Diseño de Bases de Datos

Prof. Pablo Thomas

Rodolfo Bertone

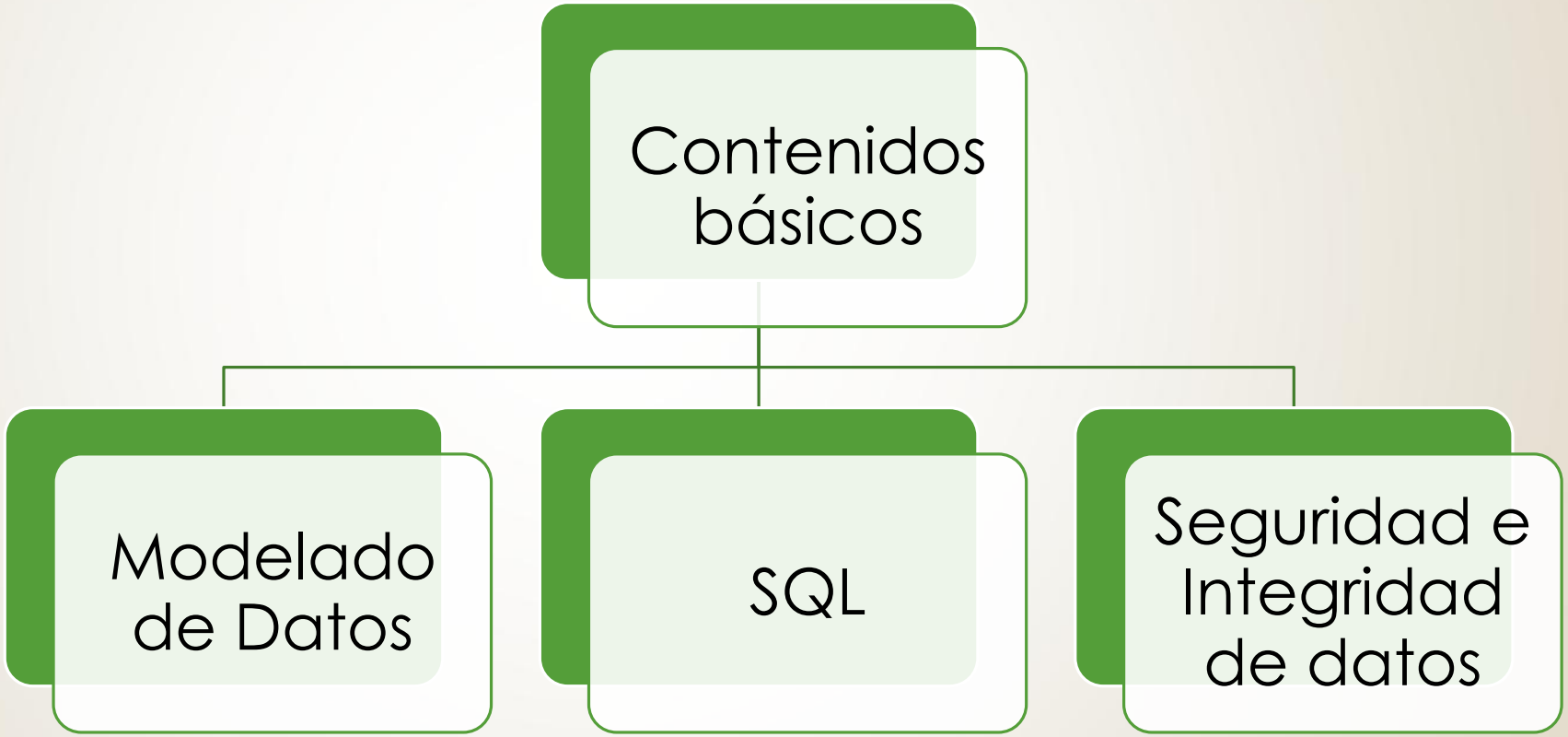
La cátedra

- Clases
  - Teóricas
  - Explicaciones de Prácticas (donde se presentan ejemplos)
  - Prácticas
- Para aprobar la cursada
  - Un Parcial
  - Dos recuperatorios

Propuesta de examen teórico

- Podrán acceder al examen teórico todos los alumnos que se encuentren inscriptos en la asignatura.
- La aprobación es con nota 5 o superior
- Se deberá aprobar la cursada durante el semestre en que rinde el examen teórico.
- **Deberá anotarse para pasar el final de la asignatura como máximo para la fecha de agosto del año siguiente que se cursó la asignatura.**

La Materia


Contenidos básicos
Modelado de Datos
SQL
Seguridad e Integridad de datos

Semana	Teoría	Práctica
15/08	Base de datos. Modelado. Modelo conceptual	Sin Actividad
22/08	Modelo Conceptual	Modelo Conceptual
29/08	Modelo lógico	Modelo Conceptual
05/09	Modelo físico	Modelo Conceptual
12/09	Ejemplificación	Modelo Lógico
19/09	Lenguaje de Consultas. Algebra Relacional (AR)	Modelo físico
26/09	AR	Modelo
03/10	SQL	AR
10/10	SQL	AR
17/10	Seguridad e Integridad Transacciones	SQL
24/10	Transacciones concurrentes	SQL
30/10	Consulta	Consulta
07/11	Consulta	Primer Parical
14/11	Consulta	Consulta
21/11	Consulta	Muestra de examenes
28/11	Consulta	Recuperatorio
12/12	Consulta	Consulta
12/12	Examen Teórico	Muestra de examenes
19/12		Recuperatorio

Bibliografia

- Introducción a las Bases de Datos. Conceptos Básicos (Bertone, Thomas)
  - Fundamentos de Bases de Datos (Korth Silvershatz)
  - Introducción a los sistemas de Bases de Datos. Date. Addison Wesley.
  - Diseño Conceptual de Bases de Datos: un enfoque entidad interrelaciones. Batini, Navatte, Cieri. Addison Wesley.
  - Fundamento de sistemas de Bases de Datos. Elmasri, Navate. Addison Wesley..

Diseño de Bases de Datos

Clase 1

Agenda

Conceptos básicos de BD

- Definiciones
- Características

Modelado

- Introducción
- Entidad Relación

Conceptos básicos

Qué es una Base de Datos?

Es una colección de datos relacionados.

Colección de **archivos** diseñados para servir a múltiples aplicaciones

Colección o conjunto de datos interrelacionados con un propósito específico vinculado a la resolución de un problema del mundo real

Un dato representa hechos conocidos que pueden registrarse y que tienen un resultado implícito.

Conceptos básicos

Propiedades implícitas de una BD:

Una BD representa algunos aspectos del mundo real, a veces denominado Universo de Discurso.

Una BD es una colección coherente de datos con significados inherentes. Un conjunto aleatorio de datos no puede considerarse una BD. O sea los datos deben tener cierta lógica.

Una BD se diseña, construye y completa de datos para un propósito específico. Está destinada a un grupo de usuarios concretos y tiene algunas aplicaciones preconcebidas en las cuales están interesados los usuarios

Una BD está sustentada físicamente en archivos en dispositivos de almacenamiento persistente de datos

Conceptos Básicos (resumiendo)

La **definición de una BD** consiste en especificar los tipos de datos, las estructuras y restricciones de los mismos.

La **construcción de la BD** es el proceso de almacenar datos concretos en algún dispositivo de almacenamiento bajo la gestión del **DBMS**.

La **manipulación de BD** incluye funciones tales como consultar la BD para recuperar datos específicos, actualizar los datos existentes, reflejar cambios producidos, etc

Conceptos Básicos

Que es un DBMS o SGBD?

Las siglas → Data Base Management System o Sistema Gerenciador de Bases de Datos

Es una colección de programas que permiten a los usuarios crear y mantener la BD

Es un sistema de software de propósito general que facilita los procesos de definición, construcción y manipulación de BD

Conceptos Básicos

Objetivos de un DBMS:

- Evitar redundancia e inconsistencia de datos
- Permitir acceso a los datos en todo momento
- Evitar anomalías en el acceso concurrente
- Restricción a accesos no autorizados → seguridad.
- Suministro de almacenamiento persistente de datos (aún ante fallos)
- Integridad en los datos
- Backups.

Conceptos Básicos

Componentes de un DBMS

DDL *(data definition languaje)*: especifica el esquema de BD.
Resultado:
Diccionario de datos

DML *(data manipulation languaje)*:

- Recuperación de información
- Agregar información
- Quitar información
- Modificar información

Conceptos Básicos

DML → Características:

- **Procedimentales (SQL)** → requieren que el usuario especifique **qué** datos se muestran y **cómo** obtener esos datos

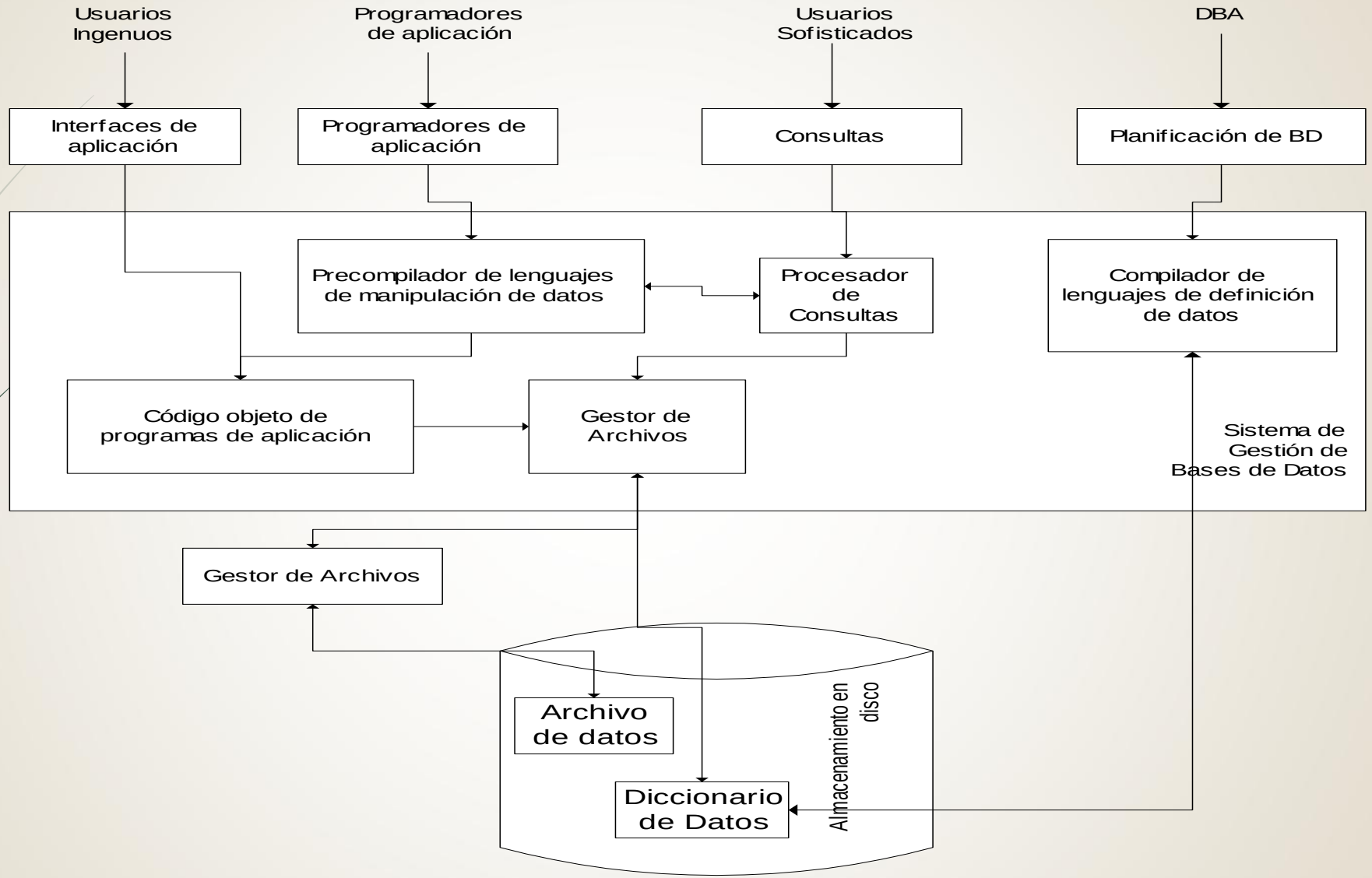
- **No Procedimentales (QBE)** → requieren que el usuario especifique **qué** datos se muestran y **sin especificar cómo** obtener esos datos

Conceptos Básicos

Actores involucrados con una BD

- **DBA o ADB**
  - Administra el recurso, que es la BD. Autoriza accesos, coordina y vigila la utilización de recursos de hardware y software, responsable ante problemas de violación de seguridad o respuesta lenta del sistema.
- **Diseñador de BD**
  - Definen la estructura de la BD de acuerdo al problema del mundo real que esté representando
- **Analistas de Sistemas**
  - Determinan los requerimientos de los usuarios finales, generando la información necesaria para el diseñador.
- **Programadores**
  - Implementan las especificaciones de los analistas utilizando la BD generada por el diseñador.
- **Usuarios (distintos tipos)**

Conceptos Básicos



Conceptos Básicos

Propósitos mas relevantes	**Aprender a diseñar una BD**	Construcción del modelo de datos → **Diseño**
		Normalización
	Aprender a manipular una BD	Lenguaje de trabajo clásico con BD
	Estudio de seguridad e integridad de la información	

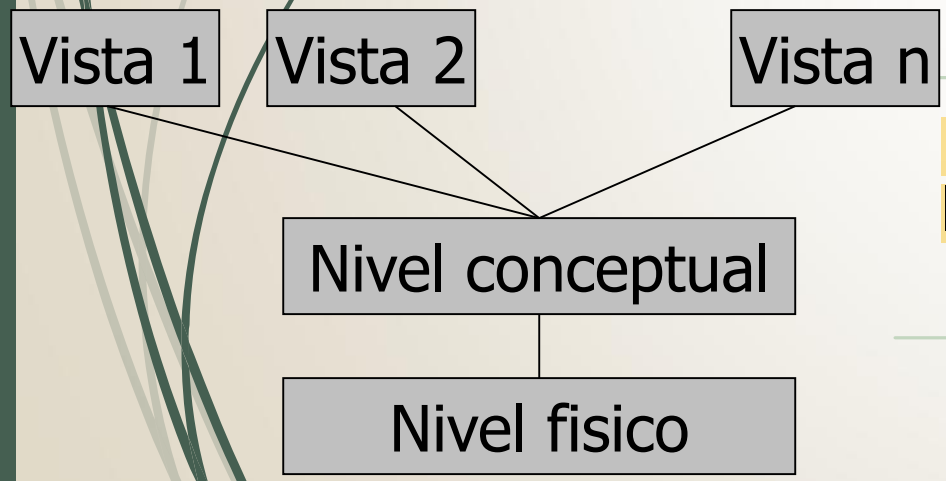
Modelado

Abstracciones:

Visión: ve solo los datos de interes (muchas vistas para la misma BD) a través de prg de aplicación.

