcionales es un trabajo que consume muchos recursos.

- Algoritmo:

Tiene un conjunto de dependencias funcionales.

Cierre-  $DF$

$$DF^+ = DF;$$

Repeat

for each  $df \in DF^+$

Aplicar a  $df$  reflexividad y aumentatividad y añadir a  $DF^+$

for each  $df_1$  y  $df_2 \in DF^+$

Aplicar transitividad y añadir a  $DF^+$ ;

until ( $DF^+$  no cambie);

→ Comprobar la equivalencia:  $\left. \begin{array}{l} (R, DF_1) \\ (R, DF_2) \end{array} \right\} \quad \text{d}DF_1^+ = DF_2^+ ?$

$$\left. \begin{array}{l} DF_1 \subseteq DF_2^+ \\ DF_2 \subseteq DF_1^+ \end{array} \right\} \Rightarrow DF_1^+ = DF_2^+$$

Calcular cierres por separado y compararlos (inviable).

Comprobar que las dependencias de  $DF_1$  están en  $DF_2^+$ , y al contrario, si se cumplen, los cierres coinciden.

Ejercicio: Consideramos dos conjuntos de DF.

$$DF_1 = \{ AB \rightarrow CDEF, A \rightarrow D, B \rightarrow E, E \rightarrow F \}$$

$$DF_2 = \{ AB \rightarrow C, A \rightarrow D, B \rightarrow E, E \rightarrow F \}$$

Suponemos que son equivalentes.

$DF_2$  es más simple, sobrecarga menos al sistema.

Comprobamos que sean equivalentes:

$$DF_1 \subseteq DF_2^+ \quad \checkmark$$

$$DF_2 \subseteq DF_1^+ \quad \checkmark \quad \text{Se cumple que son equivalentes.}$$

$dAB \rightarrow C \subseteq DF_1^+ ? \rightarrow$  Sí, porque en  $DF_1$  está esa dependencia.

$$(AB \rightarrow CDEF \Rightarrow AB \rightarrow C)$$

↓  
Desc.

$dAB \rightarrow CDEF \subseteq DF_2^+ ? \rightarrow AB \rightarrow C \subseteq DF_2^+ \quad (\text{Hipótesis})$ .

$$AB \rightarrow E \subseteq DF_2^+ \quad (\text{Desc.})$$

$$AB \rightarrow F \subseteq DF_2^+ \quad (\text{Transitividad})$$

$$AB \rightarrow D \subseteq DF_2^+$$

$$\left. \begin{array}{l} AB \rightarrow CDEF \\ \subseteq DF_2^+ \end{array} \right\} \quad \begin{array}{l} AB \rightarrow C \\ \subseteq DF_2^+ \end{array}$$

(Unión).

#### 5.4.4. Cierre de un conjunto de atributos.

Suponemos que tenemos  $(R, DF)$  con unos atributos  $A \subseteq U$ . Se llama cierre de un conjunto de atributos  $A$  bajo el conjunto de dependencias funcionales  $DF$ , y se denota como  $A^+$  al conjunto de todos los atributos que dependen funcionalmente de  $A$ .

- Ejemplo:  $R(A, B, C, D, E, F)$

$$DF = \{ A \rightarrow BC, CD \rightarrow EF, B \rightarrow E \}$$

Calcular el cierre de  $AD^+$ :

$$AD^+ = \{ A, D, B, C, E, F \}$$

$$\left\{ \begin{array}{l} AD \rightarrow AD \text{ (Reflexiva).} \\ AD \rightarrow C \\ AD \rightarrow D \end{array} \right\} \Rightarrow AD \rightarrow CD$$
$$CD \rightarrow EF \Rightarrow AD \rightarrow EF$$

Se han conseguido en el cierre todos los atributos. No tenía porque serlo.

$$AD \rightarrow ADBCEF \Rightarrow AD \rightarrow U \Rightarrow AD \text{ es superclave.}$$

Define a todo el conjunto de atributos.

Calcular el cierre también sirve para comprobar si un conjunto de atributos es superclave.

#### \* Algoritmo de Ullman:

Algoritmo para calcular el cierre de un conjunto de atributos.

Recibe como entrada el conjunto de atributos  $A$  y las dependencias funcionales  $DF$ .

Inicialmente, por reflexiva, el cierre es:  $A \Rightarrow A^+ = A$ .

$$A^+ = A;$$

while ( $A^+$  cambie)

for each df  $X \rightarrow Y \in DF$  // Comprobar las dep. funcionales.

$$\text{if } X \subseteq A^+ \Rightarrow A^+ = A^+ \cup Y$$

sí el antecedente  
está en el cierre.

Añade el consecuente.

#### → Usos del Algoritmo de Ullman:

1º - Comprobar si un conjunto de atributos es superclave.

Si  $A$  es superclave?

Calcular  $A^+$ .

if  $A^+ = U \Rightarrow A$  es superclave.

2º Calcular  $DF^+$ .

$$c\ DF^+? \quad 2^{U^1} - \emptyset.$$

$\forall X \subseteq U \Rightarrow X^+; //$  Para cualquier subconjunto de  $U$ , calcula el cierre de  $X$ .

$$\forall Y \subseteq X^+ \Rightarrow X \rightarrow Y \in DF^+;$$

3º Comprobar si una DF pertenece al cierre de DF.

$$c\ X \rightarrow Y \in DF^+?$$

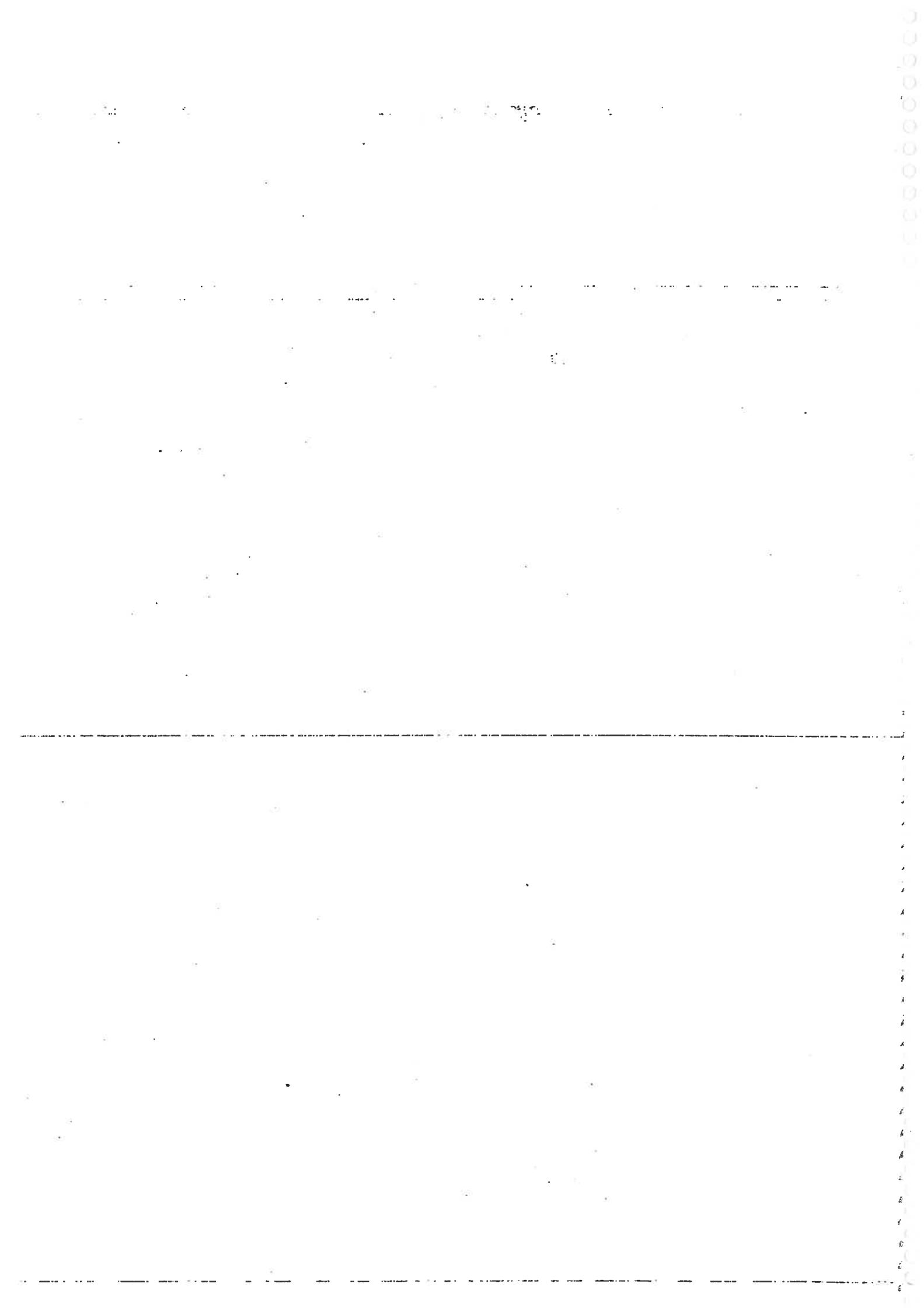
- Calcular  $X^+$ .

$$- \text{if } Y \subseteq X^+ \Rightarrow X \rightarrow Y \in DF^+.$$

4º Comprobar si 2 DF son equivalentes. (Usando el 3º recurso).

NOTACIÓN:  $X_{DF1}^+ \Rightarrow$  Cierre de  $X$  usando DF1.

$X_{DF2}^+ \Rightarrow$  Cierre de  $X$  usando DF2.



# Tema 5: Diseño de Bases de Datos

## 5.1. Introducción

Para diseñar bien una base de datos usamos Teoría de la Normalización, que genera diseños que están en alguna forma normal de las BD. Hay múltiples formas normales; cuanto más alto más difícil llegar a ello.

Objetivos de la Teoría de Normalización:

- Por un lado, intentan evitar (o minimizar) la redundancia, porque la repetición de los datos
- Quieren facilitar las consultas
- Minimizar la verificación de las condiciones de integridad que tiene que satisfacer nuestra base de datos

## 5.2. Diseños equivalentes

$B'$  se dice que contiene al menos la misma cantidad de información que  $B$  si existe un algoritmo en álgebra relacional que nos permita obtener una extensión cualquiera de  $B$ , a partir de una de  $B'$ .

Ejemplo:

Esquema B:

BANCO(DNI,NBC,CDC,CS,S,CDS,NC,SLD) → Tabla universal, ya que el diseño tiene una única tabla

Condiciones de integridad:

$$CP = (CS, NC, DNI)$$

$(\forall b_1, b_2) ((b_1[DNI] = / = b_2[DNI]) \vee (b_1[N] = b_2[N] \wedge \dots))$  // para un mismo DNI coinciden el nombre, ciudad, etc.

// si una cuenta tiene varios titulares → sus saldos coinciden

Esquema  $B'$ :

CLIENTE(DNI,NBC,CDC)

DEPOSITO(CS,NC,DNI,SLD)

SUCURSAL(CS,S,CDS)

Analizamos:

$B'$  contiene al menos la misma cantidad de información que  $B$ , porque BANCO no es más que la unión natural CLIENTE\*DEPOSITO\*SUCURSAL.

$$\text{CLIENTE} = P(\text{DNI}, \text{NBC}, \text{CDC})(\text{BANCO})$$

$$\text{DEPOSITO} = P(\text{CS}, \text{NC}, \text{DNI}, \text{SLD})(\text{BANCO})$$

$$\text{SUCURSAL} = P(\text{CS}, \text{S}, \text{CDS})(\text{BANCO})$$

Dos esquemas (diseños) son equivalentes, en cuanto a la cantidad de información, si...

Condiciones de integridad de claves primarias:

CLIENTE(DNI,NBC,CDC)

DEPOSITO(CS,NC,DNI,SLD)

SUCURSAL(CS,S,CDS)

BANCO(DNI,NBC,CDC,CS,S,CDS,NC,SLD)

El esquema B es mucho más redundante que B', con todo lo que conlleva (tablas más grandes, mucha información repetida,...).

Además, el esquema de B' permite hacer cosas que B no deja, como tener sucursales dadas de alta sin ninguna cuenta, o clientes sin cuentas.

Dos tipos de condiciones de integridad: intrarelación o interrelación, de estado (si para comprobar que se verifica sólo hay que mirar el estado actual de BD) o de transición (si hay que comprobar algo anterior o posterior de la modificación).

De estado e intrarelación:

[...]

### 5.3. Descomposición reversible por yunción

Se llama descomposición al proceso que permite pasar de una tabla R a un conjunto de tablas R1, R2, ...Rn, que se han obtenido a partir de proyecciones de R; de tal manera que el esquema de la yunción natural de R1\*R2\*...\*Rn coincide con el de R.

Ejemplo:

R(DNI,NBC,CDC,NC,SLD)

- Descomposición 1:

R1(DNI,NBC,CDC)

R2(NC,SLD)

R3(DNI,NC)

- Descomposición 2:

S1(DNI,NBC,CDC,NC)

S2(NC,SLD)

- Descomposición 3:

T1(DNI;NBC;CDC)

T2(DNI,NC,SLD)

- Descomposición 4:

V1(DNI,NBC,CDC,SLD)

V2(NC,SLD)

➔ No es una descomposición válida, no es adecuada porque al hacer la yunción natural se pueden obtener filas que son basura, y se llaman t-uplas espuria.

R	A	B	C
a1	b1	c1	
a2	b1	c1	
a3	b2	c1	
a4	b3	c2	

S	=	B	C
P(B,C)(R)		b1	c1
		b2	c1
		b3	c2

T	=	A	C
P(A,C)(R)		a1	c1
		a2	c1
		a3	c1
		a4	c2

Las t-uplas espurias son todas aquellas t-uplas de la yunción natural de las

Se dice que la descomposición de R en R1,R2,...Rn es reversible por yunción (o sin pérdida de información), cuando  $R = R_1 * R_2 * \dots * R_n$ , y si eso ocurre, el nuevo diseño tiene al menos tanta información como R, es decir, son equivalentes.

Ejemplo:

Vamos a fijarnos en un caso: realizar una descomposición binaria.

$R(U); S = P(A,B)(R); T = P(A,C)(R)$

A, B, C forman una partición de U.

A (intersección) B = B (intersección) C = A (intersección) C = (vacío)

$A \cup B \cup C = U$

¿Cuándo ocurre que la partición es reversible por yunción ( $R = S*T$ )?

Prop:  $R$  (subconjunto)  $S*T$

Dem: Sea  $\langle a, b, c \rangle$  (pertenece)  $R \Rightarrow \langle a, b \rangle$  (pert.)  $S; \langle a, c \rangle$  (pert.)  $T \Rightarrow \langle a, b, c \rangle$  (pert.)  $S*T$

En general,  $S*T$  no está contenido en R.

Contraejemplo:

$\langle a, b, x \rangle$  (pertenece)  $R \Rightarrow \langle a, b \rangle$  (pert.)  $S$

$\langle a, y, c \rangle$  (pertenece)  $R \Rightarrow \langle a, c \rangle$  (pert.)  $T \quad \langle a, b, c \rangle$  (pertenece)  $S*T$

#### Dependencia plural (multivaluado)

Se dice que hay una dependencia plural entre A y B ( $A \rightarrow\rightarrow B$  ("A define pluralmente a B" o "B depende pluralmente de A")), si para cualesquiera dos t-uplas de R con el mismo valor en A, las t-uplas que resultan de intercambiar los valores de B también están en R.

Ejemplo:

R	A	B	C
	a1	b1	c1
	a1	b2	c2
	a1	b2	c1
	a1	b1	c2

Cuando hay una dependencia natural, se puede enunciar el teorema:

$R = S*T$  (si y solo si)  $A \rightarrow\rightarrow B$  ( $A \rightarrow\rightarrow C$ )

(Si hay dependencia plural, la descomposición es reversible por yunción)

Ejemplo:

$\{R = S^*T\}$

$\{R \text{ (subconjunto) } S^*T\} \rightarrow \text{Se da, demostrado antes.}$

$\{S^*T \text{ (subconj.) } R\}$

Sea  $\langle a, b, c \rangle$  (pert.)  $S^*T$

$\langle a, b \rangle$  (pert.)  $S \rightarrow \exists x, \langle a, b, x \rangle$  (pert.)  $R \rightarrow A \rightarrow \rightarrow B$

$\langle a, c \rangle$  (pert.)  $T \rightarrow \exists y, \langle a, y, c \rangle$  (pert.)  $R \rightarrow \langle a, b, c \rangle$  (pert.)  $R$

"(implica)"

(para todo)  $\langle a, b, c \rangle, \langle a, x, y \rangle$  (pertenece)  $R$

$\rightarrow \langle a, x, c \rangle, \langle a, b, y \rangle$  (pertenece)  $R$

$\rightarrow \langle a, b \rangle$  (pert.)  $S, \langle a, c \rangle$  (pert.)  $T; \langle a, x \rangle$  (pert.)  $S, \langle a, y \rangle$  (pert.)  $T$

**Dependencia funcional (una sola flecha):**

$A \rightarrow B$ , si y solo si, para cualesquiera dos t-uplas de  $R$ , con el mismo valor en  $A$ , los valores de  $B$  coinciden.

Proposición:

Si  $A \rightarrow B$ , entonces  $A \rightarrow \rightarrow B$ .

Corolario:

$A \rightarrow B$ , entonces  $R = S^*T$ . (Lo contrario no es cierto, en general)

Una condición de integridad, en general, es trivial si se da siempre, en cualquier extensión de la base de datos.

Ejemplo:

$(R(A, B, C), C.\text{Int.} = \{A \rightarrow \rightarrow B\})$

$(S = P(A, B)(R), T = P(A, C)(R), C.\text{Int.} = \{A \rightarrow \rightarrow B\}) \rightarrow$  Ahora esta dependencia plural es trivial, no hace falta

Por lo tanto:  $(S = P(A, B)(R), T = P(A, C)(R), C.\text{Int.} = \{\}) \rightarrow$  Es un diseño mejor, porque no hay condiciones de integridad, ...

## 5.4. Dependencia funcional

$R(U)$

Def:

$A \rightarrow B$  (si y solo si) (para todo)  $t_1, t_2$  (pert.)  $R, t_1^1[A] = t_2^1[A] \rightarrow t_1^1[B] = t_2^1[B]$

Si teniendo el mismo valor en  $A$ , los valores de  $B$  coinciden.

Si el consecuente ( $B$ ) está contenido en el antecedente ( $A$ ), entonces la dependencia funcional  $A \rightarrow B$  es trivial.

SELECT MAX(COUNT(DISTINCT B))  $\rightarrow$  El MAX y el COUNT tienen que ser 1

FROM R

GROUP BY A;

```
CREATE ASSERTION ***
CHECK NOT EXISTS
    (SELECT A
     FROM R
     GROUP BY A
     HAVING COUNT(DISTINCT B)>1);
```

Ejemplo:

R(A,B,C,D,E,F)

DF = {A→BC, E→AF}

Descomposición 1:

R1(A,B,C), DF1={A→BC}

R2(A,D,E,F), DF2={E→AF}

Descomposición 2:

S1(A,E,F) , DF1={E→AF}

S2(B,C,D,E), DF2={}.

Las dos descomposiciones son reversibles por yunción. Nuestra A de antes es A, la B es B, C, y la C es D,E,F.

Se pueden obtener múltiples descomposiciones reversibles por yunción a partir de una tabla.

En la descomposición 2, hay que hacer una condición de integridad para comprobar la segunda dependencia funcional de R, y para ello hay que hacer una yunción de S1 y S2, porque están en tablas diferentes.

R(U)

K (subconjunto) U

[...]

Grafo de un conjunto de DF

Ejemplo:

R(A,B,C,D,E,F)

DF = {A→BC, CD→D, B→E}

Clave(A,D,F)

#### 5.4.1. Sistemas de inferencia para dependencias funcionales

(R, DF)

Se dice que una dependencia funcional es inferible a partir de DF del conjunto, o bien está lógicamente implicada por DF, si se cumple en todas las extensiones posibles de R en las que se verifica DF.

Axiomas de Armstrong.

1. Reflexiva

Si Y está contenida en X, entonces X depende funcionalmente de Y.

2. Aumentativa

Si aumentas el antecedente de la DF, tienes que aumentar el consecuente para que se mantenga.

### 3. Transitiva

[...]

Los axiomas de Armstrong son un sistema consistente y completo para crear DFs.  
(pedir a Daura)

#### Propiedades

P1: Unión

[Pedir]

P2: Descomposición

Si  $X \rightarrow Y, Z \subseteq Y \rightarrow X \rightarrow Z$

Demostración:

$X \rightarrow Y \rightarrow$

Si  $Z \subseteq Y \rightarrow Y \rightarrow Z$

P3: Pseudotransitiva

[...]

#### **5.4.2. Tipos de Dependencias Funcionales**

Dependencia funcional total, plena, completa: se dice que  $Y$  tiene una dependencia funcional total, plena o completa respecto a  $X$  si  $X$  define funcionalmente a  $Y$  ( $X \rightarrow Y$ ) y además, no se le puede quitar ningún atributo al antecedente ( $X$ ) sin que se pierda la dependencia funcional.

Dependencia funcional parcial: una dependencia funcional es parcial, si no es completa.

$X \rightarrow Y, (\exists Z) Z \subseteq X, Z \rightarrow Y$

Dependencia funcional elemental (es irreducible, no se puede simplificar más): si es completa (o plena o total), no trivial y el consecuente es un atributo único.

Dos descriptores  $X$  e  $Y$  son equivalentes si  $X$  define funcionalmente a  $Y$  e  $Y$  define funcionalmente a  $X$  ( $X \leftrightarrow Y$ )

Dependencia funcional transitiva: se dice que  $Z$  tiene una dependencia funcional transitiva respecto a  $X$  a través de un conjunto de atributos  $Y$  si se verifica pues lo siguiente, que  $X$  define a  $Y$ , que  $Y$  define a  $Z$ ; y además  $Y$  no define a  $X$ , porque si  $Y$  definiese a  $X$  tendrían la misma potencia (es otro caso).

Cuando además  $Z$  no define a  $X$ , se dice que la dependencia funcional transitiva es estricta.

#### **5.4.3. Cierre de un conjunto de Dependencias Funcionales**

Se llama cierre o clausura de DF (y se denota por  $DF^+$ ) al conjunto de todas las dependencias funcionales que son lógicamente inferibles a partir de DF. Ese conjunto puede tener un número extraordinariamente grande de elementos, de tal forma que en la práctica nunca se calcula de forma explícita el cierre.

( $R(U)$ ,  $DF_1$ ) (Son  
( $R(U)$ ,  $DF_2$ ) equivalentes)

Si  $DF_1$  y  $DF_2$  son equivalentes, deberemos quedarnos con el más fácilmente programable.

→ Dos conjuntos de dependencias funcionales serán equivalentes si tienen el mismo cierre ( $DF_1^+ = DF_2^+$ ).

Proposición que arregla este problema

$DF_1$  y  $DF_2$  son equivalentes, si y solo si se verifican estas dos condiciones:

$$\begin{aligned} DF_1 &\subseteq DF_2^+ \\ DF_2 &\subseteq DF_1^+ \end{aligned}$$

Demostración:

Por hipótesis:

[...]

#### 5.4.4. Cierre de un conjunto de atributos

Se llama cierre o clausura de un conjunto de atributos A bajo el conjunto de DF de R, y se denota por  $A^+$ , al conjunto de todos los atributos que dependen funcionalmente de A, es decir, es el conjunto de todos los X tales que A define funcionalmente a X ( $A \rightarrow X \in DF^+$ ).

Ejemplo:

$$\begin{aligned} R(A, B, C, D, E, F) \\ DF = \{A \rightarrow BC, CD \rightarrow EF, B \rightarrow E\} \\ AD^+ = \{A, D, B, C, E, F\} \\ AD \rightarrow U, \text{ lo que significa que } AD \text{ es superclave.} \end{aligned}$$

#### Algoritmo de Ullman (algoritmo para calcular el cierre de un conjunto de atributos)

```
Ullman(DF, A)
    A+ = A;
    Do
        For each X → Y ∈ DF
            If X ⊂ A+ then
                A+ = A+ ∪ Y;
    While (A+ cambie);
```

#### Aplicaciones del cierre de un conjunto de atributos:

Permite saber si un conjunto de atributos es superclave o no:

Calcular  $A^+$ ;

If  $A^+ = U \rightarrow A$  es superclave

Permite calcular el cierre de un conjunto de dependencias funcionales (da una forma alternativa de calcular  $DF^+$ )

$\forall X \subseteq U \quad X^+ \text{ (calcular el cierre)}$

$\forall Y \subseteq X^+ \quad X \rightarrow Y \in DF^+$

Permite comprobar si una DF ( $A \rightarrow B$ ) pertenece al cierre  $DF^+$

Calcular  $A^+$ ;

If  $B \subseteq A^+ \rightarrow A \rightarrow B \in DF^+$ ;

#### 5.4.5. Recubrimiento minimal (canónico) de un conjunto de dependencias funcionales

Atributo raro (o extraño, redundante) en una dependencia funcional:

(R, DF)  $X \rightarrow Y \in DF \quad Z$  es raro...

1) En el antecedente si  $Z \subseteq X \wedge DF' = DF - \{X \rightarrow Y\} \cup \{(X-Z) \rightarrow Y\}$

$DF'^+ = DF^+ \rightarrow DF \subseteq DF'^+ \quad DF' \subseteq DF^+$

2) En el consecuente si  $Z \subseteq Y \wedge DF' = DF - \{X \rightarrow Y\} \cup \{X \rightarrow (Y-Z)\}$

$DF'^+ = DF^+ \rightarrow DF \subseteq DF'^+ \quad DF' \subseteq DF^+$

El recubrimiento canónico, redundante o minimal de DF lo denotaremos por DFC y es un conjunto de dependencias funcionales que verifican las siguientes condiciones:

1. El cierre de DFC es igual que el cierre de DF ( $DFC^+ = DF^+$ )
2. Ninguna dependencia funcional de DFC contiene atributos raros
3. El antecedente de todas las dependencias funcionales de DFC+ es único, es decir, ninguna dependencia funcional comparte el antecedente (aplicar unión todo lo que puedas)

Algoritmo para calcular el conjunto minimal de un conjunto de dependencias funcionales

Recubrimiento\_minimal(DF) {

$DFC = DF;$

    Do

        Aplicar unión en las dependencias f. de DFC

        Eliminar atributos raros

    While  $DFC$  cambie;

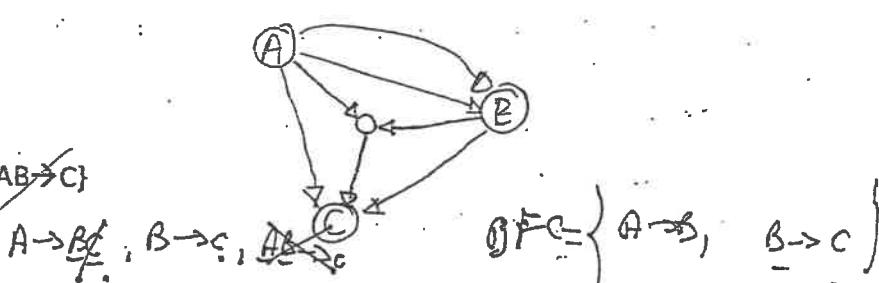
}

Ejemplo:

$R(A, B, C)$

~~$DF = \{A \rightarrow BC, B \rightarrow C, A \not\rightarrow B, AB \rightarrow C\}$~~

$DFC = \{A \rightarrow B, B \rightarrow C\}$



#### 5.4.6. Propagación de las dependencias funcionales al descomponer

[...]

test\_preservacion\_de\_dep\_func.

```

    For each  $X \rightarrow Y \in DF$ 
        If  $X \rightarrow Y \notin DF'$ , then return FALSE;
    // If  $Y \subset X + DF$  then return FALSE;
    Return TRUE;

```

#### 5.4.7: Anomalías de actualización asociadas a dependencias funcionales (R, DF)

$X \rightarrow Y$  no trivial,  $\in DF$       X no es superclave

Anomalías de modificación:

X'	Y
X	Y
X'	Y

Anomalías de borrado:

[...]

Anomalías de inserción:

[...]

Ninguna de esas anomalías ocurre si X es superclave. Si X es supérclave, las anomalías asociadas a esa dependencia funcional no ocurren.

Me interesa descomponer R en tablas más pequeñas si:

- i) Reversible por yunción
- ii) Se preservan las dependencias funcionales
- iii) X no es superclave

#### 5.5. Forma normal de Boyce-Codd (FNBC)

Una tabla R se dice que está en FNBC respecto a un conjunto de dependencias funcionales DF si y solo si para cualquier dependencia funcional ( $X \rightarrow Y$ ) no trivial de DF se verifica que el antecedente (X) es superclave.

Una base de datos (esquema de BBDD) está en FNBC si y solo si todas las tablas de ese esquema lo están.

Ejemplo:

R(A,B,C,D,E)

DF = { $A \rightarrow B$ ,  $BC \rightarrow D$ }

Comprobar si está en FNBC.

Representamos el grafo.

La única clave es  $(A,C,E)$ . Para este ejemplo, ninguna de las dos es el antecedente es superclave; por lo que no está en FNBC.

Nuevo diseño:

$R1(A,B) \quad DF1 = \{A \rightarrow B\} \quad$  Claves:  $(A)$  → Está en FNBC

$R2(A,C,D,E) \quad DF2 = \{AC \rightarrow D\} \quad$  Claves:  $(A,C,E)$  → No está en FNBC

Se ha perdido una dependencia funcional.

Se puede seguir descomponiendo recursivamente hasta llegar a la FNBC, pero no se puede garantizar que no se pierdan dependencias funcionales.

Proposición:

Siempre es posible descomponer una tabla  $R$  que no esté en FNBC en proyecciones que sean reversibles por yunción que sí estén en FNBC.

→ Algoritmo de descomposición en FNBC:

Resultado =  $R$ ;

While ( $V(U) \in \text{Resultado}$  no sea FNBC) do {

    Sea  $X \rightarrow Y \in DF$ , no trivial, con  $X \subseteq U$ , no superclave

    Resultado = (Resultado -  $V$ )  $\cup S(X,Y) \cup T(X,U-XY)$

}

Ejemplo: Comprobar si  $R$  está en FNBC, y si no lo está, descomponerla en FNBC.

$R(A,B,C)$

$DF = \{C \rightarrow A, AB \rightarrow C\}$

Dibujar el diagrama.

Claves:  $(A,B), (B,C)$

→ No está en FNBC, porque  $C \rightarrow A \in DF$ , y  $C$  no es superclave.

Vamos a descomponer

Algoritmo:

Resultado =  $S(A,C) \cup T(B,C)$

$S(A,C) \quad DFS = \{C \rightarrow A\} \quad$  Claves:  $(C)$  → Está en FNBC

$T(B,C) \quad DFT = \{\text{vacío}\} \quad$  Claves: → Está en FNBC

El nuevo diseño está en FNBC, pero se pierden dependencias funcionales  $(AB \rightarrow C)$ , que hay que implementar (en este caso, se define un aserto sobre la yunción natural de las dos tablas, o definir  $C$  como clave primaria).

Para que no sean tan restrictivas, aparecen las demás formas normales.

## 5.6. Tercera forma normal (3FN o FN3)

Si la descomposición no verifica que se preservan las dependencias funcionales, tienes dos opciones: o asumes el coste de implementarlas, o conformarte con menos.

Atributo principal (prime attribute): cualquier atributo que pertenezca a alguna clave; y el resto de atributos son no principales (los que no están en ninguna clave).

R está en 3FN respecto a un DF si y solo se verifica que para cualquier dependencia funcional ( $X \rightarrow Y$ ) que pertenezca a DF y que sea no trivial se verifica, o que el antecedente (X) es superclave, o que Y es un atributo principal.

Un esquema de bases de datos está en 3FN si todas las tablas del esquema están en 3FN.

Si R está en FNBC  $\Rightarrow$  R está en 3FN.

Ejemplo: Una tabla que está en 3FN pero no en FNBC.

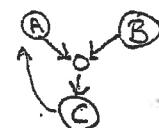
$R(A,B,C)$

$DF = \{C \rightarrow A, AB \rightarrow C\}$

Claves: (AB), (BC).

$C^+ = \{C, AC\} \rightarrow \text{No FNBC}$

$AB^+ = \{A, B, C\}$



R está en 3FN respecto a DF si ningún atributo no principal depende parcial o transitivamente respecto a alguna clave de R.

Algoritmo de síntesis de 3FN: se prueba que no se pierden dependencias funcionales, y  
Algoritmo( $R(U)$ , DFC) {

    Resultado = vacío;

    For each  $X \rightarrow Y$  perteneciente a DFC do

        If ( $XY \subseteq S$ , para todo  $S$  perteneciente a Resultado) then

            Resultado = Resultado U  $S(X,Y)$

        If (para todo  $S$  perteneciente a R, S no contiene ninguna clave) then

            Resultado = Resultado U  $S(K)$ ;     — K clave arbitraria de R

}

Ejemplo: Comprobar si está en alguna forma normal, y si no lo está normalizar a 3FN.

$R(A,B,C,D)$   
 $DF = \{C \rightarrow AD, AB \rightarrow C\}$   
 Claves: (A,B), (B,C)

No está en FNBC, porque  $C \rightarrow AD$  verifica que C no es superclave en la tabla.

No está en 3FN, porque  $C \rightarrow AD$  verifica que no es superclave y D no es principal porque no está en ninguna clave.

$DFC = DF$

$R1(A,C,D)$

$DFR1 = \{C \rightarrow AD\} \rightarrow$  está en 3FN y FNBC.

Claves: (C)

$R2(A,B,C)$

$DFR2 = \{AB \rightarrow C\} \rightarrow$  también está en 3FN y FNBC.

Claves: (A,B)

Vamos a hacerlo aplicando el algoritmo de descomposición de FNBC:

$S1(A,C,D)$

$DFS1 = \{C \rightarrow AD\}$



Claves: (C)

S2(B,C)

DFS2 = {vacío}

Claves: (B,C)

Ejemplo:

R(A,B,C)

DF = {B → C}

Claves: (A,B)

No está en 3FN, porque C no es atributo principal (no está en ninguna clave), y por lo tanto tampoco en FNBC.

DFC = DF

R1(B,C)

DF1 = {B → C}

Clave: (B)

— Aquí se acaba el for each, y entra en el último if.

R2(A,B)

DF2 = {vacío}

Claves: (A,B)

Ejemplo:

R(A,B,C,D,E,F,G)

DF = {BC → D, E → FG}

Claves: (A,B,C,E)

Pasar a 3FN:

R1(B,C,D)

DF1 = {BC → D}

Claves: (B,C)

R2(E,F,G)

DF2 = {E → FG}

Claves: (E)

R3(A,B,C,E)

DF3 = {}

Claves: (A,B,C,D)

Ejemplo: con la BD de prácticas.

R(CD,D,CAR,AR,DNI,P,CAS,AS,T,CUR,CT,CP,CL,CTA,CPA,CLA,FI,FF)

Condiciones de Integridad = {CD → D; CAR → (CD, AR), DNI → P, CAS → (AS, T, CUR, CT, CP, CL), (DNI, CAS, FI) → (CTA, CPA, CLA, FF), ... }

Clave: (CAR, DNI, CAS, FI)

R1(CD,D)

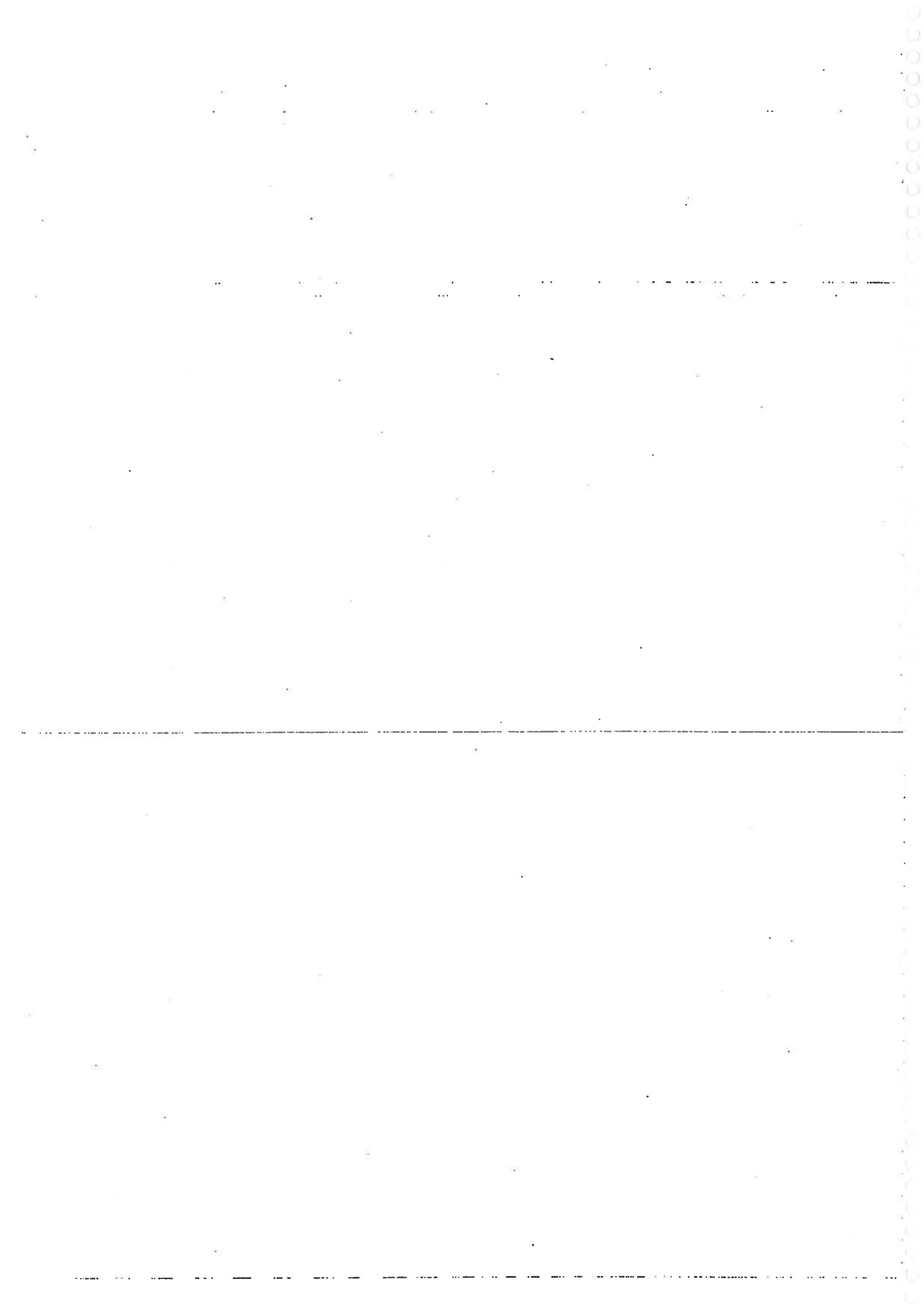
DF1 = {CD → D}

R2( [todo menos D] )

DF = { [...] }

R21(CAR,AR,CD)  
DFR21 = {CAR → CD, AR}

R22( [...] )  
DFR22 = { [...] }



## Resumen Normalización

### \* Axiomas de Armstrong:

1. Reflexiva: Si  $B \subseteq A \Rightarrow A \rightarrow B$

2. Aumentativa: Si  $B \rightarrow A \Rightarrow BC \rightarrow AC$

3. Transitiva: Si  $A \rightarrow B$  y  $B \rightarrow C \Rightarrow A \rightarrow C$

4. Descomposición: Si  $X \rightarrow Y$ ,  $Z \subseteq Y \Rightarrow X \rightarrow Z$

$$\left. \begin{array}{l} X \rightarrow Y \\ Z \subseteq Y \Rightarrow Y \rightarrow Z \end{array} \right\} \underline{X \rightarrow Z}$$

5. Unión: Si  $X \rightarrow Y$ ,  $X \rightarrow Z \Rightarrow X \rightarrow YZ$

6. Pseudotransitiva: Si  $X \rightarrow Y$ ,  $WY \rightarrow Z \Rightarrow WX \rightarrow Z$

$$\left. \begin{array}{l} X \rightarrow Y \\ WY \rightarrow Z \\ WY \rightarrow Z \end{array} \right\} \underline{WX \rightarrow Z}$$

### \* Algoritmo de Ullman: Para calcular el cierre de un conjunto de atributos.

$$A^+ = A$$

while ( $A^+$  cambie) {

    for each df  $X \rightarrow Y \in DF$  //Comprobar las dep. funcionales.

        if  $X \subseteq A^+ \Rightarrow A^+ = A^+ \cup Y$

            Si el antecedente  
            está en el cierre      Añade el consecuente.

### \* Forma normal de Boyce-Codd (FNBC)

Para cualquier dependencia funcional ( $X \rightarrow Y$ ) no trivial de DF se verifica que el antecedente ( $X$ ) es superclave.

### - Algoritmo de descomposición en FNBC:

Resultado = R.

while ( $V(U) \in \text{Resultado}$  no sea FNBC) do {

    Sea  $X \rightarrow Y \in DF$ , no trivial, con  $X \in U$ , no superclave

    Resultado = (Resultado - V)  $\cup$  (S(X,Y))  $\cup$  T(X, U - XY)

}

### \* Tercera forma normal (3FN)

Si se verifica que para cualquier dependencia funcional ( $X \rightarrow Y$ ) que pertenezca a DF y que sea no trivial se verifica que, o el antecedente  $X$  es superclave, o  $Y$  es un atributo principal.

### - Algoritmo de síntesis de 3FN:

Resultado = Vacío;

for each  $(X \rightarrow Y)$  perteneciente a DFC do {

    if  $(XY \in S, \forall S \in \text{Resultado})$  then

        Resultado = Resultado  $\cup$   $S(X,Y)$ ;

    if  $(\nexists S \in R, S \text{ no contiene ninguna clave})$  then

        Resultado = Resultado  $\cup$   $S(k)$ ;     ( $k$  clave arbitraria de  $R$ ).  
    }

### \* Recubrimiento minimal (canónico).

DFC = DF;

DO {

    Aplicar unión en las dep. f. de DFC

    Eliminar atributos raraos. }

while DFC cambie;

## NORMALIZACIÓN

Ejercicios libro "E. Ríos Cornelio".

1) Sea el esquema  $R(A,B,C,D,E)$  con las dependencias:

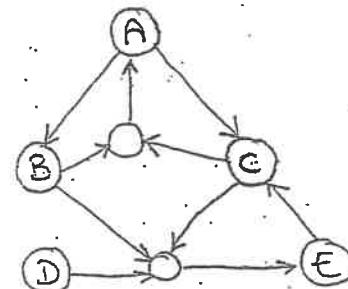
$\{A \rightarrow BC, BC \rightarrow A; BCD \rightarrow E, E \rightarrow C\}$ . Normalizar  $R$  en FNBC.

$$A^+ = \{A, BC\}$$

$$BC^+ = \{B, C, A\}$$

$$BCD^+ = \{B, C, D, E, A\} \rightarrow \text{Define a todo el conjunto de atributos superclave.}$$

$$E^+ = \{E, C\}$$



Claves:  $(AD), (BCD), (BDE)$

Proyectando sobre los atributos de  $(BC \rightarrow A)$  y  $(E \rightarrow C)$ , en este orden, se produce la descomposición siguiente:

\*  $R_1(A, B, C)$

$$A \rightarrow B; A \rightarrow C; BC \rightarrow A$$

Claves:  $(A), (BC)$

FNBC.

\*  $R_{21}(E, C)$

$$E \rightarrow C$$

Clave:  $(E)$

FNBC.

\*  $R_2(B, C, D, E)$

$$BCD \rightarrow E; E \rightarrow C$$

Claves:  $(BCD), (BDE)$

\*  $R_{22}(B, D, E)$

Clave:  $(BDE)$

FNBC

El resultado de la descomposición es:

$R_1(A, BC)$

$R_2(E, C)$

$R_4(B, D, E)$

En el proceso se ha perdido la dependencia  $\{BCD \rightarrow E\}$ .

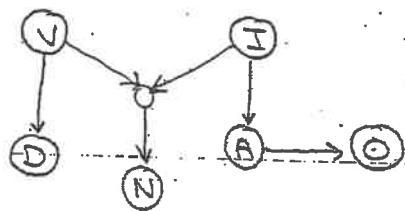
2). Señal el esquema de relación  $R(A, O, I, V, N, D)$  con las siguientes dependencias  
 $DF = \{V \rightarrow D, I \rightarrow A, IV \rightarrow N, A \rightarrow O\}$ . Normalizar  $R$  en FNBC.

$$V^+ = \{V, D\}$$

$$I^+ = \{I, A, O\}$$

$$IV^+ = \{I, V, D, A, N, O\} \rightarrow \text{Si FNBC}$$

$$A^+ = \{A, O\}$$



clave ( $I, V$ ) \*

- Proyectamos sobre los atributos de ( $A \rightarrow O$ ):

$$R_1(A, O)$$

$$DF_1 = \{A \rightarrow O\}$$

clave ( $A$ ). FNBC.

$$R_2(A, I, V, N, D)$$

$$DF_2 = \{V \rightarrow D, I \rightarrow A, IV \rightarrow N\}$$

$$V^+ = \{V, D\} \leftarrow$$

$$I^+ = \{I, A\}$$

clave ( $V, I$ ).

$$IV^+ = \{I, V, N, D, A\}$$

- Proyectamos  $R_2$  sobre los atributos de ( $V \rightarrow D$ ):

$$R_{21}(V, D)$$

$$DF_{21} = \{V \rightarrow D\}$$

clave ( $V$ ). FNBC.

$$R_{22}(A, I, V, N)$$

$$DF_{22} = \{I \rightarrow A, IV \rightarrow N\}$$

$$I^+ = \{I, A\}$$

clave ( $I, N$ ) ? \*

$$IV^+ = \{I, V, N, A\}$$

- Proyectamos  $R_{22}$  sobre los atributos de ( $I \rightarrow A$ ):

$$R_{221}(I, A)$$

$$DF_{221} = \{I \rightarrow A\}$$

clave ( $I$ ). FNBC.

$$R_{222}(I, V, N)$$

$$DF_{222} = \{IV \rightarrow N\}$$

clave ( $IV$ ). FNBC.

