1. Introducción (PDF)

1.1. Información y datos

La información es valor que se extrae de los datos. Mucho software se dedica a gestionar datos. Esto es complicado debido a la cantidad de datos y a la complejidad de las relaciones entre estos datos.

> Por ejemplo, en una compañía telefónica gestionan, referido a la cantidad, datos sobre millones de llamadas, de clientes, tarifas, *tickets*...; y, referido a la complejidad, gestionan una gran cantidad de condiciones de tarificación como pueden ser fijo/móvil, horario, planes, roaming, datos, etc.

1.2. Base de datos (BD)

Una BD es una **colección organizada de datos** que **modela aspectos relevantes de la realidad** y **da soporte a procesos de información**.

Un sistema gestor de BD (SGBD) es un **sistema de software que gestiona los datos** (MySQL, Redis, etc).

Las BD se diseñan, y los SGBDs se usan.

1.3. Gestión de datos

La gestión de datos involucra: almacenamiento, extracción, modificación, borrado, búsqueda, seguridad, integridad, compartición, etc.

Puede usarse un **modelo relacional en el que los datos se describen como tablas relacionadas**. Hay más modelos, cada uno con sus ventajas e inconvenientes.

Existe una excesiva orientación hacia el almacenamiento de los datos sin estructurarlos, lo que puede ser contraproducente a largo plazo.

1.4. Almacenamiento vs BD

1. Hardware
   - Visión demasiado hardware.
   - En último término, los datos tienen que ser almacenados así.
   - Persistencia de los datos (¿Qué se hace con los no persistentes?).
2. Físico

- Datos de bajo nivel pero con forma de estructura de datos (lista encadenada).
- Archivo en memoria secundaria.
- La estructura de datos me da una forma de acceso.
- Puedo servirme del sistema de archivos del SO.

3. Lógico
   - Comprensible por agrupación de columnas.
   - Tiene estructura de tabla basada en entidades, relaciones y propiedades
   - La tabla no es una estructura de datos, aunque el fichero secuencial sí que lo es.
   - Problemas para almacenar todo en una sola tabla (Añadir las pruebas que se realizan a cada paciente).
   - Las líneas 1 a 3 son especiales (nombre del fichero, formato, esquema de los datos - Datos pacientes, CSV, nombre apellidos etc)

     Grupo repetitivo: Almacenaje de un número indeterminado de valores del mismo tipo en una sola casilla de la tabla.

4. Conceptual
   - Modelado de los conceptos que describen una organización en base a los datos que gestiona.
   - Visión global de alto nivel, basada en entidades, relaciones y propiedades o en clasificadores, asociaciones y atributos.
   - No se ven las instancias de los datos.
   - Confusión entre modelos/esquemas/diseños conceptuales y lógicos.

1.5. Sistemas de archivos

Los sistemas de archivos plantean varios problemas:

Estructura de almacenamiento: enormes volúmenes de datos. Compartición o replicación. Identificación o indexación de los datos.

Programas de acceso a los datos: Complejidad de las consultas y actualizaciones. Comprobaciones de integridad (en cada uno). Consistencia de accesos concurrentes. Consistencia ante fallos.

Seguridad: Directivas de seguridad para distintos accesos.

1.6. Sistemas de BDs

Las BDs dan respuesta completa y eficaz a los problemas descritos anteriormente.

1. Independencia lógica-física:

- El usuario trata los datos a nivel lógico-conceptual, e internamente se puede cambiar el nivel físico. El usuario vería los datos igual, manteniendo intactas las aplicaciones.
- La traducción la hace automáticamente el SGBD, manteniendo ocultos los detalles.
- Facilita el cambio y mantenimiento haciendo eficiente el acceso a datos.

2. Integridad y seguridad de los datos:
   - El SGBD asegura las restricciones de integridad, en vez de los múltiples programas que acceden a los datos.
   - Se puede controlar los datos que son accesibles a cada usuario.
3. Centralización de los datos:
   - Minimiza redundancia, evitando inconsistencias.
4. Acceso concurrente y recuperación.
5. Reducción del tiempo de desarrollo y mantenimiento de aplicaciones.

1.7 Niveles de abstracción

1. Esquema externo: Vista (tabla) que combina datos para una presentación específica de usuario. Se define sobre el conceptual.
2. Esquema conceptual: Incluye todas las tablas / relaciones con información sobre entidades y relaciones.
3. Esquema interno: Esquema físico con los detalles de almacenamiento.