Conceptual: ***qué*** datos se almacenan en la BD y ***qué*** relaciones existen entre ellos

Físico: describe ***cómo*** se almacenan realmente los datos (archivos y hardware)



Modelado

Colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia

Modelado

Modelos

Basado en objetos (visión, conceptual). Estructura flexible, especifican restricciones explícitamente

- Modelo de Entidad-Relación
- Modelo Orientado a Objetos

Basado en registros (conceptual, físico). La BD se estructura en reg. de formato fijo. Se dispone de lenguaje asociado para expresar consultas

- OO
- relacional
- jerárquico
- red

Físico de datos

Modelado

Independencia de datos

- Capacidad de modificar esquemas sin alterar otro nivel
  - **Físico** (modificar el esquema físico sin provocar que los programadores tengan que reescribir los prg de aplicación-> gralm. para mejorar el funcionamiento)
  - **Lógico** (modificar el esquema conceptual)

Modelado

Categorías de soft de procesamiento de datos:

- Sin independencia de datos (SO, transferencia a un sector en particular)
- Independencia física (leer un registro de un archivo, SO)
- Independencia lógica parcial (leer siguiente registro de un archivo)
- Independencia lógica y física (leer siguiente registro de un tipo particular, DBMS)
- Independencia geográfica (BD distribuidas)

Diseño

Diseño de Base de datos: tres etapas

- **Conceptual** (representación abstracta)
- **Lógico** (representación en una computadora)
- **Físico** (determinar estructuras de almacenamiento físico)

Modelado

Un modelo/esquema de datos sirve para hacer más fácil la compresión de los datos de una organización

- Se modela para
  - Obtener la perspectiva de cada actor asociado al problema
  - Obtener la naturaleza y necesidad de cada dato
  - Observar como cada actor utiliza cada dato

Modelado

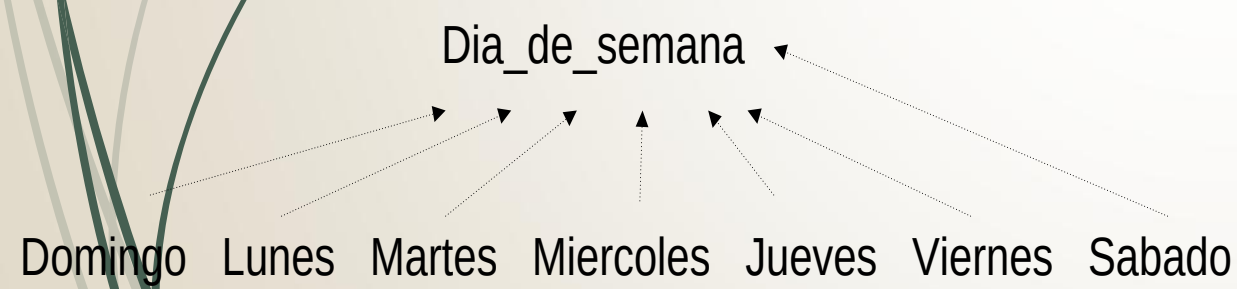
Abstracciones

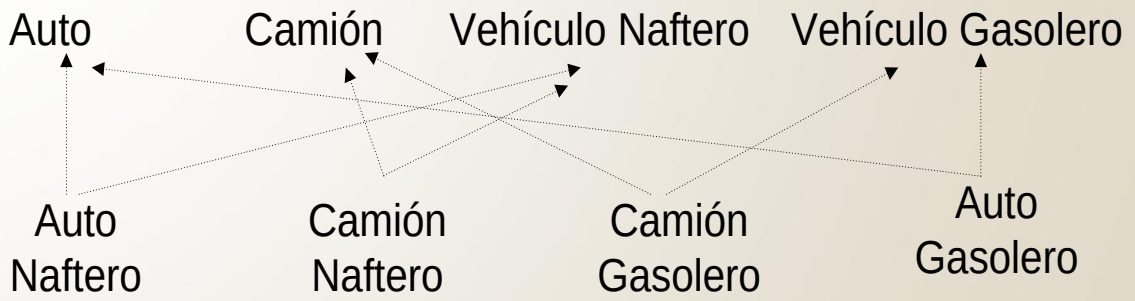
- **Abstracción:** Proceso que permite seleccionar algunas características de un conjunto de objetos del mundo real, dejando de lado rasgos que no son de interés
- Tres abstracciones:
  - Clasificación
  - Agregación
  - Generalización

Modelado

Abstracción de Clasificación

- Define una clase
- Características comunes que tiene los objetos que la componen

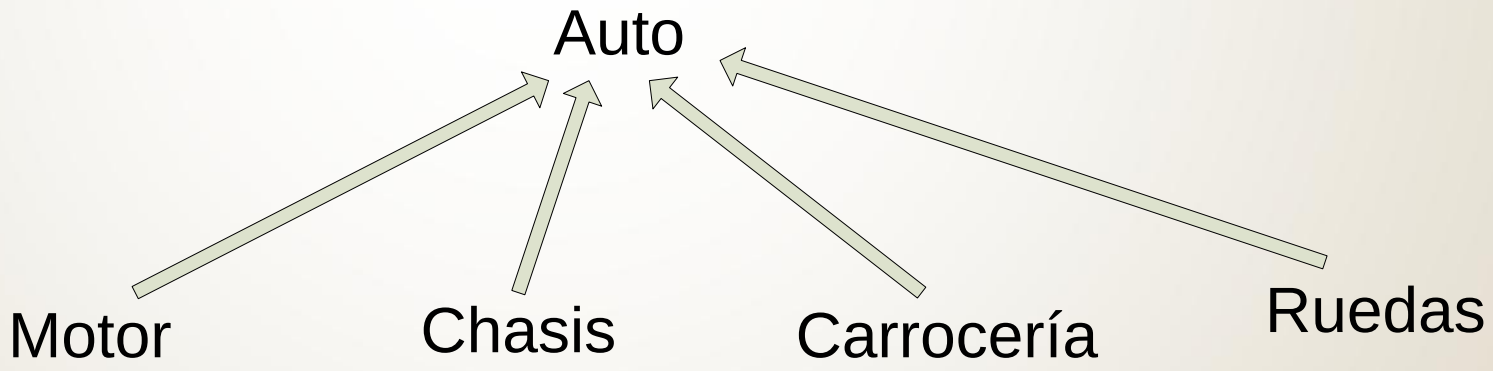




Modelado

Abstracción de agregación

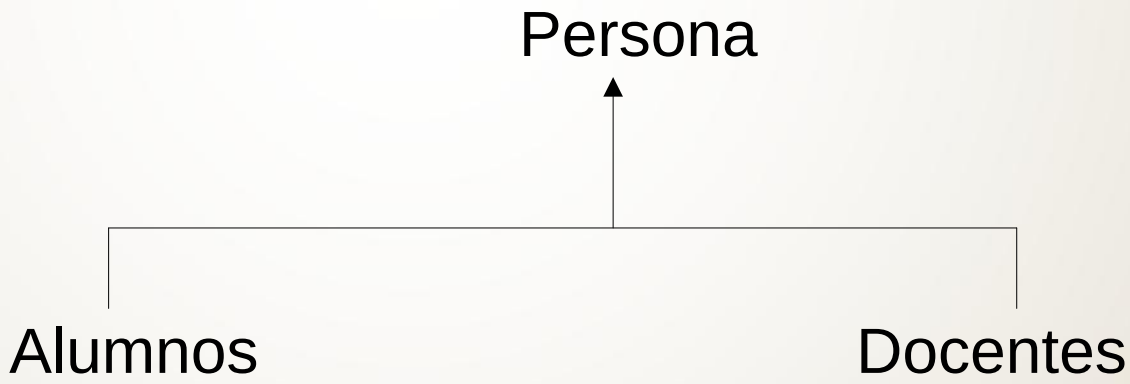
- Define una nueva clase a partir de otras clases que representan sus partes componentes



Modelado

Abstracción de Generalización

- Define una relación de subconjunto entre los elementos de dos o mas clases



Modelado

Propiedades de las abstracciones

- Agregación
  - Cardinalidad
- Generalización
  - Cobertura

Modelo ER

Características del Modelo ER

- Estándar internacional desde 1988
- Propuesto por Chen en 1976
- Ampliado por Codd en 1979
- Se basa en la concepción del mundo real como un conjunto de objetos llamadas entidades y las relaciones que existen entre ellas
- Permite modelar el nivel conceptual y lógico de una BD

Modelo Conceptual ER

Objetivos

- Representar la información de un problema en un alto nivel de abstracción
- Captar la necesidad de un cliente respecto del problema que enfrenta
- Mejora la interacción cliente / desarrollador disminuyendo la brecha entre la realidad del problema y el sistema a desarrollar

Modelo Conceptual ER

Características

Expresividad: disponer de todos los medios necesarios para describir un problema

Formalidad: cada elemento representado sea preciso y bien definido, con una sola interpretación posible

Minimalidad: cada elemento tiene una única representación posible

Simplicidad: el modelo debe ser fácil de entender por el cliente y por el desarrollador

Modelo Conceptual ER → Componentes

Entidades

Relaciones

Atributos

Modelo Conceptual ER

Entidades

- Representa un elemento u objeto del mundo real con identidad
- Se diferencia de cualquier otro objeto o cosa
- Ejemplos

Conjunto de entidades

- Representación que, a partir de las características propias de cada entidad con propiedades comunes, se resume en un núcleo

Alumnos

Vehículos

Materias

Modelo Conceptual ER

Relaciones

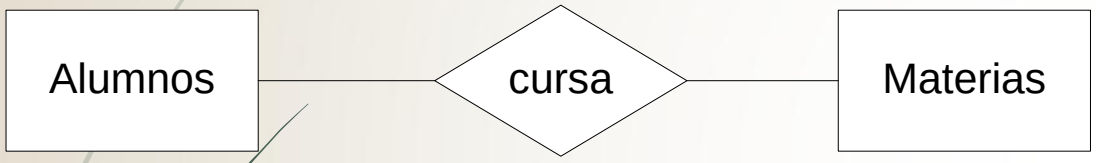
- Representan agregaciones entre dos (binaria) o mas entidades
- Ej: el alumno Perez cursa la materia Matemática I

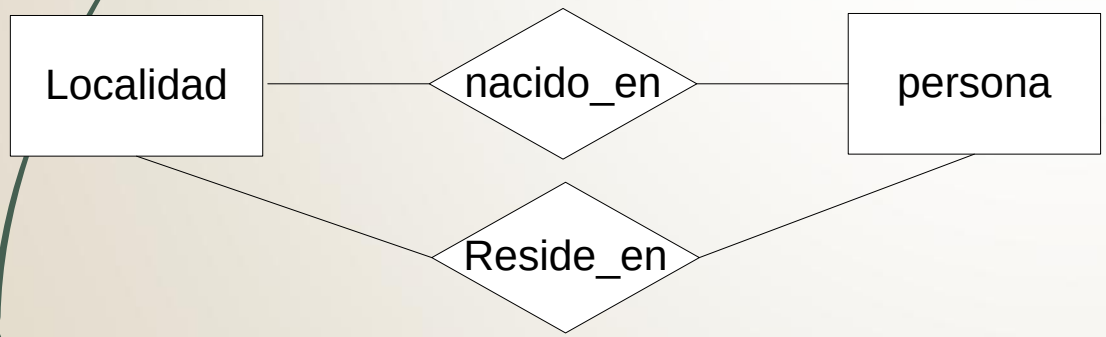
Conjunto de Relaciones

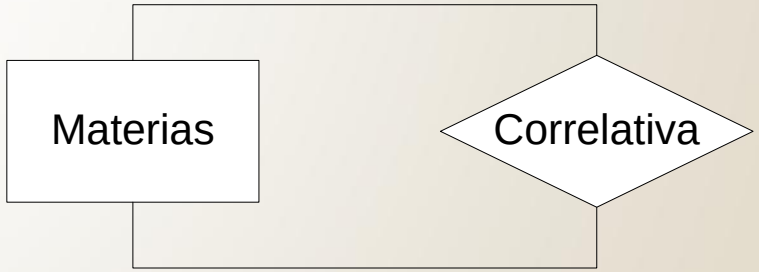
- Es una representación que, a partir de las características propias de cada relación existente entre dos entidades, las resume en un núcleo

Modelo Conceptual ER

- Ejemplos





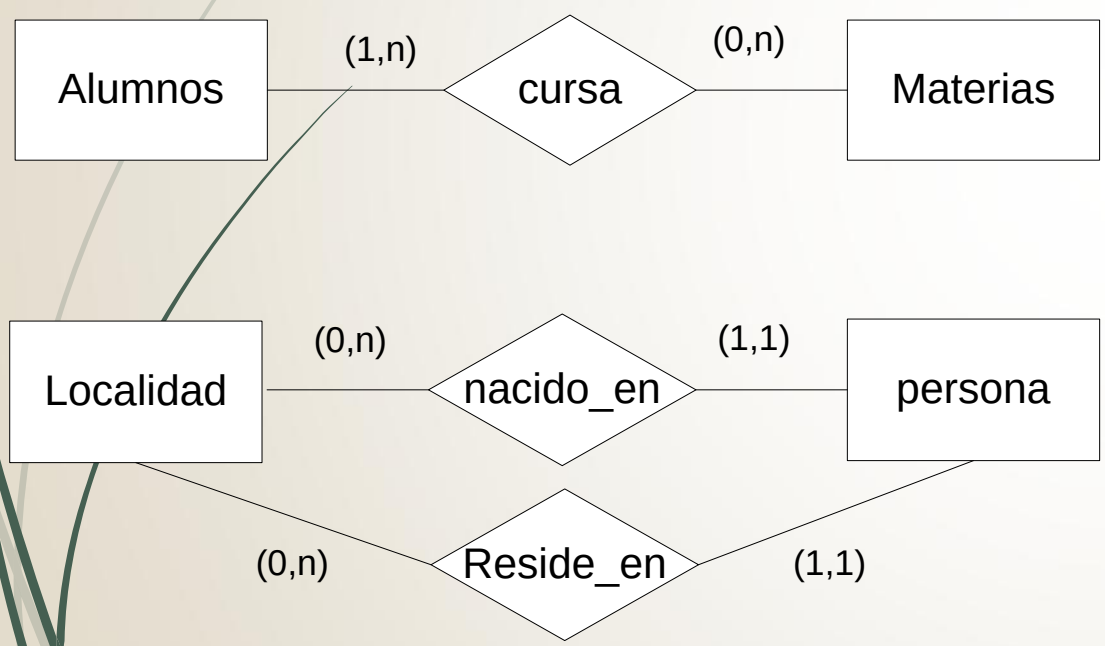


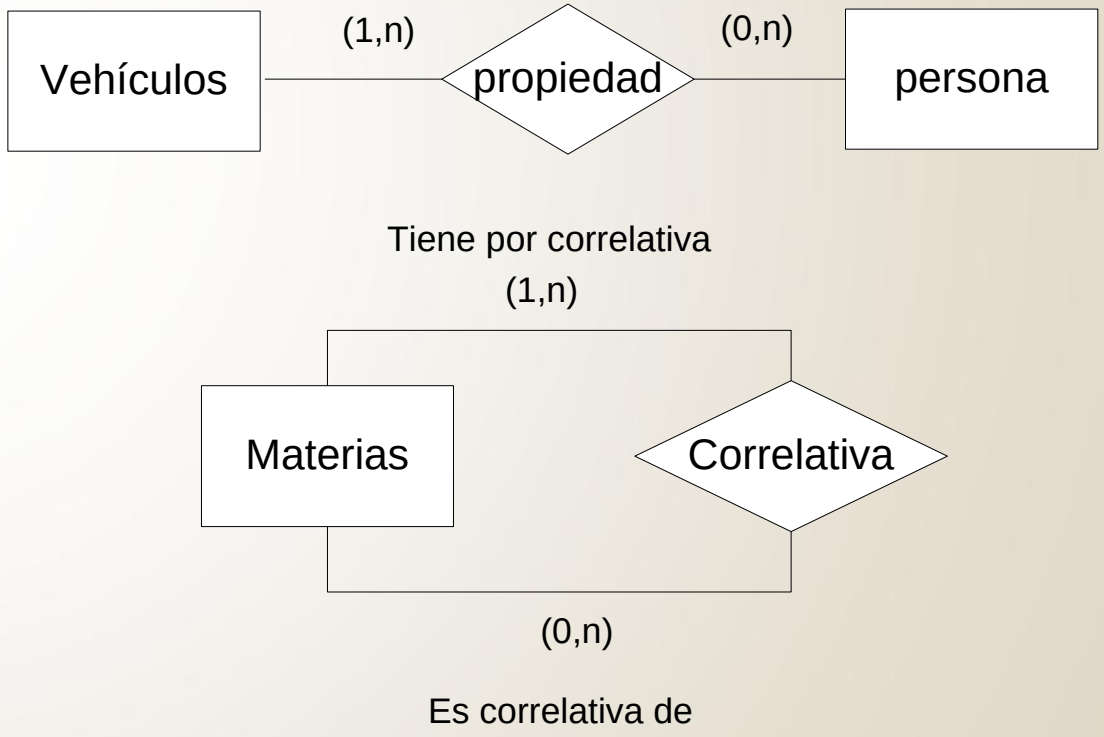
Modelo Conceptual ER

Tipos de relación	Binaria
	Ternaria
	N-aria
	Recursiva
Cardinalidad de la relación	Define el grado de relación existente en una agregación
	Cardinalidad Máxima
	Cardinalidad Mínima