El resultado de la descomposición es:  $R_1(A, O)$

$$R_{21}(V, D)$$

$$R_{221}(I, A)$$

$$R_{222}(I, V, N).$$

Y además se han preservado las dependencias.

# Estructura de Datos y de la Información

## Bases de Datos Relacionales

### Problemas de Álgebra Relacional

1) Sean las relaciones R y S siguientes:

R	A	B
a	b	
c	d	
d	e	

S	B	C
b	c	
b	d	
e	a	

(Suponemos que los dominios de los atributos i-ésimos coinciden)  
Hallar los resultados de las siguientes expresiones:

- a) R U S
- b) R - S
- c) R x S
- d) R \* S
- e) P(A)(R)
- f) S(A=C)(R x S)
- g) P (R.B,C) (R Y(S.B=b) S)
- h) P(A)((P(B)(S)) \* R)

2) Considérese el esquema relacional siguiente:

VIVE( nombre-persona, calle, ciudad-persona )  
CP: (nombre-persona)

TRABAJA( nombre-persona, nombre-compañía, salario )  
CP: (nombre-persona, nombre-compañía)

LOCALIZACIÓN( nombre-compañía, ciudad-compañía )  
CP: (nombre-compañía)

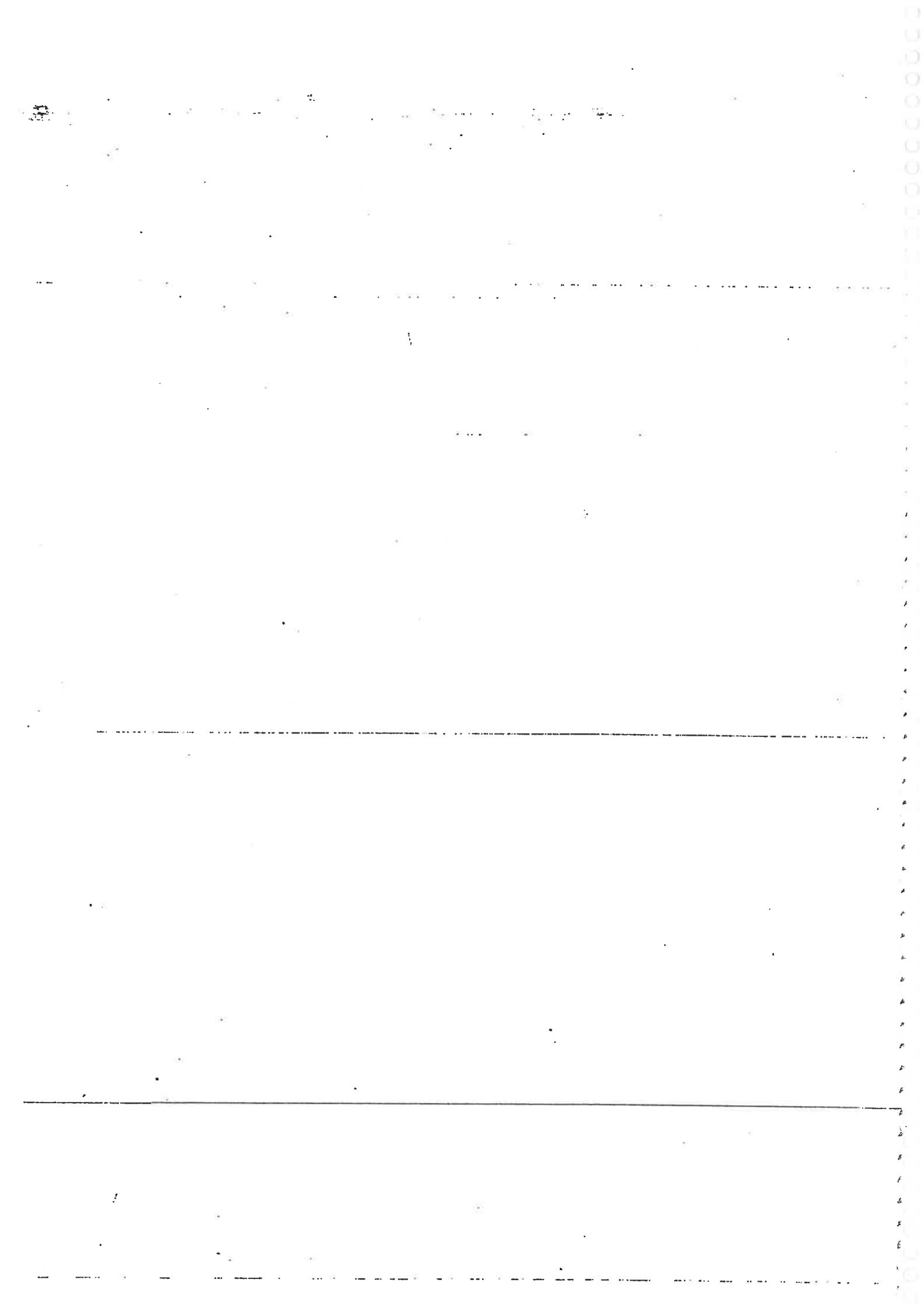
SUPERVISA( nombre-persona, nombre-supervisor, nombre-compañía )

CP: (nombre-persona, nombre-compañía)

CA: (nombre-persona, nombre-compañía), (nombre-supervisor, nombre-compañía) 2 claves fijas.  
La persona trabaja en esa compañía. El supervisor trabaja en la misma compañía.

Constrúyase una expresión en álgebra relacional para cada una de las siguientes consultas:

- a) ¿Qué información se extrae a partir de las claves para cada una de las relaciones anteriores?
- b) Hallar el nombre de todas las personas que trabajan para la compañía C1.
- c) Hallar el nombre de todas las personas que viven en la misma ciudad en que se halla la compañía para la que trabajan.
- d) Hallar los nombres de las personas que trabajan en compañías situadas en La Laguna.
- e) Hallar los nombres de personas que trabajan en más de una compañía.
- f) Hallar las personas que son supervisadas por el mismo supervisor que supervisa a la persona P1 en la compañía C1



# HOJA 1: PROBLEMAS DE ÁLGEBRA RELACIONAL

Corregir soon!

1) Sean las relaciones R y S siguientes:

(Suponemos que los dominios de los atributos i-ésimos coinciden).

	A	B
a	b	
c	d	
d	e	

	B	C
b	c	
b	d	
e	a	

Hallar los resultados de las expresiones:

a) RUS:

RUS
a b
c d
d e
b c
b d
e a

b) R-S: Túplas de la primera relación que no aparecen en la segunda.

R-S
a b
c d
d e

c) R x S: Todas las combinaciones posibles de t-uplas.

R x S	A	B	B	C
a	b	b	c	
a	b	b	d	
a	b	e	a	
c	d	b	c	
c	d	b	d	
c	d	e	a	
d	e	b	c	
d	e	b	d	
d	e	e	a	

d) R \* S: las t-uplas con los atributos comunes, tengan los mismos valores.  
B = Campo común.

R * S	A	B	C
a	b	c	
a	b	d	
d	e	a	

f) S(A=C) (R x S)

	A	B	B	C
a	b	e	a	
c	d	b	c	
d	e	b	d	

e) P(A)(R): Tabla R, proyectamos A.

P(A)(R)	A
a	
c	
d	

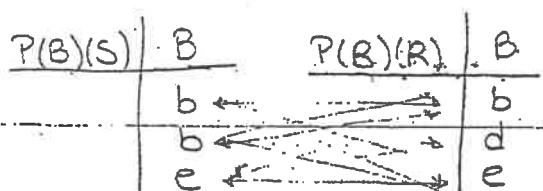
En la tabla (R x S) seleccionaremos las t-uplas que cumplen A = C.

g)  $P(R.B, C) \cap (R \setminus (S.B = b)) S$

<u>R.B , C</u>	<u><math>R \setminus (S.B = b)</math></u>	<u>A . R.B S.B , C</u>
b ; c		a ; b ; b ; c
b ; d		a ; b ; b ; d
d ; c		c ; d ; b ; c
d ; d		c ; d ; b ; d
e ; c		d ; e ; b ; c
e ; d		d ; e ; b ; d

( $S.B = "b"$ )  $\rightarrow$  De la tabla S, las tuplas de la columna B que sean "b".

h)  $P(A)((P(B)(S)) * R)$  :



Primero hacemos el producto cartesiano,  
y luego me quedo con las tuplas que  
coincidan en el campo común.

	<u>A</u>	<u>R.B</u>	<u>S.B</u>
(a)	<u>b</u>	<u>b</u>	<u>b</u>
(a)	<u>b</u>	<u>b</u>	<u>b</u>
a	b	e	
c	d	b	
c	d	b	
c	d	e	
d	e	b	
d	e	b	
(d)	<u>e</u>	<u>e</u>	<u>e</u>

	<u>P(A)</u>	<u>A</u>
		a
		d

2. Considerese el esquema relacional. Constrúyase una expresión en álgebra relacional para cada una de las consultas.

a) ¿Qué información se extrae a partir de las claves para cada una de las relaciones anteriores?

1) Una persona sólo vive en una sola dirección.

Cuanto menos atributos tiene la clave, menos libertad tiene la pers.

1) Nombre\_persona.

2) Nombre\_persona, Nombre\_compañía.

3) Nombre\_compañía → Una compañía sólo puede tener una localización.

2) La compañía puede tener varios empleados y un empleado puede trabajar en varias compañías.

4) Nombre\_persona, Nombre\_compañía.

4) Una persona en una compañía sólo tiene un supervisor.

b) Hallar el nombre de todas las personas que trabajan para la compañía C1.

$P(\text{Nombre\_Persona})(S(\text{Nombre\_Compañía} = \text{C1}) (\text{TRABAJA}))$ .

$P(\text{N.P.}(S(\text{NC} = \text{C1}) (\text{TRABAJA})))$ . ✓

c) Hallar el nombre de todas las personas que viven en la misma ciudad en que se halla la compañía para la que trabajan.

$P(\text{N.P.}(S(\text{CDP} = \text{CDC})(\text{VIVE} * \text{TRABAJA} * \text{LOCALIZACION}))$ . ✓

Unión Natural. Los tipos participantes en la combinación tengan los mismos valores en los atributos comunes.

SQL  $\rightarrow$  NATURAL JOIN

d) Hallar los nombres de las personas que trabajan en compañías situadas en la Legua.

$P(\text{N.P.}(S(\text{CDC} = "la Legua") (\text{LOCALIZACION} * \text{TRABAJA})))$ .

f) Hallar los nombres de las personas que son supervisadas por el mismo supervisor que supervisa a la persona  $P_1$  en la compañía  $C_1$ .

$$T_3: P(N\_S)(S(N\_P = P_1) \wedge (N\_C = C_1)) \text{ (SUPERVISA)}$$

$$P(N\_P) \text{ (T * SUPERVISA)}$$

Primero hallamos el nombre del supervisor de dicha persona en dicha compañía.

Hacemos una junión natural con el nombre del supervisor y la tabla supervisa.

e) Hallar los nombres de las personas que trabajan en más de una compañía.

Sea  $T_1 = T_2 = \text{TRABAJA} \rightarrow$  (Suponemos 2 compañías)

$$P(N\_P)(S(T_1.N\_C \neq T_2.N\_C) \wedge (T_1.N\_P = T_2.N\_P) \text{ (T}_1 \times T_2\text{)}).$$

Nombres de compañías distintas

Nombres de persona igual en ambas compañías

Relación que contiene todas las combinaciones posibles de tipos, una de cada uno de los factores

f) Personas que sólo trabajan en compañías situadas en la misma ciudad donde ellos viven.

## Estructura de Datos y de la Información

### Bases de Datos Relacionales

### Problemas de Álgebra Relacional

3) Sean los esquemas de relación siguientes:

HOMBRE( nombre-hombre, edad )

CLAVE: (nombre-hombre)

MUJER( nombre-mujer, edad )

CLAVE: (nombre-mujer)

HSIM( nombre-hombre, nombre-mujer )

CLAVE: (nombre-hombre, nombre-mujer)

SIGNIFICADO: El hombre nombre-hombre cae simpático a la mujer nombre-mujer.

MSIM( nombre-hombre, nombre-mujer )

CLAVE: (nombre-hombre, nombre-mujer)

SIGNIFICADO: La mujer nombre-mujer cae simpática al hombre nombre-hombre.

MATRIMONIO( nombre-hombre, nombre-mujer )

CLAVE: (nombre-hombre) (nombre-mujer)

Escribir las siguientes consultas en álgebra relacional:

- Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos.  $HSIM \times HSIM$
- Hallar las parejas casadas cuyos componentes se caen mutuamente simpáticos.  $HSIM \cap HSIM$
- Hallar las mujeres casadas a quienes no cae simpático su marido.  $P(NH)(Matrimonio) - HSIM$
- Hallar los hombres misóginos a quienes no cae simpática ninguna mujer.  $P(NH)(Hombre) - P(NH)(HSIM)$
- Hallar los hombres y mujeres asociales a quienes no cae nadie simpático.
- Hallar las mujeres casadas que caen simpáticas a algún hombre.
- Hallar los hombres a quienes sólo caen simpáticas mujeres casadas.
- Hombres a quienes sólo cae simpática su esposa.

4) Sean las relaciones siguientes:

SOCIO( aficionado, videoclub )

SIGNIFICADO: El aficionado es socio de videoclub.

GUSTA( aficionado, película )

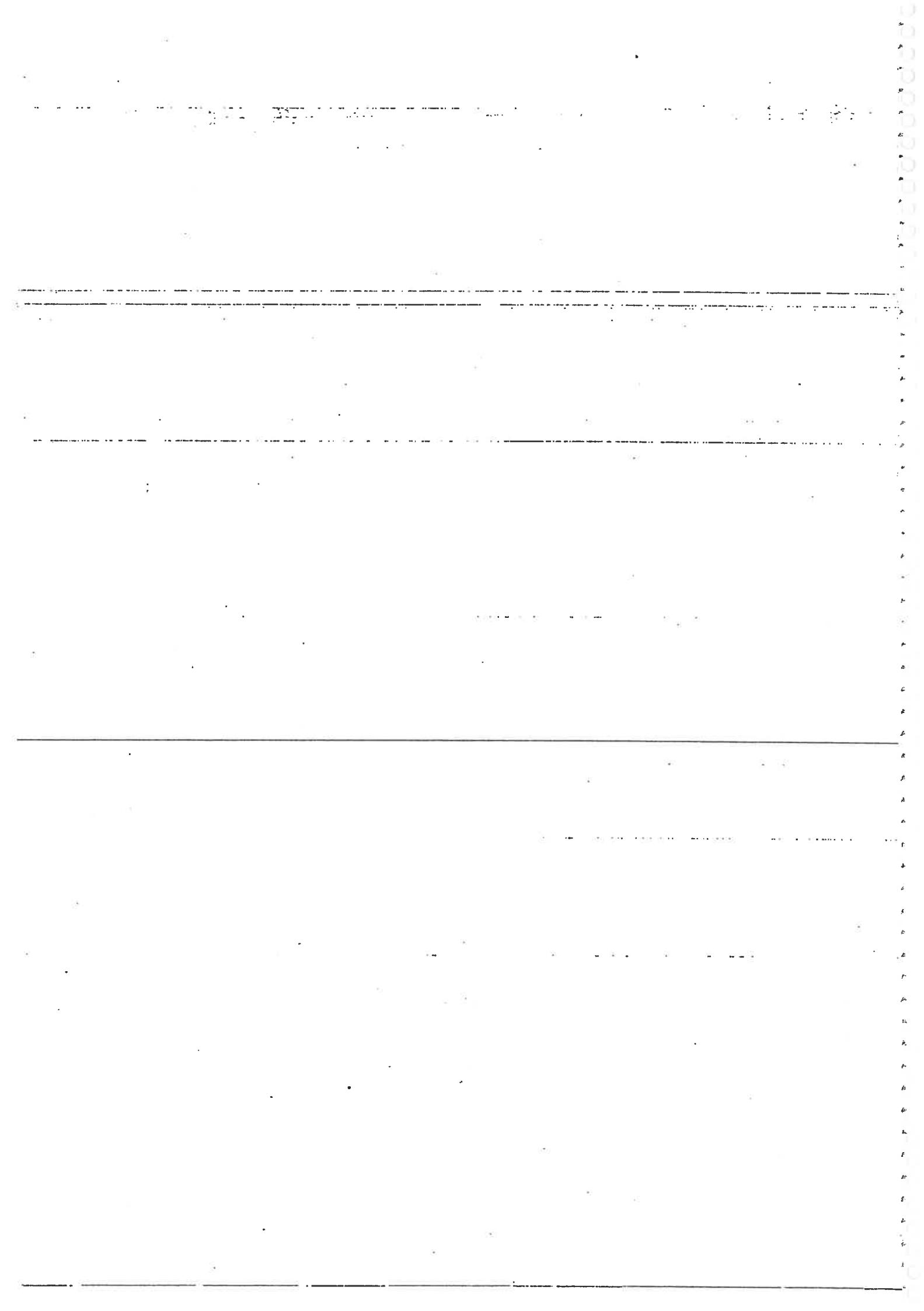
SIGNIFICADO: Al aficionado le gusta la película.

VIDEOTECA( videoclub, película )

SIGNIFICADO: El videoclub dispone en su videoteca de la película.

Escribir las consultas siguientes en álgebra relacional:

- Videoclubs que disponen de alguna película que le guste al aficionado X.
- Aficionados que son socios al menos de un videoclub que dispone de alguna película de su gusto.
- Aficionados que son socios solamente de videoclubes que disponen de alguna película de su gusto.
- Aficionados que sólo son socios de videoclubs que no tienen películas de su gusto.
- Aficionados que son socios de algún videoclub que tiene todas las películas de su gusto.
- Aficionados que son socios solamente de videoclubs que tienen todas las películas de su gusto.



**HOJA 2 : PROBLEMAS DE ÁLGEBRA RELACIONAL**

3). Sean los esquemas de relación; escribir las siguientes consultas en álgebra relacional.

a) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos.

$$(HSIM \cap MSIM) \text{ ó } (HSIM * MSIM)$$

$$\text{Cálculo} = \{ t_2 / t \in HSIM \wedge t \in MSIM \}$$

$$\text{Cálculo dom} = \{ \langle nh, nm \rangle / \langle nh, nm \rangle \in HSIM \wedge \langle nh, nm \rangle \in MSIM \}$$

b) Hallar las parejas casadas cuyos componentes se caen mutuamente simpáticos.

$$HSIM \cap MSIM \cap \text{MATRIMONIO}$$

$$\text{Cálculo} = \{ t_2 / t \in HSIM \wedge t \in MSIM \wedge t \in \text{MATRIMONIO} \}$$

$$\text{Cálculo dom} = \{ \langle nh, nm \rangle / \langle nh, nm \rangle \in HSIM \wedge \langle nh, nm \rangle \in MSIM \wedge \langle nh, nm \rangle \in \text{MATRIMONIO} \}$$

c) Hallar las mujeres casadas a quienes no cae simpática su marido.

$\checkmark P(N\_H (\text{MATRIMONIO} - HSIM))$ .  $\rightarrow$  La diferencia construye una relación formada por todas las t-uplas de la primera relación que NO aparecen en la segunda.

$$\{ t_1 / (\exists \text{mat}) (t[NH] = \text{mat}[NH]) \wedge \exists HSIM (HSIM[NH] = \text{mat}[NH]) \wedge (HSIM[NH] \neq \text{mat}[NH]) \}$$

d) Hallar los hombres misóginos a quienes no cae simpática ninguna mujer.

$P(N\_H (HSIM)) : A \rightarrow$  Mujeres que caen simpáticas a N-H.

$P(N\_H (\text{HOMBRE})) : B \rightarrow$  Nombres de los hombres.

$R = (B - A) \rightarrow$  T-uplas de la primera relación que no aparecen en la segunda.

$\rightarrow$  Cálculo de dominios:

$$\{ \langle nm \rangle / (\exists nh) (\langle nm, nh \rangle \in \text{HOMBRE}) \wedge (\langle nm, nh \rangle \notin HSIM) \}$$

e) Hallar los hombres y mujeres asociados a quienes no cae nadie simpática

- $P(N\_H \_ (MSIM)) : A \rightarrow$  Nombres\_H, al que cae simpática; Nombre\_H
- $P(N\_H \_ (HOMBRE)) : B \rightarrow$  Nombres de los Hombres.
- $P(N\_M \_ (HSIM)) : D \rightarrow$  Nombre\_M, a la que cae simpática Nombre\_H
- $P(N\_M \_ (MUJER)) : E \rightarrow$  Nombres de las Mujeres

$$(B - A) = \boxed{C}$$
$$\boxed{(C \cup F)}$$

$$(D - E) = \boxed{F} \rightarrow (E - D)$$

f) Hallar las mujeres casadas que caen simpáticas a algún hombre.

$$P(N\_M \_ (MSIM)) = A(H)$$
$$P(N\_M \_ (MATRIMONIO)) = B(N)$$
$$(A \cap B)$$

g) Hallar los hombres a quienes sólo caen simpáticas mujeres casadas.

$$P(N\_H \_ (MATRIMONIO)) = A(N) \rightarrow$$
 Mujeres casadas

$$P(N\_H \_ (A * HSIM)) = B(N) \rightarrow$$
 Unión natural  $\rightarrow$  mismos valores en atributo comunes  
HSIM - B

$$P(NOH) \_ (MSIM) - P(NOH) \_ (HSIM - (HSIM * P(NOH) \_ (MATRIMONIO)))$$

h) Hombres a quienes sólo cae simpática su esposa.

$$P(NH) \_ (MATRIMONIO) - P(NH) \_ (HSIM - \underline{\text{MATRIMONIO}})$$

Al hombre le gusta la mujer, y no es su esposa.

$$\left\{ t_{(1)} / (\forall msim) (msim[NH] \neq t[NH]) \vee (msim \in MAT) \right\}$$

$$\left\{ \langle nh \rangle / (\forall nm) (\langle nh, nm \rangle \notin HSIM) \vee (\langle nh, nm \rangle \in MAT) \right\}$$

4). Sean las relaciones. Escribir las consultas en álgebra relacional.

a) Videoclubs que disponen de alguna película que le guste al aficionado X.

$P(\text{Película. } (S(\text{aficionado} = X), (\text{GUSTA})) : A$

$P(\text{Videoclubs. } (A * \text{VIDEOTECA}))$ .

$P(\text{Videoclubs}) (\text{VIDEOTECA} * P(P)(S(A=X') (\text{GUSTA}))$

$$\text{otra} = P(V) S(A=X') (\text{VIDEOTECA} * \text{GUSTA})$$

b) Aficionados que son socios, al menos, de un videoclub que dispone de alguna película de su gusto.

$P(\text{aficionado } (\text{GUSTA})) : A \quad \} \times$

$P(\text{aficionado } (A * \text{socio}))$ .

$\checkmark P(\text{AFICIONADO}) (\text{socio} * \text{VIDEOTECA} * \text{GUSTA})$

c) Aficionados que son socios sólo de videoclubs que disponen de alguna película de su gusto.

$P(\text{AFICIONADO}) (\text{socio}) - P(\text{AFICIONADO}) (\text{socio} - P(\text{AFICIONADO}, \text{VIDEOCLUB})$   
 $(\text{VIDEOTECA} * \text{GUSTA}))$

videoclub que no tienen películas de su gusto.

d) Aficionados que sólo son socios de videoclubs que no tienen películas de su gusto.

$$A = P(\text{aficionado}) (\text{socio} * \text{GUSTA}) \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \begin{array}{l} * \text{VIDEOTECA} \\ \\ \end{array}$$
$$B = P(\text{aficionado}) (\text{socio}) \quad \begin{array}{l} \\ \\ \end{array} \quad \begin{array}{l} \\ \\ \end{array}$$
$$\underline{\underline{B - A}}$$

$P(\text{aficionado}) (\text{socio}) - P(\text{aficionado}) (\text{socio} * \text{GUSTA} * \text{VIDEOTECA})$

e) Aficionados que son socios de algún videoclub que tienen todas las películas de su gusto.

$P(\text{aficionado})$

f) Aficionados que son socios solamente de videoclubs que tienen todas las películas de su gusto.

$P(\text{aficionado})(\text{GUSTA}) = A$

$P(\text{aficionado})(\text{socio}) = B$

$\left. \begin{array}{l} \\ \end{array} \right\} A \cap B \quad \rightarrow \text{Tuplas que aparecen en ambas relaciones.}$

## Bases de Datos

### Problemas de Cálculo Relacional

- 1) Resolver el ejercicio 2 de álgebra relacional, utilizando el cálculo relacional de t-uplas.
- 2) Resolver el ejercicio 3 de álgebra relacional, utilizando el cálculo relacional de dominios.
- 3) Resolver el ejercicio 4 de álgebra relacional utilizando el cálculo relacional de t-uplas.
- 4) Sean los esquemas de relación siguientes:

**PROVEEDOR**( número-proveedor, nombre-proveedor, ciudad-proveedor )  
CLAVE: (número-proveedor)

**ARTÍCULO**( número-artículo, descripción-artículo, color, talla )  
CLAVE: (número-artículo)

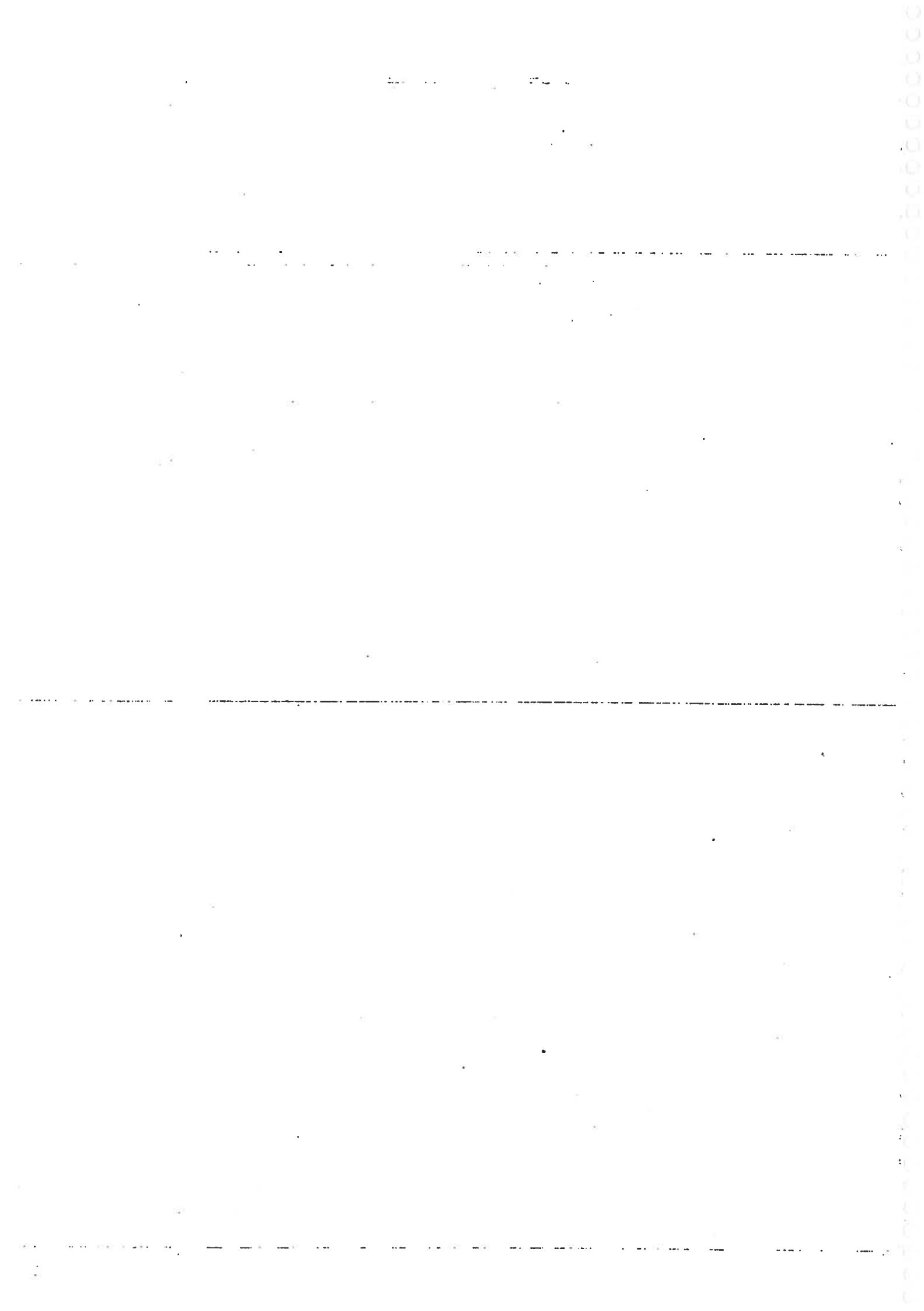
**FÁBRICA**( número-fábrica, nombre-fábrica, ciudad-fábrica )  
CLAVE: (número-fábrica)

**PEDIDO**( número-pedido, número-artículo, número-proveedor, número-fábrica, cantidad )  
CLAVE: (número-pedido, número-artículo, número-fábrica)  
**SIGNIFICADO:** Cada registro representa un pedido del artículo número-artículo al proveedor número-proveedor para la fábrica número-fábrica.

Responder a las preguntas siguientes en:

a) Cálculo relacional de t-uplas. b) Cálculo relacional de dominios.

1. Hallar los identificadores (número-fábrica) de todas las fábricas.
2. Hallar los nombres de las fábricas situadas en Madrid.
3. Artículos tales que ningún otro tiene talla más pequeña.
4. Proveedores que suministran a la fábrica F1.
5. Proveedores que suministran a la fábrica F1 el artículo A1. /
6. Nombre de las fábricas a las que suministra el proveedor P1.
7. Colores de los artículos suministrados por el proveedor P1.
8. Artículos suministrados a las fábricas de Madrid.
9. Proveedores que suministran algún artículo azul a las fábricas de Madrid o Lisboa.
10. Artículos suministrados por proveedores en cuya ciudad hay alguna fábrica.
11. Fábricas que no son abastecidas de artículos azules por proveedores de Madrid.
12. Proveedores que suministran al menos un artículo suministrado por al menos otro proveedor que suministra al menos un artículo azul.
13. Parejas de ciudades tales que un proveedor de la primera abastece a una fábrica de la segunda.
14. Proveedores que suministran un mismo artículo al menos a todas las fábricas.
15. Fábricas que usan sólo artículos que pueden ser suministrados por el proveedor P1.



## PROBLEMAS DE CÁLCULO RELACIONAL

CORRER!  
todo

- 1) Resolver el ejercicio 2 de álgebra relacional, utilizando el cálculo relacional de t-uples.

VIVE (nombre\_persona, calle, ciudad\_p)

TRABAJA (nomb\_persona, nombre\_compañía, salario)

LOCALIZACIÓN (nombre\_compañía, ciudad\_compañía)

SUPERVISA (nombre\_persona, nombre\_supervisor, nombre\_compañía)

- a) Hallar el nombre de todas las personas que trabajan para la compañía C1.

dom(tr) = TRABAJA

$$\{ t_u / (\exists t) (t[N\_P] = tr[N_P]) \wedge (t[N\_C] = 1) \}$$

- b) Hallar el nombre de todas las personas que viven en la misma ciudad en que se halla la compañía para la que trabajan.

dom(tr) = TRABAJA, dom(u) = VIVE, dom(l) = LOCALIZACIÓN

$$\begin{aligned} & \{ t_u / (\exists tr) (tr[N\_P] = t[N_P]) \wedge (tr[N\_P] = u[N_P]) \wedge \\ & (\exists l) (l[C\_C] = u[C\_C]) \wedge (l[N\_C] = tr[N_C]) \} \end{aligned}$$

- c) Hallar los nombres de las personas que trabajan en compañías situadas en la legura.

dom(tr) = TRABAJA ; dom(l) = LOCALIZACIÓN

$$\begin{aligned} & \{ t_u / (\exists l) (l[C\_C] = 'la legura') \\ & (\exists tr) (tr[N_P] = t[N_P]) \wedge (l[N\_C] = tr[N_C]) \} \end{aligned}$$

e) Hallar los nombres de personas que trabajan en más de una compañía.

$$\text{dom}(\text{tr}^1) = \text{dom}(\text{tr}^2) = \text{TRABAJA}$$

$$\{ t \in / (\exists \text{tr}) (\text{tr}[\text{N\_P}] = t[\text{NP}]) \wedge \\ (\text{tr}^1[\text{NP}] = \text{tr}^2[\text{NP}]) \wedge (\text{tr}^1[\text{N\_C}] \neq \text{tr}^2[\text{N\_C}]) \}$$

f) Hallar las personas que son supervisadas por el mismo supervisor que supervisa a la persona P1 en la compañía C1.

$$\text{dom}(s) = \text{SUPERVISA} = \text{dom}(s^1)$$

$$\{ t \in / (\exists s) (s^1[\text{NP}] = 'P1') \wedge (s^1[\text{NC}] = 'C1') \\ (s^1[\text{NS}] = s[\text{NS}]) \wedge (s[\text{NS}] = t[\text{N\_S}]) \}$$

② Resolver el ejercicio 3 de álgebra relacional, utilizando el cálculo relacional de dominios.

HOMBRE (N-H, EDAD)

MUJER (N-H, EDAD)

HSIM (N-H, N-H) → El hombre N-H cae simpático a la mujer N-H.

MSIM (N-H, N-H) → La mujer N-H cae simpática al hombre N-H

MATRIMONIO (N-H, N-H)

a) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos.

$$\{ \langle n\_h, n\_m \rangle / (\exists n\_m) (\langle n\_h, n\_m \rangle \in HSIM) \wedge \\ (\exists n\_h) (\langle n\_h, n\_m \rangle \in MSIM) \}$$

b) Hallar las parejas casadas cuyos componentes se caen mutuamente simpáticos.

$$\{ \langle n\_h, n\_m \rangle / (\langle n\_h, n\_m \rangle \in MATRIMONIO) \wedge (\langle n\_h, n\_m \rangle \in HSIM) \wedge \\ (\langle n\_h, n\_m \rangle \in MSIM) \}$$

c) Hallar las mujeres casadas a quienes no cae simpático su marido.

$$\{ \langle n\_m \rangle / (\exists n\_h) (\langle n\_h, n\_m \rangle \in MATRIMONIO) \wedge \\ (\langle n\_m, n\_h \rangle \notin HSIM) \}$$

Cálculo relacional de t-uplas:  $\text{dom}(m) = \text{MATRIMONIO}$

$$\{ t_6 / (\exists m) (t[N\_H] = m[N\_M]) \wedge (m \in HSIM) \}$$

d) Hallar los hombres misóginos a quienes no cae simpática ninguna mujer.

$$\{ \langle n\_h \rangle / (\exists \text{edad}) (\langle n\_h, \text{edad} \rangle \in HOMBRE) \wedge \neg (\exists n\_m) (\langle n\_h, n\_m \rangle \in MSIM) \}$$

Cálculo relacional de t-uplas:  $\text{dom}(h) = HOMBRE$ ,  $\text{dom}(m) = HSIM$

$$\{ t_6 / (\exists h) (t[N\_H] = h[N\_H]) \wedge \neg (\exists m) (m[N\_M] = t[N\_H]) \}$$

e) Hallar los hombres y mujeres asociados a quienes no cae nadie simpatico

$$A = P(N\_H) \text{ (HOMBRE)} - P(NH) \text{ (HSIM)} \quad ] \quad A \cup B \rightarrow \text{Álgebra relacional}$$

$$B = P(N\_H) \text{ (MUJER)} - P(NH) \text{ (HSIM)} \quad ]$$

$$\text{dom}(h) = \text{HOMBRE} \quad \text{dom}(m) = \text{MUJER}$$

$$\{ t_n / (\exists h) ((t[N\_H] = h[N\_H]) \wedge (\forall m) (t[N\_H] \neq m \text{ sim}[N\_H])) \vee$$

$$(\exists m) ((t[N\_M] = m[N\_M]) \wedge (\forall h) (t[N\_H] \neq h \text{ sim}[N\_H])) \}$$

Calculo de dominios:

$$\{ < n\_h, n\_m > / (\exists \text{edad}) (< n\_h, \text{edad} > \in \text{HOMBRE}) \wedge (\exists n\_m) (< n\_h, n\_m > \in \text{HSIM}) \}$$

$$(\exists \text{edad}) (< n\_m, \text{edad} > \in \text{MUJER}) \wedge (\exists n\_h) (< n\_h, n\_m > \in \text{HSIM}) \}$$

f) Hallar las mujeres casadas qe caen simpaticas a algún hombre.

$$\{ t_n / (\exists mat) (mat[N\_M] = t[N\_H]) \wedge (\exists m) (m \text{ sim}[N\_M] = t[N\_H]) \}$$

$$\{ < n\_m > / (\exists n\_h) (< n\_h, n\_m > \in \text{HATRIMONIO}) \wedge (\exists n\_h) (< n\_h, n\_m > \in \text{HSIM}) \}$$

$n\_h' \rightarrow$  Mejorar la legitimidad de la consulta. Ambos atributos se llamaran igual.

g) Hallar los hombres a quienes sólo caen simpaticas mujeres casadas.

$$\{ t_n / (\forall m) (t[N\_H] \neq m \text{ sim}[N\_H]) \wedge (\exists mat) (mat[N\_H] = t[N\_H]) \}$$

$$\{ < n\_h > / (\forall n\_m) (< n\_h, n\_m > \notin \text{HSIM}) \vee (\exists n\_h) (< n\_h, n\_m > \in \text{HATRIMONIO}) \}$$

$$\{ < n\_h > / (\forall n\_m) (< n\_h, n\_m > \in \text{HSIM}) \wedge (\exists n\_h') (< n\_h, n\_m > \in \text{HATRIMONIO}) \}$$

h) Hombres a quienes sólo cae simpatica su esposa.

$$\{ t_n / (\forall m) (m \text{ sim}[N\_H] \neq t[N\_H]) \vee (m \text{ sim} \in \text{HATRIMONIO}) \}$$

$$\{ < n\_h > / (\forall m) (< n\_h, n\_m > \notin \text{HSIM}) \vee (< n\_h, n\_m > \in \text{HATRIMONIO}) \}$$

$\{ < a > /$

- 3) Resolver el ejercicio 4 de álgebra relacional utilizando el cálculo relacional de t-uplas.

SOCIO (aficionado, videoclub) → El aficionado es socio del videoclub

GUSTA (aficionado, película) → Al aficionado le gusta la película.

VIDEOTECA (videoclub, película). → El videoclub dispone de la película.

- a) Videoclubs que disponen de alguna película que le guste al aficionado X.

$$\text{dom}(g) = \text{GUSTA}, \text{dom}(v) = \text{VIDEOTECA}$$

$$\{tu / (\exists v)(v[\text{VIDEoclub}] = t[\text{VIDEoclub}]) \wedge$$

$$(\exists g)(g[\text{AFICIONADO}] = 'x') \wedge (g[\text{PELICULA}] = v[\text{PELICULA}])\}$$

- b) Aficionados que son socios al menos de un videoclub que dispone de alguna película de su gusto.

$$\text{dom}(s) = \text{SOCIO}, \text{dom}(g) = \text{GUSTA}, \text{dom}(v) = \text{VIDEOTECA}$$

$$\{tu / (\exists s)(s[\text{AFICIONADO}] = t[\text{AFICIONADO}]) \wedge$$

$$(\exists v)(s[\text{VIDEoclub}] = v[\text{VIDEoclub}]) \wedge$$

$$(\exists g)(g[\text{PELICULA}] = v[\text{PELICULA}] \wedge (g[\text{AFICIONADO}] = s[\text{AFICIONADO}]))\}$$

- c) Aficionados que son socios solamente de videoclubes que disponen de alguna película de su gusto

$$\{tu / (\forall s)(s[\text{AFICIONADO}] \neq t[\text{AFICIONADO}] \vee (s[\text{AFICIONADO}] = t[\text{AFICIONADO}]) \wedge$$

$$(\exists g)(\exists s)((s[\text{VIDEoclub}] = v[\text{VIDEoclub}]) \wedge$$

$$(v[\text{PELICULA}] = g[\text{PELICULA}] \wedge (g[\text{AFICIONADO}] = t[\text{AFICIONADO}])))\}$$

- d) Aficionados que sólo son socios de videoclubes que no tienen películas de su gusto.

$$\{tu / (\exists s)(s[\text{AFICIONADO}] = t[\text{AFICIONADO}) \wedge$$

$$(\exists v)(v[\text{VIDEoclub}] = s[\text{VIDEoclub}]) \wedge$$

$$\neg(\exists g)(g[\text{PELICULA}] = v[\text{PELICULA}])\}$$

e) Aficionados que son socios de algún videoclub que tiene todas las películas de su gusto.

$$\{ t_4 / (\exists s)(\exists g)$$

(\*)

f) Aficionados que son socios solamente de videoclubs que tienen todas las películas de su gusto.

$$\{ t_4 / (\exists s)(s[\text{AFICIONADO}] = t[\text{AFICIONADO}]) \wedge (s[\text{VIDEoclUB}] = \sigma[\text{VIDEoclUB}]) \\ \wedge (\forall g)(\forall q)(g[\text{PELICULA}] = \sigma[\text{PELICULA}]) \}$$

- Aficionados que no son socios de ningún videoclub donde tengan alguna película de su gusto.

$$\{ t_4 / (\forall s)(s[\text{AFICIONADO}] \neq t[\text{AFICIONADO}]) \vee ((s[\text{AFICIONADO}] = t[\text{AFICIONADO}]) \wedge \\ \forall(\exists \sigma)^{\neg}(\exists g)(s[\text{VIDEoclUB}] = \sigma[\text{VIDEoclUB}]) \wedge \\ (\sigma[\text{PELICULA}] = g[\text{PELICULA}]) \wedge \\ (g[\text{AFICIONADO}] = t[\text{AFICIONADO}]) \}$$

Sean las relaciones siguientes:

PROVEEDOR (NP, NOMP, CIUDADP)

Clave Primaria = NP.

$\left\{ \begin{array}{l} NP = \text{Número Proveedor.} \\ NOMP = \text{Nombre Proveedor.} \\ CIUDADP = \text{Ciudad Proveedor.} \end{array} \right.$

ARTICULO (NA, DESA, COLOR, TALLA)

Clave Primaria = NA.

$\left\{ \begin{array}{l} NA = \text{Número Artículo.} \\ DESA = \text{Descripción Artículo.} \end{array} \right.$

FABRICA (NF, NOMF, CIUDADF)

Clave Primaria = NF.

$\left\{ \begin{array}{l} NF = \text{Número Fábrica.} \\ NOMF = \text{Nombre Fábrica.} \\ CIUDADF = \text{Ciudad Fábrica.} \end{array} \right.$

PEDIDO (NP, NA, NF, CANTIDAD)

Clave Primaria = NP, NA, NF.

1. Hallar los identificadores de todas las fábricas.

P(NF) (FABRICA)

2. Hallar los nombres de las fábricas situadas en Madrid.

P(NOMF) S (CIUDADF = 'Madrid') (FABRICA)

3. Artículos tales que, ningún otro tiene talla más pequeña.

ARTICULO = ARTICULO<sub>1</sub>      ARTICULO = ARTICULO<sub>2</sub>

P(NA)(ARTICULO) - P(ART1.NA)(S (ART1.TALLA > ART2.TALLA) (ART1 x ART2))

4. Procedores que suministran a la fábrica F1.

P(NP) S (NF = 'F1') (PEDIDO)

5. Procedores que suministran a la fábrica F1 el artículo A1.

P(NP) S (NF = 'F1') ^ (NA = 'A1') (PEDIDO)  $\odot$  Duda!

P(NP) S (NF = 'F1') (PEDIDO) = A  
P(NP) S (NA = 'A1') (PEDIDO) = B } A \cap B

6. Nombres de las fábricas a las que suministra el proveedor P1.

P(NOMP) S (NP = 'P1') (FABRICA \* PEDIDO)

7. Colores de los artículos suministrados por el proveedor P1.

$P(\text{COLOR}) \cap S(NP = 'P1') (ARTICULO * PEDIDO)$

8. Qué proveedores suministran a las fábricas F1 y F2. (simultáneamente)

$P(NP) \cap S(NF = 'F1') (PEDIDO) = A$  } A ∩ B - Intersección en ambas tablas.  
 $P(NP) \cap S(NF = 'F2') (PEDIDO) = B$  }

9. Proveedores que suministran artículos azules a la fábrica F1.

$P(NA) \cap S(\text{COLOR} = 'Azul') (ARTICULO) = A$  }  
 $P(NP) \cap S(NF = 'F1') (PEDIDO) = B$  } A \* B

→ Solución del libro:

$P(NP) \cap S((NF = 'F1') \wedge (\text{COLOR} = 'Azul')) (PEDIDO * ARTICULO)$

10. Artículos suministrados a las fábricas de Madrid.

$P(NA) \cap S(CIUDADF = 'Madrid') (FABRICA * PEDIDO)$

11. Proveedores que suministran algún artículo azul a las fábricas de Madrid o Lisboa.

$P(NP) \cap (S((CIUDADF = 'Madrid') \vee (CIUDADF = 'Lisboa')) \wedge (\text{COLOR} = 'Azul')) (PEDIDO * FABRICA * ARTICULO)$

c?

$P(NP) \cap S(\text{COLOR} = 'Azul') (ARTICULO * PEDIDO) : A$

$P(NP) \cap S(CIUDADF = 'Madrid') (FABRICA * PEDIDO) : B$

$P(NP) \cap S(CIUDADF = 'Lisboa') (FABRICA * PEDIDO) : C$

$A \cap B \cap C$

↓  
Solución Libro  
Cornelio.

?? Duda!

