

Bases de Datos

Tema 01. Introducción a las BD Relacionales



Marta Elena Zorrilla Pantaleón

Rafael Duque Medina

DPTO. DE MATEMÁTICAS, ESTADÍSTICA Y
COMPUTACIÓN

Este tema se publica bajo Licencia:

[Creative Commons BY-NC-SA 3.0](https://creativecommons.org/licenses/by-nc-sa/3.0/)

Tabla de contenidos

- ▶ La importancia de las BDs
- ▶ Concepto de Base de Datos y SGBD
- ▶ De los sistemas de ficheros a la BD relacional. Niveles de abstracción
 - ▶ Razones que justifican el uso de BD y cuando No
- ▶ Una introducción al modelo relacional.
 - ▶ Tablas, relaciones e índices
 - ▶ Problemas de un mal diseño de BD
 - ▶ Introducción al lenguaje SQL
 - ▶ Restricciones de integridad y reglas de negocio
- ▶ Concepto de transacción
- ▶ Arquitectura de un SGBDR
 - ▶ Componentes
 - ▶ Usuarios
- ▶ Cuestiones

Bibliografía

▶ Básica

- ▶ Cap. 1 y 2. Elmasri, R., Navathe, S.B., Fundamentos de Sistemas de Bases de Datos, 5ª edición, Pearson Education, 2008.
- ▶ Cap. 1. Mora, E., Zorrilla, M. E., Díaz de Entresotos, J. Iniciación a las bases de datos con Access 2002. Díaz de Santos, 2003.
- ▶ Cap. 1. Silberschatz, A., Korth, H.F., Sudarshan, S., Fundamentos de Bases de Datos, 5ª edición, Madrid, 2006.

▶ Complementaria

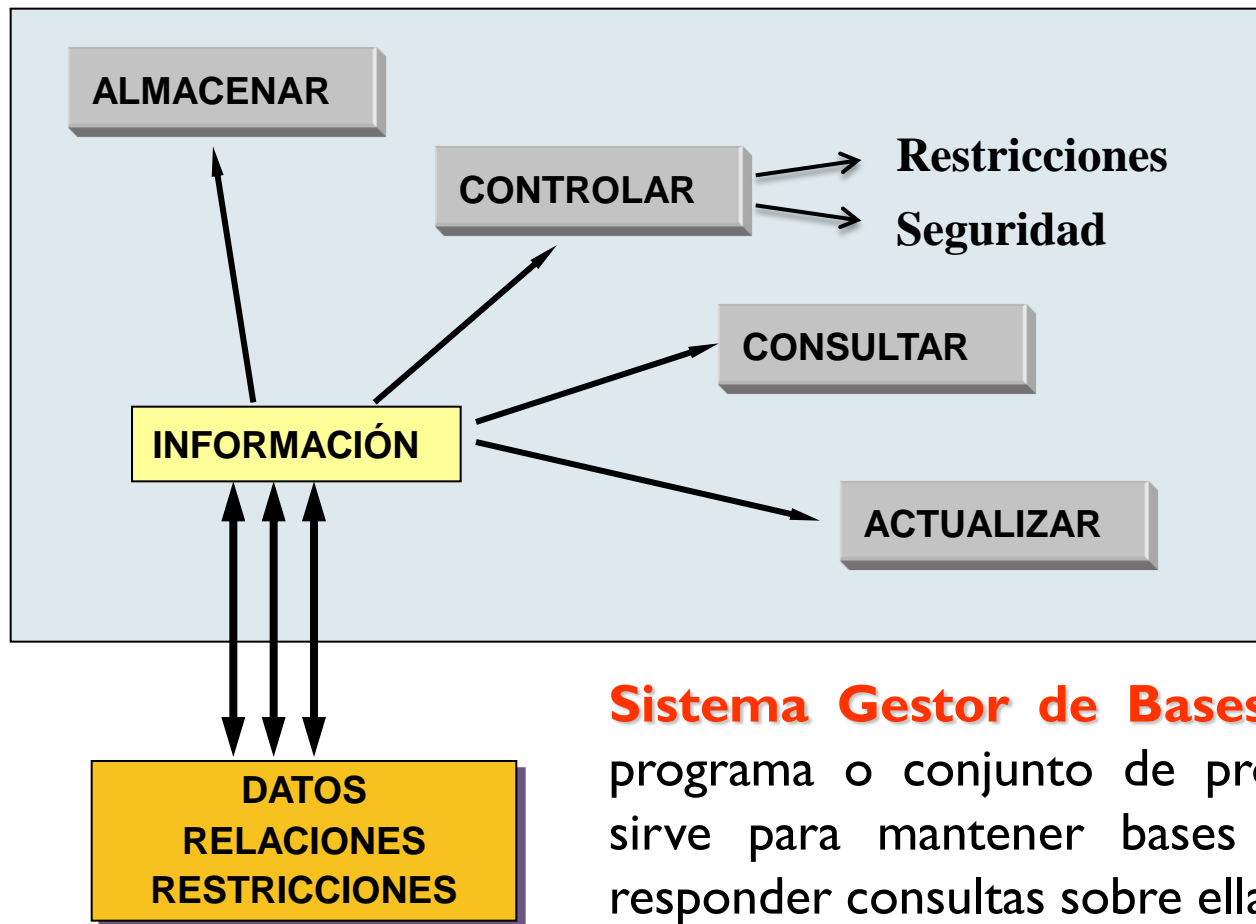
- ▶ Piattini et al. Tecnología y diseño de bases de datos. RA-MA, 2006.
- ▶ Pons, O. et al. Introducción a los sistemas de bases de datos. Paraninfo. 2008
- ▶ García Molina, H., Ullman, J., Widom, J. Database systems: the complete book. 2nd ed. Pearson Education International, cop. 2009

- ▶ **¿Qué es una base de datos?**
- ▶ **¿Utilizáis alguna?**
- ▶ **¿Consideráis que son importantes?**
- ▶ **¿Qué interfaces presentan?**

- ▶ Estructuras de datos gestionadas por un conjunto de programas que permiten almacenar grandes cantidades de información y manipularla de forma eficiente
- ▶ Dan soporte a:
 - ▶ Procesos transaccionales
(compras, préstamos, gestión académica, experimentos científicos...)
 - ▶ Planificación de trabajos
(Workflow, scheduler, ...)
 - ▶ Oferta de servicios
(búsquedas bibliográficas, videotecas,...)
 - ▶ La inteligencia de negocio
 - ▶ DW, OLAP, minería de datos



Base de Datos (def): colección organizada de datos, relativa a un problema concreto, que puede ser compartida por un conjunto de usuarios/aplicaciones. Sirven para:



Sistema Gestor de Bases de Datos: programa o conjunto de programas que sirve para mantener bases de datos y responder consultas sobre ellas.