Modelo Conceptual ER

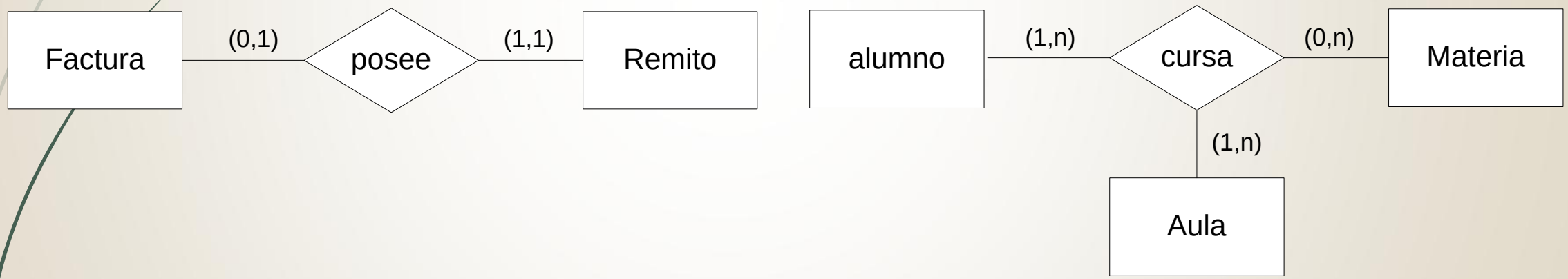
- Ejemplos





Modelo Conceptual ER

- Otros ejemplos



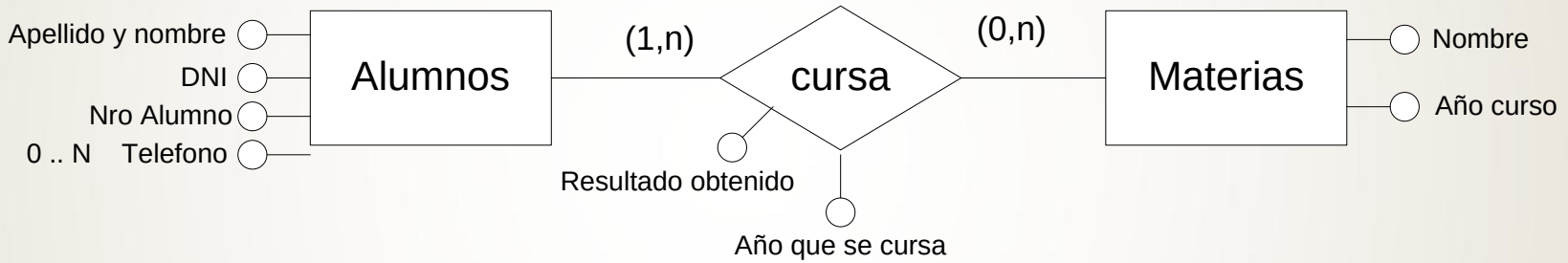
Modelo Conceptual ER

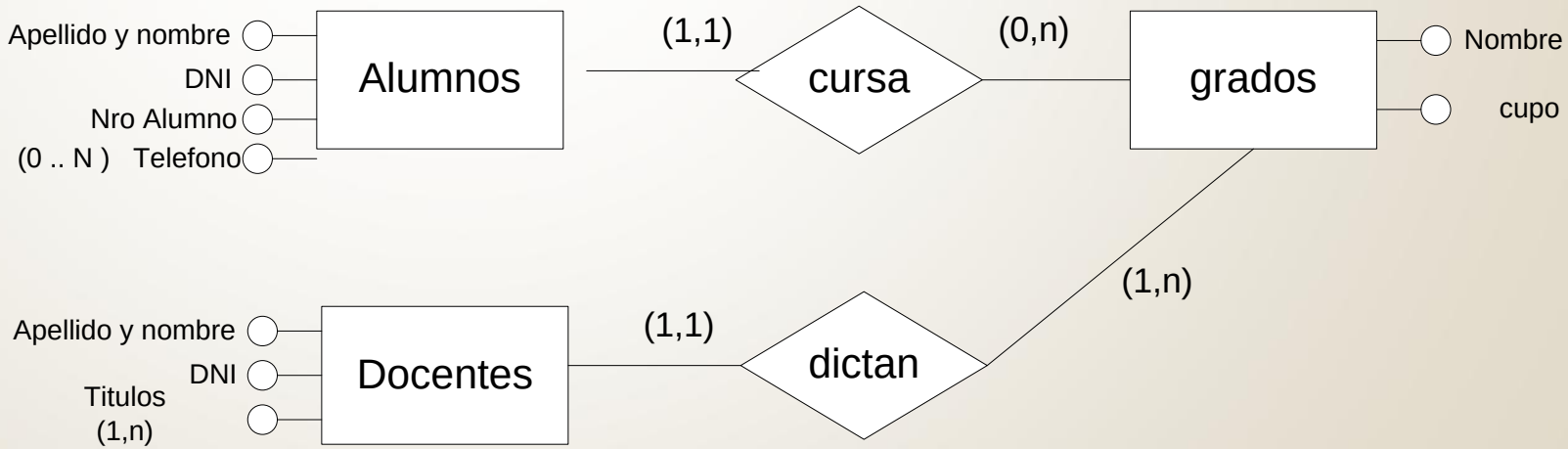
Atributos

- Representa una propiedad básica de una entidad o relación
- Equivale a un campo de un registro
- Cardinalidad
  - Monovalente/polivalente
  - Obligatorio/opcional (nulo)

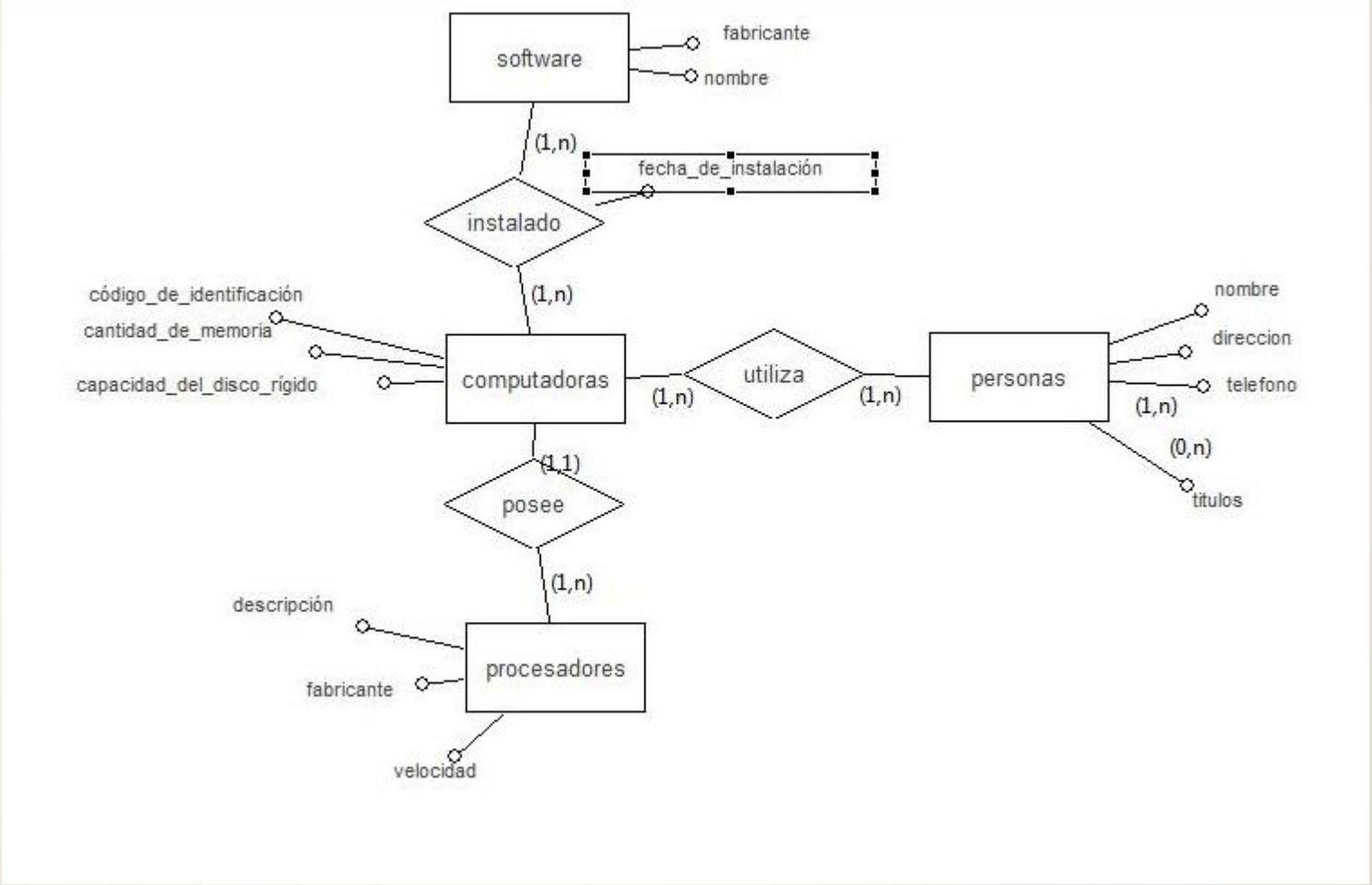
Modelo Conceptual ER

- Ejemplos





Modelo Conceptual ER



Modelo Conceptual ER

Componentes adicionales de modelado

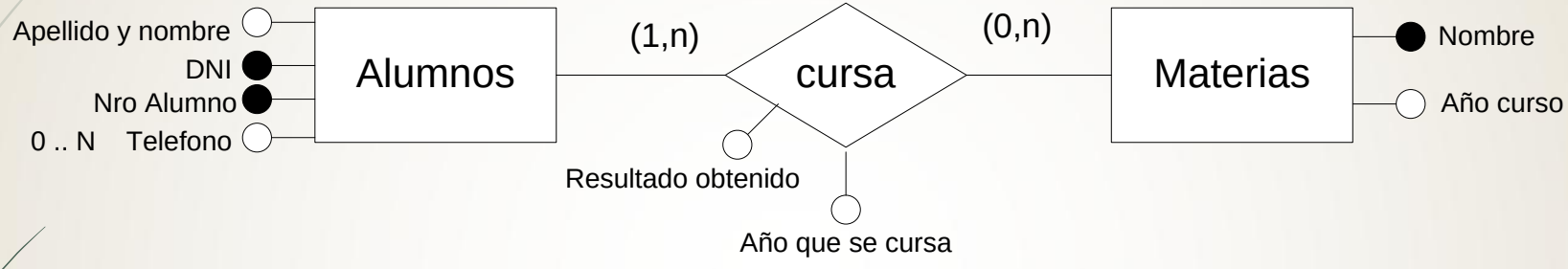
- Identificadores
- Atributos compuestos
- Jerarquías/subconjuntos

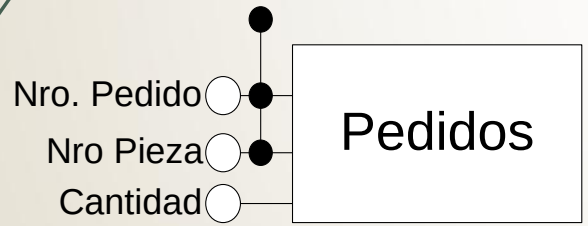
Modelo Conceptual ER

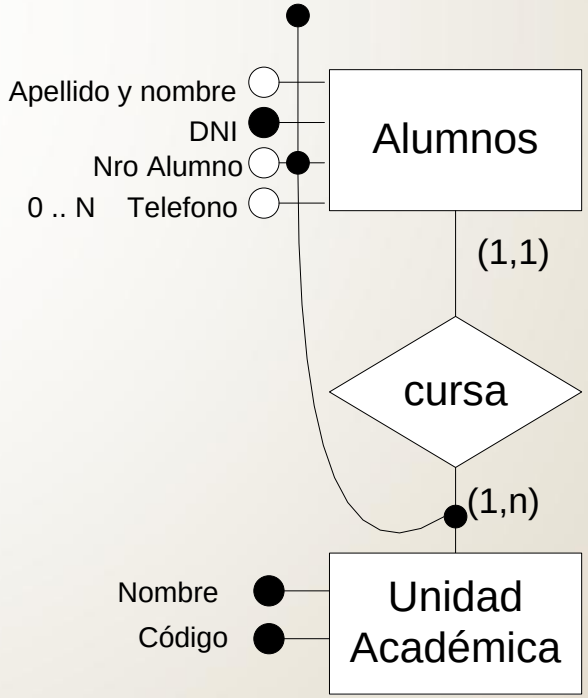
Identificadores

- Es un atributo o conjunto de atributos que permite reconocer una entidad de manera unívoca dentro del conjunto de entidades
- Pueden ser
  - simples o compuestos
  - Internos o externos

Modelo Conceptual ER



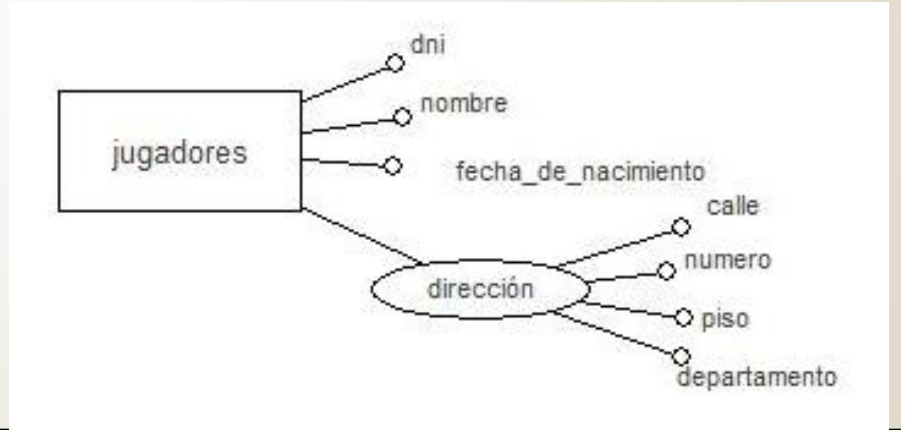




Modelo Conceptual ER

Atributos compuestos

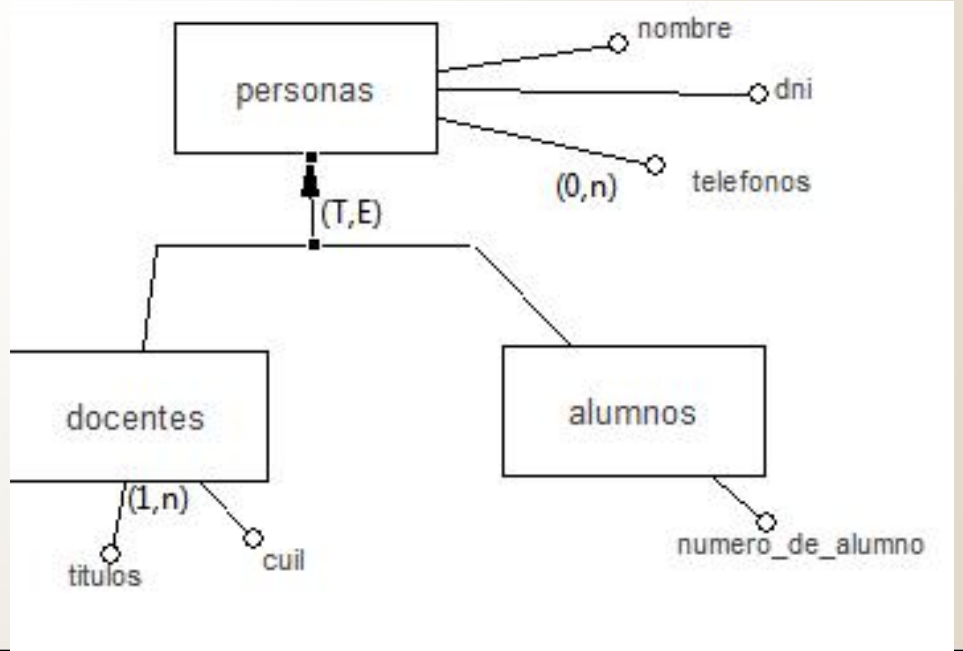
- Representan un atributo generado a partir de una combinación de atributos simples
- Puede ser polivalente y no obligatorio
- Sus atributos simples pueden ser polivalentes y no obligatorios también.