12. Artículos suministrados por proveedores en cuya ciudad hay alguna fábrica.

$P(NP) \cdot S(CIUDADP = CIUDADF) (FÁBRICA * PROVEEDOR) = A$

$P(NA) (PEDIDOS * A)$

13. Artículos suministrados a las fábricas de Madrid por proveedores de Madrid.

$P(NF) \cdot S(CIUDADF = 'Madrid') (FÁBRICA) = A$

$P(NP) \cdot S(CIUDADP = 'Madrid') (PROVEEDOR) = B$

$P(NA) (PEDIDOS * A * B)$

14. Fábricas abastecidas por al menos un proveedor de distinta ciudad.

$P(NF) \cdot S(CIUDADP \neq CIUDADF) (PEDIDO * PROVEEDOR * FÁBRICA)$

④ Duda!

15. Fábricas que no son abastecidas de artículos azules por proveedores de Madrid.

$P(NF) \cdot S(CIUDADP = 'Madrid') (PEDIDO * PROVEEDOR) = A$

$P(NF) \cdot S(COLOR = 'Azul') (PEDIDO * ARTICULO) = B$

$P(NF) (FÁBRICA) - (A * B)$

Solución libro Cornelio:

$P(NF)(FÁBRICA) - P(NF)(S(CIUDADP = 'Madrid') \wedge (COLOR = 'Azul')) (PEDIDO * PROVEEDOR * ARTICULO)$

16. Proveedores que suministran al menos un artículo suministrado por al menos otro proveedor que suministra al menos un artículo azul.

A:  $P(NP) \cdot S(COLOR = 'Azul') (ARTICULO * PEDIDO) \rightarrow$  Proveedores de artículos azules.

B:  $P(NA) (PEDIDO * A) \rightarrow$  Artículos suministrados por proveedores de art. azules.

$P(NP) (PEDIDO * B) \rightarrow$  Proveedores de los artículos de B.

En conclusión:

$P(NP) (PEDIDO * (P(NA) (PEDIDO * P(NP) \cdot S(COLOR = 'Azul')) (ARTICULO * PEDIDO)))$

17. Fábricas que usan al menos un artículo suministrado por el proveedor P1.

A:  $P(NA) \wedge S(NP = 'P1') (PEDIDO) \rightarrow$  Artículos suministrados por el proveedor P1  
 $P(NF) (PEDIDO \neq A) \rightarrow$  Fábricas que usan dicho proveedor.

Libro Cornelio:

$P(NF) (PEDIDO \neq P(NA) \wedge S(NP = 'P1') (PEDIDO))$

18. Parejas de ciudades tales que un proveedor de la primera abastece a una fábrica de la segunda.

$P(CIUDADP, CIUDADF) (PEDIDO \neq PROVEEDOR \neq FABRICA)$

19. Obtener las tripletas de los valores de <CIUDAD, NA, CIUDAD> tales que un proveedor de la primera ciudad abastece el artículo NA a una fábrica de la segunda ciudad.

$P(CIUDADP, NA, CIUDADF) (PEDIDO \neq PROVEEDOR \neq FABRICA).$

20. Repetir la pregunta anterior, pero sin obtener las tripletas en que ambas ciudades son la misma.

$P(CIUDADP, NA, CIUDADF) \wedge S(CIUDADP \neq CIUDADF) (PEDIDO \neq PROV \neq FAB)$

21. Proveedores que suministran un mismo artículo, al menos, a todas las fábricas.

$P(NP) \left( P(NP, NA, NF) (PEDIDO) / P(NF) (FABRICA) \right) \rightarrow$  Se aplica al primer cjo  
Elimina todas las fábricas que no coinciden.

Lista las fábricas con el num del producto

Al dividir elimina en el primer conjunto las fábricas que no coinciden en el primero.

22. Fábricas que tienen como único proveedor al P1.

$$(P(NF) \wedge S(NP = 'P1') (PEDIDO)) - (P(NF) \wedge S(NP \neq 'P1') (PEDIDO))$$

23. Artículos que son suministrados a todas las fábricas de Madrid (excluyendo los que sólo suministran a algunas).

$$P(NF) \wedge S(CIUDADF = 'Madrid') (FABRICA) : A \quad \left. \begin{array}{l} \\ B/A \end{array} \right\} \quad \text{A estas se terció en 5}$$
$$P(NA, NF) (PEDIDO) : B$$

$$P(NA, NF) (PEDIDO) / P(NF) \wedge S(CIUDADF = 'Madrid') (FABRICA)$$

24. Fábricas que usan, al menos, todos los artículos suministrados por el proveedor P1.

$$P(NA) \wedge S(NP = 'P1') (PEDIDO) : A \quad \left. \begin{array}{l} \\ B/A \end{array} \right\}$$
$$P(NF, NA) (PEDIDO) : B$$

25. Fábricas que usan sólo artículos que pueden ser suministrados por el proveedor P1.

$$P(NF) (FABRICA) - (P(NF) (PEDIDO * (P(NA) (ART) - P(NA) (S(NP = 'P1')) (PEDIDO)))$$

Artículos que no suministra P1.

Fábricas que usan artículos que no son suministrados por el proveedor P1.

En la tabla fabrica, restaremos las fábricas que usan artículos de otros proveedores que no son el P1.

26.- Fábricas abastecidas por el proveedor P1 con todos los artículos que éste suministra.

$$P(NP, NA, NF) \text{ (PEDIDO)} / P(NP, NA) S(NP = 'P1') \text{ (PEDIDO)}$$

27.- Fábricas que obtienen del proveedor P1, total o parcialmente, todos los artículos que usan.

$$P(NF) \text{ (FAB)} - P(NF) P(NP, NF) \text{ (PEDIDO)} - P(NA, NF) S(NP = 'P1') \text{ (PEDIDO)}$$

Artículos y fábricas de P1

Restamos las fábricas que obtienen de P1 artículos:

Restamos todo lo anterior.

28.- Fábricas abastecidas por todos los proveedores que suministran algún artículo de color azul.

$$P(NF, NP) \text{ (PEDIDO)} / P(NP) S(COLOR = 'Azul') \text{ (PEDIDO * ART(COLO))}$$

## Bases de Datos Problemas de SQL

1) Sean los esquemas de relación siguientes:

PROPIETARIO( nombre, DNI, domicilio )  $\rightarrow$  Nombre del propietario, con un DNI y un domicilio.  
CLAVE: DNI.

COCHE(DNI, marca, modelo, matrícula)  $\rightarrow$  Coche propiedad de DNI.  
CLAVE: matrícula.

ACCIDENTE( matrícula, fecha-accidente, nombre-conductor, importe )  
CLAVE: matrícula, fecha-accidente

Responder en SQL:

- a) Hallar el número de usuarios que tuvieron un accidente en 1983.
- b) Hallar el número de accidentes en que estuvieron implicados los vehículos de "Juan Pérez" durante 1980.

Vive (n\_P, calle, c\_P)  
Trabaja (n\_P, n\_c, Suelo)  
Localiz (n\_c, c\_C)  
Supervisa (n\_P, n\_S, n\_c)

2, hoja 1, tema 2. Construyase una expresión en SQL

bajan para la compañía C1.  
ven en la misma ciudad en que se halla la compañía para  
una ciudad que su supervisor.  
a la compañía C1.  
cualquier empleado de la compañía C1.  
ue el salario promedio de las personas que trabajan en la

s.  
omedio, que el sueldo medio de la compañía C1.

utilizando SQL 

4) Resolver el ejercicio 3 de álgebra relacional utilizando SQL.

5) Resolver el ejercicio 4 de cálculo relacional utilizando: SQL.

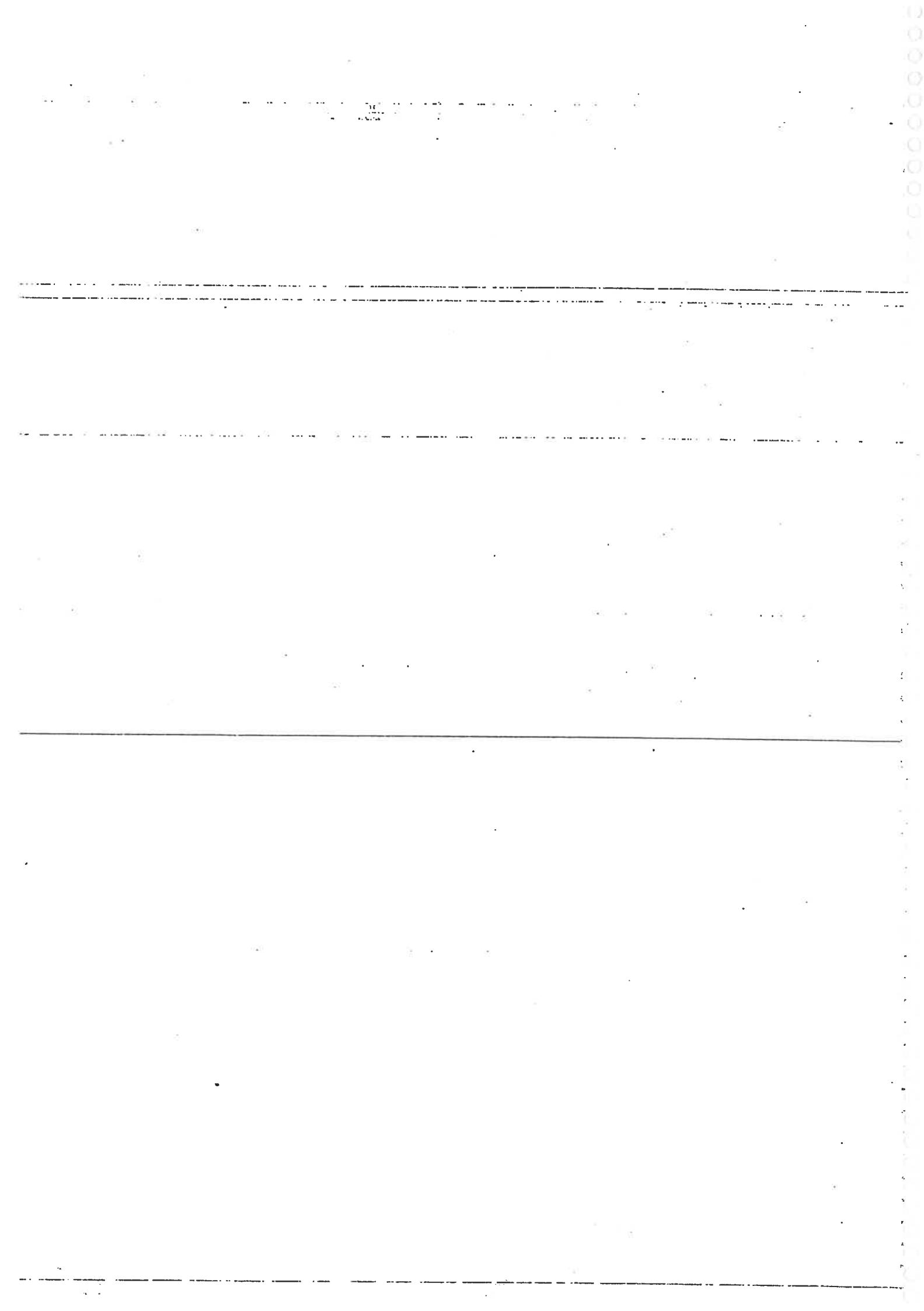
6) Sean las relaciones: R(A, B) y S(A, C). Escribanse expresiones equivalentes en: SQL para las siguientes consultas en cálculo relacional de dominios:

- a) {  $\langle a \rangle / (\exists b)((\langle a, b \rangle \in R) \wedge (b = 17))$  }
- b) {  $\langle a, b, c \rangle / ((\langle a, b \rangle \in R) \wedge (\langle a, c \rangle \in S))$  }
- c) {  $\langle b \rangle / (\forall a)((\langle a, b \rangle \in R) \vee (\exists c)(\langle a, c \rangle \in S)))$  }
- d) {  $\langle a \rangle / (\exists c)((\langle a, c \rangle \in S) \wedge (\exists b, d)((\langle a, b \rangle \in R) \wedge (\langle c, d \rangle \in R)) \wedge$  }

Propietario (nombre, DNI  
domicilio)

Coche (DNI, MARCA, Modelo,  
matrícula)

Accidente (matrícula, Fecha,  
nº Conductor, importe)



- 1)
- a) SELECT COUNT (NºC)  
FROM ACCIDENTE  
WHERE (F\_A = 1983);
- b) SELECT COUNT (MATRICULA)  
FROM ACCIDENTE  
WHERE (NLC = "Juan Pérez")  
AND (F\_A = 1990); X Mal  
Durante todo 1990.
- Propietario = 'Juan Pérez'. 4 propietarios  
no conductor
- b) SELECT COUNT (Matricula)  
FROM ACCIDENTE, PROPIETARIO  
WHERE (Nombre = 'Juan Pérez')  
AND (P.DNI = C.DNI) AND (C.MAT = A.MAT)  
AND (F\_A = 1990);
- c) Propietario con más coches.
- SELECT DNI  
FROM COCHE  
GROUP BY DNI  
HAVING COUNT (MATRICULA) >= ALL (SELECT COUNT (MATRICULA)  
FROM COCHE  
GROUP BY DNI). ✓
- d) SELECT NOMBRE\_C  
FROM ACCIDENTE  
WHERE FA >= ALL (SELECT FA  
FROM ACCIDENTE)
- ← Conductor implicado en el accidente más reciente.
- e) SELECT AVG (COUNT (\*))  
FROM ACCIDENTE  
~~WHERE (F\_A < 1990)~~ → "Por año". ¿?  
GROUP BY YEAR (FA);

## ② Hoja 1, ej. 2:

- VIVE (nombre\_persona, calle, ciudad\_persona) C.P: (nombre persona)
- TRABAJA (nombre persona, nombre\_compañía, salario). CP: (nombre persona,  
nombre\_compañía)
- LOCALIZACIÓN (nombre\_compañía, ciudad\_compañía) CP (nombre\_compañía)
- SUPERVISA (nombre\_persona, nombre\_supervisor, nombre\_compañía)  
C.A = CP: (nombre persona, nombre\_compañía)  
C.A = (nombre supervisor, nombre\_compañía)

a) SELECT (N-P)  
FROM TRABAJA  
WHERE (N-C = C1); → Hallar el nombre de todas las personas que trabajan para la compañía C1.

b) SELECT (N-P)  
FROM VIVE V, TRABAJA T, LOCALIZACIÓN L → Natural Join  
WHERE (L.CIUDAD\_COMPAÑIA = V.CIUDAD\_PERSONA)  
AND (V.NP = T.NP) AND (T.NC = L.NC);

c) Hallar todas las personas que viven en la misma ciudad que su supervisor.

d) Hallar todas las personas que no trabajan para la compañía C1.

```
SELECT N-P  
FROM TRABAJA  
WHERE NOT (N-C = C1);
```

e) SELECT NOMBRE\_PERSONA  
FROM TRABAJA  
WHERE SALARIO > (SELECT SALARIO  
FROM TRABAJA  
WHERE (N-C = C1));

→ Hallar todas las personas que ganan más que cualquier empleado de la compañía C1.

f) SELECT NOMBRE\_PERSONA  
FROM TRABAJA1  
WHERE SALARIO > (SELECT AVG(SALARIO)  
FROM TRABAJA2  
WHERE (T1.NC = T2.NC));

→ Hallar todas las personas que ganan más que el salario promedio de las personas que trabajan en la misma compañía

g) Hallar la compañía que tiene más empleados. (\*)

SELECT NOMBRE\_COMPANIA  
FROM TRABAJA  
GROUP BY NOMBRE\_COMPANIA  
HAVING COUNT(\*) >= ALL (SELECT COUNT(\*)  
FROM TRABAJA  
GROUP BY NOMBRE\_COMPANIA)

h) Hallar las compañías que pagan más, en promedio, que el sueldo medio de la compañía C1.

SELECT NOMBRE\_COMPANIA  
FROM TRABAJA  
GROUP BY NOMBRE\_COMPANIA  
HAVING AVG(SALARIO) > (SELECT AVG(SALARIO)  
FROM TRABAJA  
WHERE (NOMBRE\_COMPANIA = C1));

④

a) Hallar las parejas de hombres y mujeres que se caen mutuamente simpaticos.

SELECT N\_H, N\_M  
FROM HSIM NATURAL JOIN HSIM;

b) Hallar las parejas casadas cuyos componentes se caen mutuamente simpaticos.

SELECT N\_H, N\_M  
FROM HSIM NATURAL JOIN HSIM NATURAL JOIN MATRIMONIO;

\* -> Natural join se puede hacer dos veces con tres tablas ??

No!

c) Hallar las mujeres casadas a quienes no cae simpatico su marido

SELECT N\_M  
FROM MATRIMONIO M  
WHERE N\_M NOT IN (SELECT N\_H  
FROM HSIM H  
WHERE M.NH = H.NM)

d) Hallar los hombres misóginos a quienes no cae simpatica ninguna mujer.

SELECT N\_H  
FROM HOMBRE  
WHERE (N\_H) NOT IN (SELECT N\_H  
FROM HSIM);



e) Hallar los hombres y mujeres asociales a quienes no cae nadie simpatico.

SELECT N\_H, N\_M  
FROM HOMBRE, MUJER  
WHERE NOT IN (SELECT NH  
FROM HSIM  
WHERE N\_H NOT IN (SELECT N\_M  
FROM HSIM)).



f) Hallar las mujeres casadas que caen simpaticas a algún hombre.

SELECT N\_H  
FROM MATRIMONIO  
WHERE EXISTS (SELECT N\_H  
FROM MSIM);

g) Hallar los hombres a quienes sólo caen simpaticas mujeres casadas.

SELECT N\_H  
FROM MSIM  
WHERE N\_H IN (SELECT N\_H  
FROM MATRIMONIO);

Creo que mal por que hay que tener en cuenta el solo

h) Hombres a quienes sólo cae simpatica su esposa.

SELECT N\_H  
FROM MATRIMONIO  
WHERE N\_H EXIST (SELECT N\_H  
FROM MSIM) = f

SELECT N\_H  
FROM MSIM  
WHERE N\_H IN (SELECT N\_H  
FROM MATRIMONIO)

⑤- Ejercicio 4 de Cálculo Relacional utilizando SQL.

1- Hallar los identificadores de todas las fábricas.

```
SELECT Numero_Fabrica  
FROM FABRICA;
```

2- Hallar los nombres de las fábricas situadas en Madrid.

```
SELECT Nombre_Fabrica  
FROM FABRICA  
WHERE (Ciudad_Fabrica = 'Madrid');
```

3- Artículos tales que ningún otro tiene talla más pequeña.

SELECT Numero\_Articulo  
FROM ARTICULO A1  
WHERE Talla = (SELECT MIN(Talla)  
FROM ARTICULO) A1 ✓

4- Proveedores que suministran a la fábrica F1.

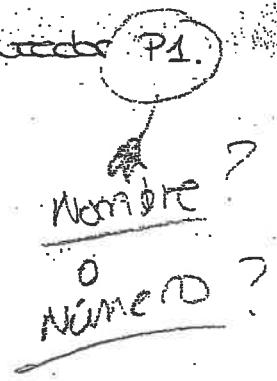
```
SELECT Numero_Proveedor  
FROM PROVEEDOR P, FABRICA F, PEDIDOS PE  
WHERE (Nombre_Fabrica = F1)  
AND (PE.Numero_Fabrica = F.Numero_Fabrica)  
AND (PE.Numero_Proveedor = P.Numero_Proveedor);
```

5- Proveedores que suministran a la fábrica F1 el artículo A1.

```
SELECT Numero_Proveedor  
FROM PROVEEDOR P, FABRICA F, PEDIDO PE  
WHERE (Nombre_Fabrica = F1)  
AND (Numero_Articulo = A1)  
AND (PE.Numero_Fabrica = F.Numero_Fabrica)  
AND (PE.Numero_Proveedor = P.Numero_Proveedor);
```

6. Nombre de las fábricas a las que suministra el proveedor P1.

```
SELECT Nombre_fabrica  
FROM FABRICA NATURAL JOIN PEDIDO  
WHERE (Número_Proveedor = P1);
```



7. Colores de los artículos suministrados por el proveedor P1.

```
SELECT color  
FROM ARTICULO NATURAL JOIN PEDIDO  
WHERE (Número_Proveedor = P1);
```



Si P1 fuera Nombre\_Proveedor:

```
SELECT color  
FROM ARTICULO NATURAL JOIN PEDIDO  
WHERE (Número_Proveedor) = (SELECT Número_Proveedor  
                           FROM PROVEEDOR  
                           WHERE (Nombre_Proveedor = P1));
```

8. Artículos suministrados a las fábricas de Madrid.

```
SELECT Número_Artículo  
FROM FABRICA NATURAL JOIN PEDIDO  
WHERE (Ciudad_Fábrica = 'Madrid');
```

9. Proveedores que suministran algún artículo azul a las fábricas de Madrid o Lisboa.

```
SELECT Número_Proveedor  
FROM FABRICA NATURAL JOIN PEDIDO  
WHERE (Ciudad_Fábrica = 'Madrid') OR (Ciudad_Fábrica = 'Lisboa')  
AND EXISTS (SELECT Número_Artículo  
            FROM ARTICULO NATURAL JOIN PEDIDO  
            WHERE (color = 'Azul'));
```



10. Artículos suministrados por proveedores en cuya ciudad hay alguna fábrica.

SELECT numero\_artículo  
FROM PEDIDO  
WHERE numero\_Proveedor IN (SELECT numero\_Proveedor  
FROM PROVEEDOR, FÁBRICA, PEDIDO PE  
WHERE (Ciudad\_Fábrica = Ciudad\_Proveedor)  
AND (PE.Numero\_Fábrica = F.Numero\_Fábrica)  
AND (PE.Numero\_Proveedor = P.Numero\_Proveedor))

11. Fábricas que no son abastecidas de artículos azules por proveedores de Madrid.

SELECT Numero\_Fábrica  
FROM PEDIDO  
WHERE NOT EXISTS (SELECT Numero\_Fábrica  
FROM PROVEEDOR P, ARTICULO A, PEDIDO PE  
WHERE (ciudad\_Proveedor = 'Madrid')  
AND (Color = 'Azul')  
AND (PE.N\_A = A.N\_A) AND (PE.N\_P = P.N\_P))

12. Proveedores que suministran al menos un artículo suministrado por al menos otro proveedor que suministra al menos un artículo azul

SELECT numero\_Proveedor  
FROM PEDIDO  
WHERE numero\_Proveedor = (SELECT numero\_Proveedor  
FROM PEDIDO NATURAL JOIN ARTICULO  
WHERE (Color = 'Azul'));

13.- Parejas de ciudades tales que un proveedor de la primera abastece a una fábrica de la segunda.

```
SELECT Ciudad_Proveedor, Ciudad_Fábrica  
FROM PROVEEDOR, FÁBRICA, PEDIDO  
WHERE (P.Número_Proveedor = PE.Número_Proveedor)  
AND (F.Número_Fábrica = PE.Número_Fábrica)
```

14.- Proveedores que suministran un mismo artículo al menos a todas las fábricas.

## EXAMEN PRACTICAS 3-D

P1 - P2

SOL-Plus Diseño  
B-D.

Ld Modelo  
Relacional

- Diag. Jerarquía (1..N).

- D. E/R (1..N)  
(N..N)

→ P3

ind. soft de B.D.

P3 → ORACLE

Ld SQL

{ - DML  
- DLL  
- QUERY }

{ - INSERT  
- UPDATE  
- DELETE }

ESTRUCTURAS

{ - CREATE  
- AFTER  
- DROP }

SELECT  
FROM  
[ WHERE  
GROUP BY  
HAVING ]

P4-P5-P6 ⇒ CONSULTAS

ENTIDAD  
ATRIBUTOS  
RELACIONES

BD

- Tablas

- Columnas → Datos

(1..N) → Columnas tabla

(N..N) → Tabla

## SQL

- DNI de los clientes que tienen alguna cuenta en la sucursal con código 1 y número de cuenta comprendido entre 100 y 200.

```
SELECT DNI
FROM DEPOSITO
WHERE (CS=1) AND (NC >= 100) AND (NC <= 200);
          ↓
          AND NC BETWEEN 100 AND 200;
```

Resultado } - Explicto →  
} - Implicito → Subconsulta o consulta anidada.

DNI de las personas que tienen más de cuatro caracteres.

```
SELECT DNI
FROM CLIENTE
WHERE (NBC LIKE '_____%');
```

clientes que tienen ~~alguna~~ cuenta en alguna sucursal de la laguna

```
SELECT DNI
FROM DEPOSITO D
WHERE CS IN (SELECT CS
              FROM SUCURSAL S
              WHERE (CDS = 'La Laguna'));
```

```
SELECT DNI
FROM DEPOSITO NATURAL JOIN SUCURSAL
WHERE CDS = 'La Laguna';
```

clientes que tienen cuenta en todas las sucursales.

```
SELECT DNI
FROM DEPOSITO D1
WHERE NOT EXISTS (SELECT *
                   FROM SUCURSAL
                   WHERE NOT EXISTS (SELECT *
                                     FROM DEPOSITO D2
                                     WHERE (D1.DNI = D2.DNI)
                                     AND (D2.CS = S.CS)));
```

Si el cliente tiene cuenta en todas las sucursales es qg no existe una sucursal en la qg no tenga cuenta.

## Algebra R

$$P = P(DNI, CS) \text{ (DEPOSITO)}$$

$$P = P(CS) \text{ (SUCURSAL)}$$

Q/B.

Calculus:

$$\{ t_{14} / (\forall s) (\exists d) (t[DNI] = d[DNI]) \wedge (d[CS] = s[CS]) \}$$

$$\{ t_{14} / \neg (\exists s) \neg (\exists d) (t[DNI] = d[DNI]) \wedge (d[CS] = s[CS]) \}$$

## PROBLEMAS DE ALGEBRA RELACIONAL:

3) Álculo de t-uplas y cálculo de dominios:

$$c) \{ t_1 / (\exists \text{mat}) (t[NH] = \text{mat}[NH]) \wedge (\text{mat} \in HSIM) \} \rightarrow \text{Mejor hecho!}$$

$$\{ \langle nm \rangle / (\exists nh) (\langle nh, nm \rangle \in MAT) \wedge (\langle nh, nm \rangle \notin HSIM) \}$$

$$d) \{ t_1 / (\exists nh) (t[NH] = h[NH]) \wedge (\forall m \neq m) (msim[NH] \neq t[NH]) \}$$

$$\neg (\exists msim) (msim[NH] = t[NH])$$

$$\{ \langle nh \rangle / (\exists eh) (\langle nh, eh \rangle \in HOMBRE) \neg (\exists nm) (\langle nh, nm \rangle \in HSIM) \}$$

$$e) A(N) = P(NH)(HOMBRE) - P(NH)(HSIM)$$

$$B(N) = P(NH)(MUJER) - P(NH)(HSIM)$$

A ∪ B

$$\{ t_1 / (\exists h) ((t[NH] = h[NH]) \wedge (\forall m \neq m) (msim[NH] \neq t[NH])) \vee$$

$$(\exists m) (t[NH] = m[NH]) \wedge (\forall nh) (hsim[NH] \neq t[NH]) \}$$

$$\{ \langle n \rangle / (\exists eh) (\langle n, eh \rangle \in HOMBRE) \vee (\exists nm) (\langle n, nm \rangle \in HSIM) \}$$

$$(\exists em) (\langle n, em \rangle \in MUJER) \vee (\exists nh) (\langle nh, n \rangle \in HSIM) \}$$

$$f) \{ t_1 / (\exists \text{mat}) (t[NH] = \text{mat}[NH]) \wedge (\exists msim) (msim[NH] = t[NH]) \}$$

$$\{ \langle nm \rangle / (\exists nh) (\langle nh, nm \rangle \in MAT) \wedge (\exists nh') (\langle nh', nm \rangle \in HSIM) \}$$

= mejorar la legibilidad de la consulta. Ambos atributos se llaman igual.

g) Conjunto de hombres - gto de hombres a los que no son simpáticos los solteros.

$$A = P(NH)(MUJER) - P(NH)(HATRIMONIO)$$

Resuelto por el complementario.

$$P(NH)(\bar{HSIM}) - P(NH)(HSIM \cap A)$$

Hombres tales que para cualquier mujer que le caiga simpática  
está casada.

$$\{ t_A / (\forall s) (msim[NH] \neq t[NH]) \vee (\exists m) (mat[NM] = msim[NH]) \}$$

$$\{ \langle nh \rangle / (\forall nm) (\langle nh, nm \rangle \notin MSH) \vee (\exists nh) (\langle nh, nm \rangle \in MATRIMONIO) \}$$

o no le gusta ninguna      este casado

mujer.

$$\{ \langle nh \rangle / (\forall nm) (\langle nh, nm \rangle \in MSH) \wedge \neg (\exists nh') (\langle nh', nm \rangle \in MAT) \}$$

4)

a)  $\{ \langle v \rangle / (\exists p) (\langle v, p \rangle \in VIDEOTECA) \wedge (\langle 'x', p \rangle \in GUSTA) \}$

b)  $\{ \langle a \rangle / (\exists v, p) (\langle a, v \rangle \in SOCIO) \wedge (\langle v, p \rangle \in VIDEOTECA) \wedge (\langle a, p \rangle \in GUSTA) \}$

c) Celulo de tuplas: si el aficionado es socio, Para cualquier videoclub  
del cual es socio, tiene películas de su gusto.

$$\{ t_A / (\forall s) (t[A] \neq s[A]) \vee (\exists g, v) ((g[A] = s[A]) \wedge (g[P] = v[P]) \wedge (g[V] = s[V])) \}$$

d) Para cualquier videoclub

## Bases de Datos

### Problemas de Normalización

- 1) Sea el esquema  $R(A, B, C, D, E)$  con las dependencias:  
 $A \rightarrow BC; BC \rightarrow A; BCD \rightarrow E; E \rightarrow C$   
Normalizar  $R$  en FNBC.
- 2) Sea el esquema  $R(A, B, C, D)$  con las dependencias:  
 $AB \rightarrow D; C \rightarrow D; AB \rightarrow C; C \rightarrow B$ 
  - a) Normalizar  $R$  en FN3. ¿Se preservan las dependencias funcionales?
  - b) Normalizar  $R$  en FNBC. ¿Se preservan las dependencias funcionales?
  - c) Aplicar a  $R$  el algoritmo de descomposición que preserva dependencias. comprobar que se obtienen relaciones FN3.
- 3) Añadir a la relación siguiente las filas necesarias para que se cumplan en ella las DPs:  $A \rightarrow\rightarrow BC; CD \rightarrow\rightarrow BE$

A	B	C	D	E
a	b	c	d	e
a	1	c	2	e
3	b	c	d	4

- 4) Sea el esquema  $R(A, B, C, D, E)$  con las dependencias:

$A \rightarrow\rightarrow BC; DE \rightarrow\rightarrow C$

Demostrar que  $AD \rightarrow\rightarrow BE$

- 5) Sea el esquema  $R(A, B, C, D, E, F)$  con las dependencias:

$A \rightarrow\rightarrow BCD; B \rightarrow AC; C \rightarrow D$

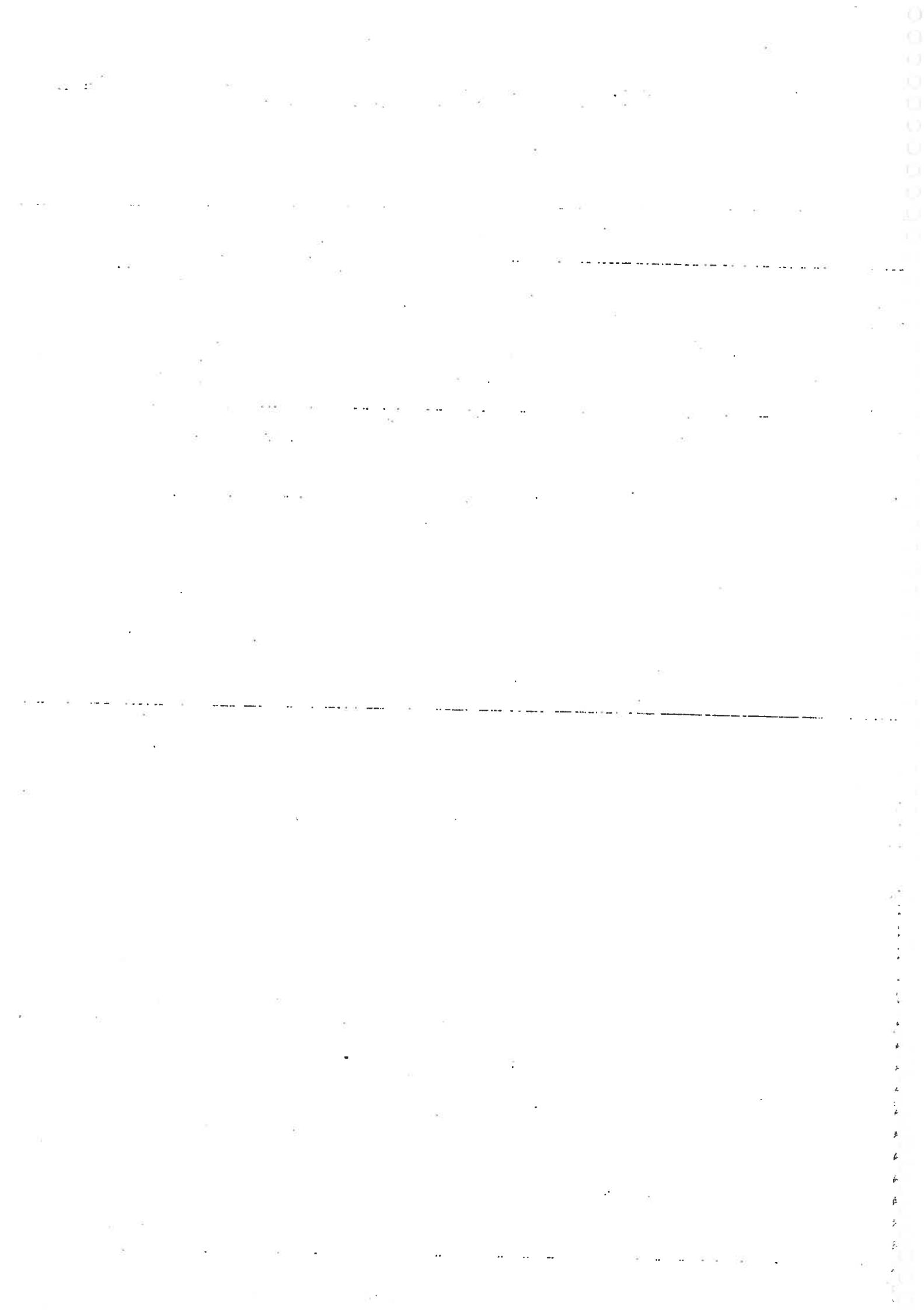
Normalizar  $R$  en FN4. ¿Se preservan las dependencias?

- 6) Sea el esquema  $R(A, B, C, D, E)$  con las dependencias:

$A \rightarrow\rightarrow BC; D \rightarrow C$

a) Demostrar que  $A \rightarrow C$ .

b) Normalizar  $R$  en FN4. ¿Se preservan las dependencias?



# BASES DE DATOS – GRADO EN INGENIERÍA INFORMÁTICA – ULL

## Práctica 1: Fundamentos procesador de consultas interactivo

Propósito:- Crear la cuenta de usuario de ORACLE y aprender a utilizar las principales sentencias del procesador de consultas interactivo SQL\*Plus incluido con el sistema gestor de base de datos ORACLE 11g.

La principal herramienta para utilizar SQL y PL/SQL es la interfaz de usuario de SQL\*Plus.

Este programa permite a los usuarios finales, desarrolladores y administradores llevar a cabo las siguientes operaciones:

- Escribir, editar y ejecutar comandos SQL y bloques PL/SQL.
- Dar formato a los resultados de las consultas.
- Visualizar los esquemas de las tablas y copiar datos entre bases de datos.
- Ejecutar comandos y operaciones de entrada/salida (introducción, presentación y visualización de variables).

La herramienta SQL\*Plus tiene su propio lenguaje: SQL\*Plus, el cual será objeto de estudio en esta práctica.

1.- Invocar al sqlplus usando la cuenta *administrador* de ORACLE con el password *administrador*.

sqlplus [nombreusuario[/contraseña]]

2.- Dar de alta el usuario correspondiente a cada alumno. La instrucción a ejecutar es:

```
CREATE USER alu#####
  IDENTIFIED BY password
  DEFAULT TABLESPACE USERS
  TEMPORARY TABLESPACE TEMP;
```

GRANT CONNECT, RESOURCE → Concede privilegios  
TO nombre\_usuario;

3.- Usando la instrucción CONNECT reconectarse al sistema con el usuario creado.

4.- Utilizando la Guía del Usuario y Referencia del SQL\*Plus estudiar la descripción, hacer un resumen y realizar pruebas con, al menos, las siguientes sentencias (empleando diversas opciones y/o parámetros cuando sea posible):

Gestión sesión

CONNECT [nombreusuario[/contraseña]] [@cadena\_de\_conexion] → Crea una conexión con  
PASSWORD [nombreusuario] → Cambiar el password. → La base de datos.

DISCONNECT

EXIT [SUCCESS/FAILURE/WARNING/n/variable] [COMMIT/ROLLBACK] → Acaba borrando los  
→ Sale del programa datos.

Gestión buffer de instrucciones

L[IST] [n|n m|n LAST] \*| \* n| m \*|\* LAST| LAST] → Muestra el contenido del buffer.

I[INPUT] [texto] → Añade texto \* → Línea activa.

A[PPEND] [texto] → Texto al final de una linea. L1 → activa linea 1.

C[CHANGE] /texto\_antiguo [/texto\_nuevo]

DEL [n|n m|n LAST] \*| \* n| m \*|\* LAST| LAST]

S[AVE] nombre\_archivo [CREATE|REPLACE|APPEND]

Guarda el contenido del buffer  
en un fichero.

→ Por defecto, acaba guardando los cambios. Igual que EXIT.

\* → Línea activa.

L1 → activa linea 1.

## BASES DE DATOS - GRADO EN INGENIERÍA INFORMÁTICA - ULL

G[ET] nombre\_archivo [LIST|NOLIST] → Carga en buffer contenido fichero.  
RUN → Ejecuta las sentencias almacenadas en el buffer → lo ejecuta.

### Gestión de scripts

ED[T] [nombre\_archivo]

REM[RK] texto

- texto

/\* texto \*/

STA[RT] nombre\_archivo

@nombre\_archivo → ejecuta todas las secuencias de un fichero, sin esperar.

@@nombre\_archivo

COPY {FROM base|TO base|FROM base TO base} {APPEND|CREATE|INSERT|REPLACE}  
table\_destino[(columna, ...)] USING consulta

### Gestión del entorno SQL\*Plus

DESCRIBE table

SPO[OL] [nombre\_archivo|OFF|OUT]

SHO[W] {ALL|parámetro}

SET parámetro valor

HELP [sentencia]

HO[ST] [sentencia SQL]

### Gestión de variables

DEF[N] [variable = texto]

UNDEF[N] variable

ACCEPT variable [CHAR|NUM] [PROMPT texto] [HIDE]

&variable

&&variable

VAR[ABLE] [variable [NUMBER|CHAR(n)]]

PRINT variable

### Presentación de resultados

PAUSE [texto]

PROMPT [texto]

TTITLE [opción [texto|variable]|OFF|ON]

BTITLE [opción [texto|variable]|OFF|ON]

COLUMN {columna|expresión} opción

SPOOL [ ] → Genera fichero.  
SPOOL OFF → Cierra el fichero y los buffers.

SPOOL OUT →

SPOOL → Comprueba el estado del spool

→ Manual → Capítulo 12.  
(sentencias!)

## BASES DE DATOS – GRADO EN INGENIERÍA INFORMÁTICA – ULL

### Práctica 2: Análisis de la Base de Datos de Gestión Docente

Propósito: Describir el modelo conceptual y la especificación de requisitos de la base de datos que se usará en las prácticas de laboratorio de la asignatura.

Para realizar la gestión docente departamental (plan docente) en la Universidad de La Laguna se va a utilizar una base de datos relacional que atiende al siguiente esquema:

Los atributos que se utilizan los vamos a abreviar de la siguiente manera:

Atributo	Significado	Entidad	Atributo
A	Asignatura		
AR	Área de conocimiento		
CAR	Código de área de conocimiento		
CAS	Código de Asignatura		
CAT	Categoría: TU, IEU, CU, ...		
CD	Código de Departamento		
CL	Créditos de Laboratorio		
CLA	Créditos de Laboratorio Asignados		
CP	Créditos Prácticos		
CPA	Créditos Prácticos Asignados		
CT	Créditos Teóricos		
CTA	Créditos Teóricos Asignados		
CUR	Curso: 1, 2, 3, ...		
D	Departamento		
DNI	D.N.I. del alumno: 1111, 2222		
FF	Fecha de Finalización Docencia		
FI	Fecha de Inicio Docencia		
P	Profesor: Juan, Pedro, ...		
T	Titulación: GI, GF, GM, MII		

El esquema de la base de datos consta de cinco tablas:

DEPARTAMENTO(CD, D)

SIGNIFICADO: El departamento con código CD se denomina con el nombre D.  
CLAVE PRIMARIA: (CD)

AREA(CAR, AR, CD)

SIGNIFICADO: El área de conocimiento con nombre AR tiene como código CAR y pertenece al departamento CD.

CLAVE PRIMARIA: (CAR)

CLAVES AJENAS: (CD)

PROFESOR(DNI, P, CAR, CAT)

SIGNIFICADO: El profesor con D.N.I. DNI se llama P, tiene categoría CAT y está adscrito al área de conocimiento CAR del departamento CD.

CLAVE PRIMARIA: (DNI)

CLAVES AJENAS: (CAR)

## BASES DE DATOS - GRADO EN INGENIERÍA INFORMÁTICA - ULL

CP      CA

**ASIGNATURA (CAS, A, T, CUR, CAR, CT, CP, CL)**

**SIGNIFICADO:** La asignatura con código CAS, tiene por nombre A, se imparte en el curso CUR de la titulación T, pertenece al área de conocimiento CAR y tiene CT/CP/CL créditos teóricos/prácticos/laboratorios.

**CLAVE PRIMARIA:** (CAS)

**CLAVES AJENAS:** (CAR)

**PLAN DOCENTE (DNI, CAS, CTA, CPA, CLA, FI, FF)** → Un profe imparte en varios cursos distintos

**SIGNIFICADO:** Al profesor con DNI, DNI tiene asignados, desde la fecha FI hasta la fecha FF, en la asignatura CAS, CTA/CPA/CLA créditos teóricos/prácticos/laboratorios. Si el profesor está impartiendo actualmente la asignatura aparece en fecha de finalización un NULL.