1.8 Lenguajes de consulta

- Definición de esquemas (DDL), manipulación de datos (DML), control.
- Formalización a través de cálculo / álgebra relacional.
- Optimización de la eficiencia.
- Estándar SQL.
- Lenguaje anfitrión.

1.9 Arquitectura de un SGBD

Compuesto de bloques que desarrollan las funciones encargadas al SGBD:

- Interacción con el usuario / aplicación y el almacenamiento secundario.
- Secuencia operativa.
- Interdependencia.
- Configuración para mejorar eficiencia.
- Trabajo del administrador de la BD.

2. Modelado de datos: Introdcción al diseño de bases de datos (PDF)

Etapas del diseño

¿Qué pasos se siguen en el diseño de bases de datos? Existen una serie de diseños que se realizan en un cierto orden.

1. Análisis de requisitos: Comprender los datos a gestionar, elicitar las necesidades del cliente a través de reuniones, discusiones, documentación... Es una etapa clave que puede ser muy costosa.
2. Diseño conceptual: Descripción de alto nivel de los datos y sus restricciones. Modelo que representa, organiza y clarifica la información. Normalmente, modelo Entidad-Relación. Es suficientemente preciso como para permitir su traducción a un modelo específico del SGBD.
3. Diseño lógico: Esquema de la BD acorde al SGBD elegido. Traducción del esquema ER a un esquema relacional.
4. Refinamiento de los esquemas: Reestructuración para garantizar propiedades importantes (normalización).
5. Diseño físico: Mejora de rendimiento en base a cargas típicas. Idealmente no supone un rediseño de las etapas anteriores.
6. Diseño de aplicaciones y seguridad: Procesos relacionados con las aplicaciones, como tareas y flujos de trabajo, y cuestiones de accesibilidad y seguridad.

Diseño lógico o conceptual primero

Hay que preguntarse cuestiones como, en el caso del sistema de gestión telefónica, si dos clientes pueden tener la misma línea, si una línea puede tener dos tarifas, si son necesarias más tablas y cuáles, etc.

El diseño conceptual se realiza como diagrama ER (Entidad-Relación) con notación UML. Describe cómo se estructuran los datos. Existen varias alternativas para un mismo escenario, y hay que decidir.

Las **entidades** son objetos que engloban los datos de interés. Se describen como colecciones de entidades similares, mediante atributos y propiedades adicionales. Las instancias se ven como elementos de un conjunto.

Los **atributos** y **claves** definen los datos para cada instancia. El **dominio** (opcional) es el conjunto posible de valores o la posibilidad de definición de tipos. La **clave** es el conjunto mínimo de atributos que identifica a cada entidad (valor mínimo sin repetición).

Las **relaciones** representan asociaciones entre dos o más entidades, en función de su grado. También se puede ver como un conjunto de relaciones con instancias

de dichas relaciones dependiendo de si hay que modelar también la relación con unas propiedades.

La **cardinalidad** es una multiplicidad simplificada a tres posibles valores: `1:1`, `1:*`, `*:*`. Se puede representar la opcionalidad poniendo el mínimo a cero.

Una relación puede ser **recursiva** si una instancia de una entidad se relaciona con otra instancia de la misma entidad.

Una entidad es **débil, dependiente o subordinada** si se identifica considerando la primary key de otra entidad denominada **propietaria o dominante**. Expresa una relación obligatoria y con dependencia en existencia.

Trampas de conexión

Relaciones que provocan ambigüedades u ocultan información.

En el caso de *fan traps*, no podemos saber a qué sección pertenece el empleado o cuáles son los empleados de una sección.

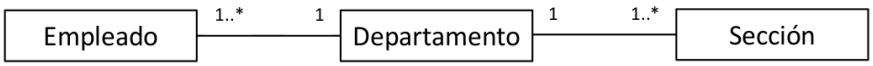

Figure 1: Fan traps

En el caso de *chasm traps*, una relación opcional hace que perdamos la referencia a alguna entidad. En este caso, puede haber cuentas que no estén asociadas a ningún empleado y por tanto que no tengan sucursal.

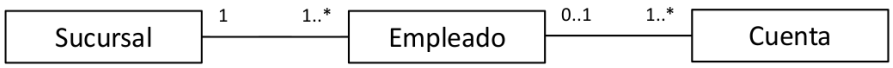

Figure 2: Chasm traps

Generalización y especialización (relación *is-a*)

Se pueden hacer entidades que tengan unas propiedades específicas más las comunes generales heredadas de otra entidad.

Agregación

Representa la composición o agregación de partes en un todo. Funciona como una asociación con un significado adicional. Si la creación o destrucción coinciden, es una composición. La agregación se representa con un rombo vacío en el 'todo', y la composición con un rombo relleno.

Elementos del modelado ER

Hay que decidir si modelar un concepto como entidad, como atributo o como relación; identificar si una relación es binaria o ternaria; y ver cuándo hay que usar la agregación.

Para incluir restricciones en el modelo ER, los datos están acompañados de condiciones de validez. Algunas de estas restricciones no se pueden capturar en diagramas ER, así que se incluyen en notas usando lenguajes de restricciones.

3. Modelo relacional

Origen e importancia

Se plantea como un sistema sencillo y potente formado por una BD como conjunto de tablas con filas y columnas, con las tablas asociadas entre sí.

El modelo relacional se origina en los 70, sustituyendo a otros modelos como el jerárquico o en red. Es el modelo más ampliamente utilizado por la sencilla representación de los datos y la facilidad para formular consultas. Hay todo un ecosistema de RDBMS de diversas empresas.

Hay otros modelos competidores como el modelo objeto-relacional o la renovación de "viejas ideas" como los datos jerárquicos.

Un lenguaje relacional permite la definición (DDL) y la manipulación (DML) de los datos. El modelo soporta consultas sencillas y potentes con una **semántica precisa de consulta relacional**. El DBMS optimiza las operaciones.

El estándar SQL ha ido evolucionando. Son reseñables SQL-92, que supuso una importante revisión, y SQL-99, que añade extensiones importantes y es el estándar actual.

Definiciones

Una base de datos relacional está compuesta de relaciones.

Una relación está compuesta de dos partes. Por un lado, un esquema que define el nombre de la relación y el nombre y tipo de cada columna (atributo, campo). Por otro lado, instancias que corresponden con valores de la tabla (registro, tupla) con todas las tuplas distintas.

Creación de relaciones

Hay que crear la relación (tabla) con sus atributos, especificando los tipos para cada campo. El tipado se impone por el DBMS al añadir o modificar tuplas.

Es habitual la existencia de diversas tablas que estarán eventualmente relacionadas.

Ejemplo de creación de tablas:

```
CREATE TABLE Estudiante (
    nia INTEGER,
    nombre CHAR(20),
    login CHAR(10),
    edad INTEGER,
    notam REAL
)

CREATE TABLE Matricula (
    nia INTEGER,
    cod CHAR(20),
    nota REAL
)
```

Destrucción y modificación de tablas

La destrucción de una relación implica su borrado del esquema y el borrado de todas su tuplas.

```
DROP TABLE Estudiante
```

La modificación de un esquema se puede hacer añadiendo, borrando o modificando campos. Para campos añadidos, todas las tuplas son extendidas con valor `null` en ese campo.

```
ALTER TABLE Estudiante
    ADD COLUMN matr DATE
```

Añadir, borrar y modificar tuplas

Se puede insertar una tupla en la relación:

```
INSERT INTO Estudiante (nia, nombre, login, edad, notam)
    VALUES (5557, 'Rojo', 'rojo@cie', 23, 8.0)
```

El borrado se realiza indicando la condición que cumplirán todas las tuplas a ser borradas:

```
DELETE FROM Estudiante E WHERE E.nombre='Ledesma'
```

La modificación se realiza igualmente con una condición de selección de tuplas:

```
UPDATE Estudiante E SET E.edad=E.edad+1 WHERE E.nia=5557
```

Restricciones de integridad

Las restricciones de integridad (RI) son condiciones de validez que tienen que cumplirse para cualquier instancia de la base de datos. Se especifican al definir el esquema y el DBMS las comprueba cuando se modifican las relaciones.

Una **instancia legal** es aquella que satisface todas las restricciones de integridad.

En las RI se usa la semántica del mundo real descrito. Hay restricciones de dominio (tipos), clave primaria, clave foránea... También se soportan otras más generales.

Claves primarias y candidatas

Una clave (candidata) identifica unívocamente cada tupla por medio de un subconjunto mínimo de campos. Hay que elegir entre una de las claves candidatas la que será la clave primaria, y el resto son claves alternativas.

Las claves primarias fuerzan ciertas condiciones. No existen dos tuplas con los mismos valores en todos los campos de la clave (a tener en cuenta para valores `null`), y ningún subconjunto de la clave puede ser un identificador único.

Las claves candidatas se especifican con `UNIQUE`, y la que es elegida como primaria, con `PRIMARY KEY`.

```
CREATE TABLE Matricula (
    nia INTEGER,
    cod CHAR(20),
    nota    REAL,
    PRIMARY KEY (nia, cod)
)

CREATE TABLE Estudiante (
    nia INTEGER,
    nombre  CHAR(20),
    login   CHAR(10),
    edad    INTEGER,
    notam   REAL,
    UNIQUE (nombre, edad),
    CONSTRAINT claveEst PRIMARY KEY (nia)
)
```

En el ejemplo anterior se le da nombre a la restricción para identificar errores más fácilmente.

Claves foráneas, integridad referencial

Una clave foránea (o externa) es un conjunto de campos en una relación que sirven para referenciar tuplas en otra relación. La referenciada debe ser una clave primaria para asegurar que solo se hace referencia a una única tupla.

La **integridad referencial** se refiere a la imposición de las restriciones referenciales. Por ejemplo: solo los estudiantes listados en Estudiante son admitidos en la Matrícula de cursos.

```
CREATE TABLE Matricula (
    nia INTEGER,
    cod CHAR(20),
    nota    REAL,
    PRIMARY KEY (nia, cod),
    FOREIGN KEY (nia) REFERENCES Estudiante
)
```

Una clave foránea puede hacer referencia a la misma relación en la que se encuentra. El `null` cumple la restricción de clave foránea pero no la de clave primaria (no puede ser clave primaria, pero sí puede servir para indicar una relación de clave foránea que no se ha dado aún).

Cumplimiento de la integridad referencial

Para cumplir la integridad referencial, hay que decidir qué hacer en cada caso cuando ésta pueda incumplirse.

En inserción con una referencia no existente, se realiza `NO ACTION`, es decir, se rechaza la inserción de la nueva tupla si tiene una referencia en una clave foránea que referencia a una tupla que no existe.

En el borrado de una tupla referenciada hay varias opciones:

- `NO ACTION`: Acción por defecto. Rechaza el borrado de la tupla referenciada si tiene referencias. (No se borra el estudiante hasta que no tenga matrículas que le hagan referencia)
- `CASCADE`: Se borra la tupla referenciada y todas aquellas que le referencian (se borra el estudiante y todas sus matrículas).
- `SET DEFAULT`: Se asignan las referencias a la tupla que se borra a una clave por defecto (al borrar el estudiante, sus matrículas pasan a referenciar un `nia` por defecto).

- `SET NULL` (en SQL): Al borrar la tupla referenciada, las referencias apuntan a `null`. Puede dar conflictos si la referencia (la clave foránea) se usa dentro de la clave primaria.

En la actualización de la clave primaria se actúa igual que en el borrado de una tupla referenciada.

```
CREATE TABLE Matricula (
    nia INTEGER,
    cod CHAR(20),
    nota    REAL,
    PRIMARY KEY (nia, cod),
    FOREIGN KEY (nia) REFERENCES Estudiante
        ON DELETE CASCADE
        ON UPDATE SET DEFAULT
)
```

La acción 'cascade' se propaga en cascada por las siguientes realciones que referencian a la tupla borrada o actualizada.

Otras restricciones

Restricciones de columna, que pueden ser `NOT NULL`, que evita que se asignen nulos a la columna, o `CHECK`, que especifica una condición de integridad.

```
CREATE TABLE Estudiante (
    nia INTEGER,
    nombre  CHAR(20),
    login   CHAR(10),
    edad    INTEGER,
    notam   REAL NOT NULL,
    UNIQUE (nombre, edad),
    CONSTRAINT claveEst PRIMARY KEY (nia),
    CONSTRAINT notampos CHECK (notam>=0)
)
```

Otras restricciones son las aserciones y disipadores, que se verán más adelante.

Diseño lógico: ER -> Relacional

ERD (Diagrama Entidad-Relación)

ER (Entidad-Relación)

SQL

Tipos de relación -> tablas

Si la relación es `1-1`, comprobamos si se pueden poner en términos de una única entidad.

Si la relación es `1-*`, referenciamos en el lado `*` el valor único del lado `1`.

Si la relación es `*-*`, habría que representar la relación como una nueva entidad que se relaciona con los dos extremos, siendo la nueva entidad el lado `1` en las dos nuevas relaciones creadas.

Traducción de la opcionalidad

Si la relación tiene `0` en algún extremo supone que pueden existir nulos.

Entidades débiles

Se da cuando una entidad se identifica con campos de otra entidad, y dicha relación es obligatoria. En otras palabras, se da cuando hay una clave foránea no nula que forma parte de la clave primaria.

Jerarquías ISA

Es una guía de elección, pero puede haber otros factores (cantidad de nulos, relaciones, consultas / combinación de tablas).

Tablas para las sublclases: diferencias entre ellas; todas en una tabla daría lugar a nulos y restricciones. Diferencia con la superclase; puede haber tablas para cada combinación superclase/subclase.

Tabla para la superclase: Si tiene asociaciones `1:*`; las tablas que incluyen las claves foráneas deben apuntar a una única tabla.

Vistas y seguridad

Una vista es una tabla que no almacena tuplas ya que estas se obtienen a partir de una definición sobre tablas base. Permite restructurar el esquema lógico base e implementar políticas de acceso restringido a los datos.

Por ejemplo, la siguiente vista evitaría que se vean más datos de los necesarios (nia y nota media) de los estudiantes, y se restringe a que solo se vean los alumnos que cumplan la condición.

```
CREATE VIEW EstudianteMH (nia,notam)
    AS SELECT E.nia,E.notam
        FROM Estudiantes E
        WHERE E.notam >= 9.0
```

Diseño de bases de datos: álgebra y cálculo relacional (PDF)

Lenguajes de consulta (QL)

Los lenguajes de consulta permiten la manipulación y consulta de los datos de la BD.

Diseño de bases de datos: SQL. Consultas y restricciones. (PDF)

Diseño de Bases de datos: Introducción a la administración de BD (PDF)

Tipos de usuarios

Hay distintos tipos de usuarios de una base de datos con distintas responsabilidades y competencias.

Un **usuario de BD** interactúa con la BD a través de aplicaciones.

Un **desarrollador de aplicaciones** diseña y desarrolla la aplicación incluyendo la estructura de la base de datos. Estima requisitos de almacenamiento, eficiencia, seguridad, etc.

El **administrador de la BD** (DBA) instala y actualiza el DBMS y las herramientas asociadas. Establece y reserva el sistema de almacenamiento. Crea los objetos primarios (tablas, vistas, índices) una vez que los desarrolladores han realizado el diseño.

Tareas típicas

Metadatos

Concurrencia y transacciones

Recuperación

Optimización

La optimización es vital porque define el rendimiento de las aplicaciones, y sabemos que este valor es crucial para la experiencia del usuario.

El SGBD puede realizar algunas optimizaciones que pueden reducir mucho el coste. Por ejemplo, normalmente las queries SQL se optimizan antes de ser ejecutadas, y si se necesita un rendimiento mayor para algunas consultas habituales se pueden generar índices.

Diseño físico

Se puede decidir en qué tipo de estructura de datos queremos que se almacene cada tabla (normalmente, en MySQL no es tan flexible como en Oracle, por ejemplo).

Secuenciales

Es una estructura de n registros almacenados uno detrás de otro en bloques con b registros.

Operación	Coste
Búsqueda	Todo el fichero: $O(n/b)$
Inserción	En el último bloque si hay espacio
Borrado	Deja un espacio libre en el bloque y hay que recolectar basura
Modificación	¿?

Indexados

Para solucionar los problemas de la estructura secuencial, creo un índice que apunta a la estructura secuencial para ayudar con la búsqueda.

En la búsqueda no densa, el índice apunta al primer registro de cada bloque. Si hay muchos cambios podría ser que el primer registro del bloque cambie y haya que reindexar.

Operación	Coste
Búsqueda	Binaria ($O(log(n))$) + localización en el bloque.

Árboles B

Es un árbol dirigido balanceado. La mejora respecto al índice es que la búsqueda no es binaria sino $O(log(k))$, y se puede conseguir un k más alto haciendo niveles anchos. Es una estructura muy usada, y suele ser la usada por defecto en los SGBD.

Dispersión

Se calcula la posición del registro a partir de la clave. Como suele haber más datos que espacio de memoria, se tienen que permitir colisiones. Si se produce una colisión, o se enlaza en ese campo otro campo a mayores, o se guarda en otro sitio.

Para evitar colisiones hay que mantener la estructura poco llena, ocupando mucho espacio. Aparte de estos problemas, la dispersión exige que se proporcione la clave completa y no solo una fracción, como sí permiten las otras estructuras. Eso permite cosas como búsquedas por rango, etc.

Índices secundarios