- ▶ En los primeros años (década de los 70 y principios de los 80), las aplicaciones de BD se construían directamente sobre los sistemas de ficheros
- ▶ Esto tenía serias desventajas :
 - ▶ **Redundancia e inconsistencia de los datos**
 - ▶ Múltiples formatos de ficheros, duplicación de información en diferentes ficheros
 - ▶ Ejemplo
 - Cuenta (nombre, direccion, tfno, n° cc, cantidad)
 - Ingreso (nombre, direccion, tfno, n° cc, cantidad, fecha)
 - ▶ **Datos aislados** — Múltiples formatos y ficheros
 - ▶ **Problemas de integridad**
 - ▶ Restricciones de integridad como “sexo= M o F” se encuentra en el código del programa y no establecido explícitamente en la estructura del fichero
 - ▶ Difícil y costoso el modificar o añadir nuevas restricciones (altura>=0, fecha>=hoy, etc.)

▶ Desventajas (cont.)

▶ **Dificultad en el acceso a los datos**

- ▶ Necesidad de escribir un programa para realizar cada proceso

▶ **Atomicidad de las actualizaciones**

- ▶ Pérdida de consistencia por realización de actualizaciones parciales
- ▶ Ej.: Transferencia de fondos de una cuenta a otra (debe restarse una cantidad de una cuenta y sumarse esa misma cantidad a la otra, o no hacerse nada)

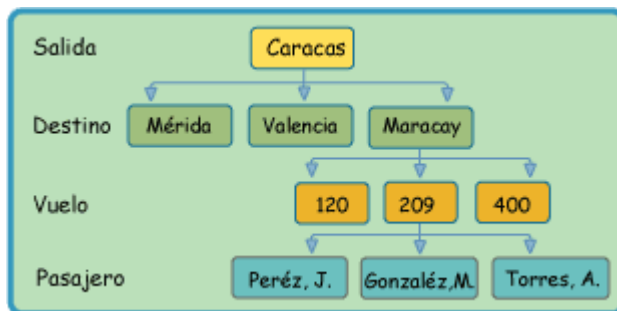
▶ **Acceso concurrente por múltiples usuarios**

- ▶ Necesidad de acceso concurrente para incrementar el rendimiento
- ▶ Acceso concurrente bajo control para evitar inconsistencias
 - Ej.: Se ha de impedir que dos personas estén actualizando la misma cuenta corriente al tiempo

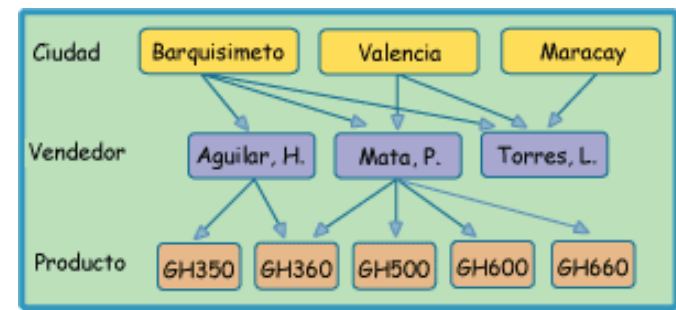
▶ **Problemas de seguridad**

- ▶ Difícil permitir el acceso a los usuarios a parte de los datos y/o determinadas acciones (actualizar cuenta, leer datos personales, insertar clientes, etc.)

- ▶ La gestión y almacenamiento de datos es el que impulsa el crecimiento del uso de las computadoras. De 1950 a 1960, se desarrollaron las cintas magnéticas para el almacenamiento de datos cuya lectura era secuencial. Apareció el lenguaje **COBOL**, primer lenguaje que tiene una parte donde se describen los datos y los ficheros que van a ser utilizados de manera independiente a las acciones que se van a realizar.
- ▶ En la década de los 70, aparecieron los discos magnéticos lo que permitió el acceso directo a los datos. Surgieron las **BD jerárquicas y en red**, en las que el tratamiento de información requería conocer detalles de implementación (uso de punteros) y la codificación de consultas se realizaba de forma procedimental
- ▶ **Codd** definió el **modelo relacional** (1970)
 - Modelo teórico bien fundamentado
 - Independencia lógica y física de los datos
 - Lenguaje de consulta declarativo, SQL



Jerárquica (IMS de IBM)



Red (IDMS/R de Cullinet)

- ▶ Pero hasta **1980** no aparecieron gestores relacionales comerciales (Oracle, IBM DB2, Ingres,...) con buen rendimiento y en los que el diseño y el mantenimiento de las BDs resulta más sencillo (independencia física y lógica).
- ▶ Asimismo comenzaron los:
 - ▶ Estudios de BD distribuidas y paralelas
 - ▶ Y de BD orientadas a objetos
- ▶ Desde **1990** se encuentran en el mercado:
 - ▶ BD objeto-relacionales
 - ▶ BD dimensionales (tecnología OLAP)
 - ▶ BD XML
 - ▶ BD geográficas
 - ▶ etc.

TABLA DE CLIENTES				
C.I.	Nombre	Identif./C.	Dirección	Teléfono
16325825	Rivas, Luis	RL708	23654 Santa Rosa	15325948
12035824	Torres, Yessy	TY011	2536 Calle Roma	12369581
10356528	Cruz, Carlos	CC125	2514 Av. Urdaneta	10256985