**CLAVE PRIMARIA:** (DNI, CAS, FI)

**CLAVES AJENAS:** (CAS), (DNI)

La información inicial que contendrán estas tablas es la siguiente:

### DEPARTAMENTO

CD	D
1	ANALISIS MATEMATICO
2	ASTROFISICA
3	ESTADISTICA, INVESTIGACION OPERATIVA Y COMPUTACION
4	MATEMATICA FUNDAMENTAL

### ÁREA

CAR	AR	CD
1	ALGEBRA	4
2	ANALISIS MATEMATICO	1
3	ASTRONOMIA Y ASTROFISICA	2
4	CIENCIAS DE LA COMPUTACION E INTELIGENCIA ARTIFICIAL	3
5	DIDACTICA DE LA MATEMATICA	1
6	ESTADISTICA E INVESTIGACION OPERATIVA	3
7	LENGUAJES Y SISTEMAS INFORMATICOS	3
8	MATEMATICA APLICADA	1

### PROFESOR

DNI	P	CAR	CAT
1111	JUAN	6	CU
2222	CARLOS	7	TU
3333	PEDRO	4	TEU
4444	MARIA	7	TU
5555	IVAN	1	CEU
6666	CARMEN	3	CD
7777	MARIO	2	TEU
8888	FRANCISCO	5	TU
9999	ANGELA	8	TEU
1010	DAVID	4	TU
2020	SOLEDAD	7	CU
3030	JOSE MANUEL	6	TEU

"Pro. impartido" = TODAS

**BASES DE DATOS – GRADO EN INGENIERÍA INFORMATICA – ULL**

**ASIGNATURA**

CAS	A	T	CUR	CAR	CT	CP	CL
1	BASES DE DATOS	GF	3	7	3	1.5	1.5
2	INTELIGENCIA ARTIFICIAL	GI	3	4	1.5	1.5	3
3	ALMACENES DE DATOS	MI	1	7	1.5	0	1.5
4	MINERIA DE DATOS	MI	1	7	1.5	0	1.5
5	INFORMATICA BÁSICA	GI	1	7	3	1.5	1.5
6	ALGEBRA	GI	1	1	3	3	0
7	CAÉCULO	GI	1	8	3	3	0
8	OPTIMIZACIÓN	GI	1	6	3	1.5	1.5
9	GESTIÓN DE RIESGOS	GI	3	4	3	0	3
10	ASTRONOMÍA	GF	2	3	3	1.5	1.5
11	ENSEÑANZA DE LA MATEMÁTICA	GM	2	5	6	0	0
12	ANÁLISIS COMPLEJO	GM	4	2	4.5	3	0

**PLAN DOCENTE**

DNI	CAS	CTA	CPA	CLA	FI	FF
4444	1	3	1.5	1.5	01-09-11	
4444	4	1.5	0	1.5	01-09-08	31-08-10
4444	5	3	0	0	01-09-10	
1111	8	3	1.5	1.5	01-09-07	31-08-09
1111	8	3	0	0	01-09-09	
3030	8	0	1.5	1.5	01-09-09	
2222	4	1.5	0	1.5	01-09-09	
2222	3	1.5	0	1.5	01-09-06	31-08-07
1010	2	1.5	1.5	3	01-09-05	31-08-08
3333	2	1.5	1.5	3	01-09-08	
1010	9	3	0	3	01-09-08	31-08-09
1010	9	1.5	0	1.5	01-09-09	
9999	7	3	0	0	01-09-10	
5555	6	3	0	0	31-03-10	
6666	10	3	1.5	1.5	01-09-08	31-08-11
8888	11	6	0	0	01-09-09	
2020	3	1.5	0	1.5	01-09-08	
7777	12	4.5	3	0	01-09-10	
3333	9	1.5	0	1.5	01-09-09	

CP-CAT CP

Se pide: CA

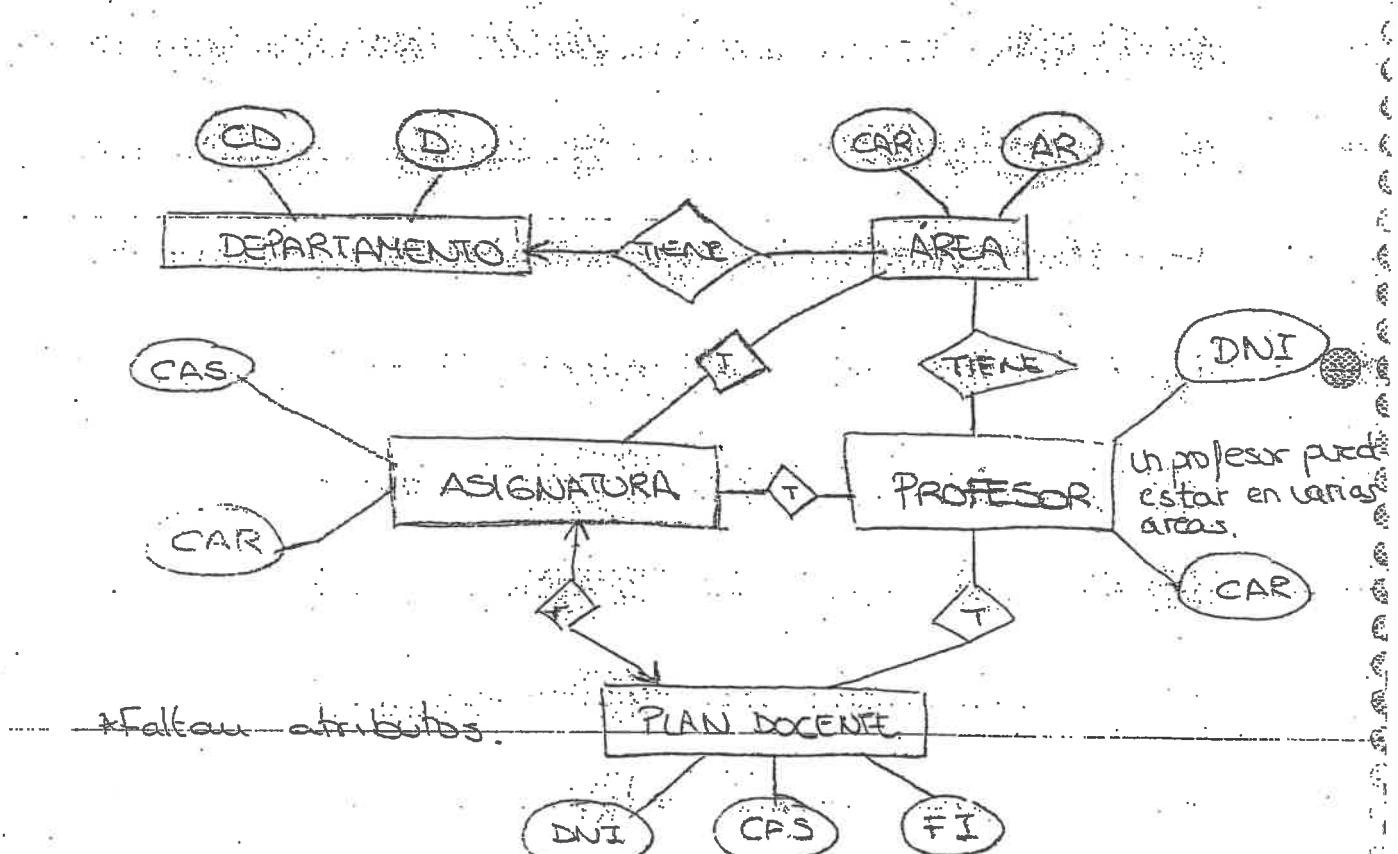
- Leer detenidamente el enunciado de la práctica hasta comprender el significado de los atributos y el sentido de las relaciones expresadas en las tablas, sabiendo interpretar la instancia (contenido) de la base de datos dada.
- Enumerar y justificar las implicaciones derivadas de las definiciones de claves primarias y ajena realizadas en el diseño.
- Representar el diagrama de jerarquía referencial de este diseño y especificar un posible orden de creación/borrado de objetos.
- Describir posibles políticas de mantenimiento de la integridad referencial (acciones compensatorias) para esta base de datos.

Datos correctos en el momento



## BASES DE DATOS – GRADO EN INGENIERÍA INFORMÁTICA – ULL

- 5.- Representar gráficamente el esquema conceptual de la base de datos utilizando el modelo E/R.
- 6.- Enuncia y justifica diferentes condiciones de integridad generales (excluyendo las asociadas a claves primarias y ajenas) que, a tu juicio, deben satisfacerse en esta base de datos.



IPZ

2

DEPARTAMENTO → ÁREA → PROFESOR → ASIGNATURA → PLAN-DOCENTE.

→ Cada depart. tiene un código y un nombre. Almacena la inf. del dep.  
Nº estatístico

ÁREA → Área de conocimiento, pertenece a un departamento.  
La clave primaria es código de área (CAR), sólo está asig a  
un único departamento.

CD → Clave ajena. → Tiene q existir el depart antes q el área.  
Nº estatístico

PROFESOR

→ DNI → Clave primaria.

Nº estatístico.

ASIGNATURA → CAS (Identifica la asignatura).

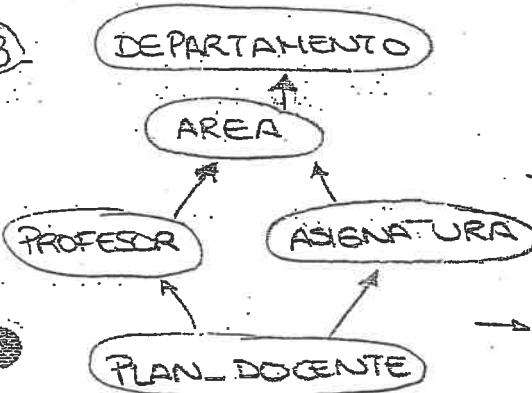
Una asignatura sólo puede pertenecer a un área.

Nº estatístico.

PLAN-DOCENTE → Asignación de asignaturas x a un determinado profesor.

Tabla grande. Tasa de crecimiento constante. Siempre se introducen datos.

②. ¿Qué implica q la clave primaria sea la que es.



→ Dos tablas hijas para el mismo padre.

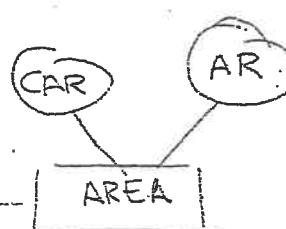
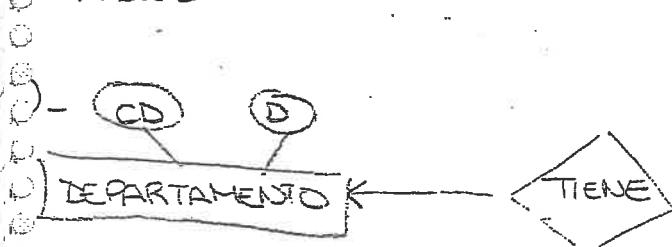
→ Un hijo con 2 padres.

4 niveles. El orden de borrado es inverso.

Carga de datos → Desde el primer nivel.

○ P.e. = Claudio borre el departamento, en cascada se borrará la tabla área.

Analizarlo de dos en dos.



se relacionan a través  
de las claves primaria.

Uno - muchos

un departamento → varias áreas

⑥  $\rightarrow$  FI anterior a FF  $\rightarrow$   $F_1 \leq F_F$

$CFA \leq CT \rightarrow$  A un prof no tienen que asignarle todos los creditos de una asignatura.

- Qx un prof no de + de 5 asig.

- Qx una asig no tenga + de 4 prof.

$CFA \leq CP$

$CLA \leq CL$

### PRACTICA 03 :

```
CREATE TABLE nombre_tabla DEPARTAMENTO  
(CD NUMBER(2) PRIMARY KEY,  
D VARCHAR2(35));
```

Lower  $\rightarrow$  Minusculas

Upper  $\rightarrow$  Mayusculas.

exp IS [NOT] NULL

FF IS NULL  $\rightarrow$  En el where

Capítulo 10  $\rightarrow$  Sentencias SQL

COMMIT WORK → Guardar

EXIT → Sale del programa, correctamente.

## BASES DE DATOS - GRADO EN INGENIERÍA INFORMÁTICA - ULL

### Práctica 3: Creación y carga de la base de datos

Propósito: Crear y cargar las tablas de datos usadas en la asignatura utilizando el sistema gestor de base de datos ORACLE 11g.

- 1.- Conéctate a tu cuenta de ORACLE a través del sqlplus.
- 2.- Crear un fichero de SPOOL denominado PRACT3.
- 3.- Crear la tabla DEPARTAMENTO como se indica a continuación:

```
CREATE TABLE DEPARTAMENTO  
(CD NUMBER(2) PRIMARY KEY,  
D VARCHAR2(60));
```

- 4.- Crear la tabla ÁREA como se indica a continuación:

```
CREATE TABLE ÁREA  
(CAR NUMBER(3) PRIMARY KEY,  
AR VARCHAR2(60),  
CD NUMBER(2) REFERENCES DEPARTAMENTO ON DELETE CASCADE);
```

- 5.- Crear la tabla PROFESOR como se indica a continuación:

```
CREATE TABLE PROFESOR  
(DNI NUMBER(8) PRIMARY KEY,  
P VARCHAR2(60),  
CAR NUMBER(3),  
CAT VARCHAR2(5),  
FOREIGN KEY (CAR) REFERENCES ÁREA ON DELETE SET NULL);
```

- 6.- Crear la tabla ASIGNATURA como se indica a continuación:

```
CREATE TABLE ASIGNATURA  
(CAS NUMBER(3) PRIMARY KEY,  
A VARCHAR2(50) NOT NULL,  
T CHAR(4) NOT NULL,  
CUR NUMBER(1) CHECK (CUR BETWEEN 1 AND 5),  
CAR NUMBER(3) REFERENCES ÁREA ON DELETE SET NULL,  
CT NUMBER(3,1) DEFAULT 0.0,  
CP NUMBER(3,1) DEFAULT 0.0,  
CL NUMBER(3,1) DEFAULT 0.0);
```

- 7.- Crear la tabla PLAN\_DOCENTE como se indica a continuación:

```
CREATE TABLE PLAN_DOCENTE  
(DNI NUMBER(8),  
CAS NUMBER(3),  
CTA NUMBER(3,1) DEFAULT 0.0,  
CPA NUMBER(3,1) DEFAULT 0.0,  
CLA NUMBER(3,1) DEFAULT 0.0,  
FI DATE DEFAULT SYSDATE,  
FF DATE,  
PRIMARY KEY (DNI, CAS, FI),  
FOREIGN KEY (CAS) REFERENCES ASIGNATURA ON DELETE CASCADE,  
FOREIGN KEY (DNI) REFERENCES PROFESOR ON DELETE CASCADE,  
CHECK (FF >= FI));
```

## BASES DE DATOS - GRADO EN INGENIERÍA INFORMÁTICA - ULL

8.- Razonar el motivo de las definiciones hechas. Poner especial atención en las claves primarias, las claves ajena, las acciones compensatorias de integridad, las restricciones de dominios (valores no nulos, condiciones check, ...)

9.- Insertar en las tablas creadas la información suministrada en la guía de prácticas. Observa los ejemplos dados a continuación:

Para la tabla DEPARTAMENTO:

```
INSERT INTO DEPARTAMENTO  
VALUES (1, 'ANALISIS MATEMÁTICO'); sin fide!  
INSERT INTO DEPARTAMENTO  
VALUES (2, 'ASTROFÍSICA');  
INSERT INTO DEPARTAMENTO  
VALUES (3, 'ESTADÍSTICA, INVESTIGACIÓN OPERATIVA Y COMPUTACIÓN');  
INSERT INTO DEPARTAMENTO  
VALUES (4, 'MATEMÁTICA FUNDAMENTAL');
```

Para la tabla AREA:

```
INSERT INTO AREA  
VALUES (1, 'ÁLGEBRA', 4);
```

Para la tabla PROFESOR:

```
INSERT INTO PROFESOR  
VALUES (1111, 'JUAN', 6, 'CU');
```

Para la tabla ASIGNATURA:

```
INSERT INTO ASIGNATURA  
VALUES (1, 'BASES DE DATOS', 'GII', 3, 7, 3, 1.5, 1.5);
```

Para la tabla PLAN\_DOCENTE:

```
INSERT INTO PLAN_DOCENTE  
VALUES (4444, 1, 3, 1.5, 1.5, '01-SEP-11', NULL);
```

To\_DATE  
TO\_CHAR

10.- Listar los contenidos de cada una de las tablas.

Ejemplo:

```
SELECT CD, D  
FROM DEPARTAMENTO;
```

11.- Cierra el fichero de SPOOL PRACT3.

12.- Desde el sqlplus comprueba el contenido del fichero de SPOOL PRACT3.

COL P FORMAT A15

ALTER TABLE

→ Cambiar el formato de  
una tabla, por defecto lo  
pone con 80 caracteres.

SELECT \*  
FROM DEPARTAMENTO;

DESCRIBE DEPARTAMENTO

EDIT PRACT3

## BASES DE DATOS - GRADO EN INGENIERÍA INFORMÁTICA - ULL

### Práctica 5 Consultas con múltiples tablas

Propósito: Realización de consultas utilizando varias tablas a través de funciones naturales y/o subconsultas.

1. Listar los nombres de asignaturas adscritas a áreas cuyo nombre empiece por 'A'.
2. Listar los nombres de asignaturas adscritas a áreas cuyo nombre termine en 'A'.
3. Listar los nombres de asignaturas que lleven la palabra 'DATOS'.
4. Listar los DNI de los profesores en cuyo nombre el tercer carácter sea 'R'.
5. Listar, sin contar duplicados, los DNI de los profesores con nombres de, a lo sumo, 5 caracteres de longitud.
6. Listar, sin contar duplicados, los DNI de los profesores con nombres de, al menos, 5 caracteres de longitud.
7. Listar los nombres de los profesores que actualmente imparten alguna asignatura.
8. Nombres de los profesores que han impartido la asignatura con código 8.
9. Listar las t-uplas de la tabla PLAN\_DOCENTE ordenadas de forma ascendente, según el campo FF.
10. Listar las t-uplas de la tabla PLAN\_DOCENTE ordenadas de forma descendente, según el campo FF.
11. Nombres de los profesores que han impartido la asignatura 'OPTIMIZACIÓN' en la titulación GI. Ordena los nombres ascendente mente.
12. Listar los nombres de los profesores del departamento 'MATEMÁTICA FUNDAMENTAL'. Ordena los nombres descendente mente.
13. Listar los nombres de las asignaturas impartidas por el profesor con DNI 1010.
14. Listar los nombres de las asignaturas impartidas por el profesor con nombre 'DAVID'.
15. Listar los nombres de las áreas adscritas al departamento 'ESTADÍSTICA, INVESTIGACIÓN OPERATIVA Y COMPUTACIÓN'.
16. Listar los nombres de las asignaturas impartidas actualmente por catedráticos de universidad (categoría CU).
17. Listar los nombres de las asignaturas que siempre han sido impartidas por catedráticos de universidad (categoría CU).
18. Listar los nombres de asignaturas adscritas a 'LENGUAJES Y SISTEMAS INFORMÁTICOS' o a 'CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL'.

\* Longitud de una cadena de caracteres. → Puedo

## BASES DE DATOS - GRADO EN INGENIERÍA INFORMÁTICA - ULL

19. Listar los nombres de profesores que actualmente dan clases en las titulaciones 'GII' o en 'MII'.
20. Listar los nombres de profesores que actualmente dan clases en las titulaciones 'GII' y en 'MII' simultáneamente.
21. Listar los nombres de profesores que actualmente no imparten ninguna asignatura.
22. Listar los nombres de asignaturas impartidas en la titulación GII.
23. Listar los nombres de las áreas de conocimiento y los nombres de las asignaturas que pertenecen a ellos.
24. Listar los nombres de departamentos y los nombres de las áreas adscritas a ellos. Ambos campos deben estar ordenados de forma alfabética.
25. Listar los nombres de departamentos y los profesores de cada uno de ellos. Ambos campos deben estar ordenados de forma alfabética.

LIKE → Comparar cadena de caracteres con un patrón.

exp [NOT] LIKE patrón → Devuelve True si se empareja.

{ % → cualquier cadena  
\_ → cualquier carácter.

AR LIKE 'A%' → Empieza por A / '%A' → Termina en A.

UPPER → Pasa a mayúsculas. '%A%' → A en medio.

ORDER BY → Última de la consulta. exp [ASC/DESC] [, ...]

criterios

[ UNION  
INTERSECT  
MINUS EXCEPT ]

## Práctica: Ampliación de Funciones

1. Buscar información y probar las siguientes funciones numéricas de SQL:

SIN, COS, TAN, SQRT, ABS, CEIL, FLOOR, ROUND, SIGN, POWER, EXP, MOD, LOG, TRUNC, ROUND

2. Buscar información y probar las siguientes funciones de cadenas de caracteres de SQL:

CONCAT, INITCAP, UPPER, LOWER, LENGTH, REPLACE, TRIM, LTRIM, RTRIM, LPAD, RPAD, SUBSTR, TO\_CHAR

3. Buscar información y probar las siguientes funciones de manejo de fechas de SQL:

ADD\_MONTHS, MONTHS\_BETWEEN, SYSDATE, NEXT\_DAY, LAST\_DAY, CURRENT\_DATE, TO\_DATE

SQRT (n) → Raíz Cuadrada.

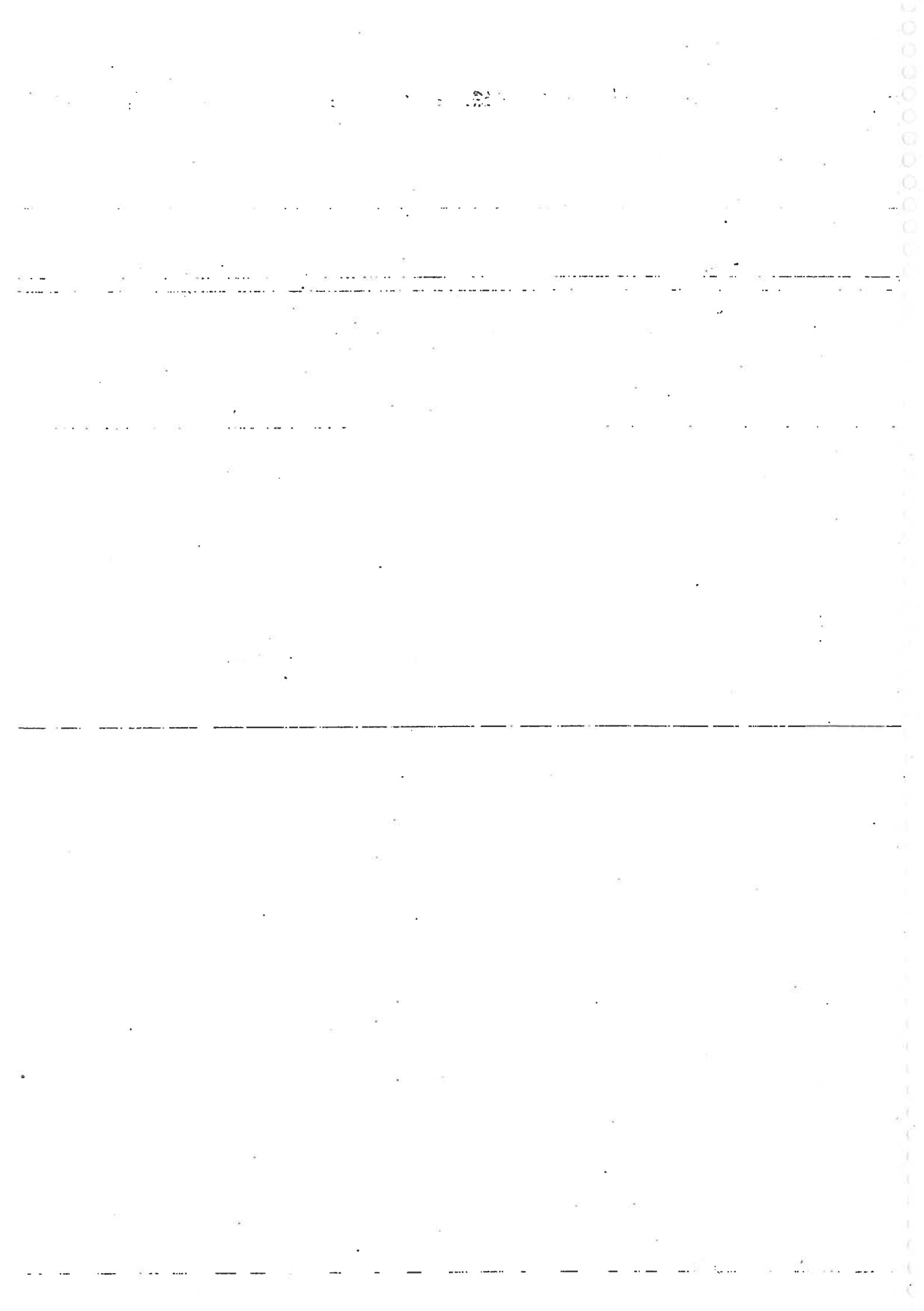
```
SELECT SQRT(64)  
FROM DUAL;
```

```
SELECT CAS, SQRT(CT+CP+CL)  
FROM ASIGNATURA;
```

```
SELECT A, LENGTH(A)  
FROM ASIGNATURA;
```

TO\_CHAR → <sup>Transforma</sup> ~~dato~~ a cadena de caracteres.

F Trigonometricas → radianes.





**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**21 de Mayo de 2014**

Un conjunto de exámenes de tipo test tiene clasificadas sus preguntas en 2 categorías distintas. Para aprobar un examen el alumno no puede cometer, en cada categoría, un número de errores superior a una cantidad estipulada previamente según la clase de examen. En el esquema de base de datos utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
CE	Clase de examen: C1, C2, ...
DNI	DNI de la persona que realiza el examen
FE	Fecha de celebración del examen
IE	Identificador del examen: E1, E2, ...
ME1	Máximo número de errores que se permite en la categoría 1
ME2	Máximo número de errores que se permite en la categoría 2
NE1	Número de errores cometidos en la categoría 1
NE2	Número de errores cometidos en la categoría 2

Clase (ce, ME1, ME2)  
Exámenes (IE, CE, FE)

Las tablas utilizadas son:

CLASE(CE, ME1, ME2)

SIGNIFICADO: Los exámenes de clase CE requieren que para aprobar el examen se cometan ME1 y ME2 errores en las categorías 1 y 2, respectivamente.

CLAVE PRIMARIA: (CE)

EXÁMENES (IE, CE, FE)

SIGNIFICADO: El examen IE es de la clase CE y se celebra en la fecha FE

CLAVE PRIMARIA: (IE)

RESULTADOS (DNI, IE, NE1, NE2)

SIGNIFICADO: La persona con dni DNI se presentó al examen IE y cometió NE1 y NE2 errores en las categorías 1 y 2, respectivamente.

CLAVE PRIMARIA: (DNI, IE)

Resultados (DNI, IE, NE1, NE2)

Resultados (DNI, IE, NE1, NE2)

1) Responder en álgebra relacional a las siguientes consultas:

- Personas que se han presentado a alguno de los exámenes realizados por la persona 1111.
- Personas que sólo han aprobado exámenes de clase C1.
- Personas que han aprobado al menos las mismas clases de exámenes que la persona 1111.
- Personas que se han presentado al menos a los mismos exámenes que alguna otra.

2) Responder en cálculo relacional de dominios a las siguientes consultas:

- Personas que no se han presentado a exámenes de la clase C1.
- Personas que se han presentado al menos una vez a cada clase de examen.
- Personas que han aprobado todos los exámenes a los que se han presentado.
- Persona que cometió el menor número de errores totales entre los que aprobaron el examen E1.

3) Responder en SQL a las siguientes consultas:

- Personas que se han presentado al último examen de la clase C1.
- Número medio de personas presentadas en los exámenes de la clase C1.
- Examen de la clase C1 al que se han presentado más personas.
- Exámenes que han sido aprobados por al menos el 70% de las personas presentadas.

4) Responder en SQL a las siguientes peticiones:

- Añade a la tabla RESULTADOS un atributo de calificación CAL que sólo permita los valores nulo, 'APTO' y 'NO APTO'.
- Actualiza el campo CAL de la tabla RESULTADOS con 'APTO' si la persona aprobó.
- Añade una condición de integridad que impida que una persona pueda ser calificado como APTO si ha cometido en total más de 20 errores en el examen.
- Crea una función PL/SQL que nos devuelva el número total de veces que se ha presentado una persona específica a exámenes de una determinada clase.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {AC → D, E → A, B → AC, D → BE}.

- Calcula todas las claves. ¿Está R en FNBC? ¿Y en 3FN? Razona la respuesta.
- Calcular CE+ aplicando el algoritmo de Ullman.
- Probar usando los axiomas de Armstrong que EC → B.
- Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**18 de Enero de 2014**

La Internet Movie Database (IMDb, Base de datos de películas en Internet) es una base de datos en línea que proporciona información relacionada con películas: directores, actores, presupuestos, ..., etc. En el esquema de base de datos utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
A	Año de recaudación: '2013', '2014', ...
ACT	Actor/Actriz Bullock, Clooney, DiCaprio, Eastwood, ...
CNT	Cantidad recaudada: 150.000.000 Dólares, ...
COM	Compañía: Warner Bros., Universal Pictures, ...
DIR	Director: Cuarón, Scorsese, ...
FE	Fecha de estreno: '04-10-2013', '17-01-2014', ...
GEN	Género: Thriller, drama, ciencia ficción,
PAÍS	País de recaudación: EEUU, España, ...
PRE	Presupuesto de la película: 120.000.000 Dólares
ROL	Role: Protagonista, actor secundario, extra, ...
TIT	Título de la película: Gravity, 'El lobo de Wall Street', ...

Las tablas utilizadas son:

PELÍCULA(TIT, DIR, FE, COM, PRE)

SIGNIFICADO: La película con título TIT es del director DIR, fue estrenada un presupuesto de PRE Dólares.

CLAVE PRIMARIA: (TIT)

TIPO(TIT, GEN)

SIGNIFICADO: La película con título TIT pertenece al género GEN.

CLAVE PRIMARIA: (TIT, GEN)

REPARTO(TIT, ACT, ROL)

SIGNIFICADO: El actor/actriz ACT participa en la película TIT desempeñando el papel ROL.

CLAVE PRIMARIA: (TIT, ACT)

RECAUDACIÓN (TIT, PAÍS, A, CNT)

SIGNIFICADO: La película TIT ha recaudado en el país PAÍS, durante el año A, la cantidad CNT Dólares.

CLAVE PRIMARIA: (TIT, PAÍS, A)

1) Responder en álgebra relacional a las siguientes consultas:

- a) Directores que han trabajado con las compañías Warner y Universal.
- b) Actores que han protagonizado películas de todos los géneros.
- c) Actores que sólo participan en películas del género drama.
- d) Directores que han trabajado en al menos las mismas compañías que algún otro director.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Actores que han participado en al menos dos películas.
- b) Directores que siempre dirigen películas para una misma compañía.
- c) Película con mayor presupuesto de la Warner.
- d) Películas que han recaudado en EEUU, cada año, al menos el presupuesto de dicha película.

3) Responder en SQL a las siguientes consultas:

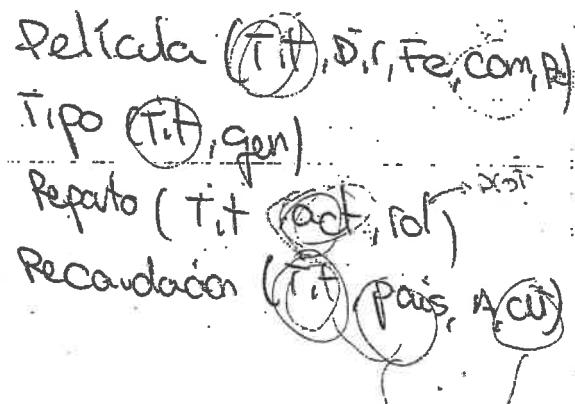
- a) Protagonistas de películas estrenadas en 2013.
- b) Directores que en 2013 sólo han dirigido una única película.
- c) Día en que se han estrenado más películas de suspense.
- d) Películas que han logrado recaudar en EEUU más del doble de su presupuesto.

4) Responder en SQL a las siguientes peticiones:

- a) Crea una vista que nos muestre para cada película su presupuesto y el total recaudado.
- b) Incrementa en un 10% la recaudación de este año, en España, de las películas de la Warner.
- c) Impide que el presupuesto de una película se incremente en más de un 10% respecto a la anterior película de esa compañía.
- d) Crea una función PL/SQL que nos devuelva el total recaudado en un país para una película dada.

5) Sea  $R(A, B, C, D, E)$  una relación con dependencias funcionales  $DF = \{E \rightarrow AD, C \rightarrow E, BD \rightarrow AC, A \rightarrow B\}$ .

- a) Calcula todas las claves. ¿Está  $R$  en FNBC? ¿Y en 3FN? Razona la respuesta.
- b) Calcular  $AD+$  aplicando el algoritmo de Ullman.
- c) Calcular un recubrimiento minimal para  $DF$ . Razona la respuesta.
- d) Descomponer  $R$  en FNBC. ¿Se preservan las dependencias funcionales?



1) Responder en álgebra relacional a las siguientes consultas.

a. Directores que han trabajado con las compañías Warner y Universal.

$P(DIR) \sqcap S(COM = 'Warner') \wedge (COM = 'Universal') (PELICULA)$

\*  $P(DIR) \sqcap S(COM = 'Warner') (PELICULA) = A$  } ANB  
 $P(DIR) \sqcap S(COM = 'Universal') (PELICULA) = B$  }

b. Actores que han protagonizado películas de todos los géneros.

$A = P(ACT, GEN) \sqcap S(ROL = 'Protagonista') (REPARTO * TIPO)$

$B = P(GEN) (TIPO)$

{  $t_0 / (\exists r) (t[ACT] = r[ACT]) \wedge (r[ROL] = "protagonista")$  }

Para cuales géneros tiene que haber una  
película de ese género

c. Actores que sólo participan en películas de género drama, que el actor  
es protagonista.

$P(ACT) \sqcap (GEN \neq 'Drama') (REPARTO * TIPO) = A$

\*  $P(ACT) (REPARTO * TIPO) - P(ACT)(A)$

d. Directores que han trabajado en al menos las mismas compañías que algún  
otro director.

$P(DIR, COM) (PELICULA) = A \Rightarrow R \subseteq B$

$P(DIR) (PELICULA) - (P(A.DIR) \sqcap S(A.DIR \neq B.DIR) \wedge (A.COM \neq B.COM) (A \times B))$

{  $t_0 / (\exists p) (t[DIR] = p[DIR]) \wedge$   
 $\neg (\exists p_1) (p[DIR] \neq p_1[DIR]) \wedge (p[COM] \neq p_1[COM])$

2) Responder en cálculo relacional de tuplas a las siguientes consultas:

a. Actores que han participado en al menos dos películas.

$\text{dom}(r) = \text{dom}(r_1) = \text{REPARTO}$

$\{ t_{12} / (\exists r)(t[\text{ACT}] = r[\text{ACT}]) \wedge$   
 $\quad (\exists r_1)(r[\text{ACT}] = r_1[\text{ACT}]) \wedge (r[\text{TIT}] \neq r_1[\text{TIT}]) \}$

b. Directores que siempre dirigen películas para una misma compañía.

$\text{dom}(p) = \text{dom}(p_1) = \text{PELICULA}$

$\{ t_{12} / (\exists p)(t[\text{EDIR}] = p[\text{EDIR}]) \wedge$   
 $\quad (\exists p_1)(p[\text{DIR}] = p_1[\text{DIR}]) \wedge (p[\text{COM}] \neq p_1[\text{COM}]) \}$

c. Película con mayor presupuesto de la Warner.

$\text{dom}(p) = \text{dom}(p_1) = \text{PELICULA}$

$\{ t_{12} / (\exists p)(t[\text{TIT}] = p[\text{TIT}]) \wedge (p[\text{COM}] = 'Warner') \wedge$   
 $\quad (\exists p_1)(p[\text{PRE}] = p_1[\text{PRE}]) \wedge (p_1[\text{PRE}] > p[\text{PRE}]) \}$

d. Películas que han recaudado en EEUU, cada año, al menos el presupuesto de la película.

$\text{dom}(p) = \text{PELICULA}$

$\text{dom}(r) = \text{RECAUDACION} = \text{dom}(r_1)$

$\{ t_{12} / (\exists r)(t[\text{TIT}] = r[\text{TIT}]) \wedge (r[\text{PAIS}] = 'EEUU')$   
 $\quad (\exists p)(p[\text{TIT}] = r[\text{TIT}]) \wedge (p[\text{PRE}] \leq r[\text{CNT}]) \wedge$   
 $\quad (\exists r_1)(p[\text{TIT}] = r_1[\text{TIT}]) \wedge (r_1[\text{PAIS}] = 'EEUU') \wedge (r_1[\text{CNT}] \geq p[\text{PRE}]) \}$



Pelicula que por cada año.

$\wedge (r[A] \neq r_1[A])$

3) Responder en SQL a las siguientes consultas:

a) Protagonistas de películas estrenadas en 2013.

SELECT ACT

FROM REPARTO NATURAL JOIN PELICULA  
WHERE (FE = '2013')

AND (ROL = 'Protagonista')

b) Directores que en 2013 sólo han dirigido una única película.

SELECT DIR

FROM PELICULA  
WHERE (FE = '2013')

GROUP BY DIR

HAVING COUNT(TIT) = '1';

d) SELECT TIT

FROM PELICULA P

WHERE (2 \* P.FE) < (SELECT SUM(CNT)

FROM RECAUDACION R

WHERE (R.TIT = P.TIT)

AND (PAIS = 'EEUU'));

c) Día en que se han estrenado más películas de suspense.

SELECT FE

FROM PELICULA NATURAL JOIN TIPO

WHERE (GEN = 'SUSPENSE')

GROUP BY TIT FE

HAVING COUNT(TIT) >= ALL (SELECT COUNT(TIT) FROM TIPO

WHERE (GEN = 'SUSPENSE')

GROUP BY TIT);

d) Películas que han logrado recaudar en EEUU, más del doble de su presupuesto.

SELECT TIT

FROM RECAUDACION RI

WHERE (PAIS = 'EEUU')

AND TIT IN (SELECT TIT

FROM PELICULA

WHERE PAIS =

GROUP BY TIT

WHERE

(SELECT ~~999~~ SUM(CNT)

FROM RECAUDACION ~~RI~~

IN WHERE (RI.TIT = RI.TIT)

GROUP BY ~~TIT~~);

AND (PAIS = 'EEUU');

4) Responder en SQL a las siguientes peticiones:

a.- Crea una vista que nos muestre para cada película su presupuesto y el total recaudado.

CREATE VIEW D1

AS (SELECT TIT, PRE, SUM(CNT)

FROM PELICULA NATURAL JOIN RECAUDACION

GROUP BY TIT);

b.- Incrementa en un 10% la recaudación de este año, en España, de las películas de la Warner.

UPDATE RECAUDACION

SET CNT = CNT + (CNT \* 0.1)

WHERE TIT IN (SELECT TIT

FROM RECAUDACION NATURAL JOIN PELICULA

WHERE (PAIS = 'España') AND (A = '2014') AND (COM = 'Warner'));

c.- Impide que el presupuesto de una película se incremente en más de un 10% respecto a la anterior película de esa compañía.

CREATE ASSERTION AS1

CHECK NOT EXIST ( SELECT \* FROM PELICULA P

GROUP BY PRE

HAVING (PRE\*0.1) > ALL (SELECT PRE FROM PELICULA P1

(+PRE)

WHERE (P.COM = P1.COM)

AND (P.FE < P1.FE);

Calcular la fecha max (MAX(FE)) → la anterior a la + reciente.

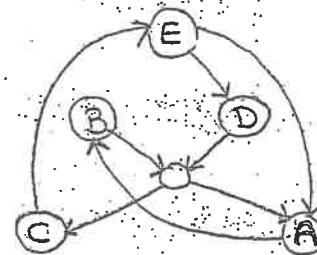
d.- Crea una función PL/SQL que nos devuelva el total recaudado en un país para una película dada.

5). Sea  $R(A, B, C, D, E)$  una relación con dependencias funcionales  
 $DF = \{E \rightarrow AD, C \rightarrow E, BD \rightarrow AC, A \rightarrow BD\}$

a. Calcula todas las claves. ¿Está  $R$  en FNBC? d) Y en 3FN? Razona la respuesta.

$$\left. \begin{array}{l} E^+ = \{E, A, D, B, C\} \\ C^+ = \{C, E, A, D, B\} \\ BD^+ = \{B, D, A, C, E\} \\ A^+ = \{A, B\} \end{array} \right\} \text{Si FNBC.}$$

claves: (E) (c) (BD) (AD)



- $R$  no está en FNBC porque la dependencia funcional  $(A \rightarrow B)$  el antecedente no es superclave.
- $R$  está en 3FN porque A y B forman partes de las claves y el resto está en FNBC por tanto también en 3FN.

b. Calcular  $AD^+$  aplicando el algoritmo de Ullman.

$$\begin{aligned} AD^+ &= \{A, D\} \quad (A \rightarrow B) \\ AD^+ &= \{A, D, B\} \quad (BD \rightarrow AC) \\ AD^+ &= \{A, D, B, C\} \quad (C \rightarrow E) \\ AD^+ &= \{A, D, B, C, E\} \end{aligned}$$

```

 $A^+ = A;$ 
while ( $A^+$  cambie)
  for each df  $X \rightarrow Y \in DF$ 
    if ( $(X \subseteq A^+)$ )  $\Rightarrow A^+ = A^+ \cup Y$ 
}
  
```

c. Calcular un recubrimiento minimal para  $DF$ . Razona la respuesta.

-  $(E \rightarrow AD)$  d) A es raro?

$$E \rightarrow D$$

$$E^+ = \{E, D\} \text{ No es raro.}$$

-  $(BD \rightarrow AC)$  d) A es raro?

$$BD \rightarrow C$$

$$BD^+ = \{B, D, C\}$$

d. Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?

R1 (A, B)

$$DF1 = \{ A \rightarrow B \}$$

Clave (A). Si FNBC.

R2 (A, C, D, E)

$$DF2 = \{ E \rightarrow AD, C \rightarrow E \}$$

$$E^+ = \{ E, A, D \}$$

$$C^+ = \{ C, E, A, D \} \rightarrow \text{Si FNBC}$$

Clave: (

R2.1 (A, D, E)

$$DF2.1 = \{ E \rightarrow AD \}$$

Clave (E)

- Se ha perdido la dependencia funcional ( $BD \rightarrow AC$ ).

APPs (CA, CAT)

Tienda (CA, MGR, RR)

Usuario (DN, CU, MGR)

Compra (CU, CA, MGR, FC)

X Compra

↓ compra & APP



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**9 de Enero de 2014**

La tienda oficial de aplicaciones (apps) de Apple (Apple Store) gestiona a través de una base de datos todas las descargas de apps que realizan sus usuarios. En el esquema de base de datos utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
CA	Código de la app: 1, 2, 3, ...
CAT	Categoría: productividad, juegos, educación, medicina, ...
CU	Código de usuario: 1, 2, 3, ...
DNI	DNI del usuario.
FC	Fecha de compra de la app: '01-10-2010', '26-07-2008'...
MER	Mercado: España, Estados Unidos, Argentina, ...
PR	Precio en Euros de una app en algún mercado. (Gratis = 0 Euros)

Las tablas utilizadas son:

APPS(CA, CAT)

TIENDA(CA, MER, PR)

USUARIO(DNI, CU, MER)

COMPRA(CU, CA, MER, FC)

APPS(CA, CAT)

SIGNIFICADO: La app con código CA pertenece a la categoría CAT.

CLAVE PRIMARIA: (CA)

TIENDA(CA, MER, PR)

SIGNIFICADO: La app con código CA está disponible en el mercado MER a un precio de PR euros.

CLAVE PRIMARIA: (CA, MER)

USUARIO(DNI, CU, MER)

SIGNIFICADO: La persona con dni DNI ha creado el usuario CU para el mercado MER.

CLAVE PRIMARIA: (CU)

COMPRA(CU, CA, MER, FC)

SIGNIFICADO: El usuario CU ha comprado en el mercado MER la app CA en la fecha FC.

CLAVE PRIMARIA: (CU, CA)

1) Responder en álgebra relacional a las siguientes consultas:

- a) Apps que sólo están disponibles en el mercado MI.
- b) App más cara del mercado MI.
- c) Usuarios que se ha descargado apps gratuitas de todas las categorías.
- d) Usuarios que se han descargado todas las apps de alguna categoría.

2) Responder en cálculo relacional de tuplas a las siguientes consultas:

- a) Personas que tienen al menos dos cuentas de usuario en un mismo mercado.
- b) Usuarios que siempre compran apps de una misma categoría.
- c) Mercados que han vendido apps de todas las categorías.
- d) Mercados que disponen al menos de las mismas apps que algún otro mercado.

3) Responder en SQL a las siguientes consultas:

- a) Usuarios que sólo han descargado apps gratuitas.
- b) Importe total medio de las apps descargadas en un mismo día en el mercado MI.
- c) Persona que ha descargado más apps en un mismo día.
- d) Mercados tales que al menos el 20% de las apps descargadas son de una misma categoría.

4) Responder en SQL a las siguientes peticiones:

- a) Implementa que si se elimina una app de la tabla APPS se propague el borrado en cascada a la tabla TIENDA.
- b) Elimina todas las apps del mercado MI que pertenezcan a la categoría de 'productividad'.
- c) Impide que una misma persona pueda crear usuarios en mercados distintos.
- d) Crea una función PL/SQL a la que se le pase un código de app y un mercado y nos devuelva su precio.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {D → B, BE → A, A → EC, C → D}.

- a) Calcula todas las claves. ¿Está R en FNBC? ¿Y en 3FN? Razóna la respuesta.
- b) Calcular CE+ aplicando el algoritmo de Ulman.
- c) Probar, usando los axiomas de Armstrong, que CE → A. Razóna la respuesta.
- d) Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?

1) Responder en álgebra relacional a las siguientes consultas:

a.- Apps que sólo están disponibles en el mercado M1.

$$\text{dom}(r) = \text{TIENDA}$$

$$\left\{ t[u] / (\exists r) (t[CA] = r[CA]) \wedge (r[HER] = '1') \wedge \right. \\ \left. \neg (\exists r_1) (r_1[CA] = r[CA]) \wedge (r_1[HER] \neq '1') \right\}$$

$$P(CA)(T1) - P(T1.CA) S (T1.HER \neq T2.HER) \wedge (T1.CA \neq T2.CA) (T1 \times T2)$$

$$T1 = T2 = \text{TIENDA}$$

$$\left\{ \langle ca \rangle / (\exists mer, pr) (\langle ca, '1', pr \rangle \in \text{TIENDA}) \wedge \right. \\ \left. \neg (\exists ca_1, mer_1, pr_1) (\langle ca_1, mer_1, pr_1 \rangle \in \text{TIENDA}) \wedge (ca = ca_1) \wedge (mer \neq mer_1) \right\}$$

b.- App más cara del mercado M1:

$$\text{dom}(r) = \text{dom}(r_1) = \text{TIENDA}$$

$$\left\{ t[u] / (\exists r) (t[CA] = r[CA]) \wedge \right. \\ \left. \neg (\exists r_1) (r[CA] \neq r_1[CA]) \wedge (r[PR] < r_1[PR]) \right\}$$

$$T1 = T2 = \text{TIENDA}$$

$$P(T1.CA) S (T1.HER = '1') (T1) - P(T2.CA) S (T1.HER = T2.HER) \wedge (T1.PR < T2.PR) \\ (T1 \times T2)$$

c.- Usuarios que se han descargado apps gratuitas de todas las categorías.

Para ~~que~~ cuál categoría tiene que

$$R = P(CU, CAT) S (PR = 0) \quad (\text{CU} \neq \text{TIENDA} \wedge \text{APPS}) \quad \left. \begin{array}{l} \\ \end{array} \right\} A/B$$

$$G = P(CAT) (\text{APPS})$$



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**23 de Enero de 2013**

Con el fin de establecer la procedencia de los cuadros que se venden en las galerías de arte, el Cuerpo Nacional de Policía ha creado una base de datos. En el esquema utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
AU	Autor de la pintura: Goya, Pablo Picasso, Juan Gris, ...
EA	Estilo artístico de la pintura: Cubismo, Impresionismo, Realismo, Surrealismo, ...
FC	Fecha de compra de la pintura: 01-12-2010, 05-07-2008, ...
GEN	Género: Bodegón, paisaje, retrato, ...
IC	Identificador del cuadro: 1, 2, 3, ...
N	Nombre del comprador
PR	Precio en Euros pagado por el cuadro. (Regalado = 0 Euros)
TC	Título del cuadro: Los Girasoles, la Gioconda, ...

Las tablas utilizadas son:

**CUADROS**(IC, AU, TC, EA, GEN)

**SIGNIFICADO:** El cuadro IC es del autor AU, lleva por título TC, pertenece al estilo EA y es del género GEN.

**CLAVE PRIMARIA:** (IC)

**PROPIETARIOS** (IC, N, FC, PR)

**SIGNIFICADO:** El cuadro IC ha sido comprado por la persona con nombre N, en la fecha FC, por un precio de PR Euros. Inicialmente el pintor es propietario de su cuadro y ha pagado por él un precio simbólico de 0 Euros.