Modelo Conceptual ER

Jerarquías

- Permite extraer propiedades comunes de varias entidades (o relaciones) y generar una superentidad que las contenga
- Cobertura:
  - Total o parcial
  - Superpuesta o exclusiva



Modelo Conceptual ER

- **Ejemplo integrador** Creación de una BD para una empresa. Manipulamos empleados, departamentos y proyectos
  - La empresa está organizada en departamentos. Cada departamento tiene un nombre único, un número único y un cierto empleado que lo dirige (debe indicarse la fecha desde cuando es director). Un departamento puede estar en varios lugares
  - Cada departamento controla un cierto número de proyectos, cada uno de los cuales tiene un nombre y número único y se lleva a cabo en un único lugar.
  - Para cada empleado se conoce el nombre, sexo, dirección, salario fecha de nacimiento, número de dni (irrepetible). Todo empleado está asignado a un departamento, pero puede trabajar en varios proyectos. Interesa conocer el número de horas semanales que cada empleado le dedica a cada proyecto. Es importante conocer el jefe de cada empleado.
  - Se administra, además, datos familiares de cada empleado, De cada familiar se guardará el nombre, el sexo, la fecha de nacimiento y parentesco con el empleado.

Diseño de Bases de Datos

Clase 2

Agenda

Revisiones del modelo conceptual	• Decisiones • Transformaciones
Modelo lógico	• Atributos derivados, compuestos y polivalentes • Ciclos de entidades • Jerarquías
Modelo físico	• Conversión de entidades • Conversión de relaciones
Normalización	• Restricciones • Dependencias • Normallización

Revisiones del modelo conceptual

Decisiones

- Conviene generar una entidad con un concepto nuevo? O agregar un atributo a una entidad existente?
- Cuando se debe utilizar una generalización y cuando el concepto representa una clasificación?
- Convienen los atributos compuestos? O se deben generar atributos simples?

Revisiones del modelo conceptual

Compleción: representa todas las características del domino de aplicación (análisis de requerimientos)

Corrección: usar con propiedad conceptos E-I

- Sintáctica: conceptos E-I se usan correctamente
- Semántica: conceptos se usan de acuerdo a su definición. Errores más frecuentes:
  - Usar atributos en lugar de entidades
  - Olvidar una generalización
  - Olvidar una propiedad de herencia
  - Usar entidades en lugar de interrelaciones
  - Olvidar un identificador de una entidad
  - Omitir cardinalidad

Revisiones del modelo conceptual

Minimalidad: cada aspecto aparece una sola vez en el esquema.

- Ciclo de relaciones
- Atributos derivados

Expresividad: representa los requerimientos de manera natural y se puede entender con facilidad.

Autoexplicación: esquema se explica a si mismos cuando puede representarse un gran número de propiedades usando el modelo conceptual, sin otros formalismos.

- Eliminar sub-entidades colgantes de la generalización
- Eliminar entidades colgantes
- Crear generalización: dos entidades similares, crea una jerarquía de generalización
- Crear Subconjuntos

Revisiones del modelo conceptual

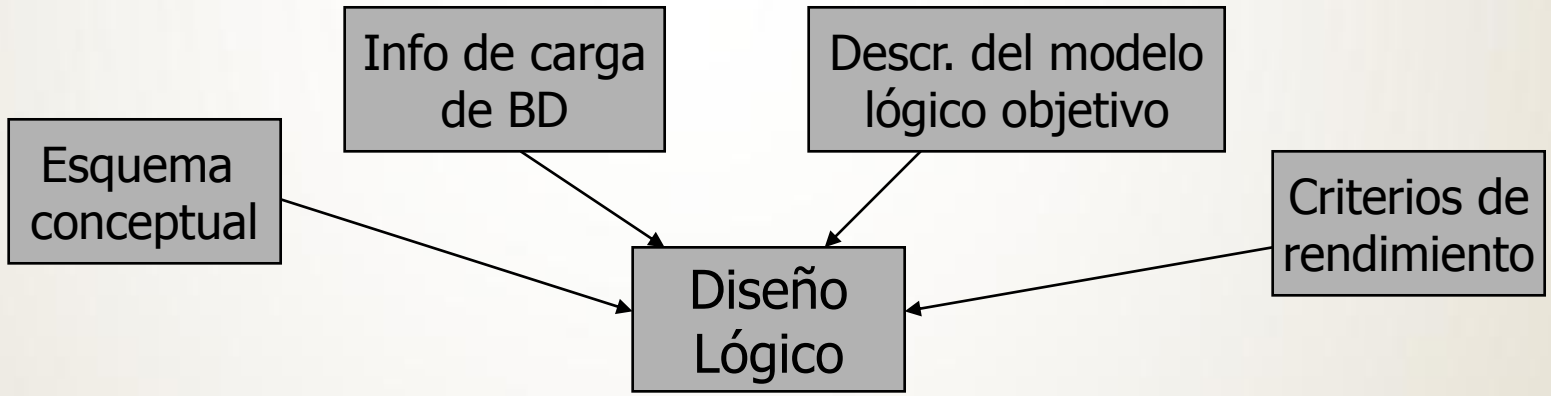
Extensibilidad: un esquema se adapta fácilmente a requerimientos cambiantes cuando puede descomponerse en partes, donde se hacen los cambios

Legibilidad:

- Utilizar herramientas automatizadas
- Estructuras simétricas
- Se minimiza el número de cruces
- Generalización sobre los hijos

Modelo lógico

- Diseño lógico de alto nivel usando E-R
  - Convertir el esquema conceptual en un esquema lógico
  - Enfoque global del diseño lógico



Modelo lógico

Decisiones

- Atributos derivados
- Atributos polivalentes
- Atributos compuestos
- Ciclo de relaciones
- Jerarquías

Modelo físico

El modelo físico (relacional) representa la BD como una colección de *relaciones*.

- En otros términos → cada relación se asemeja a una tabla de valores, o a un archivo plano de registros.
- Un registro o un elemento de una relación (tabla) se denomina *tupla*.
- Una atributo mantiene su nombre
- Cada tabla de valores resultante se denomina *relación*
  - Cada *relación* se obtiene a partir de una entidad o una relación ER.
- El tipo de datos que describe los tipo de valores de un atributo se denomina *dominio*.

Modelo físico

Pasos

- Eliminación de identificadores externos
- Selección de claves
  - Primaria
  - Candidata
- Conversión de entidades
- Relaciones

Modelo físico

Relaciones

- Cardinalidad Muchos a muchos
- Cardinalidad Uno a Muchos
  - **Clave foránea: atributo/s de una tabla que en otra tabla es/son CP y que sirven para establecer un nexo entre ambas estructuras**
  - Cobertura total
  - Cobertura Parcial
- Relaciones recursivas
- Relaciones ternarias

Modelo físico

Integridad referencial

- Propiedad deseable de las BD
- Asegura que un valor que aparece para un atributo en una tabla, aparezca además en otra tabla
- Tipos de IR:
  - Restringir la operación
  - Realizar la operación en cascasda
  - Establecer la clave Foránea en nulo
  - No hacer nada

Restricciones

Restricciones de dominio

- Especifican que el valor de c/atributo A debe ser un valor atómico del dominio de A.

Restricciones de clave

- Evita que el valor del atributo clave genere valores repetidos

Restricciones sobre nulos

- Evita que un atributo tome *nulo* en caso de no ingresarle valor

Restricciones

Restricciones de integridad

- Ningún valor de la clave primaria puede ser nulo.

Restricción de integridad referencial

- Se especifica entre dos relaciones y sirve para mantener la consistencia entre tuplas de la dos relaciones
- Establece que una tupla en una relación que haga referencia a otra relación deberá referirse a una tupla existente en esa relación
- *Clave foránea* está representada por un atributo de una relación que en otra es clave primaria.

Restricciones

Las operaciones de Alta, Baja y Modificación (ABM) pueden generar violaciones a las restricciones anteriores.

- Alta
  - Puede violar: valor nulo para clave, repetición de la clave, integridad referencial, restricciones de dominio.
  - Si se viola la regla, la operación se rechaza
- Baja
  - Puede violar: integridad referencial (se procede como en el caso anterior)
- Modificación
  - Puede violar: cualquiera de las operaciones.

Dependencias Funcionales

Definición

- Una DF es una restricción ente dos conjuntos de atributos de la BD.
- Formalmente → una DF X → Y entre dos conjuntos de atributos X e Y que son subconjuntos los atributos (R) de una relación (r), especifica una *restricción* sobre las posibles tuplas que podrían formar un estado de la relación r en R.
- La restricción indica que si t1 y t2 son dos tuplas cualesquiera en r y que si t1[X] = t2[X] entonces debe ocurrir que t1[Y] = t2[Y].
- Esto significa que los valores del componente Y de una tupla de r dependen de los valores del componente X.

Dependencias Funcionales

X → Y

- El atributo Y depende del atributo X, ó
- El atributo X determina el valor único del valor Y, ó
- El valor del atributo Y está determinado por el valor del atributo X, ó
- Y depende funcionalmente de X.

Son todos sinónimos

En general

- si una restricción en R dice que no puede haber más de una tupla con un valor X en r (convirtiendo a X en ***clave primaria***) entonces X → Y para cualquier Y de R
- Si X → Y en R, no se puede afirmar ni negar que Y → X. Cuando si y cuando no de esta afirmación???

Dependencias Funcionales

Ejemplo 1

- Departamento= (NroDpto, Nombre, #empleados)
  - Nrodpto → nombre
  - Nrodpto → #empleado
  - Nombre → #empleado ??
    - Cuando sí?
    - Cuando no?

Ejemplo 2

- Empleado = (NroEmpl, Nombre, DNI, Sexo)
  - Nroempl → nombre
  - Nroempl → dni
  - Nroempl → sexo
  - DNI → nroempl??
    - Cuando sí?
    - Que otras dependencias pueden surgir?

Que conclusiones obtenemos de estos ejemplos?

Dependencias Funcionales

Ejemplo 3

- Empl_proyecto=(nro_empl, nro_proy, horasTrabajadas, nombre_empleado, nombre_proyecto)
  - (Nro_empl, nro_proy) → horastrabajadas
  - Nro_empl → nombre_empleado
  - Nro_proy → nombre_proyecto
  - Si continuamos en análisis de la transparencia anterior
    - (nro_empl, nro_proy) → nombre_empleado ??
    - (nro_empl, nro_proy) → nombre_proyecto ??
- Que conclusión podemos obtener?

Dependencias Funcionales

Dependencia funcional completa

- Si A y B son atributos de una relación r, B depende funcionalmente de manera completa de A, si B depende de A pero de ningún subconjunto de A.
- En la transparencia anterior
  - (nro_empl, nro_proy) → nombre_empleado
  - Nro_empl → nombre_empleado
  - Ambas funcionales, cual completa?

  - (nro_empl, nro_proy) → nombre_proyecto
  - Nro_proy → nombre_proyecto
  - Idem anterior

Dependencias Funcionales

Dependencia funcional parcial

- A → B es una dependencia funcional parcial si existe algún atributo que puede eliminarse de A y la dependencia continúa verificándose
- En la transparencia anterior
  - (nro_empl, nro_proy) → nombre_empleado
  - Nro_empl → nombre_empleado
  - La primera es una **dependencia PARCIAL**

  - (nro_empl, nro_proy) → nombre_proyecto
  - Nro_proy → nombre_proyecto
  - Idem anterior

Dependencias Funcionales

Dependencia funcional transitiva

- Una condición en la que A, B y C son atributos de una relación tales que A → B y B → C entonces C depende transitivamente de A a través de B
- Ejemplo
  - Nro_empleado → nombre, posición, salario, nro_depto, nombre_depto
  - Nro_depto → nombre_depto.
  - En este ejemplo
    - A = nro_empleado
    - B = nro_depto
    - C = nombre_depto.

Dependencias Funcionales

Resumen

- Dependencia funcional
- Dependencia parcial
  - Parte_clave → no_clave
- Dependencia transitiva
- Dependencia Boyce Codd (explicada más adelante)
  - No_clave → parte_clave

Normalización

Definición:

- Técnica de diseño de BD que comienza examinando las relaciones que existen entre los atributos (dependencias funcionales). La normalización identifica el agrupamiento óptimo de estos atributos, con el fin de identificar un conjunto de relaciones que soporten adecuadamente los requisitos de datos de la organización.

Propósito

- Producir un conjunto de relaciones (tablas) con una serie de propiedades deseables partiendo de los requisitos de datos de una organización.

Normalización

La normalización es una técnica formal que puede utilizarse en cualquier etapa del diseño de BD.

La redundancia de datos en un modelo es la causa primaria de posibles inconsistencias.

Primer paso para un proceso de normalización

- Identificar la CP y las CC de cada relación (tabla) del modelo.

Normalización

Inicialmente (1972)

- Primera Forma Normal
- Segunda Forma Normal → sobre DF
- Tercera Forma Normal → sobre DF

Se incorpora luego (1974)

- Forma Normal de Boyce Codd → sobre DF

Luego 1977 y 1979

- Cuarta forma normal → sobre DM
- Quinta forma normal

Normalización

Proceso → incremental → cada vez más restrictivo

- Comienza con BD en forma NO normal
- A medica que se avanza las relaciones (tablas) tiene un formato cada vez más restringido y son menos vulnerables a anomalías de actualización.
- En general, 1NF es muy restrictiva (se aplica siempre)
- El resto puede ser opcional, de hecho 2NF y 3NF normalmente se aplican siempre.

Normalización

Primera Forma Normal (1NF)

- Una tabla que contienen uno o más grupos repetitivos no está en 1FN, o sea una tabla que tenga atributos polivalentes.
- Un modelo estará en 1FN si para toda relación r del modelo (tabla) cada uno de los atributos que la forman es si y solo sí monovalente.
- Ej persona = (dni, nombre, sexo, títulos*)
  - Se observa que el atributo títulos es polivalente
  - Solución
    - Persona = (dni, nombre, sexo)
    - títulos = (id, descripción)
    - Posee = (dni, id_titulo)

Normalización

Segunda forma normal (2NF)

- Una tabla que tenga atributos que dependan parcialmente de otro no está en 2NF
- Un modelo está en 2NF si y solo sí está en 1NF y para toda relación r del mismo (tabla) no existen dependencia parciales.
- Ej renta = (#cliente, #propiedad, nombrecliente, nombre propietario, monto renta, fecha inicio, duración)

Normalización

Dependencias

- #cliente, #propiedad → nombrecliente, nombrepropietario, monto renta, fecha inicio, duración (DF)
- #cliente → nombrecliente (DP)
- #propiedad → nombrepropietario (DP)

Solución

- Cliente = (#cliente, nombre)
- Propiedad = (#propiedad,nombrepropietario)
- Renta = (#cliente, #propiedad, monto renta, fecha inicio, duración))

Normalización

Ej 2: empleadoproyecto = (dniempleado, #proyecto, horastrabajadas, nombreempleado, nombreproyecto, fecha inicio proyecto, fecha inicio empleado proyecto)

- Dependencias funcionales
  - Dniempleado, #proyecto → horas trabajadas, nombreempleado, nombreproyecto, fecha inicio proyecto, fecha inicio empleadoproyecto (DF)
  - Dniempleado → nombreempleado (DP)
  - #proyecto → nombre proyecto (DP)
- Solución
  - Empleados = (dniempleado, nombreempleado)
  - Proyectos = (#proyecto, nombreproyecto, fecha inicio proyecto)
  - Empleadoproyecto = (dniempleado, #proyecto, fecha incio empleado proyecto)

Normalización

Tercera forma normal (3NF)

- Una tabla que tenga atributos que dependan transitivamente de otro no está en 3NF
- Un modelo está en 3NF si y solo sí está en 2NF y para toda relación r del mismo (tabla) no existen dependencia transitivas.
- Ej empleado = (dniempleado, nombreempleado, #depto, nombredepto)
  - Dependencias
    - Dniempleado → nombreempleado, #depto, nombre depto (DF)

Normalización

- #depto → nombredepto (DT)
- Solución
  - Empleado = (dniempleado, nombre, #depto)
  - Departamento = (#depto, nombredepto)

- Ej2 parcelas = (#propiedad, municipio, númeroparcela, area, precio, tasa fiscal)
- Dependencias
  - #propiedad → municipio, numeroparcela, area, precio, tasa fiscal (DF)
  - Municipio → tasa fiscal (DT)
  - Area → precio (DT)
- Solución
  - Parcela = (#propiedad, municipio, númeroparcela , area)
  - Municipio = (municipio, tasa fiscal)
  - Areas = (area, precio)

Normalización

Boyce Codd forma normal (BCNF)

- Una tabla que tenga atributos que dependan de acuerdo a la definición de Boyce Codd de otro no está en BCNF
- Un modelo está en BCNF si y solo sí está en 3NF y para toda relación r del mismo (tabla) no existen dependencia de Boyce Codd
- Algunos comentarios
  - Fue propuesta como una "suavización" de 3NF
  - Pero resultó ser más restrictiva

Otra acepción de Boyce Codd

- Una relación (tabla) está en BCNF si y solo sí todo determinante es una clave candidata.

Normalización

- Ejemplo entrevista = (#cliente, fechaentrevista, horaentrevista, empleado, lugarentrevista)
  - las DF existentes son:
    - #cliente, fechaentrevista → hora entrevista, empleado, lugarentrevista (CP)
    - Empleado, fechaentrevista, horaentrevista → #cliente (CC)
    - Lugarentrevista, fechaentrevista, horaentrevista → empleado, #cliente (CC)
    - Empleado, fechaentrevista → lugarentrevista
  - Como los tres primeros determinantes son CP o CC no generan inconvenientes.
  - Debemos, entonces, analizar la cuarta DF.
    - No hay problema con DP o DT
    - Pero el determinante no es CC o CP → no está en BCNF

Normalización

- Veamos los problemas que pueden surgir

#cliente	*fechaentrevista*	*Horaentrevista*	*Empleado*	*lugarentrevista*
C123	12/12/2004	12:30 hs.	García	Aula 4
C332	12/12/2004	12:30 hs.	Perez	Aula 3
C340	15/12/2004	13:00 hs.	García	Aula 2
C124	12/12/2004	13:00 hs.	Perez	Aula 3

- Si el empleado Perez cambia su cita del día 12/12/2004 del aula 3 al aula 20, que pasa? Cuantos renglones hay que cambiar?
- Entonces es claramente visible que la información está repetida

Normalización

- Como resolvemos el problema anterior
  - Entrevista= (#cliente, fechaentrevista, horaentrevista, empleado)
  - lugarreunión = (empleado, fechaentrevista, lugarenrevista)
- En la conversión realizada
  - #cliente, fechaentrevista → hora_entrevista, empleado (CP)
  - Empleado, fechaentrevista, horaentrevista → #cliente (CC)
  - Empleado, fecha entrevista → lugarentrevista (CP)
  - Pero se ha perdido una CC del problema
    - Lugarentrevista, fechaentrevista, horaentrevista → empleado, #cliente (CC)

Normalización

Entonces? Que hacer?

- La decisión de si es mejor detener el proceso en 3NF o llegar a BCNF depende de
  - la cantidad de redundancia que resulte de la presencia de una DF de Boyce Codd.
  - De la posibilidad de perder una CC con la cual se podrían realizar muchos más controles sobre los datos.

Dependencias Multivaluadas

- La posible existencia de DM en una relación se debe a 1NF, que impide que una tupla tenga un conjunto de valores diferentes.
- Así, si una tabla tiene dos atributos multivaluados, es necesario repetir cada valor de uno de los atributos con cada uno de los valores del otro. Así se garantiza la coherencia en la BD.
- En general, una DM se da entre atributos A, B y C en una relación de modo que para cada valor de A hay un conjunto de valores se B y un conjunto de valores de C, sin embargo los conjuntos B y C no tienen nada entre sí.

Dependencias Multivaluadas

- Ejemplo T1=(nombre_empleado, nombre_propietario, sucursal)

Sucursal	Empleado	propietario
Alfa	Gomez	Perez
Alfa	Gomez	García
Alfa	Rodriguez	Perez
Alfa	Rodriguez	García

- Supongamos que aparece el empleado Fernandez →

Sucursal	Empleado	propietario
Alfa	Gomez	Perez
Alfa	Gomez	García
Alfa	Rodriguez	Perez
Alfa	Rodriguez	García
Alfa	Fernandez	Perez
Alfa	Fernandez	García

Dependencias Multivaluadas

- Supongamos que aparece el propietario Alvarez→

Sucursal	Empleado	propietario
Alfa	Gomez	Perez
Alfa	Gomez	García
Alfa	Rodriguez	Perez
Alfa	Rodriguez	García
Alfa	Fernandez	Perez
Alfa	Fernandez	García
Alfa	Gomez	Alvarez
Alfa	Rodriguez	Alvarez
Alfa	Fernandez	Alvarez

- La cantidad de información que se repite es muy alta

Dependencia multivaluada

- Se dice que A →→ B y que A →→ C
- Se lee A multidetermina B y A multidetermina C
- No es un problema el hecho que un atributo esté multideterminado.

Normalización

- Cuarta forma Normal (4FN)
  - Un modelo está en 4FN si y solo sí está en BCNF y para toda relación r del mismo (tabla) sólo existen dependencia multivaluadas triviales.
  - Cuales DM son triviales?
    - A →→ B
    - A →→ C
    - Las anteriores son triviales.
  - Volvamos al ejemplo anterior →
    - (A,B) →→ C
    - (A,C) →→ B

Normalización

- Por que la afirmación anterior?
  - (sucursal, empleado) →→ propietario
  - (sucursal, propietario) →→ empleado
- Esto genera mucha repetición de información.
- Solución
  - t1= (sucursal, empleado)
  - t2= (sucursal, propietario)

Normalización

- Quinta Forma Normal (5FN)
  - Un modelo está en 5FN si está en 4FN y no existen relaciones con dependencias de combinación
  - Una dependencia de combinación es una propiedad de la descomposición que garantiza que no se generen tuplas espurias al volver a combinar las relaciones mediante una operación del álgebra relaciónal.
  - En otras palabra ….→

Normalización

- Supongamos que tenemos una tabla
  - Exporta = (compañía, país, producto)

Compañía	Pais	Producto
A	Alfa	Uno
A	Alfa	Dos
A	Beta	Uno
A	Beta	Tres
A	Gamma	Uno

  - Que podemos decir?

- Supongamos ahora que la compañía A empieza a fabricar el producto cuatro y que lo compran el país Alfa y Gamma

Compañía	Pais	Producto
A	Alfa	Uno
A	Alfa	Dos
A	Beta	Uno
A	Beta	Tres
A	Gamma	Uno
A	Alfa	Cuatro
A	Gamma	Cuatro

Normalización

- Se genera una gran repetición de información
- Caemos en 4FN?
- Por que?
- Solución
  - Exporta = (compañía, pais)
  - Fabrica = (Compañía, producto)
  - Compra = (país, producto)
- Hay mas formas normales?
  - Si algunos casos más…, no los veremos

Diseño de Bases de Datos

Clase 3

Prof. Pablo Thomas

Rodolfo Bertone

Lenguajes de consulta

Lenguajes de consulta: utilizados para operar con la BD.

- Procedurales: (instrucciones para realizar secuencia de operaciones) (que y como)
- No procedurales: (solicita directamente la información deseada) (que).

Analizaremos primero las consultas

- Las consultas representan el 80% de las operaciones registradas sobre una BD

Álgebra Relacional

Álgebra Relacional:

- Lenguaje de consultas procedural
- Operaciones de una o dos relaciones de entrada que generan una nueva relación como resultado

Operaciones fundamentales

- Unitarias (selección, proyección, renombre)
- Binarias(Producto cartesiano, Unión, diferencia)

Álgebra Relacional

Dadas las siguientes tablas

- **Asociados**=(idsocio, nombre, dirección, teléfono, sexo, estadocivil, fechanacimiento, idlocalidad)
- **Deportes**=(iddeporte, nombre, monto_cuota, idsede)
- **Practica** = (idsocio, iddeporte)
- **Localidad** =(idlocalidad, nombre)
- **Sedes** = (idsede, nombre, dirección, idlocalidad)

Álgebra Relacional

Selección:

- selecciona tuplas que satisfacen un predicado dado. Operador: σ
- Ejemplo 1: mostrar todos los asociados casados
- Ejemplo 2: deportes con cuota superior a $2000 o inferior a $1000

Álgebra Relacional

Proyección:

- devuelve la relación argumento con columnas omitidas. Operador: π
- *En el resultado se eliminan las tuplas repetidas*
- Ejemplo 3: nombres de los asociados
- Ejemplo 4: monto de cuota y nombre de cada deporte

Álgebra Relacional

Producto Cartesiano:

- Conecta dos entidades de acuerdo a la definición matemática de la operación. Operador **x**
- Ejemplo 5: Mostrar cada asociado y la localidad donde vive.
- Ejemplo 6: mostrar las sedes de La Plata.
- Ejemplo 7: mostrar cada deporte y el nombre del asociado que lo practica.

Álgebra Relacional

Renombrar:

- permite utilizar la misma tabla en un, por ej., producto cartesiano.

  Operación ρ
- Ejemplo 7: mostrar todos los asociados que viven en la misma dirección que el socio con id 75

Unión:

- tuplas comunes a dos relaciones, equivalente a la unión matemática. Debe efectuarse entre relaciones con sentido.
- Las instancias repetidas se eliminan automáticamente.
- Las dos tablas deben ser de unión compatibles
  - Igual cantidad de atributos
  - i-ésimo atributo de 1° tabla y i-ésimo atributo de 2° tabla deben tener el mismo dominio (i:1..n)
- Operación $\cup$

- Ejemplo 8: asociados que practiquen vóley o futbol

Álgebra Relacional

Diferencia:

- diferencia de Conjuntos. Operación -
- Las dos tablas deben ser de unión compatibles
- Ejemplo 9: asociados que practiquen vóley y no o futbol
- Ejemplo 10: deporte por el que se pague mayor cuota

Álgebra Relacional

Definición formal de Álgebra Relacional:

- Una expresión básica en AR consta de
  - Una relación de una Base de Datos
  - Relación constante
- Una expresión general se construye a partir de sub-expresiones (E1,E2,...En)
- Expresiones:
  - E1 $\cup$ E2
  - E1 - E2
  - E1 x E2
  - σ_p(E1) P predicado con atributos en E1
  - π_s (E1) S lista de atributos de E1
  - ρ_x (E1) X nuevo nombre de E1

Álgebra Relacional

Operaciones adicionales

- Intersección
- Producto Natural
- Asignación temporal

Álgebra Relacional

Producto Natural:

- hace el producto cartesiano con una selección de tuplas "con sentido" eliminando las columnas (atributos) repetidas. Si R y S dos relaciones no tienen atributos en común es igual al prod.cart. Operación |x|
- Ejemplo11: asociados que practican futbol
- Ejemplo 12: nombre y dirección de los asociados que son de La Plata

Álgebra Relacional

Intersección:

- equivalente a la intersección matemática.
- Las dos tablas deben ser de unión compatibles Operación ∩

Asignación:

- expresión que asigna a una variable temporal el resultado de una operación. Operación ⇐
- Temp ⇐ Operación del Álgebra

Álgebra Relacional

Operaciones de Updates:

- **Agregar tuplas**
  - $r \Leftarrow r \cup E$ (r relación y E nueva tupla
- **Eliminar tuplas**
  - $r \Leftarrow r - E$
- **Actualización de datos**
  - $\delta_{A \Leftarrow E}$ (r)
  - Ej: $\delta_{saldo \Leftarrow saldo * 1.05}$ (depósito)

Ejercicios

- Dadas las siguientes tablas

  Cliente (id_cliente, nombre_cliente, renta_anual, tipo_cliente)

  Embarque (embarque_#, id_cliente, peso, camión_#, destino, fecha)

  Camión (camión_#, nombre_chofer)

  Ciudad (nombre_ciudad, población)

- Resolvamos en AR

1. Cuales son los números de los camiones que han llevado paquetes (embarques) por encima de 100 kg?
2. Clientes que tuvieron embarques de mas de 100 kg con destino Córdoba
3. Incrementar el peso de los envios a cordoba un un 50%
4. Mostrar los clientes con envíos a Tucuman y que tengan renta anual superior a 200.000$

Diseño de Bases de Datos

Clase 4

Prof. Pablo Thomas

Rodolfo Bertone

Agenda

Lenguaje de Consultas Estructurado (SQL)

- Definiciones
- DDL / DML
- Consultas/Updates
- Ejemplos

Lenguaje de Consultas Estructurado (SQL)

Historia

1986 Es un estándar ANSI

1992 se amplia el estándar (SQL2 o SQL 92)

1999 Se crea SQL 2000 incorporando expresiones regulares, consultas recursivas y características de OO

2003 Surge SQL 3 agrega características de XML

2006 se definen características que lo acercan al mundo W3C

SQL → DDL

Lenguaje de definición de datos

- Es muy amplio
- Solo veremos las operaciones mas comunes
- CREATE DATABASE
- DROP DATABASE
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

SQL → DDL

- Ejemplos de Creación de Tablas

```
CREATE TABLE empresas ( idempresa INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
                        empresa VARCHAR(100) NOT NULL,
                        abreviatura VARCHAR(10) NULL,
                        cuit VARCHAR(13) NULL,
                        direccion VARCHAR(100) NULL,
                        observaciones TEXT NULL,
                        PRIMARY KEY(idempresa),
                        UNIQUE INDEX empresas_index19108(empresa));
```

SQL → DDL

- Ejemplos de Creación de Tablas

```
CREATE TABLE pacientesempresas
        ( idpacienteempresa INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
          dpaciente_os INTEGER UNSIGNED NOT NULL,
          idempresa INTEGER UNSIGNED NOT NULL,
          fecha_desde DATE NOT NULL,
          fecha_hasta DATE NULL,
          PRIMARY KEY(idpacienteempresa),
          INDEX empleadosempresas_FKIndex1(idempresa),
          INDEX pacientesempresas_FKIndex2(idpaciente_os),
          FOREIGN KEY(idempresa) REFERENCES empresas(idempresa)
                                 ON DELETE RESTRICT
                                 ON UPDATE NO ACTION,
          FOREIGN KEY(idpaciente_os) REFERENCES pacientes_os(idpaciente_os)
                                 ON DELETE RESTRICT
                                 ON UPDATE NO ACTION);
```

SQL → DDL

- Ejemplo de modificación de tablas

```
ALTER TABLE empresas (
  Add column razon_social VARCHAR(100) NOT NULL,
  Drop column cuit,
  Alter column direccion VARCHAR(50) NULL);
```

SQL → DML

- Estructura básica

```
SELECT   lista_de_atributos
FROM     lista_de_tablas
WHERE    condición
```

```
SELECT  atr1, atr2,atr3
FROM tabla1, tabla2
WHERE atr4="Valor"
```

Equivale a

$\pi_{atr1, atr2, atr3} (\sigma_{atr4 = 'valor'} (tabla1 \times tabla2))$

SQL → DML

Tablas sobre las cuales se resolverán las consultas

- Asociados=(idsocio, nombre, dirección, teléfono, sexo, estadocivil, fechanacimiento, idlocalidad)
- Deportes=(iddeporte, nombre, monto_cuota, idsede)
- Practica = (idsocio, iddeporte)
- Localidad =(idlocalidad, nombre)
- Sedes = (idsede, nombre, dirección, idlocalidad)

SQL → DML

- Ejemplo 1: nombre de todos los asociados

```
SELECT  nombre
FROM    asociados
```

  - La Cláusula Select puede contener
    - * (incluye todos los atributos de las tablas que aparecen en el from)
    - Distinct (eliminan tuplas duplicadas)
    - All (valor por defecto, aparecen todas las tuplas)
  - Ejemplo 2: mostrar todos los datos de los asociados
  - Ejemplo 3: mostrar todos los id de localidades donde viven los asociados, sin repetir valores

SQL → DML

- Ejemplo 4: mostrar de cuanto sería la cuota de cada deporte si se incrementara en un 25%

```
SELECT montocuota*1.25
FROM deportes
```

SQL → DML

- Cláusula WHERE
  - Ejemplo 5: **Operadores lógicos**: mostrar los asociados varones casados.
  - Ejemplo 6: **Operador BETWEEN**: mostrar los deportes cuya cuota esté entre 400 y 600 pesos mensuales

```
SELECT  nombre
FROM   deporte
WHERE montocuota>= 400 AND  montocuota <= 600
```

```
SELECT  nombre
FROM   deporte
WHERE montocuota BETWEEN 400 and 600
```

SQL → DML

- Cláusula FROM
  - Ejemplo 7: **Producto Cartesiano**: mostrar para cada asociado su nombre y la localidad de residencia
  - Ejemplo 8: **Producto Natural** mostrar para cada asociado su nombre y la localidad de residencia

```
SELECT  asociado.nombre, localidad.nombre
FROM    asociado, localidad
WHERE asociado.idlocalidad = localidad.idlocalidad
```

```
SELECT  asociado.nombre, localidad.nombre
FROM    asociado  INNER JOIN  localidad ON (asociado.idlocalidad=localidad.idlocalidad)
```

SQL → DML

- Operación de Renombrar
  - Ejemplo 9: mostrar todos los deportes salvo el mas costoso.
  - Ejemplo 10: presentar todos los asociados con nombre, dirección y localidad. El listado debe figurar con la leyenda DIRECCIÓN LEGAL.
- Operaciones sobre cadenas
  - Operador **Like:** **%**, _
    - "Alfa%": cualquier cadena que empiece con Alfa
    - "%casa%": cualquier cadena que tenga casa en su interior
    - "_ _ _": cualquier cadena con tres caracteres
    - "_ _ _%": cualquier cadena con al menos tres caracteres.
    - Ejemplo 11: mostrar aquellos asociados cuyo nombre empiece con RA
    - Ejemplo 12: mostrar los deportes que incluyan la palabra BALL en su definición

SQL → DML

- Operadores que permite **ORDENAR** las tuplas
  - **ORDER BY** *atributo*:
    - especifica el atributo por el cual las tuplas serán ordenadas
    - Ejemplo 13: presentar todos los asociados ordenados por nombre.

```
SELECT  nombre
FROM    asociados
ORDER BY nombre
```

    - **Desc, asc**: por defecto ascendente, se puede especificar descendente.
    - Ejemplo 14 : presentar los asociados que nacieron el mes de agosto ordenados por fecha desde el 31 al 1 de agosto.

SQL → DML

Otras operaciones del algebra, Operaciones sobre conjuntos

- Unión: agrupa las tuplas resultantes de dos subconsultas.
- Union all conserva duplicados
  - Ejemplo 15: asociados que practican futbol o voley
- Intersección:
  - Ejemplo 16: asociados que practican futbol y voley
- Diferencia: (except)
  - Ejemplo 17: asociados que p

```
(SELECT  nombre
FROM   asociados a INNER JOIN practica p ON a.idsocio = p.idsocio
                   INNER JOIN deportes d ON p.iddeporte = d.iddeporte
WHERE d.nombre = "FUTBOL" )
UNION  /  UNION ALL  /   INTERSECT  / EXCEPT
(SELECT  nombre
FROM   asociados a INNER JOIN practica p ON a.idsocio = p.idsocio
                   INNER JOIN deportes d ON p.iddeporte = d.iddeporte
WHERE d.nombre = "VOLEY" )
```

SQL → DML

- Funciones de agregación:
  - Promedio (avg): aplicable a atributos numéricos, retorna el promedio de la cuenta
  - Mínimo (min): retorna el elemento más chico dentro de las tuplas para ese atributo
  - Máximo (max): retorna el elemento más grande dentro de las tuplas para ese atributo
  - Total (sum): aplicable a atributos numéricos, realiza la suma matemática
  - Cuenta (count): cuenta las tuplas resultantes.
- Ejemplos:
  - Ejemplo 18: mostrar la cantidad de asociados del club
  - Ejemplo 19: mostrar el asociado que aparece primero en una lista alfabética
  - Ejemplo 20: mostrar cual es la cuota promedio que se cobra
  - Ejemplo 21: mostrar cual será la recaudación mensual de futbol.
  - Ejemplo 22: mostrar el deporte que cobra la cuota mas alta

SQL → DML

- Operación de Agrupamientos **(GROUP BY):**
  - Permite agrupar un conjunto de tuplas por algún criterio
    - Ejemplo 23: obtener la cantidad de asociados que practica cada deporte
    - Ejemplo 24: obtener la recaudación mensual de cada deporte
    - Ejemplo 25: informar para cada asociado cuantos deportes practica y el valor en cuotas que abona
    - Ejemplo 26: que modificaría al problema anterior para mostrar solamente los asociados de La Plata
  - **HAVING**: permite aplicar condiciones a los grupos
    - Ejemplo 27: mostrar todos los deportes (Id es suficiente) y la cantidad de asociados que lo practican siempre y cuando haya mas de 100 miembros

SQL → DML

- Valores nulos:
  - Los atributos con valores nulos deben ser tratados de manera especial
  - Ejemplo 28: Mostrar aquellos deportes que no tengan valor ingresado en su cuota

```
SELECT nombre
FROM deportes
WHERE montocuota IS NOT NULL
```

SQL → DML

- Subconsultas anidadas
  - Pertenencia a conjuntos: **IN**
    - Ejemplo 29: mostrar aquellos asociados que practiquen Basquet.
    - Ejemplo 30: mostrar los asociados de Gonnet que practiquen Handball.
  - Comparación de Conjuntos
    - > **some** (<, =, >=, <=, <>)
    - Ejemplo 31: mostrar todos los deportes, menos el mas económico

```
Select  nombre
From deporte
Where monto_cuota > some ( select monto_cuota from deporte)
```

    - > **all** (<, =, >=, <=, <>)
    - Ejemplo 32: presentar el deporte mas oneroso

SQL → DML

- Cláusula **EXIST**: devuelve verdadero si la subconsulta argumento no es vacía.
  - Ejemplo 33: mostrar los asociados que practican futbol y voley (una tercera variante)

Select a.nombre

From asociados a, deporte d, practica p

Where a.idasociado=p.idasociado and. p.iddeporte=d.iddeporte and d.nombre="futbol" and exist (select * from… where …. And a.idasociado=a2.idasociado)

  - Ejemplo 34: obtener los asociados que practiquen todos los deportes.
- **Creación de vistas** (tienen la estructura de una tabla y almacena temporalmente una consulta). Cada vez que la vista es utilizada, la consulta almacenada es "recalculada"
  - Create View *nombre* as <expresion>
  - Ejemplo 35: crea una vista con todos los asociados que practican yudo
  - CreateView Judo as (select nombre from….)

SQL → DML

- Variantes del producto natural
  - **Left outer Join:** primero se calcula el inner join (ídem anterior) y luego cada tupla t perteneciente a la relación de la izquierda que no encontró par aparece en el resultado con valores nulos en los atributos del segundo lado.
  - **Right outer Join:** ídem anterior pero aparecen las tuplas t de la relación de la derecha
  - **Full outer join:** aparecen las tuplas colgadas de ambos lados.
  - Ejemplos:
    - Ejemplo 36: Presentar para cada deporte un listado de asociados que lo practican. Pueden haber deportes que aun no tengan asociados y deben aparecer igualmente en el listado
    - Ejemplo 37:Presentar para cada asociado los deportes que practican. Pueden que haya alguno que no practique deportes, igualmente debe aparecer.

SQL → DML

- ABM sobre tablas:
  - Inserción:
  - **INSERT INTO *tab_name* (<*column_name*>,) VALUES (<*valor*>,)**
    - Ejemplo 38: agregar un nuevo asociado a la tabla
  - Borrado
  - **DELETE FROM *tab_name***

    [WHERE *condición*];
    - Ejemplo 39: quitar de la tabla el deporte bochas
    - Ejemplo 40: como haría en el caso anterior si bochas tuviera deportistas y estuviera definida integridad referencial
  - Actualización
  - **UPDATE tab_name**

    set column name = valor
    - Ejemplo 41: incrementar la cuota de cada deporte en un 20%
    - Ejemplo 42: incrementar la cuota de hockey en un 30%

SQL → DML

- Ejercicios para resolver

Dadas las siguientes tablas

Cliente (id_cliente, nombre_cliente, renta_anual, tipo_cliente)

Emparque (embarque_#, id_cliente, peso, camión_#, destino, fecha)

Camión (camión_#, nombre_chofer)

Ciudad (nombre_ciudad, población)

SQL → DML

1. Cuál es el nombre del cliente 433?
2. Cuál es la ciudad destino del embarque 3244?
3. Cuales son los números de los camiones que han llevado paquetes (embarques) por encima de 100 kg?
4. Dé todos los datos de los embarques de más de 20 kg?
5. Cree una lista por orden alfabético de los clientes con renta anual de más de 10 millones?
6. Cual es el Id del cliente José García?
7. Mostrar los nombres de los clientes que han enviado embarques a las ciudades cuyo nombre empieza con C.
8. Mostrar los nombres de los clientes que han enviado embarques a las ciudades cuyo nombre termina con City.
9. Mostrar los nombres de los clientes que tienen una D como tercera letra del nombre.
10. Mostrar los nombres de los clientes que sean minoristas

SQL → DML

11. Cómo se llaman los clientes que han enviado paquetes a Bariloche?
12. A cuales destinos han hecho envíos las compañías con renta anual menor que 1 millón?
13. Cuales son los nombres y las poblaciones de las ciudades que han recibido embarques que pesen más de 100 kg?
14. Cuales con los clientes que tienen más de 5 millones de renta anual y que han enviado embarques de menos de 1 kg?
15. Quienes son los clientes que tienen más de 5 millones de renta anual y que han enviado embarques de menos de 1kg. O han enviado embarques a Villa La Angostura?
16. Quienes son los choferes que han conducido embarques de clientes que tienen renta anual mayor de 20 millones a ciudades con más de 1 millón de habitantes?
17. Indique los choferes que han transportado embarques a cada una de las ciudades.
18. Indique las ciudades que han recibido embarques de clientes que tienen más de 15 millones de renta anual.
19. Indique el nombre y la renta anual de los clientes que han enviado embarques que pesan más de 100 kg.
20. Indique los clientes que han tenido embarques transportados en cada camión.

SQL → DML

21. Cual es el peso promedio de los embarques?
22. Cual es el peso promedio de los embarques que van a Neuquén?
23. De un alista de los clientes para los que todos sus embarques han pesado más de 25 kg.
24. Cuales ciudades de la BD tienen la menor y la mayor población?
25. Agregue el camión 95 con el chofer García a la BD
26. Borre de la BD todas las ciudades con población de menos de 5000 habitantes. Debe sacar, además los embarques que haya en dicha ciudad.
27. Borre de la BD todas las ciudades con población de menos de 5000 habitantes que no posean embarques enviados.
28. Convierta el peso de cada envío a libras, para ello se sabe que una libra son 2.2 kg. (aproximadamente).

SQL → DML

29. Indique las ciudades que han recibido embarques de todos los clientes
30. Cuantos embarques han sido enviados pro el cliente 433?
31. Para cada cliente ¿cuál es el peso medio de los paquetes enviados por él?
32. Para cada ciudad ¿cuál es el peso máximo de un paquete que haya sido enviado a dicha ciudad?
33. Para cada ciudad con población por encima de un millón de habitantes ¿cuál es el peso menor de un paquete enviado a dicha ciudad?
34. Para cada ciudad que haya recibido al menos diez paquetes, ¿cuál es el peso medio de los paquetes enviados a dicha ciudad?

Diseño de Bases de Datos

Clase 5

Prof. Luciano Marrero

Pablo Thomas

Rodolfo Bertone

Agenda

Optimización de Consultas

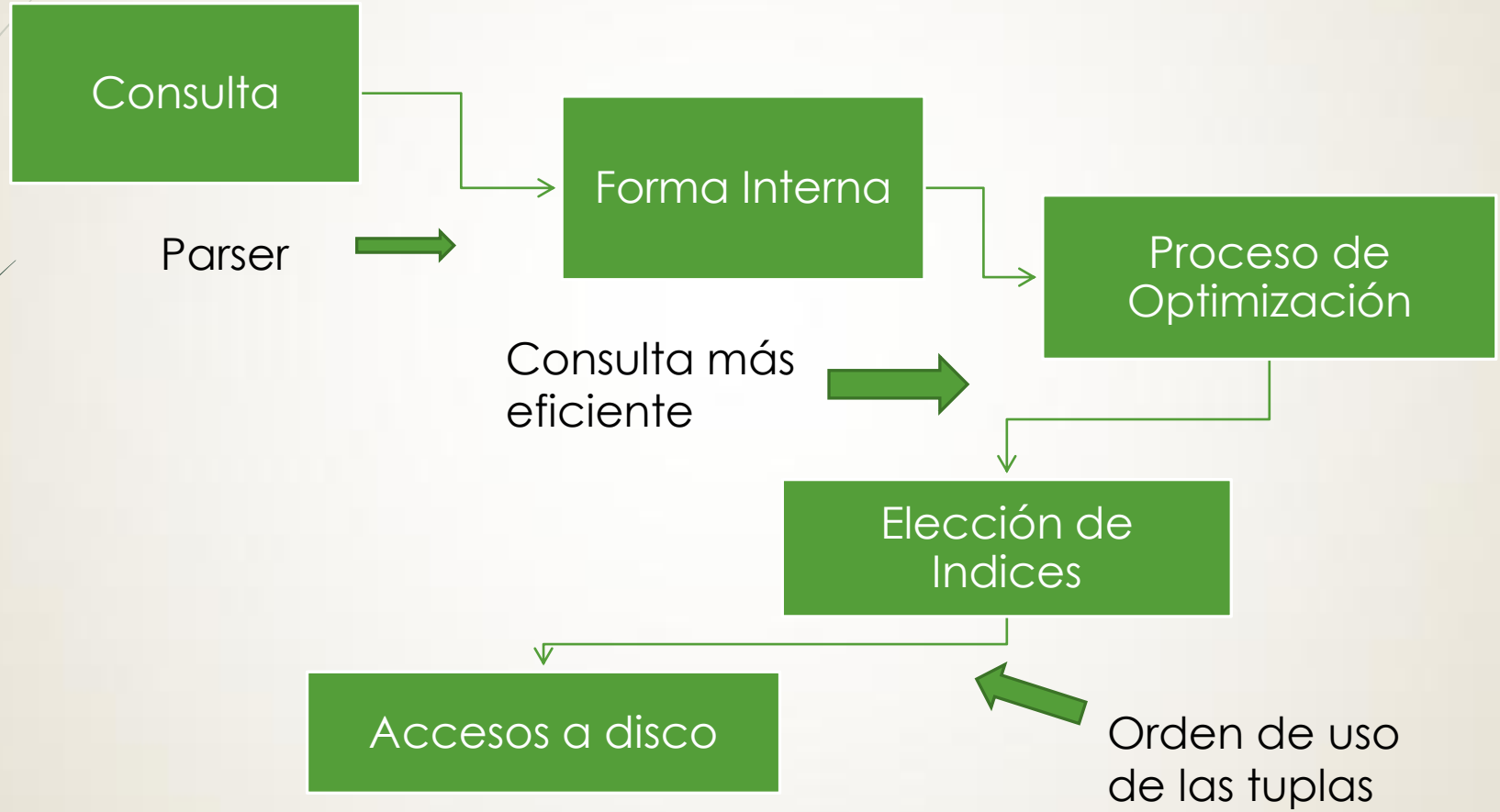
- Costo de Consulta
- Accesos
- Operaciones básicas

Optimización de Consultas

Componentes del "costo" de ejecución de una consulta:

- Costo de acceso a almacenamiento secundario → acceder al bloque de datos que reside en disco.
- Costo de cómputo → Costo de realizar operaciones sobre memoria RAM
- Costo de comunicación → Costo de enviar la consulta y los resultados (si es un Sistema Distribuido)

Optimización de Consultas



Optimización de consultas

Optimización Lógica

- Expresiones equivalentes → Algebra relacional
  - existe una secuencia de resolución
  - se puede encontrar una expresión más eficiente que otra.

Optimización de consultas

Selección: Personas del género masculino que sean solteros

- $\sigma_{\text{Genero='M'} \wedge \text{ECivil='Soltero'}}$ (Persona) →
  - Se aplican 2 condiciones a 7 tuplas
- $\sigma_{\text{Genero='M'}}(\sigma_{\text{ECivil='Soltero'}}$ (Persona)) →
  - Se aplica 1 condición a 7 tuplas y 1 condición a 1 tupla

- Conclusión: el caso 2 es mejor, por lo que conviene realizar la selección lo antes posible

DNI	Nombre	Genero	ECivil
22456980	Josefina	F	Casado
32456789	Juan	M	Casado
24567876	María	F	Casado
21345654	Roberto	M	Soltero
20987654	Alfredo	M	Casado
20897656	Fernanda	F	Casado
21345678	Raul	M	Casado

Optimización de consultas

DNI	Nombre	IdCiudad
22456980	Josefina	1
32456789	Juan	2
24567876	María	3
21345654	Roberto	1
20987654	Alfredo	2
20897656	Fernanda	3
21345678	Raul	1

IdCiudad	Nombre
1	Junín
2	Pergamino
3	La Plata

- Proyección: DNI de las personas que vivan en la ciudad de Junín
  1. π_{DNI} (Persona |x| $\sigma_{Nombre='Junín'}$ (Ciudad))
  2. π_{DNI} ($\pi_{DNI,IdCiudad}$ (Persona) |x| $\pi_{IdCiudad}$ ($\sigma_{Nombre='Junín'}$ (Ciudad))

- Conclusión: el caso 2 es mejor, por lo que conviene realizar la proyección para disminuir la cantidad de información que se almacena en buffers de memoria.

Optimización de consultas

La conclusión anterior respecto a la proyección se puede aplicar a otras operaciones binarias:

- Union,
- Intersección,
- Diferencia

Optimización de consultas

Algunos valores:

- CT tabla (cantidad de tuplas de la **tabla**)
- CB tabla (cantidad de bytes que ocupa cada tupla de la **tabla**)
- CV (a, tabla) (cantidad de ocurrencias de distintas del atributo **a** en la **tabla**)

Costo en bytes selección: $\sigma_{(at = \text{“valor”})}$ (Tabla)

- (CT tabla / CV (at, tabla)) * CB tabla

Costo en bytes proyección: $\pi_{at1, at2, .. atn}$ (Tabla)

- (CB at1 + CB at2 + .. + CB atn) * CT tabla

Costo en bytes producto cartesiano: T1 X T2

- (CT t1 * CT t2) * (CB t1 + CB t2)

Optimización de consultas

Costo producto natural: T1 |X| T2

- Sin atributos en común → T1 X T2
- Con atributo "a" en común, donde: a es PK en T1 y FK en T2.
  - T1 |X| T2 → un fila de T1 con muchas de T2.
    - Clave secundaria.
  - T2 |X| T1 → un fila de T2 con una de T1.
    - Clave primaria.
- Con atributo "a" en común:
  - (CT t1 * CT t2) / MAX(CV (a, t1) , CV (a, t2))

Optimización de Consultas

Dado el siguiente modelo relacional:

- PRODUCTOS (**idproducto**, código, descripción, precio, idvendedor)
  - FK (vendedor, VENDEDORES) la clave foránea no permite nulos
- VENDEDORES (**idvendedor**, nombre_vendedor, sucursal)

Ejemplo 1: la siguiente **consulta**: "Listar los datos de los productos que vende la sucursal de JUNIN"

- **SELECT p.producto, p.descripción, p.precio, v.nombre_vendedor**
- **FROM PRODUCTOS p, VENDEDORES v**
- **WHERE p.idvendedor = v.idvendedor and v.sucursal = ´JUNIN´;**

$$\Pi_{\text{p.producto, p.descripción, p.precio, v.nombre\_vendedor}}(\sigma_{\text{p.idvendedor = v.idvendedor } \wedge \text{ sucursal = 'JUNIN'}}(\text{PRODUCTOS X VENDEDORES}))$$

- Sabiendo que: - CT(productos) = 7000 - CT(vendedores) = 300
  - CV (sucursal = 'JUNIN', vendedores) = 10
  - 1000 productos de vendedores de JUNIN

Optmizacion de Consultas

Árbol Inicial

Optimización de Consultas

$$\pi_{p.producto,\ p.descripción,\ p.precio,\ v.nombre\_vendedor}(\sigma_{p.idvendedor = v.idvendedor\ \wedge\ sucursal = \text{´JUNIN´}}(PRODUCTOS \times VENDEDORES))$$

Plan	Pasos	Operación	Cantidad de lecturas	Costo de acceso	Cantidad de Tuplas	Costo Total
A	1	Producto Cartesiano	7.000 + 300	7.300	2.100.000	7.300
	2	σ (A1) p.idvendedor = v.idvendedor	2.100.000	2.100.000	7.000	2.107.300
	3	σ (A2) sucursal= ´JUNIN´	7.000	7.000	1.000	2.114.300

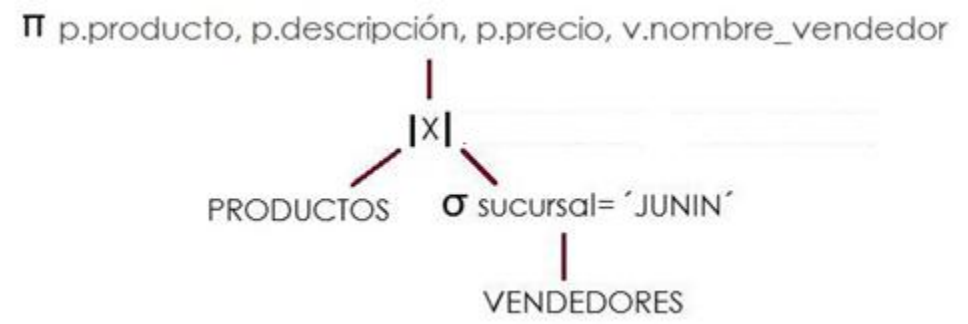
Optimización de Consultas

$$\pi_{p.producto,\ p.descripción,\ p.precio,\ v.nombre\_vendedor}(\sigma_{p.id.vendedor = v.id.vendedor}(\text{PRODUCTOS} \times \sigma_{sucursal = \text{´JUNIN´}}(\text{VENDEDORES})))$$

Plan	Paso	Operación	Cantidad de lecturas	Costo de acceso	Cantidad de Tuplas	Costo Total
B	1	$\sigma_{sucursal = \text{´JUNIN´}}$ (vendedores)	300	300	10	300
	2	B1 x PRODUCTOS	10 + 7.000	7.010	70.000	7.310
	3	σ(B2) p.idvendedor = v.idvendedor	70.000	70.000	1.000	77.310

Optimización de Consultas

$$\pi_{p.producto,\ p.descripción,\ p.precio,\ v.nombre\_vendedor}(PRODUCTOS \mid x \mid (\sigma_{sucursal = \text{´JUNIN´}} VENDEDORES))$$



Plan	Nivel	Operación	Cantidad de lecturas	Costo de acceso	Cantidad de Tuplas	Costo Total
C	1	σ (vendedores) sucursal = ´JUNIN´	300	300	10	300
	2	PRODUCTOS Ix I C1	10 + 7.000	7.010	1.000	7.310

Optimizacion de Consultas

CONSULTA ORIGINAL:

```
SELECT p.producto, p.descripción, p.precio, v.nombre_vendedor
FROM PRODUCTOS p, VENDEDORES v
WHERE p.idvendedor = v.idvendedor and v.sucursal = ´JUNIN´;
```

CONSULTA MÁS EFICIENTE:

```
SELECT p.codigo, p.descripcion, p.precio, v.nombre_vendedor
FROM Productos p NATURAL JOIN (SELECT Idvendedor,nombre_vendedor
                               FROM Vendedores
                               WHERE sucursal = 'JUNIN')  v
```

EFICIENCIA
vs
LEGIBILIDAD

CONSULTA MÁS LEGIBLE:

```
SELECT p.codigo, p.descripcion, p.precio, v.nombre_vendedor
FROM Productos p NATURAL JOIN Vendedores v
WHERE  v.sucursal = 'JUNIN'
```

Diseño de Bases de Dato

CLASE 6

Seguridad e Integridad de datos: Agenda

Transacciones	• Propidades • Estados
Transacciones monusuarias	• Atomicidad • Protocolos
Transacciones centralizadas	• Aislamiento • Consistencia • Durabilidad

Transacciones

Transacción: colección de operaciones que forman una única unidad lógica de trabajo.

- Propiedades **ACID**
  - **Atomicidad:** todas las operaciones de la transacción se ejecutan o no lo hacen ninguna de ellas
  - **Consistencia:** la ejecución aislada de la transacción conserva la consistencia de la BD
  - **Aislamiento (isolation):** cada transacción ignora el resto de las transacciones que se ejecutan concurrentemente en el sistema, actúa c/u como única.
  - **Durabilidad:** una transacción terminada con éxito realiza cambios permanentes en la BD, incluso si hay fallos en el sistema

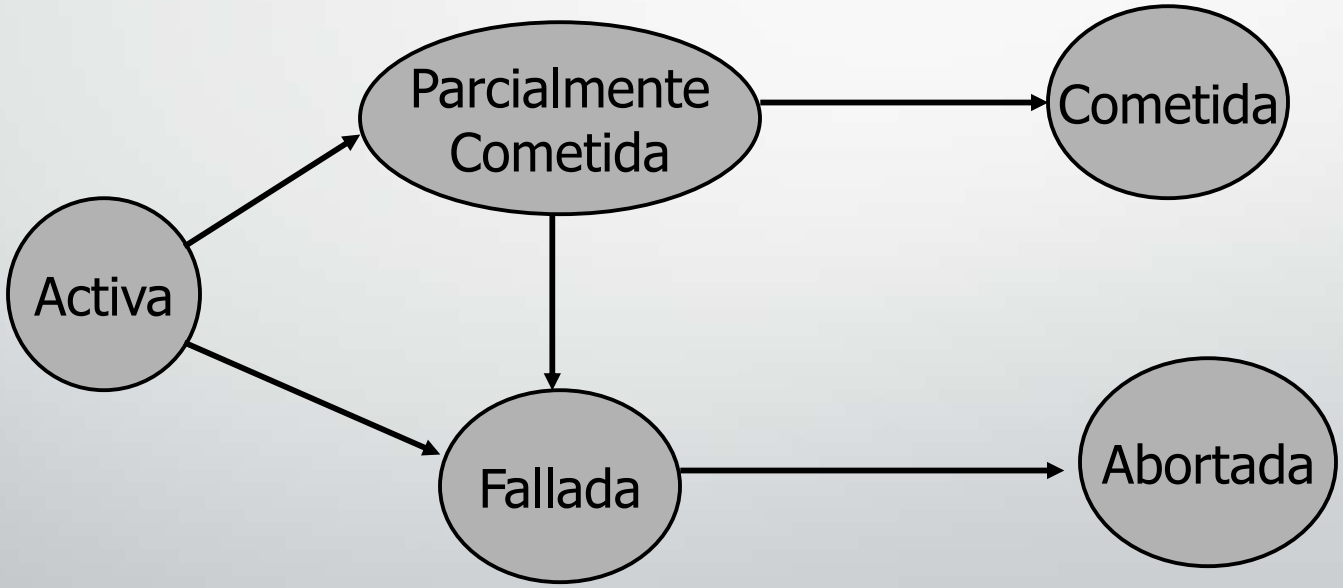
Transacciones

Estados de una transacción

- Activa: estado inicial, estado normal durante la ejecución.
- Parcialmente Cometida: después de ejecutarse la última instrucción
- Fallada: luego de descubrir que no puede seguir la ejecución normal
- Abortada: después de haber retrocedido la transacción y restablecido la BD al estado anterior al comienzo de la transacción.
- Cometida: tras completarse con éxito.

Transacciones

Diagrama de estado de una transacción



Transacciones

Modelo de transacción

- READ (A, a1)
- a1 := a1 – 100;
- WRITE(A, a1)
- READ (B, b1)
- b1 := b1 + 100;
- WRITE(B, b1)

Diferencia entre READ, WRITE y INPUT, OUTPUT.

Uso de transacciones:

- En sistemas monousuario
- En sistemas concurrentes
- En sistemas distribuidos

Transacciones

1.READ (A, a1)
2.a1 := a1 – 100;
3.WRITE(A, a1)
4.READ (B, b1)
5.b1 := b1 + 100;
6.WRITE(B, b1)

BD A = 1000 900 (3)
B = 2000 2100(6)

Memoria local
A = 1000 (1) 900 (2)
B = 2000 (4) 2100 (5)

FALLO LUEGO DE 3 Y ANTES DE 6 ? QUE PASA?

Que hacer luego de un fallo?

- Re-ejecutar la transacción fallada → no sirve
- Dejar el estado de la BD como está → no sirve

Problema: modificar la BD sin seguridad que la transacción se va a cometer.

- Solución: indicar las modificaciones

Soluciones

- Registro Historico
- Doble paginación

Registro Histórico

Bitácora

- secuencia de actividades realizadas sobre la BD.
- Contenido de la bitácora
  - <T iniciada>
  - <T, E, Va, Vn>
    - Identificador de la transacción
    - Identificador del elemento de datos
    - Valor anterior
    - Valor nuevo
  - <T Commit>
  - <T Abort>

Registro Historico

Las operaciones sobre la BD deben almacenarse luego de guardar en disco el contenido de la Bitácora

Dos técnicas de bitácora

- Modificación diferida de la BD
- Modificación inmediata de la BD

Registro Histórico

Modificación diferida

- Las operaciones write se aplazan hasta que la transacción esté parcialmente cometida, en ese momento se actualiza la bitácora y la BD

Registro Historico

1. READ (A, a1)
2. a1 := a1 – 100;
3. READ (B, b1)
4. b1 := b1 + 100;
5. WRITE(A, a1)
6. WRITE(B, b1)

Bitacora

1. <T start> 2 no produce efecto
5. <T, A, 900 > 3, 4 sin efecto
6. <T, B, 2100>
7. <T commit>

Memoria RAM

1. A = 1000
2. A = 900
3. B = 2000
4. B = 2100

Plan1 1, 2, 3, 4, 5, 6, 7 , 8 , 9 fallo
Recupera del Fallo y no habria que hacer nada???

Plan2 1,2,3,4,5,6,7, 8 fallo
Recupera del Fallo y que hacemos???

Plan3 1,2,3, 4, 5, 6, 7 fallo
Recupera del fallo y no habria que hacer nada???

Planotro fallo antes de 7
Base de datos bien.

Base de Datos

A = 1000
900 (8)
B = 2000
2100 (9)

Base de Datos

A = 1000
900 (8)
B = 2000

Base de Datos

A = 1000

B = 2000

Registro Histórico

Dada la siguiente transacción

- < To Start >
- < To, A, 900 >
- < To, B, 2100 >
- < To Commit >

Recién con To parcialmente cometida, entonces se actualiza la BD.

- No se necesita valor viejo, se modifica la BD al final de la transacción o no se modifica.

Ante un fallo, y luego de recuperarse:

- REDO (Ti), para todo Ti que tenga un Start y un Commit en la Bitácora.
- Si no tiene Commit entonces se ignora, dado que no llegó a hacer algo en la BD.

Registro Histórico

Modificación inmediata:

- La actualización de la BD se realiza mientras la transacción está activa y se va ejecutando.
- Se necesita el valor viejo, pues los cambios se fueron efectuando.
- Ante un fallo, y luego de recuperarse:
  - REDO(Ti), para todo Ti que tenga un Start y un Commit en la Bitácora.
  - UNDO(Ti), para todo Ti que tenga un Start y no un Commit.

Registro Histórico

Transacción:

- Condición de idempotencia.

Buffers de Bitácora

- Grabar en disco c/registro de bitácora insume gran costo de tiempo → se utilizan buffer, como proceder?
  - Transacción está parcialmente cometida después de grabar en memoria no volátil el Commit en la Bitácora.
  - Un Commit en la bitácora en memoria no volátil, implica que todos los registros anteriores de esa transacción ya están en memoria no volátil.
  - **Siempre** graba primero la Bitácora y luego la BD.

Registro Histórico

Puntos de verificación:

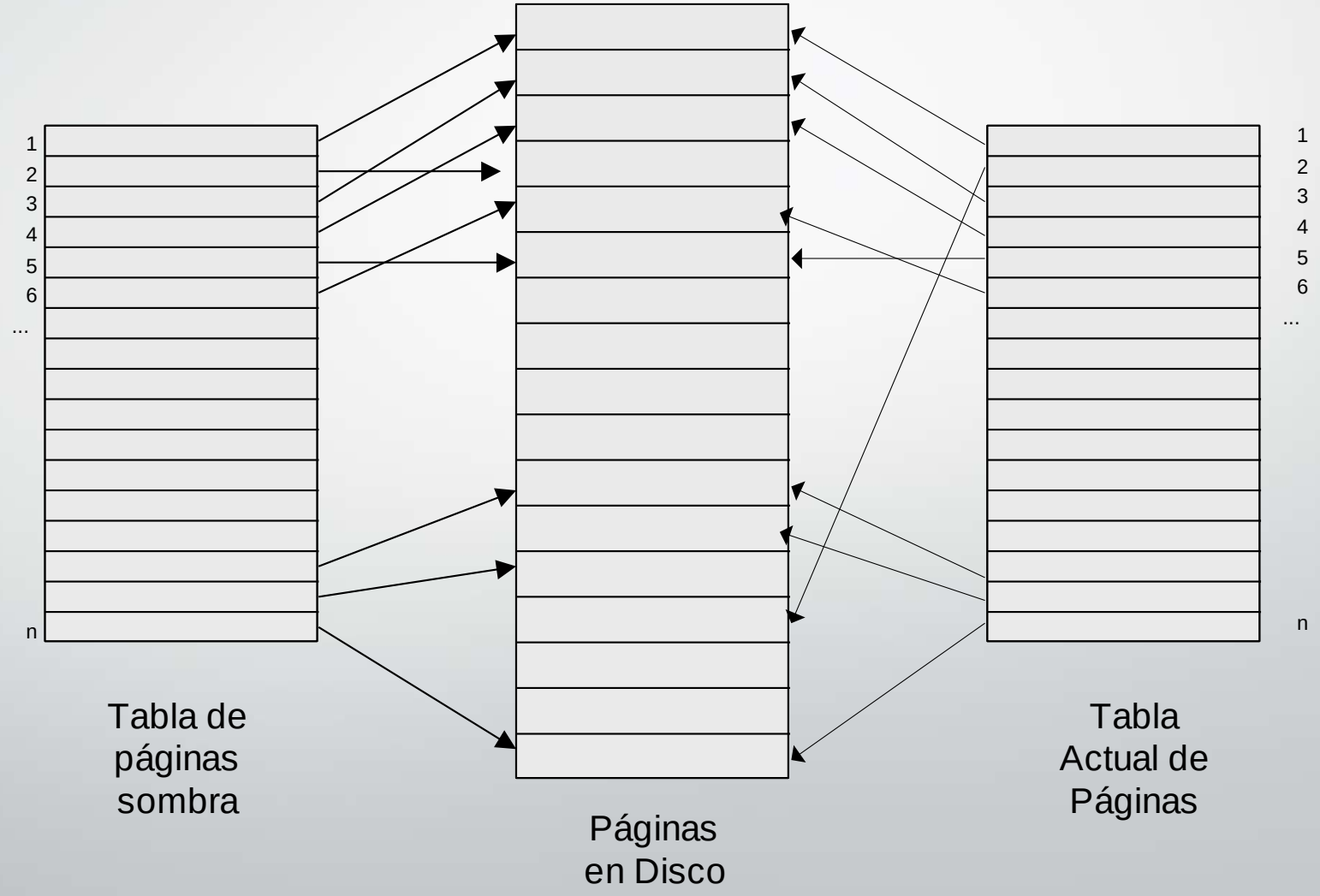
- Ante un fallo, que hacer
  - REDO, UNDO: según el caso
- Revisar la bitácora:
  - Desde el comienzo?: probablemente gran porcentaje esté correcto y terminado.
  - Lleva mucho tiempo.
- Checkpoints (monousario)
  - Se agregan periódicamente indicando desde allí hacia atrás todo OK.
  - Periodicidad?

Doble Paginación

Paginación en la sombra:

- Ventaja: menos accesos a disco
- Desventaja: complicada en un ambiente concurrente/distribuido.
- N páginas equivalente a páginas del SO.
  - Tabla de páginas actual
  - Tabla de páginas sombra

Doble Paginación



Doble Paginación

Ejecución de la operación *escribir*

- Ejecutar **entrada**(X) si página i-ésima no está todavía en memoria principal.
- Si es la primer escritura sobre la página i-ésima, modificar la tabla actual de páginas así:
  - Encontrar una página en el disco no utilizada
  - Indicar que a partir de ahora está ocupada
  - Modificar la tabla actual de página indicando que la i-ésima entrada ahora apunta a la nueva página

Doble Paginación

En caso de fallo y luego de la recuperación

- Copia la tabla de páginas sombra en memoria principal.
- Abort automáticos, se tienen la dirección de la página anterior sin las modificaciones.

Recuperación en caso de Fallo

Ventajas:

- Elimina la sobrecarga de escrituras del log
- Recuperación más rápida (no existe el REDO o UNDO).

Desventajas:

- Sobrecarga en el compromiso: la técnica de paginación es por cada transacción.
- Fragmentación de datos: cambia la ubicación de los datos continuamente.
- Garbage Collector: ante un fallo queda una página que no es mas referenciada.

Entornos Concurrentes

Entorno centralizado

- Varias transacciones ejecutándose simultáneamente compartiendo recursos.
- Deben evitarse los mismos problemas de consistencia de datos
- Transacciones correctas, en ambientes concurrente pueden llevar a fallos

Entornos Concurrentes

Seriabilidad

- Garantiza la consistencia de la BD

```
To Read (a)
   a:= a – 50
   Write (a)
   Read ( b )
   b := b + 50
   Write ( b )
```

```
T1 Read (a)
   temp := a * 0,1
   a:= a – temp
   Write (a)
   Read ( b )
   b := b + temp
   Write ( b )
```

- Resolver To, T1 o T1, To se respeta A+B
- Ahora bien To T1 <> T1 To

Entornos concurrentes

T0 → T 1
BD A = 1000
B = 2000 3000

T0 → transfiere 50

BD A = 950
B = 2050

T1 → transfiere el 10%

BD A = 855
B = 2145 3000

T1 → T 0
BD A = 1000
B = 2000 3000

T1 → transfiere el 10%

BD A = 900
B = 2100

T0 → transfiere 50

BD A = 850
B = 2150 3000

A + B se respeta!

Entornos Concurrentes

Planificación: secuencia de ejecución de transacciones

- Involucra todas las instrucciones de las transacciones
- Conservan el orden de ejecución de las mismas
- Un conjunto de m transacciones generan m! planificaciones en serie
- La ejecución concurrente no necesita una planificación en serie.

Entornos concurrentes

1.	READ(A)	
2.	A := A – 50	
3.	WRITE(A)	
4.		READ(A)
5.		TEMP := A * 0.1
6.		A := A – TEMP
7.		WRITE(A)
8.	READ(B)	
9.	B := B + 50	
10.	WRITE(B)	
11.		READ(B)
12.		B := B + TEMP
13.		WRITE(B)

A + B se conserva

BD A = 1000 3. 950 7. 945
B =2000 10.2050 13. 2145

To 1, A =1000
2 A = 950
8 B = 2000
9 B – 2050

T1 4 A = 950
5 temp = 95
6 A = 945
11 B = 2050
12 B = 2145

1.	READ(A)	
2.	A := A – 50	
3.		READ(A)
4.		TEMP := A * 0.1
5.		A := A – TEMP
6.		WRITE(A)
7.		READ(B)
8.	WRITE(A)	
9.	READ(B)	
10.	B := B + 50	
11.	WRITE(B)	
12.		B := B + TEMP
13.		WRITE(B)

A + B no se conserva

BD A = 1000 6. 900 8.
B =2000 11.2050 13.

To 1, A =1000
2 A = 950
9 B = 2000
10 B = 2050

T1 3 A = 1000
4 temp = 100
5 A = 900
7 B = 2000
12 B = 2100

Entornos Concurrentes

Conclusiones

- El programa debe conservar la consistencia
- La inconsistencia temporal puede ser causa de inconsistencia en planificaciones en paralelo
- Una planificación concurrente debe equivaler a una planificación en serie
- Solo las instrucciones READ y WRITE son importantes y deben considerarse.

Entornos Concurrentes

Conflicto en planificaciones serializables

- I1, I2 instrucciones de T1 y T2
  - Si operan sobre datos distintos. NO hay conflicto.
  - Si operan sobre el mismo dato
    - I1 = READ(Q) = I2, no importa el orden de ejecución
    - I1 = READ(Q), I2 = WRITE(Q) depende del orden de ejecución (I1 leerá valores distintos)
    - I1 = WRITE(Q), I2 = READ(Q) depende del orden de ejecución (I2 leerá valores distintos)
    - I1 = WRITE(Q) = I2, depende el estado final de la BD
- I1, I2 está en conflicto si actúan sobre el mismo dato y al menos una es un write. Ejemplos.

Entornos Concurrentes

Definiciones

- **Una Planificación S se transforma en una S´ mediante intercambios de instrucciones no conflictivas, entonces S y S´ son *equivalentes en cuanto a conflictos.***
- Esto significa que si
  - S' es consistente, S también lo será
  - S' es inconsistente, S también será inconsistente
- **S´ es serializable en conflictos si existe S/ son equivalentes en cuanto a conflictos y S es una planificación serie.**

Entornos concurrentes

1.	READ(A)	
2.	A := A – 50	
3.	WRITE(A)	
4.		READ(A)
5.		TEMP := A * 0.1
6.		A := A – TEMP
7.		WRITE(A)
8.	READ(B)	
9.	B := B + 50	
10.	WRITE(B)	
11.		READ(B)
12.		B := B + TEMP
13.		WRITE(B)

A + B se conserva

1.	READ(A)	
2.	A := A – 50	
3.		READ(A)
4.		TEMP := A * 0.1
5.		A := A – TEMP
6.		WRITE(A)
7.		READ(B)
8.	WRITE(A)	
9.	READ(B)	
10.	B := B + 50	
11.	WRITE(B)	
12.		B := B + TEMP
13.		WRITE(B)

A + B no se conserva

Control de Concurrencia

Métodos de control de concurrencia

- Bloqueo
- Basado en hora de entrada

Control de Concurrencia

Bloqueo

- Compartido *Lock_c*(dato)(solo lectura)
- Exclusivo *Lock_e*(dato) (lectura/escritura)
- Las transacciones piden lo que necesitan.
- Los bloqueos pueden ser compatibles y existir simultáneamente (compartidos)

Una transacción debe:

- Obtener el dato (si está libre, o compartido y solicita compartido)
- Esperar (otro caso)
- Usar el dato
- Liberarlo.

T1 a → b

1. Lock_e(a)
3. Read (a)
4. a := a – 50
5. Write (a)
6. Unlock (a)

10. Lock_e(b)
13. Read (b)
14. b := b + 50
15. Write (b)
16. Unlock (b)

T2 a + b

2. Lock_c(a)
7. Read (a)
8. Unlock (a)
9. Lock_c(b)
11. Read (b)
12. Unlock (b)
17. informar (a+b)

BD A = 1000 1.exc 5. 950 6. libera 6'.comp 8. libera
B = 2000 9.comp 12. libera 12'.exclusivo 15. 2050 16. libera

T1 3. A = 1000
4. A = 950
10. bloquea!
13. B = 2000
14 B = 2050

T2 2. Bloquea!
7. A =950
11. B = 2000

17??????????????????

T1 → T2 o T2 → T1 en serie, no genera problemas

Control de Concurrencia

Si se ejecutan en orden verde, azul, celeste

Que pasa con los resultados

Se deben llevar los bloqueos de las transacciones al comienzo.

Control de Concurrencia

Deadlock

- situación en la que una transacción espera un recurso de otra y viceversa

Conclusiones:

- Si lo datos se liberan pronto → se evitan posibles deadlock
- Si los datos se mantienen bloqueados → se evita inconsistencia.

Control de Concurrencia

Protocolos de bloqueo

- Dos fases
  - Requiere que las transacciones hagan bloqueos en dos fases:
    - Crecimiento: se obtienen datos
    - Decrecimiento: se liberan los datos
  - Garantiza seriabilidad en conflictos, pero no evita situaciones de deadlock.
  - Como se consideran operaciones
    - Fase crecimiento: se piden bloqueos en orden: compartido, exclusivo
    - Fase decrecimiento: se liberan datos o se pasa de exclusivo a compartido.

Control de Concurrencia

Protocolo basado en hora de entrada

- El orden de ejecución se determina por adelantado, no depende de quien llega primero
- C/transacción recibe una HDE
  - Hora del servidor
  - Un contador
- Si HDE(Ti) < HDE(Tj), Ti es anterior
- C/Dato
  - Hora en que se ejecutó el último WRITE
  - Hora en que se ejecutó el último READ
  - Las operaciones READ y WRITE que pueden entrar en conflicto se ejecutan y eventualmente fallan por HDE.

Control de Concurrencia

Algoritmo de ejecución:

- **Ti Solicita READ(Q)**
  - HDE(Ti) < HW(Q): rechazo (solicita un dato que fue escrito por una transacción posterior)
  - HDE(Ti) ≥ HW(Q): ejecuta y se establece HR(Q)=Max{HDE(Ti), HR(Ti)}
- **Ti solicita WRITE(Q)**
  - HDE(Ti) < HR(Q): rechazo (Q fue utilizado por otra transaccion anteriomente y suposu que no cambiaba)
  - HDE(Ti) < HW(Q): rechazo (se intenta escribir un valor viejo, obsoleto)
  - HDE(Ti) > [HW(Q) y HR(Q)]: ejecuta y HW(Q) se establece con HDE(Ti).
- **Si Ti falla, y se rechaza entonces puede recomenzar con una nueva hora de entrada.**

Control de Concurrencia

Casos de Concurrencia. Granularidad

- A registros caso más normal
- Otros casos
  - BD completa
  - Áreas
  - Tablas

Otras operaciones conflictivas

- Delete(Q) requiere un uso completo del registro
- Insert(Q) el dato permanece bloqueado hasta la operación finalice.

Registro Histórico en entornos concurrentes

Consideraciones del protocolo basado en bitácora

- Existe un único buffer de datos compartidos y uno para la bitácora
- C/transacción tiene un área donde lleva sus datos
- El retroceso de una transacción puede llevar al retroceso de otras transacciones

Retroceso en cascada

- Falla una transacción → pueden llevar a abortar otras
- Puede llevar a deshacer gran cantidad de trabajo.

Registro Histórico en entornos concurrentes

Durabilidad

- Puede ocurrir que falle Ti, y que Tj deba retrocederse, pero que Tj ya terminó. Como actuar?
- Protocolo de bloqueo de dos fases: los bloqueos exclusivos deben conservarse hasta que Ti termine.
- HDE, agrega un bit, para escribir el dato, además de lo analizado, revisar el bit si está en o proceder, si está en 1 la transacción anterior no termino, esperar….

Registro Histórico en entornos concurrentes

Bitácora

- Similar sistemas monousuarios
- Como proceder con checkpoint
  - Colocarlo cuando ninguna transacción esté activa. Puede que no exista el momento.
  - Checkpoint<L> L lista de transacciones activa al momento del checkpoint.
- Ante un fallo
  - UNDO y REDO según el caso.
  - Debemos buscar antes del Checkpoint solo aquellas transacciones que estén en la lista.