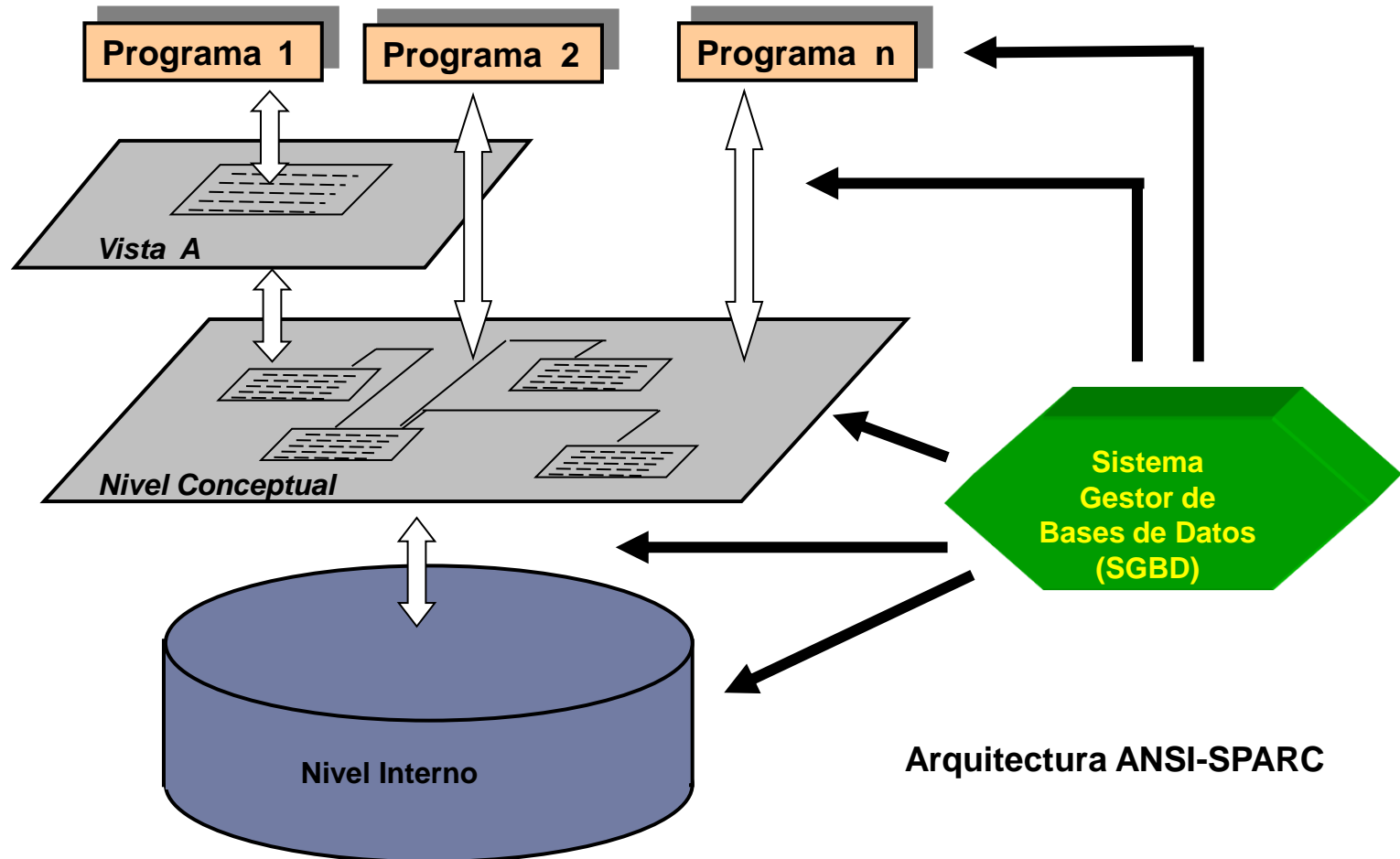
CAMPO CLAVE DE:

TABLA DE PEDIDOS					
Núm. Pedido	Identif./C.	Fecha	Monto	Embarque	Cargo envío
000454	RL708	11/02/2005	4.000.080,50	E401	10
000455	TY011	06/05/2005	1.032.200,00	E406	15
000456	CC125	07/05/2005	7.000.230,20	E900	10

Modelo relacional

Niveles de abstracción: independencia física y lógica

La finalidad de trabajar con técnicas de BD relacionales es disfrutar de una visión abstracta de los datos que facilite el desarrollo y uso de aplicaciones. Esto es, centrarse en determinar qué datos se han de almacenar y cómo se relacionan, olvidándose de los detalles internos de cómo el gestor los almacenan y gestiona.



- ▶ **ANSI-SPARC (1975)** propuso la arquitectura en tres niveles para conseguir la separación entre los programas de aplicación y los datos
 - ▶ **Nivel interno:** Describe la estructura de almacenamiento físico de base de datos (árboles B+, montones, estructura de índices...). Los datos se almacenan en este nivel. Cada SGBD implementa su propio nivel interno. No estándar.
 - ▶ **Nivel conceptual:** Representación del conjunto de datos correspondiente al problema de información a gestionar. Recoge el esquema conceptual, esto es, la estructura de la base de datos en términos de elementos lógicos (entidades, atributos, relaciones y restricciones), ocultando los detalles físicos de almacenamiento.
 - ▶ **Nivel externo o de vistas:** Esquemas que recogen las distintas perspectivas de los usuarios y/o aplicaciones para cada proceso (compras, matrícula, expedientes, etc.). Permiten ocultar información por cuestiones de seguridad (Ley de protección de datos, datos sensibles para la seguridad de una empresa, etc.)

- ▶ La arquitectura de tres niveles es útil para explicar el concepto de *independencia de datos* :
 - ▶ **Independencia lógica:** capacidad de cambiar el nivel conceptual sin tener que cambiar las vistas ni los programas de aplicación
 - ▶ Añadir restricciones de dominio, incorporar un nuevo atributo, añadir nuevas relaciones,...
 - ▶ **Independencia física:** capacidad de cambiar el nivel interno sin tener que cambiar ni el nivel conceptual ni nivel externo.
 - ▶ Añadir índices, ampliar espacio de almacenamiento, realizar particiones, etc.

DATE (1981) define la **independencia** como “la inmunidad de las aplicaciones ante cambios de la estructura del almacenamiento y de los métodos de acceso”

- Flexibilidad de adaptación a cada problema.
- Optimización en la gestión de la información.
- Garantiza la independencia física y lógica de los datos.
- Control de la integridad de los datos.
- Garantía sobre la consistencia de la información.
- Facilidad de acceso concurrente.
- Protección ante fallos del sistema.
- Seguridad ante accesos restringidos.

⊕ Ficheros

- Aplicaciones sencillas cuyo esquema es poco probable que cambie
- Cuando hay restricciones de tiempo real
- Cuando no hay acceso multiusuario

⊕ NoSQL (non-relational databases)

- Grandes volúmenes de datos sin requisitos ACID (transacciones)
- Sistemas distribuidos y escalables
 - Ej: Google (BigTable), Facebook (Cassandra), etc.

Personal

NOMBRE	PROFESION	LOCALIDAD
Pedro	profesor	Santander
Luis	estudiante	Santander
María	estudiante	Las Palmas
Ana	estudiante	Madrid

Los datos se conciben agrupados en forma de tablas

Cada fila establece una relación entre un conjunto de valores

Los operadores de consulta generan nuevas tablas

```
SELECT NOMBRE, LOCALIDAD FROM Personal
WHERE PROFESION = 'estudiante'
```

NOMBRE	LOCALIDAD
Luis	Santander
María	Las Palmas
Ana	Madrid

BANCOS

ENTIDAD	NOMBRE
0893	Santander
0059	Popular
3428	BBVA
5632	Banesto

- Toda tabla tiene una columna o conjunto de columnas que permiten identificar cada una de sus filas; éstas componen la llamada **clave principal (Primary Key, PK)** de la tabla.

- Los valores de la clave principal no se pueden repetir (Entidad en Bancos, y la pareja Entidad-Codigo_Oficina en Oficinas) .

- Unas tablas se refieren a otras mediante vínculos de tipo jerárquico.

- Este vínculo de referencia entre dos tablas se establece mediante columnas de igual tipo de dato en las dos tablas y se denomina **clave ajena o Foreign Key (FK)**.

OFICINAS

ENTIDAD	CODIGO_OFICINA	POBLACION	DIRECCION
0893	001	Madrid	Castellana, 73
3428	022	Las Palmas	Triana, 21
0893	022	Gáldar	R. Moreno, 3
5632	213	Oviedo	Uría, 43
0893	300	Barcelona	Diagonal, 435

- La referencia de una fila de una tabla a otra de la otra tabla se produce cuando ambas tienen el mismo valor (columna Entidad en Oficinas con relación a Entidad en Bancos).

TIPOS DE DATOS: cada columna de una tabla tiene asociado un tipo de dato. Existen un subconjunto estándar pero hay otros dependientes del gestor que se utilice

Cadena de caracteres (char).

Cada carácter requiere un byte para su almacenamiento.

Numérico (numeric).

Enteros: Cortos (smallint).

Largos (integer).

Decimales: definidos por su precisión y escala (decimal)

Notación científica: Simple precisión (smallfloat)

Doble precisión (float)

Fecha (date) y hora (datetime).

Diferentes opciones según nivel de precisión.

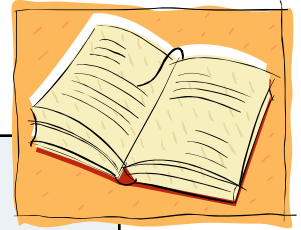
Objeto grande (large object, LOB).

Binary large object (blob).

Character large object (clob).

Tipos definidos por el usuario.

ÍNDICES: estructuras de datos adicionales que permiten



- **Realizar búsquedas más ágiles en las tablas....**

aunque suponen una sobrecarga para realizar las actualizaciones de datos

ej: campo “título” en una tabla que recoja los libros disponibles en una biblioteca. Se justifica por ser el campo sobre el que se realizan más consultas

Conviene definirlos sobre las columnas con FK

- **Establecer restricciones de unicidad**

no permitir repeticiones de un valor en la columna o columnas afectadas por el índice

ej: campo “ISBN” en una tabla que recoja los libros disponibles en una biblioteca


El problema del diseño (1)

Un administrador de fincas urbanas quiere gestionar la siguiente información:

<u>PROPIETARIOS:</u>	DNI (único)	<u>LOCALES:</u>	CODIGO (único)
	NOMBRE		UBICACION
	DIRECCION		SUPERFICIE

Primera alternativa

Locales_propietarios

 CODIGO	UBICACION	SUPERFICIE	DNI	NOMBRE	DIRECCION

 PK

Problemas del diseño

- Repetición de información
- Posibilidad de contradicciones en los datos
- Problemas en inserciones
- Pérdida de información al borrar

Segunda alternativa

Propietarios

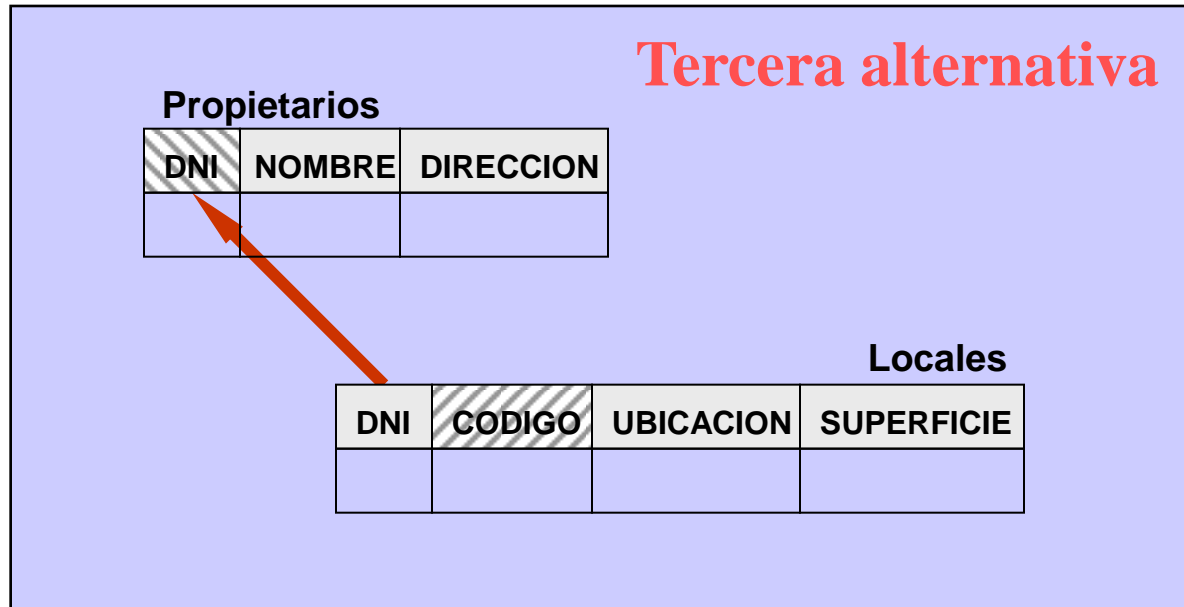
DNI	NOMBRE	DIRECCION

Locales

CODIGO	UBICACION	SUPERFICIE

Problemas del diseño

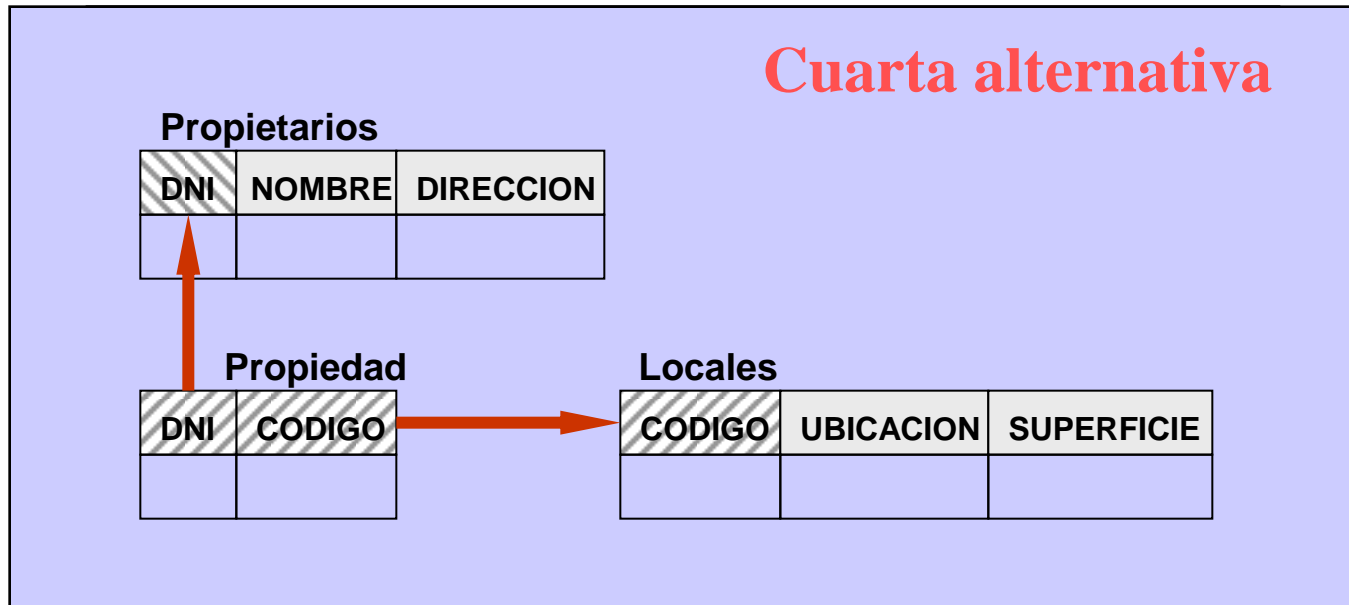
- Pérdida de dependencias funcionales
¿de quién es cada local?



Problemas del diseño

- Sólo un propietario para cada local

La referencia entre tablas siempre es una relación “de 1 a n” o “de n a 1”



Si se desea que un propietario pueda tener varios locales y, al mismo tiempo, que un local pueda ser de varios propietarios, la relación es simétrica, es “de n a n” y no puede ser resuelta con sólo dos tablas. Para conseguirlo, es necesario introducir una tabla auxiliar que tenga relaciones de “de n a 1” con las de propietarios y locales.

Lenguaje declarativo de acceso a los datos.

Estándar para las bases de datos relacionales y objeto-relacionales.

Incluye la capacidad de actuar tanto sobre la estructura de la base de datos como sobre los propios datos.

Desarrollado en el San José Research Center (IBM)
Fue utilizado por primera vez en 1970.

En 1986: ANSI (American National Standards Institute) e
ISO (International Standards Organization)
publicaron las normas SQL/ANSI y SQL-86.

Actualizaciones SQL-92, SQL-1999 y, en vigor SQL:2003

```
CREATE DATABASE GESTION;
```

```
CREATE TABLE PROPIETARIOS
```

```
(DNI                CHAR(10) NOT NULL CONSTRAINT pk_prop PRIMARY KEY,  
 NOMBRE            CHAR(25) NOT NULL,  
 DIRECCION         CHAR(30));
```

```
CREATE TABLE LOCALES
```

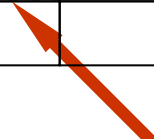
```
(CODIGO            CHAR(5) NOT NULL CONSTRAINT pk_loc PRIMARY KEY,  
 DNI               CHAR(10) NOT NULL,  
 UBICACIÓN        CHAR(4) NOT NULL,  
 SUPERFICIE       DEC(8,2) NOT NULL,  
 CONSTRAINT fk_loc FOREIGN KEY ( DNI ) REFERENCES PROPIETARIOS ( DNI ));
```

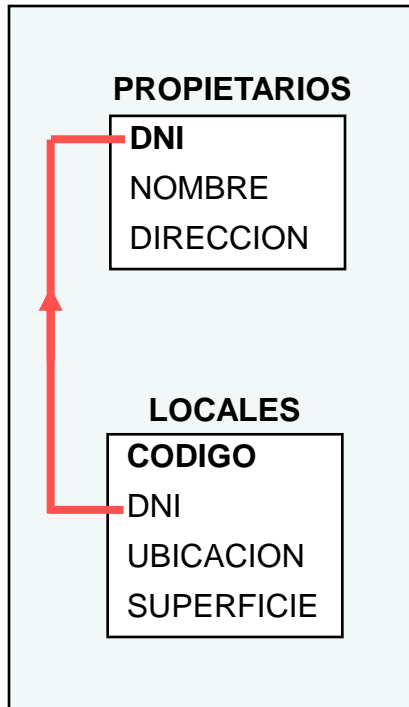
PROPIETARIOS

DNI	NOMBRE	DIRECCION

LOCALES

CODIGO	DNI	UBICACION	SUPERFICIE





Insertar una nueva fila en la tabla **PROPIETARIOS**

```
INSERT INTO PROPIETARIOS (DNI, NOMBRE, DIRECCION)  
VALUES ('13234567R', 'Sanz, Luis', 'Gran Vía 26')
```

Encontrar los locales con superficie mayor que 200 y su propietario

```
SELECT CODIGO, UBICACION, NOMBRE, DIRECCION  
FROM LOCALES, PROPIETARIOS  
WHERE LOCALES.DNI = PROPIETARIOS.DNI AND  
SUPERFICIE > 200
```

Resultado

CODIGO	UBICACION	NOMBRE	DIRECCION
L-31	Alta 236	Sanz, Luis	Gran Vía 26
L-234	Bailén 46	Laso, Ana	Isabel II 38
L-9	Cuesta 2	Sanz , Luis	Gran Vía 26
L-302	Becedo 10	Fe, Pedro	

Modificar la dirección del propietario cuyo D.N.I.
es 20333444F

```
UPDATE PROPIETARIOS SET DIRECCION ='Alta 87'  
WHERE DNI = '20333444F'
```

Borrar el local de código L-234

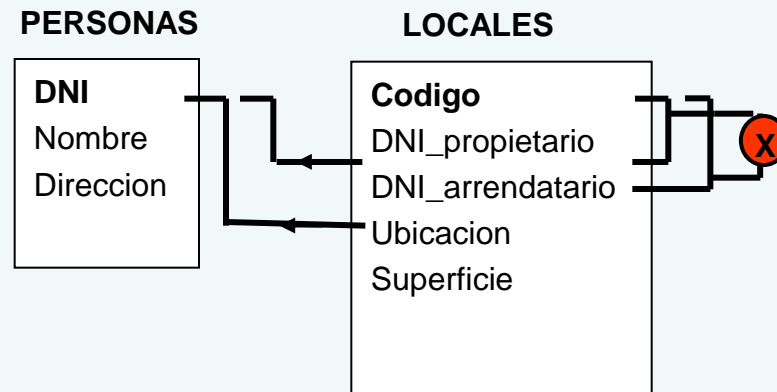
```
DELETE FROM LOCALES  
WHERE CODIGO = 'L-234'
```

Ejemplo: para cada fila de la tabla JUGADORES_BALONCESTO

- **NOT NULL**: Todas las columnas excepto ESTATURA son obligatorias de rellenar
- **CHECK**: el SEXO solo puede tomar los valores M o F y la ESTATURA debe ser superior a 1,75
- **DEFAULT**: la fecha de alta toma por defecto la del sistema, si ésta no se especifica (no es restricción en sí misma, si no una ayuda para el usuario final)

```
CREATE TABLE JUGADORES_BALONCESTO
(DNI          CHAR(10)  NOT NULL,
 NOMBRE       CHAR(25)  NOT NULL,
 DIRECCION    CHAR(30)  NOT NULL,
 TELEFONO     CHAR(15),
 SEXO         CHAR(1)   CHECK ( SEXO in ('M', 'F') NOT NULL,
 FE_ALTA      DATE      DEFAULT getdate()      NOT NULL,
 ESTATURA    DEC(3,2)
 CONSTRAINT Valor_estatura CHECK (ESTATURA > 1,75)
PRIMARY KEY ( DNI ));
```

Ejemplo: para cada fila de la tabla LOCALES creada previamente, los valores de DNI_propietario y DNI_arrendatario no pueden ser iguales.



ALTER TABLE LOCALES WITH NOCHECK ADD

CONSTRAINT CK_locales CHECK (DNI_propietario <> DNI_arrendatario)

Los **triggers** (disparadores) de manipulación son procesos predefinidos que entran en acción en respuesta a eventos específicos de manipulación de datos (insert, update, delete).

Son más flexibles que los **asertos** para expresar restricciones semánticas.

Generalmente se utilizan para:

- recoger restricciones complejas (reglas de negocio)
- automatizar procesos
- anotar acciones (log)

Los incluyen generalmente los gestores aunque su codificación no es estándar

Ejemplo: para cada fila de la tabla LOCALES, los valores de DNI_propietario y DNI_arrendatario no pueden ser iguales.

**AHORA LO CONTROLAMOS POR DISPARADOR,
SOLO A MODO DE EJEMPLO,
YA QUE SI SE PUEDE DEFINIR MEDIANTE CHECK
RESULTARÁ MÁS EFICIENTE**

PERSONAS

DNI
Nombre
Direccion

LOCALES

Codigo
DNI_propietario
DNI_arrendatario
Ubicacion
Superficie



Ejemplo disparador en T-SQL

```
UPDATE LOCALES SET DNI_propietario = '60601602'
WHERE DNI_propietario = '40401402'
```

LOCALES

Codigo	DNI_propietario	DNI_arrendatario	Ubicacion	Superficie
L-31	50501502	60601602	Alta 236	220
L-234	40401402	50501502	Bailén 46	350
L-9	30301302	70701702	Cuesta 2	280
L-302	40401402	60601602	Becedo 10	255

Al ejecutar la instrucción UPDATE sobre la tabla LOCALES, entra en acción el disparador. Este hace uso de la tabla **inserted**, tabla que SQL Server utiliza para almacenar las filas afectadas por la instrucción y que guardará en la BD si no hay error. Esta tabla y la tabla **deleted** (que contiene las filas que se sustituyen o eliminan de la BD) se utilizan para realizar el control de reglas integridad. En TSQL, el programador es el responsable de cerrar la transacción con error (rollback)

```
CREATE TRIGGER CTRL_locales ON LOCALES FOR INSERT, UPDATE AS
BEGIN
```

```
IF ( SELECT count(*) FROM inserted
WHERE inserted .dni_propietario= inserted.dni_arrendatario) >0
```

```
BEGIN
```

```
RAISERROR ('El DNI del propietario no puede coincidir con el DNI del Arrendatario.', 16, 1)
```

```
ROLLBACK TRANSACTION
```

```
RETURN
```

```
END
```

```
END
```

inserted

Codigo	DNI_propietario	DNI_arrendatario	Ubicacion	Superficie
L-234	60601602	50501502	Bailén 46	350
L-302	60601602	60601602	Becedo 10	255

Transacción: conjunto de operaciones de manipulación de datos que deben ser consideradas como una unidad. Las debe definir el programador. Todas las operaciones que se ejecutan individualmente en un gestor ACID son transaccionales.

Propiedades (ACID):

ATOMICIDAD: todo o nada

CONSISTENCIA: coherencia de los datos

AISLAMIENTO: serialización de transacciones

DURABILIDAD: los cambios son permanentes

BEGIN TRANSACTION

UPDATE CUENTA

SET saldo = saldo + 50

WHERE numero_cuenta = '0893'

UPDATE CUENTA

SET saldo = saldo - 50

WHERE numero_cuenta = '2345'

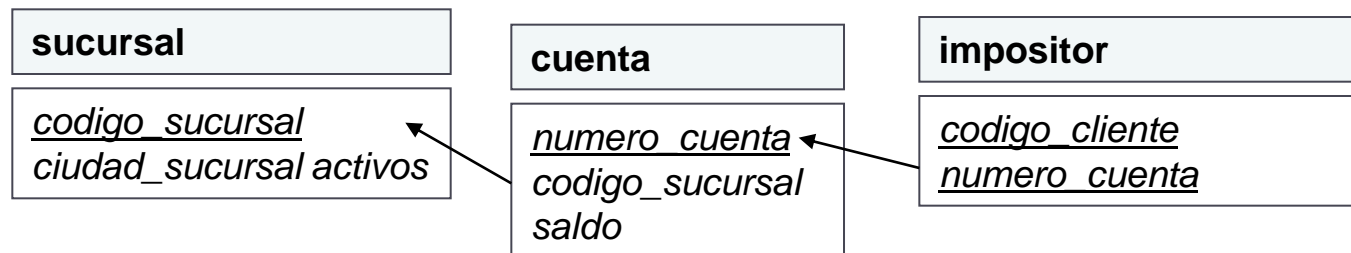
Si no ha habido ningún error, se confirman los cambios

COMMIT TRANSACTION

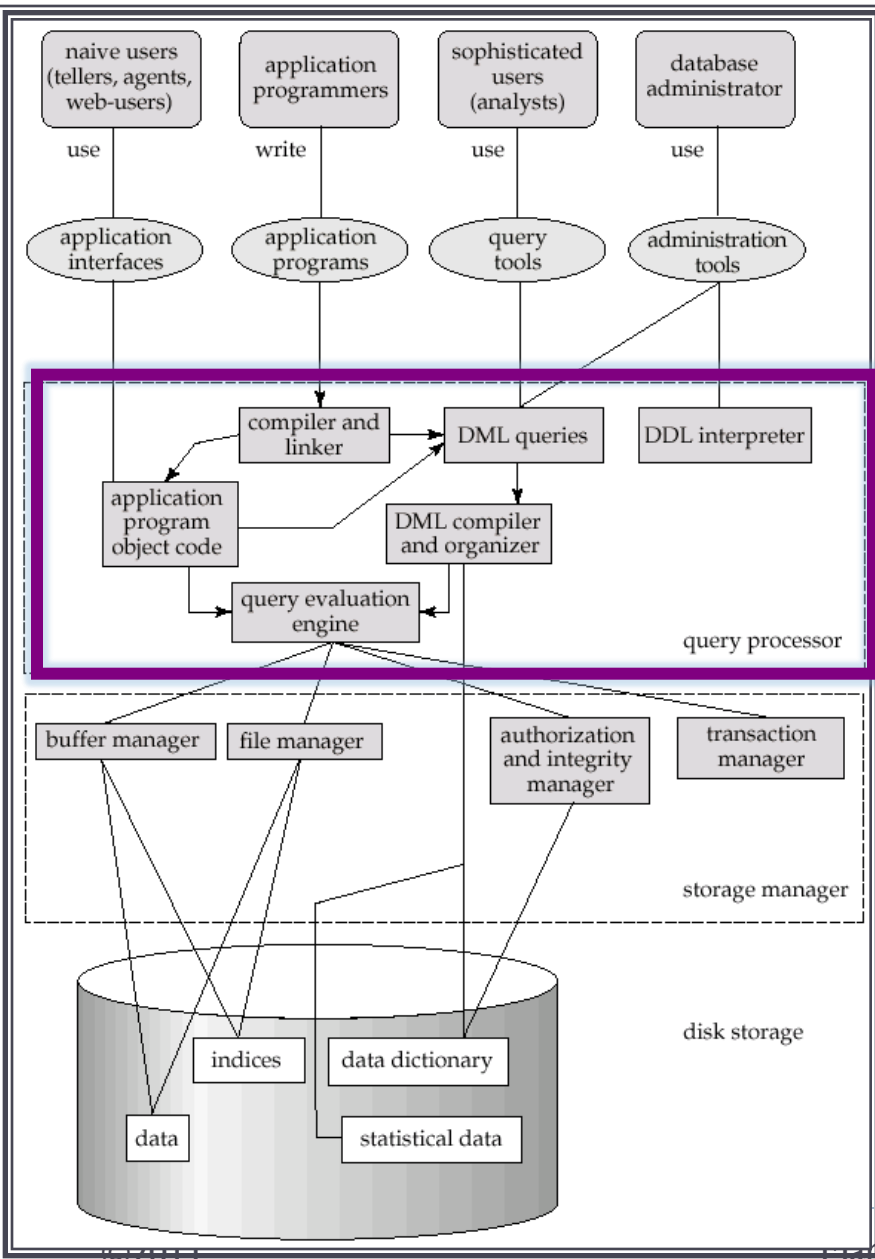
*Y si ha habido algún error, **no** se almacenan los cambios*

ROLLBACK TRANSACTION

Ejemplo: Transferencia de 50€ de la cuenta nº 0893 a la 2345

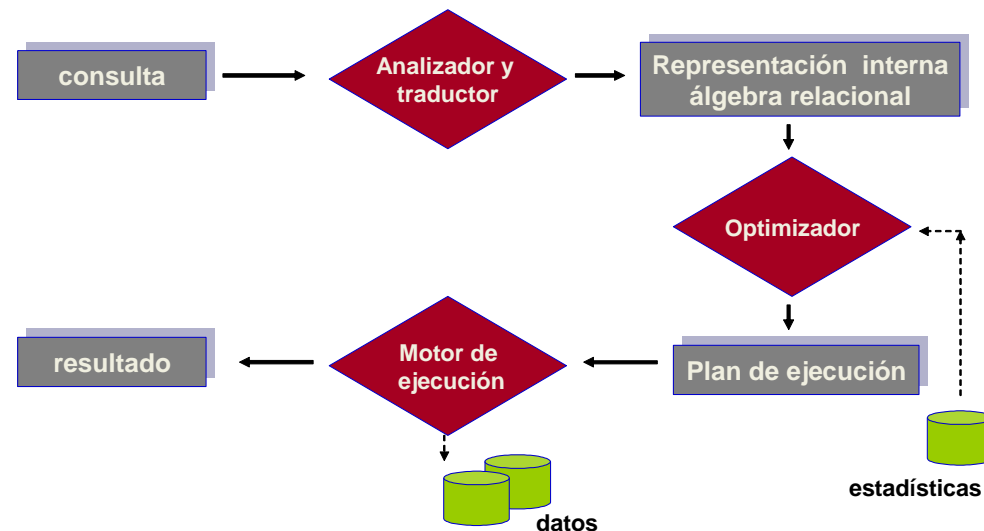


Arquitectura de un SGBDR

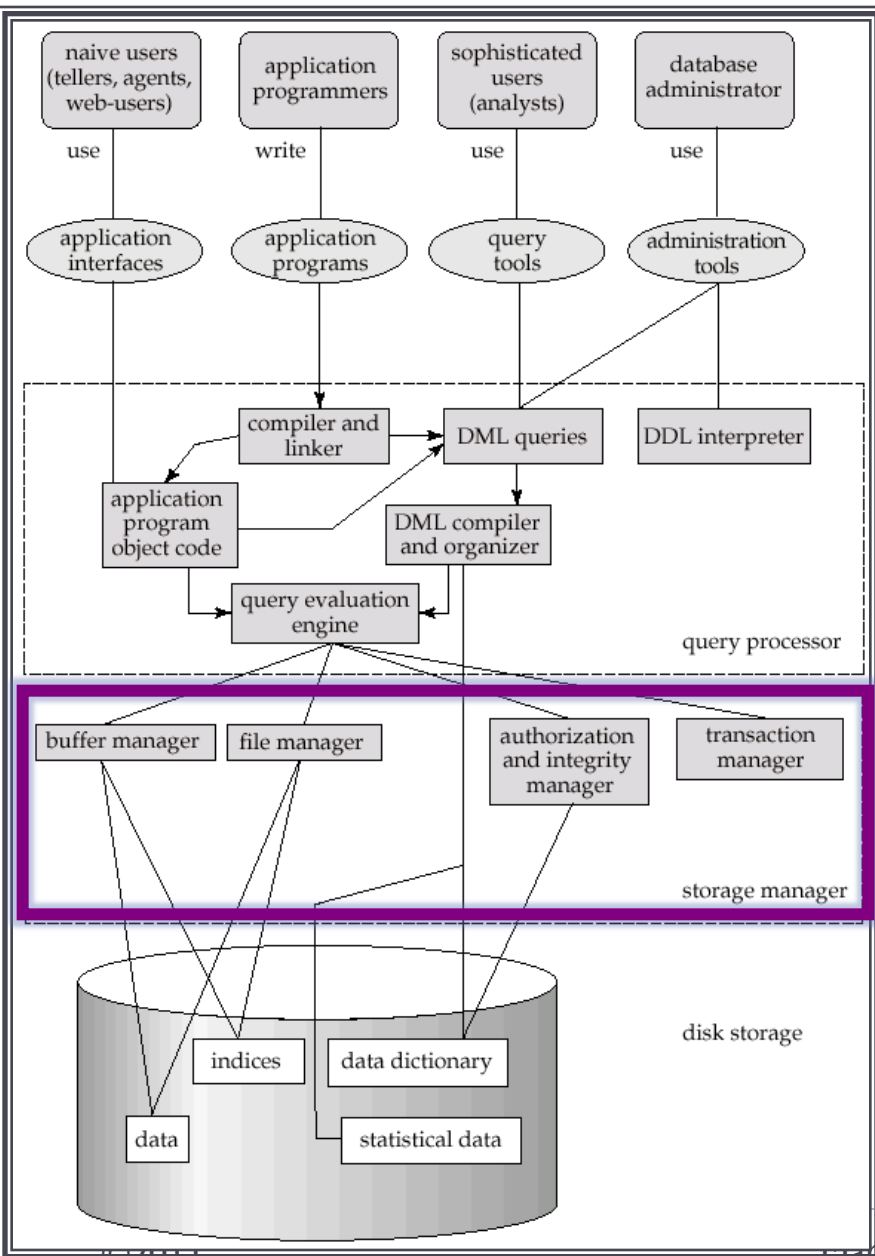


El **Procesador de consultas** es el módulo responsable de:

- * interpretar las instrucciones de definición y registrar su definición en el diccionario de datos
- * traducir las instrucciones LMD en el lenguaje del motor de evaluación para determinar el plan de ejecución.
- * ejecutar las consultas solicitadas



Arquitectura de un SGBDR (y 2)



El **Gestor de almacenamiento** es el módulo que proporciona la interfaz entre los datos de bajo nivel almacenados en la BD y los programas de aplicación y las consultas remitidas al sistema.

Es el responsable de:

- interactuar con el gestor de archivos
- almacenamiento eficiente y recuperación y actualización de datos

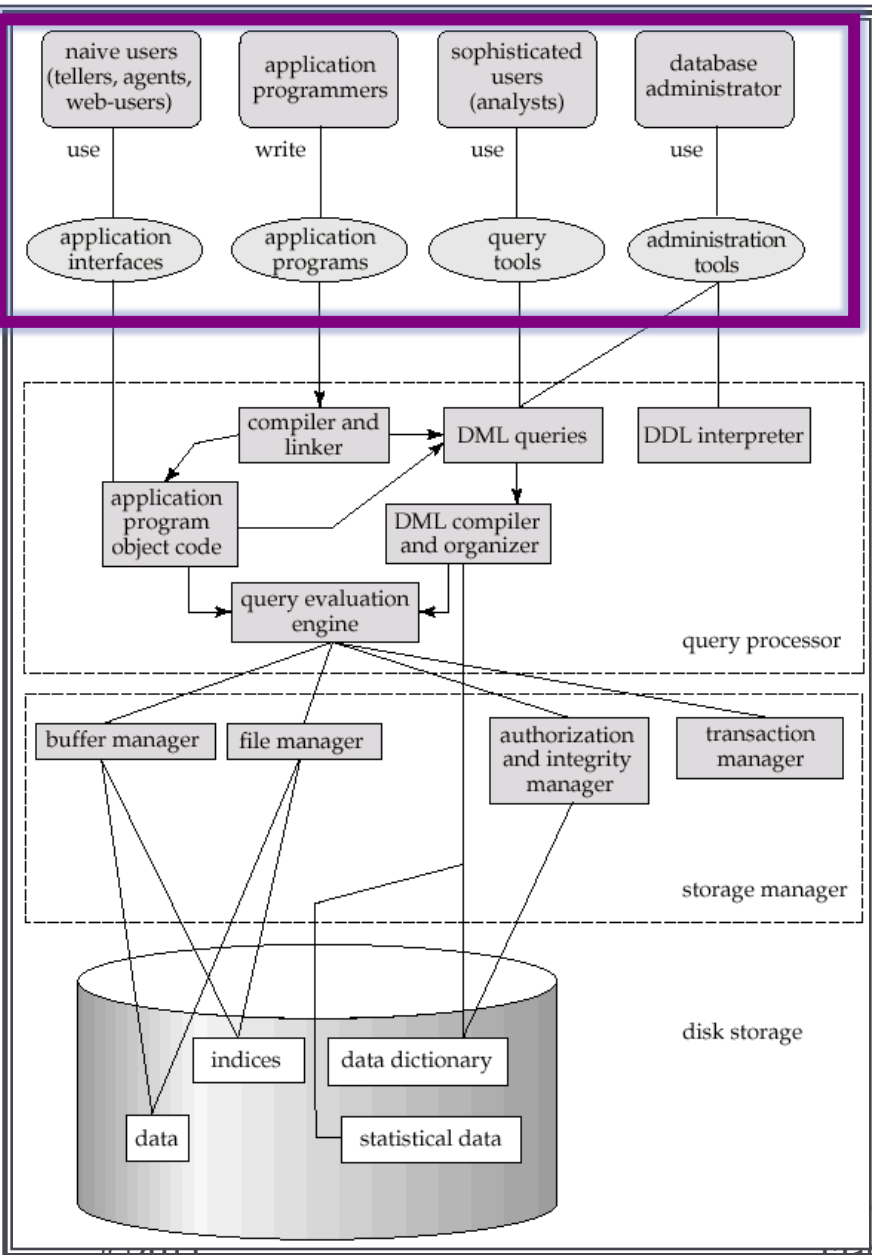
* El **Gestor de Transacciones** asegura la Atomicidad y la Durabilidad de las transacciones a pesar de fallos en el sistema (p. ej. Corte de luz, caída del S.O.) o de las transacciones establecidas en los programas.

- **Gestor de Concurrencia** controla la interacción entre las transacciones concurrentes (aislamiento) para garantizar la consistencia de la información

- **Gestor de Recuperación** permite retornar a una situación estable.

* El **Gestor de Archivos** gestiona la asignación de espacio en disco y las estructuras de datos

* El **Gestor de Memoria Intermedia** trae los datos del disco a la caché.



Los tipos de usuario se determinan por el tipo de interacción que realizan:

- ▶ **Usuarios normales** – invocan programas de aplicación que se han escrito previamente
 - ▶ E.j. acceso a BD en la Web (cuentas bancarias, carritos de la compra, etc...)
- ▶ **Programadores de aplicación** – escriben programas que embeben las llamadas a la BD. Utilizan herramientas como Eclipse, .Net,...)
- ▶ **Usuarios sofisticados** – interactúan con el sistema sin escribir programas, trabajan con el LDD y LMD
- ▶ **Usuarios especializados** – escriben aplicaciones de bases de datos especializadas que no encajan con el procesamiento tradicional.
 - ▶ E.j. BD con tipos de datos complejos (CAD, multimedia...)
- ▶ **Administrador de bases de datos** – responsable del mantenimiento del gestor (instalación, sintonizado, rendimiento, seguridad, etc.)

Cuestiones



1. Defina qué es una Base de datos relacional
2. ¿Qué restricciones se pueden recoger en una BD Relacional? Ponga un ejemplo de cada una.
3. ¿Qué significa independencia física y lógica de los datos?
4. ¿Qué se entiende por integridad y consistencia de los datos? ¿Cómo se pierde la consistencia?
5. Indique las instrucciones del lenguaje SQL que permiten realizar la manipulación de datos.
6. ¿Qué es una transacción? ¿Cuándo se deben utilizar?
7. Enumere los elementos que componen un SGBDR e indique la función de cada uno.