**CLAVE PRIMARIA:** (IC, FC)

1) Responder en álgebra relacional a las siguientes consultas:

- a) Autores que no han pintado cuadros del género bodegón.
- b) Cuadros que han tenido al menos dos propietarios distintos desde el 01-03-2007.
- c) Propietarios que en un mismo día han comprado cuadros de todos los géneros.
- d) Personas que actualmente son propietarias de algún cuadro de Van Goth.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Autores que han pintado cuadros de diferentes estilos artísticos.
- b) Autores que sólo han pintado bodegones impresionistas.
- c) Precio más caro pagado por Los Girasoles de Van Goth.
- d) Pintores que han pintado cuadros de al menos los mismos géneros que algún otro autor.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

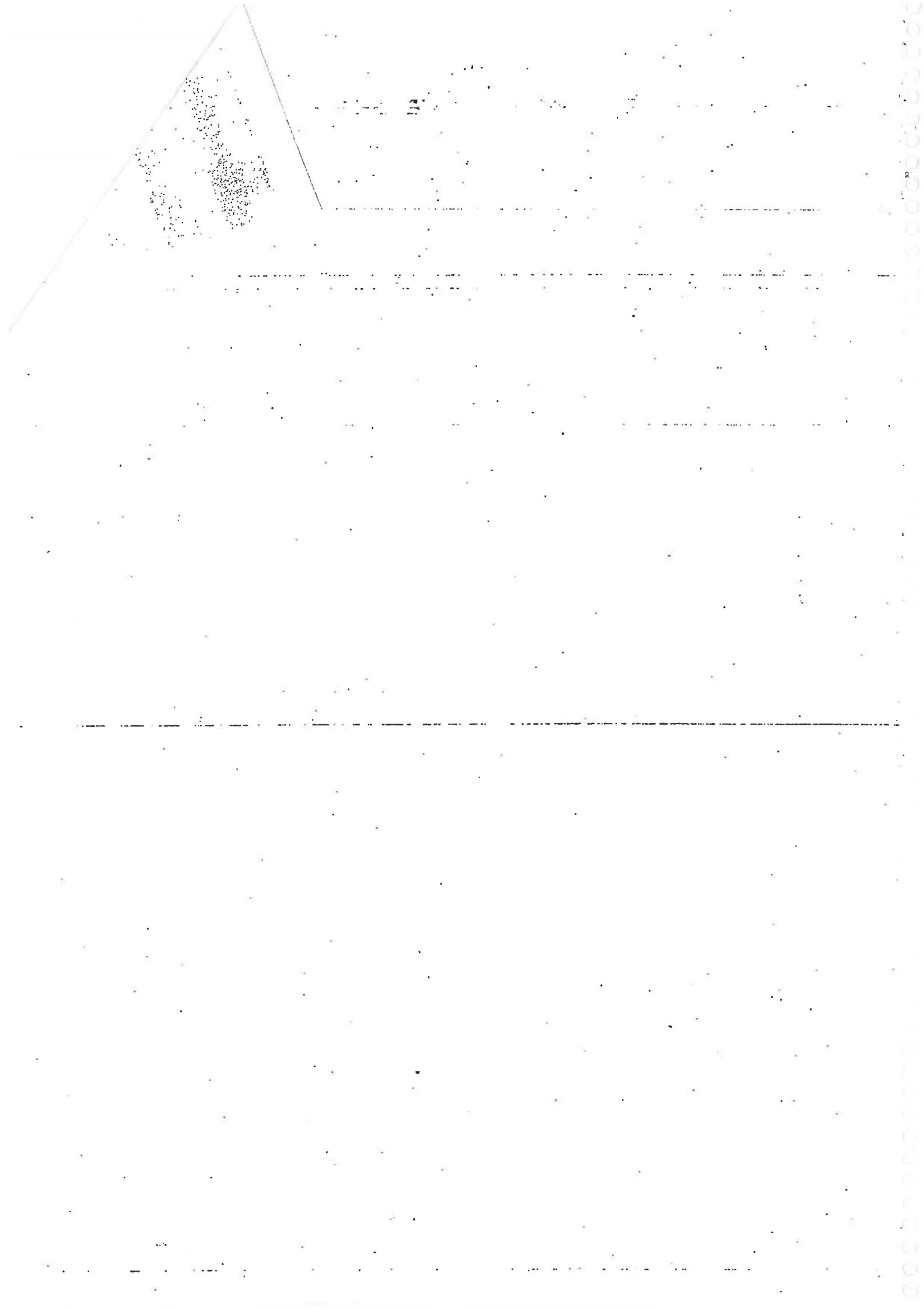
- a) Autores que han pintado algún cuadro que se haya vendido alguna vez por más de 10.000 Euros.
- b) Propietarios tales que todos los cuadros que han comprado cuestan más de 10.000 euros.
- c) Propietario actual de Los Girasoles de Van Goth.
- d) Propietarios que han comprado cuadros de todos los géneros pintados por algún autor.

4) Responder en SQL a las siguientes consultas:

- a) Personas tales que en 2010 les han regalado algún retrato.
- b) Número medio de cuadros correspondientes a cada estilo artístico.
- c) Cuadro de estilo impresionista más caro comprado en 2011.
- d) Propietarios tales que al menos el 40% del valor de sus compras corresponden al estilo impresionista.

5) Responder en SQL a las siguientes peticiones:

- a) Crea una vista con los pintores que sólo tengan cuadros surrealistas.
- b) Añade el campo 'Nacionalidad' a la tabla cuadros.
- c) Impide que una persona pueda gastar en un mismo año más de 10.000 euros.
- d) Impide que los cuadros puedan bajar de precio en sus ventas sucesivas.



23/ABRIL/2013

①

a)  $P(AU)(CUADROS) = P(AU) \wedge S(SEN = 'Bodegon') (CUADROS)$

b)  $A = B = PROPIETARIOS$

$P(IC) \wedge S(A.N \neq B.N) \wedge (A.FC \geq '01-03-2007') \wedge (A.IC = B.IC) (A \times B)$

c)  $P(SEN)(PROPIETARIOS) = A$

$P(N.SEN)(CUADROS \times PROPIETARIOS) = B$

$C = D$

$P(C.N) \wedge S(C.FC = D.FC) (C \times D)$

X Pueden haberse comprado un cuadros de algún género en diferentes días.

d)  $P(N') \wedge S(AU = 'Van Gogh') (CUADRO \neq PROPIETARIO) = A = B$

$P(N')(A) - P(B.N) \wedge S(A.F < B.F) (A \times B)$

②

a)  $\text{dom}(c) = CUADROS, \text{dom}(c1) = CUADROS$

$\{ t_u / (\exists c) (t[AU] = c[AU]) \wedge$   
 $\quad (\exists c1) (c1[AU] = c[AU]) \wedge (c1[EA] \neq c[EA]) \}$

b)  $\text{dom}(c) = CUADROS = \text{dom}(c1)$

$\{ t_u / (\exists c) (t[AU] = c[N]) \wedge$   
 $\quad (\exists c1) (c[AU] = c1[AU]) \wedge (c1[EA] \neq 'Impresionismo') \wedge (c1[EN] \neq 'Bodegon') \}$

c)  $\text{dom}(c) = CUADROS, \text{dom}(p) = PROPIETARIO = \text{dom}(p1)$

$\{ t_u / (\exists p) (t[PR] = p[PR]) \wedge$   
 $\quad (\exists c) (c[AU] = 'Van Gogh') \wedge (c.TC = 'los girasoles') \wedge$   
 $\quad (\exists p1) (p1[PR] > p[PR]) \wedge (p1[IC] = p[IC]) \}$

⑤

c) CREATE ASSERTION AS1

```
CHECK NOT EXIST (SELECT CD  
FROM MERCADO  
GROUP BY CD  
HAVING COUNT(IH) > 2);
```

d) CREATE ASSERTION AS2

CHECK

23/ENERO/2013

⇒ Fotocopiar campus

⑤ a) CREATE VIEW V1

```
AS (SELECT AU  
FROM CUADROS  
WHERE AU NOT IN (SELECT AU  
FROM CUADROS  
WHERE (ER <> 'Surrealismo')));
```

b) ALTER TABLE CUADROS

```
ADD (NOMBRELOAD VARCHAR2(20));
```

c) Impide que una persona pueda gastar en un mismo año más de 10.000 €

CREATE ASSERTION AS1

```
CHECK NOT EXIST (SELECT N  
FROM PROPIETARIOS  
WHERE (SUM(PR) > 10000)  
GROUP BY N, TO CHAR(FC, 'YYYYYY'));
```

Impedir que los cuadros bajen de precio en ventas sucesivas.

d) CREATE ASSERTION AS2

```
CHECK NOT EXISTS (SELECT P1.IC  
FROM PROPIETARIOS P1  
WHERE IC IN (SELECT IC  
FROM PROPIETARIOS P2  
WHERE (P1.PR > P2.PR)  
AND (P1.FC > P2.FC));
```

(3)

(4)

a) {<au>/( $\exists$ ic, tc, ea, gen)(<ic, au, tc, ea, gen>  $\in$  CUADROS) ^

$(\exists$ ict, nt, pct, pr1)(<ict, nt, pct, pr1>  $\in$  PROPIETARIOS) ^ (ic = ic1) ^

$(pr1 > 10\ 000)$  }

pr

b) {<n>/( $\exists$ ic, pc, pr)(<ic, n, pc, pr>  $\in$  PROPIETARIOS) ^

$(\exists$ ic1, pct1, nt1, pr1)(<ic1, pct1, nt1, pr1>  $\in$  PROPIETARIOS) ^ (n = nt1) ^

$(pr1 < 10\ 000)$  }

c) Propietario actual de los girasoles de Van Gogh.

{<n>/( $\exists$ ic, pc, pr)(<ic, n, pc, pr>  $\in$  PROPIETARIOS) ^ (ic = 'Los Girasoles') ^

$(\exists$ ict1, aut1, tct1, ea1, gen1)(<ict1, aut1, tct1, ea1, gen1>  $\in$  CUADROS) ^

$(\exists$ ic2, pc2, pr2)(<ic2, n2, pc2, pr2>  $\in$  PROPIETARIOS) ^ (aut1 = 'Van Gogh') ^

(ic = ic1) ^

$(ic1 = ic2) \wedge (pc2 > pc)$  }

d) Propietarios que han comprado cuadros de todos los géneros pintados por algún autor.

{<n>/( $\exists$ ic, pc, pr)(<ic, n, pc, pr>  $\in$  PROPIETARIOS) ^

$(\exists$ ic, tc, ea, au)( $\forall$ gen)(<ic, au, tc, ea, gen>  $\in$  CUADROS)}

(\*)

③

d)  $\text{dom}(c) = \text{CUADROS} = \text{dom}(c_1) = \text{dom}(c_2)$

$\{t_4 / (\exists c)(t[\text{AU}] = c[\text{AU}])\}$

$(\exists c_1)(c_1[\text{AU}] \neq c[\text{AU}])$

$\neg (\exists c_2)(c_2[\text{AU}] = c[\text{AU}]) \wedge (c_2[\text{GEN}] \neq c[\text{GEN}])\}$

---

④ a) SELECT N

FROM PROPIETARIOS NATURAL JOIN CUADROS

WHERE (AU <> N)

AND (PR = '0')

AND TO\_CHAR(FE, 'YYYY' = '2010')

AND (GEN = 'Retrato');

b) SELECT AVG(COUNT(\*))

FROM CUADROS

GROUP BY EA;

c) SELECT IC

FROM CUADROS NATURAL JOIN PROPIETARIOS

WHERE (EA = 'Impresionista')

AND TO\_CHAR(FE, 'YYYY' = '2011')

AND PR >=

--> (SELECT MAX(PR)

FROM PROPIETARIOS

WHERE (EA = 'Impresionista')

AND TO\_CHAR(FE, 'YYYY' = '2011'));

d) SELECT N

FROM PROPIETARIOS NATURAL JOIN CUADROS A

WHERE (EA = 'Impresionista')

GROUP BY N

HAVING (SUM(PR) \* 0.4) > (SELECT SUM(PR)

FROM PROPIETARIOS P

WHERE (P.N = A.N),



Bases de Datos  
Escuela Técnica Superior de Ingeniería Informática  
Universidad de La Laguna  
11 de Enero de 2013

La tienda oficial de aplicaciones (apps) de Apple (Apple Store) gestiona a través de una base de datos todas las descargas de apps que realizan sus usuarios. En el esquema de base de datos utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
A	Año de compra de la app: 2010, 2008, 2012, ...
CA	Código de la app: 1, 2, 3, ...
CU	Código de usuario: 1, 2, 3, ...
DNI	DNI del usuario
MER	Mercado: España, Estados Unidos, Argentina, ...
NE	Número de estrellas otorgadas: Entero de 0 a 5.
PR	Precio en Euros de una app en algún mercado. (Gratis = 0 Euros)

Las tablas utilizadas son:

TIENDA(CA, MER, PR)

SIGNIFICADO: La app con código CA está disponible en el mercado MER a un precio de PR euros.

CLAVE PRIMARIA: (CA, MER)

USUARIO(DNI, CU, MER)

SIGNIFICADO: La persona con dni DNI ha creado el usuario CU para el mercado MER.

CLAVE PRIMARIA: (CU)

COMPRA(CU, CA, A, NE)

SIGNIFICADO: El usuario CU ha comprado en su mercado la app CA en el año A y le ha dado una valoración de NE estrellas.

CLAVE PRIMARIA: (CU, CA)

1) Responder en álgebra relacional a las siguientes consultas:

- Apps que están disponibles simultáneamente en los mercados M1 y M2.
- Personas tales que todos los usuarios que han creado pertenecen a un mismo mercado.
- Apps que están disponibles gratuitamente en todos los mercados.
- Usuarios que se han descargado exactamente todas las apps gratuitas de su mercado.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Apps que están disponibles en el mercado M1 pero no en el M2.
- Usuarios que han valorado todas las apps que han descargado en 2012 con al menos 3 estrellas.
- App del mercado M1 más cara.
- Usuarios que se han descargado al menos las mismas apps que algún otro usuario.

$$P(CU)(compr) \rightarrow P(A, CU) \wedge (A, CU \neq B, CU) \wedge (A \cdot CA \neq B \cdot CA) \quad (A \times B)$$

3) Responder en cálculo relacional de dominios a las siguientes consultas:

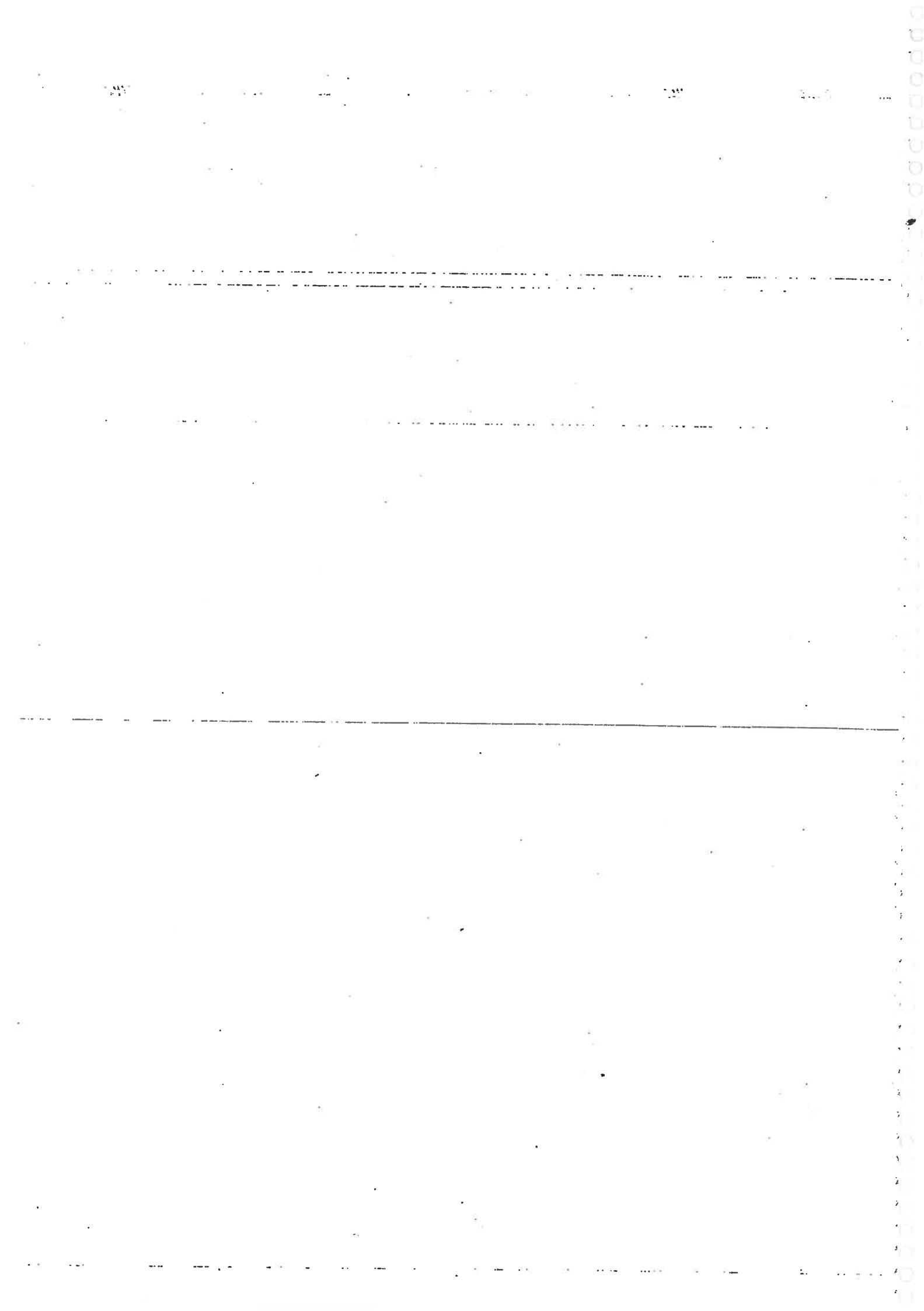
- Personas con más de un usuario en un mismo mercado.
- Apps que están disponibles, al mismo precio, en todos los mercados.
- Usuarios tales que todas las apps que han descargado cuestan menos de 1 euro.
- Mercados que tienen al menos las mismas apps que algún otro mercado.

4) Responder en SQL a las siguientes consultas:

- Personas que han descargado alguna app gratuita.
- Número medio de apps que se han descargado en 2012 los usuarios del mercado M1.
- Persona que ha gastado más en apps en 2012.
- Personas tales que al menos el 40% de las apps descargadas han sido realizadas a través de un mismo usuario.

5) Responder en SQL a las siguientes peticiones:

- Elimina aquellos usuarios que nunca han comprado nada.
- Incrementa en un 10% el precio de todas las apps del mercado M1 que valgan menos de un euro.
- Impide que una persona pueda crear más de dos usuarios en un mismo mercado.
- Límita el número de descargas gratuitas por año que pueden realizar los usuarios a 100.



EXAMEN BASES DE DATOS (18/Enero/2013)

1) Responder en álgebra relacional a las siguientes consultas:

a.- Apps que están disponibles simultáneamente en los mercados H1 y H2.

→  $P(CA) \circ S(HER=1) (TIENDA) \cap P(CA) (HER=2) (TIENDA)$

$$\left\{ \begin{array}{l} \text{dom}(t) = \text{dom}(t_1) = \text{TIENDA} \\ \{ t_{(1)} / (\exists t)(t[CA] = t_1[CA]) \wedge (t[HER] = 1) \wedge \\ (\exists t_1)(t_1[CA] = t[CA]) \wedge (t_1[HER] = 2) \} \end{array} \right.$$

$$\left\{ \begin{array}{l} \{ \langle ca \rangle / (\exists pr) (\langle ca, 1, pr \rangle \in \text{TIENDA}) \wedge \\ (\exists ca_2, pr_2) (\langle ca_2, 2, pr_2 \rangle \in \text{TIENDA}) \wedge (ca_2 = ca) \} \end{array} \right.$$

b.- Personas tales que todos los usuarios que han creado pertenecen a un mismo mercado.

$U1 = U2 = \text{USUARIO}$

→  $(P(DNI) (USUARIO)) -$

$(P(U1.DNI) \circ S(U1.DNI = U2.DNI) \wedge (U1.CU \neq U2.CU) \wedge (U1.MER \neq U2.MER))$

$$\left\{ \begin{array}{l} \text{dom}(u_1) = \text{dom}(u_2) = \text{USUARIO} \\ \{ t_{(1)} / (\exists u_1)(t[DNI] = u_1[DNI]) \wedge \\ (\exists u_2)(u_1[DNI] = u_2[DNI]) \wedge (u_1[MER] \neq u_2[MER]) \} \end{array} \right.$$

$$\left\{ \begin{array}{l} \{ \langle dni \rangle / (\exists cu, mer) (\langle dni, cu, mer \rangle \in \text{USUARIO}) \wedge \\ (\exists dni_2, cu_2, mer_2) (\langle dni_2, cu_2, mer_2 \rangle \in \text{USUARIO}) \\ \wedge (dni = dni_2) \wedge (mer \neq mer_2) \} \end{array} \right.$$

c. Apps que están disponibles gratuitamente en todos los mercados.

$$\begin{array}{|c|c|} \hline A = P(CA, HER) \wedge PR = 0 & (TIENDA) \\ \hline B = P(HER) & (TIENDA) \\ \hline \end{array}$$

$$\Rightarrow A/B$$

→ Elimina todas las filas que no coincidan con la 2<sup>a</sup> columna.

$$\left\{ \begin{array}{l} \text{dom}(t_1) = \text{dom}(t_2) = \text{TIENDA} \\ \{ t_1 / (\exists t_1) (t_1[CA] = t_1[CA]) \wedge (t_1[PR] = 0) \wedge \\ \neg (\exists t_2) (t_2[CA] = t_2[CA]) \wedge (t_1[HER] \neq t_2[HER]) \wedge (t_2[PR] \neq 0) \} \\ \{ <ca>/(\forall mer) (<ca, mer, 0> \in \text{TIENDA}) \end{array} \right.$$

d. Usuarios que se han descargado exactamente todas las apps gratuitas de su mercado.

2). Responder en cálculo relacional de tuplas las siguientes consultas:

a.- Apps que están disponibles en el mercado H1 pero no en el H2.

$$\left. \begin{array}{l} A = P(CA) \setminus S(HER = 'H1') (TIENDA) \\ B = P(CA) \setminus S(HER = 'H2') (TIENDA) \end{array} \right\} \boxed{A - B}$$

$$\left. \begin{array}{l} \text{dom}(t_1) = \text{dom}(t_2) = \text{TIENDA} \\ \{ t_1 / (\exists t_1) (t_1[CA] = t_1[CA]) \wedge (t_1[HER] = 'H1') \wedge \\ \quad \neg (\exists t_2) (t_2[CA] = t_1[CA]) \wedge (t_2[HER] = 'H2') \} \end{array} \right\}$$

$$\left. \begin{array}{l} \{ <ca> / (\exists pr) (<ca, 1, pr> \in \text{TIENDA}) \wedge \\ \quad \neg (\exists ca2, pr2) (<ca2, 2, pr2> \in \text{TIENDA}) \wedge (ca = ca2) \} \end{array} \right\}$$

b.- Usuarios que han valorado todas las Apps que han descargado en 2012 con al menos 3 estrellas.

$$\left. \begin{array}{l} A = P(CU, NE) \setminus S(A = 2012) (COMPRA) \\ B = P(NE) \setminus S(NE \geq 3) (COMPRA) \end{array} \right\} \begin{array}{l} A/B \rightarrow B \text{ está contenido en } A. \\ \text{Parejas de datos, el CU que cumpla ambas condiciones.} \end{array}$$

$$\left. \begin{array}{l} \text{dom}(c) = \text{dom}(c_1) = \text{COMPRA} \\ \{ t_1 / (\exists c) (t_1[CU] = c[CU]) \wedge (c[A] = '2012') \wedge (c[NE] \geq 3) \wedge \\ \quad \neg (\exists c_1) (c_1[CU] = c[CU]) \wedge (c_1[A] = '2012') \wedge (c_1[NE] < 3) \} \end{array} \right\}$$

$$\left. \begin{array}{l} \{ <cu> / (\exists ca, ne) (<cu, ca, 2012, ne> \in \text{COMPRA}) \wedge (ne \geq 3) \wedge \\ \quad \neg (\exists cu2, ca2, ne2) (<cu2, ca2, 2012, ne2> \in \text{COMPRA}) \wedge (cu = cu2) \wedge \\ \quad (ne < 3) \} \end{array} \right\}$$

c- App del mercado M1 más cara

$$A = B = \text{TIENDA}$$

$$P(CA) S(MER=1)(A) - P(A.CA) S(A.CA \neq B.CA) \wedge (A.MER=1) \wedge (B.MER=1) \\ \wedge (A.PR < B.PR)$$

$$\text{dom}(t_1) = \text{dom}(t_2) = \text{TIENDA}$$

$$\{ t_1 / (\exists t_1) (t_1[CA] = t_1[CA]) \wedge (t_1[MER] = M1) \wedge \\ \wedge (\exists t_2) (t_1[CA] \neq t_2[CA]) \wedge (t_2[MER] = M1) \wedge (t_2[PR] > t_1[PR]) \}$$

$$\{ \langle ca \rangle / (\exists pr) (\langle ca, M1, pr \rangle \in \text{TIENDA}) \wedge \\ \neg (\exists ca_2, pr_2) (\langle ca_2, M1, pr_2 \rangle \wedge (ca \neq ca_2) \wedge (pr < pr_2)) \}$$

d- Usuarios que se han descargado al menos las mismas Apps que algún otro usuario.

$$\text{dom}(c) = \text{dom}(c1) = \text{COMPRA}$$

$$\{ t_{cu} / (\exists c) (\underline{t_{cu}} = c[cu]) \wedge \\ \cancel{(\exists c1)} (c1[cu] \neq c[cu]) \cancel{(\cancel{t_{cu}} = \cancel{c1[cu]})} \} \quad X$$

Para cualquier opción de descargar pu c1[cu]  
entonces c[cu] también se le ha descargado.

$$(\forall c2) ($$

USUARIO = (DNI, CU, MER)

3) Resolver en cálculo relacional de dominios a las siguientes consultas.

a.- Personas con más de un usuario en un mismo mercado

A = B = USUARIO

P(DNI) S (A.DNI = B.DNI) ^ (A.CU ≠ B.CU) ^ (A.MER = B.MER)

dom(u<sub>1</sub>) = dom(u<sub>2</sub>) = USUARIO

{ tu / ( ∃ u<sub>1</sub>) (t[DN<sub>1</sub>] = u<sub>1</sub>[DN<sub>1</sub>]) ^

( ∃ u<sub>2</sub>) (u<sub>1</sub>[DN<sub>1</sub>] = u<sub>2</sub>[DN<sub>1</sub>]) ^ (u<sub>1</sub>[MER] = u<sub>2</sub>[MER]) ^ (u<sub>1</sub>[CU] ≠ u<sub>2</sub>[CU]) }

{ <dh<sub>i</sub>> / ( ∃ cu, mer) (<dh<sub>i</sub>, cu, mer> ∈ USUARIO) ^

( ∃ dni<sub>2</sub>, cu<sub>2</sub>, mer<sub>2</sub>) (<dni<sub>2</sub>, cu<sub>2</sub>, mer<sub>2</sub>> ∈ USUARIO) ^

(dh<sub>i</sub> = dni<sub>2</sub>) ^ (mer<sub>1</sub> = mer<sub>2</sub>) ^ (cu ≠ cu<sub>2</sub>) }

b.- Apps que están disponibles, al mismo precio, en todos los mercados.

TIENDA = (CA, MER, PR)

A = B = TIENDA

P(A.CA) (A) - P(A.CA) S (A.CA ≠ B.CA) ^ (A.MER ≠ B.MER) ^ (A.PR ≠ B.PR)

dom(t<sub>1</sub>) = dom(t<sub>2</sub>) = TIENDA

{ tu / ( ∃ t<sub>1</sub>) (t[CA] = t<sub>1</sub>[CA]) ^

( ∃ t<sub>2</sub>) (t<sub>2</sub>[CA] ≠ t<sub>1</sub>[CA]) ^ (t<sub>2</sub>[MER] ≠ t<sub>1</sub>[MER]) ^ (t<sub>2</sub>[PR] ≠ t<sub>1</sub>[PR]) }

{ <ca> / ( ∃ mer, pr) (<ca, mer, pr> ∈ TIENDA) ^

( ∃ ca<sub>2</sub>, mer<sub>2</sub>, pr<sub>2</sub>) (<ca<sub>2</sub>, mer<sub>2</sub>, pr<sub>2</sub>> ∈ TIENDA) ^

(ca ≠ ca<sub>2</sub>) ^ (mer ≠ mer<sub>2</sub>) ^ (pr ≠ pr<sub>2</sub>) }

c. Usuarios tales que todas las apps que han descargado cuestan menos de 1€.

$$P(CU) \cap (\text{COMPRAS} * \text{TIENDA}) - P(CU) \cap S(PR > 1) \cap (\text{COMPRAS} * \text{TIENDA})$$

$$\begin{aligned} \text{dom}(t_1) &= \text{TIENDA}; \text{dom}(c) = \text{COMPRAS} \\ \{ t_1 / (\exists c) (t_1[CU] = c[CU]) \wedge \\ &\quad \neg (\exists t_2) (t_2[CA] = c[CA]) \wedge (t_1[PR] > 1) \} \end{aligned}$$

$$\begin{aligned} \{ <cu> / (\exists ca, a, ne) (<cu, ca, a, ne> \in \text{COMPRAS}) \wedge \\ &\quad \neg (\exists ca_2, mer, pr) (<ca_2, mer, pr> \in \text{TIENDA}) \wedge (pr > 1) \wedge (ca = ca_2) \} \end{aligned}$$

### \* CORREGIR

d. Mercados que tienen al menos las mismas apps que algún otro mercado.

$$\text{TIENDA} = (CA, MER, PR)$$

$$A = B = \text{TIENDA}$$

$$P(A.MER) \cap S(A.CA = B.CA) \wedge (A.MER = B.MER)$$

$$(\star) \quad \text{dom}(t_1) = \text{dom}(t_2) = \text{TIENDA}$$

$$\{ t_{11} / (\exists t_1) (t_1[MER] = t_1[MER]) \wedge \\ (\exists t_2) (t_2[MER] = t_1[MER]) \wedge (t_2[CA] = t_1[CA]) \}$$

$$(\star) \quad \{ <mer> / (\exists ca, pr) (<ca, mer, pr> \in \text{TIENDA}) \wedge$$

$$(\exists ca_2, mer_2, pr_2) (<ca_2, mer_2, pr_2> \in \text{TIENDA}) \wedge$$

$$\neg (\cancel{ca = ca_2}) \wedge (mer \neq mer_2) \} \wedge (\cancel{ca_3, pr_3})$$

$$\cancel{\exists A?} \quad <ca_3, mer_12, pr_3> \notin \text{TIENDA} \vee$$

mer tiene la app.

4). Responder en SQL a las siguientes consultas:

a- Personas que han descargado alguna App gratuita.

```
SELECT DNI  
FROM USUARIO  
WHERE CU IN (SELECT CU  
              FROM COMPRA  
              WHERE CA IN (SELECT CA FROM TIENDA  
                            WHERE (PR=0))),
```

④ b- SELECT DNI

```
FROM USUARIO U, COMPRA C, TIENDA T  
WHERE (U.CU=C.CU) AND (C.CA=T.CA)  
AND (T.PR=0),
```

b- Número medio de Apps que se han descargado en 2012 los usuarios del mercado H1.

```
SELECT AVG(COUNT(*))  
FROM COMPRA  
WHERE (A='2012')  
AND CA IN (SELECT CA  
              FROM TIENDA  
              WHERE (MER='H1'));
```

④ c- SELECT AVG(COUNT(\*))

```
FROM TIENDA T, COMPRA C  
WHERE (T.CA=C.CA)  
AND (T.MER='H1')  
AND (C.A='2012');
```

c.- Persona que ha gastado más en Apps en 2012.

SELECT DNI

FROM TIENDA T, COMPRA C, USUARIO U

WHERE (T.CA = C.CA)

AND (U.CU = C.CU)

GROUP BY U.DNI

HAVING (SUM(T.PR)) >= ALL (SELECT SUM(PR)

FROM TIENDA T<sub>1</sub>, COMPRA C<sub>1</sub>, USUARIO U<sub>1</sub>

WHERE (T<sub>1</sub>.CA = C<sub>1</sub>.CA)

AND (U<sub>1</sub>.CU = C<sub>1</sub>.CU)

GROUP BY U<sub>1</sub>.DNI);

d.- Personas tales que, al menos el 40% de las Apps descargadas han sido realizadas a través de un mismo usuario.

SELECT DNI

FROM USUARIO U, COMPRA C A

WHERE (U.CO = C.CU)

GROUP BY DNI;

HAVING (COUNT(CA) \* 0'4) >= ANY

(SELECT COUNT(CA) NATURAL JOIN R  
FROM USUARIO U<sub>1</sub>, COMPRA C<sub>1</sub>  
WHERE (U<sub>1</sub>.CU = C<sub>1</sub>.CU) A.DNI = B.DNI  
GROUP BY CU);

5) Responder en SQL a las siguientes peticiones:

a. Elimina aquellos usuarios que nunca han comprado nada.

DELETE FROM USUARIO

WHERE CU NOT IN (SELECT CU

FROM COMPRA);

SELECT CU ?

FROM COMPRA C, USUARIO

WHERE (C.CU = U.CU);

b. Incrementa en un 10% el precio de todas las Apps del mercado H1 que valgan menos de 1€.

UPDATE TIENDA

SET PR = PR + (PR \* 0.1)

WHERE (PR < 1)

AND (MER = 'H1');

c. Impide que una persona pueda crear más de dos usuarios en un mismo mercado.

CREATE ASSERTION AS1

CHECK NOT EXIST (SELECT DNI ;

FROM USUARIO

GROUP BY DNI

HAVING COUNT(CU) > 1);

AS1

M B

d. Limita el número de descargas gratuitas por año que puedan realizar los usuarios, a 100.

CREATE ASSERTION AS2

CHECK NOT EXIST (SELECT \*

FROM TIENDA NATURAL JOIN COMPRA

WHERE (T.MER = U.MER)

AND (T.PR = 0)

GROUP BY A

HAVING COUNT(\*) > 100);



Estructura de Datos y de la Información  
Escuela Técnica Superior de Ingeniería Informática  
Universidad de La Laguna  
4 de Julio de 2012

El Ministerio de Industria y Energía quiere almacenar información sobre la producción industrial del país referida a aparatos eléctricos y sus componentes. Para ello tiene una base de datos que tiene el siguiente esquema:

Atributo	Significado
F	Fábrica
C	Componente
A	Aparato
CNT	Cantidad

**PRODUCE(F, C)**

SIGNIFICADO: La fábrica F produce el componente C.

CP: (F, C) CA: (C)

**NECESITA(A, C, CNT)**

SIGNIFICADO: El aparato A necesita, del componente C, CNT unidades.

CP: (A, C)

**ENSAMBLA(F, A)**

SIGNIFICADO: La fábrica F ensambla (monta) el aparato A.

CP: (F, A) CA: (A)

1) Responder en álgebra relacional a las siguientes preguntas:

- a) Fábricas que no producen el componente C.
- b) Aparatos que precisan al menos dos componentes distintos.
- c) Componente que interviene en mayor cantidad en el aparato X.
- d) Fábricas que ensamblan algún aparato para el cual producen todas sus componentes.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Fábricas que sólo producen los componentes C1 o C2.
- b) Fábricas que producen todos los componentes del aparato A1.
- c) Componente que interviene en mayor cantidad en el aparato A.
- d) Fábricas que ensamblan algún aparato para el cual producen todas sus componentes.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Aparatos que necesitan el componente A pero no el C.
- b) Aparatos que precisan, a lo sumo, dos componentes distintos.
- c) Componentes que son necesarios en todos los aparatos.
- d) Fábricas que producen al menos un componente de cada aparato que ensamblan.

4) Responder en SQL a las siguientes consultas:

- a) Fábricas que sólo ensamblan aparatos.
- b) Número medio de componentes distintos que necesitan los aparatos ensamblados en la fábrica A.
- c) Componente más frecuentemente utilizado en la construcción de aparatos.
- d) Fábricas que producen al menos el 60% de los componentes que necesita alguno de los aparatos que ellas ensamblan.

5) Sea el esquema R(A, B, C, D, E) con dependencias funcionales  $DF = \{A \rightarrow CE, CD \rightarrow AB, D \rightarrow E, B \rightarrow E, E \rightarrow D\}$ .

- a) ¿Está R en FNBC? Razona la respuesta.
- b) Calcular  $CE^+$ .
- c) Probar algebraicamente que  $CB \rightarrow A \in DF^+$
- d) Descomponer R en FNBC. ¿Se pierden dependencias funcionales? Razona la respuesta.

1) Responder en álgebra relacional a las siguientes preguntas:

a) Fábricas que no producen el componente c.

$$P(F) \text{ (PRODUCE)} - P(F) S(c = 'C') \text{ (PRODUCE)}$$

b) Aparatos que precisan al menos dos componentes distintos.

$$N1 = N2 = \text{NECESITA}$$

$$P(N1.A) S(N1.A = N2.A) \wedge (N1.C = N2.C) \quad (N1 \times N2)$$

c) Componente que interviene en mayor cantidad en el aparato X.

$$N1 = N2 = \text{NECESITA}$$

$$P(A.C) S(A.A = 'X')(A) - P(A.C) (A.CNT < B.CNT) \wedge (B.A = 'X') \quad (A \times B)$$

$$P(N1.C) S(N1.A = 'X')(N1) = A$$

$$P(N2.C) S(N2.CNT < N1.CNT) \wedge (N2.A = N1.A) \quad (N1 \times N2) = B \quad \left\{ \begin{array}{l} \\ \\ \end{array} \right. \quad \underline{\underline{A - B}}$$

d) Fábricas que ensamblan algún aparato para el cual producen todas sus componentes.

\* PRODUCE

$$\left. \begin{array}{l} A = P(F, A, C) \text{ (NECESITA * ENSAMBLA)} \\ B = P(A, C) \text{ (NECESITA)} \end{array} \right\} A / B \quad (B \text{ está contenido en } A)$$

$$P(F, C) \text{ (PRODUCE)}$$

2) - Responder en cálculo relacional de t-uplas a las siguientes consultas.

a.- Fábricas que sólo producen los componentes C1 o C2.

$$\text{dom}(p) = \text{dom}(p_1) = \text{PRODUCE}$$

$$\left\{ \begin{array}{l} t_u / (\exists p) (t[F] = p[F]) \wedge \\ \neg (\exists p_1) (p_1[F] = p[F]) \wedge (p_1[C] \neq 'C1') \wedge (p_1[C] \neq 'C2') \end{array} \right\}$$

b.- Fábricas que producen todos los componentes del aparato A1.

$$\text{dom}(p) = \text{PRODUCE}, \text{dom}(n) = \text{NECESITA}$$

$$\left\{ \begin{array}{l} t_u / (\exists p) (t[F] = p[F]) \wedge \\ \neg (\exists n) (n[A] = 'A1') \wedge (n[C] \neq p[C]) \end{array} \right\}$$

$$\{ \neg (\exists n) (n[A] = 'A1') \wedge (n[C] = p[C]) \}$$

c.- Componente que interviene en mayor cantidad en el aparato A.

$$\text{dom}(n) = \text{dom}(n_1) = \text{NECESITA}$$

$$\left\{ \begin{array}{l} t_u / (\exists n) (t[C] = n[C]) \wedge (n[A] = 'A') \wedge \\ \neg (\exists n_1) (n_1[A] = n[A]) \wedge (n_1[C] \neq n[C]) \wedge (n[CNT] < n_1[CNT]) \end{array} \right\}$$

d.- Fábricas que ensamblan algún aparato para el cual producen todas sus componentes.

$$\text{dom}(e) = \text{ENSAMBLA}; \text{dom}(p) = \text{PRODUCE}; \text{dom}(n) = \text{NECESITA}$$

$$\left\{ \begin{array}{l} t_u / (\exists e) (t[F] = e[F]) \wedge \\ \neg (\exists p, n) (n[C] = p[C]) \wedge (n[A] = e[A]) \wedge (p[F] = e[F]) \end{array} \right\}$$

Q)  $R(A, B, C, D, E)$        $DFT = \{A \rightarrow CE, CD \rightarrow AB, D \rightarrow E, B \rightarrow E, E \rightarrow D\}$

$$\Rightarrow A^+ = \{A, C, E, D, B\} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{superclaves}$$

$$CD^+ = \{C, D, A, B, E\}$$

$$D^+ = \{D, E\}$$

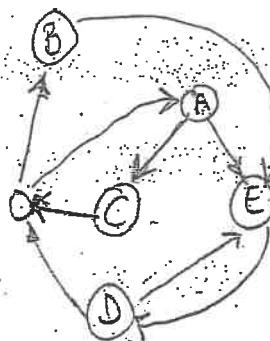
$$B^+ = \{B, E, D\}$$

$$E^+ = \{E, D\}$$

Buscamos llegar a todos los nodos. Se puede hacer probando los caminos en el b.

claves:  $(A)(CD)(BC)(CE) \rightarrow$  las superclaves son claves.

No está en FNBC porque en la dep. funcional  $(D \rightarrow E), (B \rightarrow E)(E \rightarrow D)$  el antecedente no es superclave.



b)  $CE^+ = \{C, E\} \quad E \rightarrow D$

$$CE^+ = \{C, E, D\} \quad CD \rightarrow AB$$

$$CE^+ = \{C, E, D, A, B\}$$

$CB \rightarrow A$  e DFT

$$CD \rightarrow AB \quad \left. \begin{array}{l} CD \rightarrow A \\ CD \rightarrow B \end{array} \right\}$$

Descomponemos.

$$\left. \begin{array}{l} B \rightarrow E \\ E \rightarrow D \end{array} \right\} \quad \left. \begin{array}{l} B \rightarrow D \Rightarrow CB \rightarrow CD \\ \text{Aumentativa.} \end{array} \right\}$$

$$\left. \begin{array}{l} CD \rightarrow A \\ CB \rightarrow CD \end{array} \right\} \quad \boxed{CB \rightarrow A} \quad \text{Transitiva.}$$

d)  $R1(D; E)$

$$DF1 = \{D \rightarrow E\}$$

Clave (D). FNBC.

$R2(A, B, C, D)$

$$DF2 = \{CD \rightarrow AB, A \rightarrow CD, B \rightarrow D\}$$

$$CD^+ = \{CD, A, B\}$$

$$A^+ = \{A, C, D, B\}$$

$$\rightarrow B^+ = \{B, D\}$$

$R2.1(B, D)$

$$DF2.1 = \{B \rightarrow D\}$$

$$B^+ = \{B, D\}$$

Clave (B). FNBC.

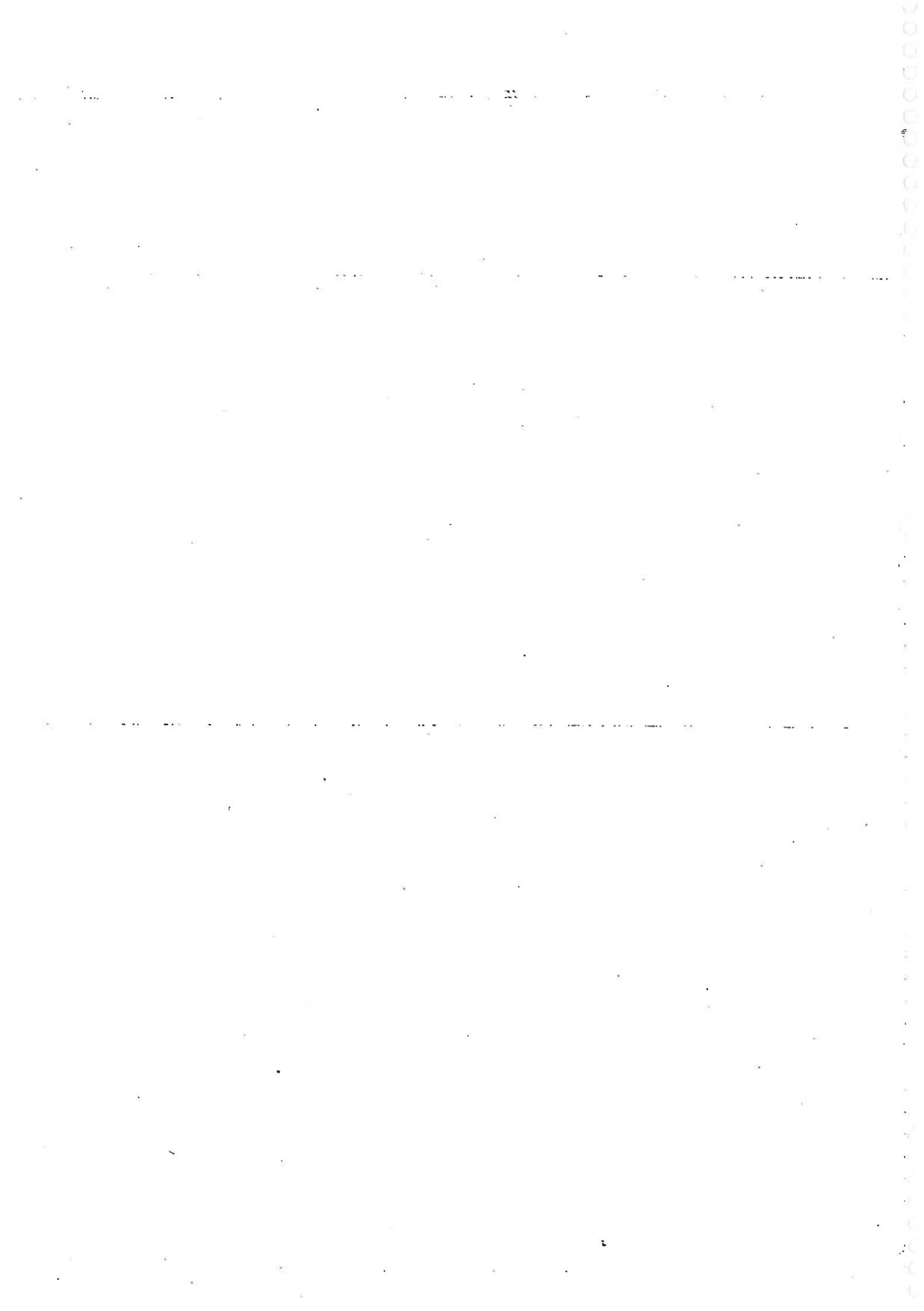
$R3(A, B, C)$

$$DF3 = \{A \rightarrow CB, CB \rightarrow AB\}$$

$$A^+ = \{A, C, B\}$$

$$CB^+ = \{C, B, A\}$$

Universal (R1) < Universal (R2)



3) Responder en cálculo relacional de dominios a las siguientes consultas:

a. Aparatos que necesitan el componente A, pero no el C.

$$\{ \langle a \rangle / (\exists c, \text{cnt}) (\langle a, 'A', \text{cnt} \rangle \in \text{NECESITA}) \\ \quad \neg (\exists a_1, c_1, \text{cnt}_1) (\langle a_1, 'C', \text{cnt}_1 \rangle \in \text{NECESITA}) \wedge (a = a_1) \}$$

\*) b. Aparatos que precisan, a lo sumo, dos componentes distintos.

$$\{ \langle a \rangle / (\exists c, \text{cnt}) (\langle a, c, \text{cnt} \rangle \in \text{NECESITA}) \wedge \\ \quad (\exists a_1, c_1, \text{cnt}_1) (\langle a_1, c_1, \text{cnt}_1 \rangle \in \text{NECESITA}) \wedge \\ \quad (a = a_1) \wedge (c \neq c_1) \}$$

c. Componentes que son necesarios en todos los aparatos.

$$\{ \langle c \rangle / (\forall a) (\exists \text{cnt}) (\langle a, c, \text{cnt} \rangle \in \text{NECESITA}) \}$$

\*) d. Fábricas que producen al menos un componente de cada aparato que ensamblan.

$$\{ \langle f \rangle / (\exists c) (\langle f, c \rangle \in \text{PRODUCE}) \wedge \neg (\exists a_1, c_1, \text{cnt}_1) (\langle a_1, c_1, \text{cnt}_1 \rangle \in \text{NECESITA}) \wedge \\ \quad (\exists f_2, a_2) (\langle f_2, a_2 \rangle \in \text{ENSAMBLA}) \wedge (c \neq c_1) \wedge (a_1 = a_2) \}$$

•  $\text{dom}(e) = \text{ENSAMBLA}, \text{dom}(p) = \text{PRODUCE}, \text{dom}(n) = \text{NECESITA}$

$$\{ t_u / (\exists e) (t[F] = e[F]) \wedge \\ \quad (\exists p, n) (p[F] = e[F]) \wedge (e[A] = n[A]) \wedge (n[C] \neq p[C]) \}$$

4) Responder en SQL a las siguientes consultas:

a. Fábricas que sólo ensamblan aparatos.

```
SELECT F  
FROM ENSAMBLA E  
WHERE NOT EXISTS (SELECT F FROM PRODUCE P  
WHERE (E.F = P.F));
```

b. Número medio de componentes distintos que necesitan los aparatos ensamblados en la fábrica A.

```
SELECT AVG (COUNT (C))  
FROM NECESITA NATURAL JOIN ENSAMBLA  
WHERE (F = 'A')  
GROUP BY A;
```

c. Componente más frecuentemente utilizado en la construcción de aparatos.

```
SELECT MAX (COUNT (*))  
FROM NECESITA  
GROUP BY C;
```

d. Fábricas que producen al menos, el 60% de los componentes que necesita alguno de los aparatos que ellos ensamblan.

Rev. →

```
SELECT F  
FROM PRODUCE P  
GROUP BY F  
HAVING (COUNT (C) * 0.6) > ALL (SELECT COUNT (C)  
FROM NECESITA N NATURAL JOIN ENSAMBLA E  
WHERE (P.F = E.F))
```



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**

4 de Julio de 2012

Una agencia de trabajo temporal tiene informatizada toda la gestión relativa a su actividad empresarial. En el esquema de base de datos propuesto los atributos que se utilizan se abrevian utilizando el siguiente convenio:

Atributo	Significado
CE	Código de empresa ofertante de empleo
DNI	DNI del trabajador
ND	Número de días requeridos: 1, 2, 3
NO	Número de oferta: 1, 2, 3
OF	Oficio del trabajador: electricista, fontanero, carpintero, ...
S	Salario por día en Euros

Las tablas utilizadas son:

**TRABAJADORES (DNI, OF)**

**SIGNIFICADO:** El trabajador con dni DNI sabe desempeñar el oficio OF.

**CLAVES:** (DNI, OF)

**OFERTAS (NO, CE, OF, ND, S)**

**SIGNIFICADO:** La oferta con número NO corresponde a la empresa CE, la cual precisa durante ND días un operario del oficio OF al que pagará un salario de S euros diarios.

**CLAVES:** (NO)

**ASIGNACIÓN (NO, DNI)**

**SIGNIFICADO:** A la oferta de empleo NO se ha asignado la persona con dni DNI.

**CLAVES:** (NO) **CLAVES AJENAS:** (NO), (DNI)

1) Responder en álgebra relacional a las siguientes consultas:

- Empresas que han solicitado un electricista durante más de 15 días.
- Personas que saben desempeñar al menos dos oficios distintos.
- Empresas que ofertan al menos los mismos oficios que la empresa E1.
- Personas que saben desempeñar al menos los mismos oficios que alguna otra persona.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Ofertas que todavía no han sido asignadas a nadie.
- Personas que sólo saben desempeñar un único oficio.
- Salario más alto ofertado por la compañía E1.
- Oficios que son siempre ofertados durante más de 15 días.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

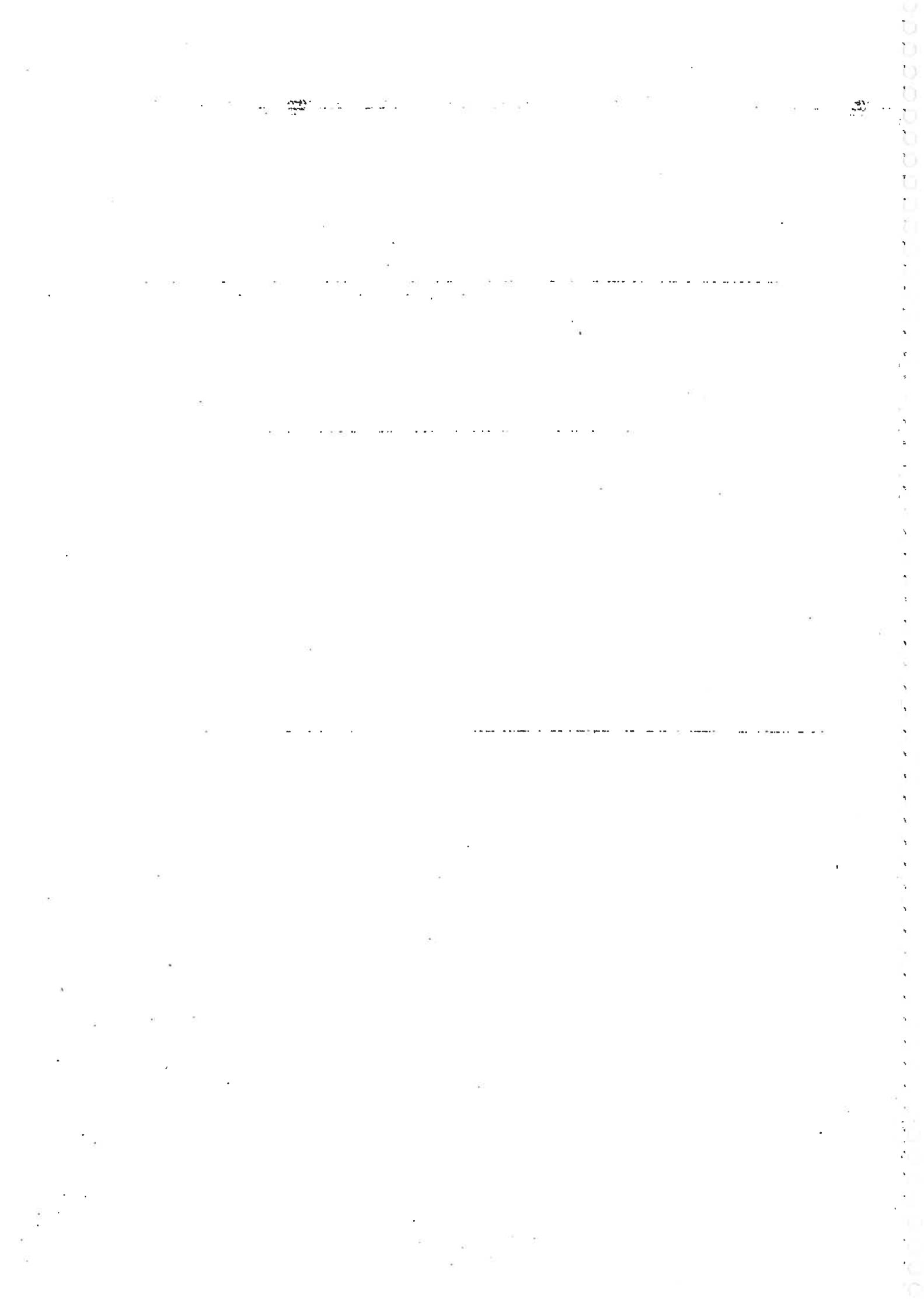
- Empresas que solicitan fontaneros pero no carpinteros.
- Oficios que ofertan todas las compañías.
- Oferta de empleo realizada por la empresa E1 de mayor duración.
- Personas que con alguna empresa sólo han trabajado como electricistas.

4) Responder en SQL a las siguientes consultas:

- Salario medio diario de las ofertas realizadas por la empresa E1.
- Oferta disponible con mayor salario diario.
- Empresa que ha realizado mayor número de ofertas.
- Empresas tales que al menos el 70% de sus ofertas son de un mismo tipo.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {A → C, CD → E, B → AE, E → BD}.

- Calcular todas las claves. ¿Está R en FNBC? Razóna la respuesta.
- Calcular AD+.
- Probar usando los axiomas de Armstrong que AD → B.
- Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?



3) Responder en cálculo relacional de dominios a las siguientes consultas:

a. Empresas que solicitan fontaneros pero no carpinteros.

$$\{ \langle ce \rangle / (\exists no, of, nd, s) (\langle no, ce, of, nd, s \rangle \in OFERTAS) \wedge (of = 'fontaneros') \wedge \\ \exists (no1, of1, nd1, s1) (\langle no1, ce, of1, nd1, s1 \rangle \in OFERTAS) \wedge \\ (of1 = 'carpinteros') \}$$

b. Oficios que ofertan todos los compañías. (ce)

$$\{ \langle of \rangle / (\exists no, ce, nd, s) (\langle no, ce, of, nd, s \rangle \in OFERTAS) \wedge \\ (\exists no1, ce1, of1, nd1, s1) (\langle no1, ce1, of1, nd1, s1 \rangle \in OFERTAS) \wedge (of = of1) \wedge \\ (ce \neq ce1) \}$$

$$\{ \langle of \rangle / (\forall ce) (\exists no, nd, s) (\langle no, ce, of, nd, s \rangle \in OFERTAS) \}$$

c. Oferta de empleo realizada por la empresa E1 de mayor duración.

$$\{ \langle of \rangle / (\exists no, ce, nd, s) (\langle no, ce, of, nd, s \rangle \in OFERTAS) \wedge (ce = 'E1') \wedge \\ (\exists no1, ce1, of1, nd1, s1) (\langle no1, ce1, of1, nd1, s1 \rangle \in OFERTAS) \wedge (ce1 = ce) \wedge \\ (nd < nd1) \} \\ (\forall ce) \dots (nd > nd1)$$

d. Personas que con alguna empresa sólo han trabajado como electricistas.

$$\{ \langle dni \rangle / (\exists no) (\langle no, dni \rangle \in ASIGNACION) \wedge \\ (\exists ce, nd, s) (\langle no, ce, 'electricista', nd, s \rangle \in OFERTA) \wedge \\ (\exists no1, of, nd, s) (\langle no1, ce, of, nd, s \rangle \in OFERTA) \wedge (of \neq 'electricista') \}$$

4). Responder en SQL a las siguientes consultas.

a) Salario medio diario de las ofertas realizadas por la empresa EI.

SELECT AVG (S)

FROM OFERTAS WHERE (CE = 'EI');

b) Oferta disponible con mayor salario diario.

SELECT NO

FROM OFERTAS WHERE S IN (SELECT MAX (S)  
FROM OFERTAS);

c) Empresa que ha realizado mayor número de ofertas.

SELECT CE

CE → Empresa.

FROM OFERTAS

GROUP BY CE

HAVING COUNT (NO) >= (SELECT COUNT (\*)

FROM OFERTAS

GROUP BY CE);

d) Empresas tales que al menos el 70% de sus ofertas son de un mismo tipo.

SELECT CE

FROM OFERTAS O1

GROUP BY CE

HAVING (COUNT (\*) \* 0.7) >= (SELECT COUNT (\*)

FROM OFERTAS O2

WHERE (O1.OF = O2.OF)

GROUP BY CE);

B) Sea  $R = (A, B, C, D, E)$  una relación con dependencias funcionales:  
 $DF = \{A \rightarrow C, CD \rightarrow E, B \rightarrow AE, E \rightarrow BD\}$ .

a. Calcular todas las claves de esta  $R$  en FNUC? Razona la respuesta.

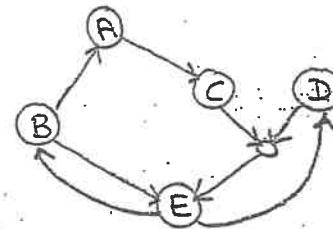
$$A^+ = \{A, C\}$$

$$CD^+ = \{C, D, E, B, A\} \rightarrow \text{Superclave. FNUC.}$$

$$B^+ = \{B, A, E, D, C\} \rightarrow \text{"}$$

$$E^+ = \{E, B, D, A, C\} \rightarrow \text{"}$$

$$\underline{\text{claves}}: (B)(E)(CD)(AD)$$



-  $R$  no está en FNUC porque en la dependencia funcional ( $A \rightarrow C$ ) el antecedente no es superclave.

b. Calcular  $AD^+$ .

$$AD^+ = \{A, D\} \quad A \rightarrow C$$

$$AD^+ = \{A, D, C\} \quad CD \rightarrow E$$

$$AD^+ = \{A, D, C, E\} \quad E \rightarrow BD$$

$$AD^+ = \{A, D, C, E, B\}$$

c. Probar usando los axiomas de Armstrong que  $AD \rightarrow B$ .

Aumentativa

$$A \rightarrow C \Rightarrow AD \rightarrow CD \quad \xrightarrow{\text{Transitiva}}$$

$$\begin{array}{c} CD \rightarrow E \\ \hline E \rightarrow BD \end{array} \quad \left. \begin{array}{c} AD \rightarrow E \\ E \rightarrow BD \end{array} \right\} \begin{array}{c} C \rightarrow DA \\ \hline AD \rightarrow BD \end{array} \Rightarrow \begin{array}{c} \xrightarrow{D \rightarrow A} \\ \text{Descr.} \end{array} \quad \boxed{\begin{array}{l} AD \rightarrow B \\ AD \rightarrow D \end{array}}$$

$$A \rightarrow C \Rightarrow AD \rightarrow CD \quad \left. \begin{array}{c} E \rightarrow B \\ AD \rightarrow E \end{array} \right\} \begin{array}{c} E \rightarrow B \\ AD \rightarrow B \end{array}$$

d. Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?

$R_1 (A, C)$

$DF = \{ A \rightarrow C \}$

$A^+ = \{ A, C \}$

Clave (A). FNBC.

$R_2 (A, B, D, E)$

$DF = \{ AD \rightarrow E, B \rightarrow AE, E \rightarrow BD \}$

$AD^+ = \{ A, D, E, B \} \quad \text{FNBC}$

$B^+ = \{ B, A, E, D \}$

$E^+ = \{ E, B, D, A \}$

Claves: (E) (B) (AD). FNBC.

El resultado de la descomposición es:  $R_1 (A, C)$

$R_2 (A, B, D, E)$

En el proceso no se ha perdido ninguna dependencia funcional.



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**1 de Junio de 2012**

La Ley 25/2007, de 18 de octubre, de conservación de datos relativos a las comunicaciones electrónicas y a las redes públicas de comunicaciones, obliga a los operadores de telecomunicaciones a retener determinados datos generados o tratados por los mismos, con el fin de posibilitar que dispongan de ellos, en el curso de una investigación, los agentes facultados (Cuerpos Policiales, Centro Nacional de Inteligencia, Vigilancia Aduanera, ...). Por tal motivo se ha decidido crear una base de datos que centralice toda la información relativa a las comunicaciones de voz a través de telefonía (fija o móvil). En el esquema de base de datos utilizado, los atributos se abrevian utilizando el siguiente convenio:

Atributo	Significado
C	Compañía operadora de telefonía.
D	Duración (en segundos) de la llamada.
DNI	DNI del usuario que dio de alta el número de teléfono.
F	Fecha de realización de la llamada.
H	Hora de realización de la llamada.
TF	Número de teléfono.
TFO	Número de teléfono origen (realiza la llamada).
TFD	Número de teléfono destino (recibe la llamada).

Las tablas utilizadas son:

**USUARIOS (DNI, TF, C)**

**SIGNIFICADO:** El usuario identificado por DNI dio de alta el teléfono TF con la compañía C.

**CLAVES:** (TF)

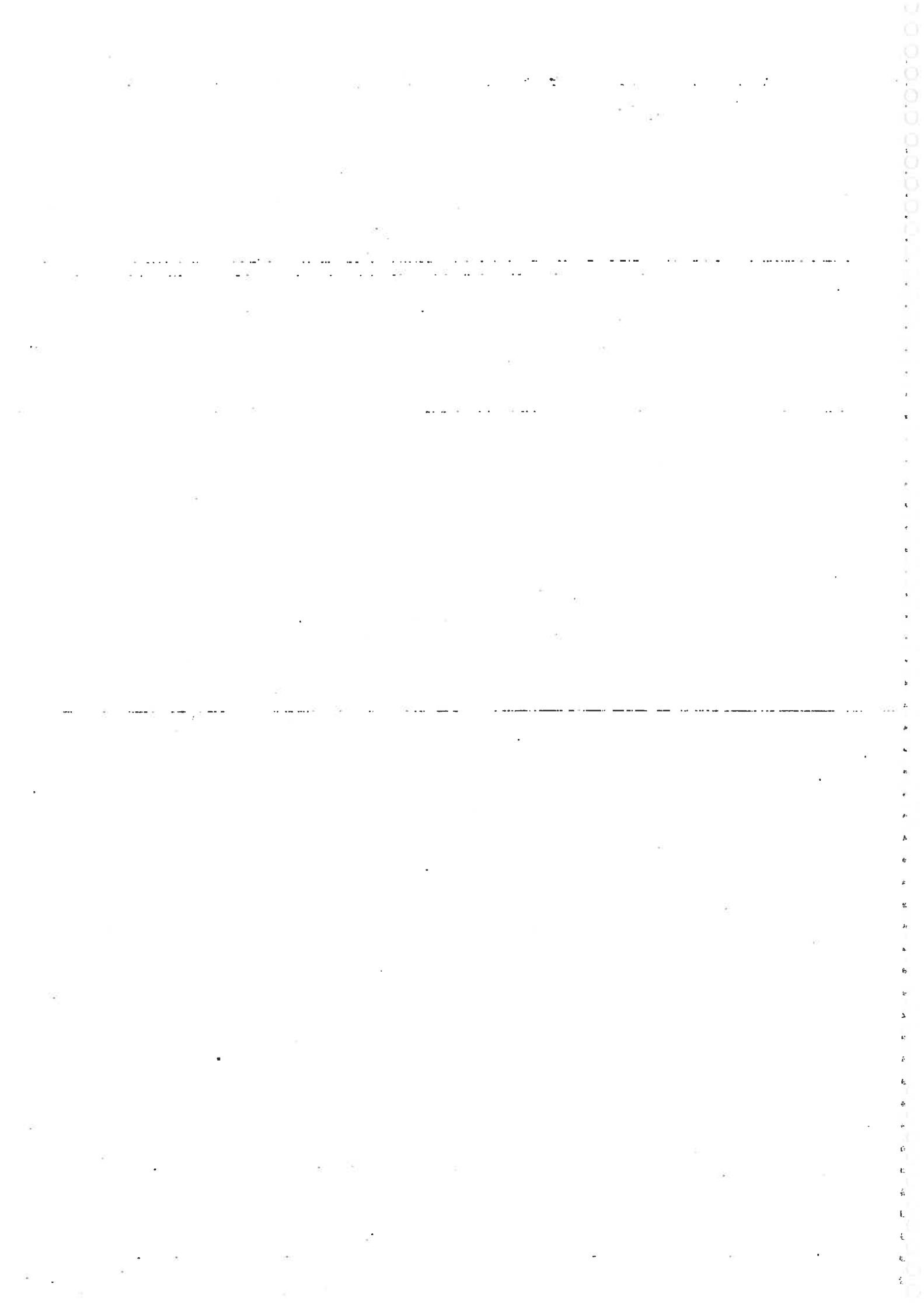
**LLAMADAS (TFO, TFD, F, H, D)**

**SIGNIFICADO:** El teléfono TFO realizó una llamada al teléfono TFD el día F, a la hora H, de duración D segundos.

**CLAVES:** (TFO, TFD, F, H)

Responer en álgebra relacional a las siguientes preguntas:

- Números de teléfonos a los que ha llamado el usuario con DNI 1111 con una duración superior a los 180 segundos.
  - Telefonos tales que todas las llamadas realizadas han sido al mismo número.
  - Usuarios que han recibido llamadas de números de todas las compañías.
  - Personas que han realizado llamadas a los mismos números que alguna otra persona.
- 2) Responder en cálculo relacional de t-uplas a las siguientes consultas:
- Telefonos que todavía no han realizado ni recibido ninguna llamada.
  - Usuarios que tienen todos sus teléfonos contratados en una única compañía.
  - Usuarios que sólo reciben llamadas en alguno de sus teléfonos.
  - Usuarios que realizan al menos una llamada diariamente.
- 3) Responder en cálculo relacional de dominios a las siguientes consultas:
- Usuarios que han llamado al teléfono 1 pero no al 2.
  - Usuarios que tienen al menos dos números contratados en una misma compañía.
  - Usuarios que sólo han contratado sus teléfonos con una única compañía.
  - Usuarios que han llamado a los mismos números que el usuario con DNI 1111.
- 4) Responder en SQL a las siguientes consultas:
- Teléfonos que sólo realizan llamadas.
  - Usuarios que han realizado más de 40 llamadas en un mismo día.
  - Compañía con mayor número de usuarios.
  - Usuarios tales que, al menos el 60% de todas las llamadas realizadas son a un mismo número.
- 5) Sea R(A, B, C; D, E) una relación con dependencias funcionales  $DF = \{E \rightarrow BD, AC \rightarrow E, D \rightarrow A, B \rightarrow C\}$ .
- ¿Está R en FNBC? Razona la respuesta.
  - Calcular  $AB^+$  aplicando el algoritmo de Ullman.
  - Probar usando los axiomas de Armstrong que  $BD \rightarrow E$  pertenece a  $DF^+$ .
  - Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?



1). Responder en álgebra relacional a las siguientes preguntas:

a. Números de teléfonos a los que ha llamado el usuario con dni '1111' con una duración superior a los 180 segundos.

$$\left. \begin{array}{l} P(TF) \wedge (DNI = '1111') \wedge (\text{USARIOS}) = A \\ P(TFO) \wedge (D > '180') \wedge (\text{LLAMADAS}) = B \end{array} \right\} \underline{A \cap B}$$

$$\{tu / (\exists u)(t[TF] = u[TF]) \wedge (u[DNI] = '1111') \wedge (\exists l)(u[TF] = l[TFO]) \wedge (l[D] > 180)\}$$

b. Teléfonos tales que todas las llamadas realizadas han sido al mismo número.

$$\text{LLAMADAS} = A \vdash B$$

$$P(A.TFO)(A) - P(B.TF) \wedge (A.TFD \neq B.TFD) \wedge (A.TFO = B.TFO) (A \times B)$$

$$\{tu / (\exists l)(t[TF] = l[TFO]) \wedge (\exists l_1)(l[TFO] = l_1[TFO]) \wedge (l[TFO] \neq l_1[TFO])\}$$

c. Usuarios que han recibido llamadas de números de todos las compañías.

$$P(DNI)$$

$$A = B = \text{USARIOS} \times \text{LLAMADAS}$$

$$P(A.DNI)(A) - P(B.DNI) \wedge (B.TFD = A.TF) \wedge (A.C = B.C) (A \times B)$$

TFO  $\rightarrow$  Realiza la llamada.

$\hookrightarrow$  Compañía de este teléfono.

$$P(C)(\text{USARIO}) = A$$

$$P(TFD, C) \wedge (TFO = TF) \wedge (\text{USARIOS} \times \text{LLAMADAS}) = B \quad \left\{ \frac{A}{B} = C \right.$$

$$P(DNI) \wedge (TF = TFD) \wedge (\text{USARIO} \times C)$$

2) - Responder en cálculo relacional de tuplas a las siguientes consultas:

a. Teléfonos que todavía no han realizado ni recibido ninguna llamada.

$$\text{dom}(l) = \text{LLAMADAS} \quad \text{dom}(u) = \text{USUARIOS} \quad \text{dom}(l1) = \text{LLAMADAS}$$

$$\{ t_{(u)} / (\exists u) (t[TF] = u[TF]) \wedge$$

$$\neg (\exists l) (l[TF0] = u[TF]) \wedge$$

$$\neg (\exists l1) (l1[TF0] = u[TF]) \}$$



b. Usuarios que tienen todos sus teléfonos contratados en una única compañía.

$$\text{dom}(u) = \text{dom}(u1) = \text{USUARIOS}$$

$$\{ t_{(u)} / (\exists u) (t[DN1] = u[DN1]) \wedge$$

$$\neg (\exists u1) (u[DN1] = u1[DN1]) \wedge (u[c] \neq u1[c]) \}$$

c. Usuarios que sólo reciben llamadas en alguno de sus teléfonos.

$$\text{dom}(u1) \cdot \text{dom}(u) = \text{USUARIOS}, \text{dom}(l) = \text{LLAMADAS}, \text{dom}(l1) = \text{LLAMADAS}$$

$$\{ t_{(u)} / (\exists u) (t[DN1] = u[DN1]) \wedge (\exists l) (u[TF] = l[TF0]) \wedge$$

$$\rightarrow (\exists u1) (u1[DN1] = u[DN1]) \wedge$$

$$\neg (\forall l1) (u1[TF] = l1[TF0]) \}$$



d. Usuarios que realizan al menos una llamada diariamente.

$$\text{dom}(u) = \text{USUARIO}, \text{dom}(l) = \text{dom}(l1) = \text{LLAMADAS}$$

$$\{ t_{(u)} / (\exists u) (t[DN1] = u[DN1]) \wedge$$

$$(\exists l) (l[TF0] = u[TF]) \wedge$$

$$(\forall l1) (l1[F] = l[F]) \}$$



5). Sea  $R = (A, B, C, D, E)$  una relación con dependencias funcionales:

$$DF = \{E \rightarrow BD, AC \rightarrow E, D \rightarrow A, B \rightarrow C\}$$

a. ¿Está  $R$  en FNBC? Razona la respuesta.

$$E^+ = \{E, B, D, C, A\} \quad \left. \right\} \text{Sí FNBC.}$$

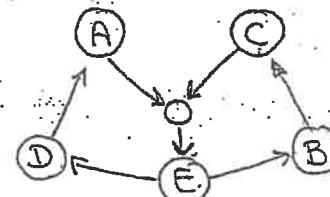
$$AC^+ = \{A, C, E, B, D\}$$

$$D^+ = \{D, A\} \quad \left. \right\} \text{No está en FNBC.}$$

$$B^+ = \{B, C\}$$

$$\rightarrow \text{Claves: } (E) (AC) (DB) (DE) (EB)$$

Mot.  $R$  no está en FNBC porque las dependencias funcionales ( $D \rightarrow A$ ) y ( $B \rightarrow C$ ), sus antecedentes no son superrelativas.



b. Calcular  $AB^+$  aplicando el algoritmo de Ullman.

$$AB^+ = \{A, B\} \quad | \quad B \rightarrow C$$

$$AB^+ = \{A, B, C\} \quad | \quad AC \rightarrow E$$

$$AB^+ = \{A, B, C, E\} \quad | \quad E \rightarrow BD$$

$$AB^+ = \{A, B, C, D, E\}$$

Algoritmo de Ullman:  $A^+ = A$

while ( $A^+$  cambie)

for each df  $X \rightarrow Y \in DF$

if  $X \subseteq A^+ \Rightarrow A^+ = A^+ \cup Y$

si el antecedente  
está en el cierre. añade el  
consecuente:

c. Probar, usando los axiomas de Armstrong que  $BD \rightarrow E$  pertenece a  $DF^+$ .

$$BD \rightarrow E$$

$$AC \rightarrow E \quad \left. \right\} \text{Transitiva}$$

$$AC \rightarrow E \quad | \quad E \rightarrow E \quad \left. \right\} \text{Transitiva}$$

$$E \rightarrow BD \quad \boxed{BD \rightarrow E}$$

$$D \rightarrow A \Rightarrow DB \rightarrow AB$$

$$B \rightarrow C \Rightarrow AB \rightarrow AC \quad | \quad AB \rightarrow E$$

$$AC \rightarrow E$$

$$\boxed{DB \rightarrow E}$$

d. Descomponer  $R$  en FNBC. ¿Se preservan las dependencias funcionales?

\*  $R_1 (D, A)$

$$DF = \{D \rightarrow A\}$$

Clave: (D). Sí FNBC.

\*  $R_2 (B, C, D, E)$

$$DF = \{E \rightarrow BD, DC \rightarrow E, B \rightarrow C\}$$

$$E^+ = \{E, B, D, C\}$$

$$DC^+ = \{D, C, E, B\}$$

$$B^+ = \{B, C\} \rightarrow \text{No FNBC.}$$

\*  $R_{21} (B, C)$

$$DF = \{B \rightarrow C\}$$

Clave (B). Sí FNBC.

\*  $R_3 (BDE)$

$$DF = \{E \rightarrow BD\}$$

Clave (E). Sí FNBC.

4) Responder en SQL a las siguientes consultas.

a - Teléfonos que sólo realizan llamadas.

SELECT TF

FROM USUARIO

WHERE NOT EXIST (SELECT TFD

FROM LLAMADAS

WHERE TF = TFD);

b - Usuarios que han realizado más de 40 llamadas en un mismo día.

SELECT DNI

FROM USUARIO U1

WHERE 40 < ANY (SELECT COUNT (TFO)

FROM LLAMADAS L1

WHERE (U1.TF = L1.TFO)

GROUP BY TFO);

c - Compañía con mayor número de usuarios.

SELECT C

FROM USUARIO

GROUP BY C

HAVING COUNT (DNI) >= ALL (SELECT COUNT (DNI)

FROM USUARIO

GROUP BY C);

d - Usuarios tales que al menos el 60% de todas las llamadas realizadas son a un mismo número.

\* ?

SELECT DNI, TFD

FROM USUARIO U1, LLAMADAS L1

WHERE (U1.TF = L1.TFO)

GROUP BY TFD

HAVING (COUNT (TFO)\*0'6) >= (SELECT COUNT (TFO)

FROM LLAMADAS L2

GROUP BY TFD);



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**20 de Enero de 2012**

La empresa de transportes TITSA tiene centralizada toda la información relativa a las rutas y turnos que realizan sus conductores en las distintas guaguas que posee. En el esquema de base de datos utilizado, los atributos se abrevian utilizando el siguiente convenio:

Atributo	Significado
CR	Código de ruta: 1, 2, 3, ...
DNI	DNI del conductor
DS	Día de la semana: lunes, martes, ...
MAR	Marca de la guagua
MAT	Matrícula de la guagua
MOD	Modelo de la guagua
NP	Número de plazas de la guagua: 45, 35, ...
TUR	Turno: Mañana, tarde o noche

Las tablas utilizadas son:

**VEHICULOS** (MAT, MAR, MOD, NP)

**SIGNIFICADO:** La guagua con matrícula MAT es de la marca MAR, modelo MOD y tiene NP plazas.

**CLAVES:** (MAT)

**PLANIFICACIÓN** (DNI, DS, TUR, MAT, CR)

**SIGNIFICADO:** El conductor con dni DNI tiene asignado el día DS, el turno TUR, la guagua con matrícula MAT y la ruta CR.  
**CLAVES:** (DNI, DS)

Responder en álgebra relacional a las siguientes preguntas:

- Rutas que no se realizan los domingos.
- Conductores que a lo largo de la semana trabajan en todos los turnos.
- Matrícula de la guagua con mayor capacidad que realiza la ruta 1.
- Conductores que han trabajado los mismos días de la semana que algún otro conductor.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Rutas que tienen asignadas más de una guagua los lunes.
- Guaguas que, a lo sumo, están asignadas a dos rutas distintas.
- Rutas que tienen asignadas guaguas de todas las marcas.
- Conductores que hacen al menos las mismas rutas que el conductor 1111.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

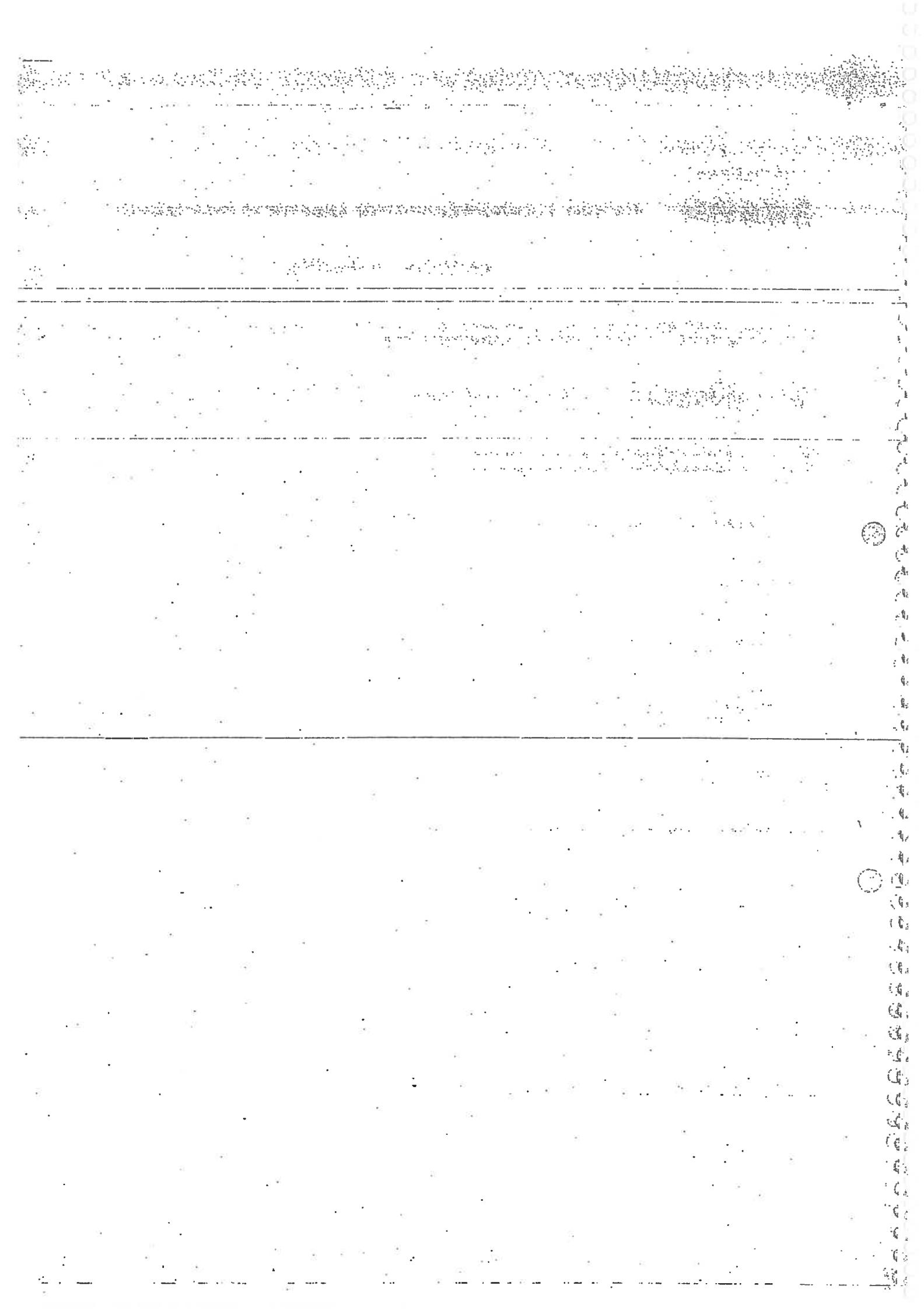
- Conductores que tienen asignadas las rutas 1 y 2.
- Rutas que están asignadas a una única guagua.
- Marcas de guaguas tales que todos sus modelos tienen más de 40 plazas.
- Rutas que se realizan los mismos días de la semana que alguna otra ruta.

4) Responder en SQL a las siguientes consultas:

- Conductores que no tienen asignada la ruta 1.
- Número medio de plazas asignadas a cada ruta diariamente.
- Conductores que tienen asignada al menos una guagua de cada marca.
- Rutas tales que, al menos el 60% de las guaguas asignadas a ellas, son de una misma marca.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {B → AC, DC → E, E → DB, A → D}.

- ¿Está R en FNBC? Razóna la respuesta.
- Calcular AC+ aplicando el algoritmo de Ullman.
- Probar usando los axiomas de Armstrong que AC → B pertenece a DF+.
- Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?



EXAMEN BASE DE DATOS (20 - Enero de 2012)

Las tablas utilizadas son:

VEHICULOS (MAT, MAR, MOD, NP)

PLANIFICACION (DNI, DS, TUR, MAT, CR)

→ La guagua con matrícula MAT es de 6  
marca MAR, modelo MOD y tiene  
NP plazas.

Lo El conductor con DNI tiene asignado el día DS, el turno TUR, la guagua  
con matrícula MAT y la ruta CR.

1) Responder en álgebra relacional a las siguientes preguntas:

a) Rutas que no se realizan los domingos:

$$P(CR) \text{ (PLANIFICACION)} - P(CR) S (DS = 'Domingo') \text{ (PLANIFICACION)}$$

Proyecta todos los códigos de ruta, menos los proyectados que  
caen en domingo;

En cálculo de t-uplas sería:  $\text{dom}(p1) = \text{dom}(p2) = \text{PLANIFICACION}$

$$\{ t_1 / (3p_1) (t_1[CR] = p_1[CR]) \wedge$$

$$\neg (\exists p_2) (p_2[DS = 'Domingo']) \wedge (p_2[CR] = p_1[CR]) \}$$

b) Conductores que a lo largo de la semana trabajan en todos los turnos.

$$A = P(TUR) \text{ (PLANIFICACION)}$$

$$B = P(DNI, TUR) \text{ (PLANIFICACION)}$$

$B/A \rightarrow$  El resultado del cociente es DNI.  
Si lo multiplicamos por A el resultado es B.

$$DNI \times A \subseteq B$$

- Cálculo relacional de dominios:

$$\{ \langle dñi \rangle / (\forall \text{tur}) (\exists ds, mat, cr) (\langle dñi, ds, tur, mat, cr \rangle \in \text{PLANIFICACION}) \}$$

- SQL:

```
SELECT DNI
FROM PLANIFICACION
GROUP BY DNI
```

HAVING COUNT(DISTINCT TUR) = (SELECT COUNT(DISTINCT TUR)

FROM PLANIFICACION) → cuenta los distintos turnos (110)

c) Matrícula de la guagua con mayor capacidad que realiza la ruta 1.

$$A = P(\text{MAT}, \text{NP}) \text{ S} (\text{CR} = 1) \text{ (VEHICULOS * PLANIFICACION)}$$

$$B = A$$

$$P(\text{MAT})(A) = P(A, \text{MAT}) (\text{S}(A, \text{NP} < B, \text{NP})) (A \times B)$$

Arreglo: Si la clave primaria fuese  $(\text{MAT}, \text{NP})$ . Tambien funciona si la clave es sólo  $\text{MAT}$ .  $A = A \cap B \subseteq A$

Forma de hacerlo más robusta que la anterior. Admite claves más complejas.

$$C = P(\text{NP})(A) - P(\text{NP}) \text{ S} (A, \text{NP} < B, \text{NP}) (A \times B)$$

$$P(\text{MAT})(A * C)$$

- Cálculo relacional de t-uplas (3 formas de hacerlo):

$$\text{dom}(p_1) = \text{dom}(p_2) = \text{PLANIFICACION} \quad \text{dom}(v_1) = \text{dom}(v_2) = \text{VEHICULOS}$$

$$\{ t_{v1} / (\exists p_1, v_1) ((t[\text{MAT}] = p_1[\text{MAT}]) \wedge (p_1[\text{MAT}] = v_1[\text{MAT}]) \wedge (p_1[\text{CR}] = 1)) \}$$

$$\textcircled{1} \quad \wedge (\forall p_2, v_2) ((p_2[\text{MAT}] \neq v_2[\text{MAT}]) \vee (p_2[\text{CR}] = 1) \vee (v_2[\text{NP}] <= v_1[\text{NP}])) \}$$

$$\textcircled{2} \quad \neg (\exists p_2, v_2) ((p_2[\text{MAT}] = v_2[\text{MAT}]) \wedge (p_2[\text{CR}] = 1) \wedge (v_2[\text{NP}] > v_1[\text{NP}])) \}$$

$$\textcircled{3} \quad (\forall p_2) ((p_2[\text{CR}] <> 1) \vee (\exists v_2) ((v_2[\text{MAT}] = p_2[\text{MAT}]) \wedge (v_2[\text{NP}] <= v_1[\text{NP}])) \}$$

d) Conductores que han trabajado los mismos días de la semana que algún otro conductor.

$$\textcircled{4} \quad P(\text{DNI} \neq, \text{DS}) \text{ (PLANIFICACION)} = A \neq B \neq C$$

$$P(\text{DNI})(\text{PLANIFICACION}) - P(A, \text{DNI}) \text{ S} (A, \text{DNI} \neq B, \text{DNI}) \wedge (A, \text{DS} \neq B, \text{DS}) \\ (A \times B)$$

→ Ejercicio 2 en álgebra relacional

a) Rutas que tienen asignadas más de una guagua los línes.

$$P_1 = P_2 = \text{PLANIFICACION} * \text{VEHICULOS}$$

$$P(P_1.CR) \cap (S(P_1.CR = P_2.CR) \wedge (P_1.DS = 'línes') \wedge (P_2.DS = 'línes')) \wedge$$

$$(P_1.HAT \neq P_2.HAT))$$

selecciona las matrículas diferentes, puesto que queremos más de una guagua.

b) Guaguas que, a lo sumo, están asignadas a dos rutas distintas.

$$P_1 = P_2 = P_3 = \text{PLANIFICACION} * \text{VEHICULOS}$$

$$A = P(P_1.HAT) S (P_1.CR = P_2.CR = P_3.CR) \wedge (P_1.HAT = P_2.HAT = P_3.HAT) \\ (\text{PLANIFICACION} * \text{VEHICULOS}) \rightarrow (P_1 \times P_2 \times P_3)$$

$$P(HAT)(\text{PLANIFICACION} * \text{VEHICULOS}) - P(HAT)(A)$$

Todas las guaguas

Guaguas que tienen

c) Rutas que tienen asignadas guaguas de todas las marcas.

$$A = P(MAR)(\text{VEHICULOS})$$

$$B = P(CR, MAR)(\text{PLANIFICACION} * \text{VEHICULOS})$$

B/A

d) Conductores que hacen al menos las mismas rutas que el conductor con dni = 1111.

$$A = P(CR) S (DNI = '1111') (\text{PLANIFICACION})$$

$$B = P(DNI, CR)(\text{PLANIFICACION})$$

B/A

→ Ejercicio 3 en álgebra relacional:

a) Conductores que tienen asignadas las rutas 1 y 2.

$$P(DNI) \left( s (CR \neq '1') \wedge (CR \neq '2') \right) (\text{PLANIFICACION}) = A$$

$$P(DNI) (\text{PLANIFICACION}) - P(DNI) (A)$$

b) Rutas que están asignadas a una única gragea.

$$P_1 = P_2 = \text{PLANIFICACION} * \text{VEHICULOS}$$

$$A = P(CR) \left( s (P_1.CR = P_2.CR) \wedge (P_1.NAT \neq P_2.NAT) \right) (P_1 \times P_2)$$

$$P(CR) (\text{PLANIFICACION}) - P(A)$$

Misma ruta pero distinta gragea

c) Marcas de grageas tales que todos sus modelos tienen más de 40 plazas.

$$A = P(NAR, MOD, NP) (\text{VEHICULOS})$$

$$B = P(MOD, NP) \left( s (NP > 40) \right) (\text{VEHICULOS})$$

$$\left. \begin{array}{l} | \\ A / B \\ \hline \end{array} \right\}$$

d) Rutas que se realizan los mismos días de la semana que alguna otra ruta.

(\*)

4) Responder en SQL a las siguientes consultas:

a.- Conductores que no tienen asignada la ruta 1.

```
SELECT DNI
FROM PLANIFICACION
WHERE NOT (CR = '1');
```

b.- Número medio de plazas asignadas a cada ruta diariamente.

```
SELECT AVG(NP)
FROM VEHICULOS V, PLANIFICACION P
WHERE (V.MAT = P.MAT);
```

c.- Conductores que tienen asignada al menos una guagua de cada marca.

```
SELECT DNI
FROM PLANIFICACION P, VEHICULOS V
WHERE (P.MAT = V.MAT)
GROUP BY DNI
HAVING (COUNT(DISTINCT(MAR))) = (SELECT COUNT(DISTINCT(MAR))
                                    FROM VEHICULOS);
```

d.- Rutas tales que, al menos el 60% de las guaguas asignadas a ellas, son de una misma marca.

```
SELECT CR, MAR
FROM PLANIFICACION P1, VEHICULOS V1
GROUP BY MAR
HAVING (COUNT(MAR) * 0'6) >= (SELECT COUNT(MAR)
                                    FROM VEHICULOS V2, PLANIFICACION P2
                                    WHERE (V2.MAR = V1.MAR)
                                    AND (P1.CR = P2.CR)
                                    GROUP BY MAR);
```

5) Sea  $R(A, B, C, D, E)$  una relación con dependencias funcionales

$$DF = \{B \rightarrow AC, DC \rightarrow E, E \rightarrow DB, A \rightarrow D\}$$

a. ¿Está  $R$  en FNBC? Razona la respuesta.

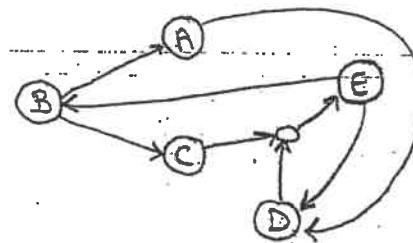
$$B^+ = \{B, A, C, D, E\} \text{ FNBC. Superclave.}$$

$$DC^+ = \{D, C, E, B, A\} \quad " "$$

$$E^+ = \{E, D, B, A, C\} \quad " "$$

$$A^+ = \{A, D\} \text{ No FNBC.}$$

$$\text{Claves: } (B)(DC)(E)(AB)(AE)(AC)$$



No está en FNBC porque en la dependencia funcional  $(A \rightarrow D)$  el antecedente no es superclave.

b. Calcular  $AC^+$  aplicando el algoritmo de Ullman.

$$AC^+ = \{A, C\} \quad A \rightarrow D$$

$$AC^+ = \{A, C, D\} \quad DC \rightarrow E$$

$$AC^+ = \{A, C, D, E\} \quad E \rightarrow DB$$

$$AC^+ = \{A, B, C, D, E\} \rightarrow \text{Se completa el cierre } AB^+, \text{ ya no cambia.}$$

c. Probar usando los axiomas de Armstrong que  $AC \rightarrow B$  pertenece a  $DF^+$ .

$$\boxed{AC \rightarrow B}$$

$$DC \rightarrow E \quad \left. \begin{array}{l} DC \rightarrow E \\ \Rightarrow AC \rightarrow DC \end{array} \right\} \text{ Pseudotransitiva.}$$

$$A \rightarrow D \quad \left. \begin{array}{l} A \rightarrow D \\ \Rightarrow AC \rightarrow DC \end{array} \right\} \text{ AC} \rightarrow E \quad \left. \begin{array}{l} AC \rightarrow E \\ E \rightarrow DB \end{array} \right\} \text{ Transitiva. Descomposición: } AC \rightarrow DB \Rightarrow \boxed{AC \rightarrow D}$$

$$\boxed{AC \rightarrow B}$$

d. Descomponer  $R$  en FNBC. ¿se preservan las dependencias funcionales?

$$R_1 (A \rightarrow D)$$

$$R_2 (A, B, C, E)$$

$$DF_1^+ = \{A \rightarrow D\}$$

$$DF_2^+ = \{B \rightarrow AC, E \rightarrow AB, AC \rightarrow E\}$$

$$\text{Clave (A). FNBC.}$$

$$B^+ = \{B, A, C, E\}$$

$$E^+ = \{E, A, B, C\}$$

$$AC^+ = \{A, C, E, B\}$$

$$\text{Claves (B) (E) (AC). FNBC.}$$

si FNBC.



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**11 de Enero de 2012**

La ULL tiene centralizada toda la información relativa a la docencia que imparten sus profesores en las distintas titulaciones. En el esquema de base de datos propuesto, los atributos que se utilizan se abrevian utilizando el siguiente convenio:

Atributo	Significado
CA	Código de asignatura
CD	Código de departamento
CR	Curso (1, 2, 3, ...)
CT	Código de Titulación (T1, T2, ...)
DNI	DNI del profesor
HP	Horas de prácticas asignadas (0, 1, 2, ...)
HT	Horas de teoría asignadas (0, 1, 2, ...)
NA	Nombre de asignatura (BD, MSS, ...)
NP	Nombre de profesor (P1, P2, ...)

Las tablas utilizadas son:

**PROFESOR** (DNI, NP, CD).

**SIGNIFICADO:** El profesor con dni DNI se llama NP y pertenece al departamento CD.

**CLAVES:** (DNI)

**ASIGNATURA** (CT, CA, NA, CR)

**SIGNIFICADO:** La asignatura con código CA de la titulación CT se llama NA y se imparte en el curso CR.

**CLAVES:** (CT, CA)

**DOCENCIA** (CT, CA, DNI, HT, HP)

**SIGNIFICADO:** En la asignatura con código CA, de la titulación CT, el profesor con dni DNI imparte HT horas de teoría y HP horas de prácticas, verificándose que  $(HT+HP) > 0$ .

**CLAVES:** (CT, CA, DNI)

1) Responder en álgebra relacional a las siguientes consultas:

- a) Profesores que no dan clases en 3º.
- b) Profesores que sólo dan clase en una única titulación.
- c) Profesores que dan clases en todos los cursos de la titulación T1.
- d) Profesores que dan clases en todos los cursos de alguna titulación.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Profesores que imparten asignaturas en la titulación T1 pero no en T2.
- b) Profesores que sólo imparten clases de teoría en T1.
- c) Profesores que dan clases en al menos las mismas titulaciones que el profesor P1.
- d) Profesores que dan clases en todos los cursos de alguna titulación.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Profesores que imparten más de una asignatura en T1.
- b) Asignaturas que son impartidas por un único profesor.
- c) Profesores que sólo imparten sus clases de teoría en 3º o 4º.
- d) Profesores que dan clases en al menos las mismas titulaciones que algún otro profesor.

4) Responder en SQL a las siguientes consultas:

- a) Profesores que sólo dan clases en primero.
- b) Número medio de horas impartidas por los profesores del DEIOC.
- c) Profesor que da más horas de teoría en una misma titulación.
- d) Profesores tales que al menos el 60% de su docencia está asignada en una única asignatura.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {D → BC, BE → A, C → E, A → D}.

- a) Calcular todas las claves.
- b) Calcular CB+ aplicando el algoritmo de Ullman.
- c) Probar formalmente (a través de los axiomas de Armstrong) que D → A pertenece a DF+.
- d) Descomponer R en FNBC. ¿Se preservan las dependencias funcionales?

(11/Enero/2012)

4) Responder en SQL a las siguientes consultas:

a. Profesores que sólo dan clase en primero.

```
SELECT DNI
```

```
FROM DOCENCIA D
```

```
WHERE NOT EXISTS (SELECT CA, CT
```

```
FROM ASIGNATURA A
```

```
WHERE (CR <> 1));
```

AND (D.CT = A.CT) ?

AND (D.CA = A.CA);

b. Número medio de horas impartidas por los profesores del DEIOC.

```
SELECT AVG (SUM (HP+HT))
```

```
FROM DOCENCIA
```

```
WHERE DNI IN (SELECT DNI, FROM PROFESOR
```

```
WHERE (CD = 'DEIOC'));
```

c. Profesor que da más horas de teoría en una misma titulación.

```
SELECT DNI
```

```
FROM DOCENCIA D1
```

```
GROUP BY CT
```

```
HAVING SUM (HT) >= ALL (SELECT SUM (HT))
```

```
FROM DOCENCIA D2
```

```
WHERE (D1.CT = D2.CT)
```

```
GROUP BY CT);
```

d. Profesores tales que al menos el 60% de su docencia esta asignada en una única asignatura.

```
SELECT CA, DNI
```

```
FROM DOCENCIA D1
```

```
GROUP BY DNI
```

```
HAVING (SUM (HT+HP) * 0'6) >= (SELECT (SUM (HT+HP)))
```

```
FROM DOCENCIA D2
```

```
WHERE (D1.CA = D2.CA)
```

```
GROUP BY DNI);
```

?



**Bases de Datos**  
**Escuela Técnica Superior de Ingeniería Informática**  
**Universidad de La Laguna**  
**12 de Julio de 2013**

Un programa de cocina cuenta con una base de datos que permite almacenar electrónicamente la información relativa a las recetas de cocina y los gustos y disponibilidades de ingredientes de los usuarios. En el esquema utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
CD	Cantidad disponible del ingrediente ( $>0$ ).
CR	Cantidad requerida del ingrediente ( $>0$ ).
I	Ingrediente.
P	Persona.
PL	Plato.

Las tablas utilizadas son:

**RECETA(PL, I, CR)**

**SIGNIFICADO:** El plato PL necesita, para su preparación, CR unidades del ingrediente I.

**CLAVE PRIMARIA:** (PL, I)

**GUSTA (P, PL)**

**SIGNIFICADO:** A la persona P le gusta el plato PL.

**CLAVE PRIMARIA:** (P, PL)

**DISPONE (P, I, CD)**

**SIGNIFICADO:** La persona P dispone de CD unidades del ingrediente I.

**CLAVE PRIMARIA:** (P, I)

Nota: Una persona puede preparar un plato si dispone de todos los ingredientes y las cantidades necesarias, según indica la receta.

1) Responder en álgebra relacional a las siguientes consultas:

- Ingredientes del plato PL1 de los que no dispone la persona P1.
- Platos que gustan a una única persona.
- Platos que requieren, al menos, los mismos ingredientes que el plato PL1.
- Personas que pueden preparar alguno de los platos que les gustan.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Platos que gustan, simultáneamente, a las personas P1 y P2.
- Ingredientes que se utilizan en todos los platos.
- Ingrediente que se utiliza en mayor cantidad en el plato PL1.
- Personas que disponen de alguno de los ingredientes de cada plato que les gusta.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- Platos que requieren al menos 2 ingredientes distintos.
- Platos que no requieren el ingrediente II.
- Ingredientes que sólo intervienen en un único plato.
- Platos que requieren al menos todos los ingredientes que requiere algún otro plato.

4) Responder en SQL a las siguientes consultas:

- Platos que no gustan a nadie.
- Número medio de platos en que interviene cada ingrediente.
- Plato en que interviene en mayor cantidad el ingrediente II.
- Ingredientes de los que disponen al menos al 70% de las personas.

5) Responder en SQL a las siguientes peticiones:

- Decrementa en un 10% las cantidades de los ingredientes disponibles por la persona P1.
- Añade un atributo GS a la tabla GUSTA que indique el grado de satisfacción: 'mucho', 'poco' y 'nada'.
- Garantiza que cualquier valor del atributo PL en la tabla GUSTA exista previamente en la tabla RECETA.
- Implementa que si se elimina completamente un plato de RECETA se propague el borrado en cascada a la tabla GUSTA.

1) Responder en álgebra relacional a las siguientes consultas:

a.- Ingredientes del plato PL1 de los que no dispone la persona P1.

$$\left. \begin{array}{l} A = P(I) S(PL = 'PL1') (RECETA * DISPONE) \\ B = P(I) S(P = 'P1') (RECETA * DISPONE) \end{array} \right\} \underline{A - B}$$

b.- Platos que gustan a una única persona.

$$A = B = GUSTA$$
$$P(A.PL)(GUSTA) - P(B.PL) S(A.P \neq B.P) (A \times B)$$

$$P(A.PL) S(A.PL = B.PL) \wedge (A.P = B.P) (A \times B)$$

c.- Platos que requieren, al menos, los mismos ingredientes que el plato PL1.

$$P(PL)$$

(12/Julio/2013)

⑤ a) UPDATE DISPONE

```
SET CD = CD + (CD * 0.1)  
WHERE (P = 'P1');
```

b) ALTER TABLE GUSTA

```
ADD (GS CHAR(5));
```

c) CREATE ASSERTION PL6

CHECK NOT EXISTS (SELECT PL

```
FROM GUSTA  
WHERE PL NOT IN
```

```
(SELECT PL  
FROM RECETA))
```

d) CREATE TRIGGER DLT

AFTER DELETE ON RECETA

REFERENCING OLD AS ofila

FOR EACH ROW

WHEN (SELECT COUNT(RI)

```
FROM RECETA R  
WHERE (R.PL = ofila.PL)  
GROUP BY R.PL) = 0
```

BEGIN

```
DELETE FROM GUSTA G  
WHERE G.PL = ofila.PL;
```

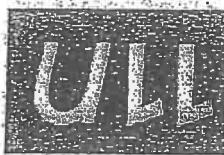
END;

Rec. N. → Tiene el mismo cliente que el primero.

{ -Mismo anteced. → Unión.  
-Elim. a. fabos.

Trig. → hace algo

Aserfo → Prohibición. Impide.



Bases de Datos  
Escuela Técnica Superior de Ingeniería Informática  
Universidad de La Laguna  
4 de Junio de 2013

Con el fin de realizar un seguimiento sobre los precios a los que se ofertan los diferentes productos disponibles en los mercados de alimentos de la isla se ha creado una base de datos. En el esquema utilizado los atributos se abrevian según el siguiente convenio:

Atributo	Significado
CAT	Categoría del producto: refrescos, vinos, lácteos, embutidos, ...
CD	Ciudad donde está ubicado el mercado: CD1, 'Santa Cruz de Tenerife', 'La Laguna', ...
CNT	Cantidad (número de unidades/kilos/litros) disponible del producto: 0, 1, 2, ...
IM	Identificador del mercado: M1, 'Nuestra Señora de África', ...
IP	Identificador del producto: P1, 'Plátanos', ...
PRE	Precio en Euros de una unidad/kilo/litro del producto

Las tablas utilizadas son:

**MERCADOS (IM, CD)**

**SIGNIFICADO:** El mercado IM está ubicado en la ciudad CD.

**CLAVE PRIMARIA:** (IM)

**PRODUCTOS (IP, CAT)**

**SIGNIFICADO:** El producto IP pertenece a la categoría CAT.

**CLAVE PRIMARIA:** (IP)

**OFERTAS (IM, IP, PRE, CNT)**

**SIGNIFICADO:** El mercado IM oferta el producto IP a un precio de PRE Euros y dispone de CNT unidades del mismo. Un valor de CNT 0 indica que actualmente dicho mercado no dispone del producto en cuestión.

**CLAVE PRIMARIA:** (IM, IP)

1) Responder en álgebra relacional a las siguientes consultas:

- a) Mercados que actualmente ofrecen productos lácteos y también embutidos.
- b) Productos que siempre se ofrecen con un precio superior a 5 Euros la unidad.
- c) Ciudades que con el conjunto de sus mercados disponen de todos los productos.
- d) Productos que están disponibles en todos los mercados de una misma ciudad.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Mercados que disponen de al menos dos productos distintos.
- b) Mercados que sólo ofrecen productos lácteos.
- c) Producto más barato disponible en el mercado M1.
- d) Mercados que ofrecen productos de al menos las mismas categorías que algún otro mercado.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Mercados que ofrecen de dos productos distintos de una misma categoría.
- b) Mercados que no ofrecen el producto P1.
- c) Mercado que ofrece el producto P1 más barato.
- d) Mercados tales que los precios de algunos de sus productos disponibles son los más baratos que se ofrecen en su ciudad.

4) Responder en SQL a las siguientes consultas:

- a) Mercados que sólo ofrecen vinos.
- b) Número medio de productos distintos que se ofrecen en cada ciudad.
- c) Producto lácteo más barato disponible en el mercado M1.
- d) Mercados tales que al menos el 60% de los productos que ofrecen corresponden a la categoría 'embutidos'.

5) Responder en SQL a las siguientes peticiones:

- a) Incrementa en un 10% el precio de todos los productos lácteos del mercado M1.
- b) Crea una vista con los productos disponibles y su precio más barato por cada ciudad.
- c) Implementa una regla que obligue a que cada ciudad tenga al menos dos mercados distintos.
- d) Impide que el precio de un producto pueda ser decrementado en más de un 50% respecto a su valor anterior.

(5)

a) UPDATE OFERTAS

SET PRE = PRE + (PRE \* 0'1)

WHERE (IM = 'M1')

AND IP IN (SELECT IP

FROM PRODUCTOS

WHERE CAT = 'Lacteos');

b) CREATE VIEW V1

AS (SELECT IP, PRE, CD

FROM OFERTAS NATURAL JOIN MERCADOS A

WHERE (CNT &gt; 0) → Disponibles.

GROUP BY CD

HAVING PRE = (SELECT MIN(PRE)

FROM OFERTAS B

WHERE (B.CD = A.CD))

→ Productos disponibles y su precio mas barato por cada ciudad.

\*)

c) Regla que obligue a que cada ciudad tenga al menos dos mercados distintos.

CREATE TRIGGER T1

AFTER INSERT ON MERCADOS

FOR EACH ROW

WHEN (SELECT CD

FROM MERCADOS M1, M2

WHERE (M1.IM &lt;&gt; M2.IM)

{ c?

BEGIN

END

d) CREATE ASSERTION AS 1

CHECK NOT EXIST (SELECT \*

FROM OFERTAS O1

GROUP BY PRE

HAVING ((PRE \* 0'5) + PRE) > ALL (SELECT PRE  
FROM OFERTAS O2  
WHERE (O1.PRE ≠ O2.PRE))

c precio anterior?



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**30 de Junio de 2000**

Un curso sobre cocina utiliza una base de datos que tiene el siguiente esquema:

Abreviaremos los atributos utilizados por:

Atributo	Significado
C	CANTIDAD
CD	CANTIDAD DISPONIBLE
I	INGREDIENTE
P	PERSONA
PL	PLATO

Las tablas son:

**RECETA(PL, I, C)**

**SIGNIFICADO:** El plato PL necesita, para su preparación, C gramos del ingrediente I.

**CLAVE PRIMARIA:** (PL, I)

**GUSTA(P, PL)**

**SIGNIFICADO:** A la persona P le gusta el plato PL.

**CLAVE PRIMARIA:** (P, PL)

**DISPONE(P, I, CD)**

**SIGNIFICADO:** La persona P dispone de CD gramos del ingrediente I.

**CLAVE PRIMARIA:** (P, I)

**Nota:** Una persona puede preparar un plato si dispone de todos los ingredientes y las cantidades necesarias, según indica la receta.

1) Responder en álgebra relacional a las siguientes preguntas:

- a) Ingredientes del plato PL1 de los que dispone en cantidad suficiente la persona P1.
- b) Platos que gustan a más de una persona.
- c) Ingrediente que interviene en mayor proporción en el plato PL1.
- d) Ingredientes que intervienen con, al menos, una cantidad de 100 gramos, en todos los platos.
- e) Personas tales que podrían preparar cualesquiera de los platos que les gustan.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Platos que gustan a la persona P1 y para los que dispone de alguno de sus ingredientes en cantidad suficiente.
- b) Personas a las que no les gusta ninguno de los platos que les gusta a la persona P1.
- c) Platos que gustan a todo el mundo.
- d) Personas que pueden preparar el plato PL1.
- e) Parejas de platos distintos que utilizan, exactamente, los mismos ingredientes.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Platos que no utilizan ni la cebolla ni el ajo.
- b) Ingredientes que se utilizan de forma exclusiva en el plato PL1.
- c) Ingrediente que se utiliza en mayor cantidad en el plato PL1.
- d) Platos que puede preparar la persona P1.
- e) Personas que sólo pueden preparar un único plato.

4) Responder en SQL a las siguientes consultas:

- a) Platos que no gustan a P1.
- b) Número medio de platos que gustan a las personas.
- c) Plato con mayor cantidad de ingredientes distintos.
- d) Platos que gustan a todo el mundo.
- e) Personas que disponen, al menos, del 60% de los ingredientes distintos (sin tener en cuenta las cantidades) que intervienen en algún plato.

1) Responder en álgebra relacional a las siguientes consultas:

a. Ingredientes del plato PL1 de los que dispone en cantidad suficiente la persona P1.

$$A = P(I, C) \wedge (PL = 'PL1') \text{ (RECETA)}$$

$$P(A.I) \wedge (AI = I) \wedge (CD \geq A.C) \wedge (P = 'P1') \text{ (DISPONE)}$$

b. Platos que gustan a más de una persona.

$$A = B = GUSTAN$$

$$P(A.PL) \wedge (A.PL = B.PL) \wedge (A.P \neq B.P) \text{ (AxB)}$$

c. Ingrediente que interviene en mayor proporción en el plato PL1.

$$A = B = RECETA$$

$$P(A.I) \wedge (A.C > B.C) \wedge (A.PL = B.PL) \wedge (B.PL = 'PL1') \text{ (AxB)}$$

d. Ingredientes que intervienen con, al menos, una cantidad de 100 gramos, en todos sus platos.

$$RECETA = (PL, I, C)$$

$$A = B = RECETA$$

$$P(A.I) \wedge (A.I = B.I) \wedge (A.PL \neq B.PL) \wedge (A.C \geq '100gr') \wedge (B.C \geq '100gr') \text{ (AxB)}$$

e. Personas tales que podrían preparar cualesquiera de los platos que les gustan.



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**20 de Junio de 2000**

Para gestionar la demanda ciudadana de actividades ofrecidas por las diferentes instituciones oficiales y privadas, se diseñó una base de datos que tiene el siguiente esquema:

Los atributos que se utilizan los vamos a abreviar utilizando la siguiente tabla:

Atributo	Significado
A	ACTIVIDAD
C	CENTRO
P	PERSONA
PR	PRECIO
T	TIPO

Las tablas son:

**TIPO (A, T)**

SIGNIFICADO: La actividad A es de tipo T. Ej: El senderismo es una actividad de ocio (y deportiva).

CLAVE PRIMARIA: (A, T)

**ORGANIZA (C, A, PR)**

SIGNIFICADO: El centro C organiza la actividad A y cobra, por persona, un precio PR.

CLAVE PRIMARIA: (C, A)

**GUSTA (P, A)**

SIGNIFICADO: A la persona P le gusta la actividad A.

CLAVE PRIMARIA: (P, A)

**PARTICIPA (P, A, C)**

SIGNIFICADO: La persona P participa en la actividad A organizada por el centro C.

CLAVE PRIMARIA: (P, A, C)

CLAVES AJENAS: (A, C)

1) Responder en álgebra relacional a las siguientes preguntas:

- Centros que sólo organizan actividades culturales o deportivas.
- Personas que no participan en actividades que no les gustan.
- Personas a las que gustan todas las actividades deportivas organizadas por el centro C1.
- Actividad más cara organizada por el centro C1.
- Personas tales que todas las actividades que les gustan son organizadas por, al menos, un mismo centro.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

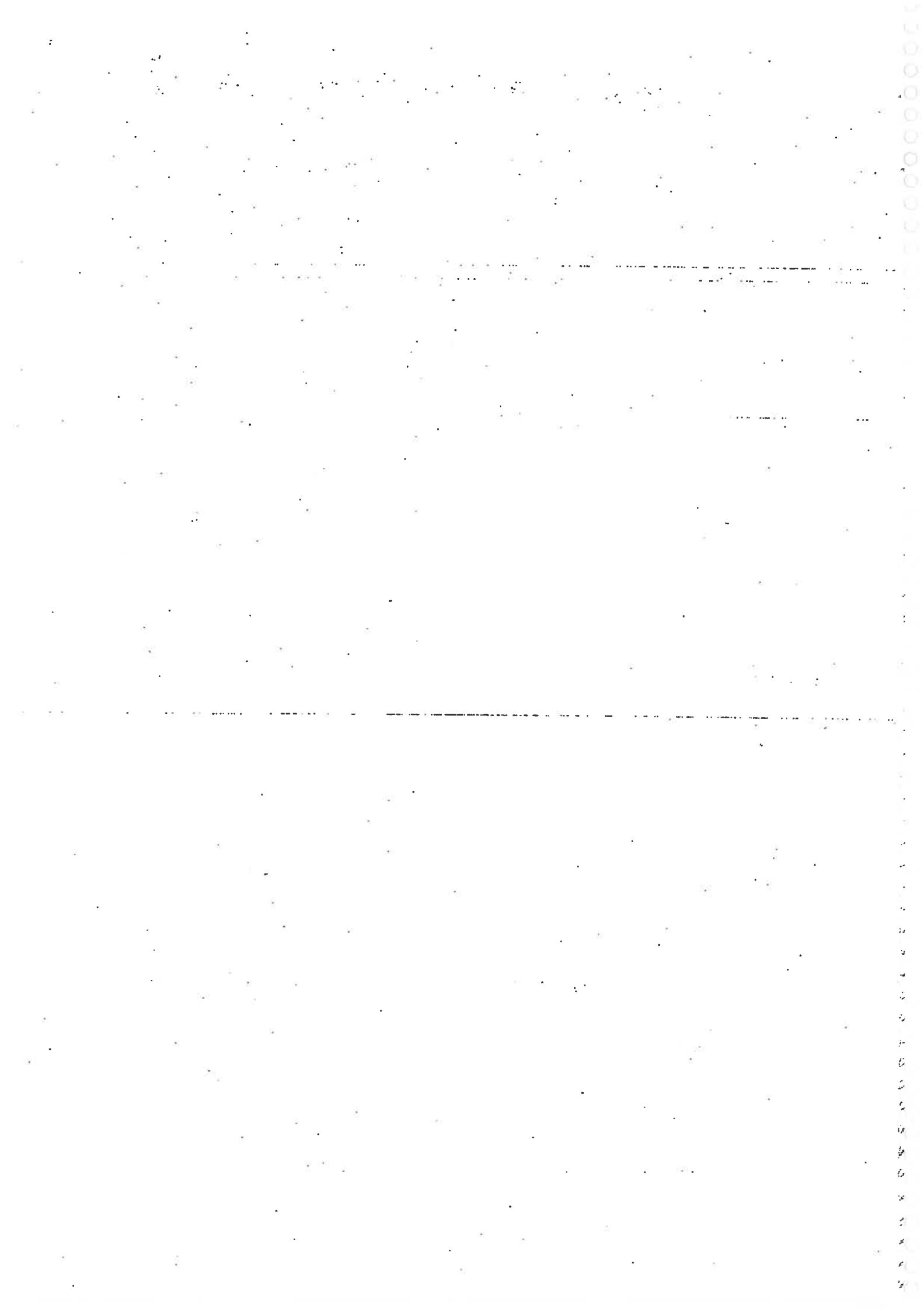
- Actividades organizadas de forma exclusiva por el centro C1.
- Centros y actividades organizadas sin éxito (0 participantes).
- Personas a las que gustan, al menos, una actividad de cada tipo.
- Personas a las que gustan solamente actividades de tipo cultural.
- Personas a las que gustan todas las actividades de ocio ofrecidas por el centro C1.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- Personas que participan en alguna actividad que les gusta.
- Centros que organizan actividades culturales y de ocio pero no de riesgo.
- Actividades organizadas por todos los centros.
- Centros que organizan, al menos, las mismas actividades deportivas que el centro C1.
- Personas que si participan en alguna actividad, sólo lo hacen en aquel centro que la organiza más barata.

4) Responder en SQL a las siguientes consultas:

- Actividades organizadas por, al menos, 2 centros.
- Precio medio de las actividades que organiza el centro C1.
- ¿Cuánto se ha gastado en actividades de ocio la persona P1?
- Personas que sólo han participado en actividades organizadas por un único centro.
- Personas tales que todas las actividades que les gustan son organizadas por algún centro.



1) Responder en álgebra relacional a las siguientes preguntas:

a - Centros que sólo organizan actividades culturales o deportivas.

$$P(C)(ORGANIZA) - P(C)S(T \neq 'cultural') \wedge (T \neq 'deportivas') (ORGANIZA * TIPO)$$

b - Personas que no participan en actividades que no les gustan.

$$A = (P(P)(PARTICIPA) - P(P)(GUSTA)) \rightarrow \text{Participa en lo que no le gusta.}$$
$$P(P)(PARTICIPA) - A$$

c - Personas a las que les gustan todas las actividades deportivas organizadas por el centro c1.

$$A = P(A)S(C = 'C1') (ORGANIZA)$$
$$B = P(P,A)(GUSTA)$$

A → Act. organizadas por C1  
A está contenido en B.

d - Actividad más cara organizada por el centro C1.

$$O1 = O2 = ORGANIZA$$

$$P(A)(ORGANIZA) - P(O1,A)S(O1.PR < O2.PR) (O1 \times O2)$$

e - Personas tales que todas las actividades que les gustan son organizadas por, al menos, un mismo centro.

$$\left\{ \begin{array}{l} tca / (\exists p)(t[P] = p[P]) \wedge (\forall g)(g[A] = t[CA]) \wedge \\ \quad \wedge (\exists p1)(g[CP] = p1[CP]) \wedge (p[C] \neq p1[C]) \end{array} \right\}$$

$\text{dom}(g) = GUSTA ; \text{dom}(p) = \text{dom}(p1) = PARTICIPA$

$$A = B = (GUSTA * PARTICIPA)$$

$$P(P)(GUSTA) - P(A,P)S(A.C \neq B.C) (A \times B)$$

2). Responder en cálculo relacional de tuplas a las siguientes consultas:

a. Actividades organizadas de forma exclusiva por el centro C1.

$$\text{dom}(\sigma) = \text{dom}(\sigma_1) = \text{ORGANIZA}$$

$$\{ t_{12} / (\exists \sigma) (t[A] = \sigma[A]) \wedge (\sigma[C] = 'C1') \}$$

$$= (\exists \sigma_1) (\sigma_1[A] = \sigma_1[EA]) \wedge (\sigma_1[C] \neq 'C1')$$

b. Centros y actividades organizadas sin éxito (0 participantes).

$$\text{dom}(p) = \text{PARTICIPA}$$

$$\{ t_{12} / (\exists p) (t[EE] = p[EE]) \wedge (t[EA] = p[EA]) \}$$

$$= (\exists p_1) (p[P] = p_1[P])$$

c. Personas a las que gustan, al menos, una actividad de cada tipo.

$$\text{dom}(t_1) = \text{TIPO} ; \text{dom}(g) = \text{GUSTA} ; \text{dom}(t_2) = \text{TIPO}$$

$$\{ t_{12} / (\exists g) (t[P] = g[P]) \}$$

$$= (\forall t_1) (\exists t_2) (g[A] \neq t_2[A]) \wedge (t_2[T] = t_1[T])$$

d. Personas a las que gustan solamente actividades de tipo cultural.

$$\text{dom}(g) = \text{GUSTA} ; \text{dom}(t) = \text{TIPO}$$

$$\{ t_{12} / (\exists g) (t[P] = g[P]) \}$$

$$= (\exists t) (g[A] = t[EA]) \wedge (t[T] \neq 'cultural')$$

e. Personas a las que gustan todas las actividades de ocio ofrecidas por el centro C1.

$$\text{dom}(g) = \text{GUSTA} ; \text{dom}(\sigma) = \text{ORGANIZA} ; \text{dom}(t) = \text{TIPO}$$

$$\{ t_{12} / (\exists g) (t[P] = g[P]) \}$$

$$= (\exists t) (\forall \sigma) (t[A] = \sigma[EA]) \wedge (t[T] = 'ocio') \wedge (t[A] = \sigma[EA]) \wedge (\sigma[C] = 'C1')$$

3). Responder en cálculo relacional de dominios a las siguientes consultas:

a. Personas que participan en alguna actividad que les gusta.

$$\{ \langle p \rangle / (\exists a, c) (\langle p, a, c \rangle \in \text{PARTICIPA}) \wedge$$

$$(\exists p_1, a_1) (\langle p_1, a_1 \rangle \in \text{GUSTA}) \wedge (a = a_1) \wedge (p = p_1) \}$$

b. Centros que organizan actividades culturales y de ocio pero no de riesgo

$$\{ \langle c \rangle / (\exists a, pr) (\langle c, a, pr \rangle \in \text{ORGANIZA}) \wedge$$

$$(\exists a_1, t_1) (\langle a_1, t_1 \rangle \in \text{TIPO}) \wedge (a = a_1) \wedge (t_1 = \text{'riesgo'}) \wedge$$

$$(\exists a_2, t_2) (\langle a_2, t_2 \rangle \in \text{TIPO}) \wedge (a_2 = a_1) \wedge (t_2 = \text{'ocio'}) \wedge (t_2 = \text{'cultural'}) \}$$

c. Actividades organizadas por todos los centros.

(\*)  $\{ \langle a \rangle / (\exists pr) (\forall c) (\langle a, c, pr \rangle \in \text{ORGANIZA}) \}$

d. Centros que organizan, al menos, las mismas actividades deportivas que el centro C1.

(\*)  $\{ \langle c \rangle / (\exists a, pr) (\langle c, a, pr \rangle \in \text{ORGANIZA}) \wedge$

$$(\forall a_1) (\exists c_1, pr_1) (\langle c_1, a_1, pr_1 \rangle \in \text{ORGANIZA}) \wedge (c_1 = \text{'C1'}) \wedge (a = a_1) \wedge$$

$$(\exists a_2, t) (\langle a_2, t \rangle \in \text{TIPO}) \wedge (t = \text{'deportivas'}) \wedge (a_2 = a_1) \}$$

e. Personas que sí participan en alguna actividad, sólo lo hacen en aquel centro que la organiza más barata.

(\*)  $\{ \langle p \rangle / (\exists a, c) (\langle p, a, c \rangle \in \text{PARTICIPA}) \wedge$

$$(\exists c_1, a_1, pr_1) (\langle c_1, a_1, pr_1 \rangle \in \text{ORGANIZA}) \wedge$$

$$(\exists c_2, a_2, pr_2) (\langle c_2, a_2, pr_2 \rangle \in \text{ORGANIZA}) \wedge (c_2 \neq c_1) \wedge (pr_2 < pr_1) \wedge \\ (a = a_1) \}$$

4) Responder en SQL a las siguientes consultas:

a. Actividades organizadas por, a lo sumo, dos centros.

```
SELECT A  
FROM ORGANIZA  
GROUP BY A  
HAVING (COUNT(*) ≤ 2);
```

b. Precio medio de las actividades que organiza el centro C1.

```
SELECT AVG(PR)  
FROM ORGANIZA  
WHERE (C = 'C1');
```

c. ¿Cuánto se ha gastado en actividades de ocio la persona P1?

(\*)  

```
SELECT SUM(PR)  
FROM ORGANIZA NATURAL JOIN PARTICIPA X  
WHERE (P = 'P1')  
AND A IN (SELECT A FROM X NATURAL JOIN TIPO  
WHERE (T = 'Ocio'));
```

d. Personas que sólo han participado en actividades organizadas por un único centro.

```
SELECT P  
FROM PARTICIPA P1  
WHERE NOT EXISTS (SELECT *  
                   FROM PARTICIPA P2  
                   WHERE (P.P1 = P.P2)  
                   AND (C.P1 ≠ C.P2));
```

e. Personas tales que todas las actividades que les gustan son organizadas por algún centro.

SELECT P

FROM GUSTA

WHERE NOT EXISTS (SELECT A

FROM ORGANIZA NATURAL JOIN GUSTA

WHERE

Centro con todas las actividades que le gustan a 1 persona

SELECT P

FROM ORGANIZA NATURAL JOIN GUSTA X

WHERE A NOT IN (SELECT \*

FROM GUSTA NATURAL JOIN ORGANIZA Y

WHERE (X.C ≠ Y.C);



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**4 de Septiembre de 1999**

En un cierto hospital se dispone de una base de datos con información sobre las enfermedades (clasificadas por tipos y que síntomas manifiestan), así como sobre los síntomas que presentan los pacientes y las enfermedades que les han sido diagnosticadas.

Los atributos que se utilizan se abrevian de la siguiente manera:

Atributo	Significado
P	Paciente
E	Enfermedad
S	Síntoma
T	Tipo

El esquema de la base de datos es:

**TIPO(E, T)**

**SIGNIFICADO:** La enfermedad E es del tipo T. Ej: El infarto es de tipo cardio-vascular.

Clave: (E, T)

**MANIFIESTA(E, S)**

**SIGNIFICADO:** La enfermedad E manifiesta el síntoma S.

Clave: (E, S)

**PRESENTA(P, S)**

**SIGNIFICADO:** El paciente P presenta el síntoma S.

Clave: (P, S)

**DIAGNÓSTICO(P, E)**

**SIGNIFICADO:** Al paciente P se le ha diagnosticado la enfermedad E.

Clave: (P, E)

Nota: Por desgracia, a algunos pacientes puede que aún no se le haya diagnosticado nada.

1) Responder en álgebra relacional a las siguientes preguntas:

- a) Pacientes a los que se haya diagnosticado alguna enfermedad de tipo oncológico.
- b) Pacientes que sólo presentan los síntomas S1 y S2 (simultáneamente).
- c) Pacientes que sólo presentan síntomas pertenecientes a enfermedades de tipo gastrointestinal.
- d) Síntomas que se manifiestan en todas las enfermedades de tipo cardio-vascular.
- e) Pacientes que presentan todos los síntomas de alguna de las enfermedades que les han sido diagnosticadas.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

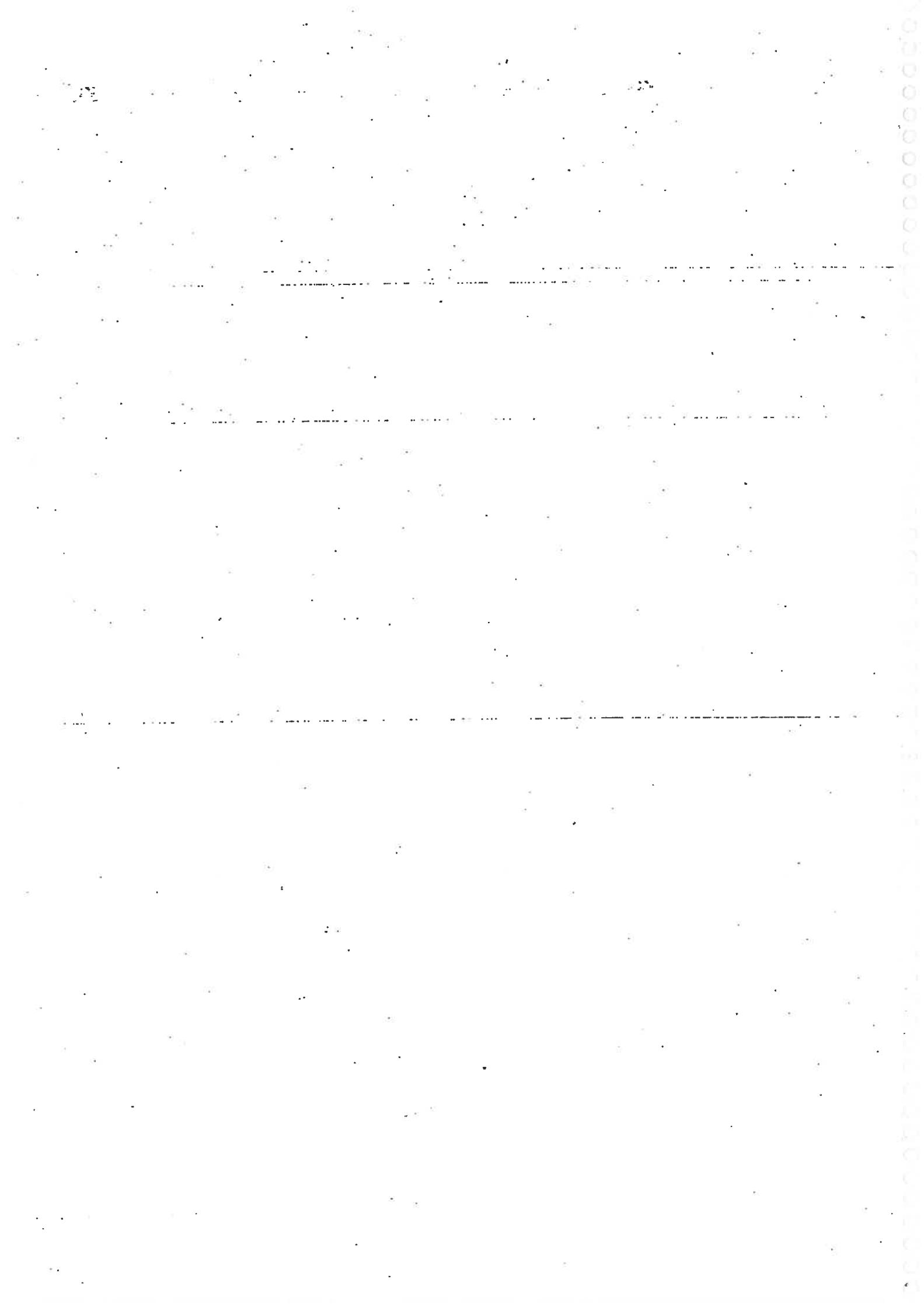
- a) Pacientes que presentan dos o más síntomas.
- b) ¿Qué enfermedades no se le pueden diagnosticar al paciente X en función de los síntomas que presenta?
- c) Pacientes que presentan, a lo sumo, los síntomas de un catarro.
- d) Pacientes que presentan todo tipo de síntomas.
- e) Síntomas que sólo presenta el paciente X.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Pacientes que presentan los síntomas S1 y S2, pero no el S3.
- b) Enfermedades que manifiestan, al menos, los mismos síntomas de la gripe.
- c) Pacientes que no presentan alguno de los síntomas de alguna de las enfermedades que les han sido diagnosticadas.
- d) Pacientes que sólo presentan síntomas pertenecientes a enfermedades de tipo vírico.
- e) Síntomas que manifiesta exclusivamente la meningitis.

4) Responder en SQL a las siguientes consultas:

- a) Número medio de síntomas que presentan los pacientes.
- b) Tipo de las enfermedades más frecuentemente diagnosticadas.
- c) Pacientes que no presentan alguno de los síntomas de alguna de las enfermedades que les han sido diagnosticadas.
- d) Pacientes tales que existe alguna enfermedad que manifiesta todos los síntomas que ellos presentan.
- e) Pacientes que presentan, al menos, el 60% de los síntomas de alguna de las enfermedades que le han sido diagnosticadas.



1) Responder en álgebra relational a las siguientes preguntas:

a - Pacientes a los que se haya diagnosticado alguna enfermedad de tipo onológico.

$$P(P) \cap (T = \text{'onológico'}) \quad (\text{TIPO} * \text{DIAGNOSIS})$$

b - Pacientes que sólo presentan los síntomas  $s_1$  y  $s_2$  (simultáneamente).

$$P(P) \cap (s = \text{'s1'}) \wedge (s = \text{'s2'}) \quad (\text{PRESENTA})$$

$$A = P(P) \cap (s \neq s_1) \wedge (s = s_2) \quad (\text{PRESENTA})$$

$$P(P)(\text{PRESENTA}) - P(P)(A)$$

c - Pacientes que sólo presentan síntomas pertenecientes a enfermedades de tipo gastrointestinal.

$$A = P(P) \cap (T \neq \text{'gastrointestinal'}) \quad (\text{PRESENTA} * \text{MANIFIESTA} * \text{TIPO})$$

$$P(P)(\text{PRESENTA}) - P(P)(A)$$

d - Síntomas que se manifiestan en todas las enfermedades de tipo cardio-vascular.

$$A = P(E) \cap (T = \text{'cardiovascular'}) \quad (\text{TIPO}) \quad \left. \begin{array}{l} \\ \end{array} \right\} B/A \rightarrow A \text{ está contenido en } B.$$

$$B = P(E, S) \quad (\text{MANIFIESTA})$$

e - Pacientes que presentan todos los síntomas de alguna de las enfermedades que les ha sido diagnosticada.

2) Responder en cálculo relacional de tuplas a las siguientes consultas.

a. Pacientes que presentan dos o más síntomas.

$\text{dom}(p) = \text{dom}(p_1) = \text{PRESENTA}$

$\{ t_{12} / (\exists p)(t[P] = p[P]) \wedge$

$(\exists p_1)(p_1[P] = p_1[P]) \wedge (p[S] \neq p_1[S]) \}$

b. ¿Qué enfermedades no se le pueden diagnosticar al paciente 'x' en función de los síntomas que presenta?

$\text{dom}(p) = \text{PRESENTA}, \text{dom}(m) = \text{MANIFIESTA}$

$\{ t_{12} / (\exists m)(t[E] = m[E]) \wedge (\exists p)(p[P] = 'x') \wedge (p[S] = m[S]) \}$

A =  $P(S) \wedge (P = 'x') \wedge (\text{PRESENTA}) \quad C = B/A \rightarrow (\text{Enfermedades que presenta el paciente } 'x')$

B =  $P(E, S) \wedge (\text{MANIFIESTA})$

$P(E)(\text{MANIFIESTA}) - P(E)(C) \quad \checkmark$

c. Pacientes que presentan, a lo sumo, los síntomas de un catarro.

$\text{dom}(p) = \text{dom}(p_1) = \text{PRESENTA}$

$\{ t_{12} / (\exists p)(t[P] = p[P]) \wedge$

$\neg (\exists p_1)(p_1[P] = p_1[P]) \wedge (p[S] \neq 'catarro') \}$

Res. ir

d. Pacientes que presentan todos tipos de síntomas.

$\text{dom}(p) = \text{PRESENTA}, \text{dom}(m) = \text{MANIFIESTA}$

$\{ t_{12} / (\exists p)(t[P] = p[P]) \wedge$

$(\forall m)(p[S] = m[S]) \}$

Res. o

e. Síntomas que sólo presenta el paciente 'x'.

$$\{ \begin{array}{l} \text{dom}(p) = \text{dom}(p_1) = \text{PRESENTA} \\ \{ t(p) / (\exists p)(t[S] = p[S]) \wedge (p[P] = 'x') \wedge \\ \neg (\exists p_1)(p[P] \neq p_1[P]) \wedge (p[S] = p_1[S]) \end{array} \}$$

---

3). Responder en cálculo relacional de dominios a las siguientes consultas:

a. Pacientes que presentan los síntomas  $s_1$  y  $s_2$ , pero no el  $s_3$ .

$$\{ \begin{array}{l} \langle p \rangle / (\exists s)(\langle p, 's_1' \rangle \in \text{PRESENTA}) \wedge \\ (\exists p_1, s_1)(\langle p_1, 's_2' \rangle \in \text{PRESENTA}) \wedge (p = p_1) \wedge \\ (\exists p_2, s_2)(\langle p_2, 's_3' \rangle \in \text{PRESENTA}) \wedge (p \neq p_2) \end{array} \}$$

b. Enfermedades que manifiestan, al menos, los mismos síntomas de la gripe.

$$\{ \begin{array}{l} \langle e \rangle / (\exists s)(\langle e, s \rangle \in \text{MANIFIESTA}) \wedge \\ \neg (\exists e_1, s_1)(\langle e_1, s_1 \rangle \in \text{MANIFIESTA}) \wedge (e_1 = 'gripe') \wedge (e_1 \neq e) \wedge (s \neq s_1) \end{array} \}$$

c. Pacientes que no presentan alguno de los síntomas de alguna de las enfermedades que les han sido diagnosticadas.

$$\{ \begin{array}{l} \langle p \rangle / (\exists s)(\langle p, s \rangle \in \text{PRESENTA}) \wedge \\ (\exists e_1, s_1)(\langle e_1, s_1 \rangle \in \text{MANIFIESTA}) \wedge (s = s_1) \\ \neg (\exists p, e)(\langle p, e \rangle \in \text{DIAGNOSIS}) \wedge (e = e_1) \end{array} \}$$

d. Pacientes que sólo presentan síntomas pertenecientes a enfermedades de tipo vírico.

$$\{ \begin{array}{l} \langle p \rangle / (\exists s)(\langle p, s \rangle \in \text{PRESENTA}) \wedge \\ (\exists e_1, s_1)(\langle e_1, s_1 \rangle \in \text{MANIFIESTA}) \wedge (s = s_1) \\ \neg (\exists e, t)(\langle e, t \rangle \in \text{TIPO}) \wedge (t \neq 'vírico') \wedge (e = e_1) \end{array} \}$$

(No existe una enfermedad con esos síntomas distinta a una de tipo vírico)

e. Síntomas que manifiesta exclusivamente la meningitis.

$\{ \langle s \rangle / (\exists e) (\langle e, s \rangle \in \text{MANIFIESTA}) \}$

$\neg (\exists e_1, s_1) (\langle e_1, s_1 \rangle \in \text{MANIFIESTA}) \wedge (e_1 \neq \text{'meningitis'}) \wedge (s = s_1)$

Re<sup>5</sup>

4) Responder en SQL a las siguientes consultas:

a. Número medio de síntomas que presentan los pacientes.

SELECT AVG (COUNT (\*))

FROM PRESENTA

GROUP BY P;

b. Tipo de las enfermedades más frecuentemente diagnosticadas.

SELECT T

FROM DIAGNOSIS NATURAL JOIN TIPO

GROUP BY T

HAVING COUNT (E) >= ALL (SELECT COUNT (E)

FROM DIAGNOSIS

GROUP BY E);

Re<sup>5</sup>

c. Pacientes que no presentan alguno de los síntomas de alguna de las enfermedades que les han sido diagnosticadas.

d- Pacientes tales que existe alguna enfermedad que manifiesta todos los síntomas que ellos presentan.

SELECT P  
FROM DIAGNOSIS NATURAL JOIN PRESENTA P  
WHERE NOT EXISTS (SELECT S  
FROM MANIFIESTA M  
WHERE (P.E = M.E) AND (M.S ≠ P.S));

Rev.

e- Pacientes que presentan, al menos, el 60% de los síntomas de alguna de las enfermedades que le han sido diagnosticadas.

SELECT P  
FROM DIAGNOSIS NATURAL JOIN MANIFIESTA  
GROUP BY E  
HAVING (COUNT(S) \* 0.6) >= ALL (SELECT COUNT(S)  
FROM MANIFIESTA N.J. DIAGNOSIS  
GROUP BY E);

Res.



**Estructura de Datos y de la Información II**  
**Escuela Técnica Superior de Ingeniería**  
**Informática**  
**Universidad de La Laguna**  
**17 de Junio de 2003**

El Servicio de Orientación al Alumno (SOA) de la ULL ha decidido crear una base de datos que mantenga información sobre la oferta de plazas de alojamiento disponibles en La Laguna para los alumnos universitarios. En el esquema de base de datos propuesto los atributos que se utilizan se abrevian utilizando el siguiente convenio:

Atributo	Significado
DNI	DNI DEL ALUMNO
T	TITULACIÓN
C	CURSO (1, 2, ...)
A	AÑO ACADÉMICO (2000, 2001, ...)
CP	CÓDIGO PISO
N	NÚMERO DE PLAZAS
P	PRECIO POR PLAZA

Las tablas utilizadas son:

**ESTUDIANTES (DNI, T, C, A)**

**SIGNIFICADO:** El alumno con dni DNI realiza el curso C de la titulación T en el año académico A.  
**CLAVE PRIMARIA:** (DNI, T, C, A)

**PISO (CP, N, P, A)**

**SIGNIFICADO:** El piso con código CP oferta N plazas en el año académico A y el precio de cada plaza es P euros.  
**CLAVE PRIMARIA:** (CP, A)

**ALQUILER (DNI, CP, A)**

**SIGNIFICADO:** El alumno con dni DNI ha alquilado el piso con código CP en el año académico A.  
**CLAVE PRIMARIA:** (DNI, A)

1) Responder en álgebra relacional a las siguientes preguntas:

- Alumnos que no se han quedado nunca en un piso de alquiler.
- Pisos que actualmente tienen un único alumno en alquiler.
- Alumnos que se hayan quedado en al menos los mismos pisos que el alumno 1111.
- Parejas de alumnos que nunca han sido compañeros de pisos.
- Alumnos que siempre han alquilado pisos donde había algún compañero de su misma titulación.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Alumnos de informática que este curso no alquilaron un piso.
- Piso que ofreció mayor número de plazas en el curso 2001.
- Alumnos que cada curso académico alquilan un piso distinto al del año anterior.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- Pisos en los que, en algún curso académico, sólo se quedaban alumnos que estudiaban una misma titulación.
- Parejas de alumnos que siempre han sido compañeros de piso.

4) Responder en SQL a las siguientes consultas:

- Pisos en los que hay simultáneamente alumnos de 'Matemáticas' e 'Informática'.
- Número de plazas de alojamiento ofertadas este curso académico.
- Número medio de alumnos que comparten piso.
- Pisos completos este curso académico.
- Pisos tales que al menos el 80% de los estudiantes que se quedan en él son de una misma titulación.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {A → BC, AC → E, E → A, BD → E, C → D}.

- Calcular todas las claves.
- Calcular E+ aplicando el algoritmo de Ullman.
- Probar formalmente (a través de los axiomas de Armstrong) que CB → E pertenece a DF+.
- Descomponer R en FNBC. Se preservan las dependencias funcionales?
- Descomponer R en FN3.



**Estructura de Datos y de la Información II**  
**Escuela Técnica Superior de Ingeniería**  
**Informática**  
**Universidad de La Laguna**  
**17 de Junio de 2003**

El Servicio de Orientación al Alumno (SOA) de la ULL ha decidido crear una base de datos que mantenga información sobre la oferta de plazas de alojamiento disponibles en La Laguna para los alumnos universitarios. En el esquema de base de datos propuesto los atributos que se utilizan se abrevian utilizando el siguiente convenio:

Atributo	Significado
DNI	DNI DEL ALUMNO
T	TIULACIÓN
C	CURSO (1, 2, ...)
A	ÁÑO ACADÉMICO (2000, 2001, ...)
CP	CÓDIGO PISO
N.	NUMERO DE PLAZAS
P	PRECIO POR PLAZA

Las tablas utilizadas son:

**ESTUDIANTES (DNI, T, C, A)**

**SIGNIFICADO:** El alumno con dni DNI realiza el curso C de la titulación T en el año académico A.

**CLAVE PRIMARIA:** (DNI, T, C, A)

**PISO (CP, N, P, A)**

**SIGNIFICADO:** El piso con código CP oferta N plazas en el año académico A y el precio de cada plaza es P euros.

**CLAVE PRIMARIA:** (CP, A)

**ALQUILER (DNI, CP, A)**

**SIGNIFICADO:** El alumno con dni DNI ha alquilado el piso con código CP en el año académico A.

**CLAVE PRIMARIA:** (DNI, A)

1) Responder en álgebra relacional a las siguientes preguntas:

- Alumnos que no se han quedado nunca en un piso de alquiler.
- Pisos que actualmente tienen un único alumno en alquiler.
- Alumnos que se hayan quedado en al menos los mismos pisos que el alumno 1111.
- Parejas de alumnos que nunca han sido compañeros de pisos.
- Alumnos que siempre han alquilado pisos donde había algún compañero de su misma titulación.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Alumnos de informática que este curso no alquilaron un piso.
- Piso que ofertó mayor número de plazas en el curso 2001.
- Alumnos que cada curso académico alquilan un piso distinto al del año anterior.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- Pisos en los que, en algún curso académico, sólo se quedaban alumnos que estudiaban una misma titulación.
- Parejas de alumnos que siempre han sido compañeros de piso.

4) Responder en SQL a las siguientes consultas:

- Pisos en los que hay simultáneamente alumnos de 'Matemáticas' e 'Informática'.
- Número de plazas de alojamiento ofertadas este curso académico.
- Número medio de alumnos que comparten piso.
- Pisos completos este curso académico.
- Pisos tales que al menos el 80% de los estudiantes que se quedan en él son de una misma titulación.

5) Sea R(A, B, C, D, E) una relación con dependencias funcionales DF = {A → BC, AC → E, E → A, BD → E, C → D}.

- Calcular todas las claves.
- Calcular E+ aplicando el algoritmo de Ullman.
- Probar formalmente (a través de los axiomas de Armstrong) que CB → E pertenece a DF+.
- Descomponer R en FNB. Se preservan las dependencias funcionales?
- Descomponer R en FNS.





Estructura de Datos y de la Información II  
Centro Superior de Informática  
Universidad de La Laguna  
5 de Julio de 1999

La Universidad de La Laguna ha decidido informatizar la gestión de préstamos bibliotecarios.

Los atributos que se utilizan los vamos a abreviar de la siguiente manera:

Atributo	Significado
CB	Código de biblioteca
FD	Fecha de devolución
FP	Fecha de préstamo
DNI	D.N.I. del alumno
I	ISBN
R	Número de registro

El esquema de la base de datos es el siguiente:

**REGISTRO(CB, I, R)**

**SIGNIFICADO:** La biblioteca con código CB, posee el libro con ISBN I y le ha asignado el número de registro R.

**CLAVE PRIMARIA:** (CB, R).

**SOCIO(DNI, CB)**

**SIGNIFICADO:** El alumno con D.N.I. DNI es socio de la biblioteca con código CB.

**CLAVE PRIMARIA:** (DNI, CB)

**PRESTAMO (DNI, CB, R, FP, FD)**

**SIGNIFICADO:** El alumno con D.N.I. DNI, ha sacado en préstamo de la biblioteca CB, el libro con número de registro R, en la fecha FP, y lo ha devuelto en la fecha FD. Si un libro aún no se ha devuelto tiene en fecha de devolución un NULL.

**CLAVE PRIMARIA:** (DNI, CB, R, FP)

**CLAVES AJENAS:** (DNI, CB), (CB, R)

- 1) Responder en álgebra relacional a las siguientes preguntas:
  - a) Bibliotecas que no tienen el libro con ISBN 1000.
  - b) Libros que están en todas las bibliotecas.
  - c) Libro registrado con mayor ISBN.
  - d) Alumnos que siempre han sacado en, al menos una biblioteca, libros en préstamo con diferentes ISBN.
  - e) Alumnos que, a lo largo del tiempo, han sacado al menos un ejemplar, de todos los libros de alguna de las bibliotecas de las que son socios.
- 2) Responder en cálculo relacional de t-uplas a las siguientes consultas:
  - a) Alumnos que sólo son socios de una única biblioteca.
  - b) Libro registrado con mayor ISBN.
  - c) Alumnos socios de, al menos, las mismas bibliotecas que el alumno con DNI 1111.
  - d) Alumnos que sólo han sacado libros en préstamo de una única biblioteca.
  - e) Alumnos que no han sacado ningún libro en préstamo de alguna de las bibliotecas de las que son socios.
- 3) Responder en cálculo relacional de dominios a las siguientes consultas:
  - a) Alumnos que sólo son socios (simultáneamente) de las bibliotecas con código 1 y 2.
  - b) Alumnos que son socios, a lo sumo, de las mismas bibliotecas que el alumno con DNI 1111.
  - c) Libros que están en todas las bibliotecas.
  - d) Bibliotecas que tienen todos los ejemplares de alguno de sus libros en préstamo.
  - e) Alumnos que han sacado, al menos, un mismo libro en préstamo, de todas las bibliotecas de las que son socios.
- 4) Responder en SQL a las siguientes consultas:
  - a) Número medio de ejemplares por título con que cuenta la biblioteca con código 1.
  - b) Biblioteca con mayor número de libros.
  - c) Alumnos que sólo son socios de una única biblioteca.
  - d) Alumnos socios de todas las bibliotecas.
  - e) Alumnos tales que, al menos, el 80% de los libros que han sacado en préstamo pertenecen a una misma biblioteca.



## Estructura de Datos y de la Información II

Centro Superior de Informática

Universidad de La Laguna

21 de Junio de 1999

En un cierto hospital se dispone de una base de datos con información sobre las enfermedades (clasificadas por tipos y que síntomas manifiestan), así como sobre los síntomas que presentan los pacientes y las enfermedades que les han sido diagnosticadas.

Los atributos que se utilizan se abrevian de la siguiente manera:

Atributo	Significado
P	Paciente
E	Enfermedad
S	Síntoma
T	Tipo

El esquema de la base de datos es:

### TIPO(E, T)

SIGNIFICADO: La enfermedad E es del tipo T. Ej: El infarto es de tipo cardio-vascular.

Clave: (E, T)

### MANIFIESTA(E, S)

SIGNIFICADO: La enfermedad E manifiesta el síntoma S.

Clave: (E, S)

### PRESENTA(P, S)

SIGNIFICADO: El paciente P presenta el síntoma S.

Clave: (P, S)

### DIAGNOSIS(P, E)

SIGNIFICADO: Al paciente P se le ha diagnosticado la enfermedad E.

Clave: (P, E)

Nota: Por desgracia, a algunos pacientes puede que aún no se le haya diagnosticado nada.

✓ Responder en álgebra relacional a las siguientes preguntas:

- a) Pacientes que presentan dos o más síntomas.
- b) Pacientes que presentan, a lo sumo, los síntomas de un catarro.
- c) Pacientes que presentan todo tipo de síntomas.
- d) Pacientes a los que sólo se le han diagnosticado enfermedades para las cuales presentan todos los síntomas.
- e) Pacientes tales que existe alguna enfermedad que manifiesta todos los síntomas que ellos presentan.

✓ Responder en cálculo relacional de tuplas a las siguientes consultas:

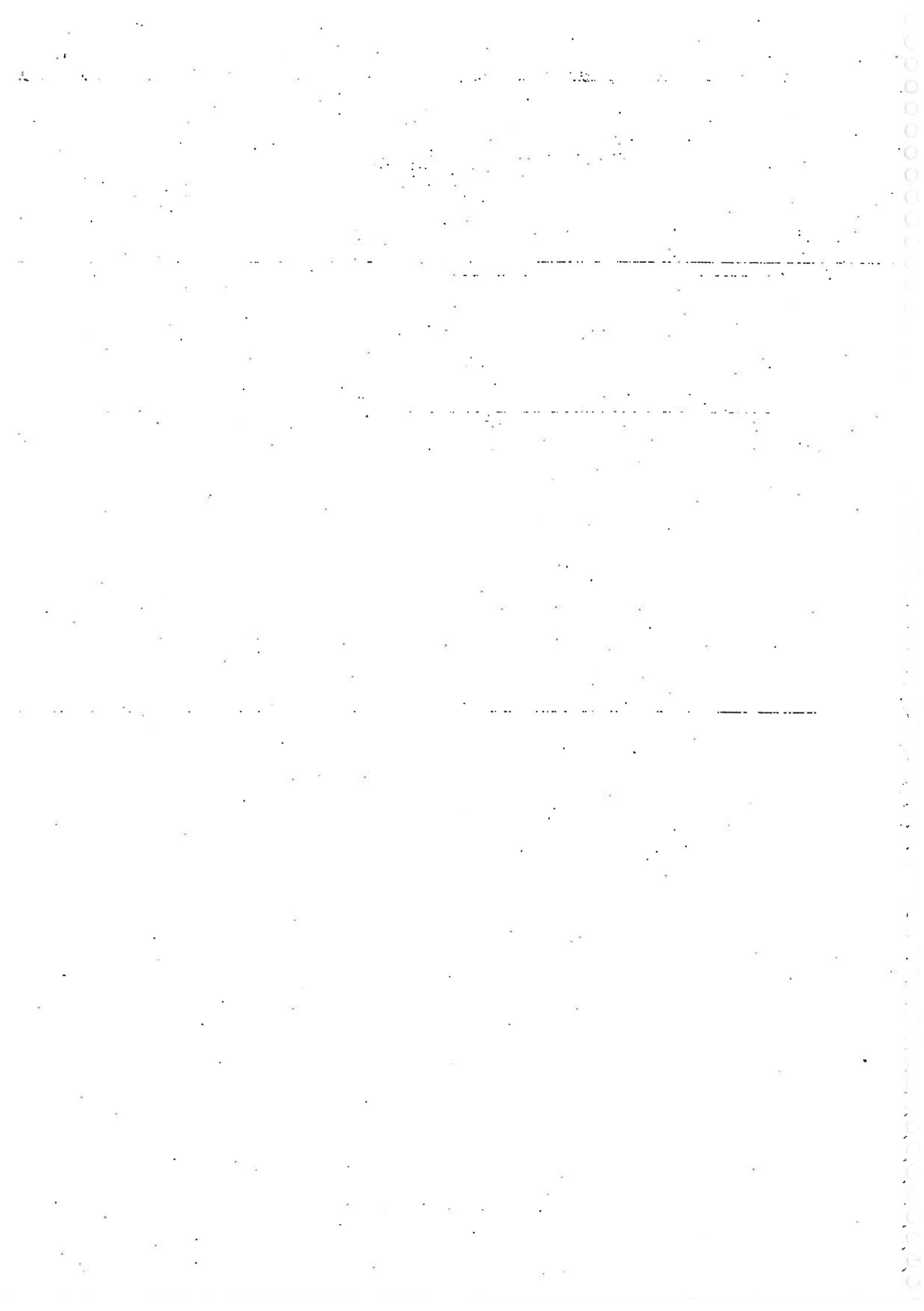
- a) Pacientes a los que se haya diagnosticado alguna enfermedad de tipo oncológico.
- b) Pacientes a los que se haya diagnosticado más de una enfermedad.
- c) Pacientes que sólo presentan los síntomas S1 y S2 (simultáneamente).
- d) Pacientes que no presentan alguno de los síntomas de alguna de las enfermedades que les han sido diagnosticadas.
- e) Enfermedades que están clasificadas como pertenecientes a todos los tipos.

✓ Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Pacientes que no presentan el síntoma de la fiebre.
- b) Enfermedades que manifiestan alguno de los síntomas de la gripe.
- c) Síntomas que se manifiestan en todas las enfermedades de tipo cardio-vascular.
- d) Pacientes con algún síntoma que no manifiestan las enfermedades que le han sido diagnosticadas.
- e) Pacientes que presentan todos los síntomas de alguna de las enfermedades que le han sido diagnosticadas.

4) Responder en SQL a las siguientes consultas:

- ✓ Síntomas que sólo manifiesta la meningitis.
- ✓ Número medio de síntomas que manifiestan las enfermedades de tipo oncológico.
- ✓ Paciente con mayor número de enfermedades diagnosticadas.
- ✓ Pacientes que sólo presentan síntomas que manifiesta la meningitis.
- ✓ Enfermedades que manifiestan, al menos, los mismos síntomas que la gripe.



**EXAMEN:** 21 JUNIO 1999.

1. Responder en álgebra relacional a las siguientes preguntas:

a) Pacientes que presentan dos o más síntomas.

$$\text{PRE1} = \text{PRE2} = \text{PRESENTA}$$

$$P(\text{PRE1}) \left( S((\text{PRE1.P} = \text{PRE2.P}) \wedge (\text{PRE1.S} = \text{PRE2.S})) \mid (\text{PRE1} \times \text{PRE2}) \right).$$

b) Pacientes que presentan, a lo sumo, los síntomas de un catarro.

(\*)  $P(S) \left( S(E = "catarro") \mid \text{MANIFIESTA} \right) \rightarrow A$  → Proyectamos los síntomas de la enfermedad del catarro.  
 $P(P) \left( \text{PRESENTA} * A \right)$

c) Pacientes que presentan todo tipo de síntomas.

MANIFIESTA (E, S)  
PRESENTA (P, S)

$P(S) \mid \text{MANIFIESTA} \rightarrow A$  → Proyectamos todos los síntomas de la tabla manifiesta.

$$P(P) \left( \text{PRESENTA} / A \right)$$

Relación formada por todos los valores de un atributo de la relación binaria que concuerden.  
Se aplica al primer conjunto (PRESENTA), elimina todas las filas que no coincidan con la segunda columna.

d) Pacientes a los que sólo se le han diagnosticado enfermedades para las cuales presentan todos los síntomas.

e) Pacientes tales qe existe alguna enfermedad que manifiesta todos los  
síntomas que ellos presentan.



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**13 de Septiembre de 2000**

Un curso sobre cocina utiliza una base de datos que tiene el siguiente esquema:

Abreviaremos los atributos utilizados por:

Atributo	Significado
C	CANTIDAD
CD	CANTIDAD DISPONIBLE
I	INGREDIENTE
P	PERSONA
PL	PLATO

Las tablas son:

**RECETA(PL, I, C)**

**SIGNIFICADO:** El plato PL necesita, para su preparación, C gramos del ingrediente I.

**CLAVE PRIMARIA:** (PL, I)

**GUSTA(P, PL)**

**SIGNIFICADO:** A la persona P le gusta el plato PL.

**CLAVE PRIMARIA:** (P, PL)

**DISPONE(P, I, CD)**

**SIGNIFICADO:** La persona P dispone de CD gramos del ingrediente I.

**CLAVE PRIMARIA:** (P, I)

Nota: Una persona puede preparar un plato si dispone de todos los ingredientes y las cantidades necesarias, según indica la receta.

1) Responder en álgebra relacional a las siguientes preguntas:

- Ingredientes del plato PL1 de los que no dispone (en ninguna cantidad) la persona P1.
- Platos que gustan a lo sumo a dos personas.
- Platos que tienen, al menos, los mismos ingredientes que la tortilla.
- Personas que pueden preparar alguno de los platos que les gustan.
- Personas que sólo pueden elegir para preparar un único plato.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- Platos que no gustan a nadie.
- Personas que disponen simultáneamente del ingrediente I1 e I2 pero no del I3.
- Ingrediente que se utiliza en mayor proporción el plato PL1.
- Personas que disponen de alguno de los ingredientes de cada plato que les gusta.
- Personas que pueden preparar alguno de los platos que les gustan.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- Platos que no gustan a la persona PRI.
- Ingredientes y cantidades que necesita la persona PRI para preparar alguno de los platos que le gustan.
- Ingredientes que intervienen en un plato exclusivamente.
- Personas que disponen de algo (alguna cantidad) de todos los ingredientes que intervienen en el 'potaje'.
- Ingredientes que sólo intervienen en los platos que gustan a PRI.

4) Responder en SQL a las siguientes consultas:

- Ingredientes que intervienen en un plato exclusivamente.
- Número medio de ingredientes distintos que tienen los platos.
- ¿De cuántos ingredientes del potaje dispone la persona PRI en cantidades suficientes?
- Platos que gustan al menos al 70% de las personas.
- ¿Qué tendría que comprar (ingredientes y cantidades), después de mirar su despensa, la persona P1 para preparar una tortilla?



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**2 de Diciembre de 2000**

La ULL gestiona la información sobre los complementos salariales de los profesores de la misma con una base de datos que tiene el siguiente esquema:

Abreviaremos los atributos utilizados por:

Atributo	Significado
C	COMPLEMENTO
P	PROFESOR
T	TIPO

Las tablas son:

**SOLICITUD(P, C)**

**SIGNIFICADO:** El profesor P solicita el complemento retributivo C.

**CLAVE PRIMARIA:** (P, C).

**CONCESIÓN(P, C)**

**SIGNIFICADO:** Al profesor P se le concede el complemento C.

**CLAVE PRIMARIA:** (P, C)

**CLAVE AJENA:** (P, C) dependiente de SOLICITUD.

**TIPO(C, T)**

**SIGNIFICADO:** El complemento C es de tipo T (docente, investigador, gestión, divulgación...).

**CLAVE PRIMARIA:** (C, T)

1) Responder en álgebra relacional a las siguientes preguntas:

- a) Profesores a los se ha concedido algún complemento de tipo docente.
- b) Profesores que han solicitado más de un complemento.
- c) Profesores que han solicitado todos los complementos de investigación.
- d) Profesores a los qué sólo se le han concedido complementos que han sido denegados al profesor P1.
- e) Profesores a los que les han concedido exactamente los mismos complementos que al profesor P1.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Profesores que no han solicitado ninguno de los complementos de investigación.
- b) Profesores que han solicitado a lo sumo 2 complementos.
- c) Profesores que han solicitado, al menos, los mismos complementos que el profesor P1.
- d) Complementos que sólo son de un único tipo.
- e) Profesores a los que les han concedido todos los complementos solicitados.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Complementos no solicitados por nadie.
- b) Profesores que han solicitado complementos de tipo T1 y T2 pero no del tipo T3.
- c) Profesores a los que les han concedido un único complemento.
- d) Profesores que sólo han solicitado complementos de tipo investigador.
- e) Profesores que han solicitado, al menos, un complemento de cada tipo.

4) Responder en SQL a las siguientes consultas:

- a) Profesores que sólo han solicitado complementos de tipo T1 y T2.
- b) Número medio de complementos solicitados.
- c) Complemento más solicitado.
- d) Profesores que han solicitado todos los complementos de investigación.
- e) Profesores que han solicitado, al menos, el 60% de los complementos de tipo docente solicitados por el profesor P1.



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**15 de Junio de 2001**

La Federación española de fútbol dispone de una base de datos para almacenar la información generada en la Liga, siendo las siguientes tablas parte de su esquema:

**PERTENECE(J, E)**

**SIGNIFICADO:** El jugador J pertenece al equipo E.

**Clave:** (J)

**RESULTADO(E, P, G)**

**SIGNIFICADO:** El equipo E disputó el partido P y marcó G goles.

**Clave:** (E, P)

**JUEGA(J, P)**

**SIGNIFICADO:** El jugador J juega el partido P.

**Clave:** (J, P)

✓ Responder en álgebra relacional a las siguientes preguntas:

- a) Equipos a los que ha ganado el equipo E1.
- b) Equipos que han jugado dos o más partidos.
- c) Jugadores que han jugado, a lo sumo, los mismos partidos que el jugador J1.
- d) Equipo que ha marcado mayor número de goles en un partido.
- e) Jugadores que han jugado todos los partidos ganados por su equipo.

✓ Responder en cálculo relacional de t-uplas a las siguientes consultas:

- ✓ Jugadores del equipo E1 que han jugado simultáneamente los partidos P1 y P2.
- ✓ Jugadores que hayan jugado con su equipo en al menos una ocasión en la que hayan ganado.
- ✓ Equipos que sólo han disputado un único partido.
- ✓ Jugadores tales que siempre que han jugado, su equipo ha ganado.
- ✓ Equipos que han perdido siempre contra un mismo equipo.

✗ Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Partidos ganados por el equipo E1.
- b) Jugadores que aún no han jugado ningún partido.
- c) Equipos que nunca han perdido.
- d) Equipos tales que todos sus jugadores han jugado en más de un partido.
- e) Equipos que han ganado a todos los equipos a los que venció el equipo E1.

✗ Responder en SQL a las siguientes consultas:

- ✗ Jugadores que sólo han jugado en el partido P1.
- ✗ Número medio de partidos jugados por los jugadores del equipo E1.
- ✗ Equipos que siempre han ganado.
- ✗ Equipo que ha ganado más veces.
- ✗ Equipos que, a lo sumo, pierden el 10% de los partidos jugados.



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**26 de Junio de 2001**

A final de curso, el nuevo equipo decanal ha realizado una encuesta entre los alumnos del centro, con la intención de poder evaluar los resultados relativos a la docencia impartida por el profesorado. El esquema de la base de datos es:

**IMPARTE(P, AS)**

SIGNIFICADO: El profesor P imparte la asignatura AS. Clave: (P, AS)

**CALIFICACIÓN(A, AS, N)**

SIGNIFICADO: El alumno A está matriculado de la asignatura AS y ha sacado en Junio la nota N (en puntos de 0 a 10). Además, una asignatura se aprueba si tiene una nota igual o mayor que 5 puntos.

Clave: (A, AS)

**GUSTA(A, AS)**

SIGNIFICADO: Al alumno A le gusta la asignatura AS. Clave: (A, AS)

**VALORA(A, P)**

SIGNIFICADO: El alumno A valora positivamente al profesor P. Clave: (A, P)

1) Responder en álgebra relacional a las siguientes preguntas:

- a) Alumnos que han aprobado alguna asignatura que les gusta.
- b) Alumnos a los que gustan, al menos, dos asignaturas.
- c) Alumnos que han aprobado a lo sumo las asignaturas que aprobó el alumno A1.
- d) Alumnos que han suspendido todas las asignaturas que imparte algún profesor.
- e) Profesores valorados positivamente por todos los alumnos matriculados en alguna de las asignaturas que él imparte.

2) Responder en cálculo relacional de t-uplas a las siguientes consultas:

- a) Asignaturas impartidas por más de un profesor.
- b) Alumno que ha sacado la nota más alta en la asignatura de EDII.
- c) Alumnos que no han suspendido ninguna asignatura que les guste.
- d) Alumnos a los que, a pesar de valorar positivamente a todos los profesores de alguna asignatura, no les gustó la misma.
- e) Alumnos que siempre sacan más de un 7 en todas las asignaturas en las que se matriculan y son impartidas por el profesor P1.

3) Responder en cálculo relacional de dominios a las siguientes consultas:

- a) Alumnos que han aprobado alguna asignatura que les gusta.
- b) Alumnos que sólo han aprobado una única asignatura.
- c) Alumnos que han aprobado todas las asignaturas que les han gustado.
- d) Alumnos que valoran positivamente a todos los profesores de alguna de las asignaturas que les gusta.
- e) Profesores valorados positivamente por todos los alumnos aprobados en las asignaturas que él imparte.

4) Responder en SQL a las siguientes consultas:

- 5) Asignaturas que no gustan a nadie.
- 5) Nota media que han sacado los alumnos del profesor P1.
- 5) Asignatura con mayor número de suspensos.
- 5) Alumnos que han aprobado todas las asignaturas que les gustan.
- 5) Profesores valorados positivamente por, al menos, el 90% de los alumnos matriculados en alguna de las asignaturas que él imparte.



**Estructura de Datos y de la Información II**  
**Centro Superior de Informática**  
**Universidad de La Laguna**  
**9 de Diciembre de 1998**

La Universidad de La Laguna ha decidido hacer un seguimiento de las actividades deportivas que practican sus alumnos. Para ello ha creado una base de datos con el siguiente esquema:

Los atributos que se utilizan los vamos a abreviar utilizando la siguiente tabla:

Atributo	Significado
A	ALUMNO
D	DEPORTE
I	INSTALACIÓN

Las tablas son:

**GUSTA** (A, D)

**SIGNIFICADO:** Al alumno A le gusta el deporte D.

**CLAVE PRIMARIA:** (A, D).

**ASISTE** (A, I)

**SIGNIFICADO:** El alumno A asiste a la instalación deportiva I.

**CLAVE PRIMARIA:** (A, I).

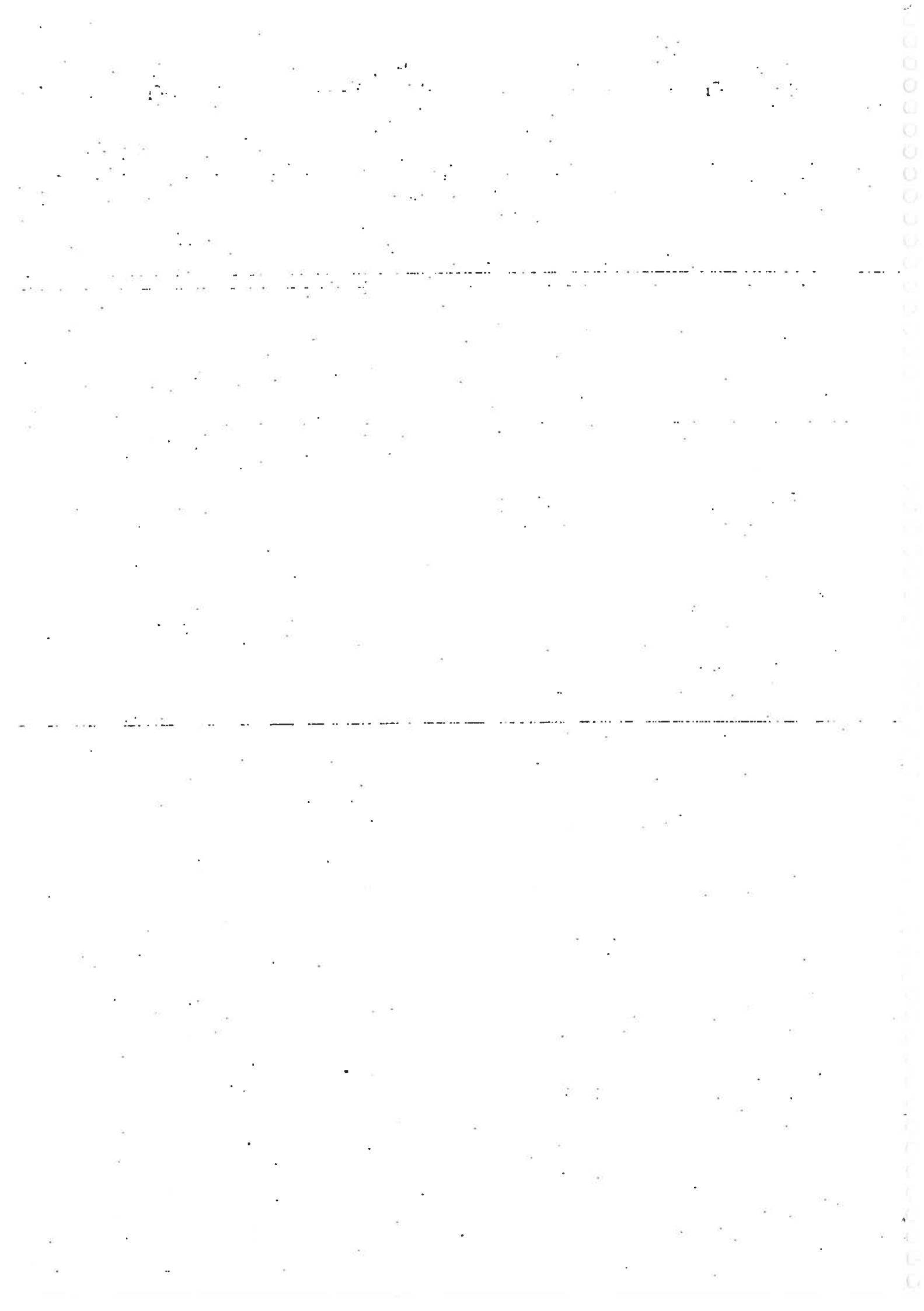
**OFERTA** (I, D)

**SIGNIFICADO:** La instalación I oferta (permite practicar) el deporte D.

**CLAVE PRIMARIA:** (I, D).

Nota: Se supone que si una persona asiste a una instalación deportiva es para practicar, entre los deportes ofertados, aquellos que le gustan.

- 1) Responder en álgebra relacional a las siguientes preguntas:
  - a) Alumnos a los que no les gusta ni el deporte 'A', ni el deporte 'B'.
  - b) Alumnos que asisten a alguna instalación que oferte, al menos, dos de los deportes que les gustan.
  - c) Alumnos que sólo asisten a instalaciones a las que no asiste 'Juan'.
  - d) Alumnos que practican todos los deportes que les gustan.
  - e) Alumnos que pueden practicar todos los deportes que les gustan en una misma instalación (aunque no asistan a ella).
- 2) Responder en cálculo relacional de t-uplas a las siguientes consultas:
  - a) Deportes que se ofertan en todas las instalaciones.
  - b) Alumnos que sólo asisten a instalaciones que ofertan algún deporte que les gustan.
  - c) Alumnos a los que les gustan alguno de los deportes que oferta, en exclusiva, la instalación 'A'.
  - d) Alumnos que practican todos los deportes que les gustan.
  - e) Alumnos que asisten a alguna instalación, en la que se puede practicar, sólo uno de los deportes que les gustan.
- 3) Responder en cálculo relacional de dominios a las siguientes consultas:
  - a) Deportes que gustan, a lo sumo, a 2 alumnos.
  - b) Deportes que son ofertados, simultáneamente, por todas las instalaciones.
  - c) Alumnos a los que no gustan ninguno de los deportes que oferta la instalación deportiva 'A'.
  - d) Alumnos que asisten a alguna instalación que no oferte ningún deporte que les gusten.
  - e) Alumnos que sólo asisten a instalaciones que ofertan todos los deportes que les gustan.
- 4) Responder en SQL a las siguientes consultas:
  - a) Número medio de deportes ofertado por las instalaciones.
  - b) Instalación a la que asisten más alumnos.
  - c) Deportes que oferta, en exclusiva, la instalación deportiva 'A'.
  - d) Instalaciones que permiten, al menos, los mismos deportes que la instalación 'A'.
  - e) Alumnos que sólo asisten a instalaciones que ofertan todos los deportes que les gustan.
  - f)



1. Responder en álgebra relacional a las siguientes preguntas.

a) Alumnos a los que no les gusta ni el deporte 'A', ni el deporte 'B'.

$$\begin{aligned} P(A)(S(D = "A")) \wedge (D = "B") & (GUSTA) \\ P(A)(GUSTA - A) \end{aligned}$$

b) Alumnos que asisten a alguna instalación que oferte, al menos, dos deportes que les gustan.

$$A_2 = A_2 = (ASISTE * OFERTA)$$

c) Alumnos que sólo asisten a instalaciones a los que no asiste Juan.

$$\begin{aligned} P(I)(S(A = "Juan"))(ASISTE) & \Rightarrow A \\ P(A)(ASISTE - A) \end{aligned}$$

d) Alumnos que practican todos los deportes que les gustan.

$$P(A)(GUSTA * ASISTE)$$

e) Alumnos que pueden practicar todos los deportes que les gustan en una misma instalación:

? (A) : (GUSTA A OFERTA)

El Gobierno de Canarias quiere llevar un control sobre el tráfico que sopor tan las diferentes carreteras de las islas. El esquema de la base de datos utiliza los siguientes atributos:

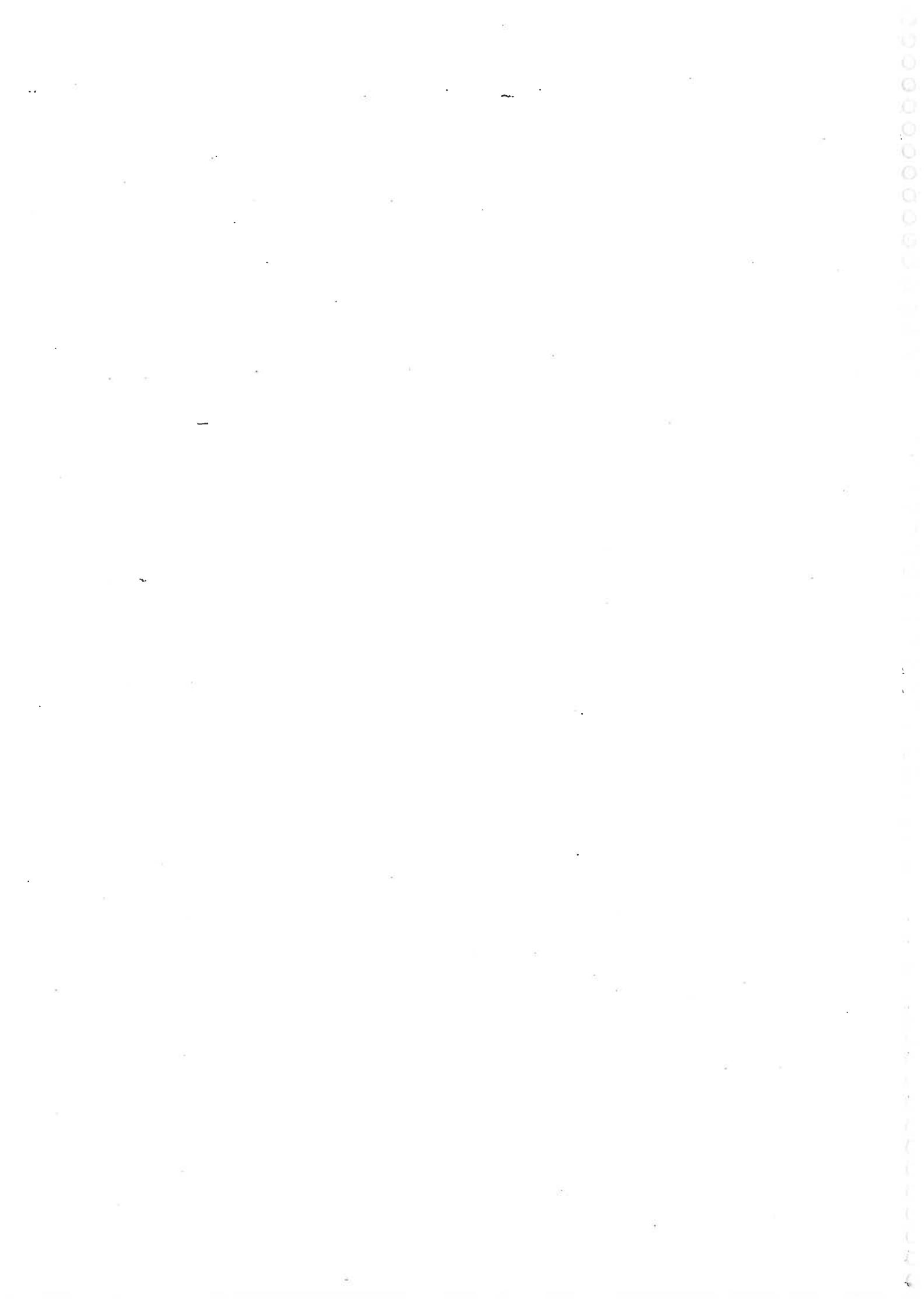
Atributo	Siguiente
CR	Carretera: TF1, TF5, ...
F	Fecha.
E	Estación de Aforo.
H	Hora en punto: 1:00, 2:00, ..., 23:00, 24:00
K	Punto kilométrico en el que está ubicada la estación.
KI	Kilómetro inicio.
KF	Kilómetro final.
M	Municipio.
NV	Número de vehículos aforados (contados).
T	Tipo de estación: espías, neumáticas, ...

Las tablas utilizadas son:

CARRERAS (CR, KI, KF, M) SIGNEFICADO: La carretera CR, en el tramo comprendido entre el kilómetro de inicio KI y el kilómetro final KF, pertenece al municipio M.  
CP: (CR, KI, KF)  
ESTACIONES (E, T, CR, K)  
SIGNEFICADO: La estación E es de tipo T y está ubicada en el kilómetro K de la carretera CR.  
CP: (E, F, H) SIGNEFICADO: La estación E, entre las H-1 y las H horas del día F, contabilizó un total de NV vehículos.  
AFOROS (E, F, H, NV) CP: (E, F, H)

Bases de Datos - Grado Ingeniería Informática  
Escuela Superior de Ingeniería y Tecnología  
Universidad de La Laguna  
1 de junio de 2017





- 1) Responder en SQL a las siguientes consultas:
- Aeropuertos que tienen al menos un avión de carga en cada una de sus pistas.
  - Aeropuertos con alguna pista en la que diariamente operan aviones de todos los tipos.
  - Aeropuertos con mayor número de operaciones realizadas en total el día '26-06-17'.
  - Aeropuertos tales que al menos el 70% de sus operaciones las realizan aviones de carga.
- 2) Responder en SQL a las siguientes consultas:
- Provincias que tienen al igual en promedio de otras dos de aeropuerto con excepción de 'Los Rodeos'.
  - Número medio de operaciones de aeropuerto con excepción de 'Los Rodeos'.
  - Aeropuerto con mayor número de operaciones realizadas en total el día '25-06-17'.
  - Aeropuerto en que la provincia de Madrid, con la pista de mayor longitud.
- 3) Responder en SQL a las siguientes consultas:
- Aeropuertos en los que cada día aterriza al menos un avión de carga en cada una de sus pistas.
  - Aeropuertos en los que diariamente operan aviones de todos los tipos.
  - Aeropuertos con alguna pista en la que tienen al igual en promedio de otras dos de aeropuerto.
  - Aeropuertos que tienen al igual en promedio de otras dos de aeropuerto.
- 4) Responder en SQL a las siguientes preguntas:
- Aeropuerto con más posibles valores: N, S, E, W, NE defecto es N.
  - Añade una columna QR a la tabla PISTAS con la orientación de las pistas. Valores posibles: N, S, E, W, NE defecto es N.
  - Crea una función PL/SQL que devuelva el número total de operaciones realizadas en un determinado aeropuerto en una fecha dada.
  - Descomponer R en FNB. ¿Se preservan las dependencias funcionales? Razóna la respuesta.

Attributo	Significado	Las tablas utilizadas son:
A	Aeropuerto	PI, PR
F	Fecha: '26-06-17', ...	OP
H	Hora: '09:15:24', ...	PI
L	Longitud de pista (en m).	OP
M	Operación aeropuerto: 1, 2, ...	PI
P	Provincia	PI
R	Tipo de avión: pasajeros, carga, mixto	PI

El Gobierno de España quiere registrar la actividad de los distintos aeropuertos del país. El esquema de la base de datos utiliza los siguientes atributos:

