

Unidad Temática 1

Introducción

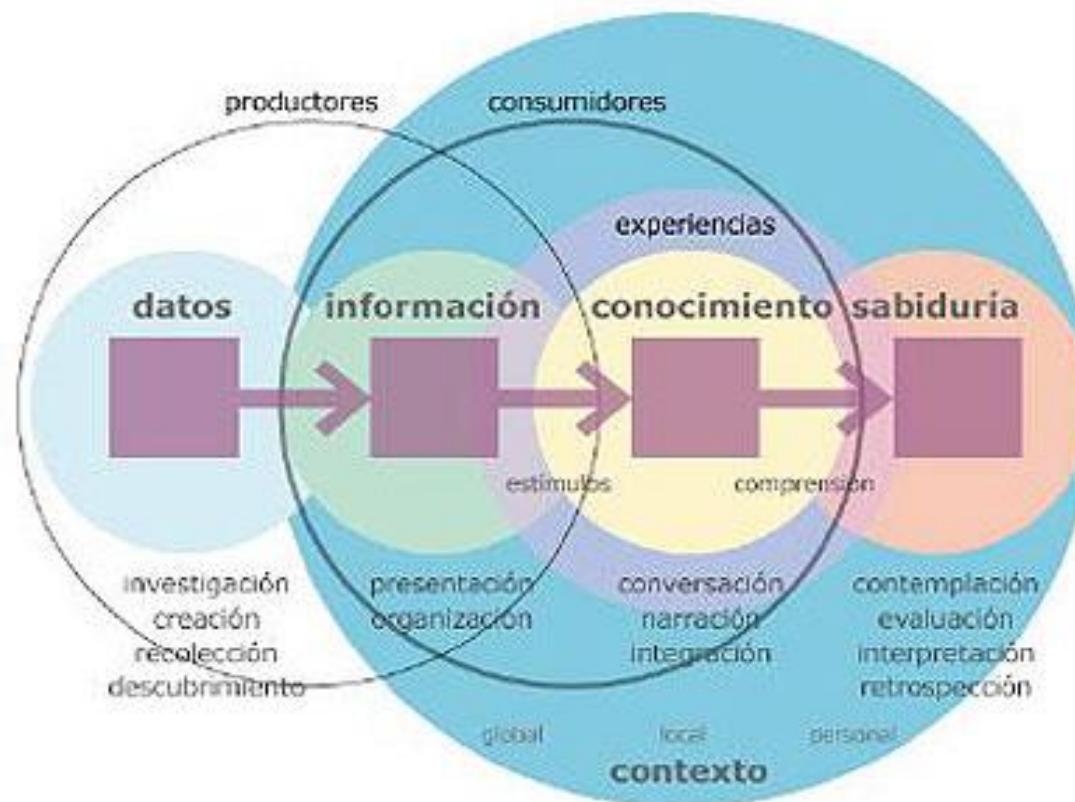
Contexto de las bases de datos

Panorámica de los conceptos asociados a bases de datos



De los datos a la sabiduría

«El proceso que lleva al entendimiento continuo del entorno, arranca en los datos y finaliza en la sabiduría, pasando por la información y el conocimiento»



<https://chau201411700812287.wordpress.com/2014/04/01/de-los-datos-a-la-sabiduria/>

Objetivo del procesamiento de datos

Sistemas de apoyo a la toma de decisiones

El pasado

Decisiones lentas, basadas sólo en la experiencia, prediciendo el futuro en base a la intuición.



El presente

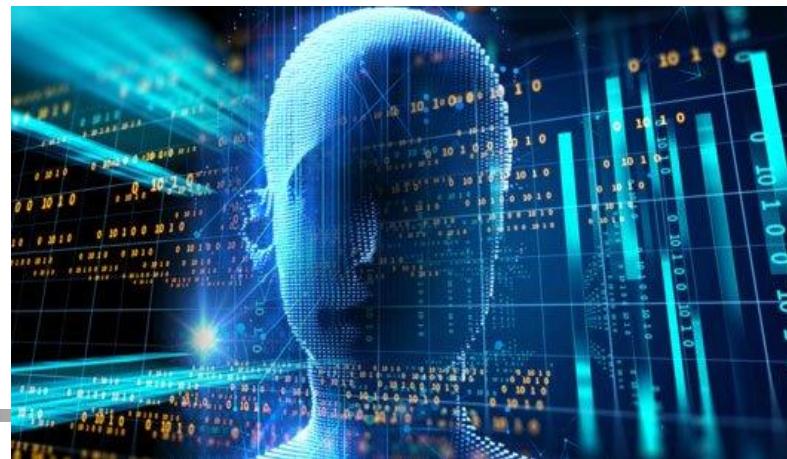
Decisiones rápidas y oportunas, tomadas sobre cimientos firmes, prediciendo el futuro en base a los datos de la infósfera.

- Business Intelligence
- Business analytics
- Data mining
- Data science



Sistemas de apoyo a la toma de decisiones

*“Quienes poseen
los datos poseen el
futuro”*

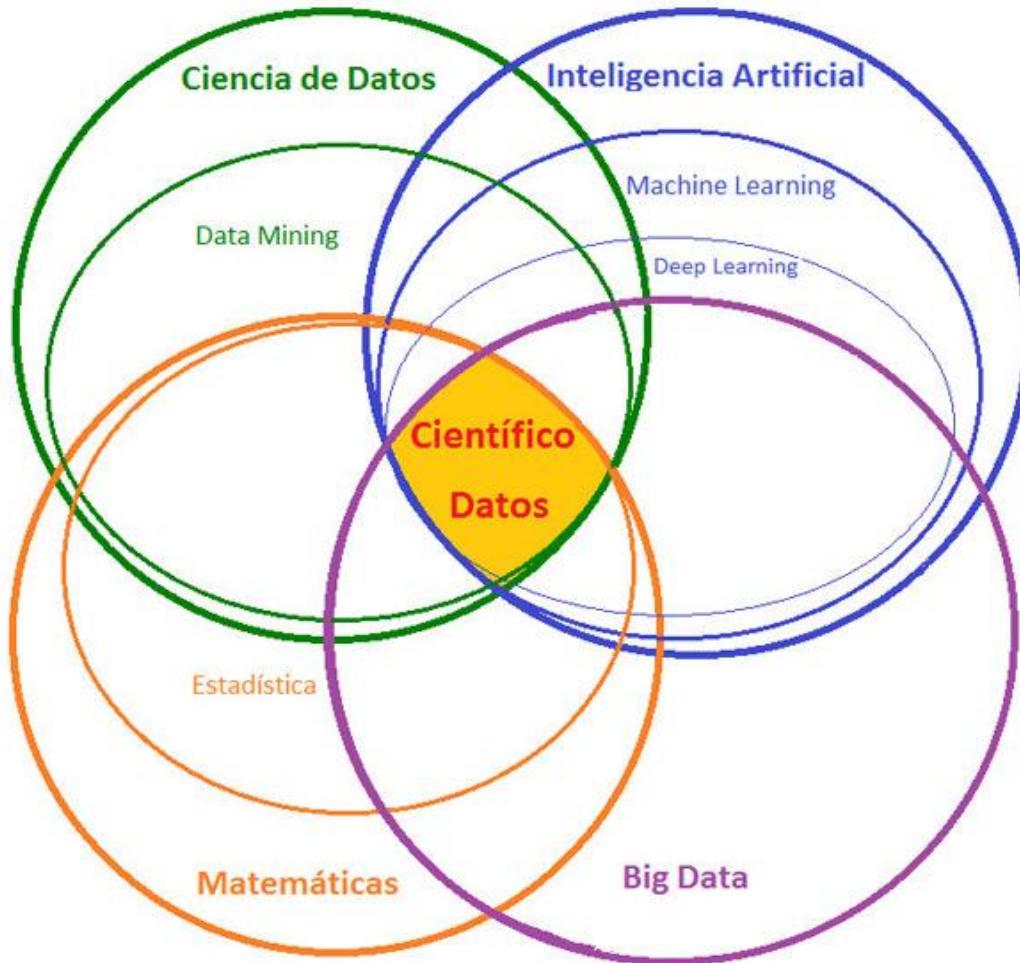


A large central blue word "DATA SCIENCE" is surrounded by various white text labels representing different fields and applications of data science, such as machine learning, big data, data mining, and information technology.

Surrounding words include:

- MACHINE LEARNING
- VISION
- INFORMATION
- RESEARCH
- PROGRAMMABILITY
- KDD
- COMPUTING
- BIG DATA
- WEB DEV
- STRATEGY
- WORLDWIDE
- VISUALIZATION
- PRICING
- SEGMENTATION
- SOCIAL NETWORKS
- SOCIAL
- MOBILE
- DATA
- ENGINEERING
- PLANNING
- MEDIA
- STATISTICS
- TARGET
- INFORMATION
- DIGITAL
- SOCIAL
- NETWORKS
- WEB SERVICES
- MATHS
- PATTERN
- ANALYTICS
- PREDICTIVE
- PROGRAM
- CONTENT
- EVENTS
- PROGRAMMING
- DATA MINING
- WEB MARKETING
- CONSUMER DEMAND
- NETS
- COMPUTER
- COMMUNICATION
- FASHION
- MARKETING
- MODELS
- INFORMATION
- TECHNOLOGY
- PROCESSING
- ORGANIZATION
- HUMAN
- ANALYST
- DATA
- SCIENCE
- DETCTION
- SOCIAL MEDIA
- SERVICES
- PROJECTS
- BIG DATA
- INFORMATION

La Ciencia de Datos y su relación con las otras áreas de ciencias de la computación



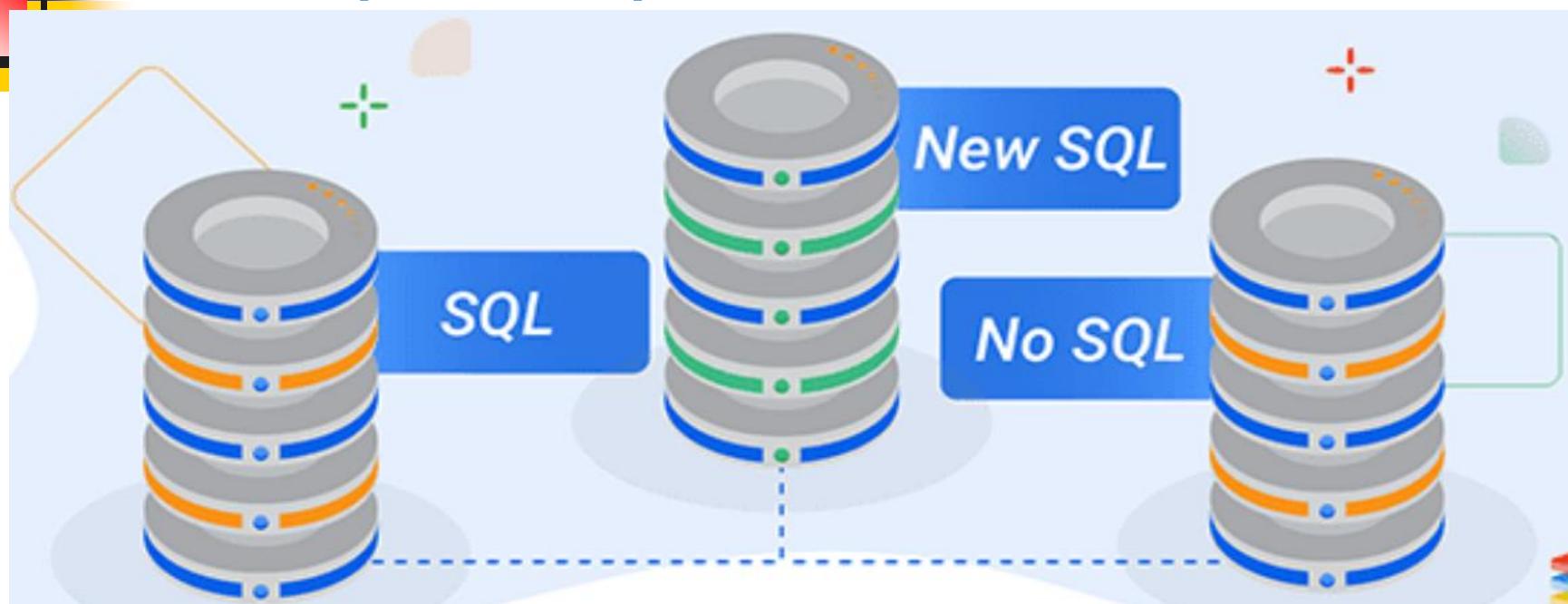
La **ciencia de datos** es una disciplina que hace uso de diferentes métodos, tecnologías analíticas avanzadas y principios científicos para extraer información valiosa de los datos para la toma de decisiones

La **minería de datos** es el conjunto de técnicas y tecnologías que permiten explorar grandes **bases de datos**, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, correlaciones, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto; haciendo uso de la estadística, inteligencia artificial, etc.

Arquitectura de datos

La arquitectura de datos es el diseño estructurado y organizado de los sistemas de almacenamiento, gestión y acceso a los datos en una organización. Define cómo se capturan, almacenan, procesan y utilizan los datos para satisfacer las necesidades de la organización y sus usuarios. La arquitectura de datos incluye la definición de los tipos de datos, su estructura y formato, los sistemas y tecnologías utilizados para gestionar los datos, y las políticas y procedimientos para garantizar la calidad, seguridad e integridad de los datos.

Arquitecturas para sistemas transaccionales

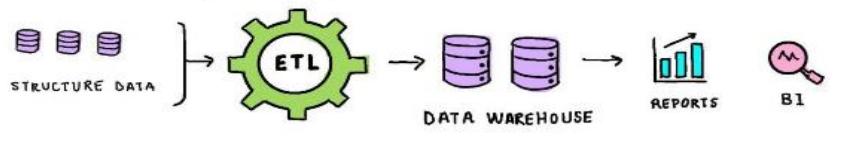


<https://www.xenonstack.com/blog/sql-vs-nosql-vs-newsql>

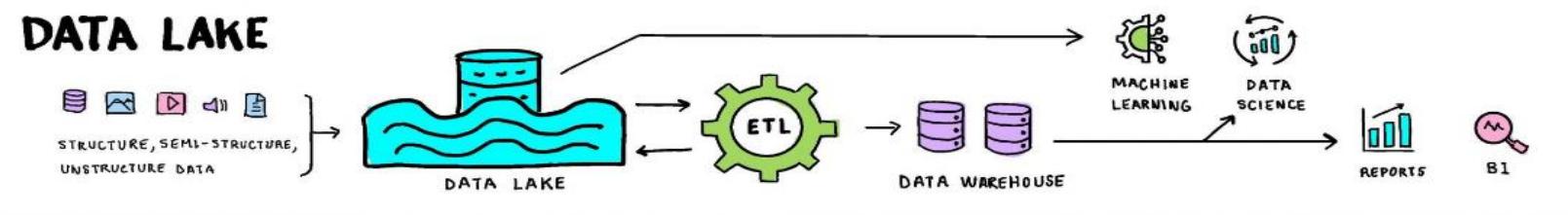
Arquitectura de datos

Arquitecturas para sistemas de apoyo a la toma de decisiones

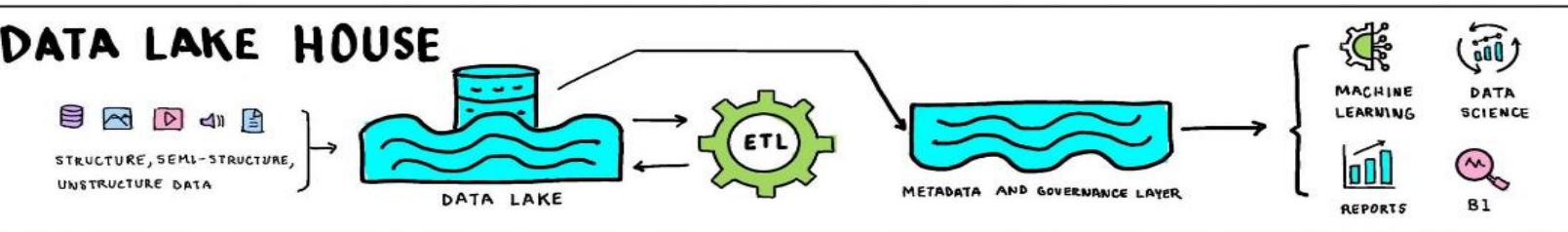
DATA WAREHOUSE



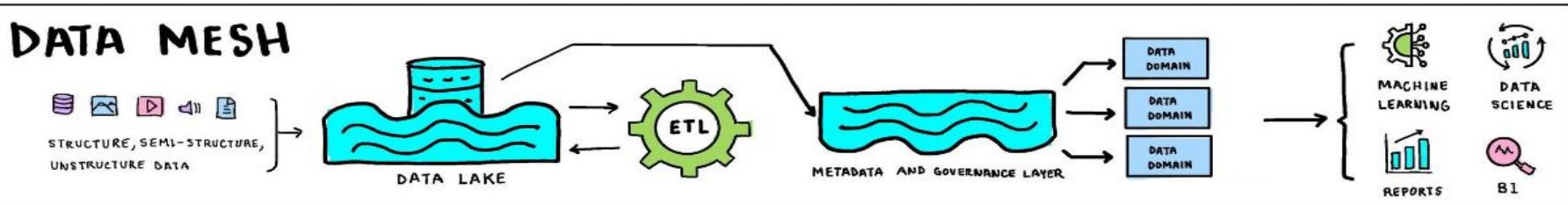
DATA LAKE



DATA LAKE HOUSE



DATA MESH

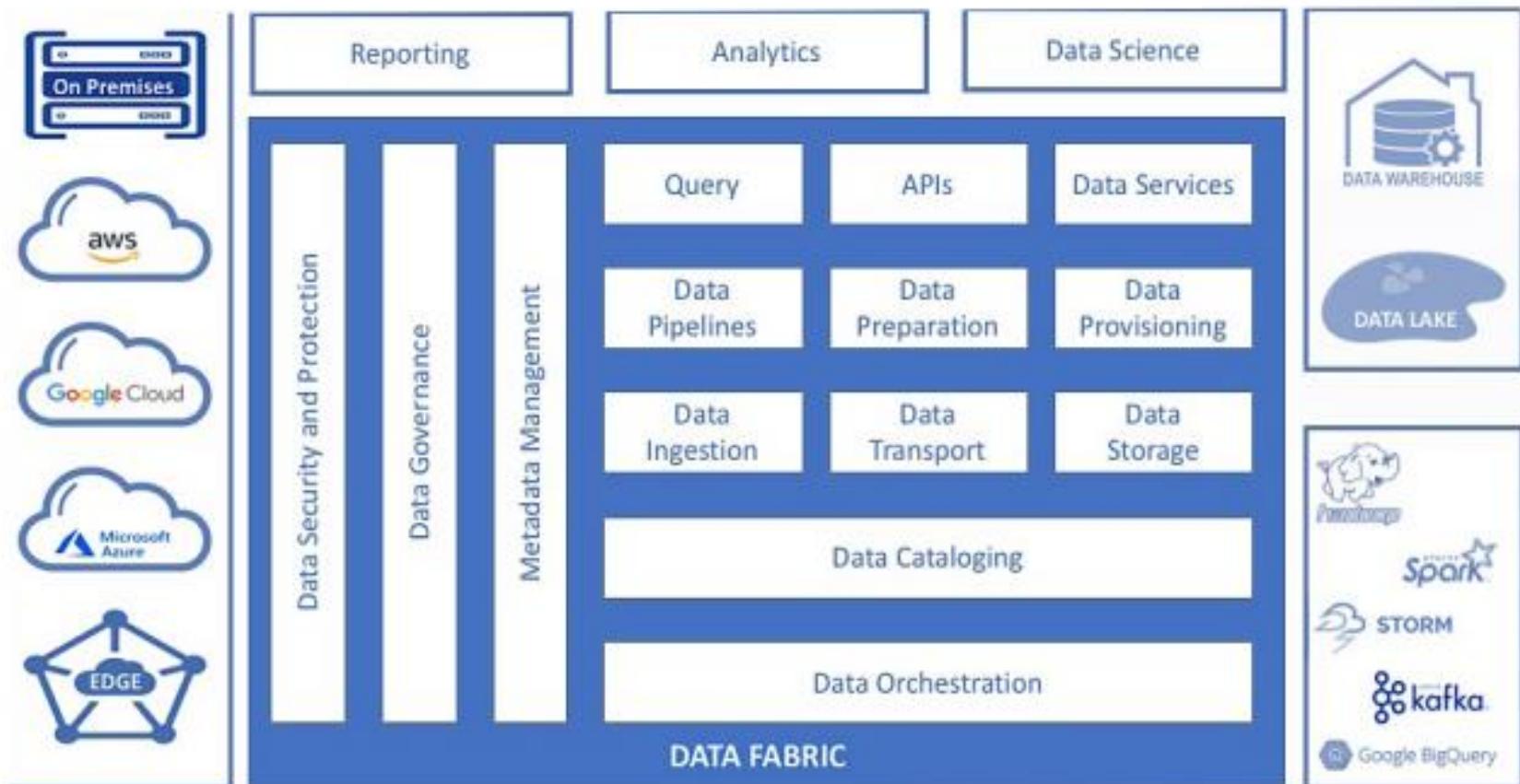


<https://hectorv.com/category/datawarehouse/>

Arquitectura de datos

Data Fabric

Data Fabric es una arquitectura de datos que integra de manera cohesiva y transparente diferentes tecnologías y fuentes de datos para proporcionar acceso unificado y en tiempo real a los datos en toda la organización. Se enfoca en la accesibilidad, la integridad y la consistencia de los datos, permitiendo a las organizaciones gestionar de manera efectiva la complejidad de sus entornos de datos distribuidos.



Arquitectura de datos

1 Arquitectura de datos tradicional:

- **Descripción:** Basada en bases de datos relacionales (SQL) y estructurada en torno a un almacén de datos centralizado.
- **Características:** Utiliza ETL (Extract, Transform, Load) para integrar datos, esquemas de datos predefinidos, y acceso a través de consultas SQL.

2 Arquitectura de datos NoSQL:

- **Descripción:** Diseñada para manejar datos no estructurados o semiestructurados.
- **Características:** Mayor flexibilidad en los esquemas de datos, escalabilidad horizontal, y modelos de datos como documentos, grafos, columnas o clave-valor.

3 Arquitectura de datos NewSQL:

- **Características:** Ofrece la capacidad de escalar horizontalmente y procesar grandes volúmenes de datos.
- **Descripción:** Combina características de bases de datos SQL tradicionales con la escalabilidad y capacidad de procesamiento distribuido de las bases de datos NoSQL.

4 Arquitectura de datos Data Warehouse:

Descripción: Almacén de datos centralizado que se utiliza para almacenar datos de diversas fuentes y facilitar el análisis y la generación de informes.

Características: Estructura orientada a consultas, integración de datos de diferentes sistemas, y acceso a través de consultas SQL.

Arquitectura de datos

5 Arquitectura de datos Data Lake:

- **Descripción:** Repositorio centralizado para almacenar grandes volúmenes de datos en su formato original.
- **Características:** Acepta datos de cualquier tipo y estructura, lo que permite el análisis de datos en bruto y la preparación para el análisis.

6 Arquitectura de datos Data Lakehouse:

- **Características:** Almacena datos en bruto y ofrece capacidades de procesamiento y consulta estructurada, permitiendo análisis avanzados y consultas SQL en tiempo real.
- **Descripción:** Combina las capacidades de un Data Lake con las funcionalidades de un Data Warehouse.

7 Arquitectura de datos Data Mesh:

- **Descripción:** Enfoque distribuido y descentralizado para gestionar datos en una organización.
- **Características:** Distribuye la responsabilidad de los datos a través de múltiples dominios o equipos, cada uno responsable de los datos en su área funcional.

8 Arquitectura de datos Data Fabric:

- **Descripción:** Integra de manera cohesiva y transparente diferentes tecnologías y fuentes de datos.
- **Características:** Proporciona acceso unificado y en tiempo real a los datos en toda la organización, garantizando la accesibilidad, integridad y consistencia de los datos.

Áreas principales de la gestión de datos

- **Data Governance (Gobernanza de los Datos):** Se considera a Data Governance como parte de Data Management. Data Governance se ocupa de la planificación, supervisión y control en la gestión y uso de datos.
- **Data Architecture (Arquitectura de Datos):** Se ocupa de establecer los modelos, políticas y reglas para gestionar los datos a lo largo de la Organización
- **Data Modeling & Design (Diseño y Modelado de Datos):** Se ocupa de diseñar las bases de datos, implementarlas y soportarlas
- **Data Storage (Almacenamiento):** Determina Cómo, Dónde, Cada cuánto y Qué se almacena. Incluye a los modelos "oncloud"
- **Data Security (Seguridad de los Datos, de la información):** Todo lo que concierne a la transferencia, integridad, privacidad, confidencialidad de los datos.
- **Data Integration&Interoperability (Operatividad e Integración de los datos):** responsable de definir la integración y transferencia de los datos.
- **Data Quality (Calidad de los Datos):** a través de la que se define, controla y mejora la calidad de los datos|

Niveles de análisis de datos

01

02

03

04

Descriptivo

¿Qué pasó?
Mirar los hechos,
las cifras y otros
datos que le pro-
porcionan una
imagen detallada.

Diagnóstico

¿Por qué sucedió?
Examinar los ele-
mentos descriptivos
permite evaluar
críticamente por
qué ocurrió un re-
sultado.

Predictivo

¿Qué sucederá?
Dados los mismos
o diferentes ele-
mentos, ¿cuál sería
el resultado?

Prescriptivo

¿Qué tengo que
hacer? ¿Cómo
puede lograrse un
resultado específi-
co mediante el uso
de elementos de-
terminados?

1

2

3

4

Roles en el procesamiento de datos



preparar los datos para el análisis, su objetivo es desarrollar, construir, probar, mantener las arquitecturas de datos necesarios para obtener la información.

Habilidades:

- Matemáticas
- Programación
- Big Data



Herramientas:

SQL
Python



Explica los datos de una forma que sea fácil de entender y resumida. Usa los datos para tomar mejores decisiones

Habilidades:

- Estadística
- Comunicación
- Conocimiento del negocio



Herramientas:

Excel
Power BI
SQL



Utiliza varios algoritmos avanzados de aprendizaje automático para identificar la ocurrencia de un evento particular en el futuro.

Habilidades:

- Matemáticas
- Programación
- Big Data



Herramientas:

SQL
Python



CAPÍTULO I

INTRODUCCIÓN

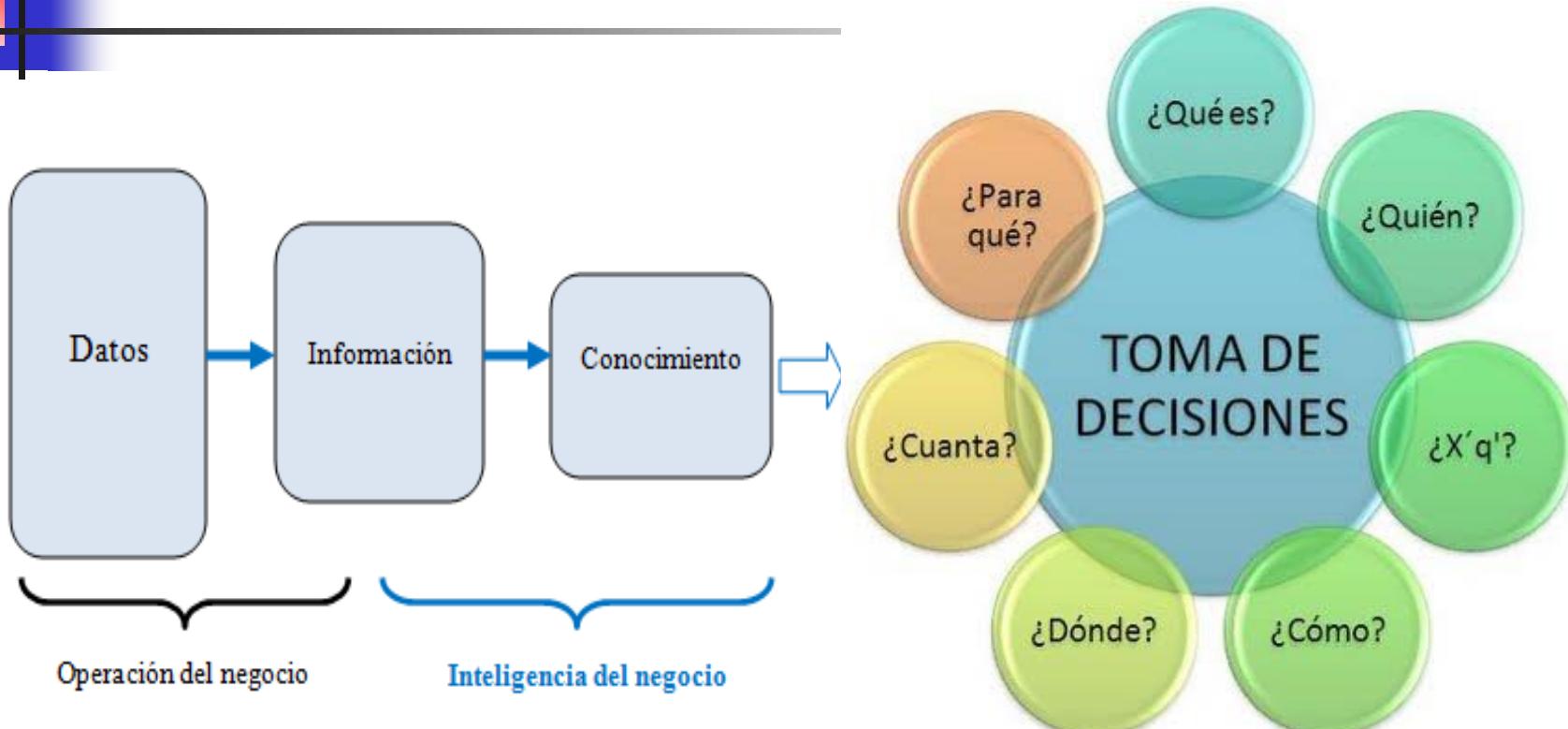
Datos, Información y Conocimiento

- **Datos**.- Son la mínima unidad semántica, y se corresponden con elementos primarios de información que por sí solos son irrelevantes como **apoyo a la toma de decisiones**. También se pueden ver como un conjunto discreto de valores, que no dicen nada sobre el por qué de las cosas y no son orientativos para la acción.
- **Información**.- Se puede definir como un conjunto de datos procesados y que tienen un significado (relevancia, propósito y contexto), y que por lo tanto son de utilidad para quién debe **tomar decisiones**, al disminuir su incertidumbre.
- **Conocimiento**.- Es una mezcla de experiencia, valores, información y *know-how* que sirve como marco para la incorporación de nuevas experiencias e información, y es útil para la acción. Se origina y aplica en la mente de los conocedores. En las organizaciones con frecuencia no sólo se encuentra dentro de documentos o almacenes de datos, sino que también está en rutinas organizativas, procesos, prácticas, y normas.

Datos, Información y Conocimiento



Datos, Información y Conocimiento



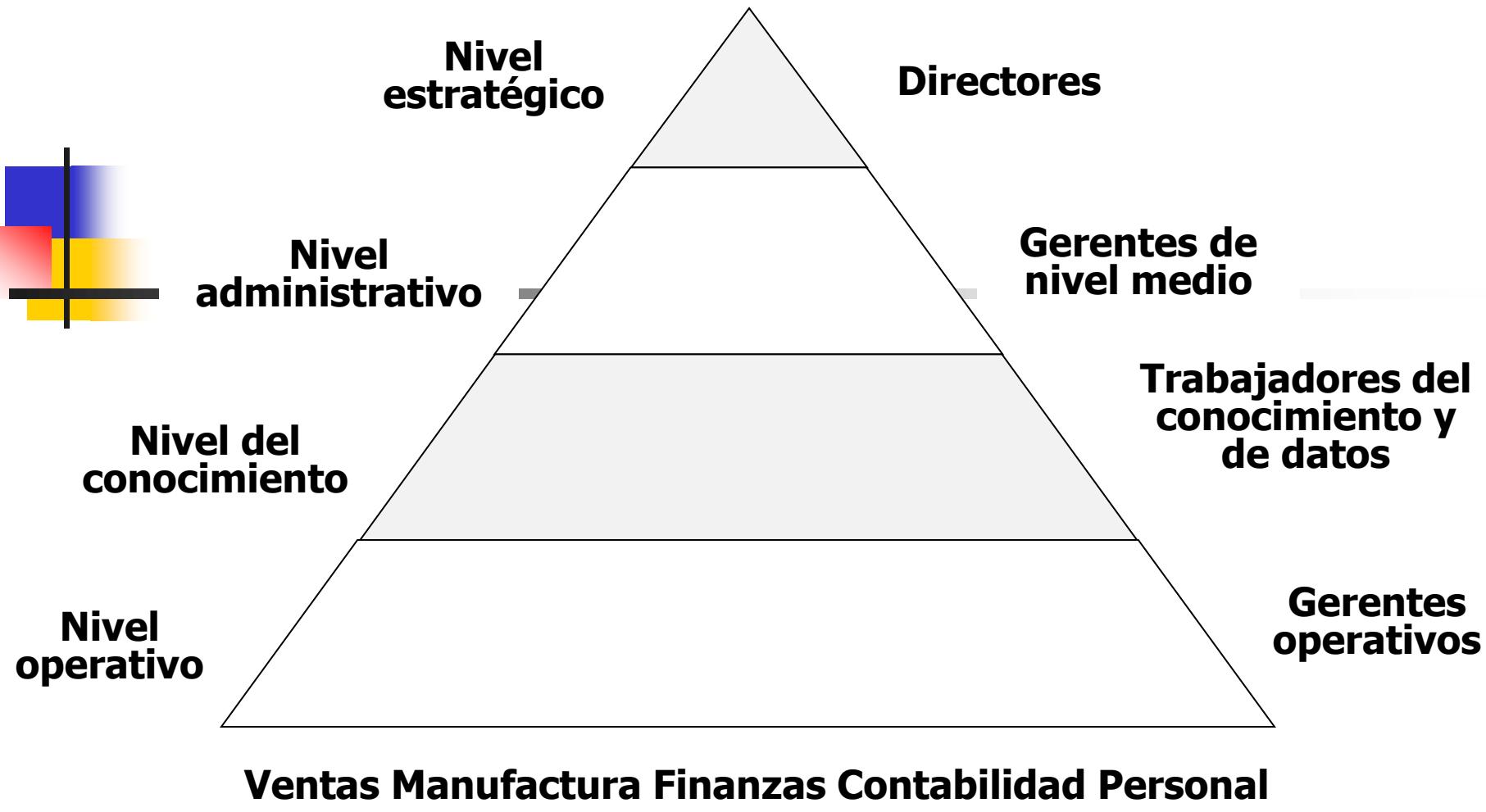
SISTEMAS DE INFORMACIÓN



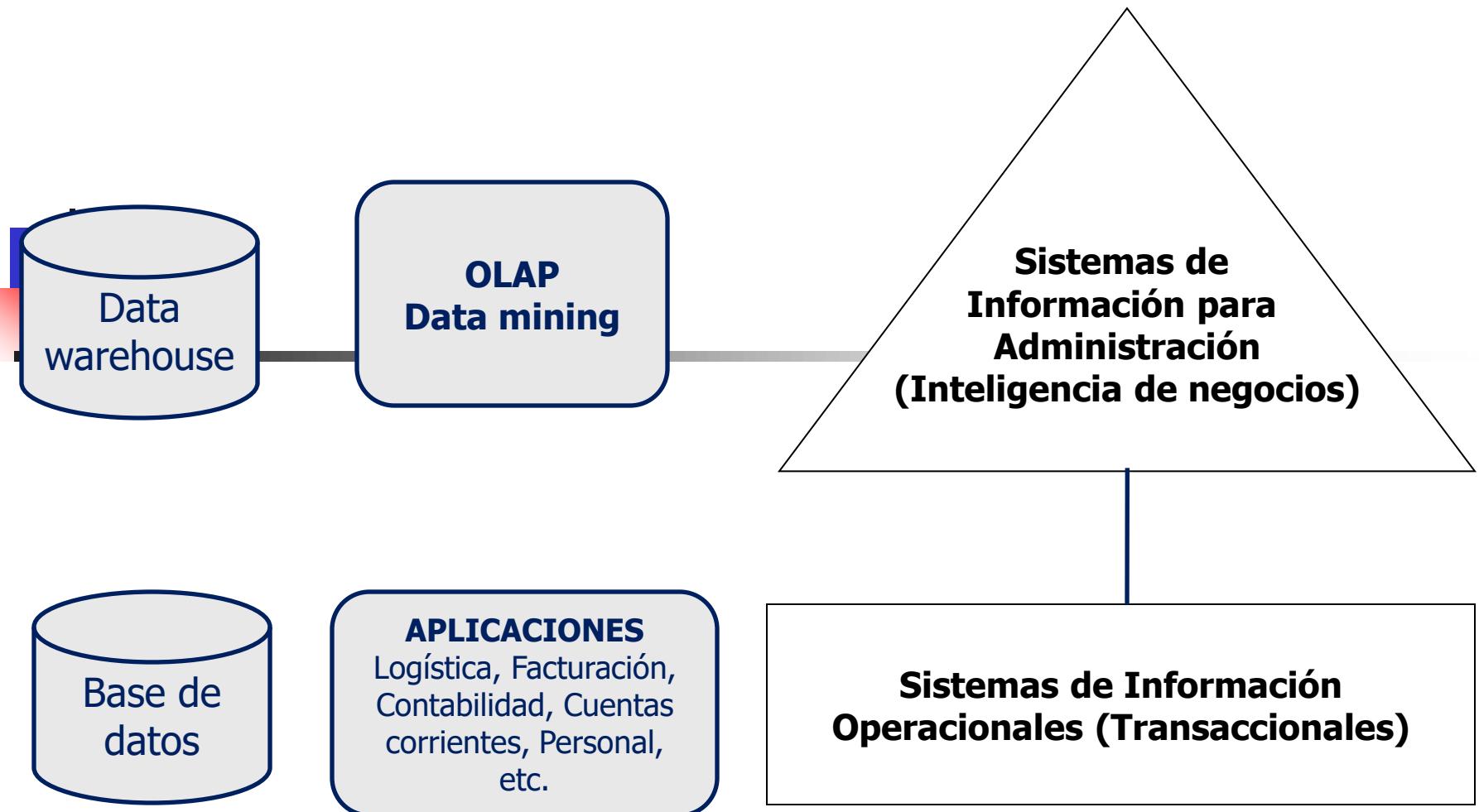
Las necesidades de información de nuestra sociedad actual se dejan sentir de forma cada vez más imperiosa. Nuestra *infósfera*, o esfera de la información, está siendo fuertemente conmocionada y no sólo se limita a acelerar el flujo de información, sino que transforma la estructura profunda de las decisiones de la que dependen nuestras acciones diarias.

SISTEMAS DE INFORMACIÓN...

Tipos de Sistemas en las organizaciones



Tipos de Sistemas de Información...



Evolución histórica de las bases de datos (Paradigmas de bases de datos)

■ **Paradigma pre-relacional (Primera generación)**

- Década del 50.- Sistema de archivos almacenados en cintas magnéticas.
- Década del 60.- (1963 se usa el término BD) Uso de "hard disk". Sistema Jerárquico y de red.
(Proceso registro a registro, no hay independencia Física/Lógica, Poca flexibilidad)

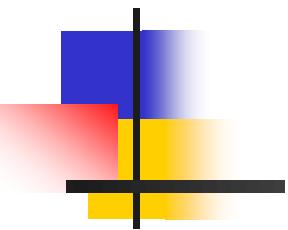
■ **Paradigma Relacional (Segunda generación)**

- Década del 70.- Edgar Frank Codd, publica: "A Relational Model of Data for Large Shared Data Banks"). Modelo con soporte matemático (Algebra y cálculo relacional).
- Década del 80.- Se afianza en el mercado, implementación de sistemas de gestión de bases de datos relacionales comerciales. Se inicia la investigación en otros modelos de datos (BDOO)
- Inicios de la década del 90.- Se crea y estandariza el SQL (ANSI SQL92).

■ **Paradigma pos-relacional (Tercera generación)**

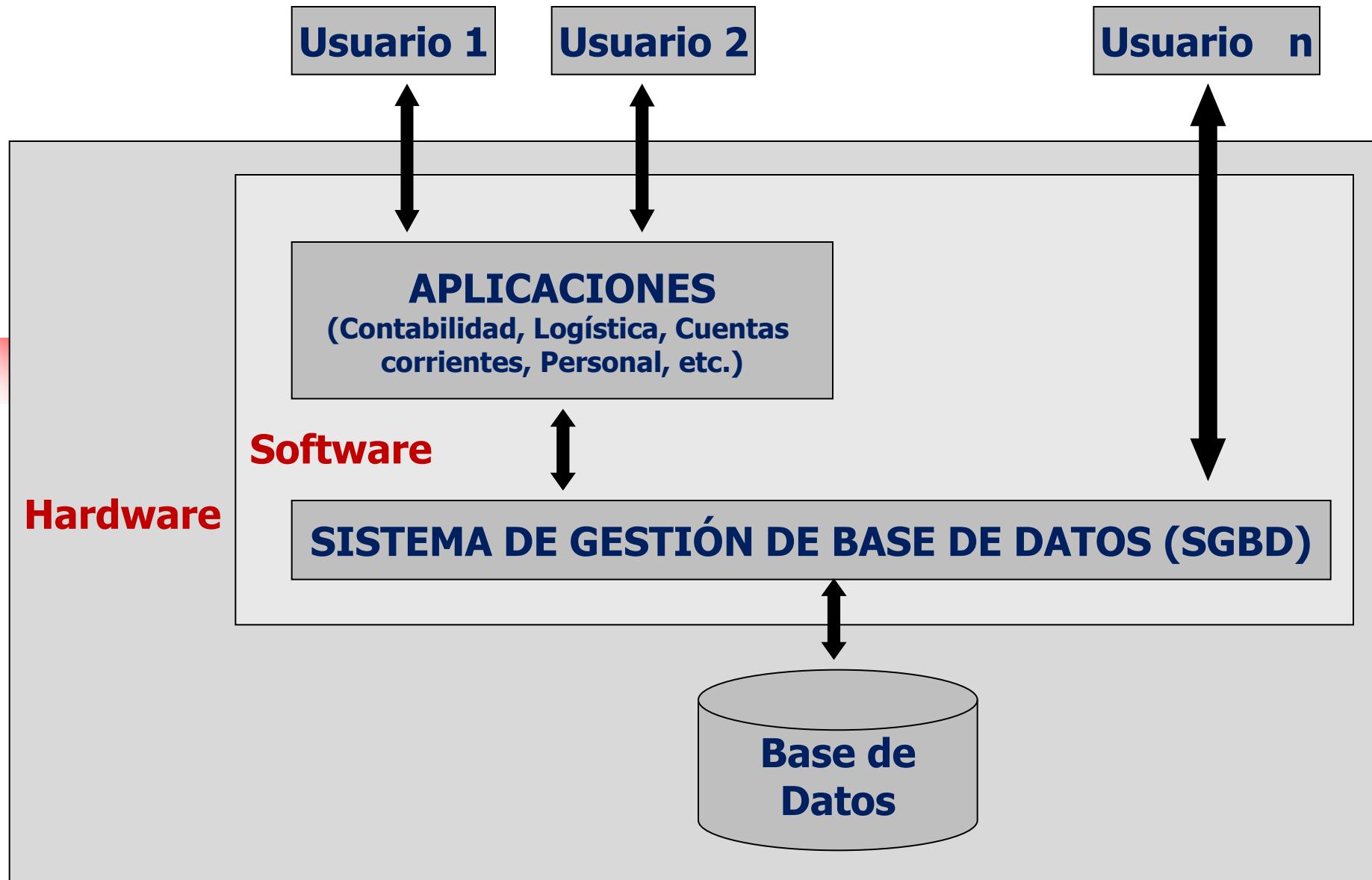
- Finales de la década del 90.- "Boom" de la década, aparece la "Word Wide Web".
- Surgen nuevos modelos de datos denominados **Not Only SQL** (NO SQL).
- Mayores exigencias y expectativas a las tecnologías de la información por parte de las organizaciones.
- Explotar las ventajas del paradigma relacional **SQL** y el no relacional **NO-SQL**. Surge **NEW-SQL**

Sistemas de base de datos



Un sistema de base de datos es el conjunto de **hardware, software, datos y personas**, que ordenadamente relacionadas entre sí contribuyen a un objetivo.

Sistemas de base de datos



Sistemas de base de datos

Ventajas:

- **Datos integrados.**
- **Reducción de datos duplicados.**
- **Independencia de Datos/Programas.**
- **Coherencia de resultados.**
- **Mayor valor informativo**
- **Representación fácil de las perspectivas del usuario.**

Sistemas de base de datos

Desventajas:

- **Instalación costosa.**
- **Personal especializado.**
- **Implantación larga y difícil.**
- **Falta de rentabilidad a corto plazo.**
- **Ausencia real de normas.**

Base de Datos

Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados. (Martin, 1975)

Conjunto estructurado de datos registrados sobre soportes accesibles por ordenador para satisfacer simultáneamente a varios usuarios de forma selectiva y en tiempo oportuno (Delobel, 1982)

Base de Datos

Colección no redundante de datos compatibles entre diferentes sistemas de aplicación. (Howe, 1983)

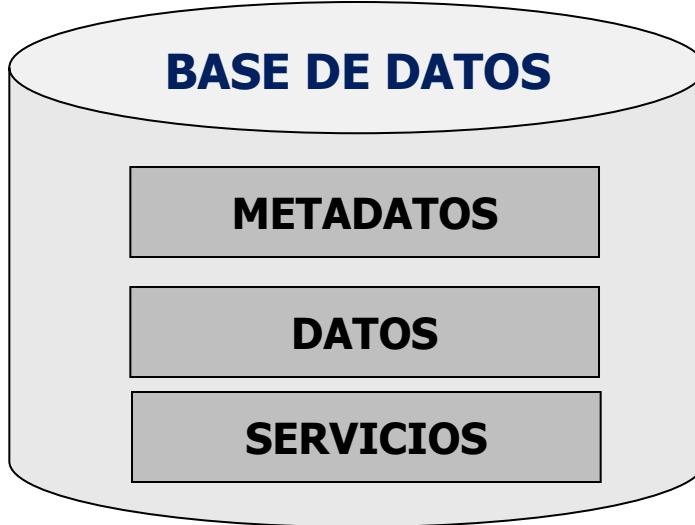
Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender toda las necesidades de los diferentes usuarios (Deen, 1985)

**Colección de datos interrelacionados.
(Emasri y Navathe, 1989)**

Base de Datos

Colección o depósito de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de los datos.

Componentes de las Bases de Datos



Los metadatos describen la estructura de los datos, las restricciones (de integridad, de dominio), las relaciones existentes entre los datos, Las operaciones aplicables a los datos. (Una base de datos es autodescriptiva).

Los Datos están organizados en forma de registros o tuplas organizados en tablas.

Los Servicios son un conjunto de subprogramas (funciones y procedimientos almacenados) escritos en el lenguaje proporcionado por el SGBD y residen en la base de datos al servicio de los usuarios que los soliciten.

Sistemas de Gestión de Base de Datos (SGBD)

Concepto

Conjunto coordinado de programas, procedimientos, lenguajes, etc. Que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad.

Sistemas de Gestión de Base de Datos (SGBD)

Funciones

- **Función de descripción o definición**

Permite describir: Los elementos de datos, Estructura, interrelaciones, restricciones y dominios.

Mediante un Data Definition Language (DDL)

- **Función de manipulación**

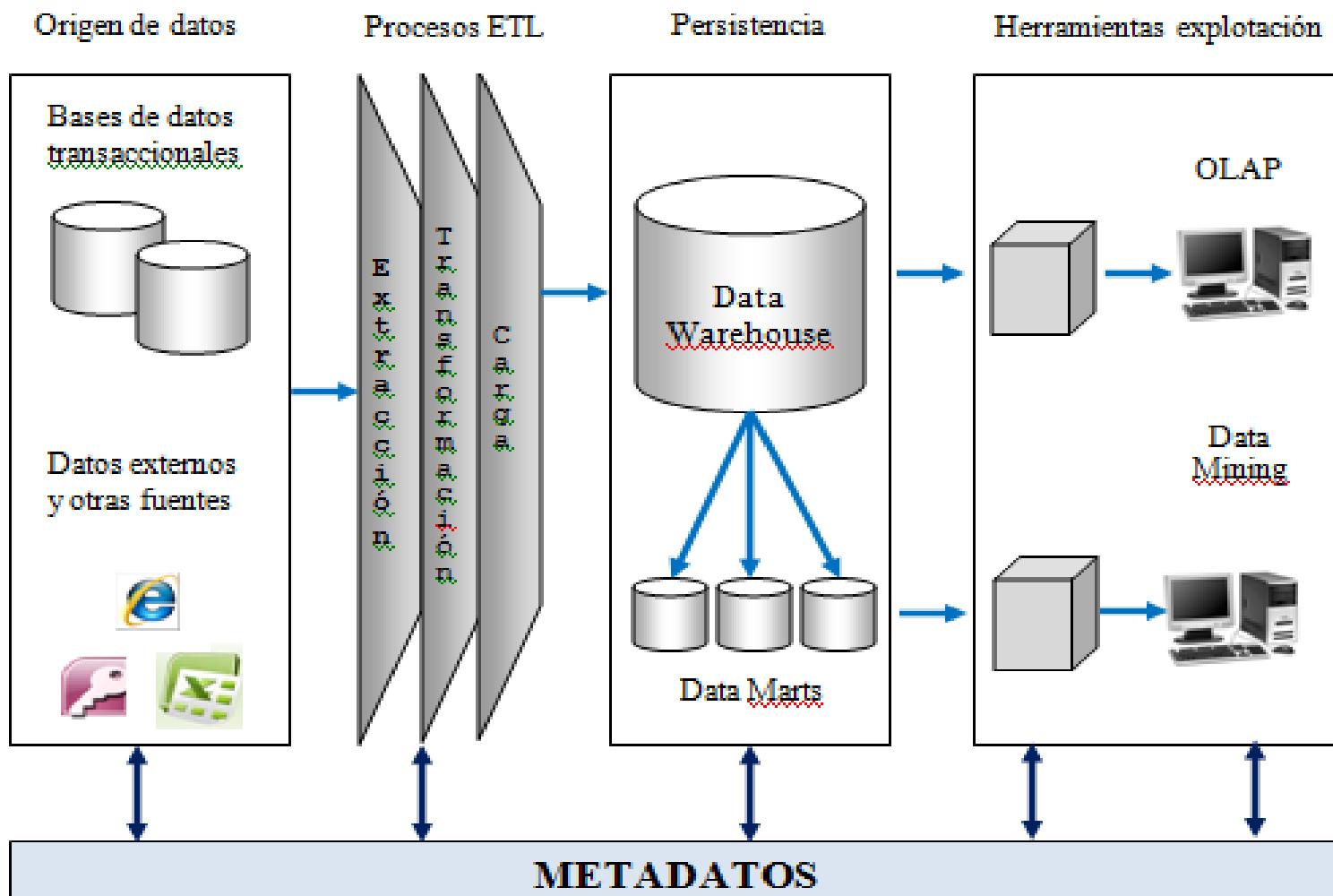
Permite: Añadir, suprimir, modificar y buscar

Mediante un Data Manipulation Language (DML)

- **Función de utilización**

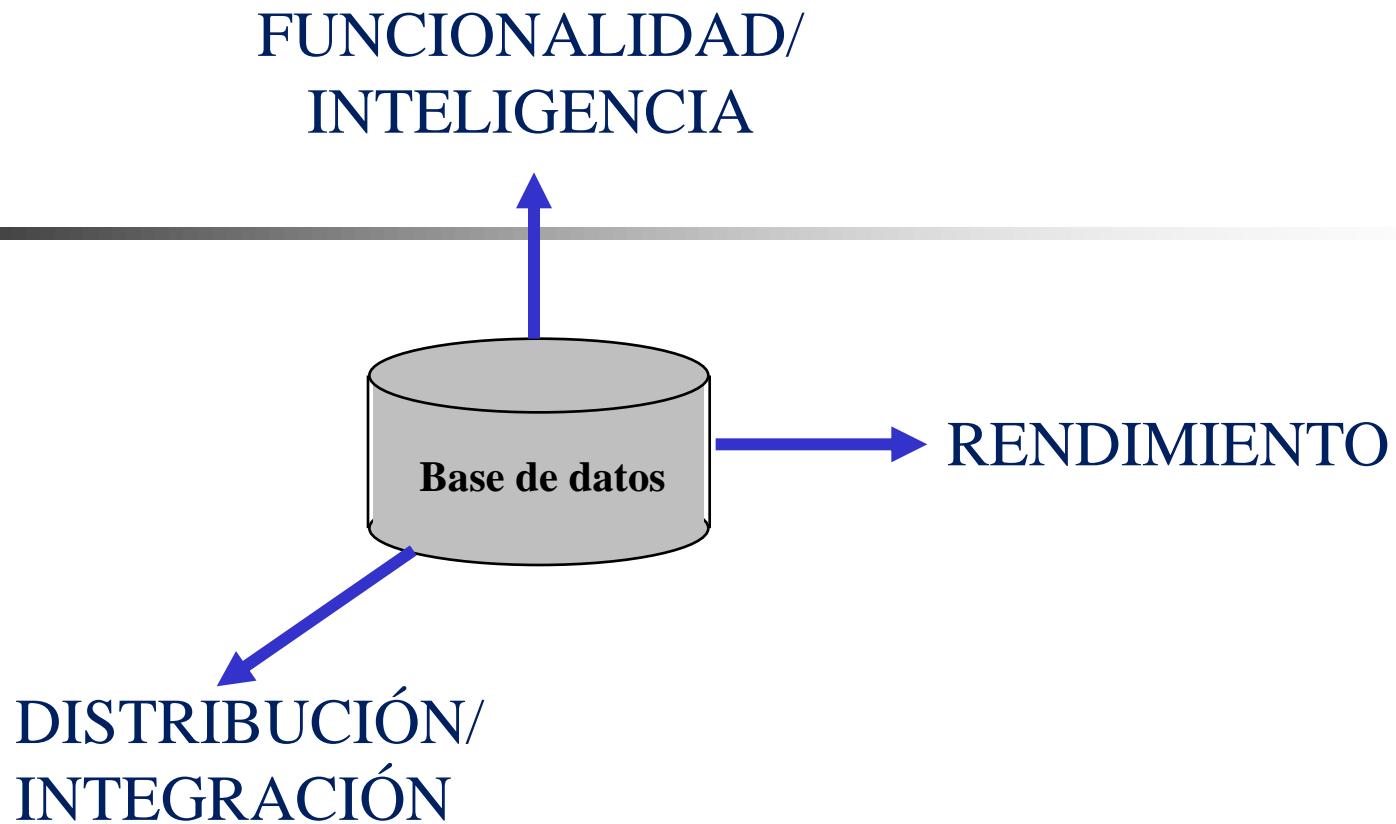
Interfaces para los usuarios y para el administrador.

Sistemas de apoyo a la toma de decisiones (Inteligencia de negocios)



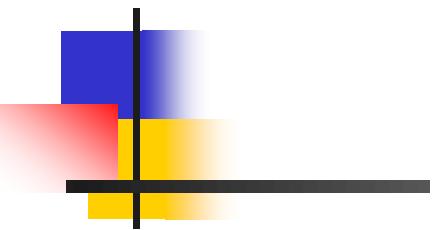
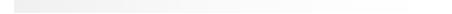
Tendencias de las Bases de Datos...

Líneas de Evolución



Tendencias de las Bases de Datos...

Línea de Evolución: Rendimiento

- 
- BD PARALELAS
 - BD EN TIEMPO REAL
 - BD EN MEMORIA PRINCIPAL
- 

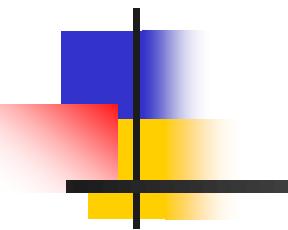
Tendencias de las Bases de Datos...

Línea de Evolución: Distribución

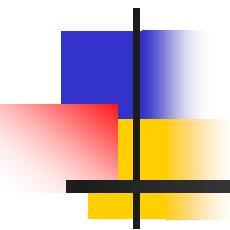
- BD DISTRIBUIDAS
- BD FEDERADAS
- MULTIBASES DE DATOS
- BD MÓVILES
- BD Y “WEB”

Tendencias de las Bases de Datos...

Línea de Evolución: Inteligencia

- 
- BD ACTIVAS
 - BD DEDUCTIVAS
 - BD ORIENTADAS A OBJETOS
 - BD MULTIMEDIA
 - BD TEMPORALES
 - BD SEGURAS
 - BD DIFUSAS
 - ALMACENES DE DATOS

Unidad Temática II



Modelos de datos

Modelos de datos

“Un conjunto de conceptos, reglas y convenciones bien definidos (matemáticamente) que nos permiten aplicar una serie de abstracciones a fin de describir (y manipular) los datos de un cierto mundo real que deseamos almacenar en la base de datos”

Consideran las propiedades del mundo real que son de dos tipos:

- **Estáticas**.- Estructuras (relativamente invariantes en el tiempo)
- **Dinámicas**.- Operaciones aplicadas a los datos almacenados en las estructuras .(Varían en el transcurso del tiempo)

$$\bullet \text{ MD} = \langle E, O \rangle$$

- Donde:
 - E = Estructura
 - O = Operaciones

Adoración de Miguel y Mario Piattini

Modelos de datos

De acuerdo con el Instituto Nacional Estadounidense de Estándares (ANSI) un modelo de datos se puede interpretar como un esquema:

- * **conceptual** que especifica las expresiones permitidas por el modelo mismo, comunica las reglas y definiciones esenciales de los datos a los usuarios
- * **lógico**, que describe la semántica de tablas y columnas, clases orientadas a objetos, etcétera, representada por una tecnología de manipulación en particular (como puede ser el lenguaje **SQL**)
- * **físico**, que detalla los medios en los que se almacena la información, como ser particiones de disco.

Paradigmas de Modelos de bases de datos (Durante su evolución histórica)

■ **Paradigma pre-relacional (Primera generación)**

- Sistema de archivos almacenados en cintas magnéticas.
- Modelo jerárquico. Modelo de Red.

(Proceso registro a registro, no hay independencia Física/Lógica, Poca flexibilidad)

■ **Paradigma Relacional (Segunda generación)**

- Modelo Relacional (Modelo SQL)

(Proceso sobre conjuntos, hay independencia Física/Lógica, flexibilidad)

■ **Paradigma pos-relacional (Tercera generación)**

- Almacenes de datos (Data warehouse, data lake)
- Not Only SQL (NO SQL).

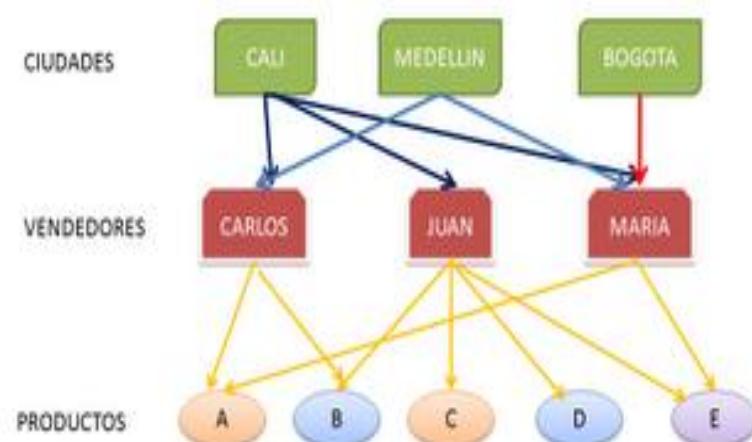
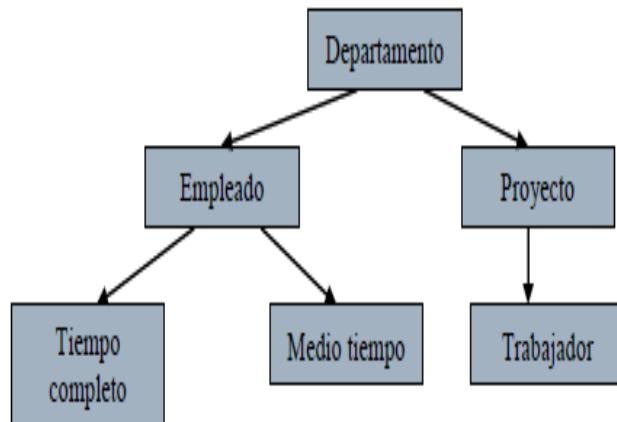
(Mayores exigencias y expectativas a las TIC's por parte de las organizaciones.

Gestión del conocimiento: DB inteligentes – Bases de Conocimientos)

Modelos del paradigma pre-relacional

I Modelos jerárquico y modelo de red.

- **Modelo Jerárquico.** Los datos se relacionan de modo jerárquico, y se representa mediante una estructura de tipo árbol.
- **Modelo de Red.**- Se entiende como una generalización del modelo jerárquico, en donde los nodos hijo pueden tener varios nodos padre.



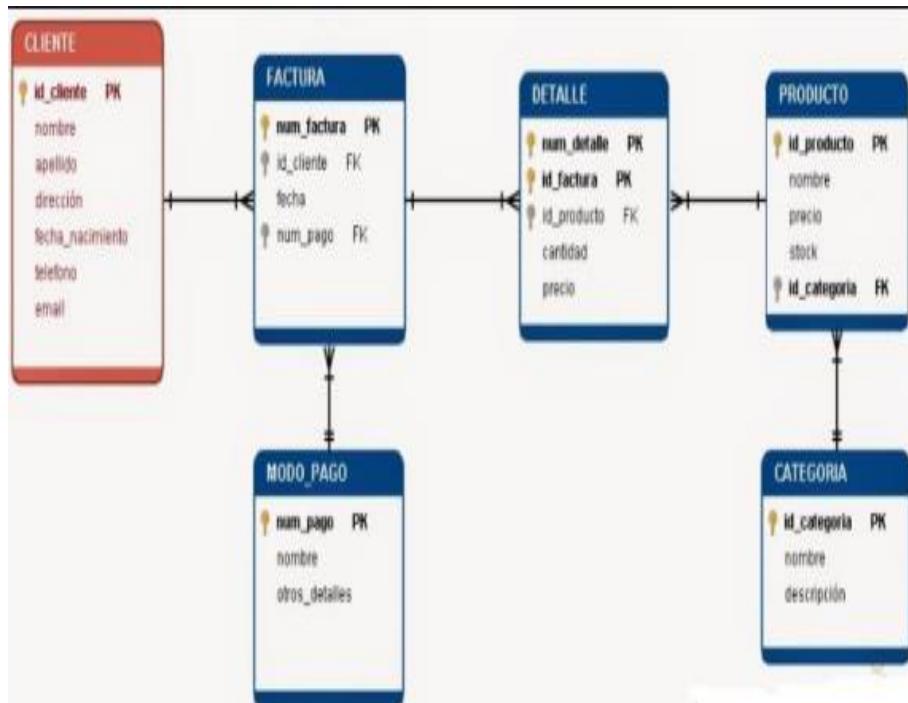
Modelos del paradigma relacional

Modelo relacional (SQL).

El modelo relacional es un conjunto de relaciones, donde cada relación representa a un conjunto de datos denominados tuplas; a su vez cada tupla abstrae los atributos o características de una entidad u objeto.

Este modelo tiene una fuerte base teórica, propuesta por Edgar Frank Codd en 1970. Se basa en la lógica de predicados y la teoría de conjuntos.

La manera más sencilla de interpretarla, es ver cada relación como una tabla, formada por tuplas (filas o registros) y cada tupla por atributos (columnas o campos).



CATEGORÍA

Id_Categoría	Nombre	Descripción

(Cada relación es una tabla)

Modelos del paradigma pos-relacional

Modelos NO SQL (Not Only SQL).

Un modelo de base de datos NoSQL es aquella que no requiere de estructuras de datos fijas como tablas; no garantizan completamente las características ACID y escalan muy bien horizontalmente. Comprende una gran variedad de modelos de datos (cualquier modelo distinto al modelo relacional).

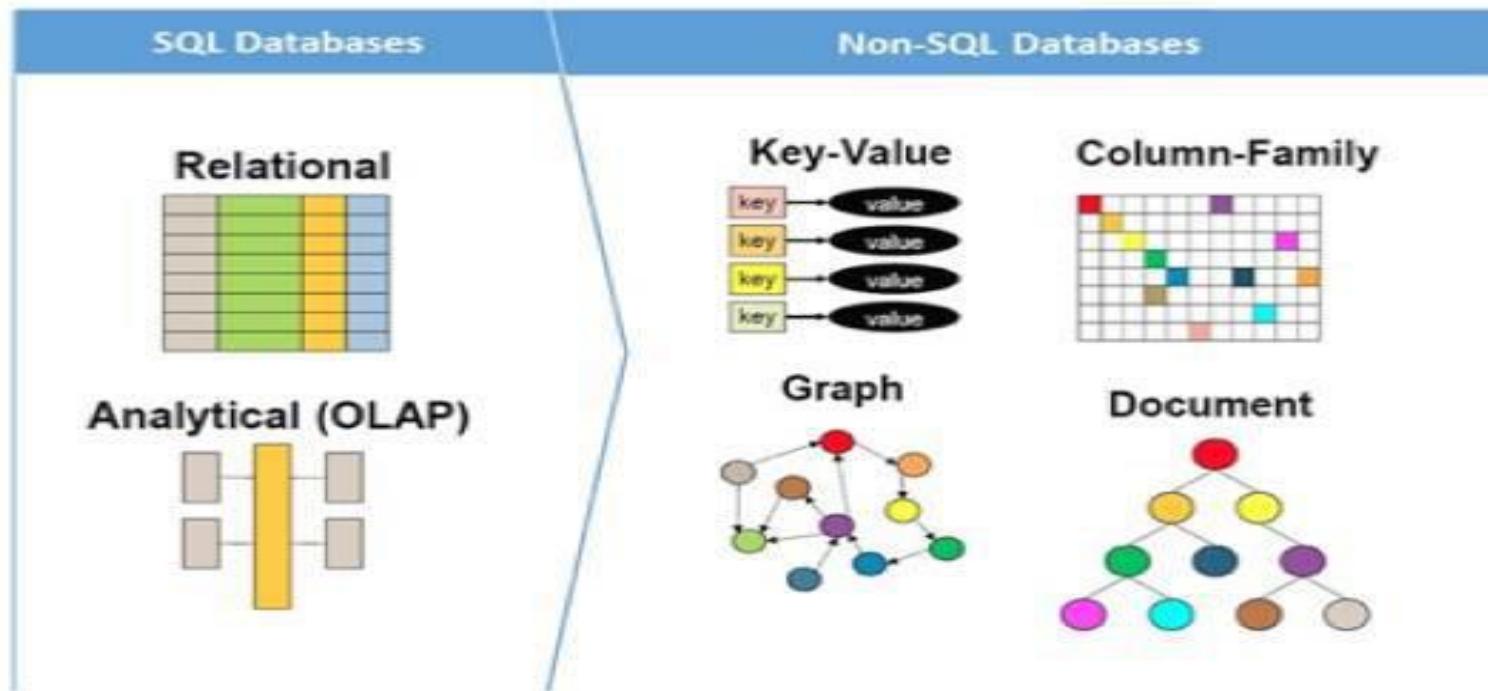
Responden a las exigencias cada vez más crecientes de procesamiento de grandes volúmenes de datos (Big Data).

Document databases	Graph databases	Key-value databases	Wide column stores
<p> Store data elements in document-like structures that encode information in formats such as JSON.</p> <p>+ Common uses include content management and monitoring Web and mobile applications.</p> <p>+ EXAMPLES: Couchbase Server, CouchDB, MarkLogic, MongoDB</p>	<p> Emphasize connections between data elements, storing related "nodes" in graphs to accelerate querying.</p> <p>+ Common uses include recommendation engines and geospatial applications.</p> <p>+ EXAMPLES: Allegrograph, IBM Graph, Neo4j</p>	<p> Use a simple data model that pairs a unique key and its associated value in storing data elements.</p> <p>+ Common uses include storing clickstream data and application logs.</p> <p>+ EXAMPLES: Aerospike, DynamoDB, Redis, Riak</p>	<p> Also called table-style databases—store data across tables that can have very large numbers of columns.</p> <p>+ Common uses include Internet search and other large-scale Web applications.</p> <p>+ EXAMPLES: Accumulo, Cassandra, HBase, Hypertable, SimpleDB</p>

Modelos del paradigma pos-relacional

■ Relacional (SQL) vs NO SQL

noSQL: “Not Only SQL”



Modelos del paradigma pos-relacional

Modelo New SQL

NewSQL es un paradigma moderno de gestión de bases de datos relacionales que tratan de conseguir el mismo rendimiento escalable de sistemas NoSQL para el procesamiento de transacciones en línea, manteniendo durante las cargas de trabajo las garantías ACID de un sistema de base de datos tradicional. El modelo de bases de datos **new sql** combina la transaccionalidad del paradigma relacional, con la alta escalabilidad del paradigma NO SQL; es decir, combina lo mejor de ambos paradigmas.



Comparativa de modelos: SQL vs NoSQL vs NewSQL

SQL

Ventajas.

- Madurez.
- Atomicidad.
- Estándares bien definidos.
- Sencillez en la escritura.

Desventajas.

- Crecimiento restringido
- Cambios en la estructura (Poca flexibilidad)
- Dificultad en la elección del mas adecuado (Diversidad de SGBD)
- Complejidad en la instalación

NoSQL

Ventajas.

- Versatilidad
- Crecimiento Horizontal
- Disponibilidad de Recursos
- Optimización

Desventajas.

- No garantiza atomicidad
- Documentación pobre del Software
- No sigue estándares en el lenguaje
- Pocas herramientas GUI(Graphical User Interface)

NewSQL

Ventajas.

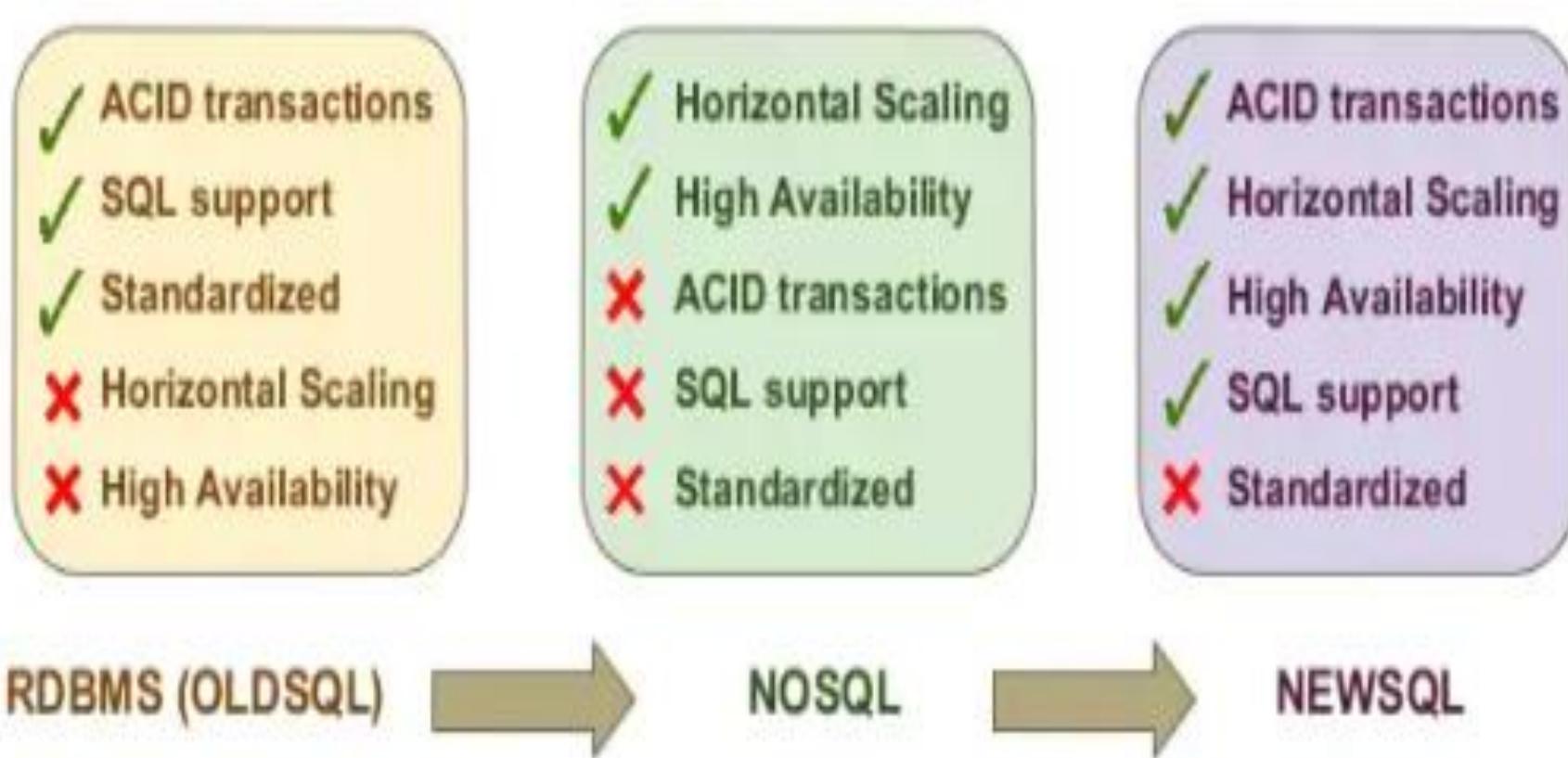
- Mayor consistencia y, a menudo, soporte transaccional completo.
- SQL familiar y herramientas estándar.
- Análisis más completos que aprovechan SQL y extensiones.
- Clústeres de estilo NoSQL con datos tradicionales y modelos de consulta.

Desventajas.

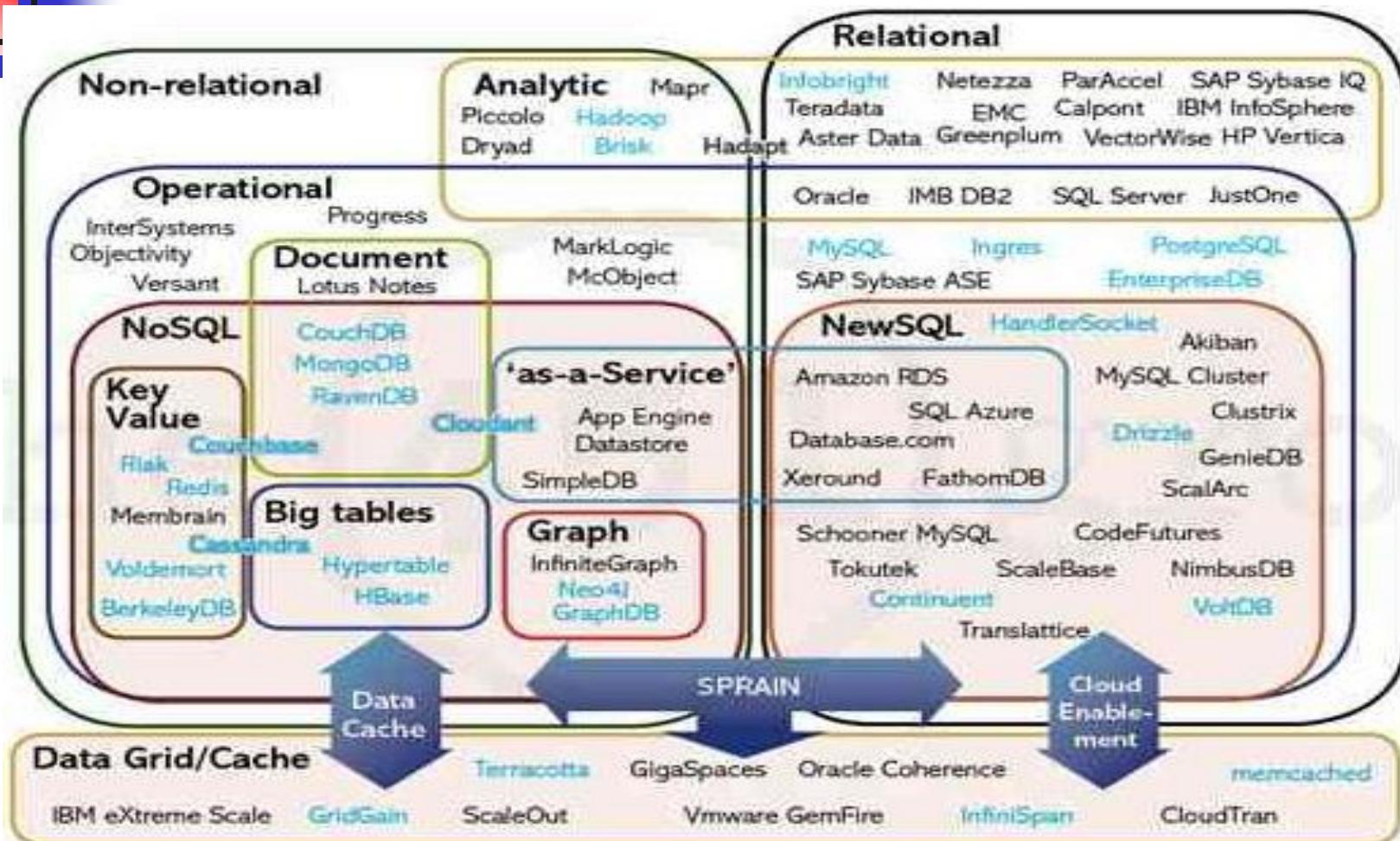
- No tiene un propósito tan general como los sistemas SQL.
- Las arquitecturas en memoria pueden ser inapropiadas para volúmenes que exceden unos pocos terabytes.

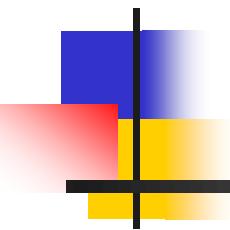
Comparativa de modelos: SQL vs NoSQL vs NewSQL

State of the Database



Panorámica de modelos de bases de datos

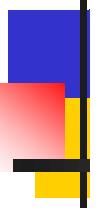




Paradigma de bases de datos relacionales

Modelo Relacional

Definición



Matemáticamente, una relación definida sobre los n dominios D₁, D₂, ..., D_n, no necesariamente distintos, es un subconjunto del producto cartesiano de estos dominios, donde cada elemento de la relación, tupla, es una serie de n valores ordenados.

Modelo Relacional

Estática (Intensión o Esquema de relación)

- **Esquema de relación $R(A_1:D_1, \dots, A_n:D_n)$** describe la relación.
 - **R** es el nombre de la relación
 - **$A_1:D_1, \dots, A_n:D_n$** es un conjunto de n pares atributo-dominio
 - **$D_i = \text{dom}(A_i)$** dominio de **A_i**
 - **Grado de la relación:** número de atributos

Modelo Relacional

Dinámica (Extensión)

Es un conjunto de n-tuplas $r=\{t_1, t_2, \dots, t_m\}$:

- cada n-tupla es una lista ordenada de n pares atributo-valor

$$t = \langle A_1:v_1,1; A_2:v_2,1; \dots; A_n:v_n,1 \rangle$$

- Donde $v_{i,j}$ es el valor j del dominio Di asociado al atributo Ai

El número de tuplas m es la cardinalidad de la relación

Modelo Relacional

Relación

Relación ≈ tabla de datos

Por eso se llaman bases de datos relacionales

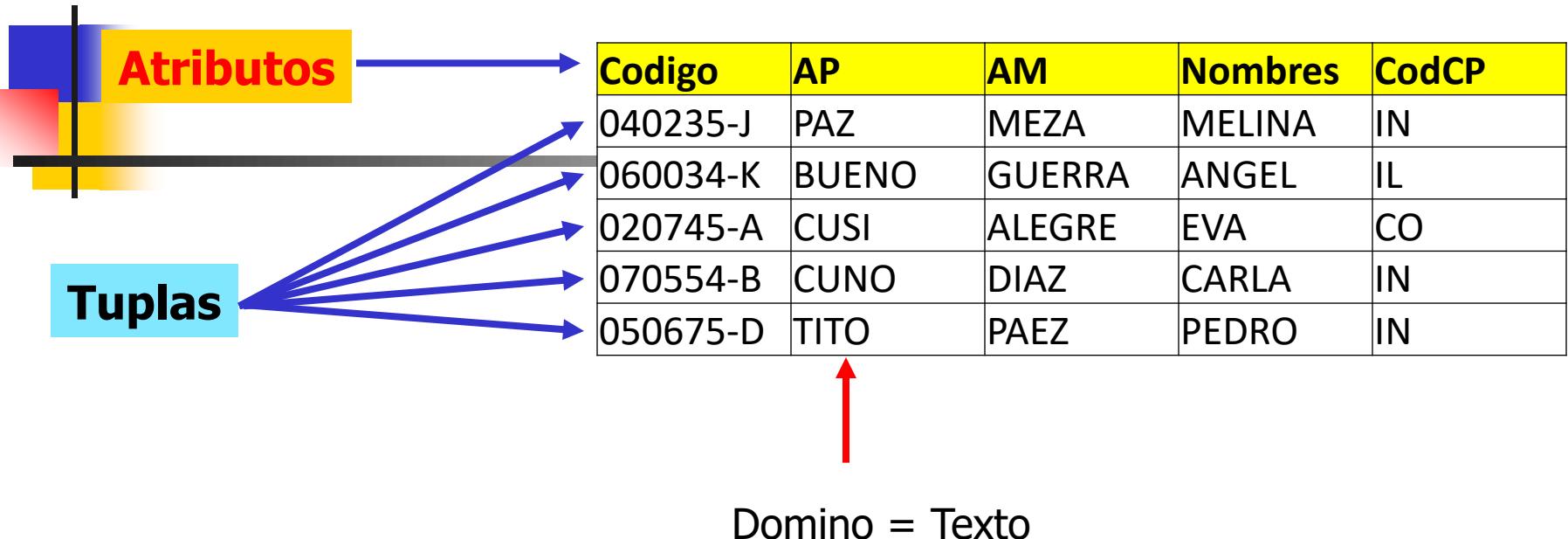
ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

Modelo Relacional

Esquema

ALUMNO (Codigo, AP, AM, Nombres, CodCP)



Modelo Relacional

Notación

- Atributos $\longrightarrow A_i$
- Relación $\longrightarrow R(A_1, A_2, A_3, \dots, A_n)$
- Dominio de $A_i \longrightarrow \text{dom}(A_i)$

$$R(A_1, A_2, A_3, \dots, A_n) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n))$$

Relación son algunas de todas las combinaciones posibles

Modelo Relacional

Restricciones de claves

Restricciones de claves únicas

$K \subseteq \text{Claves(Relación)}$

$\forall t_1, t_2 \in \text{Relación}, t_1[K] \neq t_2 [K]$

Una clave puede ser:

- Un Atributo
- Varios Atributos (Clave compuesta)

Una clave no puede ser nula (NULL)

Modelo Relacional

Restricciones.- Estructuras u ocurrencias no permitidas.

Restricciones Inherentes

- **Integridad de Entidad**
 - No hay dos tuplas iguales
 - El orden de las tuplas no es significativo
 - El orden de los atributos (columnas) no es significativo
 - Cada atributo sólo puede tomar un único valor del dominio, no admitiéndose por tanto los grupos repetitivos.

Modelo Relacional

Restricciones de Usuario

Integridad Referencial

“Si una relación R2 (relación que referencia) tiene un atributo que es la clave primaria de la relación R1 (relación referenciada), todo valor de dicho descriptor debe concordar con un valor de la clave primaria de R1, o ser nulo”. El atributo es, por tanto, una clave foránea de la relación R2.

CARRERA(CodCP, NombreCP)

CO	Contabilidad
----	--------------

ALUMNO(Codigo, AP ,AM ,Nombres, CodCP)

990104-A	PAZ	DIAZ	Ana	CO
----------	-----	------	-----	----

990305-D	ARCE	MEZA	Pedro	AD
----------	------	------	-------	----

Modelo Relacional

Operaciones en la Integridad Referencial

- **Restringida**
- **Con transmisión en cascada**
- **Con puesta a nulos**
- **Con puesta a valor por defecto**
- **Que desencadena un procedimiento de usuario.**

CARRERA(CodCP, NombreCP)

CO	Contabilidad
AD	Administración

ALUMNO(Codigo, AP ,AM ,Nombres, CodCP)

990104-A	PAZ	DIAZ	Ana	CO
990305-D	ARCE	MEZA	Pedro	AD

Modelo Relacional

Restricciones entre elementos

■ Restricción intrarrelación sobre atributos

PRESTAMO(ID_Prestamo, Fecha, Importe, FechaVcto,...)

(Fecha <= FechaVcto)

■ Restricción intrarrelación sobre tuplas

ASIENTO_DETALLE(ID_Asiento, Cod_Cta, Debe, Haber,...)

(En el mismo se debe cumplir: Total_Debe = Total_Haber)

■ Restricción interrelación

PRESTATARIO (ID_Prestatario,Nombres,Límite_Cred,...)

PRESTAMO(ID_Prestamo, Fecha, Importe, FechaVcto,...)

(El Importe no puede ser mayor al Límite de crédito)

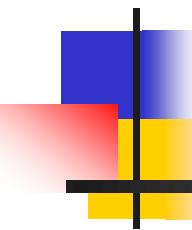
■ Restricción sobre dominios

PREDIO (ID_Predio, Dirección, Distrito, ID_Propietario,...)

(Distrito ∈ {Cusco, Wanchaq, Santiago, Poroy, ...})

Modelo Relacional

Objetivos

- 
- **Independencia Física**
 - **Independencia Lógica**
 - **Flexibilidad**
 - **Uniformidad**
 - **Sencillez**

Unidad Temática III

Álgebra relacional

Álgebra Relacional

- Se basa en el álgebra de la teoría de conjuntos donde los operandos son tablas o relaciones.
- Manipula relaciones produciendo nuevas relaciones.
- Cualquier operación da como resultado una tabla o relación con la que se puede operar de nuevo.
- Consiste en operaciones que algunas de ellas son tomadas de la matemática, otras del lenguaje relacional y otras de lenguajes de programación comunes.

Álgebra Relacional...

Clasificación de las operaciones relacionales:

Operaciones básicas

Unarias: operan con una sola tabla.

Selección

Proyección

Binarias o de conjunto: Operan con dos tablas.

Unión

Diferencia

Producto cartesiano

Operaciones derivadas o adicionales:

Realizan en su proceso llamadas a las operaciones básicas.

Intersección

Cociente o división

Join o reunión

Álgebra Relacional...

Clasificación de las operaciones relacionales:

- Las de origen matemático son:

- Unión
- Intersección
- Diferencia de conjuntos
- Producto Cartesiano

- La del lenguaje de programación es:

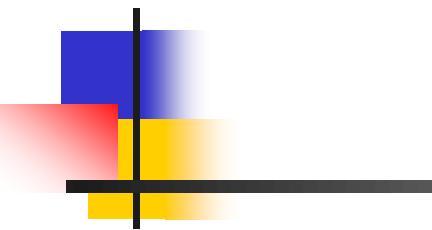
- Asignación

- Las de lenguaje relacional son:

- Proyección
- Selección
- División o cociente
- Join

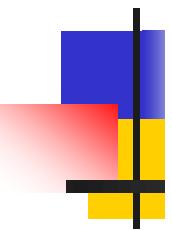
Álgebra Relacional...

Operaciones en forma genérica:

- 
- Proyectar (π)
 - Seleccionar (σ)
 - Producto cartesiano (\times)
 - Juntar o Join (\bowtie)
 - Unir (u)
 - Intersectar (\cap)
 - Diferencia ($-$)
 - Dividir (\div)

Álgebra Relacional...

Proyectar (π)



Selecciona columnas (valores de ciertos atributos) de todas las tuplas de una relación

$$\pi_{A_1, A_2, \dots, A_n}(R) = \{ t[A_1, A_2, \dots, A_n] \} : t \in R$$

Selecciona columnas completas.

La relación resultante tiene la misma cardinalidad de la relación original, mientras que el grado de la relación disminuye.

Álgebra Relacional...

Proyectar (π) Ejemplos

ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

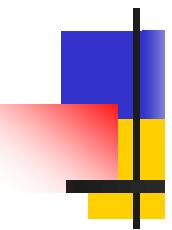
π Codigo, AP, AM, Nombres (ALUMNO)

Codigo	AP	AM	Nombres
040235-J	PAZ	MEZA	MELINA
060034-K	BUENO	GUERRA	ANGEL
020745-A	CUSI	ALEGRE	EVA
070554-B	CUNO	DIAZ	CARLA
050675-D	TITO	PAEZ	PEDRO

Álgebra Relacional...

Seleccionar (σ)

Selecciona tuplas(valores de tuplas) de una relación


$$\sigma_{\text{Condición}}(R) = \{t \in R : \text{Condición}(t) \text{ es cierto}\}$$

Selecciona filas completas.

La relación resultante tiene el mismo grado de la relación original, mientras que la cardinalidad de la relación disminuye.

Álgebra Relacional...

Seleccionar (σ) Ejemplos

ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

σ CodCP = 'IN' (ALUMNO)

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

Operaciones combinadas con: π y σ

Considerando la siguiente tabla:

ALUMNO

Codigo	AP	AM	Nombres	CodCP
100235-J	PAZ	MEZA	MELINA	IN
110034-K	BUENO	GUERRA	ANGEL	IL
120745-A	CUSI	ALEGRE	EVA	CO
090554-B	CUNO	DIAZ	CARLA	IN
120675-D	TITO	PAEZ	PEDRO	IN

Seleccionar a los alumnos ingresantes del 2012, con los siguientes atributos:

$R(\text{Codigo, AP, AM, Nombres})$

Operaciones combinadas con: π y σ

Los alumnos ingresantes del 2012 son aquellos cuyos códigos empiezan con 12. Se tiene dos alternativas de solución:

Alternativa 1:

$\sigma_{\text{Codigo} > '120000' \text{ y } \text{Codigo} < '129999'} (\pi_{\text{Codigo}, \text{AP}, \text{AM}, \text{Nombres}} (\text{ALUMNO}))$

Alternativa 2:

$\pi_{\text{Codigo}, \text{AP}, \text{AM}, \text{Nombres}} (\sigma_{\text{Codigo} > '120000' \text{ y } \text{Codigo} < '129999'} (\text{ALUMNO}))$

El resultado de un operador es una tabla, sobre la que se puede aplicar otro operador.

Ambas alternativas son válidas y dan el mismo resultado:

Codigo	AP	AM	Nombres
120745-A	CUSI	ALEGRE	EVA
120675-D	TITO	PAEZ	PEDRO

¿Cuál de las alternativas es más adecuada?

En operaciones combinadas efectuar primero la operación de selección (porque reduce la cardinalidad de la tabla) luego recién la proyección o cualquier otra operación.

UNION (U)

ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

Obtener la relación de alumnos de Ingeniería Informática (IN) e Ingeniería Electrónica (IL).

$$\sigma_{\text{CodCP} = \text{'IN'}}(\text{ALUMNO}) \cup \sigma_{\text{CodCP} = \text{'IL'}}(\text{ALUMNO})$$

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

La operación de unión se realiza sobre relaciones con la misma estructura

Álgebra Relacional...

DIFERENCIA (-)

ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

Obtener la relación de alumnos sin considerar a los de Contabilidad (CO)

ALUMNO - $\sigma_{\text{CodCP} = \text{'CO'}}(\text{ALUMNO})$

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

La operación de Diferencia se realiza sobre relaciones con la misma estructura

INTERSECCIÓN (\cap)

ALUMNO

Codigo	AP	AM	Nombres	CodCP
100235-J	PAZ	MEZA	MELINA	IN
110034-K	BUENO	GUERRA	ANGEL	IL
120745-A	CUSI	ALEGRE	EVA	CO
090554-B	CUNO	DIAZ	CARLA	IN
120675-D	TITO	PAEZ	PEDRO	IN

- Obtener la relación de alumnos de Ingeniería Informática y que hayan ingresado en el 2012.

$$\sigma_{\text{CodCP} = \text{'IN'}}(\text{ALUMNO}) \cap \sigma_{\text{Codigo} > \text{'120000'} \text{ y } \text{Codigo} < \text{'129999'}}(\text{ALUMNO})$$

Codigo	AP	AM	Nombres	CodCP
120675-D	TITO	PAEZ	PEDRO	IN

La operación de Intersección se realiza sobre relaciones con la misma estructura

Ejercicios:

Dada la siguiente Base de datos:

CARRERA(IdCarrera, NombreCarrera)

ALUMNO(IdAlumno, AP, AM, Nombres, IdCarrera)

ASIGNATURA(IdAsignatura, NombreAsignatura, Cat, Cred, PreRequisito)

MATRICULA(Semestre, IdAsignatura, IdAlumno, Nota)

Ejercicios:

1.- Determinar la relación de alumnos invictos en el semestre 2013-I

-- Proyectar el atributo **IdALumno** de la tabla **ALUMNO**

$T1 \leftarrow \pi_{IdCodigo} (\text{ALUMNO})$

-- Seleccionar los alumnos matriculados en el semestre 2013-I

$T2 \leftarrow \sigma_{\text{Semestre} = '2013-I'} (\text{MATRICULA})$

-- Obtener la relación de alumnos con por lo menos una asignatura desaprobada en el semestre 2013-I

$T3 \leftarrow \pi_{IdCodigo} (\sigma_{\text{Nota} < '14' \text{ o } \text{Nota} = 'NSP'} (T2))$

-- Obtener la relación de alumnos invictos en el semestre 2013-I

$R \leftarrow T1 - T3$

¿Es este algoritmo correcto?

Es incorrecto, porque en la relación T1 se tiene la totalidad de alumnos, incluso los que no se matricularon en el semestre 2013-I, y éstos (falsamente) estarían considerados como alumnos invictos del semestre 2013-I.

Ejercicios:

1.- Determinar la relación de alumnos invictos en el semestre 2013-I (Algoritmo correcto)

-- Seleccionar los alumnos matriculados en el semestre 2013-I

$T1 \leftarrow \sigma_{\text{Semestre} = '2013-I'} (\text{MATRICULA})$

-- Proyectar el atributo **IdAlumno** de los matriculados en el semestre 2013-I

$T2 \leftarrow \pi_{\text{IdCodigo}} (T1)$

-- Obtener la relación de alumnos con por lo menos una asignatura desaprobada en el semestre 2013-I

$T3 \leftarrow \pi_{\text{IdCodigo}} (\sigma_{\text{Nota} < '14' \text{ o } \text{Nota} = 'NSP'} (T1))$

-- Obtener la relación de alumnos invictos en el semestre 2013-I

$R \leftarrow T2 - T3$

Este algoritmo si considera sólo a los que no se matricularon en el semestre 2013-I, y de éstos selecciona a los alumnos invictos del semestre 2013-I.

Producto cartesiano (X)

CARRERA

CodCP	NombreCP
IN	INFORMATICA
IL	ELECTRONICA
CO	CONTABILIDAD

ALUMNO

Codigo	AP	AM	Nombres	CodCP
120235-J	PAZ	MEZA	MELINA	IN
090034-K	BUENO	GUERRA	ANGEL	IL
110745-A	CUSI	ALEGRE	EVA	CO
100554-B	CUNO	DIAZ	CARLA	IN
120675-D	TITO	PAEZ	PEDRO	IN

ALUMNO X CARRERA

Producto cartesiano (X)

ALUMNO X CARRERA

ALUMNO

Codigo	AP	AM	Nombres	CodCP
120235-J	PAZ	MEZA	MELINA	IN
090034-K	BUENO	GUERRA	ANGEL	IL
110745-A	CUSI	ALEGRE	EVA	CO
100554-B	CUNO	DIAZ	CARLA	IN
120675-D	TITO	PAEZ	PEDRO	IN

CARRERA

CodCP	NombreCP
IN	INFORMATICA
IL	ELECTRONICA
CO	CONTABILIDAD

Cada tupla de la primera tabla se asocia (junta) con todas las tuplas de la segunda tabla. El producto cartesiano es el único mecanismo para obtener información de dos o más tablas, mediante el proceso de “juntar” tablas.

Álgebra Relacional...

Producto cartesiano (X)

ALUMNO X CARRERA

Codigo	AP	AM	Nombres	CodCP	CodCP	NombreCP
120235-J	PAZ	MEZA	MELINA	IN	IN	INFORMATICA
120235-J	PAZ	MEZA	MELINA	IN	IL	ELECTRONICA
120235-J	PAZ	MEZA	MELINA	IN	CO	CONTABILIDAD
090034-K	BUENO	GUERRA	ANGEL	IL	IN	INFORMATICA
090034-K	BUENO	GUERRA	ANGEL	IL	IL	ELECTRONICA
090034-K	BUENO	GUERRA	ANGEL	IL	CO	CONTABILIDAD
110745-A	CUSI	ALEGRE	EVA	CO	IN	INFORMATICA
110745-A	CUSI	ALEGRE	EVA	CO	IL	ELECTRONICA
110745-A	CUSI	ALEGRE	EVA	CO	CO	CONTABILIDAD
100554-B	CUNO	DIAZ	CARLA	IN	IN	INFORMATICA
100554-B	CUNO	DIAZ	CARLA	IN	IL	ELECTRONICA
100554-B	CUNO	DIAZ	CARLA	IN	CO	CONTABILIDAD
120675-D	TITO	PAEZ	PEDRO	IN	IN	INFORMATICA
120675-D	TITO	PAEZ	PEDRO	IN	IL	ELECTRONICA
120675-D	TITO	PAEZ	PEDRO	IN	CO	CONTABILIDAD

La cardinalidad de |ALUMNO X CARRERA| = | ALUMNO| | CARRERA|

UNSAAC - Mgt. Arturo Rozas

Huacho

Producto cartesiano (X)

El producto cartesiano al juntar las tablas genera asociaciones incongruentes, por lo que generalmente debe ir acompañado de una operación de selección, para seleccionar sólo las tuplas congruentes (Combinaciones correctas).

Codigo	AP	AM	Nombres	CodCP	CodCP	NombreCP
120235-J	PAZ	MEZA	MELINA	IN	IN	INFORMATICA
120235-J	PAZ	MEZA	MELINA	IN	IL	ELECTRONICA
120235-J	PAZ	MEZA	MELINA	IN	CO	CONTABILIDAD
090034-K	BUENO	GUERRA	ANGEL	IL	IN	INFORMATICA
090034-K	BUENO	GUERRA	ANGEL	IL	IL	ELECTRONICA
090034-K	BUENO	GUERRA	ANGEL	IL	CO	CONTABILIDAD
110745-A	CUSI	ALEGRE	EVA	CO	IN	INFORMATICA
110745-A	CUSI	ALEGRE	EVA	CO	IL	ELECTRONICA
110745-A	CUSI	ALEGRE	EVA	CO	CO	CONTABILIDAD
100554-B	CUNO	DIAZ	CARLA	IN	IN	INFORMATICA
100554-B	CUNO	DIAZ	CARLA	IN	IL	ELECTRONICA
100554-B	CUNO	DIAZ	CARLA	IN	CO	CONTABILIDAD
120675-D	TITO	PAEZ	PEDRO	IN	IN	INFORMATICA
120675-D	TITO	PAEZ	PEDRO	IN	IL	ELECTRONICA
120675-D	TITO	PAEZ	PEDRO	IN	CO	CONTABILIDAD

Producto cartesiano (X) Seleccionar combinaciones correctas

$\sigma_{ALUMNO.CodCP = CARRERA.CodCP} (ALUMNO \times CARRERA)$

Codigo	AP	AM	Nombres	CodCP	CodCP	NombreCP
120235-J	PAZ	MEZA	MELINA	IN	IN	INFORMATICA
090034-K	BUENO	GUERRA	ANGEL	IL	IL	ELECTRONICA
110745-A	CUSI	ALEGRE	EVA	CO	CO	CONTABILIDAD
100554-B	CUNO	DIAZ	CARLA	IN	IN	INFORMATICA
120675-D	TITO	PAEZ	PEDRO	IN	IN	INFORMATICA

Generalmente se efectúa el producto cartesiano sobre tablas que están relacionadas por claves primarias y claves foráneas. La selección de combinaciones correctas se efectúa sobre estas claves.

Ejercicios:

Dada la siguiente Base de datos:

VENDEDOR(IdVendedor, Nombres, Telefono, EMail)

SUCURSAL(IdSucursal, RazonSocial, Responsable)

CLIENTE(IdCliente, RazonSocial, RUC, Direccion)

ARTICULO(IdArticulo, Descripción, UnidadMedia, Precio)

COMPROBANTE(NroComprobante, Tipo, Fecha, IdCliente, IdSucursal, IdVendedor)

COMPROBANTE_DETALLE(NroComprobante, IdArticulo, Cantidad, PrecioUnitario)

Ejercicios:

1.- Relación de artículos que durante el año 2006 no se han vendido en ninguna de las sucursales

-- Obtener la relación de COMPROBANTES emitidos en el 2006

$T1 \leftarrow \sigma_{\text{Fecha} \geq '01/01/2006' \text{ and } \text{Fecha} \leq '31/12/2006'} (\text{COMPROBANTE})$

-- Obtener la relación de COMPROBANTES emitidos en el 2006 con su respectivo detalle

$T2 \leftarrow \sigma_{T1.\text{NroComprobante} = \text{COMPROBANTE_DETALLE}.\text{NroComprobante}} (T1 \times \text{COMPROBANTE_DETALLE})$

-- Determinar la relación de artículos vendidos en el 2006

$T3 \leftarrow \pi_{\text{IdArticulo}} (T2)$

-- Determinar la relación total de artículos

$T4 \leftarrow \pi_{\text{IdArticulo}} (\text{ARTICULO})$

-- Determinar la relación de artículos que no se han vendido en el 2006

$T5 \leftarrow T4 - T3$

-- Completar la información de artículos

$T1 \leftarrow \sigma_{T5.\text{IdArticulo} = \text{ARTICULO}.\text{IdArticulo}} (T5 \times \text{ARTICULO})$

Álgebra Relacional...

Reunir (JOIN) (\bowtie)

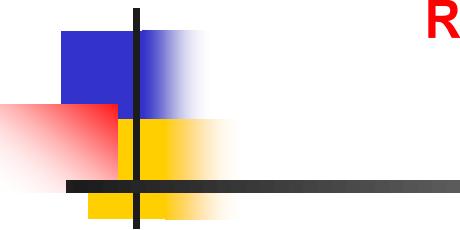
$$\sigma_{R1.k = R2.k}(R1 \times R2)$$



$$R1 \bowtie_k R2$$

Álgebra Relacional...

Reunir (JOIN) natural (\bowtie)



Reunir (JOIN)

R1 \bowtie_k R2

Omitir el subíndice implica juntar según todos los atributos que tengan el mismo nombre en ambas tablas

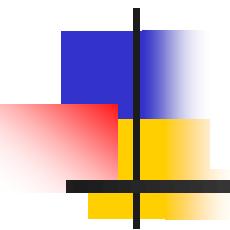
Reunir (JOIN) natural

R1 \bowtie R2

Álgebra Relacional Extendida

PROYECCIÓN GENERALIZADA

Extiende la proyección permitiendo que se utilicen funciones o expresiones aritméticas en la lista de proyección.


$$\pi_{F_1, F_2, \dots, F_n}(E)$$

Dado el siguiente esquema de relación:
NOTAS(CodAlumno, Nota1, Nota2, Nota3)

Ejemplo:

$$\pi_{\text{CodAlumno}, (\text{Nota1}+\text{Nota2}+\text{Nota3})/3 \text{ as Promedio}}(\text{NOTAS})$$

La relación resultante tendrá la siguiente estructura:
R(CodAlumno, Promedio)

Álgebra Relacional Extendida

FUNCIONES DE AGREGACIÓN

Las funciones de agregación operan sobre un conjunto de valores y devuelven como resultado un único valor.

 Las funciones de resumen son:

COUNT, SUM, AVG, MAX y MIN

Sintaxis:

$G_1, G_2, \dots G_n \text{ } \mathfrak{G} \text{ } F_1(A_1), F_2(A_2), \dots, F_n(A_n) \text{ (E)}$

El símbolo \mathfrak{G} es la letra “G caligráfica”.

Álgebra Relacional Extendida

FUNCIONES DE AGREGACIÓN (Ejemplos)

Determinar cuántos alumnos hay en la base de datos:

$\text{G COUNT}(\text{Codigo})(\text{ALUMNO})$

ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN

COUNT(Codigo)
5

Álgebra Relacional Extendida

FUNCIONES DE AGREGACIÓN (Ejemplos)

Determinar cuántos alumnos hay en cada carrera:

$\text{CodCP} \text{ COUNT}(\text{Codigo}) \text{ as NroAlumnos}$ (**ALUMNO**)

ALUMNO

Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN
090122-E	FUENTES	MAR	LUIS	CO



Codigo	AP	AM	Nombres	CodCP
040235-J	PAZ	MEZA	MELINA	IN
070554-B	CUNO	DIAZ	CARLA	IN
050675-D	TITO	PAEZ	PEDRO	IN
060034-K	BUENO	GUERRA	ANGEL	IL
020745-A	CUSI	ALEGRE	EVA	CO
090122-E	FUENTES	MAR	LUIS	CO

CodCP	NroAlumnos
IN	3
IL	1
CO	2

FUNCIONES DE AGREGACIÓN (Ejercicios)

Dada la siguiente Base de datos:

CARRERA(IdCarrera, NombreCarrera)

ALUMNO(IdAlumno, AP, AM, Nombres, IdCarrera)

ASIGNATURA(IdAsignatura, NombreAsignatura, Cat, Cred, PreRequisito)

MATRICULA(Semestre, IdAsignatura, IdAlumno, Nota)

FUNCIONES DE AGREGACIÓN - Ejercicios:

1.- Determinar el número de alumnos en cada Carrera Profesional.

-- Contar el número de alumnos en cada carrera profesional

$T1 \leftarrow \text{IdCarrera} \text{ } \text{G COUNT(IdAlumno) as NroAlumnos (ALUMNO)}$

-- Completar la información de carreras profesionales

$R \leftarrow \pi_{T1.IdCarrera, CARRERA.NombreCarrera, T1.NroAlumnos} (T1 \bowtie CARRERA)$

FUNCIONES DE AGREGACIÓN - Ejercicios:

2.- Determinar el número de alumnos matriculados por semestre y por Carrera Profesional.

-- Obtener la relación de Matrículas de Alumnos con sus respectivas carreras profesionales

$$T1 \leftarrow \text{MATRICULA} \bowtie \text{ALUMNO}$$

-- Contar el número de alumnos matriculados por semestre y por carrera profesional

$$T2 \leftarrow \text{Semestre, IdCarrera} \text{ } \text{G COUNT(IdAlumno) as NroAlumnos} (T1)$$

-- Completar la información de carreras profesionales

$$R \leftarrow \pi_{T2.IdCarrera, CARRERA.NombreCarrera, T2.Semestre, T2.NroAlumnos} (T2 \bowtie CARRERA)$$

FUNCIONES DE AGREGACIÓN - Ejercicios:

3.- Obtener la relación de alumnos con su respectivo número de créditos acumulados.

-- Obtener la relación de Matrículas de asignaturas aprobadas y número de créditos

$T1 \leftarrow \sigma_{\text{Nota} \geq 14} (\text{MATRICULA}) \bowtie \text{ASIGNATURA}$

-- Determinar la relación de alumnos con su respectivo número de créditos acumulados

$T2 \leftarrow \text{IdAlumno } \text{G} \text{ SUM(cred) as CredAcumulados(T1)}$

-- Completar información de Alumnos

$T3 \leftarrow \pi_{T2.IdAlumno, AP, AM, Nombres, IdCarrera, CredAcumulados} (T2 \bowtie \text{ALUMNO})$

FUNCIONES DE AGREGACIÓN - Ejercicios:

1.- Obtener la relación de los mejores estudiantes de cada carrera profesional en el último semestre. (Considerar como mejores estudiantes a los que hayan obtenido el mejor promedio aritmético)

$R(IdCarrera, IdAlumno, AP, AM, Nombres, Promedio)$

-- Obtener el último semestre

$T1 \leftarrow \text{G}_{MAX(Semestre)} \text{ as Semestre (MATRICULA)}$

-- Obtener la relación de matrículas del último semestre

$T2 \leftarrow T1 \bowtie \text{MATRICULA}$

-- Agregar el atributo carrera a los alumnos de las matrículas del último semestre

$T3 \leftarrow \pi_{ALUMNO.IdCarrera, T2.IdAlumno, T2.IdAsignatura, T2.Nota} (T2 \bowtie ALUMNO)$

-- Determinar el promedio aritmético para cada alumno

$T4 \leftarrow \text{IdCarrera, IdAlumno G}_{AVG(Nota)} \text{ as Promedio (T3)}$

-- Determinar el promedio más alto de cada carrera

$T5 \leftarrow \text{IdCarrera G}_{MAX(Promedio)} \text{ as Promedio (T4)}$

-- Determinar a los alumnos cuyos promedios son iguales a los promedios más altos de cada carrera

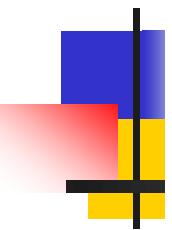
$T6 \leftarrow T4 \bowtie T5$

-- Completar información de Alumnos

$R \leftarrow \pi_{T6.IdCarrera, T6.IdAlumno, AP, AM, Nombres, Promedio} (T6 \bowtie ALUMNO)$

Álgebra Relacional Extendida...

Left Join (\bowtie)



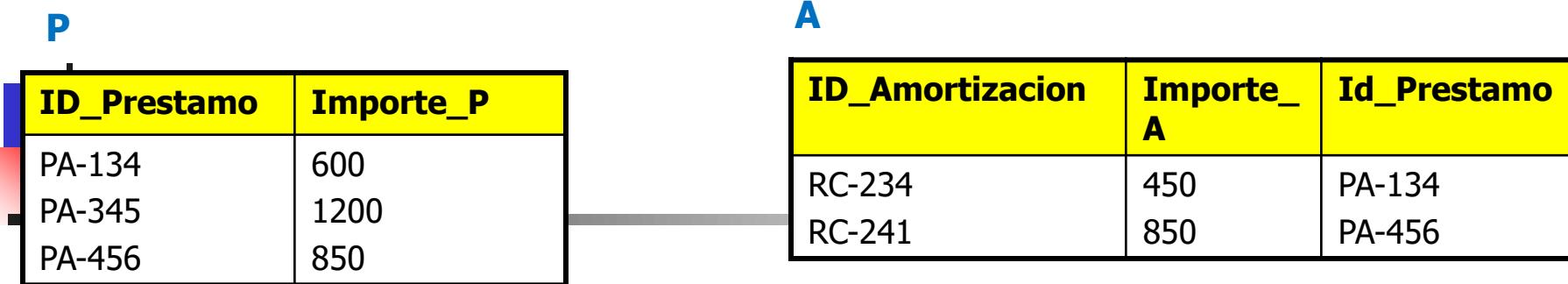
En la operación JOIN se pueden perder algunas tuplas, las que no están relacionadas en las dos relaciones.

Mientras que la operación LEFT JOIN, mantiene todas las tuplas de la relación de la izquierda y completa con valores nulos los atributos que no existan.

R1 \bowtie R2

Composición Externa

Dadas las siguientes tablas, determinar la relación de préstamos con sus respectivos saldos



¿Es la siguiente sentencia la solución?

$\pi_{P.ID_Prestamo, P.\text{Importe}, (P.\text{Importe} - A.\text{Importe}) \text{ as Saldo}} (P \bowtie A)$

Composición Externa...

El producto cartesiano genera la siguiente tabla

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

El juntar natural ($P.ID_Prestamo = A.ID_Prestamo$) selecciona las tuplas que no están subrayadas

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

Composición Externa...

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

En Consecuencia la relación resultante es:

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

El préstamo PA-345 ha sido ignorado debido a que no empareja con ninguna de las tuplas de la tabla de AMORTIZACION

Composición Externa...

Este tipo de composición se utiliza cuando las cardinalidades mínimas son 0 (No son obligatorios).



Como el modelo anterior indica que hay PRESTAMOS que no necesariamente están relacionados con AMORTIZACIONES, entonces estas instancias desaparecerán cuando se hace una operación de **Juntar simple (JOIN)**; en su lugar se debe realizar una operación **JOIN LEFT**

Composición Externa...

La operación LEFT JOIN

P

Produce la siguiente relación

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-345	1200	null	null	null
PA-456	850	RC-241	850	PA-456

Composición Externa...

Finalmente, la solución al problema propuesto es:

π P.ID_Prestamo, P. Importe, (P.Importe – A.Importe) as Saldo (P \bowtie A)

La relación resultante es:

P.ID_Prestamo	P.Importe	Saldo
PA-134	600	150
PA-345	1200	null
PA-456	850	0

Composición externa (Ejercicios)

Dada la siguiente Base de datos:

CARRERA(IdCarrera, NombreCarrera)

ALUMNO(IdAlumno, AP, AM, Nombres, IdCarrera)

ASIGNATURA(IdAsignatura, NombreAsignatura, Cat, Cred, PreRequisito)

MATRICULA(Semestre, IdAsignatura, IdAlumno, Nota)

Composición externa - Ejercicios:

1.- Obtener la relación de estudiantes de ingeniería informática con el número de créditos aprobados.

$R(IdAlumno, AP, AM, Nombres, NroCreditos)$

-- Obtener la relación de alumnos de Ingeniería Informática

$T1 \leftarrow \sigma_{IdCarrera = 'IN'}(ALUMNO)$

-- Obtener la relación de matrículas con asignaturas aprobadas

$T2 \leftarrow \sigma_{Nota > 14}(MATRICULA)$

-- Obtener la relación de asignaturas aprobadas por los alumnos de Ingeniería Informática

$T3 \leftarrow \pi_{T2.IdAlumno, T2.IdAsignatura}(T2 \bowtie T1)$

-- Agregar a asignaturas aprobadas el número de créditos

$T4 \leftarrow \pi_{T3.IdAlumno, T3.IdAsignatura, ASIGNATURA.Cred}(T3 \bowtie ASIGNATURA)$

-- Sumar el número de créditos de cada alumno

$T5 \leftarrow IdAlumno \text{ } \text{G} \text{ } \text{SUM}(\text{Cred}) \text{ as } NroCreditos(T4)$

-- Completar información de Alumnos

$R \leftarrow \pi_{T1.IdAlumno, AP, AM, Nombres, EsNulo(NroCreditos, 0)}(T1 \bowtie T5)$

Notar que en la última sentencia se requiere composición externa (Left Join), porque puede haber alumnos de Ingeniería Informática que no hayan aprobado ninguna asignatura y no estén en la tabla T5.

Álgebra Relacional... DIVISION

R

PROVEEDOR	PRODUCTO
Alfa	Impresora
Alfa	LapTop
Alfa	Scanner
Alfa	DVD
Omega	Scanner
Omega	LapTop
Delta	Impresora
Delta	DVD
Delta	LapTop
Delta	Scanner
Beta	Scanner
Beta	DVD
Beta	Impresora

S

PRODUCTO
Impresora
LapTop
Scanner
DVD

Determinar la relación de proveedores que proveen todos los Productos

Álgebra Relacional...

1.- Obtener relación de Proveedores

$$T1 \leftarrow \pi_{PROVEEDOR}(R)$$

T1

PROVEEDOR
Alfa
Omega
Delta
Beta

Álgebra Relacional...

2.- Obtener el producto de Proveedores y Productos

T2

T2 \leftarrow T1 X S

PROVEEDOR	PRODUCTO
Alfa	Impresora
Alfa	LapTop
Alfa	Scanner
Alfa	DVD
Omega	Impresora
Omega	LapTop
Omega	Scanner
Omega	DVD
Delta	Impresora
Delta	LapTop
Delta	Scanner
Delta	DVD
Beta	Impresora
Beta	LapTop
Beta	Scanner
Beta	DVD

Álgebra Relacional...

3.- Obtener Relación de Proveedores que no proveen todos los productos

T3 ← T2 - R

T3

PROVEEDOR	PRODUCTO
Omega	Impresora
Omega	DVD
Beta	LapTop

4.- Obtener Relación de Proveedores que no proveen todos los productos

T4 ← $\pi_{\text{PROVEEDOR}}(\text{T3})$

T4

PROVEEDOR
Omega
Beta

5.- Obtener Relación de Proveedores que proveen todos los productos

T5 ← T1 - T4

T5

PROVEEDOR
Alfa
Delta

Álgebra Relacional...

DIVISION

R / S

$$\pi_{\text{PROVEEDOR}}(R) - \pi_{\text{PROVEEDOR}}(\pi_{\text{PROVEEDOR}}(R) \times S - R)$$

Modificación de la base de datos

Relación constante:

Las relaciones constantes se escriben poniendo una relación de sus tuplas entre llaves. Por ejemplo:

{(090123,Paz,Arce,Ana,IN)(080533,Díaz,Cusi,Pedro,CO)}

Modificación de la base de datos

Borrado:

Se expresa mediante:

$$r \leftarrow r - E$$

Donde:

r es una relación y E es una consulta del álgebra relacional

Ejemplo:

Dada la tabla:

Alumno(CodAlumno, AP, AM, Nombres, CodCP)

Borrar la relación de alumnos de la carrera profesional 'CO'

Alumno \leftarrow Alumno - $\sigma_{ALUMNO.CodCP = 'CO'}(Alumno)$

Modificación de la base de datos

Inserción:

Se expresa mediante:

$$r \leftarrow r \cup E$$

Donde:

r es una relación y E es una expresión del álgebra relacional

Ejemplo:

Dada la tabla:

Alumno(CodAlumno, AP, AM, Nombres, CodCP)

Insertar una tupla a la tabla Alumno.

Alumno \leftarrow Alumno \cup {('080024', 'Meza', 'Tito', 'Luis', 'IN')}

Modificación de la base de datos

Actualización:

Se realiza mediante la proyección generalizada y se expresa mediante:

$$r \leftarrow \pi_{F1, F2, \dots, Fn}(\sigma_p(r)) \cup (r - \sigma_p(r))$$

Donde:

r es una relación, F una expresión de la operación proyección generalizada y p una condición de la operación selección.

Ejemplo:

Dada la tabla:

Alumno(CodAlumno, AP, AM, Nombres, CodCP)

Actualizar la carrera profesional 'CO' a 'CC'.

T1 $\leftarrow \sigma_{\text{CodCP} = 'co'}(\text{ALUMNO})$

Alumno $\leftarrow \pi_{\text{CodAlumno}, \text{AP}, \text{AM}, \text{Nombres}, 'cc'}(T1) \cup (\text{Alumno} - T1)$

Cálculo Relacional



Fue propuesto por Codd en 1971 como alternativa al álgebra. La diferencia fundamental entre un lenguaje algebraico y un lenguaje predicativo (Denominado así porque utiliza el cálculo de predicados para la formulación de consultas), es que en el primero hay que especificar qué operadores se tienen que aplicar a las relaciones para obtener el resultado, mientras que en los segundos sólo es preciso indicar cuál es el resultado que se quiere obtener, expresándolo mediante cálculo de predicados de primer orden.

Cálculo Relacional...

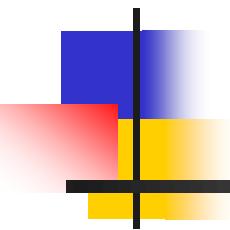
TIPOS

- **Orientados a la tupla**, en los que una variable se interpreta como si representase las tuplas de una relación.

Ejemplo el lenguaje ALPHA (Nunca fue implementado)

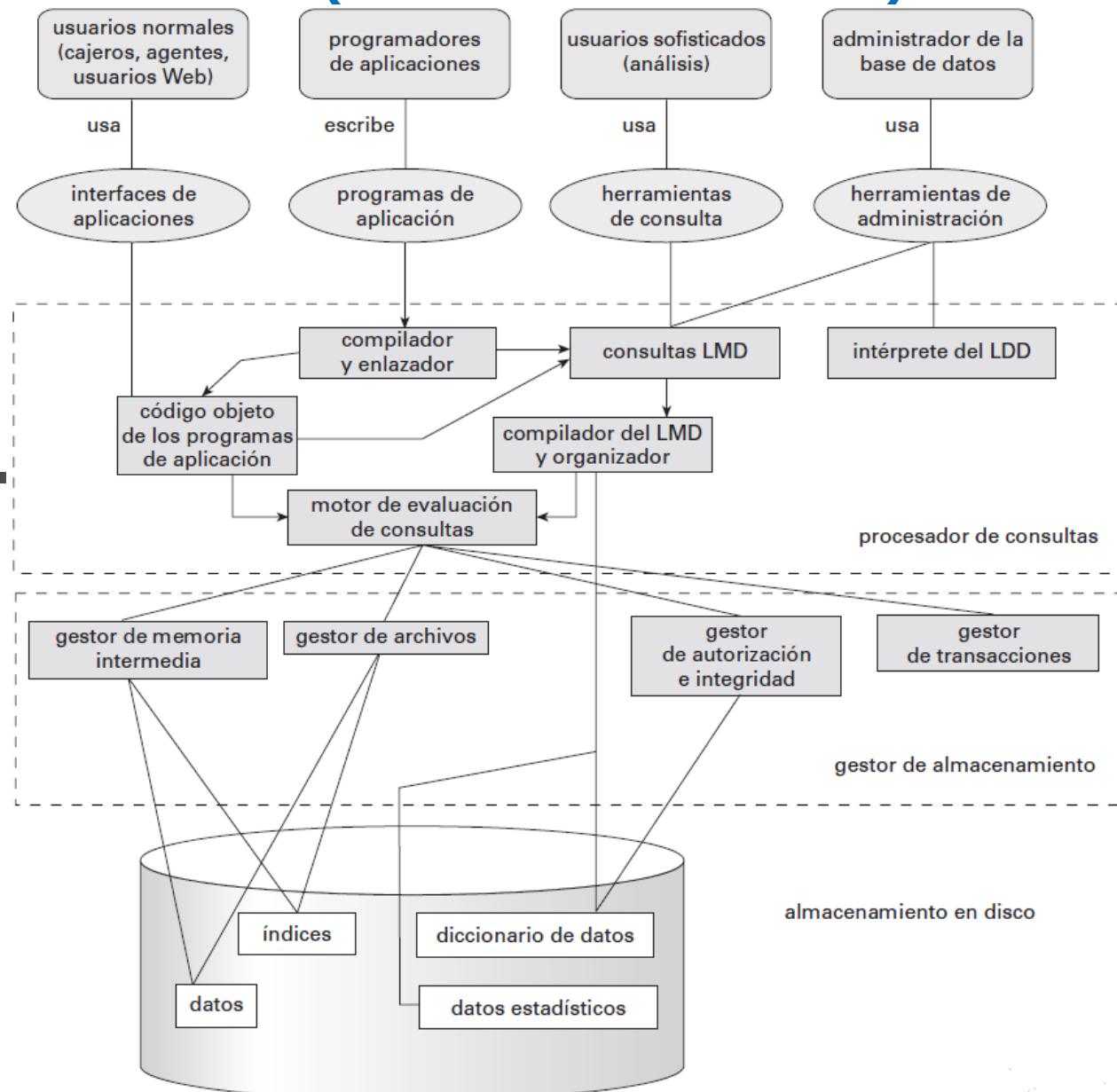
- **Orientados al dominio**, en los que una variable se interpreta como si representase los valores de un dominio.

El ejemplo más característico es el lenguaje QBE (Query By Example), desarrollado en Yorktown Heights por IBM.

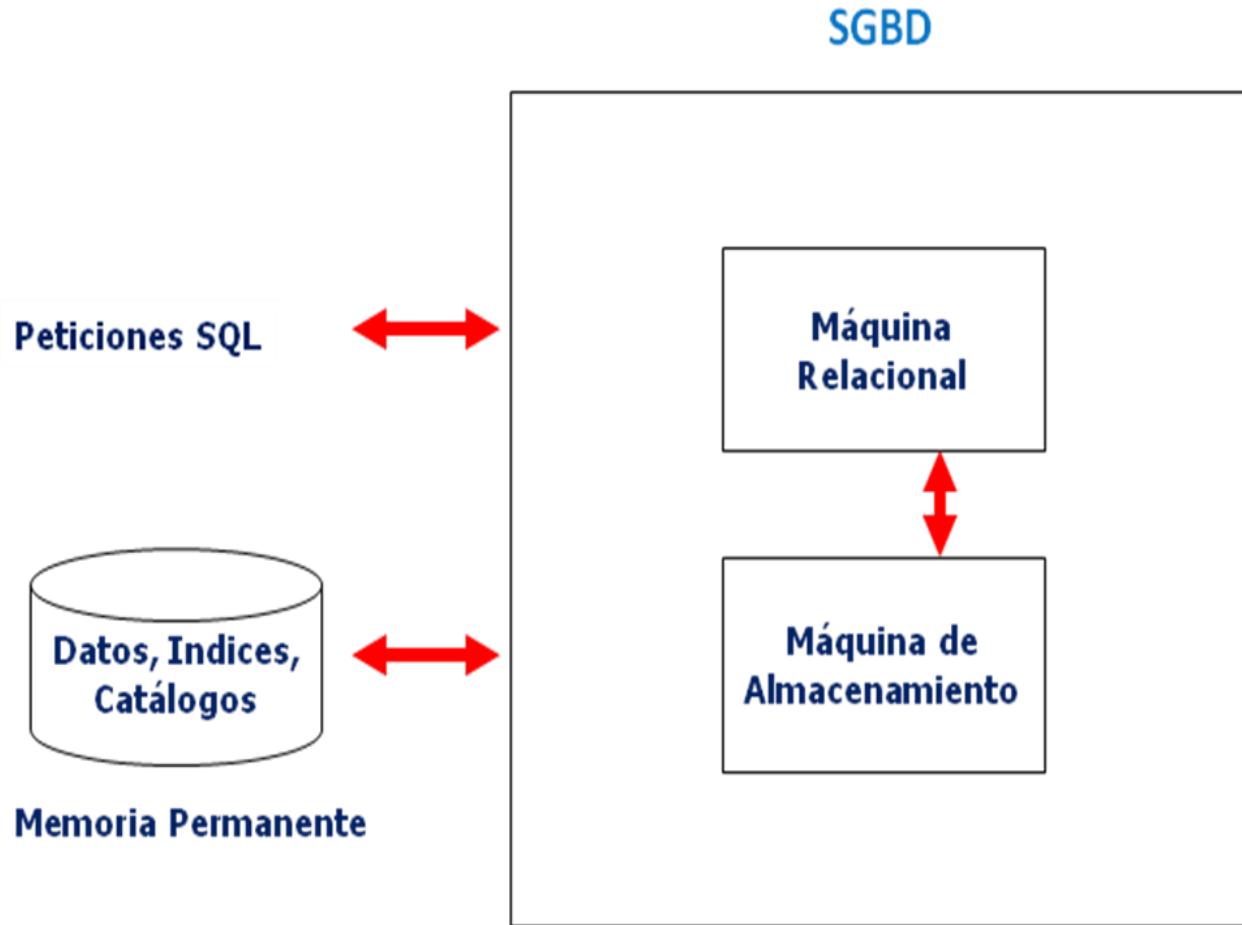
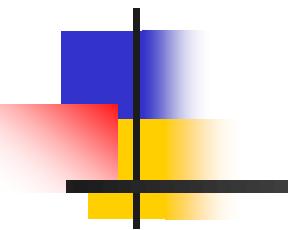


ALMACENAMIENTO DE DATOS Y CONSULTAS

Ideas preliminares (Estructura de un SGBD)

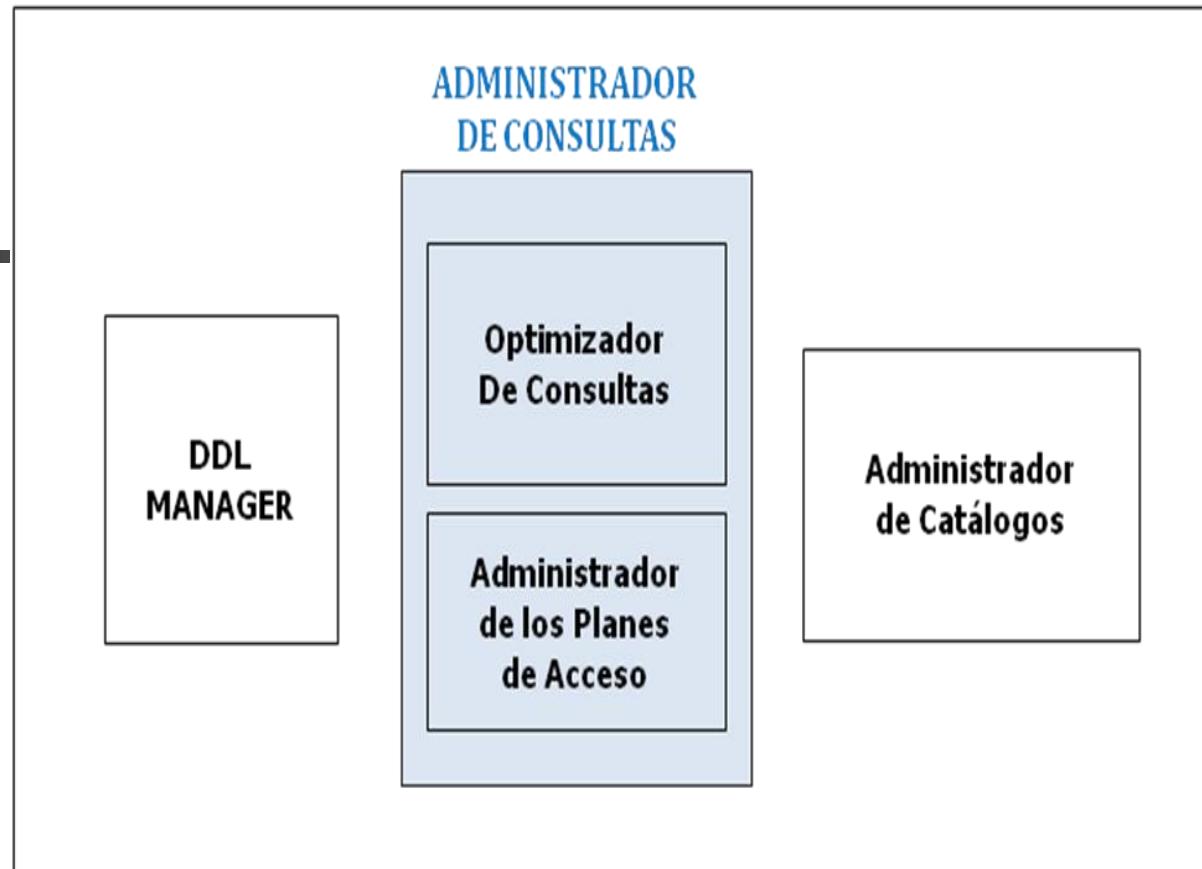


Ideas preliminares (Diagrama de bloques de la estructura de un SGBD)



Ideas preliminares (Diagrama de bloques de la estructura de un SGBD)

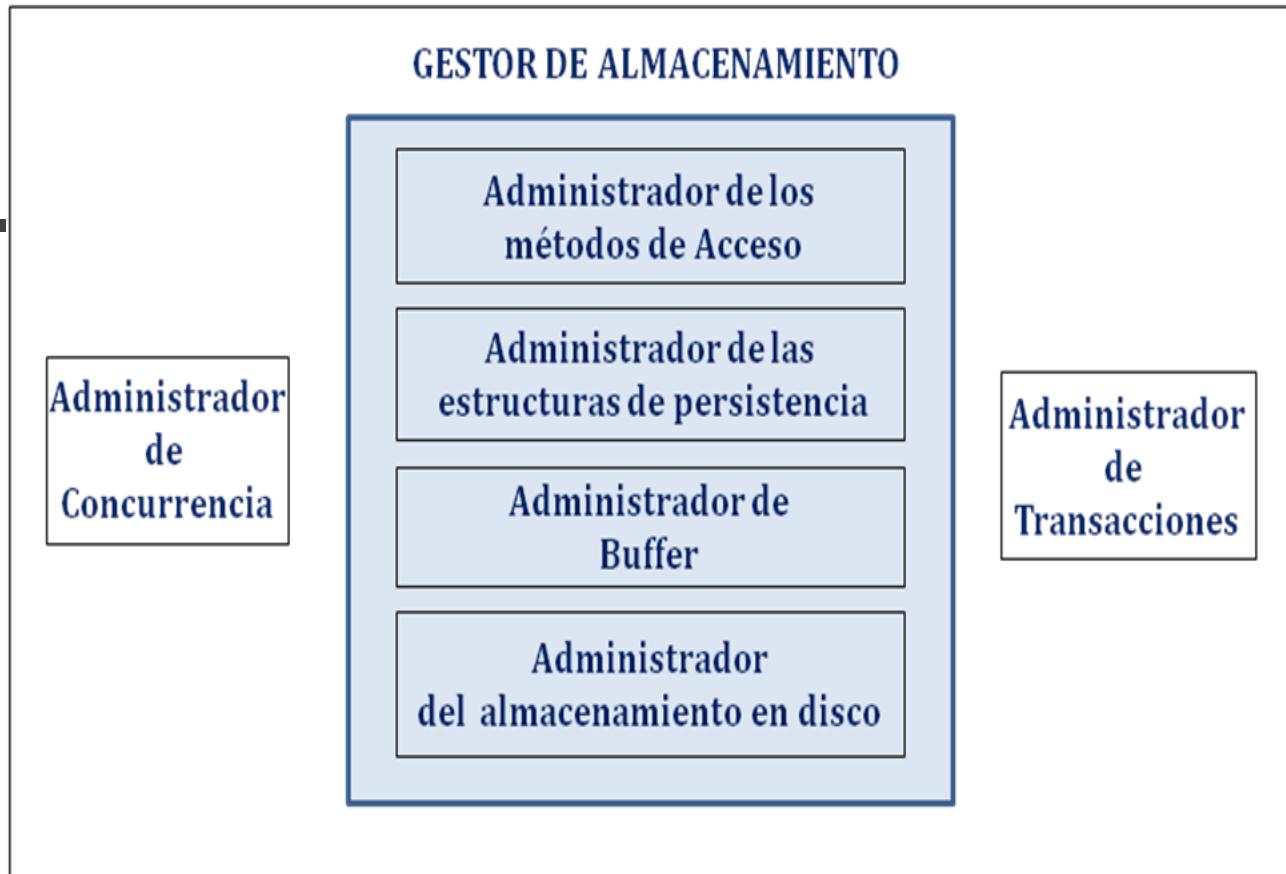
Máquina Relacional



Ideas preliminares

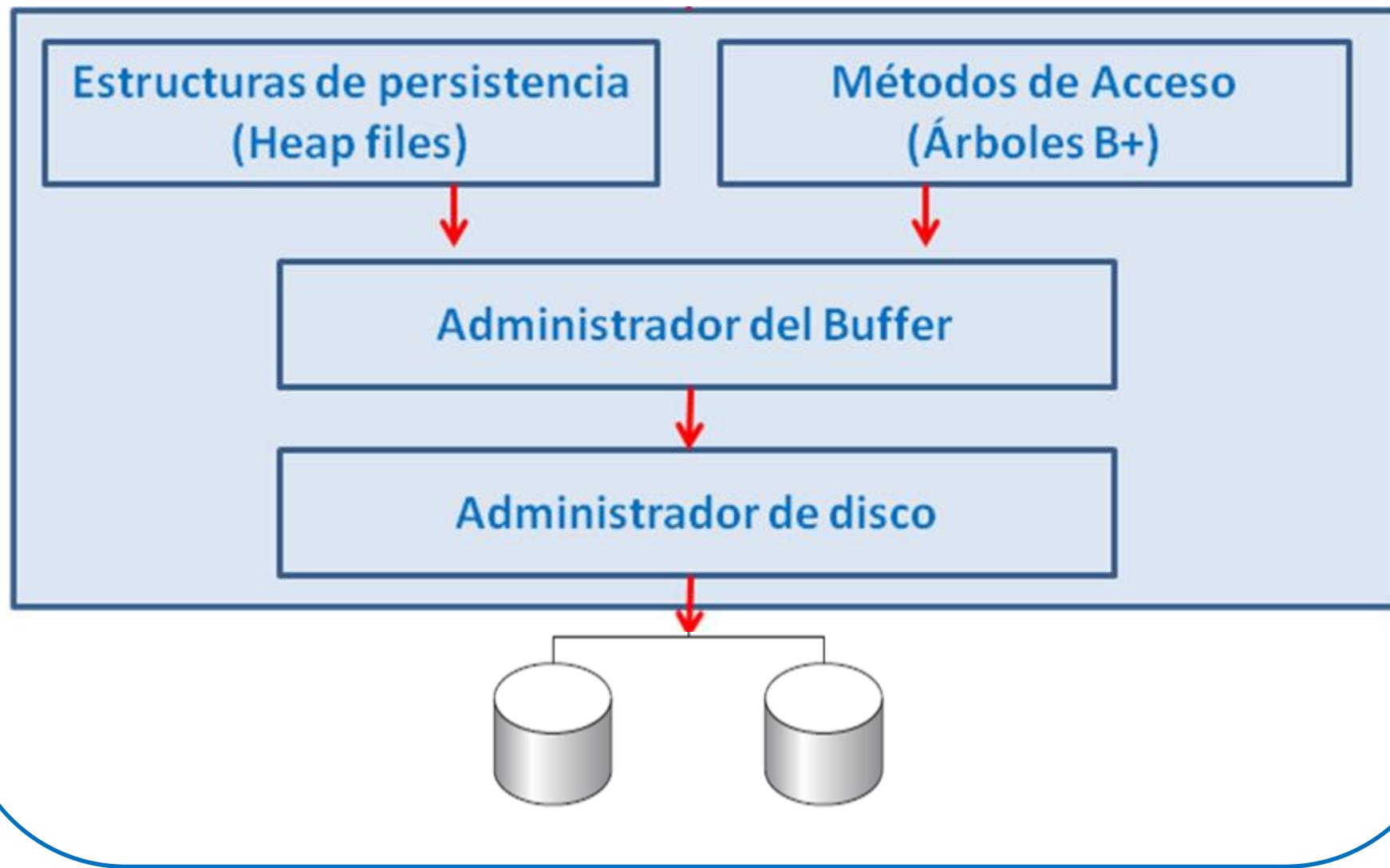
(Diagrama de bloques de la estructura de un SGBD)

Motor de almacenamiento



Almacenamiento y estructura de archivos

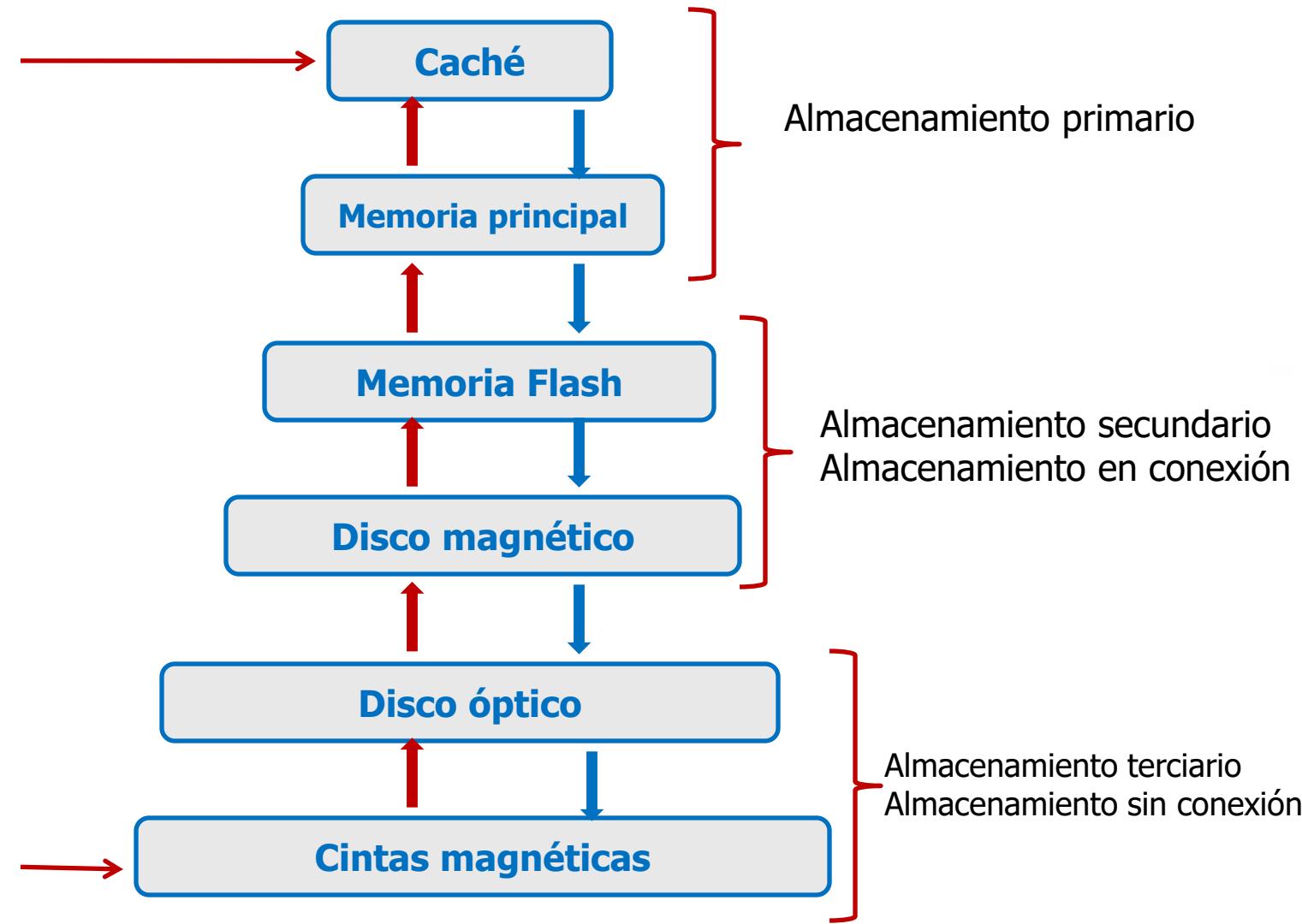
Máquina de almacenamiento



Almacenamiento y estructura de archivos

Visión general de los medios físicos de almacenamiento

Más rápido y
más caro



Más lento y
más barato

Almacenamiento y estructura de archivos

Visión general de los medios físicos de almacenamiento

- **Caché.**- Es pequeña, es el mecanismo de almacenamiento más rápido y costoso; su uso lo gestiona el hardware y no el SGBD.
- **Memoria principal.**- Las instrucciones máquina utilizan la memoria principal como el medio de almacenamiento para operar con los datos. Se pierde su contenido en caso de caídas del sistema.
- □ **Memoria flash.**- Se diferencian de la memoria principal en que los datos no se pierden en caso de caídas de energía. (Cámaras digitales, USB, etc.)
- **Discos magnéticos.**- Es de acceso directo. Es principal medio de almacenamiento persistente. Generalmente se guarda en ellos toda la base de datos.
- **Almacenamiento óptico.**- También es de acceso directo. CD y DVD.
- **Almacenamiento en cinta.**- Es de acceso secuencial. Es más barata que los disco y se utiliza principalmente para copias de seguridad.

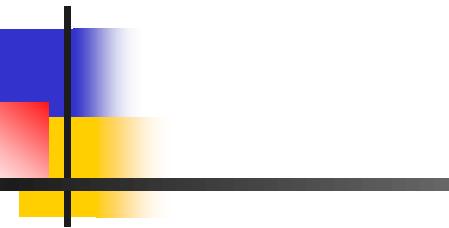
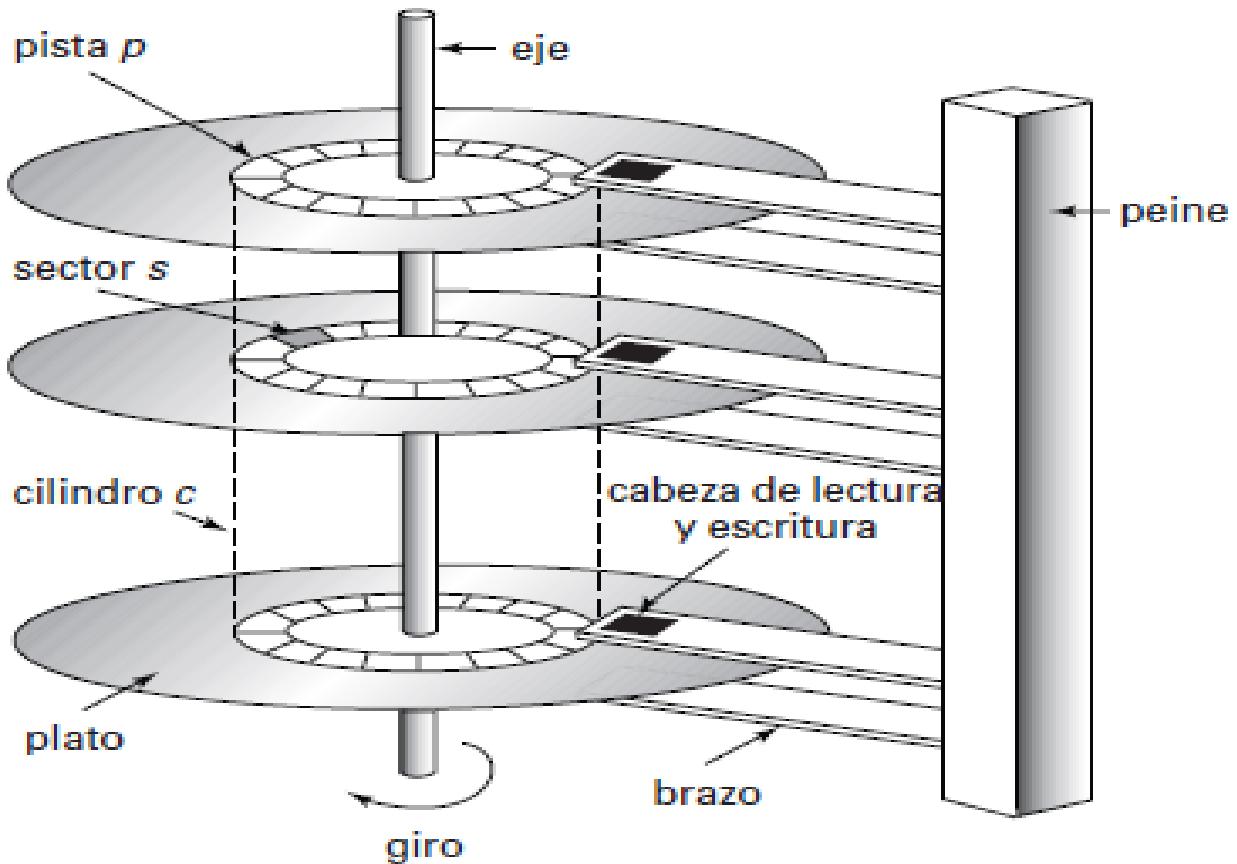
Almacenamiento y estructura de archivos

Discos magnéticos.- Proporcionan la mayor parte del almacenamiento secundario.

- **IDE.- Integrated Drive Electronics.**
- **ATA.- AT Attachment.**
- **SATA.- ATA serie, Serial ATA.**
- **SCSI.- Small Computer System Interconnect.**
- **RAID.- Redundant arrays of independent disk**

Almacenamiento y estructura de archivos

Discos magnéticos



- Está organizado en platos. Cada plato tiene dos superficies cubiertas por un material magnético.
- La superficie del disco se divide a efectos lógicos en pistas.
- Cada pista se subdivide en sectores.
- Un sector es la unidad mínima de información que se puede leer o escribir en el disco.
- En los discos actuales una lectura/escritura implica varios sectores (Cluster - Bloque)

Almacenamiento y estructura de archivos

Organización de los archivos

Desde la perspectiva de los lenguajes de programación, se tiene:

- **Archivos de texto.**

Son de acceso secuencial. Cada texto termina con una marca de retorno de carro

- **Archivo con tipo (de acceso directo).**

Son de acceso directo, en virtud de que almacenan registro de longitud fija.

- **Archivos sin tipo.**

Se manejan en bloques de bytes .

Almacenamiento y estructura de archivos

Organización de los archivos : Registros de longitud fija

registro 0	C-102	Navacerrada	400
registro 1	C-305	Collado Mediano	350
registro 2	C-215	Becerril	700
registro 3	C-101	Centro	500
registro 4	C-222	Moralzarzal	700
registro 5	C-201	Navacerrada	900
registro 6	C-217	Galapagar	750
registro 7	C-110	Centro	600
registro 8	C-218	Navacerrada	700

Lo utilizan algunos SGBD (Paradox). Cada tabla es un archivo

Almacenamiento y estructura de archivos

Organización de los archivos: Registros de longitud fija

Presenta inconvenientes:

- Resulta difícil borrar registros.
- La longitud de los registros generalmente no es múltiplo del tamaño del bloque.
Algún registro se saltará los límites de los bloques.

Al borrar un registro se puede desplazar los registros situados por debajo del registro borrado para ocupar el espacio dejado por éste. Puede implicar el desplazamiento de un gran número de registros.

registro 0	C-102	Navacerrada	400
registro 1	C-305	Collado Mediano	350
registro 2	C-215	Becerril	700
registro 3	C-101	Centro	500
registro 4	C-222	Moralzarzal	700
registro 5	C-201	Navacerrada	900
registro 6	C-217	Galapagar	750
registro 7	C-110	Centro	600
registro 8	C-218	Navacerrada	700

Antes de borrar el registro 2

registro 0	C-102	Navacerrada	400
registro 1	C-305	Collado Mediano	350
registro 3	C-101	Centro	500
registro 4	C-222	Moralzarzal	700
registro 5	C-201	Navacerrada	900
registro 6	C-217	Galapagar	750
registro 7	C-110	Centro	600
registro 8	C-218	Navacerrada	700

Después de borrar el registro 2

Almacenamiento y estructura de archivos

Organización de los archivos: Registros de longitud fija

Para evitar el desplazamiento de un gran número de registros, se puede desplazar simplemente el último registro del archivo al espacio ocupado por el registro borrado.

registro 0	C-102	Navacerrada	400
registro 1	C-305	Collado Mediano	350
registro 2	C-215	Becerril	700
registro 3	C-101	Centro	500
registro 4	C-222	Moralzarzal	700
registro 5	C-201	Navacerrada	900
registro 6	C-217	Galapagar	750
registro 7	C-110	Centro	600
registro 8	C-218	Navacerrada	700

Antes de borrar el registro 2

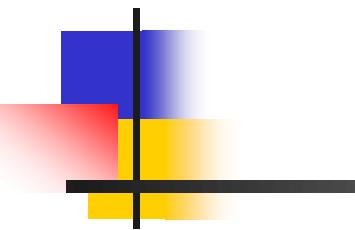
registro 0	C-102	Navacerrada	400
registro 1	C-305	Collado Mediano	350
registro 8	C-218	Navacerrada	700
registro 3	C-101	Centro	500
registro 4	C-222	Moralzarzal	700
registro 5	C-201	Navacerrada	900
registro 6	C-217	Galapagar	750
registro 7	C-110	Centro	600

Después de borrar el registro 2

Almacenamiento y estructura de archivos

Organización de los archivos: Registros de longitud fija

Considerando que las operaciones de inserción tienden a ser más frecuentes que las de borrado, resulta aceptable no desplazar registros, sino, dejar libre el espacio dejado por los registros borrados y esperar a una inserción posterior antes de volver a utilizar ese espacio.



cabecera

registro 0

registro 1

registro 2

registro 3

registro 4

registro 5

registro 6

registro 7

registro 8

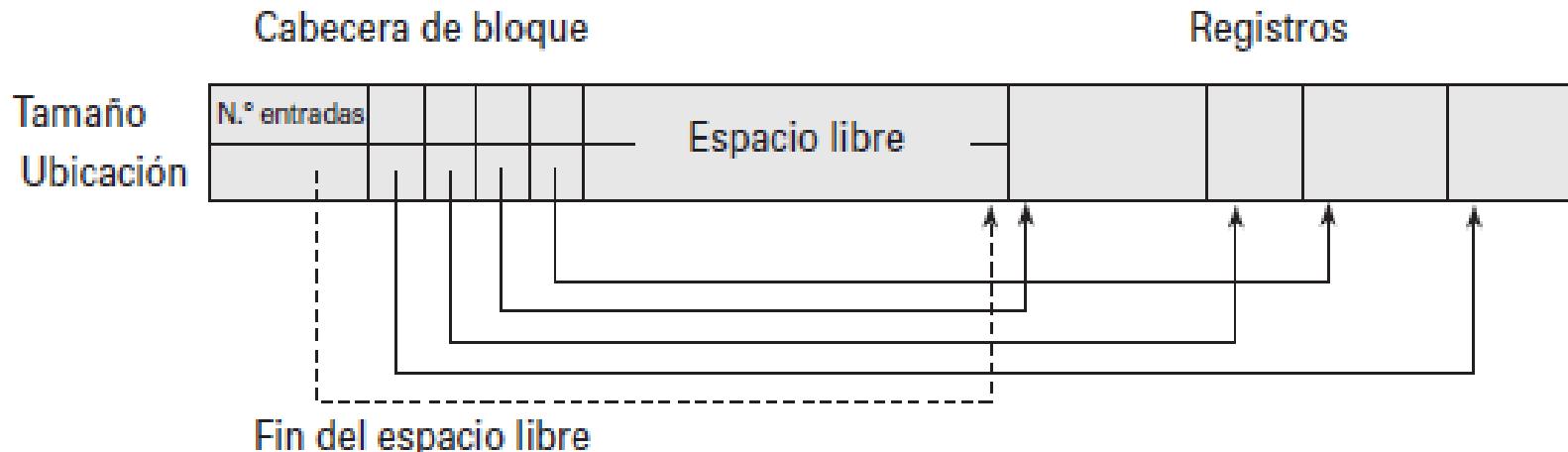
C-102	Navacerrada	400	
C-215	Becerril	700	
C-101	Centro	500	
C-201	Navacerrada	900	
C-110	Centro	600	
C-218	Navacerrada	700	

Almacenamiento y estructura de archivos

Organización de los archivos. Registros de longitud variable

Existen diferentes técnicas para implementarlos, por ejemplo: **Estructura de páginas con ranuras**, que utiliza habitualmente registros organizados en bloques. Cada bloque contiene:

- El número de elementos del registro de cabecera
- El final del espacio vacío del bloque
- Un array cuyas entradas contienen la ubicación y el tamaño de cada registro.



Almacenamiento y estructura de archivos

Organización de Bases de datos

Un sistema de base de datos relacionales necesita tener datos sobre las relaciones. Esta información se denomina **diccionario de datos** o **catálogo del sistema**. Entre los tipos de información que debe guardar tenemos los siguientes:

- El nombre de las relaciones
- El nombre de los atributos de cada relación
- El dominio y la longitud de los atributos
- El nombre de las vistas definidas en la base de datos, y la definición de esas vistas.
- Las restricciones de integridad (Por ejemplo, las restricciones de las claves)

Almacenamiento y estructura de archivos

Organización de Bases de datos

Ejemplo de la estructura de los metadatos

Metadatos_relación(nombre_relación, número_atributos, organización_almacenamiento, ubicación)

Metadatos_atributos(nombre_atributo, nombre_relación, tipo_dominio, posición, longitud)

Metadatos_usuarios (nombre_usuario, contraseña_cifrada, grupo)

Metadatos_índices (nombre_índice, nombre_relación, tipo_índice, atributos_índice)

Metadatos_vistas(nombre_vistas, definición)

Indexación y asociación

Conceptos básicos

Acceder a una parte de la información de una tabla es un proceso lento si se utiliza algoritmos que recorren registro a registro toda la tabla.

Se requieren mecanismos que permitan localizar y acceder directamente a los registros .

Por ejemplo en una guía telefónica es fácil acceder a la información en base a los apellidos del abonado. No resulta sencillo acceder a los nombres a partir del número telefónico.

Nombres	Dirección	Teléfono
1 Arce Díaz Pedro	AV. Cusco 435	246389
2 Bueno Guerra Eva	Av. Pardo 345	228947
3 Cuno Meza Luis	Av. de la Cultura 789	237510
4 Dávila Fuentes Ana	Jr. Paruro 478	247800
5 Mena Casas Carlota	Calle Oropesa 123	224578
6 Salcedo Husno Rosa	Av. Pachacuteq 478	245511
7 Yañez Cuno Carla	Av. Cusco 3456	226688

Indexación y asociación

Índices

Los índices de los sistemas de base de datos juegan el mismo papel que los índices de los libros en las bibliotecas, facilitan el acceso a la información.

Hay dos tipos de índices:

- Índices ordenados
- Índices asociativos

Criterios de valoración

- Tipos de acceso
- Tiempo de acceso
- Tiempo de inserción
- Tiempo de borrado
- Espacio adicional requerido

Indexación y asociación

Índices Ordenados

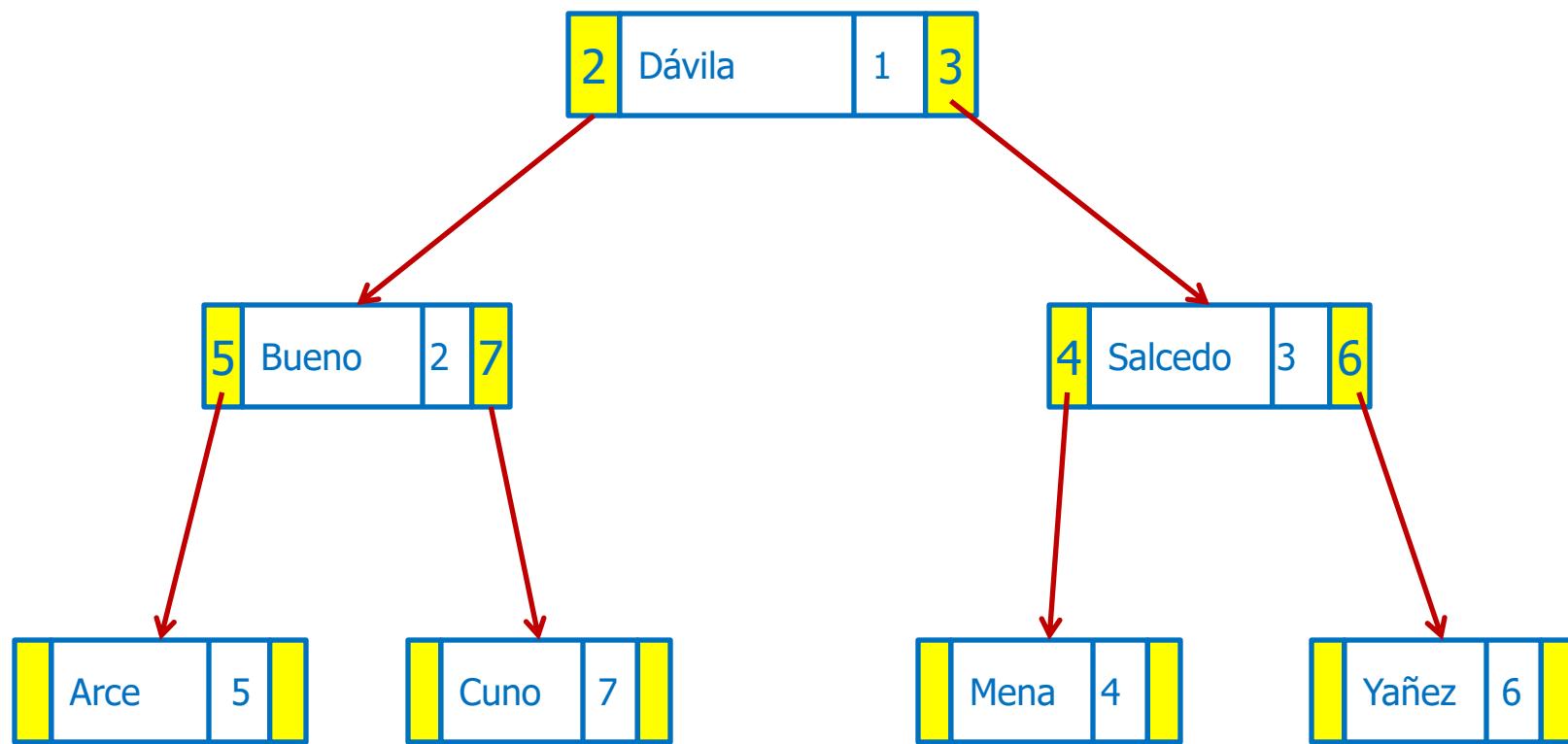
Los registros no necesariamente están ordenados

	Nombres	Dirección	Teléfono
1	Dávila Fuentes Ana	Jr. Paruro 478	247800
2	Bueno Guerra Eva	Av. Pardo 345	228947
3	Salcedo Husno Rosa	Av. Pachacuteq 478	245511
4	Mena Casas Carlota	Calle Oropesa 123	224578
5	Arce Díaz Pedro	AV. Cusco 435	246389
6	Yañez Cuno Carla	Av. Cusco 3456	226688
7	Cuno Meza Luis	Av. de la Cultura 789	237510

Indexación y asociación

Índices Ordenados

Árboles binarios de búsqueda (AVL)



Sistemas de Base de Datos

Indexación y asociación

Índices Ordenados

Archivo de datos

	Nombres	Dirección	Teléfono
1	Dávila Fuentes Ana	Jr. Paruro 478	247800
2	Bueno Guerra Eva	Av. Pardo 345	228947
3	Salcedo Husno Rosa	Av. Pachacuteq 478	245511
4	Mena Casas Carlota	Calle Oropesa 123	224578
5	Arce Díaz Pedro	AV. Cusco 435	246389
6	Yañez Cuno Carla	Av. Cusco 3456	226688
7	Cuno Meza Luis	Av. de la Cultura 789	237510

Índice por Nombres (Árbol AVL)

	H.IZQ	Clave	Nro.Reg.	H.Der
1	2	Dávila	1	3
2	5	Bueno	2	7
3	4	Salcedo	3	6
4		Mena	4	
5		Arce	5	
6		Yañez	6	
7		Cuno	7	

Sistemas de Base de Datos

Indexación y asociación

Índices Ordenados

Archivo de datos

	Nombres	Dirección	Teléfono
1	Dávila Fuentes Ana	Jr. Paruro 478	247800
2	Bueno Guerra Eva	Av. Pardo 345	228947
3	Salcedo Husno Rosa	Av. Pachacuteq 478	245511
4	Mena Casas Carlota	Calle Oropesa 123	224578
5	Arce Díaz Pedro	AV. Cusco 435	246389
6	Yañez Cuno Carla	Av. Cusco 3456	226688
7	Cuno Meza Luis	Av. de la Cultura 789	237510

Índice por Teléfono (Árbol BB)

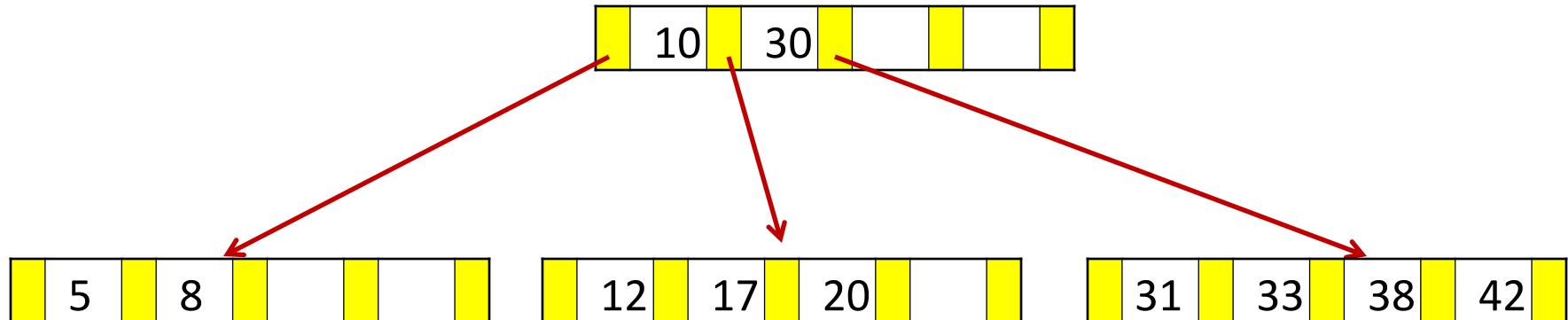
	H.IZQ	Clave	Nro.Reg.	H.Der
1	2	247800	1	
2	4	228947	2	3
3	7	245511	3	5
4		224578	4	6
5		246389	5	
6		226688	6	
7		237510	7	

Sistemas de Base de Datos

Indexación y asociación

Archivos de Índices de árbol B

- Los árboles-B son árboles de búsqueda cuyos nodos pueden tener un número múltiple de hijos.
- Fue desarrollado por "Rudolf Bayer" y "Eduard M. McCreight", que trabajan para la empresa "Boeing". ("Karl Unterauer").
- Cuando se crean árboles-B para indexar ficheros de datos en disco, se intenta minimizar el número de accesos a disco.
- Para calcular el ORDEN se usa como dato el tamaño del cluster. Un cluster es el bloque de disco más pequeño que se lee o se escribe en una operación de acceso a disco
- El ORDEN se ajusta de modo que el tamaño del nodo sea lo más próximo posible, menor o igual, al tamaño del cluster.



Indexación y asociación

Archivos de Índices de árbol B

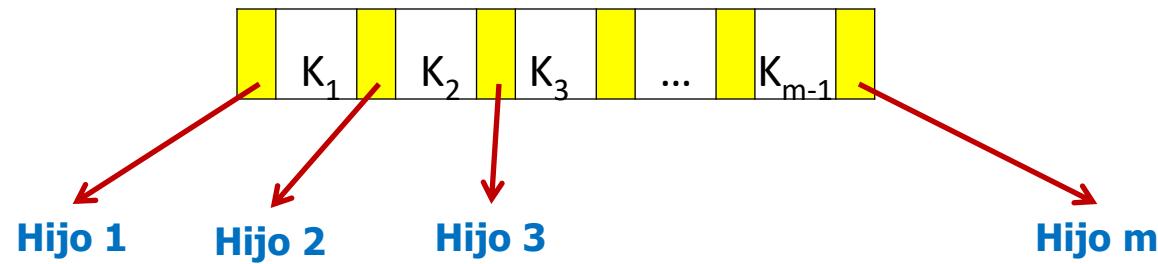
Propiedades de un árbol B de orden n:

- Cada nodo contiene a lo sumo $2n$ claves
- Cada nodo, excepto la raíz, contiene como mínimo n claves
- Cada nodo que no es una hoja tiene $m+1$ nodos descendientes, siendo m su ocupación.
 - Para la raíz: $2n \geq m \geq 1$
 - Para los demás nodos: $2n \geq m \geq n$
- Todos los nodos hojas están en el mismo nivel

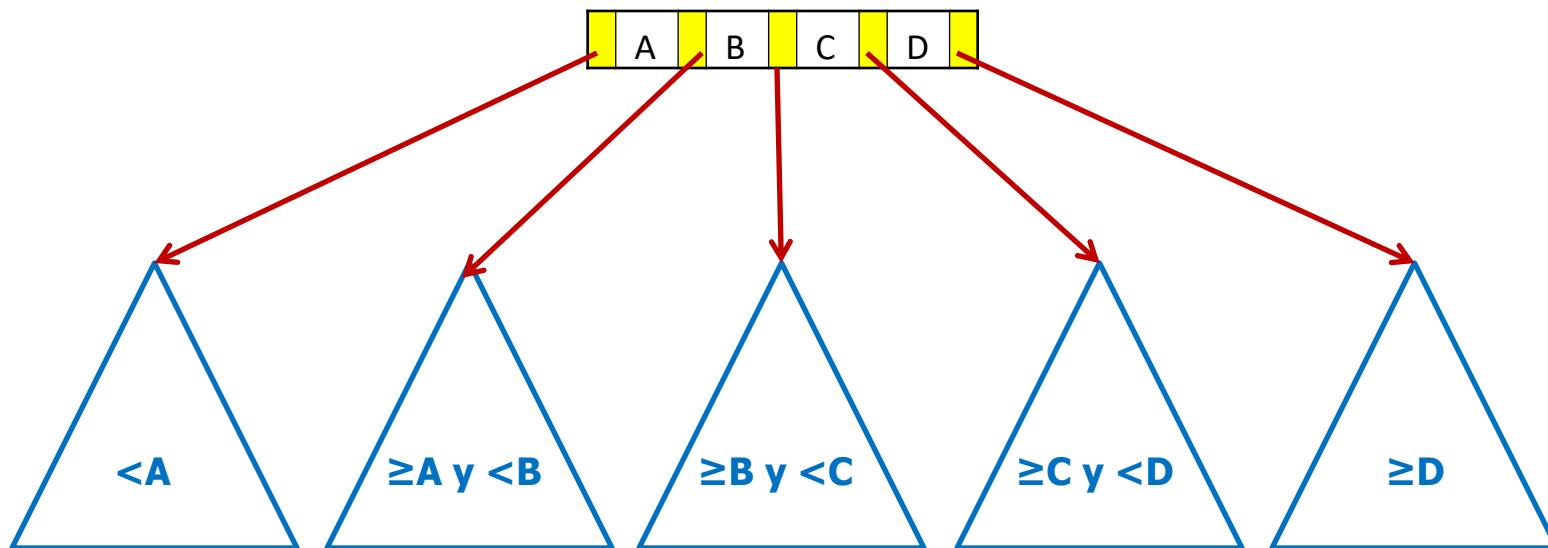
Indexación y asociación

Archivos de Índices de árbol B

Estructura de un árbol B



Ordenamiento de las claves en un árbol B

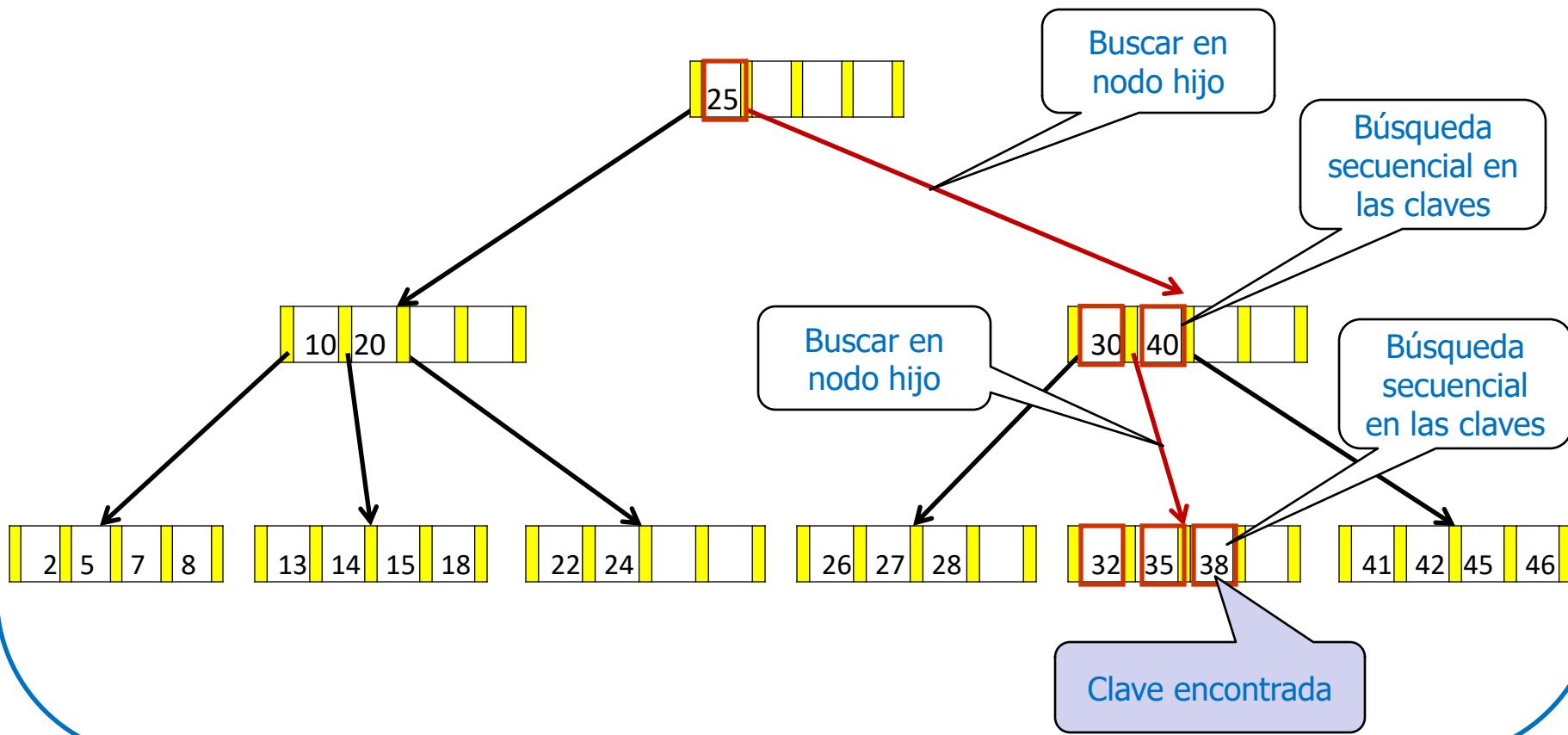


Indexación y asociación

Archivos de Índices de árbol B

Operación de búsqueda en un árbol B

Búsqueda del elemento 38



Indexación y asociación

Archivos de Índices de árbol B

Operación de Inserción en un árbol B

1 Buscar la hoja donde deberíamos encontrar el valor de la clave (si en este proceso de búsqueda encontramos la clave a insertar en algún nodo, el algoritmo termina). Si la clave no se encuentra en el árbol, entonces, estaremos en la hoja donde se debe insertar la clave.

2 Si ya estamos situados en el nodo a insertar, entonces, se pueden dar varias situaciones:

2.1 Si el nodo no está completo, la clave se inserta y termina el algoritmo.

2.2 Si el nodo está completo, entonces, dividimos en dos nuevos nodos conteniendo cada uno de ellos la mitad de las claves y tomando la clave del medio (mediana) para insertarla en el nodo padre.

2.3 Si el nodo padre está también completo, repetir el proceso desde el punto 2.1. hasta encontrar un nodo padre que no esté completo o llegar a la raíz.

2.4 En caso de que el nodo raíz también este completo, la altura del árbol aumenta en 1, creándose un nuevo nodo raíz con una única clave.

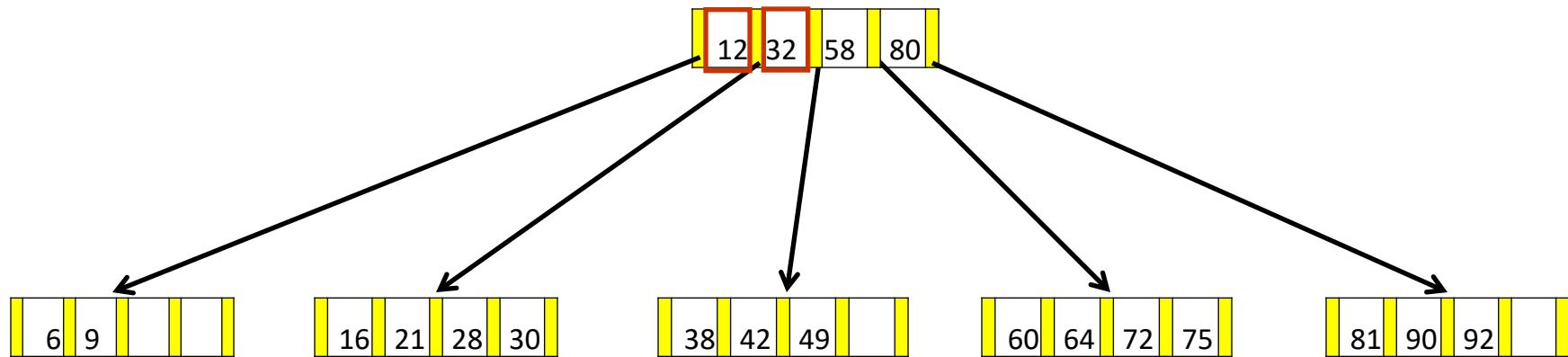
Indexación y asociación

Archivos de Índices de árbol B

Operación de Inserción en un árbol B

Considerando el siguiente árbol

Agregar la clave 18

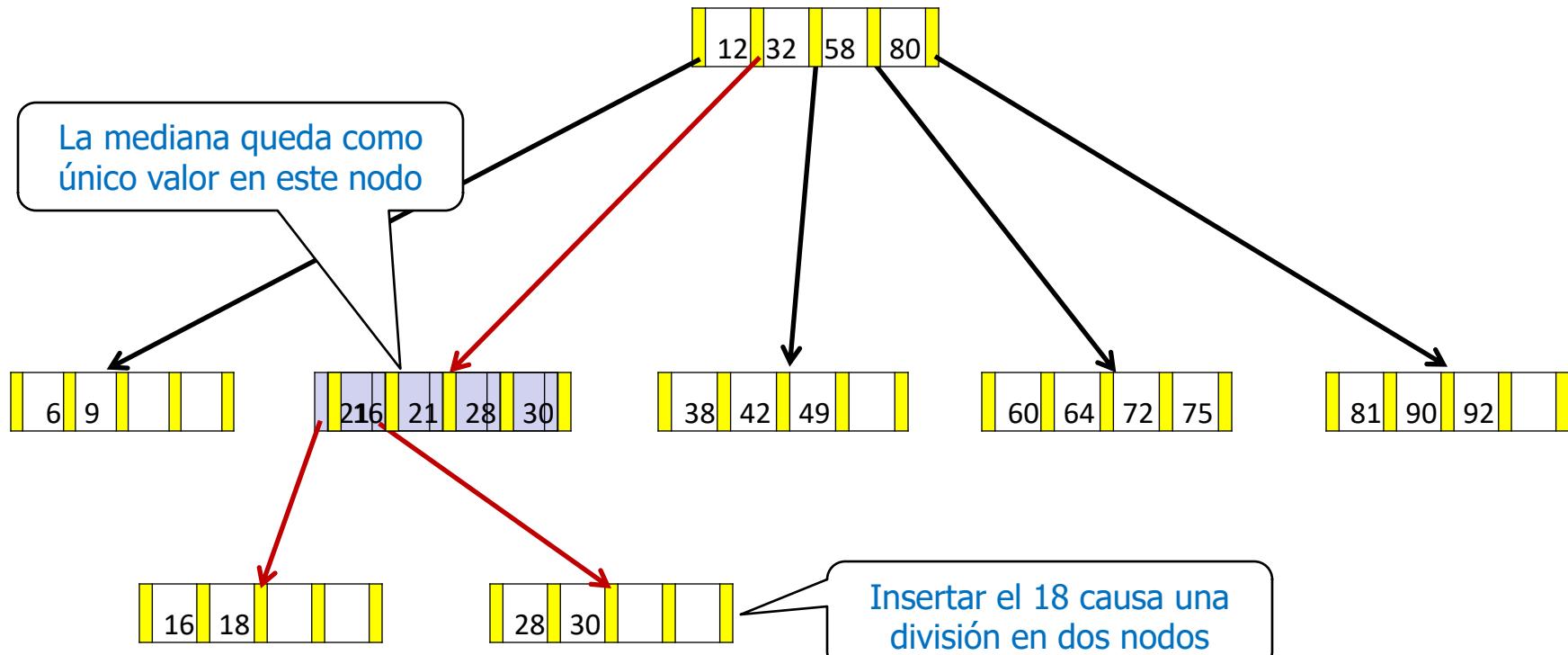


Indexación y asociación

Archivos de Índices de árbol B

Operación de Inserción en un árbol B

El nodo donde se debe insertar la clave 18 está completo, en consecuencia, se tendrá que dividir en dos nodos

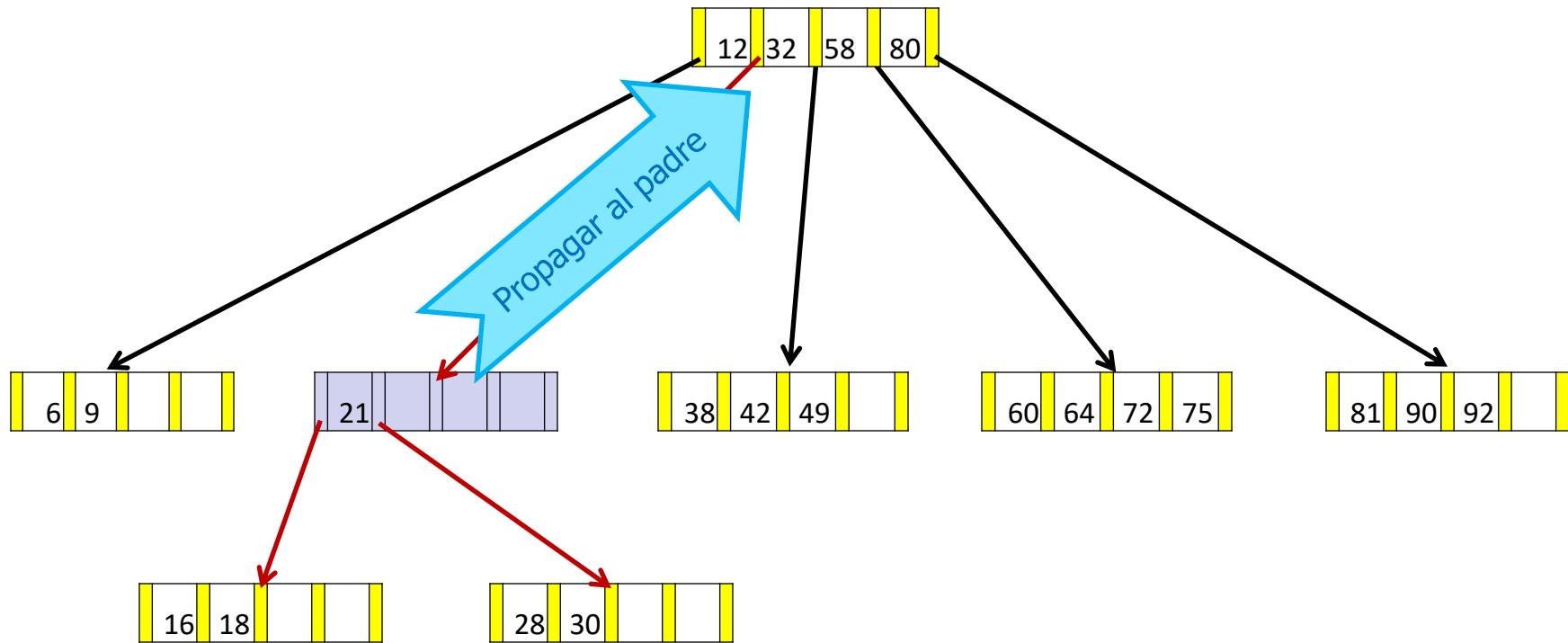


Indexación y asociación

Archivos de Índices de árbol B

Operación de Inserción en un árbol B

Luego, la clave 21 se debe agregar al nodo padre.

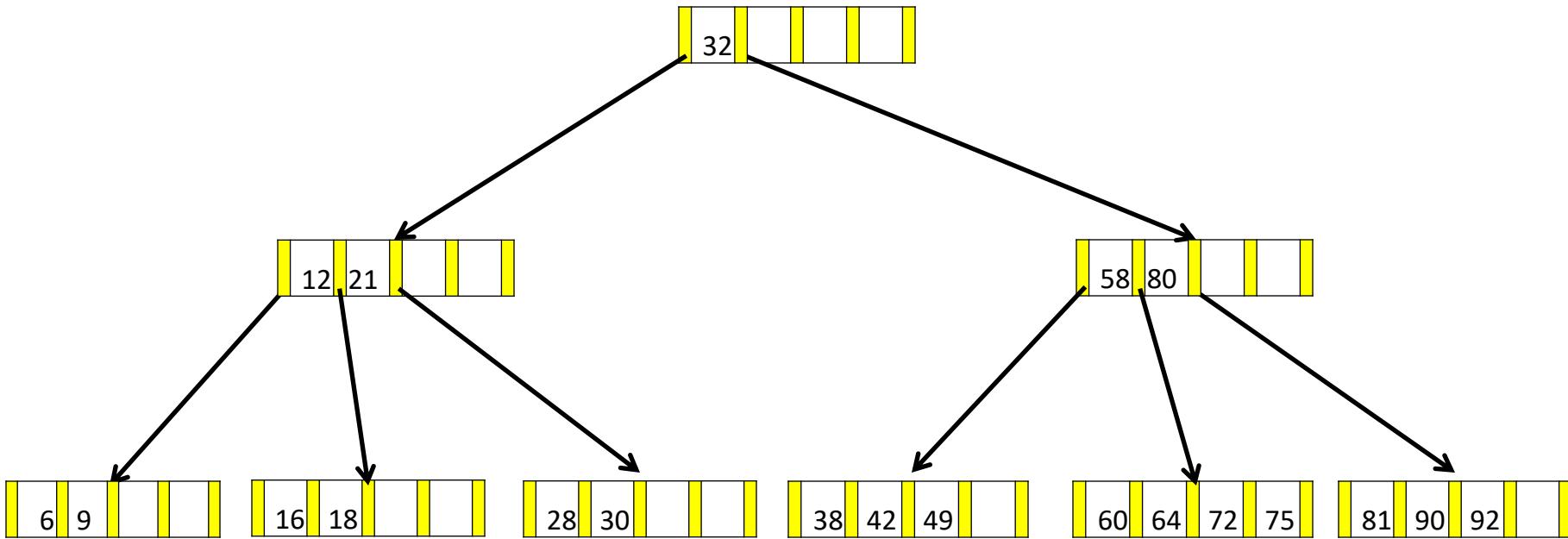


Indexación y asociación

Archivos de Índices de árbol B

Operación de Inserción en un árbol B

Luego, como el nodo padre también está completo, entonces éste se divide en dos nodos creándose un nuevo nodo para la raíz. Es en esta circunstancia que la altura del árbol aumenta en 1.



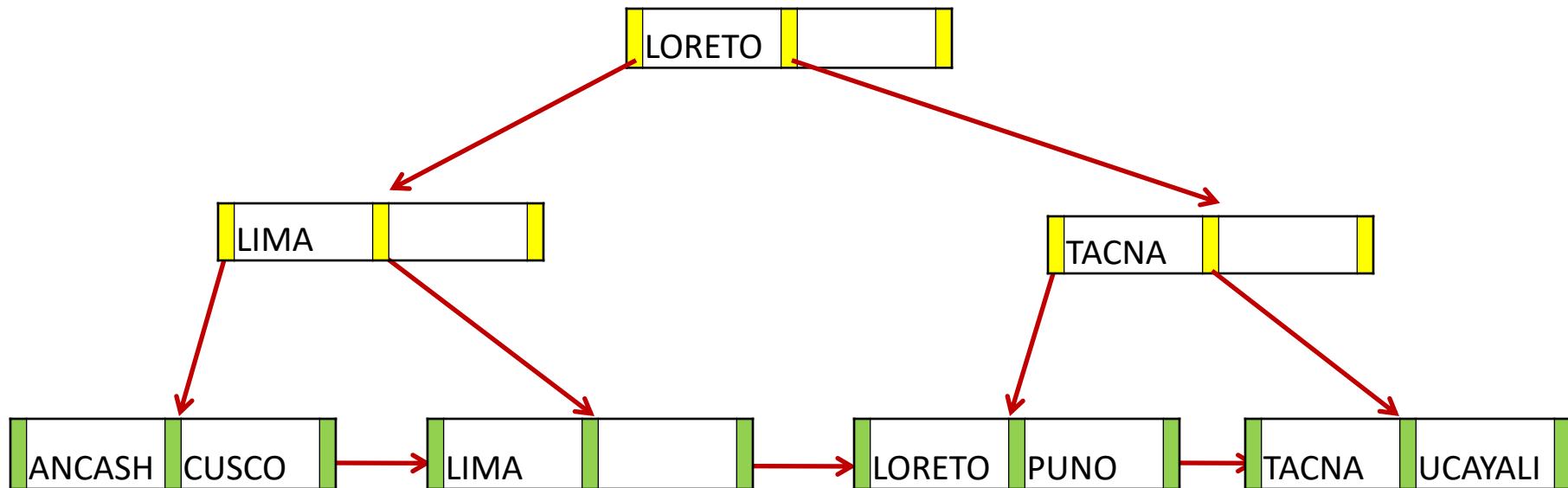
Sistemas de Base de Datos

Indexación y asociación

Archivos de Índices de árbol B⁺

Los árboles-B⁺ son una variante árboles-B, donde se tiene dos tipos de nodos:

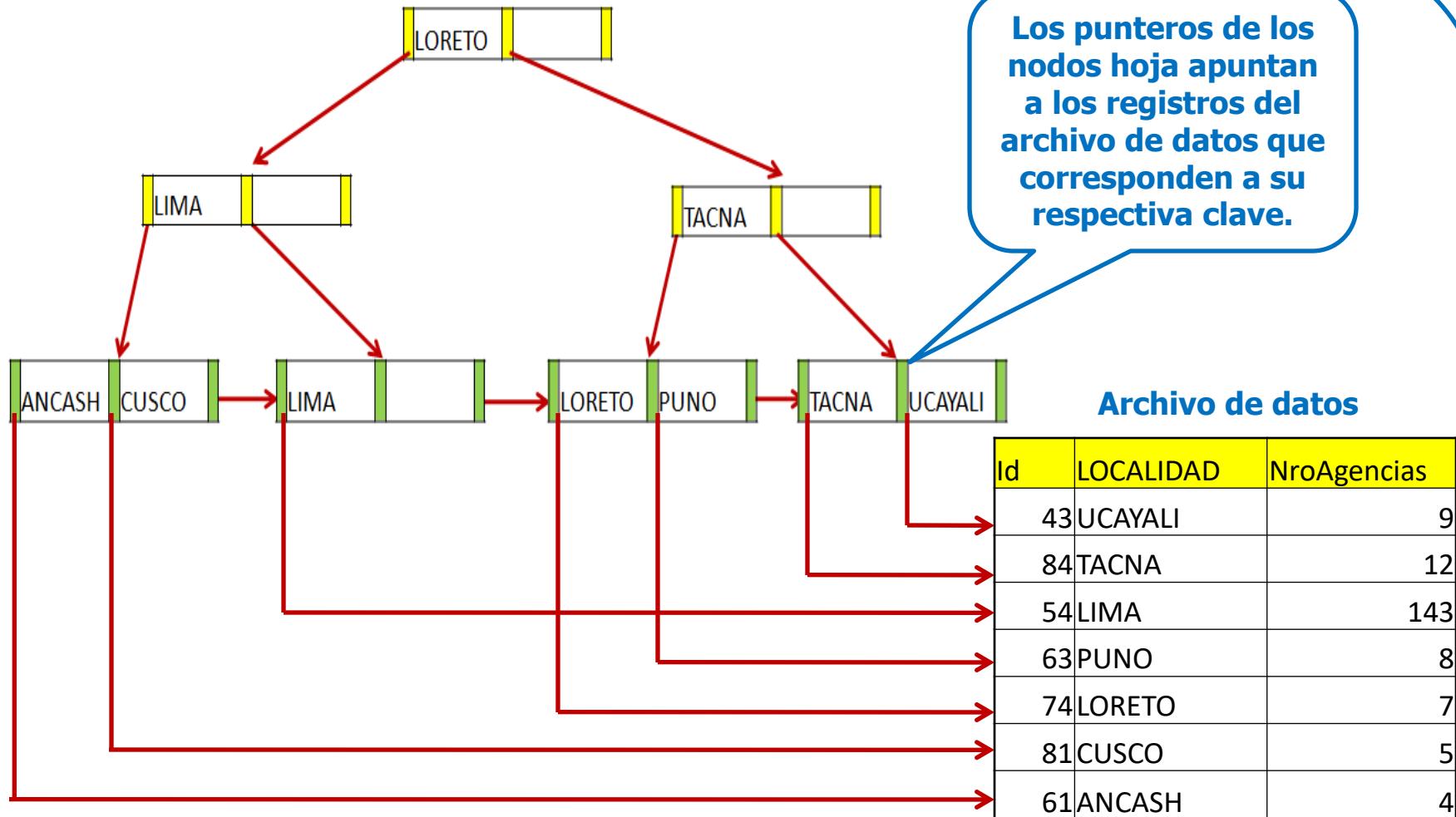
- **Nodos internos.**- Que almacenan las claves y los punteros a otros nodos.
- **Nodos hoja.**- Que almacenan las claves y los punteros a sus respectivos registros de datos



Sistemas de Base de Datos

Indexación y asociación

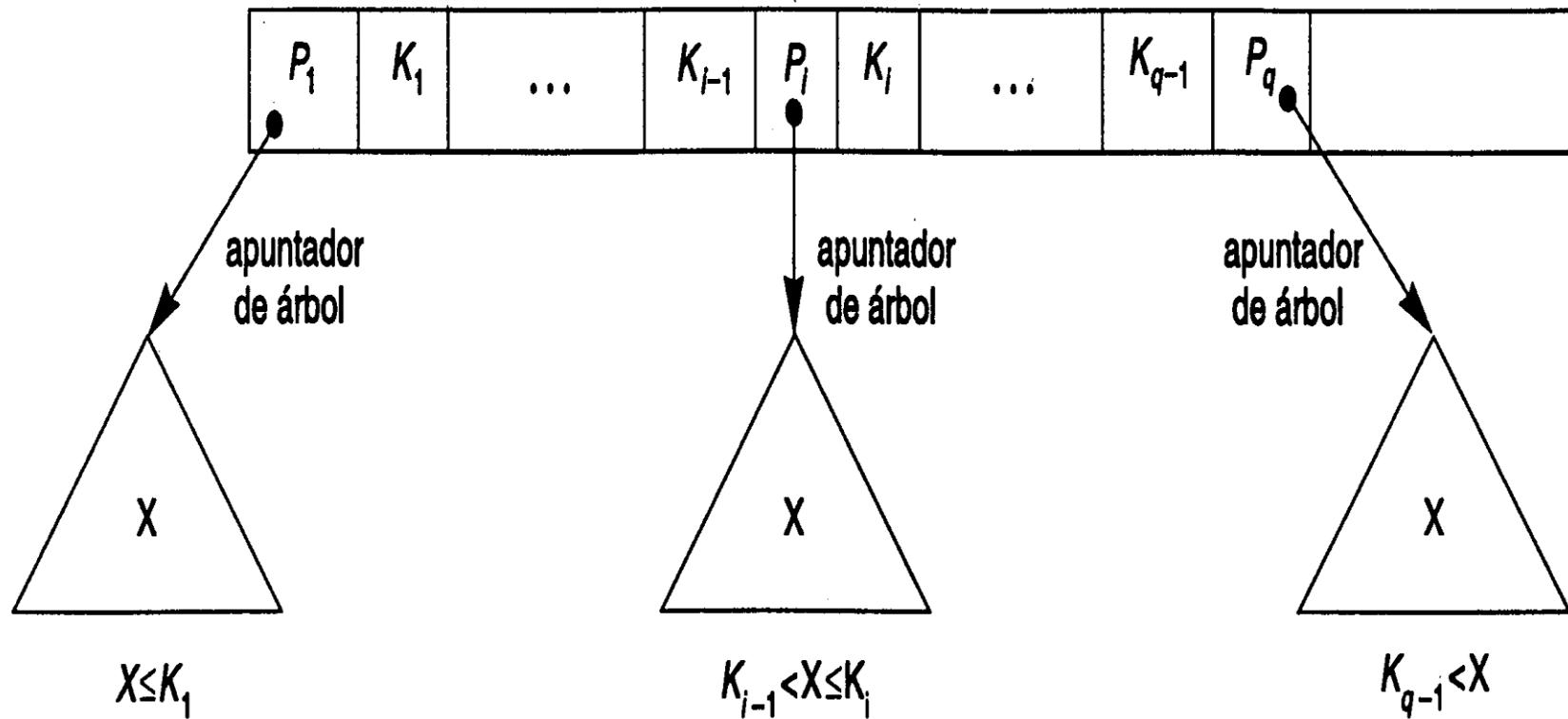
Archivos de Índices de árbol B+



Indexación y asociación

Archivos de Índices de árbol B⁺

Estructura de un nodo interno



Indexación y asociación

Archivos de Índices de árbol B⁺

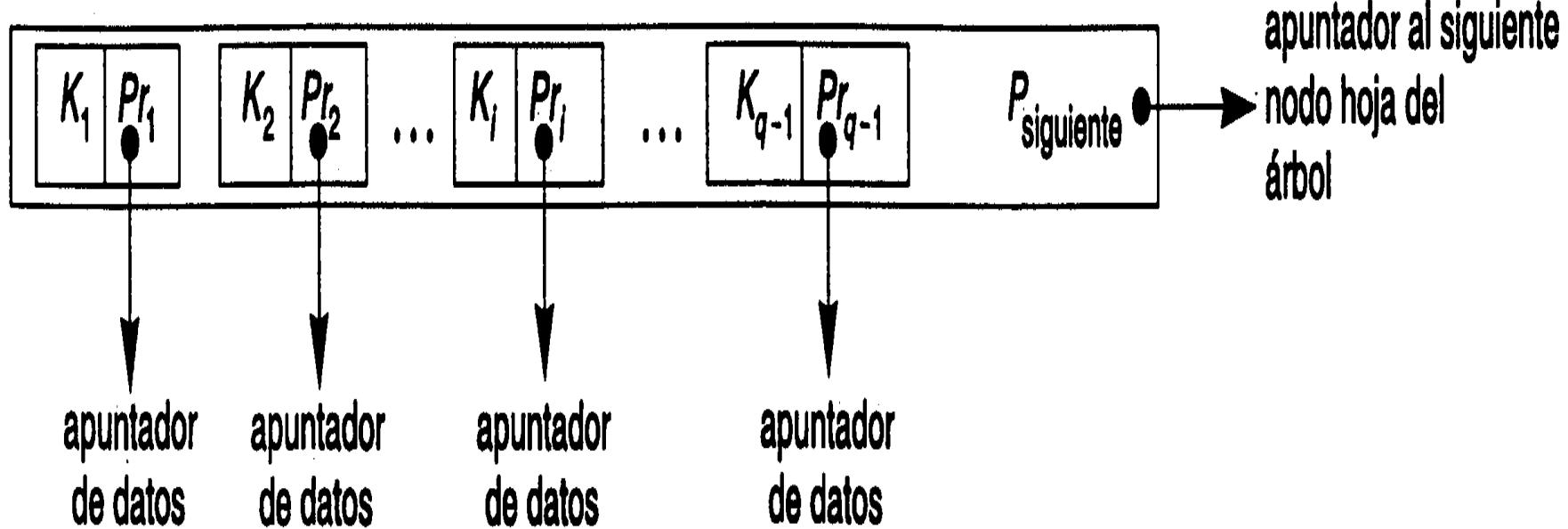
La estructura de los nodos internos de los árboles B+ se define como:

- Todo nodo interno tiene la forma:
$$\langle P_1, K_1, P_2, K_2, \dots, P_{q-1}, K_{q-1}, P_q \rangle$$
- Dentro de cada nodo interno, $K_1 < K_2 < \dots < K_{q-1}$
- Para todos los valores X del campo de búsqueda en el subárbol al que apunta P_i , tenemos $K_{i-1} < X \leq K_i$, para $1 < i < q$, $X \leq K_i$, para $i = 1$, y $K_{i-1} < X$, para $i=q$.
- Cada nodo interno tiene cuando más p apuntadores de árbol.
- Cada nodo interno, con excepción de la raíz, tiene por lo menos $[(p/2)]$ apuntadores de árbol. El nodo raíz tiene por lo menos dos apuntadores de árbol si es un nodo interno.
- Un nodo interno con q apuntadores, donde $q \leq p$, tiene $q-1$ valores del campo de búsqueda.

Indexación y asociación

Archivos de Índices de árbol B⁺

Estructura de un nodo hoja:



Indexación y asociación

Archivos de Índices de árbol B⁺

La estructura de los nodos internos de los árboles B+ se define como:

- Todo nodo hoja tiene la forma:

$\langle \langle K_1, Pr_1 \rangle, \langle K_2, Pr_2 \rangle, \dots, \langle K_{q-1}, Pr_{q-1} \rangle, Pr_{siguiente} \rangle$

donde $q \leq p$, cada Pr_i es un apuntador de datos y $Pr_{siguiente}$ apunta al siguiente nodo hoja del árbol B+.

- Dentro de cada nodo interno, $K_1 < K_2 < \dots < K_{q-1}$, donde $q \leq p$
- Cada Pr_i es un apuntador de datos que apunta al registro cuyo valor de clave de búsqueda es K_i , o a un bloque de archivo que contiene dicho registro (o a un bloque de apuntadores que apuntan a registros cuyo valor del campo de búsqueda es K_i , si el campo de búsqueda no es clave).
- Cada nodo hoja tiene por lo menos $[(p/2)]$ valores.
- Todos los nodos hoja están en el mismo nivel.

Indexación y asociación

Archivos de Índices de árbol B⁺

Inserción en arboles B⁺

- Es similar al proceso de inserción en árboles-B.
- La dificultad se presenta cuando desea insertarse una clave en un nodo que se encuentra lleno ($m = 2d$). Se presenta dos casos:
 - Si el nodo es una hoja.- La hoja afectada se divide en 2, distribuyéndose las $m + 1$ claves de la siguiente forma:
 - Las d primeras claves en la hoja de la izquierda.
 - Las $d + 1$ restantes claves en la hoja de la derecha.
 - Una copia de la clave del medio sube a la pagina antecesora (duplicidad de clave).
 - Si el nodo es uno interno.- La hoja afectada se divide también en 2, distribuyéndose las $m + 1$ claves de la siguiente forma:
 - Las d primeras claves en el nodo de la izquierda.
 - Las d últimas claves en el nodo de la derecha.
 - La clave del medio sube a la pagina antecesora.
- El proceso de propagación puede llegar hasta la raíz, en cuyo caso la altura del árbol puede incrementarse en una unidad

Indexación y asociación

Archivos de Índices de árbol B⁺

Operación de Inserción en un árbol B⁺

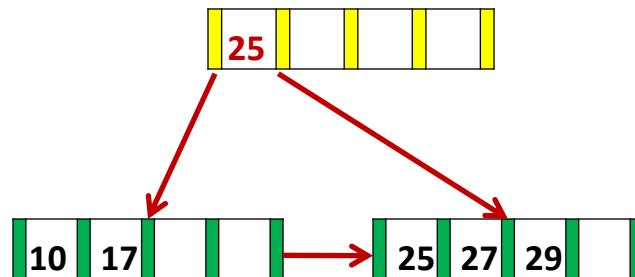
Ejemplo: Insertar las siguientes claves en un árbol-B+ de orden 2 que se encuentra vacío:

claves: 10-27-29-17-25-21-15-31-13-51-20-24-48-19-60-35-66

a) Agregando las claves: 10, 27, 29 y 17



b) Agregando la clave: 25

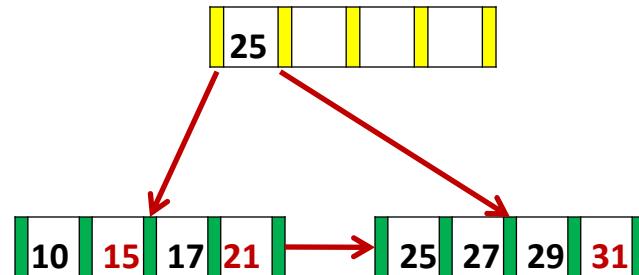


Indexación y asociación

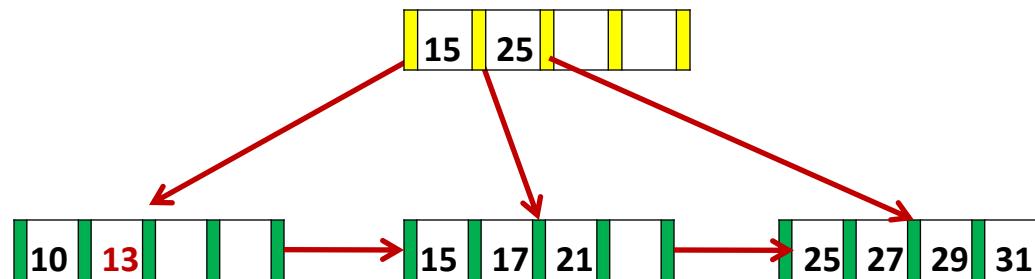
Archivos de Índices de árbol B⁺

Operación de Inserción en un árbol B⁺

c) Agregando las claves: 21, 15, 31



d) Agregando la clave: 13

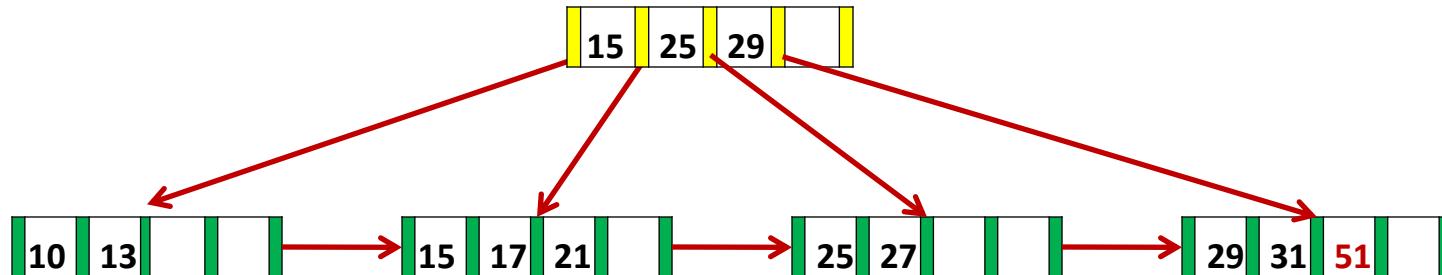


Indexación y asociación

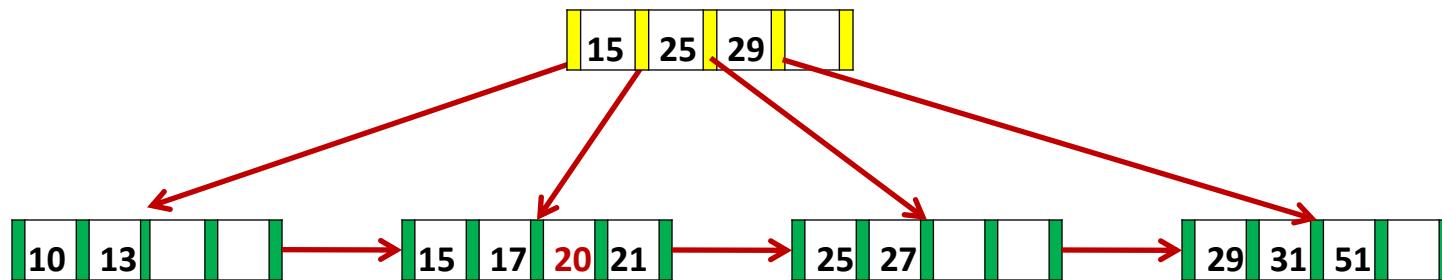
Archivos de Índices de árbol B⁺

Operación de Inserción en un árbol B⁺

e) Agregando la clave: 51



f) Agregando la clave: 20

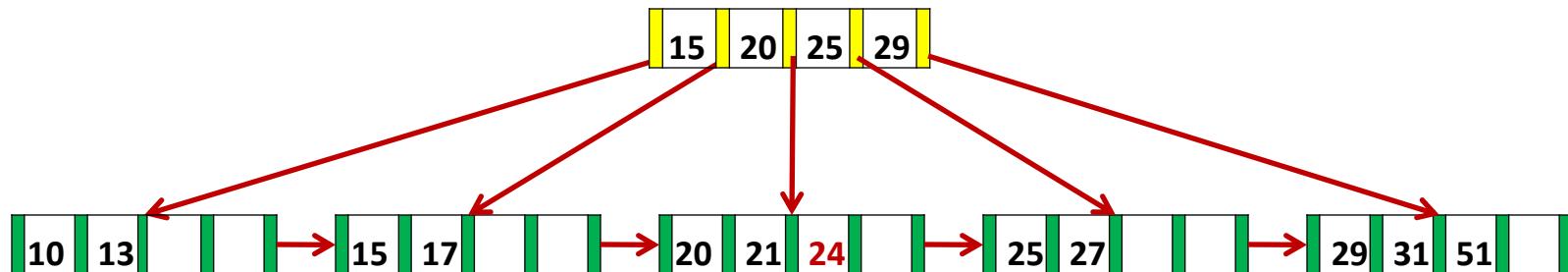


Indexación y asociación

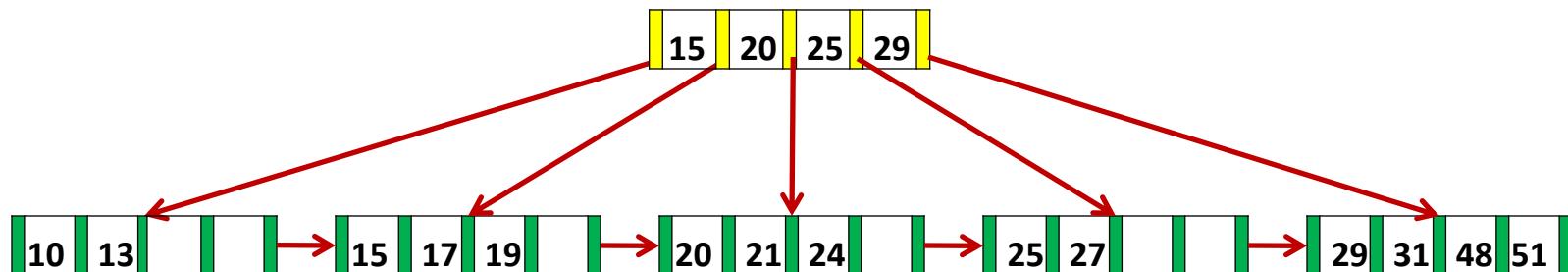
Archivos de Índices de árbol B⁺

Operación de Inserción en un árbol B⁺

g) Agregando la clave: 24



h) Agregando las claves: 48 y 19

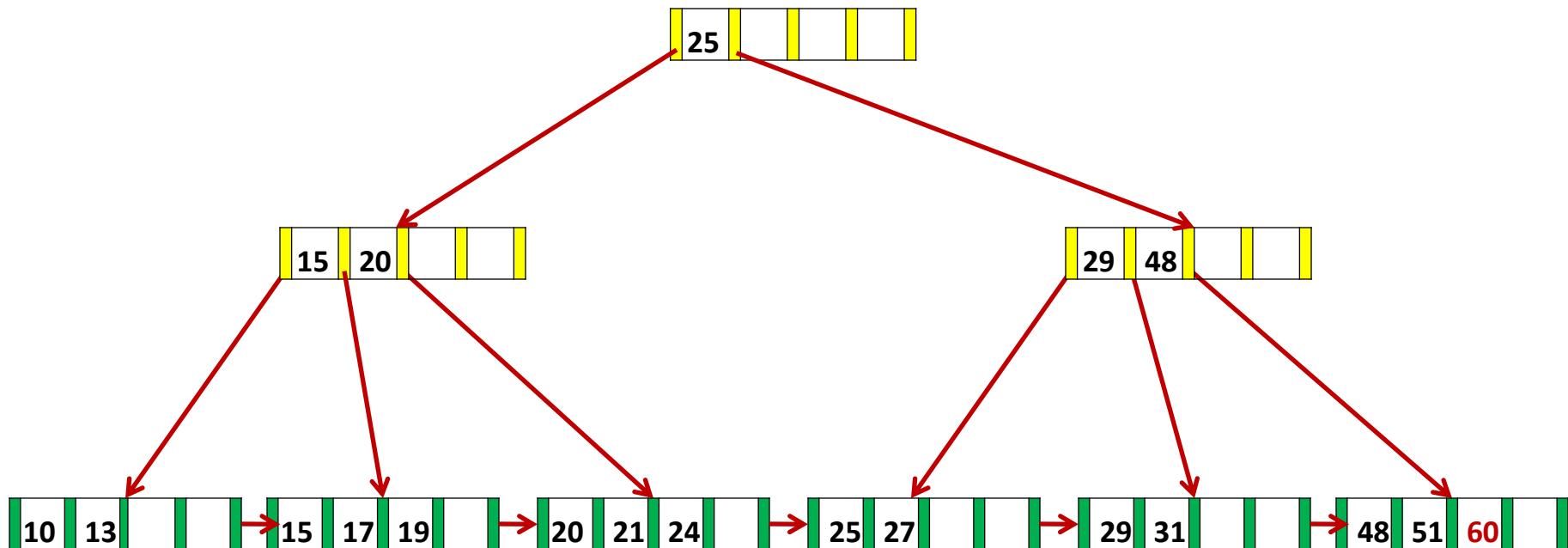


Indexación y asociación

Archivos de Índices de árbol B⁺

Operación de Inserción en un árbol B⁺

i) Agregando la clave: 60

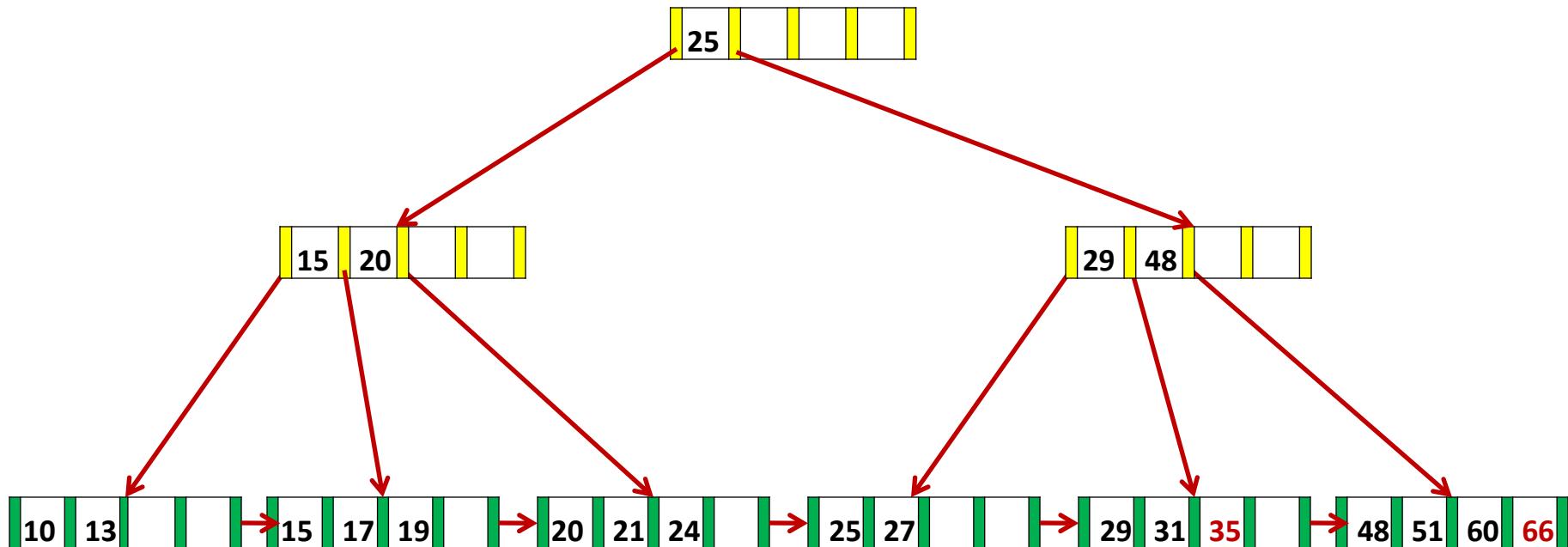


Indexación y asociación

Archivos de Índices de árbol B⁺

Operación de Inserción en un árbol B⁺

j) Agregando las claves: 35 y 66



Asociación

Asociación estática

Un inconveniente de la organización de archivos secuenciales es que hay que acceder a una estructura de índices para localizar los datos o utilizar una búsqueda binaria y, como resultado, más operaciones de E/S. La organización de archivos basada en la técnica de asociación (*hashing*) permite evitar el acceso a la estructura de índice. La asociación también proporciona una forma de construir índices.

En una organización de archivos por asociación se obtiene la dirección del bloque de disco que contiene el registro deseado mediante el cálculo directo de una función sobre el valor de la clave búsqueda del registro.

$$\text{Dirección_bloque_disco} = f(\text{Clave_Búsqueda})$$

Asociación

Asociación estática

En esta descripción de asociación, utilizaremos el término **cajón** (*bucket*) para indicar una unidad de almacenamiento que puede guardar uno o más registros. Un cajón es normalmente un bloque de disco, aunque también se podría elegir más pequeño o más grande que un bloque de disco.

- Formalmente, sea K el conjunto de todos los valores de clave de búsqueda y sea B el conjunto de todas las direcciones de cajón. Una función de asociación h es una función de K a B . Sea h una función asociación.

$$B = h(K)$$

Asociación

Asociación estática

Para insertar un registro con clave de búsqueda K_i , calcularemos $h(K_i)$, lo que proporciona la dirección del cajón para ese registro.

Para realizar una búsqueda con el valor K_i de la clave de búsqueda, simplemente se calcula $h(K_i)$ y luego se busca el cajón con esa dirección.

Supongamos que dos claves de búsqueda, K_5 y K_7 , tienen el mismo valor de asociación; esto es, $h(K_5) = h(K_7)$. Si se realiza una búsqueda en K_5 , el cajón $h(K_5)$ contendrá registros con valores de la clave de búsqueda K_5 y registros con valores de la clave de búsqueda K_7 . Así, hay que comprobar el valor de clave de búsqueda de cada registro en el cajón para verificar que el registro es el que queremos.

Asociación

Asociación estática

Ejemplo de cajones con información de cuentas.

cajón 0

--	--	--

cajón 1

--	--	--

cajón 2

--	--	--

cajón 3

C-217	Barcelona	750
C-305	Ronda	350

cajón 4

C-222	Reus	700
-------	------	-----

cajón 5

C-102	Pamplona	400
C-201	Pamplona	900
C-218	Pamplona	700

cajón 6

--	--	--

cajón 7

C-215	Mianus	700
-------	--------	-----

cajón 8

C-101	Daimiel	500
C-110	Daimiel	600

cajón 9

--	--	--

Asociación

Funciones de asociación (Funciones Hash)

La peor función posible de asociación asigna todos los valores de la clave de búsqueda al mismo cajón.

Una función de asociación ideal distribuye las claves almacenadas uniformemente a través de los cajones para que cada uno de ellos tenga el mismo número de registros.

La función de asociación es deseable que cumpla con lo siguiente:

- **Distribución *uniforme*.** - *Esto es, cada cajón tiene asignado el mismo número de valores de la clave de búsqueda dentro del conjunto de todos los valores posibles de la clave de búsqueda.*
- **Distribución *aleatoria*.** - *Esto es, en el caso promedio, cada cajón tendrá casi el mismo número de valores asignados a él, sin tener en cuenta la distribución actual de los valores de la clave de búsqueda.*

Asociación

Funciones de asociación (Funciones Hash)

Ejemplos:

- **función de asociación que asigna a los nombres que empiezan con la letra *i*-ésima del alfabeto el *i*-ésimo Cajón.**

Esta función tiene la virtud de la simplicidad, pero no logra proporcionar una distribución uniforme, ya que se espera que haya más nombres que comiencen con letras como la C que con X.

- **función de asociación para claves numéricas (por ejemplo saldo que varía en el rango de 1 a 100000), se puede utilizar una función que divida el número en rangos: 1 a 10000, 10001 a 20000, etc.**

Otra vez aquí, se tiene una función sencilla de implementar, pero tampoco proporciona una distribución uniforme, porque los saldos no son uniformes.

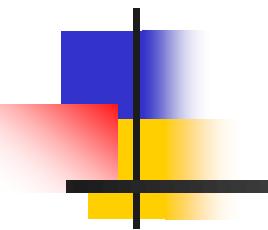
- **Una función más elaborada: Si la clave es un texto *S* de *n* caracteres. Se puede utilizar la siguiente función:**

$$S[0]*31^{(n-1)} + S[1]*31^{(n-2)} + \dots + S[n-2]*31^1 + S[n-1]$$

Esta función se puede implementar en cualquier lenguaje de programación, tratándose cada carácter como un entero.

Asociación

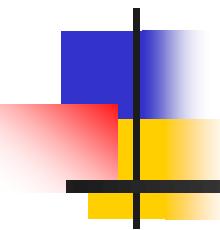
Funciones de asociación (Funciones Hash)



Las funciones de asociación requieren un diseño cuidadoso. Una mala función de asociación podría provocar que una búsqueda tome un tiempo proporcional al número de claves de búsqueda en el archivo. Una función bien diseñada en un caso medio de búsqueda toma un tiempo constante (pequeño), independiente del número de claves búsqueda en el archivo.

Asociación

Determinación del número de cajones



La asociación estática supone que se conoce el número total de registros o se puede obtener el número aproximado de ellos al momento de definir la función de asociación. Bajo esta suposición se puede determinar el número de cajones como sigue:

$$n_B > n_r / f_r$$

Donde: n_B es el número de cajones

n_r es el número total de registros

f_r es el número de registros que caben en cada cajón.

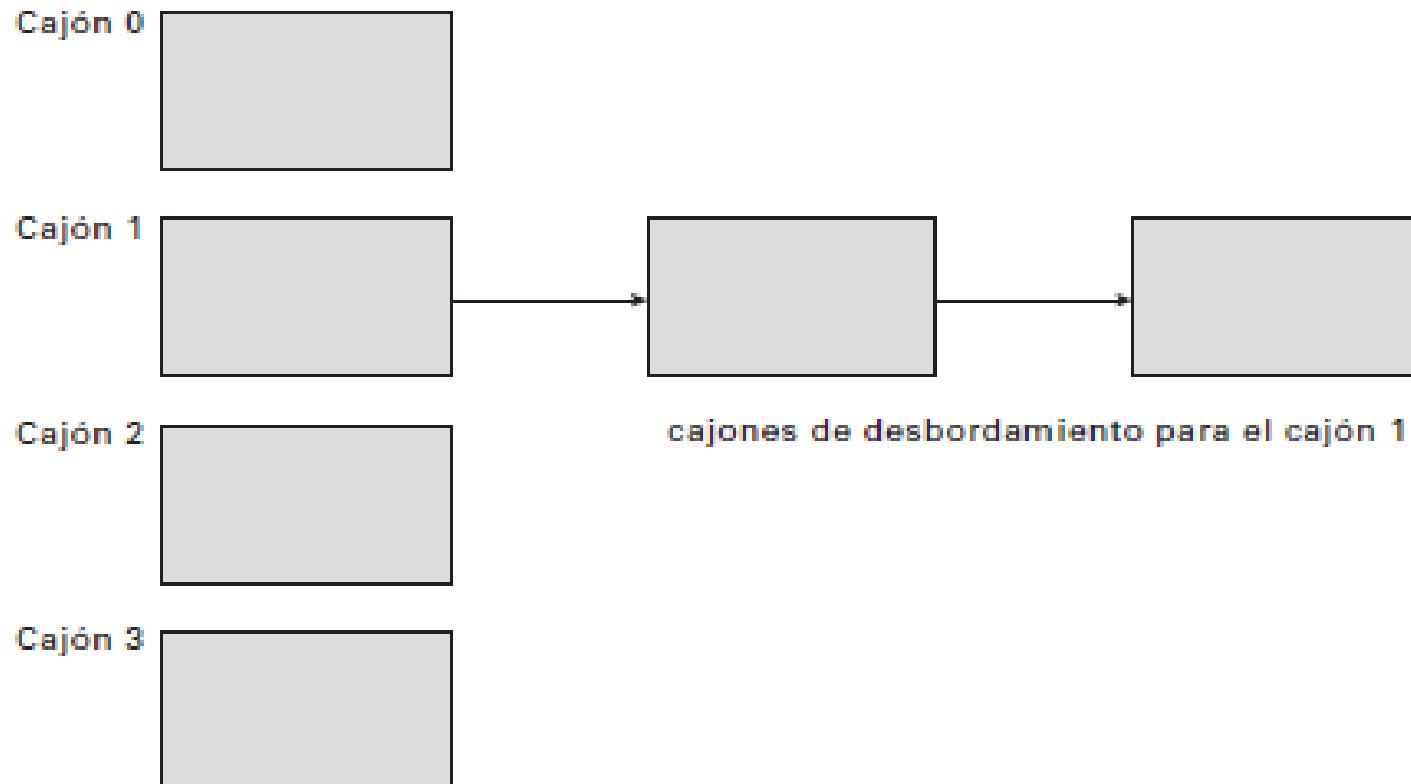
Sistemas de Base de Datos

Asociación

Gestión de desbordamiento de cajones

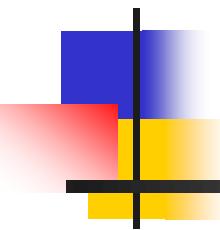
Cuando se insertan registros en un cajón en base a la función de asociación elegida, puede darse el caso de que el cajón ya esté lleno y no haya espacio para insertar un registro más. A este hecho se conoce como desbordamiento de cajones.

Existen diferentes técnicas para gestionar el desbordamiento. La más simple es la denominada **cadena de desbordamiento**, que consiste en una lista enlazada de cajones , tal la figura:



Asociación

Gestión de desbordamiento de cajones



Para que la probabilidad de desbordamiento de cajones se reduzca, se recomienda ajustar el número de cajones mediante un factor de corrección:

$$n_B > (n_r / f_r) * (1+d)$$

Donde: d es el factor de corrección

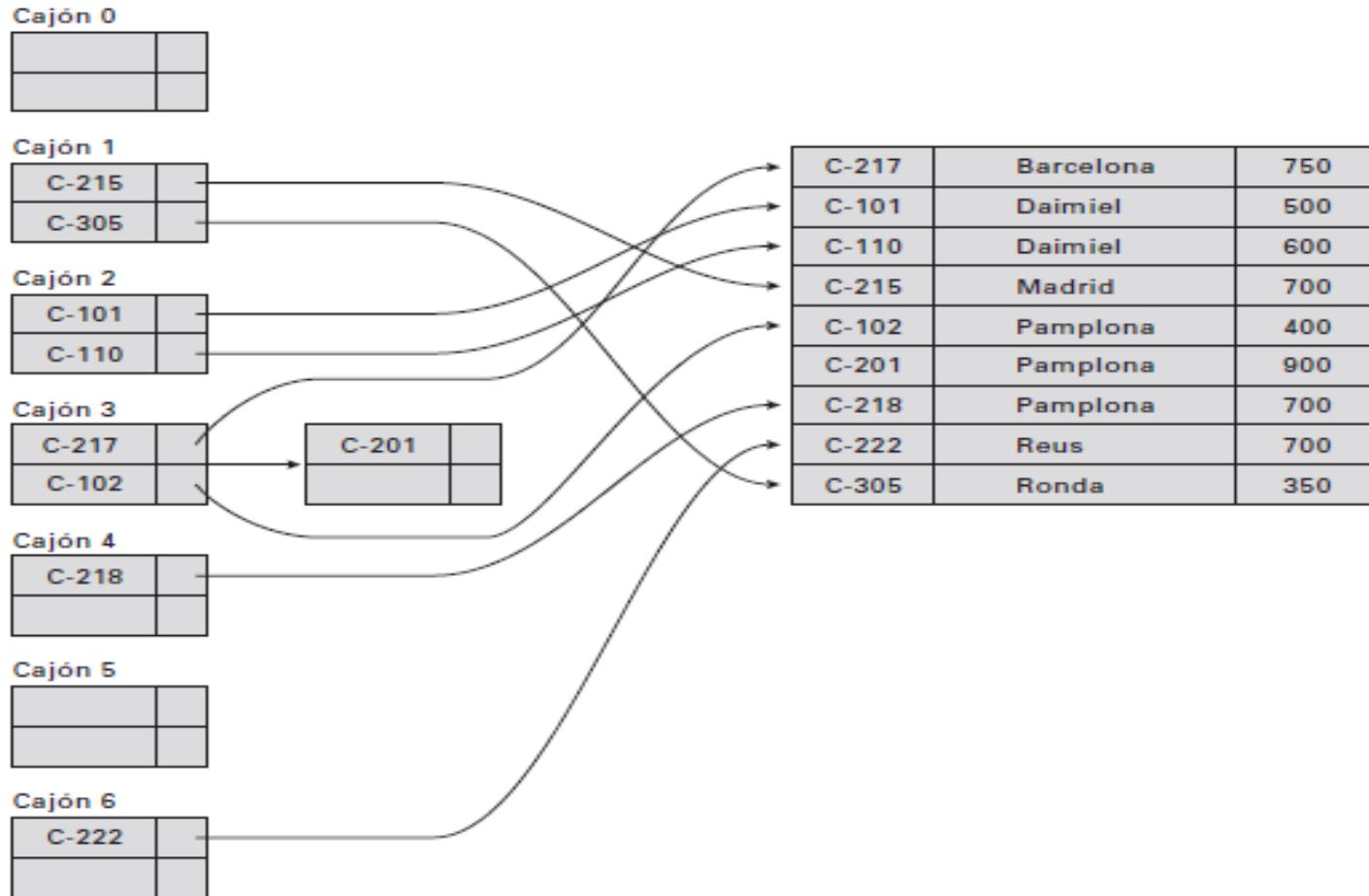
Se recomienda aumentar el número de cajones en un 20% más del requerido ($d = 0.2$). Se pierde algo de espacio, pero la ventaja es que la probabilidad de desbordamiento se reduce.

Sistemas de Base de Datos

Asociación

Índices asociativos

La asociación se puede utilizar no solamente para la organización de archivos, sino también para la creación de estructuras de índice. En los cajones se almacena las claves y sus respectivas direcciones en el archivo de datos.



Asociación

Asociación dinámica

La necesidad de fijar el número de direcciones de los cajones en la asociación estática, presenta un problema serio. La mayor parte de las bases de datos aumenta de tamaño con el tiempo.

Algunas técnicas de asociación dinámica permiten modificar la función de asociación dinámicamente para acomodarse al aumento o disminución de la base de datos.

Se revisará sólo una forma de asociación dinámica, llamada **asociación extensible.**

La **asociación extensible hace frente a los cambios del tamaño de la base de datos dividiendo y fusionando los cajones a medida que la base de datos aumenta o disminuye.**

Asociación

Asociación extensible

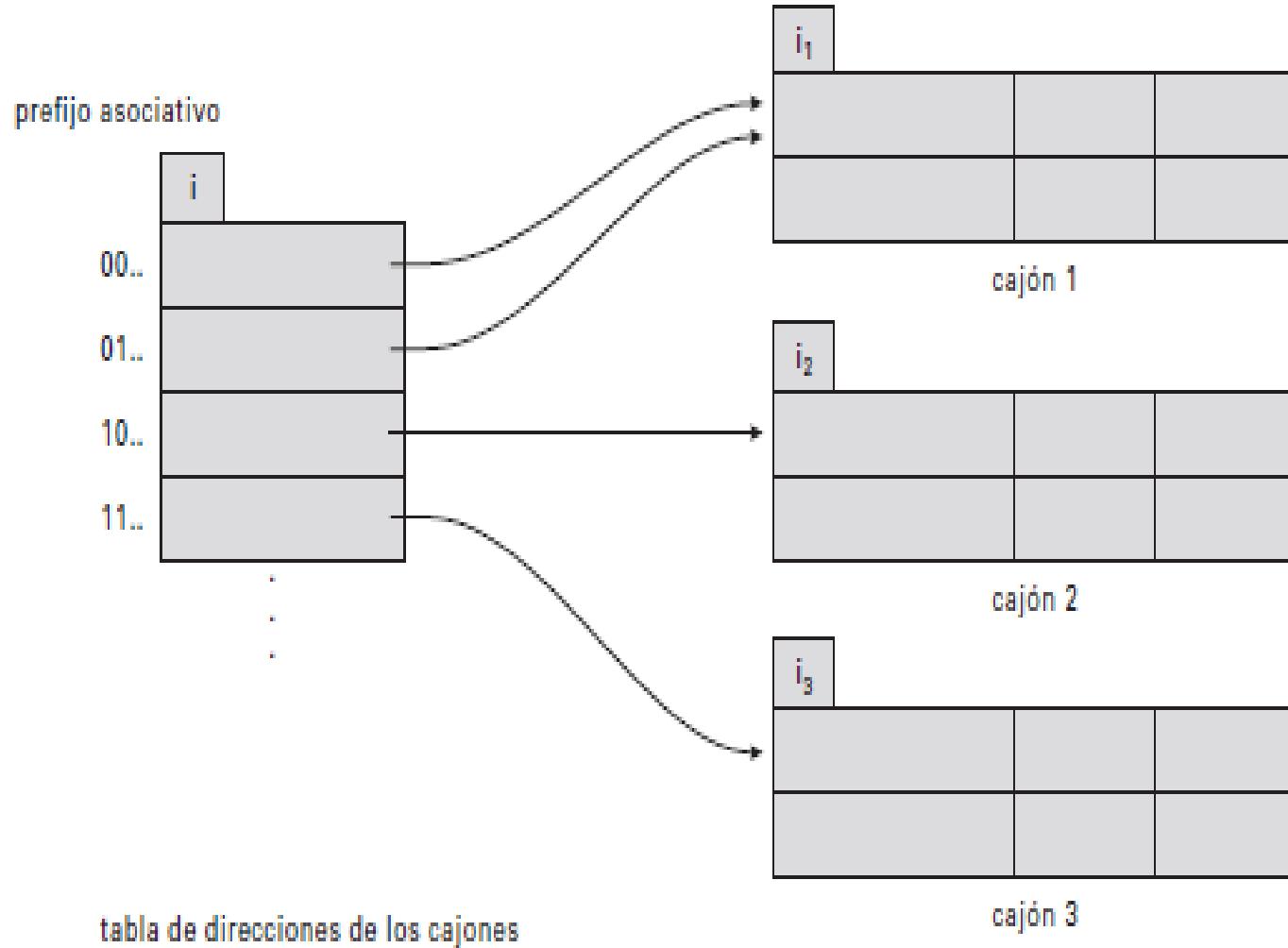
- Se elige una función de asociación h con las propiedades deseadas de uniformidad y aleatoriedad.
- Se genera valores dentro de un rango relativamente amplio, llamado, enteros binarios de b bits. Un valor normal de b es 32.
- Se crean tantos cajones bajo demanda, esto es, tantos como registros haya insertados en el archivo. Inicialmente no se utiliza el total de b bits del valor de la función de asociación.
- En cualquier caso, empleamos i bits, donde $0 < i < b$. Estos i bits son utilizados como desplazamiento en una tabla adicional con las direcciones de los cajones. El valor de i aumenta o disminuye con el tamaño de la base de datos.
- Ver la figura de la siguiente diapositiva.

Sistemas de Base de Datos

Asociación

Asociación extensible

Estructura de datos de la Asociación extensible



Asociación

Asociación extensible

Para ilustrar la técnica de la asociación extensible, vamos a considerar los siguientes datos y a partir de éstos construir la tabla de direcciones y los cajones con los registros respectivos.

Barcelona	C-217	750
Daimiel	C-101	500
Daimiel	C-110	600
Madrid	C-215	700
Pamplona	C-102	400
Pamplona	C-201	900
Pamplona	C-218	700
Reus	C-222	700
Ronda	C-305	350

Asociación

Archivos de Índices Asociativos

Supongamos que deseamos registrar los datos considerando como clave el nombre de la sucursal. Supongamos también que se implementó una función hash que generó los siguientes valores para las claves.

nombre-sucursal	$h(\text{nombre-sucursal})$
Barcelona	0010 1101 1111 1011 0010 1100 0011 0000
Daimiel	1010 0011 1010 0000 1100 0110 1001 1111
Madrid	1100 0111 1110 1101 1011 1111 0011 1010
Pamplona	1111 0001 0010 0100 1001 0011 0110 1101
Reus	0011 0101 1010 0110 1100 1001 1110 1011
Ronda	1101 1000 0011 1111 1001 1100 0000 0001

Asociación

Archivos de Índices Asociativos

Originalmente fichero vacío

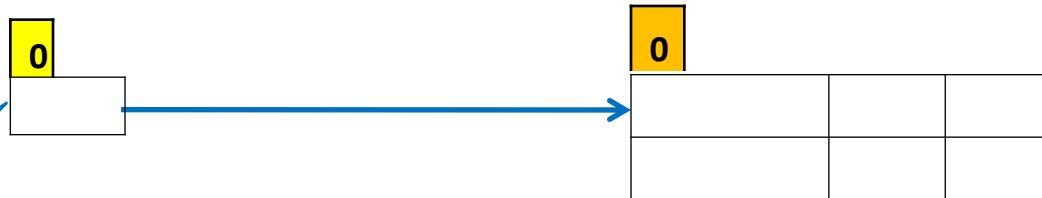
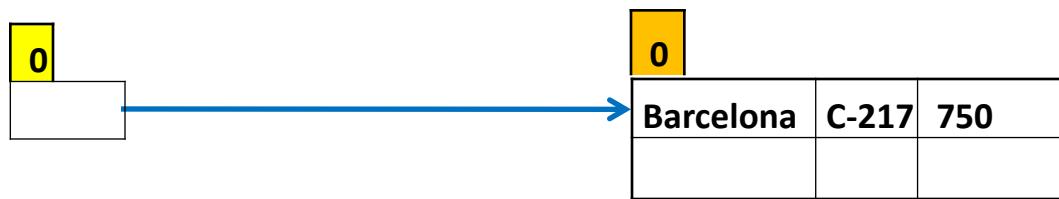


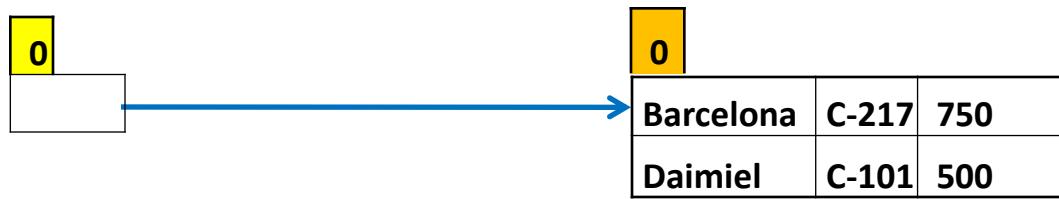
Tabla de
direcciones de
los cajones

Primer cajón

1. Inserción de: Barcelona, C-217, 750 (0010 1101 1111 1011 0010 1100 0011 0000)



2. Inserción de: Daimiel, C-101, 500 (1010 0011 1010 0000 1100 0110 1001 1111)



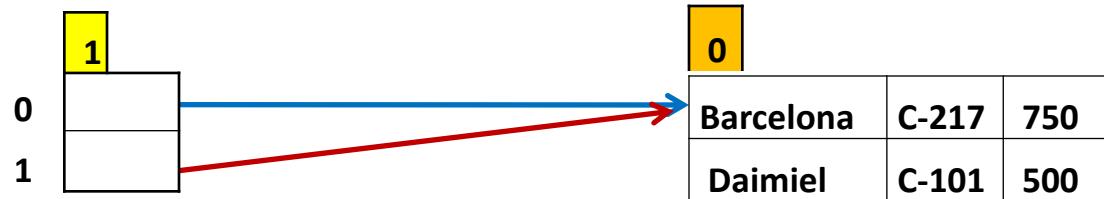
Asociación (Asociación extensible)

3. Inserción de: Daimiel, C-110, 600 (1010 0011 1010 0000 1100 0110 1001 1111)

No es posible insertar este registro en el único cajón, entonces, es necesario incrementar i (el número de dígitos que se utilizará para la función hash). Se produce los siguientes cambios:

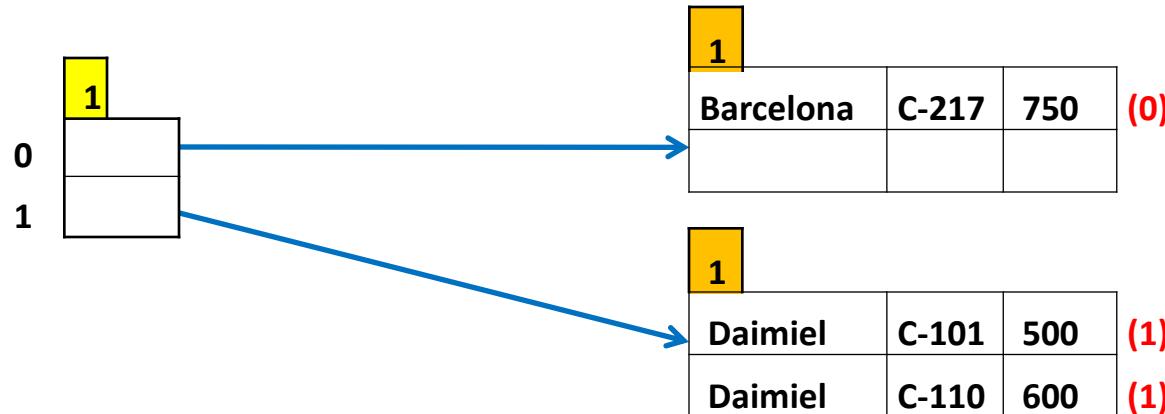
- Se duplica las entradas en la tabla de direcciones de los cajones
- Cada entrada se sustituye por dos entradas, ambas con el mismo puntero que la entrada original

Paso a



Luego, se divide el cajón lleno y se redistribuye los registros actuales, entre el cajón actual y el cajón nuevo. Para determinar a qué cajón va cada clave, se vuelve a calcular la función hash en base a los i primeros bits.

Paso b



Sistemas de Base de Datos

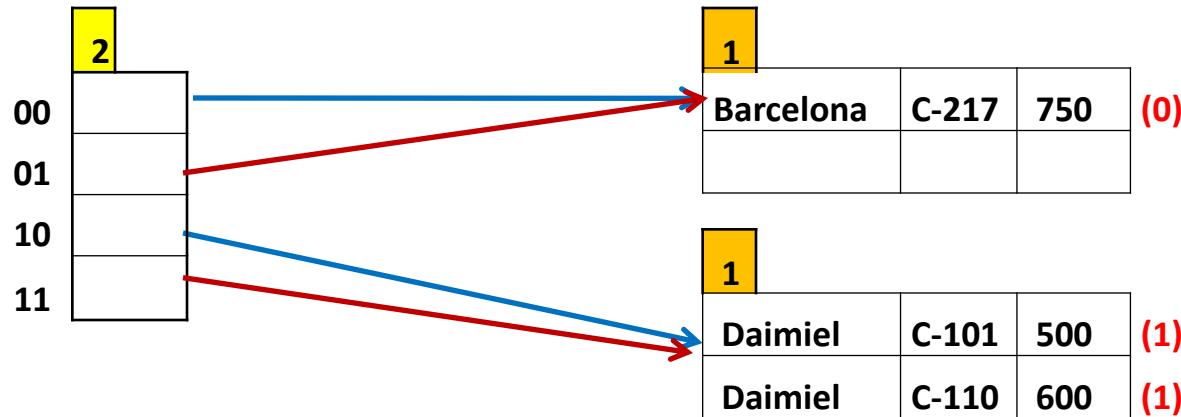
Asociación (Asociación extensible)

4. Inserción de: Madrid, C-215, 700 (1100 0111 1110 1101 1011 1111 0011 1010)

Otra vez no es posible insertar este registro, entonces, es necesario incrementar i_a a 2:

- Se duplica las entradas en la tabla de direcciones de los cajones
- Cada entrada se sustituye por dos entradas, ambas con el mismo puntero que la entrada original

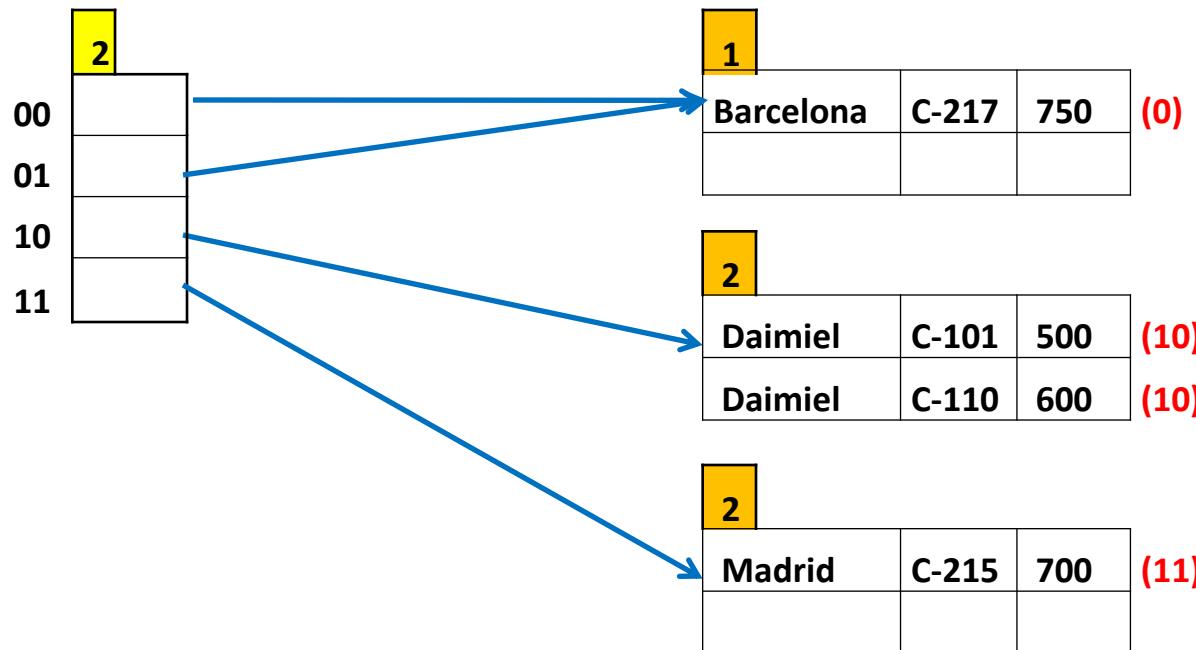
Paso a



Asociación (Asociación extensible)

Luego, se divide el cajón lleno y se redistribuye los registros actuales, entre el cajón actual y el cajón nuevo. Todas estas entradas tendrán un prefijo de asociación común, pero la longitud de ese prefijo puede ser menor que i . Los registros con el mismo prefijo van al mismo cajón

Paso b



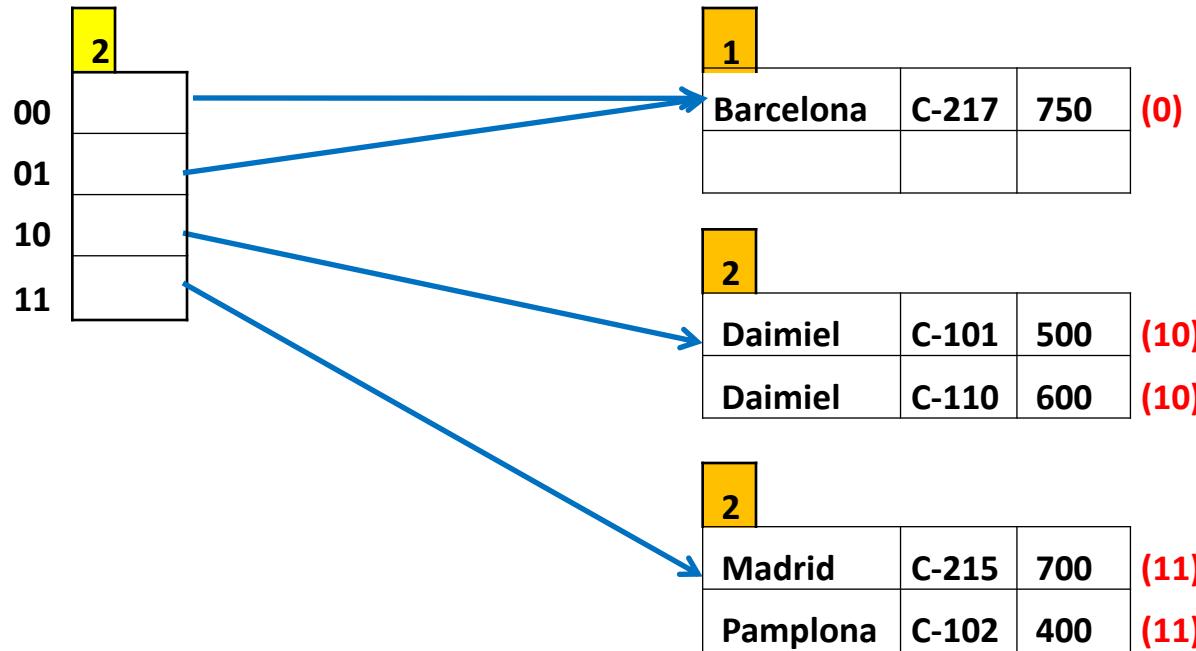
Sistemas de Base de Datos

Asociación (Asociación extensible)

5. Inserción de: Pamplona, C-102, 400 (1111 0001 0010 0100 1001 0011 0110 1101)

Considerando los dos primeros bits de la función hash, Pamplona (11) corresponde al último cajón; como hay espacio, se agrega simplemente.

Paso a



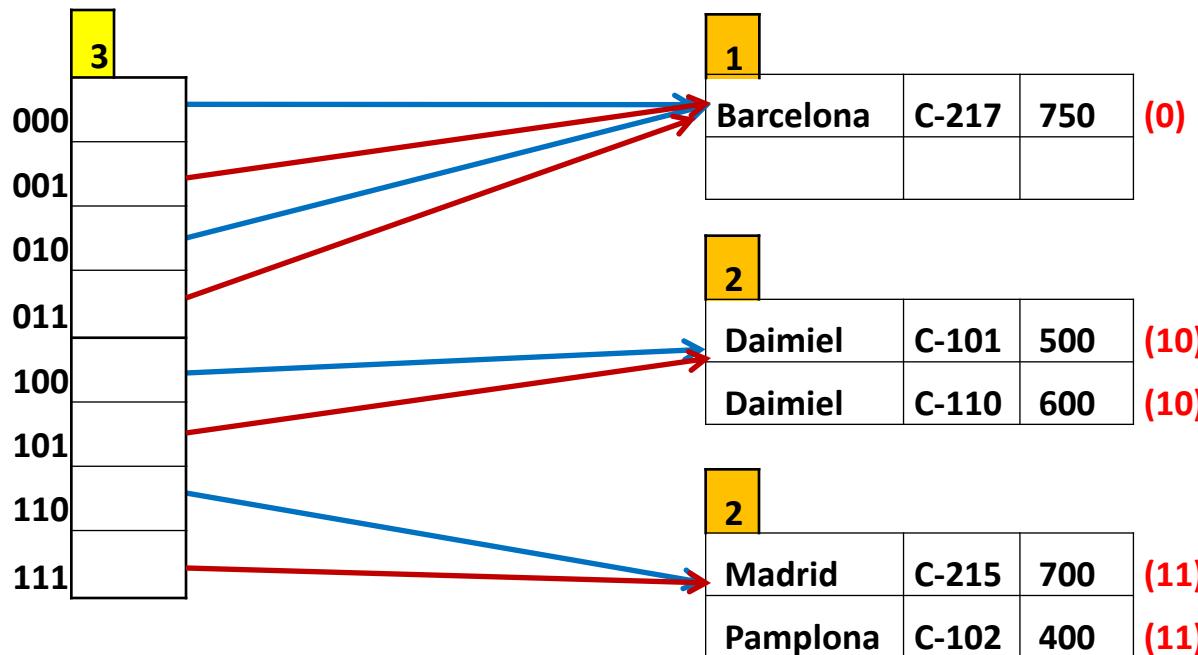
Sistemas de Base de Datos

Asociación (Asociación extensible)

6. Inserción de: Pamplona, C-201, 900 (1111 0001 0010 0100 1001 0011 0110 1101)

Considerando los tres primeros bits de la función hash, Pamplona (111) corresponde al último cajón, Esta vez no hay espacio, entonces se debe incrementar el valor de i.

Paso a

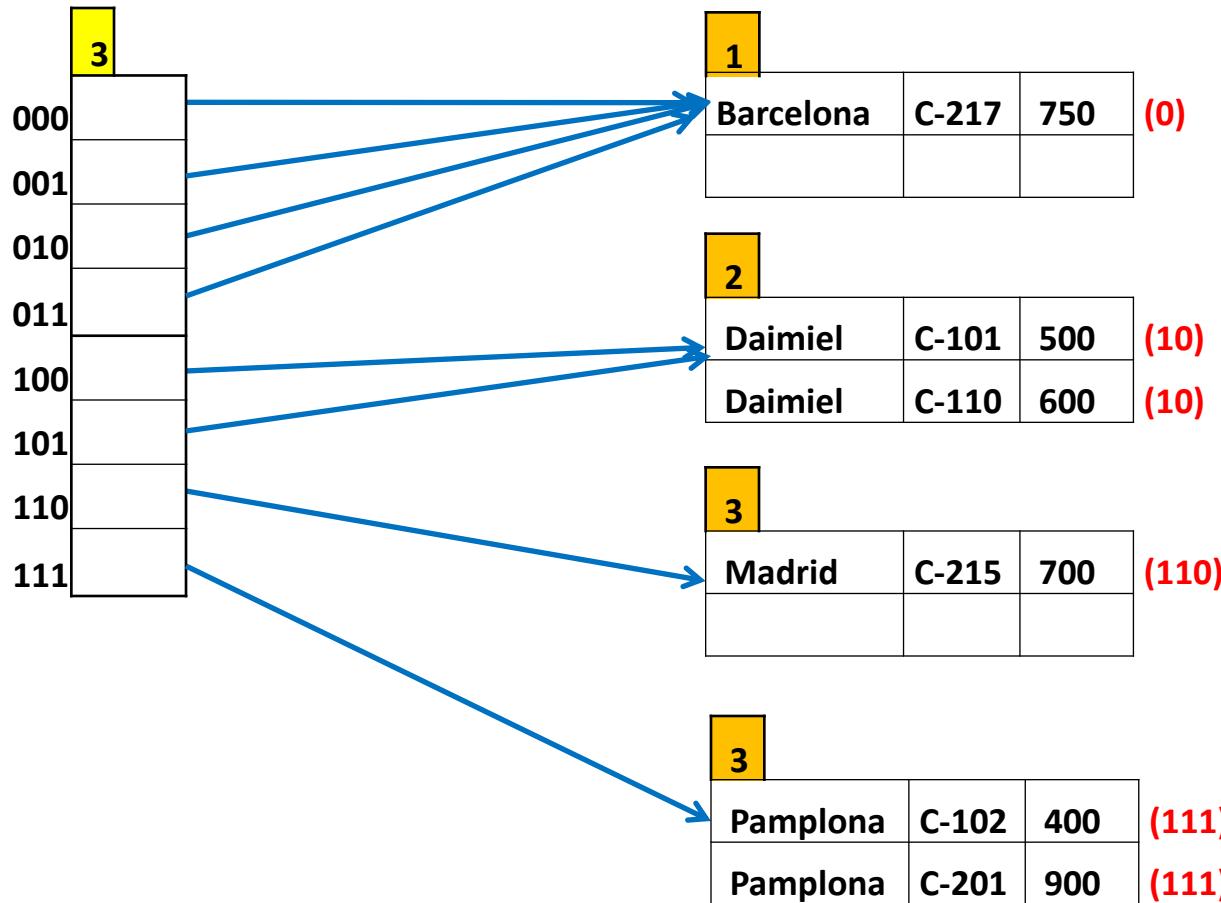


Sistemas de Base de Datos

Asociación (Asociación extensible)

Los registros se redistribuyen en los dos últimos cajones de acuerdo a sus prefijos comunes.

Paso b

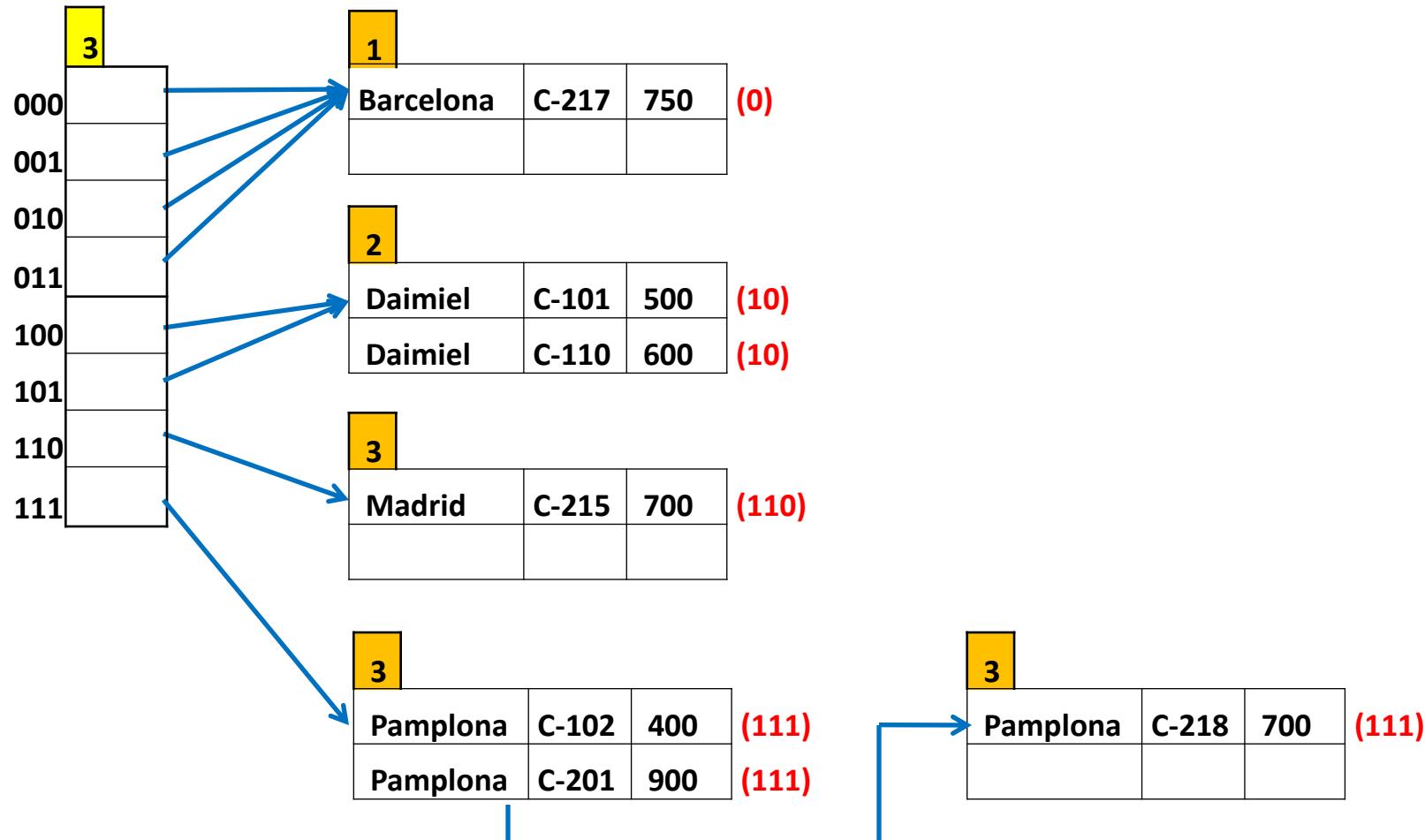


Sistemas de Base de Datos

Asociación (Asociación extensible)

7. Inserción de: Pamplona, C-218, 700 (1111 0001 0010 0100 1001 0011 0110 1101)

La inserción de este registro no es posible en el último cajón, porque éste ya está lleno. Esta situación no se puede resolver incrementando el número de bits, ya que hay otros tres registros con el mismo valor de asociación exactamente. Por tanto se utiliza un cajón de desbordamiento.

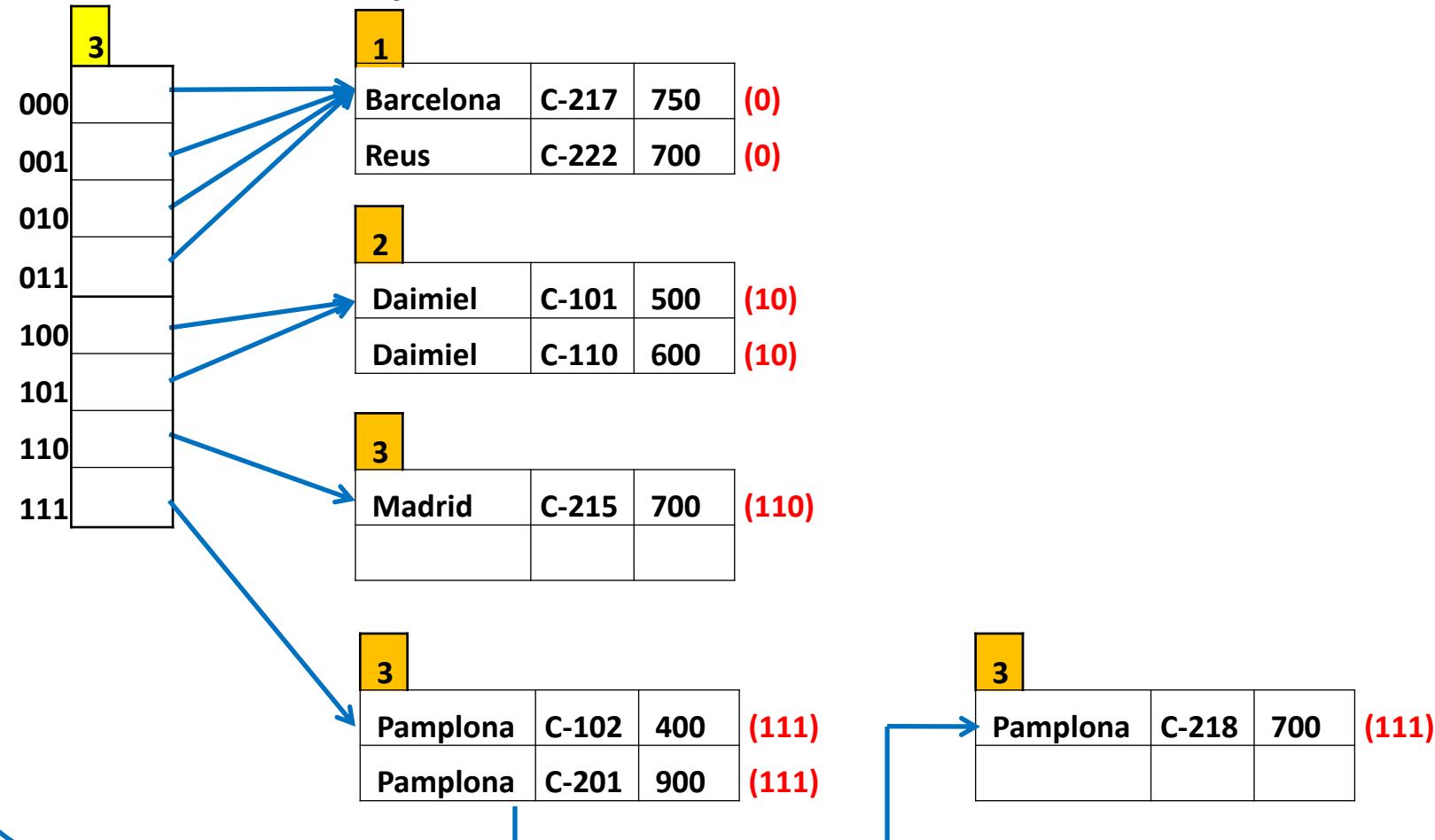


Asociación (Asociación extensible)

8. Inserción de: Reus, C-222, 700

(0011 0101 1010 0110 1100 1001 1110 1011)

Considerando los tres primeros bits de la función hash, Reus (001) corresponde al segundo casillero de la tabla de direcciones y éste apunta al primer cajón. En este cajón se debe considerar un prefijo de longitud 1, luego, Barcelona y Reus tienen el mismo prefijo; y como aún hay espacio, entonces Reus se inserta en este cajón.

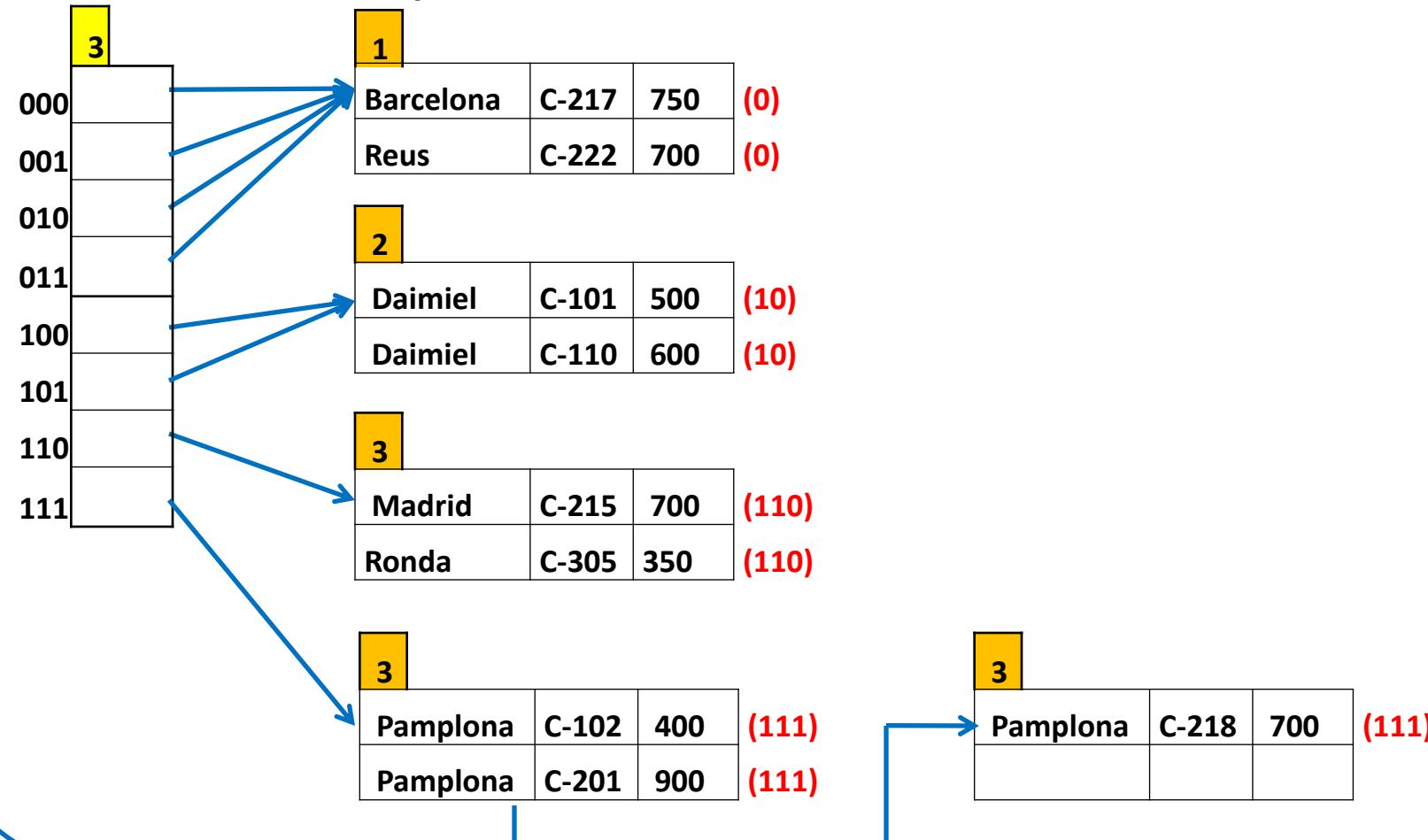


Asociación (Asociación extensible)

9. Inserción de: Ronda, C-305, 350

(1101 1000 0011 1111 1001 1100 0000 0001)

Considerando los tres primeros bits de la función hash, Ronda (110) corresponde al séptimo casillero de la tabla de direcciones y éste apunta al tercer cajón. En este cajón se debe considerar un prefijo de longitud 3, luego, Madrid y Ronda tienen el mismo prefijo; y como aún hay espacio, entonces Ronda se inserta en este cajón.



Asociación

Índices de mapas de bits

Los índices de mapas de bits son un tipo de índices especializado diseñado para la consulta sencilla sobre varias claves, aunque cada índice de mapas de bits se construya para una única clave.

Para que se utilicen los índices de mapas de bits, los registros de la relación deben estar numerados secuencialmente comenzando, por ejemplo, por 0. Dado un número n debe ser fácil recuperar el registro con número n.

Asociación

Índices de mapas de bits

Estructura de los índices de mapas de bits

Un mapa de bits es simplemente un array de bits.

Un índice de mapas de bits sobre el atributo A de la relación r, consiste en un mapa de bits para cada valor que pueda tomar A.

Cada mapa de bits tiene tantos bits como el número de registros de la relación.

El i-ésimo bit del mapa de bits toma el valor de 0 o 1. Toma el valor de 1 si el registro con número i tiene el valor asociado al mapa de bits para el atributo A.

Asociación

Índices de mapas de bits

Por ejemplo, tengamos la siguiente relación:

Número de registro	Nombre	Sexo	Ciudad	Nivel Estudios
0	Pedro	M	Cusco	SUPERIOR
1	Melina	F	Tacna	PRIMARIA
2	Carla	F	Puno	SUPERIOR
3	Ana	F	Cusco	SECUNDARIA
4	Angel	M	Ica	ANALFABETO
5	Luis	F	Lima	PRIMARIA

Deseamos crear mapas de bits para los atributos: Sexo y Nivel de estudios.

Sistemas de Base de Datos

Asociación

Índices de mapas de bits

Mapas de bits para el atributo Sexo

Número de registro	Nombre	Sexo	Ciudad	Nivel Estudios
0	Pedro	M	Cusco	SUPERIOR
1	Melina	F	Tacna	PRIMARIA
2	Carla	F	Puno	SUPERIOR
3	Ana	F	Cusco	SECUNDARIA
4	Angel	M	Ica	ANALFABETO
5	Luis	F	Lima	PRIMARIA

Mapa de bits para Sexo

M 100010

F 011101

En el mapa de bits M (Masculino) sólo el primer y quinto bits son iguales a 1, correspondientes a los registros 0 y 4.

Sistemas de Base de Datos

Asociación

Índices de mapas de bits

Mapas de bits para el atributo Sexo

Número de registro	Nombre	Sexo	Ciudad	Nivel Estudios
0	Pedro	M	Cusco	SUPERIOR
1	Melina	F	Tacna	PRIMARIA
2	Carla	F	Puno	SUPERIOR
3	Ana	F	Cusco	SECUNDARIA
4	Angel	M	Ica	ANALFABETO
5	Luis	F	Lima	PRIMARIA

Mapa de bits para Nivel Estudios

ANALFABETO	000010
PRIMARIA	010001
SECUNDARIA	000100
SUPERIOR	101000

En el mapa de bits SUPERIRO sólo el primer y tercer bits son iguales a 1, correspondientes a los registros 0 y 2.

Asociación

Índices de mapas de bits

Los índices de mapas de bits resultan útiles para las selecciones sobre todo cuando hay selecciones bajo varias claves.

Considérese la siguiente consulta:

$$\sigma_{\text{Sexo} = 'M' \wedge \text{Nivel_Estudios} = 'SUPERIOR'} (r)$$

Para evaluar esta selección se busca el valor **F** en el mapa de bits de **Sexo** y el valor **SUPERIOR** en el mapa de bits **Nivel Estudios** y se realiza la intersección de ambos mapas de bits, generándose un nuevo mapa de bits y sobre este se recuperan los registros que cumplen ambas condiciones.

F

011101

SUPERIOR

101000

Nuevo mapa de bits

001000

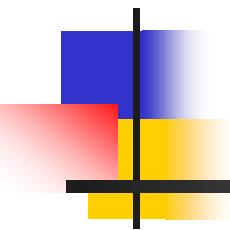
Asociación

Índices de mapas de bits

Los índices de mapas de bits son generalmente bastante pequeños en comparación con el tamaño real de la relación. Los registros suelen tener entre decenas y centenares de bytes de longitud, mientras que un solo bit representa a un registro en el mapa de bits. Por tanto, el espacio ocupado por cada mapa de bits suele ser menos del uno por ciento del espacio ocupado por la relación.

Por ejemplo, si el tamaño del registro de una relación dad es de 100 bytes, el espacio ocupado por cada mapa de bits sería la octava parte del uno por ciento del espacio ocupado por la relación

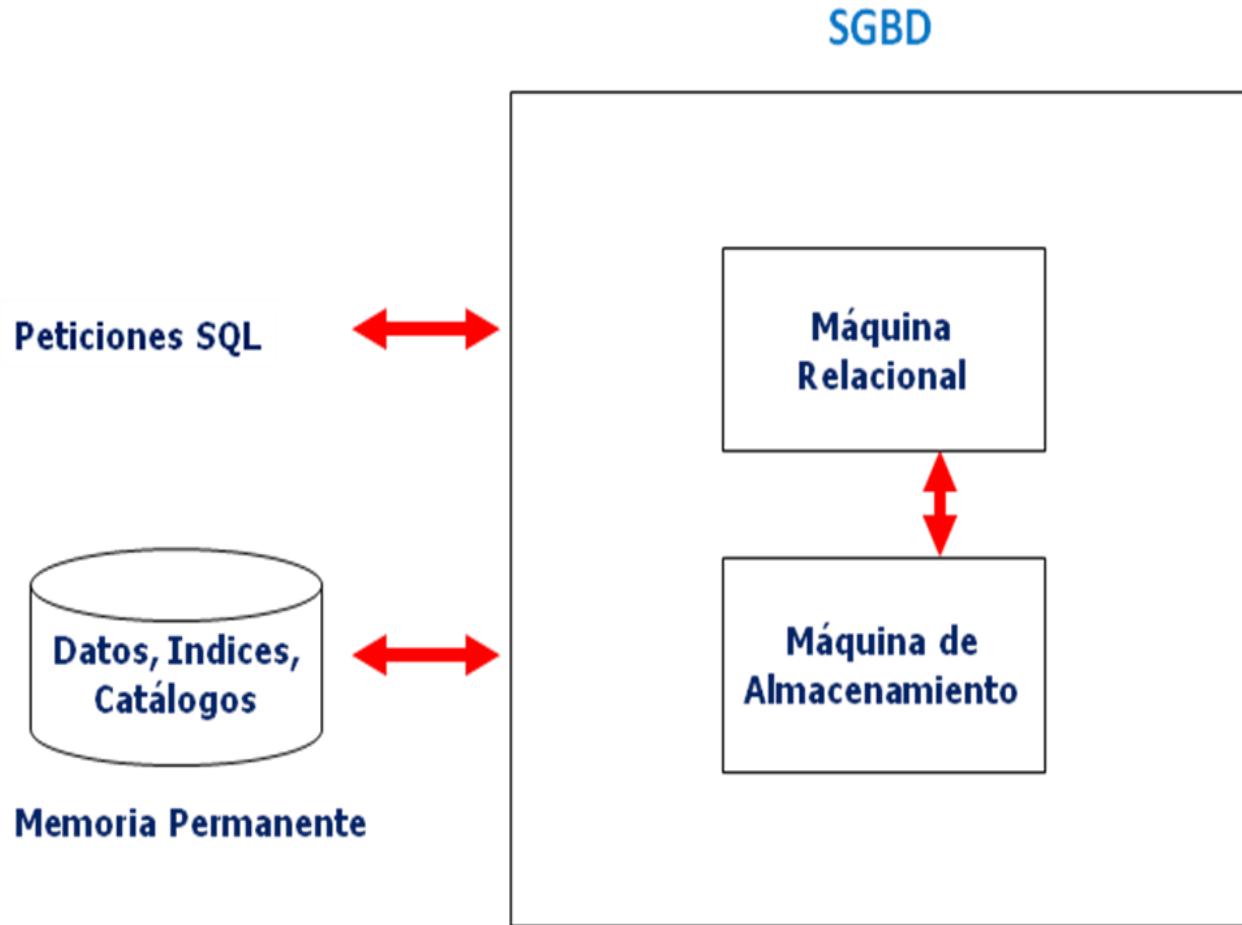
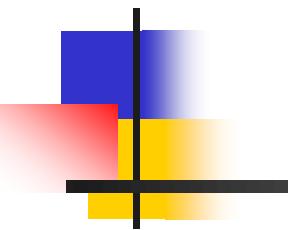
Consultas en Bases de datos



Procesamiento y optimización de consultas

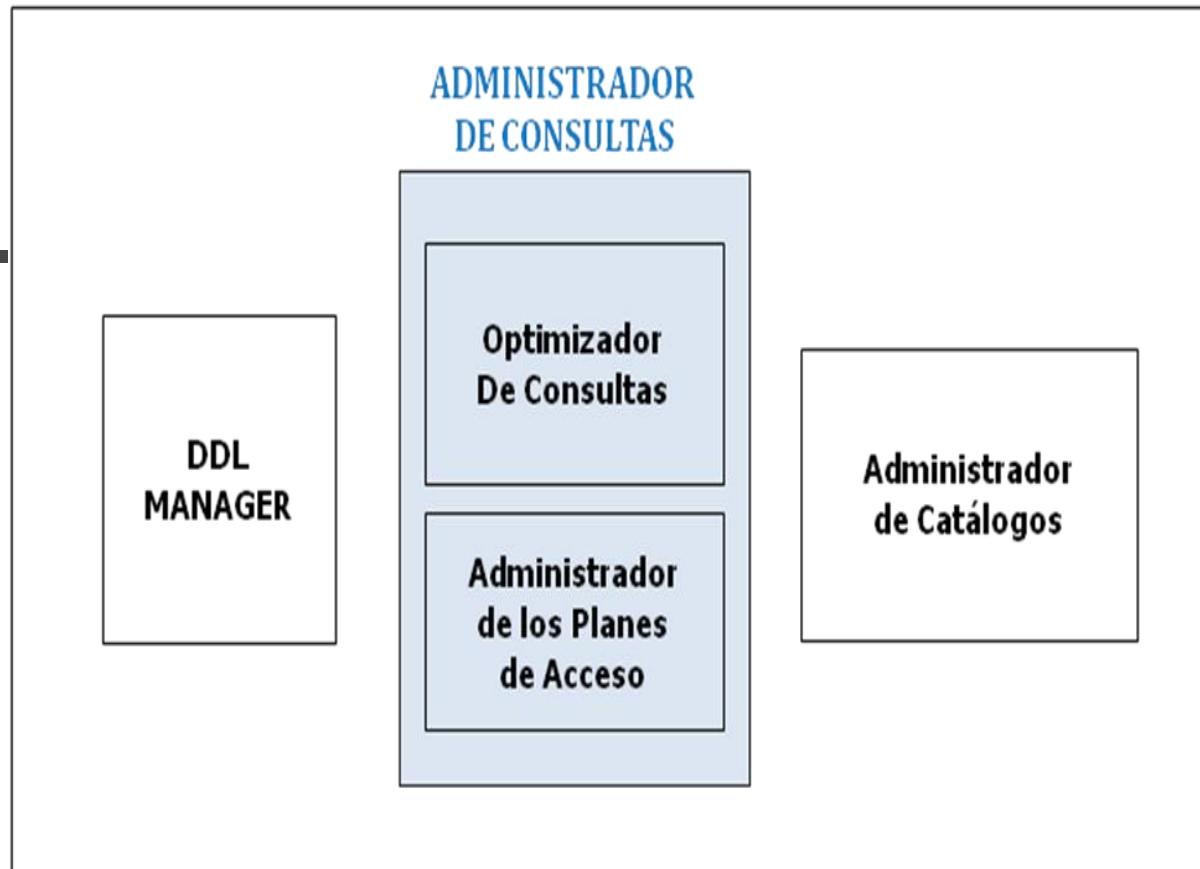
Ideas preliminares

(Diagrama de bloques de la estructura de un SGBD)



Ideas preliminares (Diagrama de bloques de la estructura de un SGBD)

Máquina Relacional

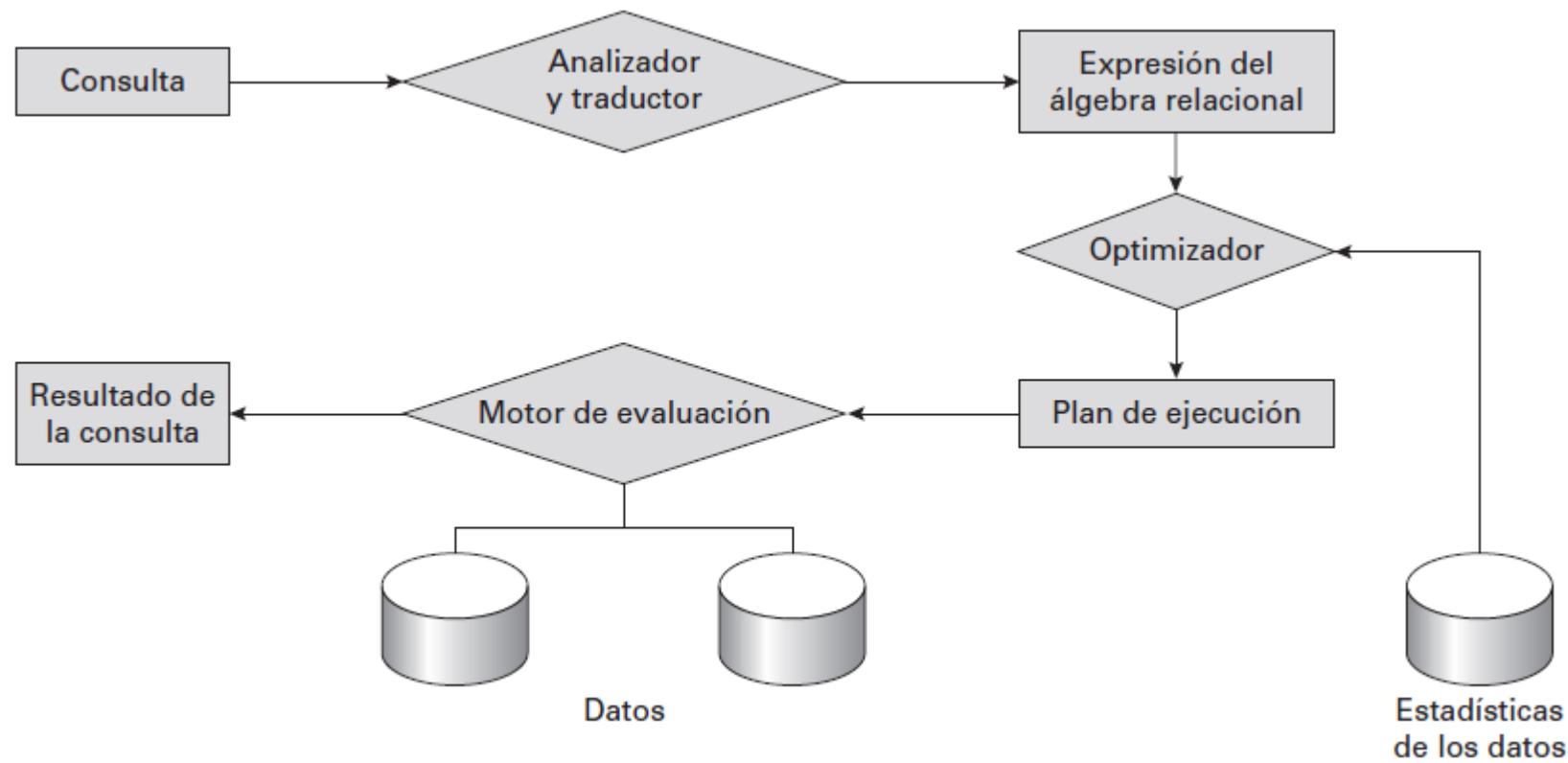


Procesamiento de consultas

En el procesamiento de consultas se debe entender la serie de actividades implicadas en la extracción de datos de una base de datos, con el objetivo de optimizar el proceso de consultas.

Etapas en el procesamiento de consultas

Un lenguaje como SQL es adecuado para el uso humano, pero es inapropiado para una representación interna en el sistema de la consulta; es más útil una representación basada en el Álgebra Relacional Extendida.



Varias maneras de expresar consultas SQL

Tener en cuenta que dada una consulta, hay generalmente varias maneras de expresar una consulta en SQL, además cada consulta SQL se puede traducir a diferentes expresiones del álgebra relacional.

```
select saldo  
from cuenta  
where saldo < 2500
```

Se puede
traducir a

Expresión 1

$$\sigma_{saldo < 2500} (\Pi_{saldo} (cuenta))$$

Expresión 2

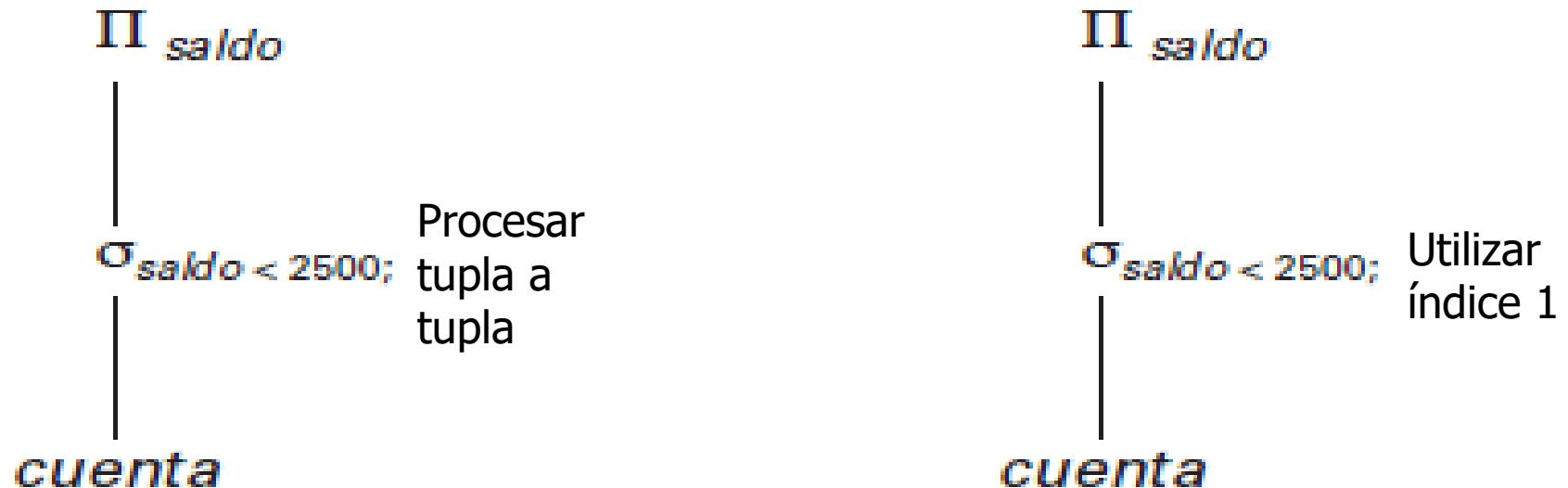
$$\Pi_{saldo} (\sigma_{saldo < 2500} (cuenta))$$

Varios algoritmos para resolver una expresión del álgebra relacional

Una expresión del álgebra relacional se puede implementar con varios algoritmos. Por ejemplo, la siguiente expresión:

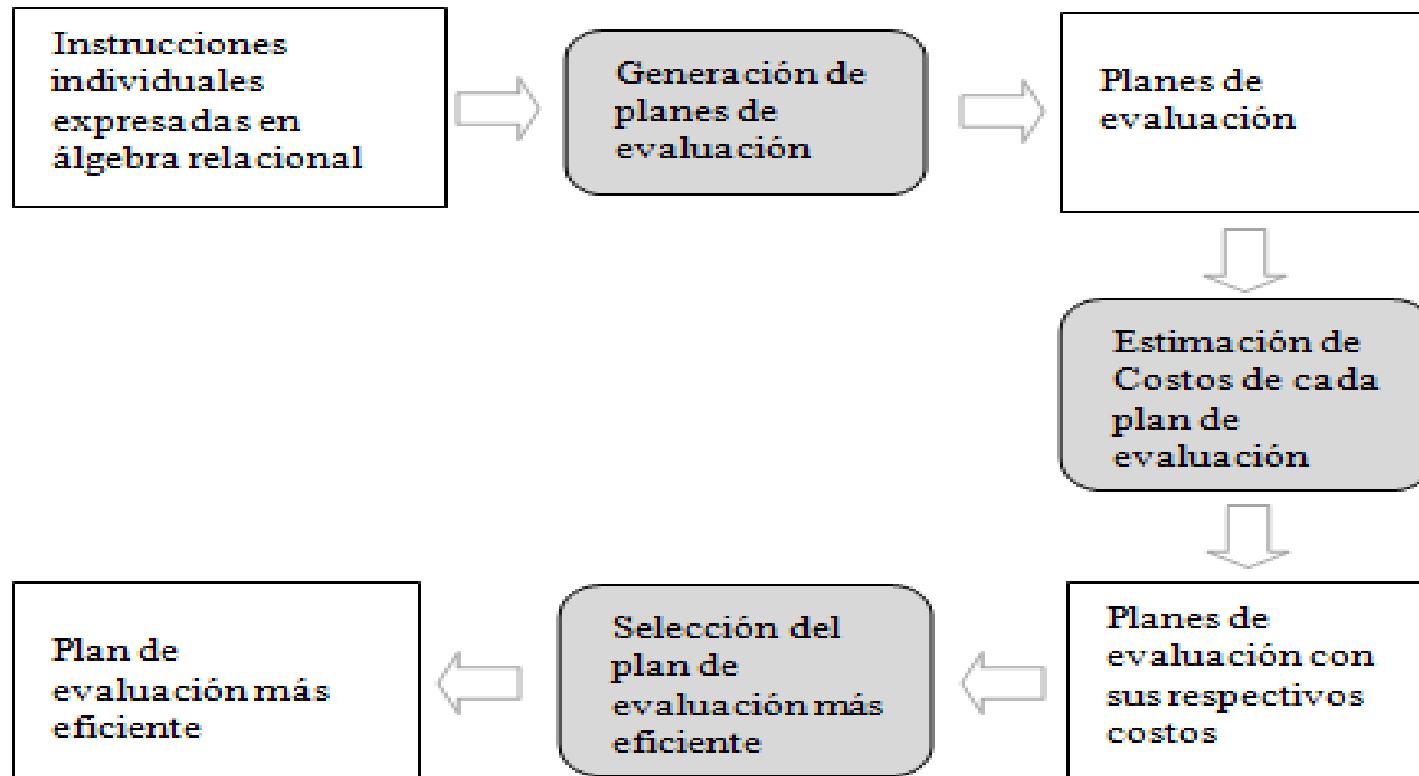
$$\Pi_{saldo} (\sigma_{saldo < 2500} (cuenta))$$

Puede implicar su implementación en dos algoritmos:



Etapas en la optimización de consultas

El proceso de optimización de la consulta implica a su vez sub etapas, al final de estas se determina el plan de evaluación más eficiente.



Optimización de consultas

La **optimización de consultas** es el proceso de selección del plan de evaluación de las consultas más eficiente de entre las muchas estrategias generalmente disponibles para el procesamiento de una consulta dada, especialmente si la consulta es compleja.

Consideraciones a tener en cuenta en la **optimización de consultas**:

- Hallar una expresión que sea equivalente a la expresión dada, pero de ejecución más eficiente.
- Elegir una estrategia detallada para el procesamiento de la consulta:
 - Selección del algoritmo que se utilizará para ejecutar una operación.
 - Selección de los índices que se van a emplear.

La diferencia en coste (en términos de tiempo de evaluación) entre una estrategia buena y una mala suele ser sustancial, y puede ser de varios órdenes de magnitud. Por tanto, merece la pena que el sistema pase una cantidad importante de tiempo en la selección de una buena estrategia para el procesamiento de la consulta.

Optimización de consultas

La discusión de optimización de consultas se ilustrará considerando el siguiente esquema de base de datos:

Cliente (id-cliente nombre-cliente calle-cliente ciudad-cliente)

Sucursal (nombre-sucursal, ciudad-sucursal, activo)

Cuenta (número-cuenta, nombre-sucursal, saldo)

Impositor (nombre-cliente, número-cuenta)

Optimización de consultas

Dada una expresión del álgebra relacional, es labor del optimizador de consultas diseñar un plan de evaluación de consultas que calcule el mismo resultado que la expresión dada, y que sea la manera menos costosa de generar ese resultado (o, como mínimo, que no sea mucho más costoso que la manera menos costosa).

Para hallar el plan de evaluación de consultas menos costoso el optimizador necesita generar planes alternativos que produzcan el mismo resultado que la expresión dada y escoger el menos costoso.

La generación de planes de evaluación de consultas implica dos etapas:

- (1) Generar expresiones equivalentes a la expresión dada. Se efectúa mediante las **reglas de equivalencia**.
- (2) Llevar la relación de las expresiones resultantes en maneras alternativas, de generar planes de evaluación de consultas alternativos. Luego escoger el plan de evaluación basado en la estimación del costo (Deseablemente el menos costoso). Este proceso se denomina: **optimización basada en costes**.

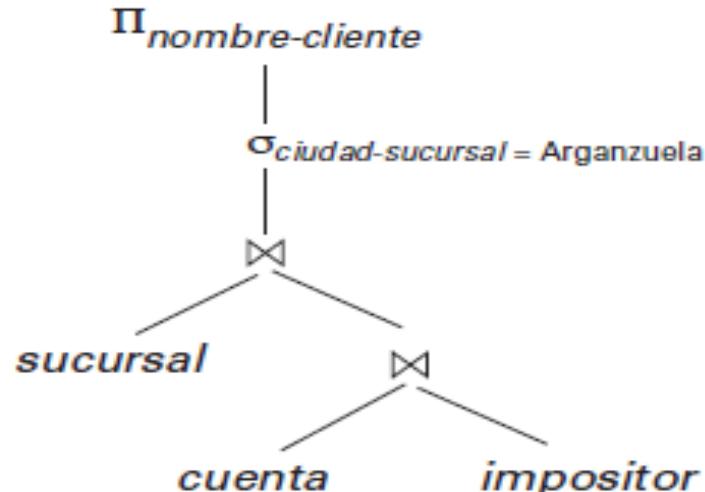
Optimización de consultas

Transformación de expresiones relacionales

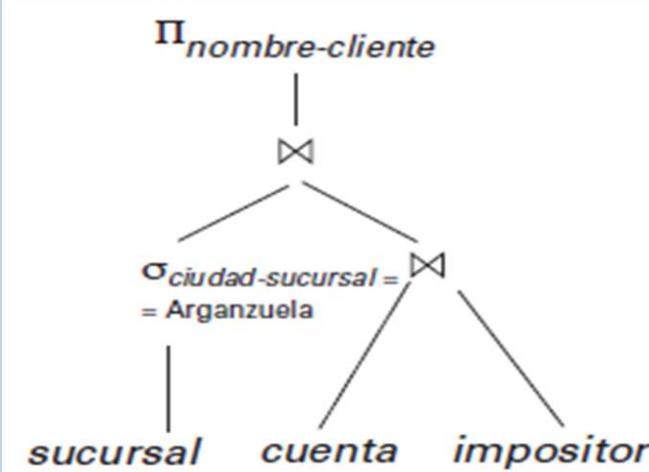
Dos expresiones del álgebra relacional son **equivalentes** si, en cada ejemplar legal de la base de datos, las dos expresiones generan el mismo conjunto de tuplas. (Tener en cuenta que un ejemplar legal de la base de datos es la que satisface todas las restricciones de integridad especificadas en el esquema de la base de datos.)

Ejemplo: «Hallar los nombres de todos los clientes que tengan una cuenta en cualquier sucursal ubicada en Arganzuela»

$$\Pi_{\text{nombre-cliente}} (\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}}} (sucursal \bowtie (cuenta \bowtie impositor)))$$



$$\Pi_{\text{nombre-cliente}} (\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}}} (sucursal)) \bowtie (cuenta \bowtie impositor)$$



Reglas de equivalencia en la optimización de consultas

Una **regla de equivalencia** dice que las expresiones de dos formas son equivalentes. Es decir, se puede sustituir una expresión de la primera forma por una expresión de la segunda forma, o viceversa, ya que las dos expresiones generan el mismo resultado en cualquier base de datos válida.

Se utilizan:

$\theta, \theta_1, \theta_2$, etc., para denotar los predicados (Condiciones).

L, L_1, L_2, L_3 , etc., para denotar las listas de atributos.

E, E_1, E_2 , etc., para denotar las expresiones del álgebra relacional.

Regla 1: Las operaciones de selección conjuntivas pueden dividirse en una secuencia de selecciones individuales. Esta transformación se denomina cascada de σ .

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

Reglas de equivalencia en la optimización de consultas

Regla 2: Las operaciones de selección son **comutativas**.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

Regla 3: Sólo son necesarias las últimas operaciones de una secuencia de operaciones de proyección, las demás pueden omitirse. Esta transformación también puede denominarse cascada de Π .

$$\Pi_{L_1}(\Pi_{L_2}(\dots(\Pi_{L_1}(E))\dots)) = \Pi_{L_1}(E)$$

Reglas de equivalencia en la optimización de consultas

Regla 4: Las selecciones pueden combinarse con los productos cartesianos y con las reuniones zeta (join).

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

Esta expresión es precisamente la definición de la reunión zeta.

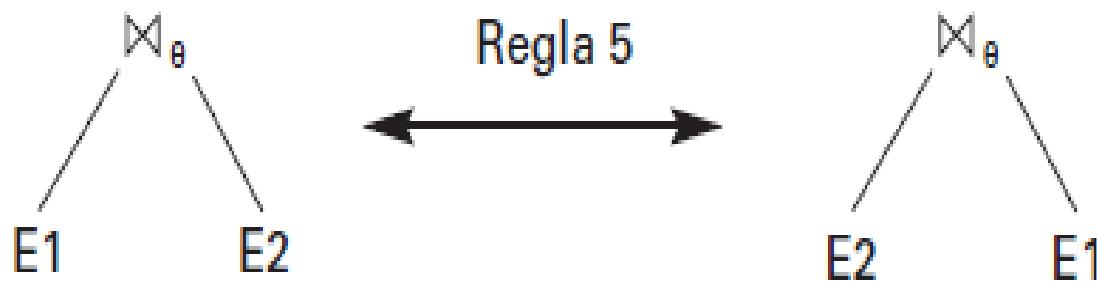
$$\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$$

Reglas de equivalencia en la optimización de consultas

Regla 5: Las operaciones de reunión zeta son commutativas.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

El orden de los atributos es diferente en el término de la derecha y en el de la izquierda, por lo que la equivalencia no se cumple si se tiene en cuenta el orden de los atributos. Sin embargo, generalmente se ignora el orden de los atributos, porque estos se pueden subsanar agregando operaciones de proyección.

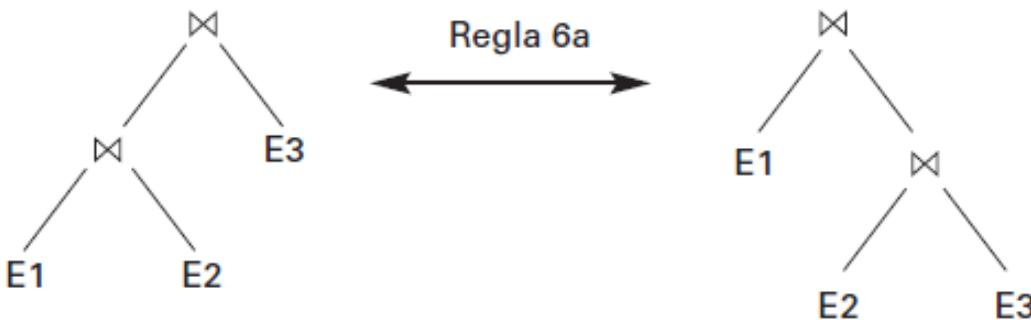


Reglas de equivalencia en la optimización de consultas

Regla 6:

- a. Las operaciones de reunión natural son **asociativas**.

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$



- b. Las reuniones zeta son asociativas en el sentido siguiente::

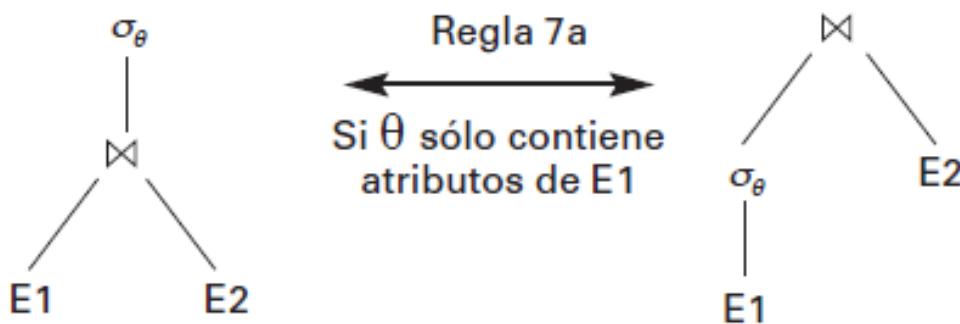
$$\begin{aligned} & (E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = \\ & = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3) \end{aligned}$$

Reglas de equivalencia en la optimización de consultas

Regla 7: La operación de selección se distribuye por la operación de reunión zeta bajo las dos condiciones siguientes:

- a. Se distribuye cuando todos los atributos de la condición de selección θ_0 implican únicamente los atributos de una de las expresiones (por ejemplo, E_1) que se están reuniendo.

$$\sigma_{\theta_0} (E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_0} (E_1)) \bowtie_{\theta} E_2$$



- b. Se distribuye cuando la condición de selección θ_1 implica únicamente los atributos de E_1 y θ_2 implica únicamente los atributos de E_2 .

$$\sigma_{\theta_1 \wedge \theta_2} (E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1} (E_1)) \bowtie_{\theta} (\sigma_{\theta_2} (E_2))$$

Reglas de equivalencia en la optimización de consultas

Regla 8: La operación proyección se distribuye por la operación de reunión zeta bajo las condiciones siguientes:

- a. Sean L_1 y L_2 atributos de E_1 y de E_2 , respectivamente. Supóngase que la condición de reunión θ implica únicamente los atributos de $L_1 \cup L_2$. Entonces,

$$\Pi_{L_1 \cup L_2} (E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1}(E_1)) \bowtie_{\theta} (\Pi_{L_2}(E_2))$$

- b. Considérese una reunión $E_1 \theta E_2$. Sean L_1 y L_2 conjuntos de atributos de E_1 y de E_2 , respectivamente. Sean L_3 los atributos de E_1 que están implicados en la condición de reunión θ , pero que no están en $L_1 \cup L_2$, y sean L_4 los atributos de E_2 que están implicados en la condición de reunión θ , pero que no están en $L_1 \cup L_2$. Entonces,

$$\Pi_{L_1 \cup L_2} (E_1 \bowtie_{\theta} E_2) = \Pi_{L_1 \cup L_2} ((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_{\theta} (\Pi_{L_2 \cup L_4}(E_2)))$$

Reglas de equivalencia en la optimización de consultas

Regla 9: Las operaciones de conjuntos unión e intersección son commutativas.

$$E_1 \cup E_2 = E_2 \cup E_1$$

$$E_1 \cap E_2 = E_2 \cap E_1$$

La diferencia de conjuntos no es commutativa.

Regla 10: La unión y la intersección de conjuntos son asociativas.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

Reglas de equivalencia en la optimización de consultas

Regla 11: La operación de selección se distribuye por las operaciones de unión, intersección y diferencia de conjuntos.

$$\sigma_P (E_1 - E_2) = \sigma_P (E_1) - \sigma_P (E_2)$$

Además (cumple para intersección más no para unión)

$$\sigma_P (E_1 \cap E_2) = \sigma_P (E_1) \cap \sigma_P (E_2)$$

Regla 12: La operación de proyección se distribuye por la operación unión..

$$\Pi_L (E_1 \cup E_2) = (\Pi_L (E_1)) \cup (\Pi_L (E_2))$$

Ejemplo de transformaciones

La ilustración del uso de las reglas de equivalencia se efectuará con el siguiente esquema de base de datos:

Cliente (id-cliente nombre-cliente calle-cliente ciudad-cliente)

Sucursal (nombre-sucursal, ciudad-sucursal, activo)

Cuenta (número-cuenta, nombre-sucursal, saldo)

Impositor (nombre-cliente, número-cuenta)

Ejemplo: «Hallar los nombres de todos los clientes que tengan una cuenta en cualquier sucursal ubicada en Arganzuela y cuyo saldo sea superior a 1000 euros.»

La nueva consulta en álgebra relacional se puede expresar como:

$$\prod_{\text{nombre-cliente}} (\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela»} \wedge \text{saldo} > 1000} (\text{sucursal} \bowtie (\text{cuenta} \bowtie \text{impositor})))$$

No se puede aplicar el predicado de la selección directamente a la relación *sucursal*, ya que el predicado implica atributos tanto de la relación *sucursal* como de la relación *cuenta*.

Ejemplo de transformaciones

Aplicando la regla 6.a (asociatividad de la reunión natural):

$$\prod_{\text{nombre-cliente}} (\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}} \wedge \text{saldo} > 1000) ((\text{sucursal} \bowtie \text{cuenta}) \bowtie \text{impositor})$$

Luego, empleando la regla 7.a, se puede reescribir la consulta como

$$\prod_{\text{nombre-cliente}} ((\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}} \wedge \text{saldo} > 1000) (\text{sucursal} \bowtie \text{cuenta})) \bowtie \text{impositor}$$

Empleando la regla 1 se puede partir la selección en dos, para obtener la sub expresión siguiente:

$$\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}} (\sigma_{\text{saldo} > 1000} (\text{sucursal} \bowtie \text{cuenta}))$$

La última forma de la expresión de selección ofrece una nueva oportunidad de aplicar la regla «llevar a cabo primero las selecciones», que da lugar a la sub expresión:

$$\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}} (\text{sucursal}) \bowtie \sigma_{\text{saldo} > 1000} (\text{cuenta})$$

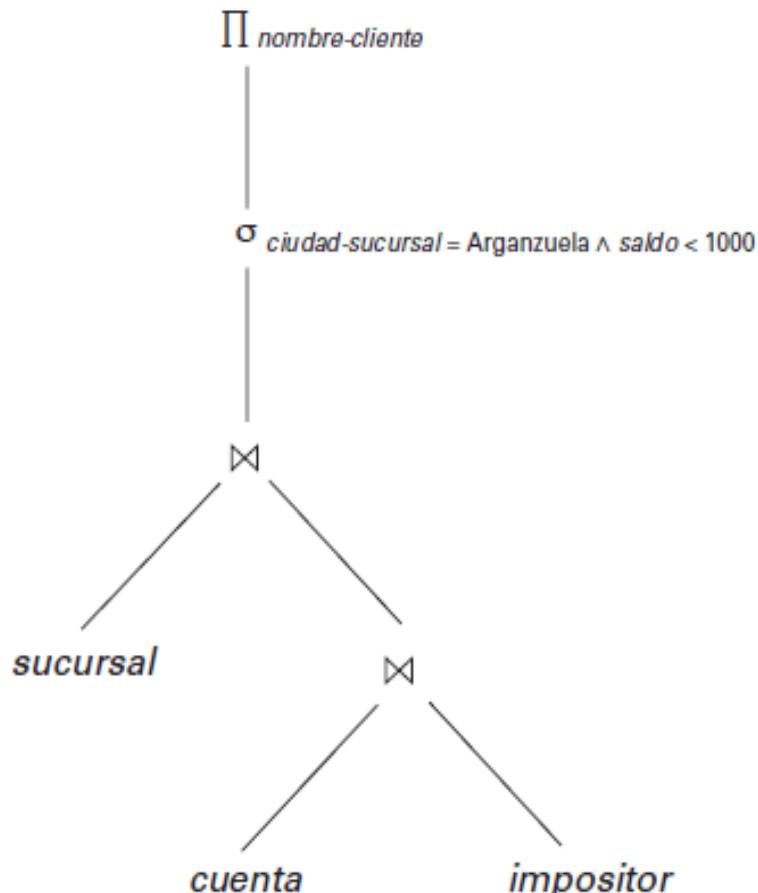
Reemplazando esta sub expresión en la expresión del álgebra, se tiene:

$$\prod_{\text{nombre-cliente}} ((\sigma_{\text{ciudad-sucursal} = \text{«Arganzuela}} (\text{sucursal}) \bowtie \sigma_{\text{saldo} > 1000} (\text{cuenta})) \bowtie \text{impositor})$$

Ejemplo de transformaciones

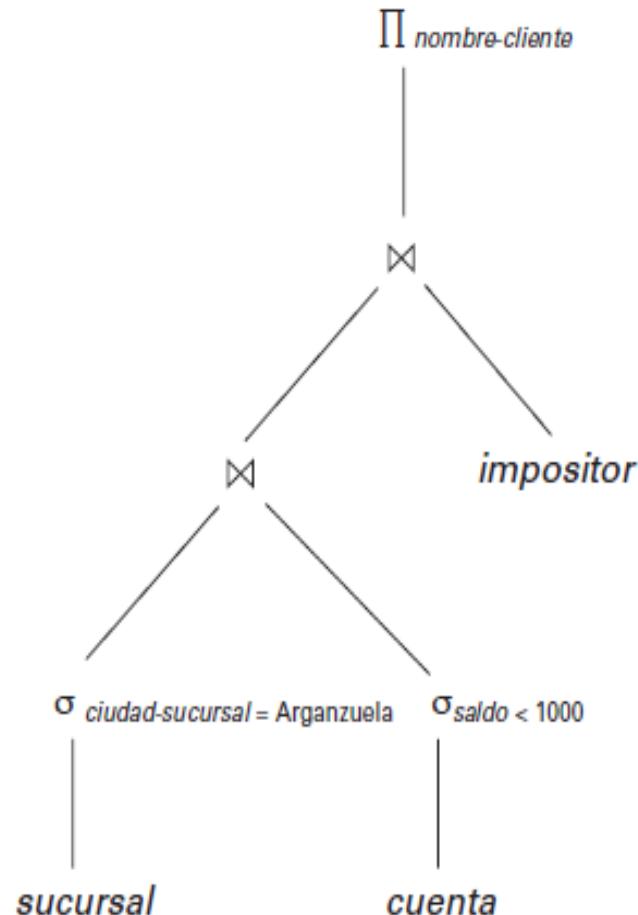
Expresión inicial

$$\prod_{\text{nombre-cliente}} (\sigma_{ciudad-sucursal = \text{Arganzuela} \wedge saldo > 1000} (sucursal \bowtie (cuenta \bowtie impositor)))$$



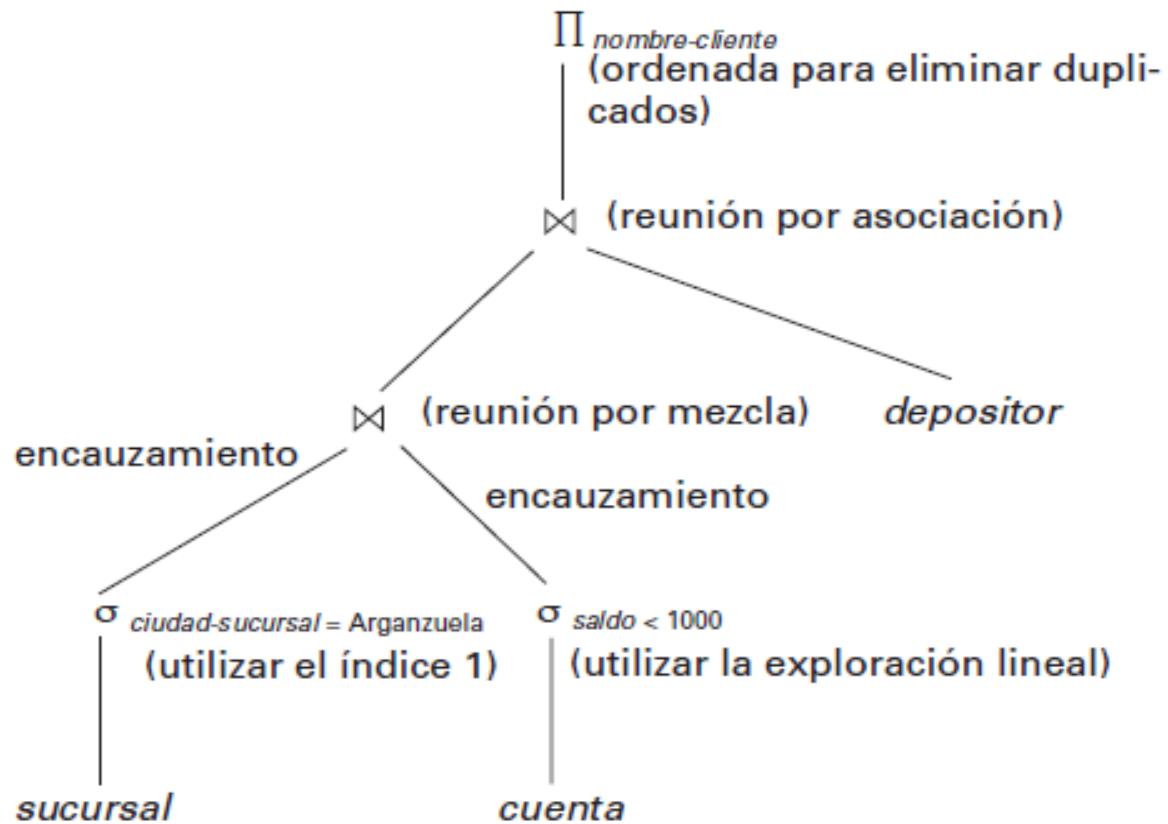
Expresión final

$$\prod_{\text{nombre-cliente}} ((\sigma_{ciudad-sucursal = \text{Arganzuela}} (sucursal) \bowtie \sigma_{saldo > 1000} (cuenta)) \bowtie impositor)$$



Elección de los planes de evaluación

La generación de expresiones sólo es una parte del proceso de optimización de consultas, ya que cada operación de la expresión puede implementarse con algoritmos diferentes. Por tanto, se necesita un plan de evaluación para definir exactamente el algoritmo que se utilizará para cada operación y el modo en que se coordinará la ejecución de las operaciones.



Optimización basada en el coste

Los **optimizadores basados en el coste** generan una gama de planes de evaluación a partir de la consulta dada empleando las reglas de equivalencia y escogen el de coste mínimo.

```
procedure hallamejorplan(S)
    if (mejorplan[S].coste ≠ ∞)
        return mejorplan[S]
    if (S contiene sólo una relación)
        establecer mejorplan[S].plan y mejorplan[S].coste en términos de la mejor forma de acceder a S
    // mejorplan[S] no se ha calculado anteriormente, hay que calcularlo ahora
    else for each subconjunto no vacío S1 de S tal que S1 ≠ S
        P1 = hallamejorplan(S1)
        P2 = hallamejorplan(S - S1)
        A = mejor algoritmo para reunir los resultados de P1 y P2
        coste = P1.coste + P2.coste + coste de A
        if coste < mejorplan[S].coste
            mejorplan[S].coste = coste
            mejorplan[S].plan = «ejecutar P1.plan; ejecutar P2.plan; reunir resultados de P1 y de P2 utilizando A»
    return mejorplan[S]
```

Optimización basada en el coste (Estimaciones estadísticas)

El costo de cada operación, depende del tamaño y de otras estadísticas de sus valores de entrada, por ejemplo en una expresión como $a \bowtie (b \bowtie c)$, para estimar el costo de combinar a con $(b \bowtie c)$, hay que hacer estimaciones de estadísticas, como el tamaño de $b \bowtie c$.

Información del catálogo

Los catálogos de los SGDD almacenan la siguiente información estadística sobre las relaciones de las bases de datos:

- nr , el número de tuplas de la relación r .
- br , el número de bloques que contienen tuplas de la relación r .
- tr , el tamaño de cada tupla de la relación r en bytes.
- fr , el factor de bloqueo de la relación r ; es decir, el número de tuplas de la relación r que caben en un bloque.
- $V(A, r)$, el número de valores distintos que aparecen en la relación r para el atributo A . Este valor es igual que el tamaño de $\Pi A(r)$. Si A es una clave de la relación r , $V(A, r)$ es nr .

Optimización basada en el coste (Estimaciones estadísticas)

Ejemplo de estimación del tamaño de la selección

- La estimación del tamaño de una operación selección depende del predicado de la selección.
- Primero se considerará un solo predicado de igualdad, luego un predicado de comparación y finalmente combinaciones de predicados.

$\sigma A=a (r)$: Si se supone una distribución uniforme de los valores, se puede estimar que el resultado de la selección tiene $nr / V(A,r)$ tuplas. En pocas palabras cada valor aparece con la misma probabilidad, lo que puede ser irreal.

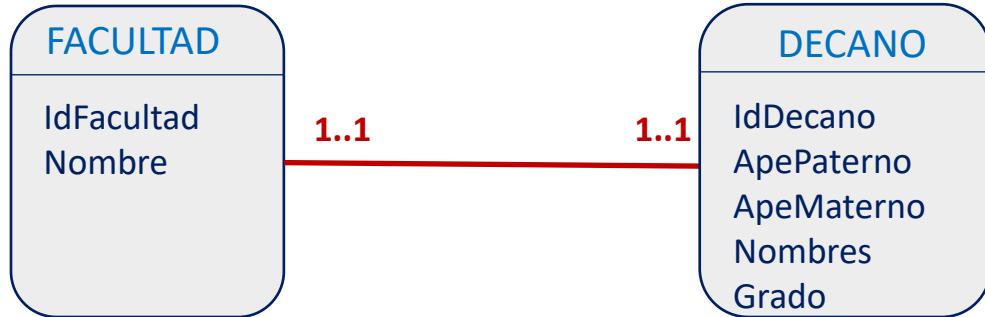
Considérese una selección de la forma $\sigma A \leq v (r)$. Si el valor usado en la comparación(v) esta disponible en el momento de la estimación del costo, puede hacerse una estimación mas precisa. Los valores mínimo y máximo ($\min (A, r)$ y $\max (A, r)$) del atributo pueden almacenarse en el catalogo.

Suponiendo que los valores están distribuidos de manera uniforme, se puede estimar el número de registros que cumplirán la condición $A \leq v$, como 0 si $v < \min (A, r)$, como nr , si $v \geq \max (A, r)$

$$n_r \cdot \frac{v - \min (A, r)}{\max (A, r) - \min (A, r)}$$

Diseño Lógico y Físico

Relaciones 1 : 1



FACULTAD(IdFacultad, Nombre, IdDecano)

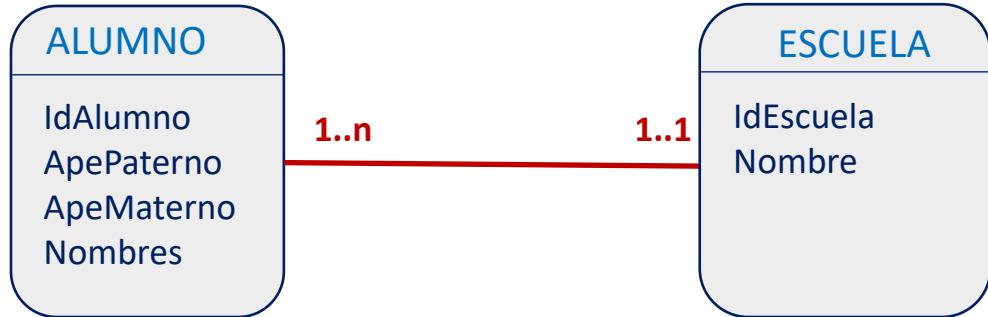
DECANO(IdDecano, ApePaterno, ApeMaterno, Nombres, Grado)

```
CREATE TABLE FACULTAD
(
    IdFacultad varchar(12) PRIMARY KEY,
    Nombre varchar(50),
    IdDecano varchar(12) UNIQUE,
    FOREIGN KEY (IdDecano) REFERENCES DECANO(IdDecano)
)
```

```
CREATE TABLE DECANO
(
    IdDecano varchar(12) PRIMARY KEY,
    ApePaterno varchar(15),
    ApeMaterno varchar(15),
    Nombres varchar(15),
    Grado varchar(64)
)
```

Diseño Lógico y Físico

Relaciones 1 : N



ALUMNO(IdAlumno, ApePaterno, ApeMaterno, Nombres, IdEscuela)

ESCUELA(IdEscuela, Nombre)

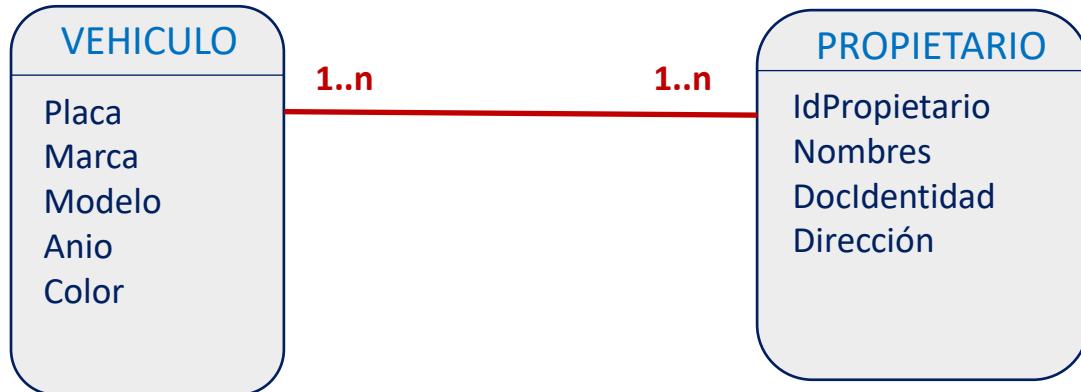
Para representar la relación, la clave primaria de la entidad que tiene cardinalidad máxima N va a la entidad que tiene la cardinalidad máxima 1

```
CREATE TABLE ALUMNO
(
    IdAlumno varchar(12) PRIMARY KEY,
    ApePaterno varchar(15),
    ApeMaterno varchar(15),
    Nombres varchar(15),
    IdEscuela varchar(12),
    FOREIGN KEY (IdEscuela) REFERENCES ESCUELA(IdEscuela)
)
```

```
CREATE TABLE ESCUELA
(
    IdEscuela varchar(12) PRIMARY KEY,
    Nombre varchar(50)
)
```

Diseño Lógico y Físico

Relaciones N : M



VEHICULO(Placa, Marca, Modelo, Anio, Color)

PROPIETARIO (IdPropietario, Nombres, DocIdentidad, Direccion)

VEHICULO_PROPIETARIO (Placa, IdPropietario)

Para representar la relación, con las claves primarias de ambas entidades se crea otra tabla.

```

CREATE TABLE VEHICULO
(
    Placa          varchar(7) PRIMARY KEY,
    Marca         varchar(15),
    Modelo        varchar(15),
    Anio          int,
    Color         varchar(20),
)
  
```

```

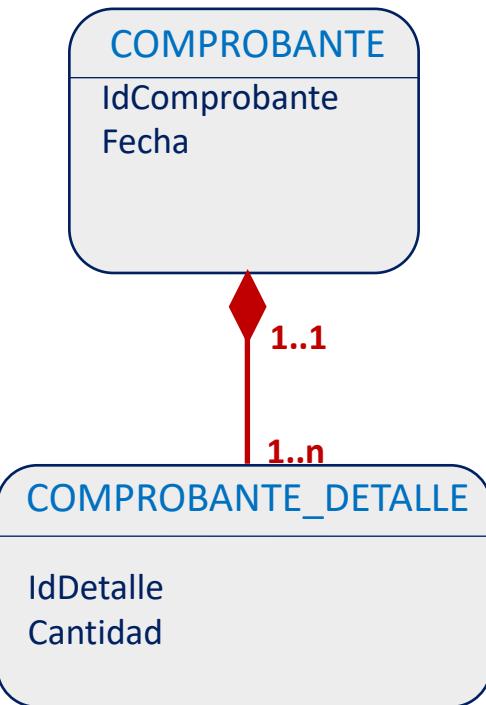
CREATE TABLE ESCUELA
(
    IdPropietario  varchar(12) PRIMARY KEY,
    Nombres        varchar(50),
    DocIdentidad   varchar(20),
    Direccion      varchar(50)
)
  
```

```

CREATE TABLE VEHICULO_PROPIETARIO
(
    Placa          varchar(7),
    IdPropietario  varchar(12),
    PRIMARY KEY (Placa, IdPropietario),
    FOREIGN KEY (Placa) REFERENCES VEHICULO(Placa),
    FOREIGN KEY (IdPropietario) REFERENCES
    PROPIETARIO(IdPropietario)
)
  
```

Diseño Lógico y Físico

Relaciones de Agregación y Composición



COMPROBANTE(IdComprobante, Fecha, ...)

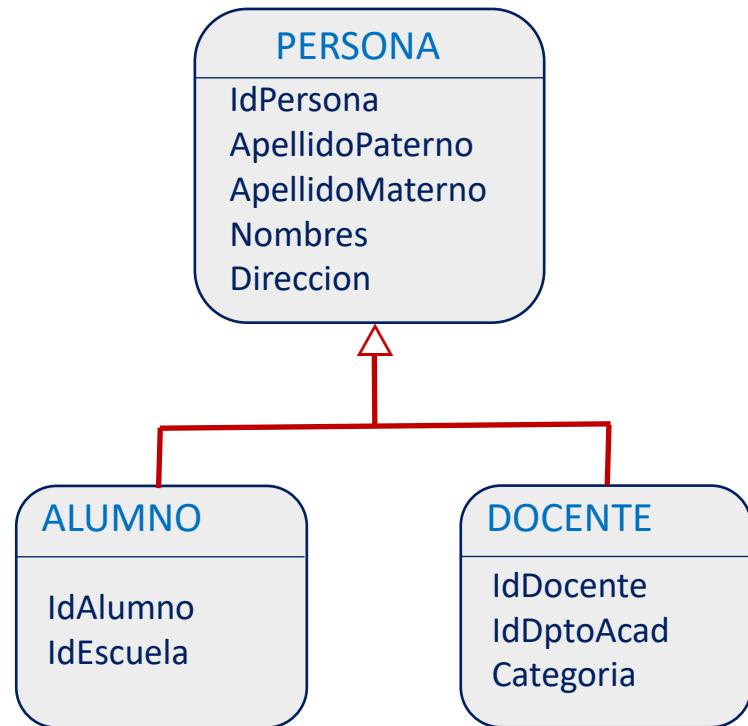
COMPROBANTE_DETALLE (IdComprobante, IdDetalle, Cantidad, ...)

```
CREATE TABLE COMPROBANTE
(
    IdComprobante      varchar(12) PRIMARY KEY,
    Fecha               DateTime,
    ...
)

CREATE TABLE COMPROBANTE_DETALLE
(
    IdComprobante      varchar(12),
    IdDetalle           varchar(12),
    Cantidad            Numeric(18, 4),
    ...
    PRIMARY KEY (IdComprobante, IdDetalle)
    FOREIGN KEY (IdComprobante) REFERENCES COMPROBANTE (IdComprobante)
)
```

Diseño Lógico y Físico

Relaciones de Generalización o especialización (Herencia)



Las relaciones de generalización, se modelan prioritariamente con el esquema de las relaciones de herencia de la POO, tal el Caso A.

Sin embargo, por cuestiones prácticas también se pueden modelar como Caso B y Caso C.

Caso A:

ALUMNO(IdAlumno, ApellidoPaterno, ApellidoMaterno, Nombres, Direccion, IdEscuela, IdPersona)
DOCENTE(IdDocente, ApellidoPaterno, ApellidoMaterno, Nombres, Direccion, IdDptoAcad, Categoria, IdPersona)

Caso B:

PERSONA (IdPersona, ApellidoPaterno, ApellidoMaterno, Nombres, Direccion)
ALUMNO(IdAlumno, IdEscuela, IdPersona)
DOCENTE(IdDocente, IdDptoAcad, Categoria, IdPersona)

Caso C:

PERSONA(IdPersona, ApellidoPaterno, ApellidoMaterno, Nombres, Direccion, IdEscuela, IdDptoAcad, Categoria)

Diseño Lógico

AGENCIA(IdAgenciaNo, Nombre, Responsable)

CLIENTE(IdCliente, Nombres, Doc_Identidad, Dirección, Email)

GARAJE(IdGaraje, Descripción, Dirección, Email)

VEHÍCULO(Placa, Marca, Modelo, Año, NroAsientos, Tipo, **IdGaraje**)

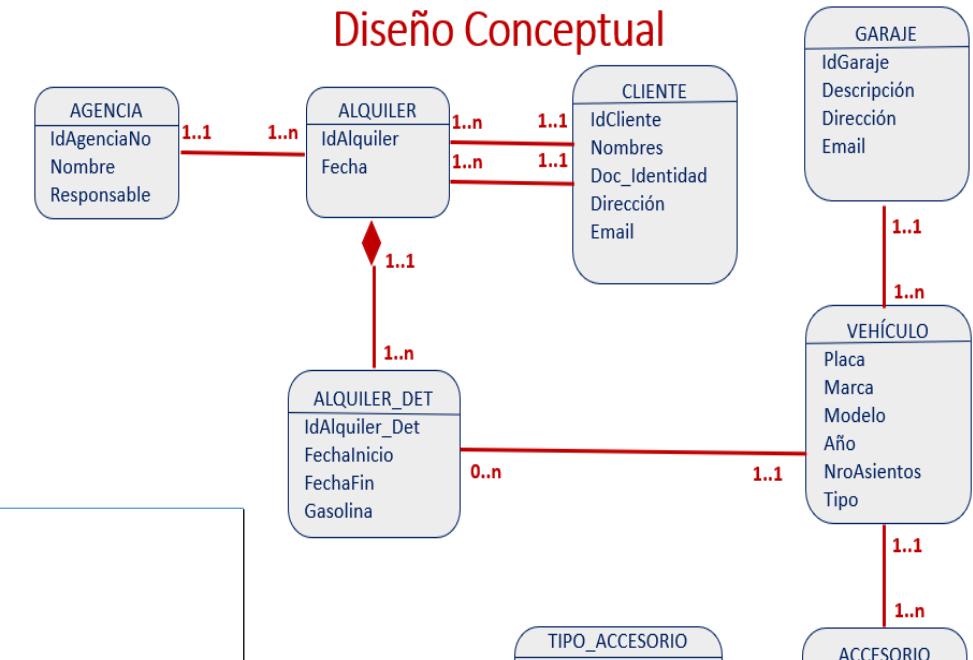
TIPO_ACCESORIO(IdTipoAccesorio, Descripción, Marca, Modelo)

ACCESORIO(IdAccesorio, Placa, **IdTipoAccesorio**)

ALQUILER(IdAlquiler, Fecha, **IdAGenciaNo**, **IdCliente**, **IdAval**)

ALQUILER_DET(IdAlquiler, IdAlquiler_Det, Fechalinicio, FechaFin, Gasolina, **IdVehículo**)

Diseño Conceptual



Diseño Físico

```

CREATE DATABASE SistemaAlquilerVehiculos;
GO
USE SistemaAlquilerVehiculos;
GO
CREATE TABLE AGENCIA (
    IdAgenciaNo VARCHAR(12) NOT NULL PRIMARY KEY,
    Nombre VARCHAR(40) NOT NULL,
    Responsable VARCHAR(40)
);
CREATE TABLE CLIENTE (
    IdCliente VARCHAR(12) NOT NULL PRIMARY KEY,
    Nombres VARCHAR(40) NOT NULL,
    Doc_Identidad VARCHAR(12) NOT NULL,
    Direccion VARCHAR(40),
    Email VARCHAR(40)
);
CREATE TABLE GARAJE (
    IdGaraje VARCHAR(12) NOT NULL PRIMARY KEY,
    Descripcion VARCHAR(40),
    Direccion VARCHAR(40),
    Email VARCHAR(40)
);
CREATE TABLE VEHICULO (
    Placa VARCHAR(12) NOT NULL PRIMARY KEY,
    Marca VARCHAR(24) NOT NULL,
    Modelo VARCHAR(24) NOT NULL,
    Año INT,
    NroAsientos INT,
    Tipo VARCHAR(40),
    IdGaraje VARCHAR(12) NOT NULL,
    FOREIGN KEY (IdGaraje) REFERENCES GARAJE(IdGaraje)
);

```

```

CREATE TABLE TIPO_ACCESORIO (
    IdTipoAccesorio VARCHAR(12) NOT NULL PRIMARY KEY,
    Descripcion VARCHAR(40),
    Marca VARCHAR(24),
    Modelo VARCHAR(24)
);
CREATE TABLE ACCESORIO (
    IdAccesorio VARCHAR(12) NOT NULL PRIMARY KEY,
    Placa VARCHAR(12) NOT NULL,
    IdTipoAccesorio VARCHAR(12) NOT NULL,
    FOREIGN KEY (Placa) REFERENCES VEHICULO(Placa),
    FOREIGN KEY (IdTipoAccesorio) REFERENCES
    TIPO_ACCESORIO(IdTipoAccesorio)
);
CREATE TABLE ALQUILER (
    IdAlquiler VARCHAR(12) NOT NULL PRIMARY KEY,
    Fecha DATE NOT NULL,
    IdAgenciaNo VARCHAR(12) NOT NULL,
    IdCliente VARCHAR(12) NOT NULL,
    IdAval VARCHAR(12),
    FOREIGN KEY (IdAgenciaNo) REFERENCES AGENCIA(IdAgenciaNo),
    FOREIGN KEY (IdCliente) REFERENCES CLIENTE(IdCliente)
);
CREATE TABLE ALQUILER_DET (
    IdAlquiler VARCHAR(12) NOT NULL,
    IdAlquiler_Det VARCHAR(12) NOT NULL,
    FechaInicio DATE NOT NULL,
    FechaFin DATE NOT NULL,
    Gasolina INT,
    IdVehiculo VARCHAR(12) NOT NULL,
    PRIMARY KEY (IdAlquiler, IdAlquiler_Det),
    FOREIGN KEY (IdAlquiler) REFERENCES ALQUILER(IdAlquiler),
    FOREIGN KEY (IdVehiculo) REFERENCES VEHICULO(Placa)
);

```

DISEÑO DE BASES DE DATOS

Niveles de Abstracción de Base de Datos

MUNDO REAL



objetos y asociaciones con
Sus propiedades y reglas

Valores

Estructura percibida
(no formalizada)

Modelado Conceptual

Esquema
Conceptual

Modelo
Conceptual

Diseño Lógico

Esquema
de base
de datos

SGBD
Modelo
de BD

Diseño Físico

Esquema
Internos

Modelo
Internos

E/R

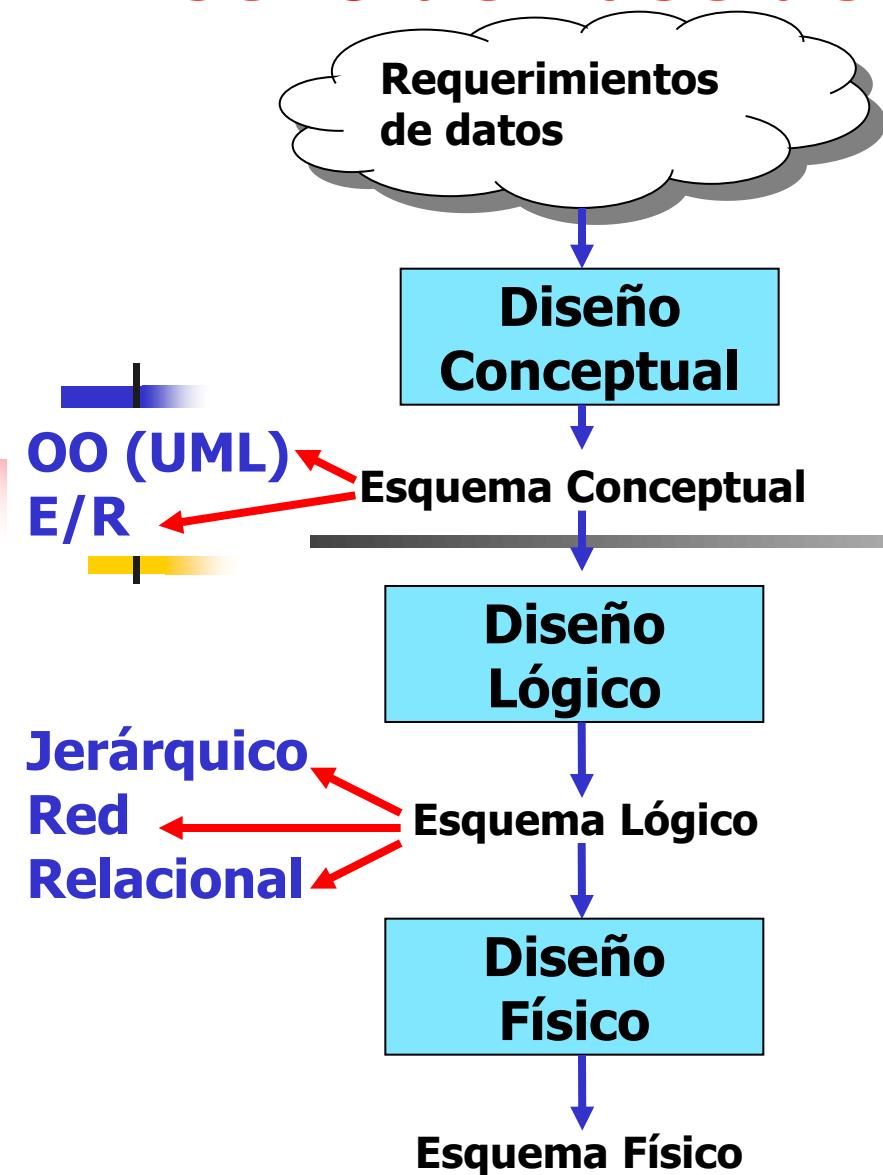
UML

Jerárquico

Red

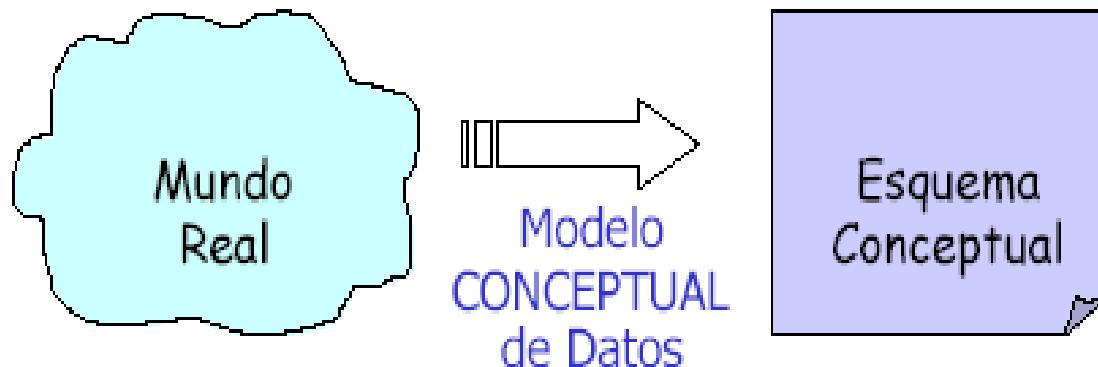
Relacional

Diseño de Base de Datos...



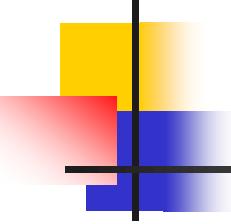
Dependencia de	
Tipo de SGBD	Un SGBD específico
NO	NO
SI	NO
SI	SI

Diseño Conceptual

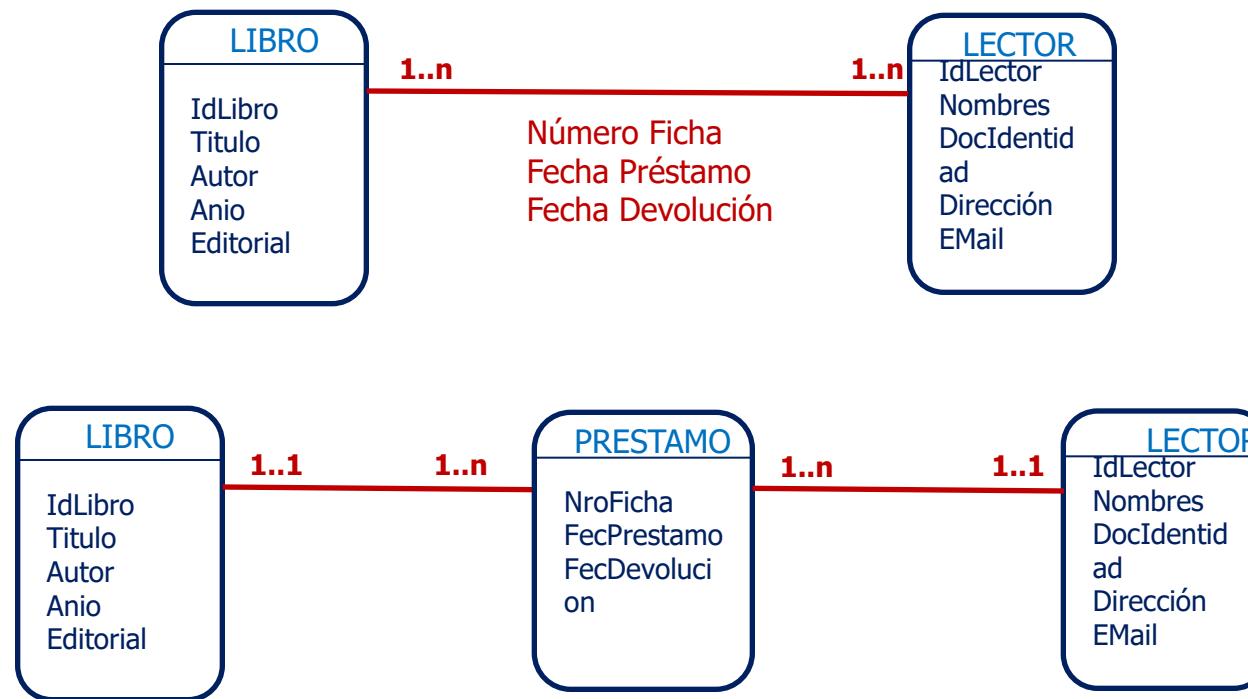


- Conceptos **cercanos** a cómo el usuario percibe la realidad (minimundo)
- Realidad descrita como **entidades** que se **relacionan** entre sí
 - **Entidad**: cosa | objeto | concepto del minimundo
 - **Atributo**: propiedad interesante de alguna entidad
 - **Relación**: asociación | vínculo | interacción entre entidades
- » Modelo Entidad/Relación, MER (ERM, entity-relationship model)
- » Modelos Orientados a Objetos (UML, Unified Modeling Language)

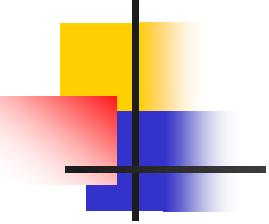
Diseño Conceptual



En relaciones N-M se debe verificar que no hayan atributos que pertenezcan a la relación; en caso de existir, se debe generar una nueva entidad con los atributos de la relación.



Diseño Conceptual

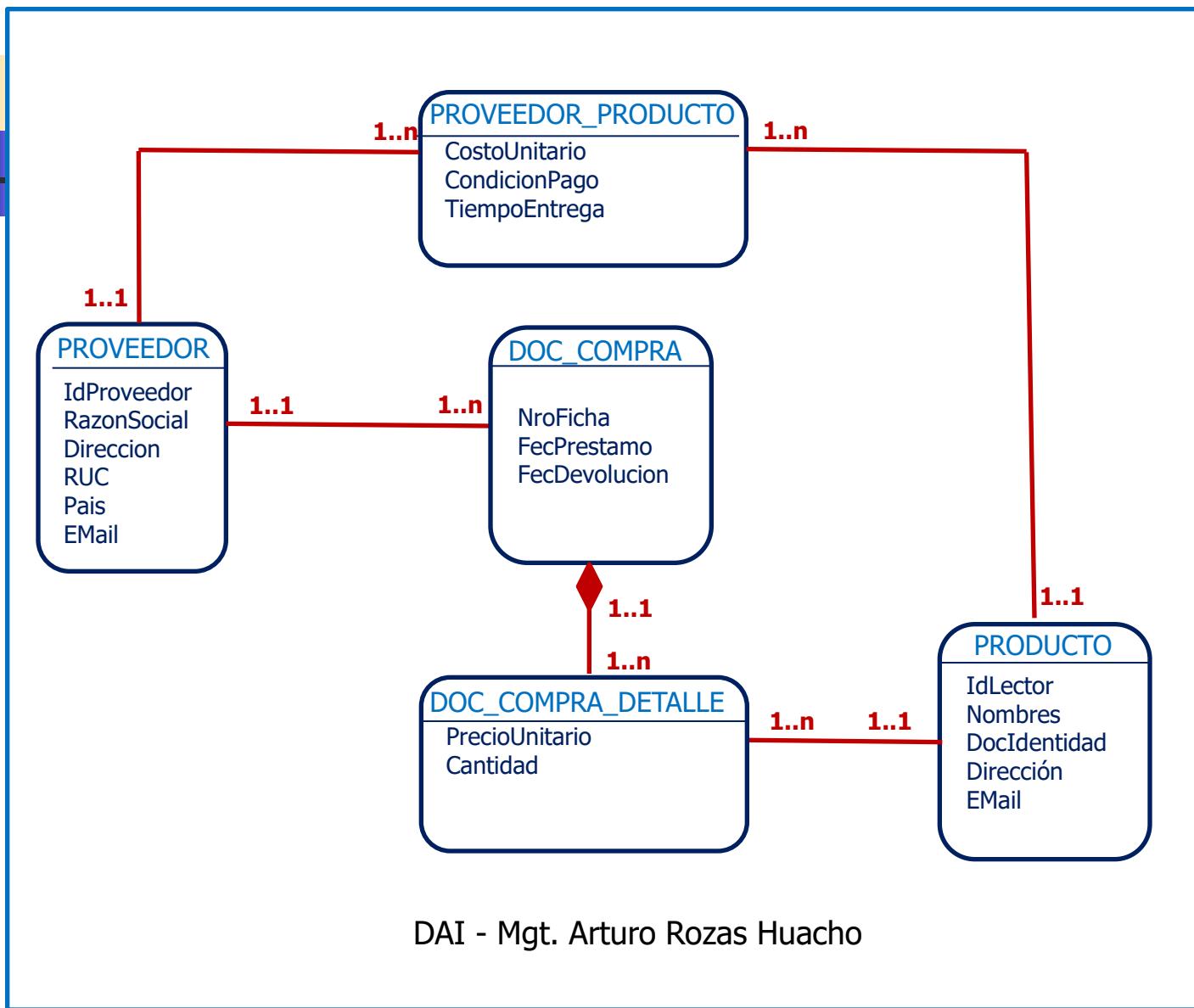


En el modelo lógico, en relaciones N-M si no existen atributos que pertenezcan a la relación, entonces estas se mantienen sin modificación.



Este tipo de relaciones, ya en el diseño lógico implicará una tercera tabla para romper la relación N-M.

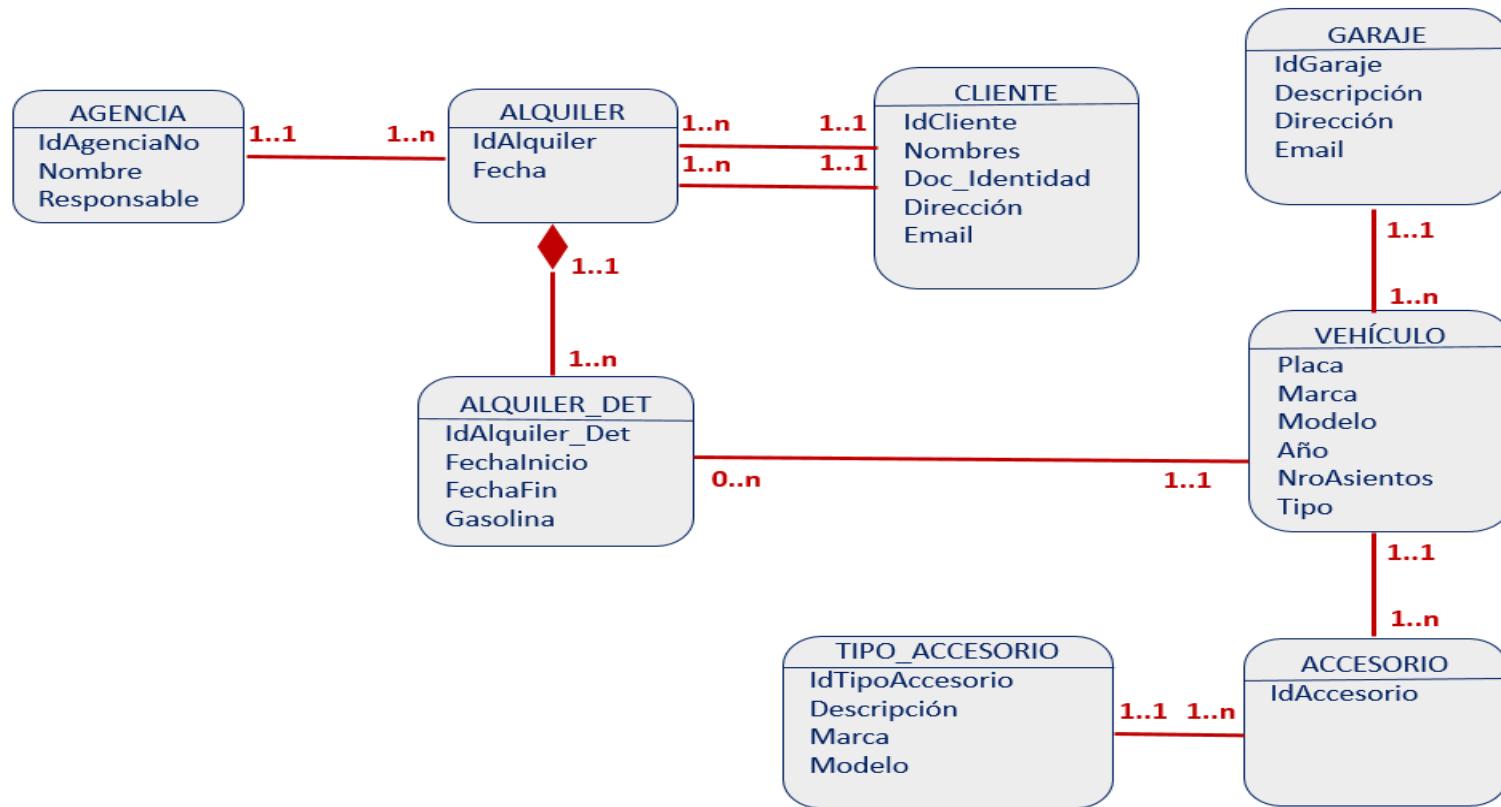
Diseño Conceptual



Diseño Conceptual

ALQUILER DE VEHÍCULOS

La empresa Inkas'car se dedica al alquiler de vehículos, para lo cual, cuenta con: Varios puntos de alquiler (agencias), varios garajes y una flota de vehículos. Los clientes de la empresa, pueden realizar varios procesos de alquiler a la vez; y en cada proceso de alquiler, alquilar varios vehículos. La empresa exige que cada cliente debe contar ser avalado por otro cliente de la empresa. En el proceso de alquiler se registra las fechas de inicio y fin del alquiler de cada vehículo, así como los galones de gasolina en el momento de realizar el alquiler. Los clientes recogen los vehículos de sus respectivos garajes. Cada vehículo está asignado a un solo garaje. De cada vehículo se debe registrar: la placa, la marca, el modelo, el año, el número de asientos y los accesorios con los que está equipado (GPS, radio, etc.). Para cada accesorio se debe registrar el identificador, la descripción y la marca.



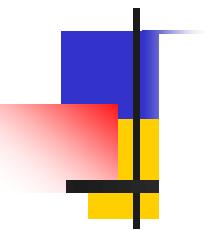
Diseño Lógico

- Permiten describir la **estructura lógica global**: descripción de la implementación
- Conceptos entendibles por usuarios finales, pero **no** lejos de organización física de datos
- Ocultan detalles de implementación, pero conceptos implementables directamente en el sistema
- Los más utilizados en los SGBD comerciales actuales (SQL SERVER, ORACLE)
- **Esquemas lógicos: Relacional, Red, Jerárquico**

Diseño Físico

- Conceptos que describen detalles de almacenamiento de los datos
- Dirigidos a usuarios especialistas en informática
- Describen la **estructura física** de la base de datos:
 - Formato y ordenamiento de registros en los ficheros de datos
 - Tamaños de página, de bloque,...
 - Caminos (o estructuras) de acceso a los datos (ficheros índices, etc.)

...



NORMALIZACIÓN

NORMALIZACIÓN... (Anomalías en el diseño)

Dada la siguiente relación:

Libro(Código, Título, Autor, Editorial, NombreEditorial)

Es está una relación diseñada apropiadamente?

Anomalías de inserción

- No es posible insertar los datos de un libro si no se conoce el nombre de la editorial.
- Se tiene que digitar redundantemente el valor del Nombre de la editorial, cada vez que se registre un libro de la misma editorial.

Anomalías de borrado

- Al borrar un libro y si éste es el único de una editorial, se pierde los datos de la editorial.
- Para borrar una editorial hay que borrar tantas tuplas como libros tenga.

Anomalías de actualización

- Si fuese necesario modificar el Nombre de una editorial, la actualización hay que realizarlo en tantas tuplas como libros tenga la editorial.
- Puede haber tuplas de libros de la misma editorial con distintos valores de nombre de editorial.

NORMALIZACIÓN

Es un proceso por fases que permite eliminar las dependencias entre atributos que originan anomalías en la inserción, borrado y actualización de datos en una base de datos relacional.

Se realiza con el propósito de:

- Optimizar la estructura de la base de datos.
- Eliminar situaciones no deseables, tales como:
 - . Redundancia innecesaria de datos
 - . Anomalías de inserción, borrado y actualización
 - . Atributos no atómicos
 - . Dependencias funcionales parciales
 - . Dependencias transitivas

Sistemas de Base de Datos

NORMALIZACIÓN...

Atributos Atómicos

Un atributo es atómico si los valores de su dominio son indivisibles

En la relación:

PERSONAL(Codigo, AP, AM, Nombres, Direccion, FechaNacimiento)

Atributos Atómicos:

Codigo, AP, AM, Nombres

Atributos no Atómicos:

Dirección y FechaNacimiento

Sistemas de Base de Datos

NORMALIZACIÓN...

Dependencias funcionales

Un atributo Y es funcionalmente dependiente de otro atributo X, si a cada valor de X le corresponde un único valor de Y.

Este hecho, también se puede expresar como: que X determina o implica a Y, X es el implicante o determinante e Y el implicado.

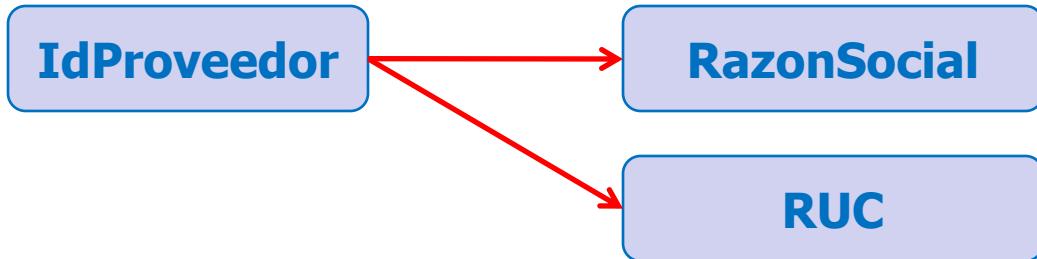
Gráficamente se representa como:



Por ejemplo, en la relación:

PROVEEDOR(IdProveedor, RazonSocial, RUC)

IdProveedor determina a RazonSocial y RUC



NORMALIZACIÓN...

Dependencias funcionales completas

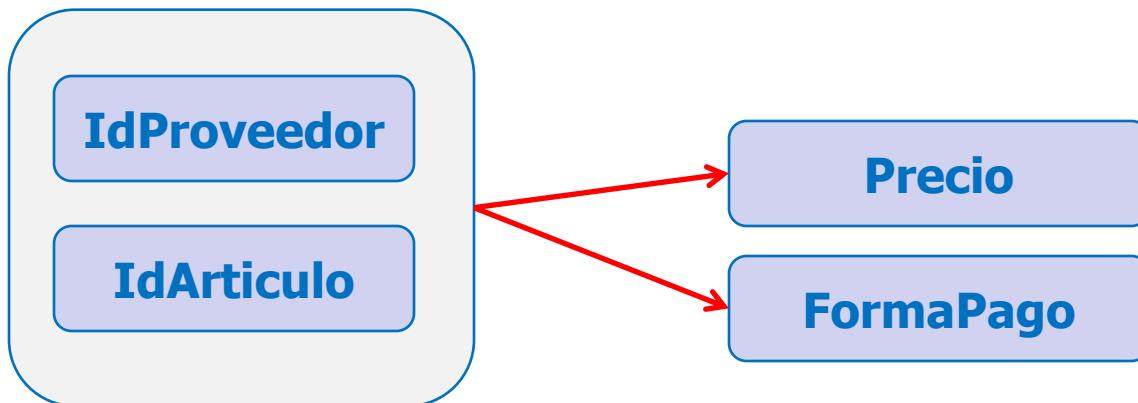
La dependencia funcional completa se da cuando un atributo Y tiene dependencia funcional de un grupo de atributos X, y no de una parte del grupo de atributos X.

El siguiente ejemplo ilustra este hecho:

En la relación:

PROVEEDOR_ARTICULO(IdProveedor, IdArticulo, Precio, FormaPago)

IdProveedor e IdArticulo (Ambos) determinan a Precio y FormaPago



Notar que **Precio** no es determinado sólo por **IdProveedor** o sólo por **IdArticulo**

NORMALIZACIÓN...

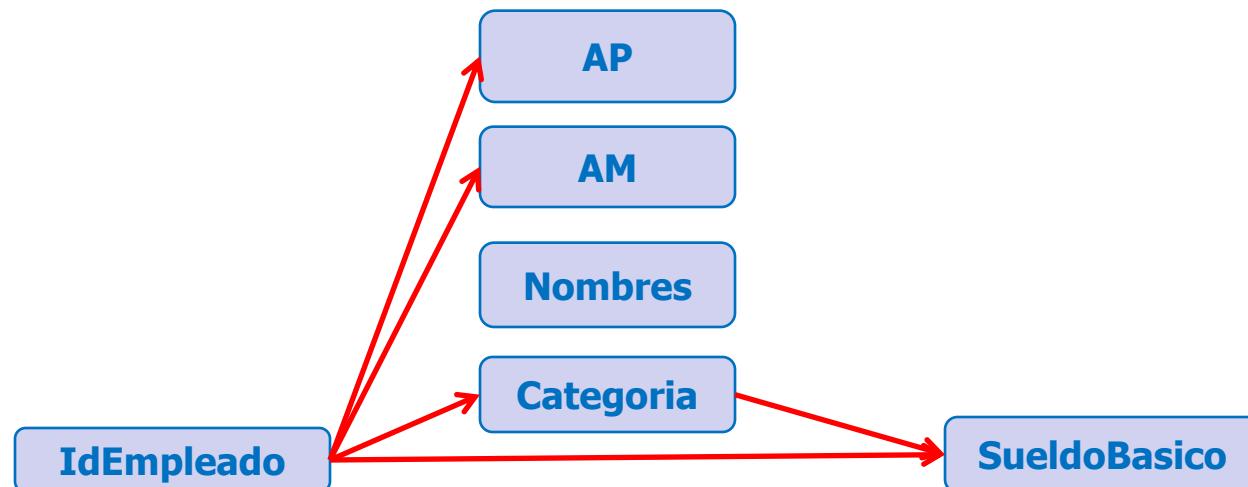
Dependencias funcionales transitivas

Si un atributo Z tiene dependencia funcional de otro atributo Y, y éste tiene dependencia funcional de otro atributo X, entonces el atributo Z tiene dependencia funcional transitiva del atributo X.

El siguiente ejemplo ilustra este hecho:

En la relación:

EMPLEADO(IdEmpleado, AP, AM, Nombres, SueldoBasico, Categoria)



SueldoBasico presenta dependencia funcional transitiva de IdEmpleado

NORMALIZACIÓN...

PRIMERA FORMA NORMAL (1FN)

Una relación está en PRIMERA FORMA NORMAL si todos sus atributos son atómicos

La relación:

PROVEEDOR(IdProveedor, RazonSocial, RUC)

Está en primera forma normal porque, todos sus atributos son atómicos

Mientras que la relación:

LIBRO (IdLibro, Titulo, Autor, Editorial)

No está en primera forma normal, porque un libro puede tener más de un autor, en consecuencia, el atributo autor no es atómico porque tiene múltiples valores.

Para que la relación cumpla con la primera forma normal se debe descomponer:

LIBRO (IdLibro, Titulo, Editorial)

LIBRO_AUTOR (IdLibro, Autor)

Ahora ambas relaciones están en primera forma normal.

NORMALIZACIÓN...

PRIMERA FORMA NORMAL (1FN)

También es deseable que una relación que esté en la Primera Forma Normal satisfaga lo siguiente:

- La tabla contiene una clave primaria.
- La clave primaria no contiene atributos nulos.
- No posee ciclos repetitivos.

NORMALIZACIÓN...

SEGUNDA FORMA NORMAL (2FN)

Una relación está en 2FN si está en 1FN y todos los atributos que no son parte de la clave tienen Dependencia funcional completa respecto de cada una de las claves, es decir, no hay dependencias parciales.

Consideremos que se desea almacenar la relación de los integrantes de proyectos de elaboración de software, para lo cual se tiene la siguiente relación:

INTEGRANTE(DNI, AP, AM, Nombres, Proyecto, NroHoras)

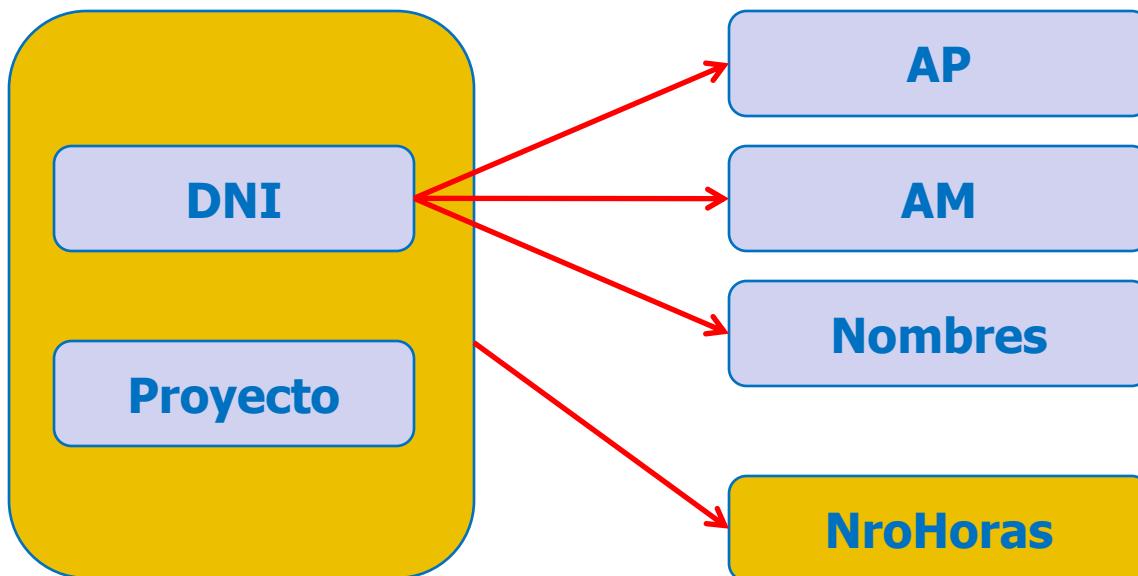
El atributo NroHoras hace referencia a las horas semanales que trabaja un integrante en un proyecto. Por ejemplo, el especialista en reportes puede trabajar alguna horas semanales en diferentes proyectos.

NORMALIZACIÓN...

SEGUNDA FORMA NORMAL (2FN)

INTEGRANTE(DNI, AP, AM, Nombres, Proyecto, NroHoras)

El gráfico correspondiente a las dependencias funcionales de la relación es:



El atributo **NroHoras** tiene dependencia funcional completa, mientras que los atributos: **AP, AM, Nombres** presentan dependencia funcional parcial. En consecuencia la relación no está en Segunda Forma Normal.

NORMALIZACIÓN...

SEGUNDA FORMA NORMAL (2FN)

La relación :

INTEGRANTE(DNI, AP, AM, Nombres, Proyecto, NroHoras)

Se debe descomponer en:

INTEGRANTE(DNI, AP, AM, Nombres)

INTEGRANTE_PROYECTO(DNI, Proyecto, NroHoras)

Sistemas de Base de Datos

NORMALIZACIÓN...

TERCERA FORMA NORMAL (3FN)

Una relación está en 3FN si está en 2FN y si no existe ninguna dependencia funcional transitiva.

Consideremos la siguiente relación:

PRESTAMO(NroPagare, Fecha, Cliente, RazonSocial, RUC, Importe, FechaVcto)

NORMALIZACIÓN...

TERCERA FORMA NORMAL (3FN)

PRESTAMO(NroPagare, Fecha, Cliente, RazonSocial, RUC, Importe, FechaVcto)

El gráfico correspondiente a las dependencias funcionales de la relación es:



Las dependencias transitivas hacen que la relación no esté en tercera forma normal.

NORMALIZACIÓN...

TERCERA FORMA NORMAL (3FN)

La relación :

PRESTAMO(NroPagare, Fecha, Cliente, RazonSocial, RUC, Importe, FechaVcto)

Se debe descomponer en:

PRESTAMO(NroPagare, Fecha, Cliente, Importe, FechaVcto)

CLIENTE(Cliente, RazonSocial, RUC)

Sistemas de Base de Datos

NORMALIZACIÓN...

FORMA NORMAL DE BOYCE-CODD(FNBC)

Una relación está en Formal Normal de Boyce/Codd (BCNF) si y sólo si cada determinante de la relación es una clave candidata.

Veamos la siguiente relación

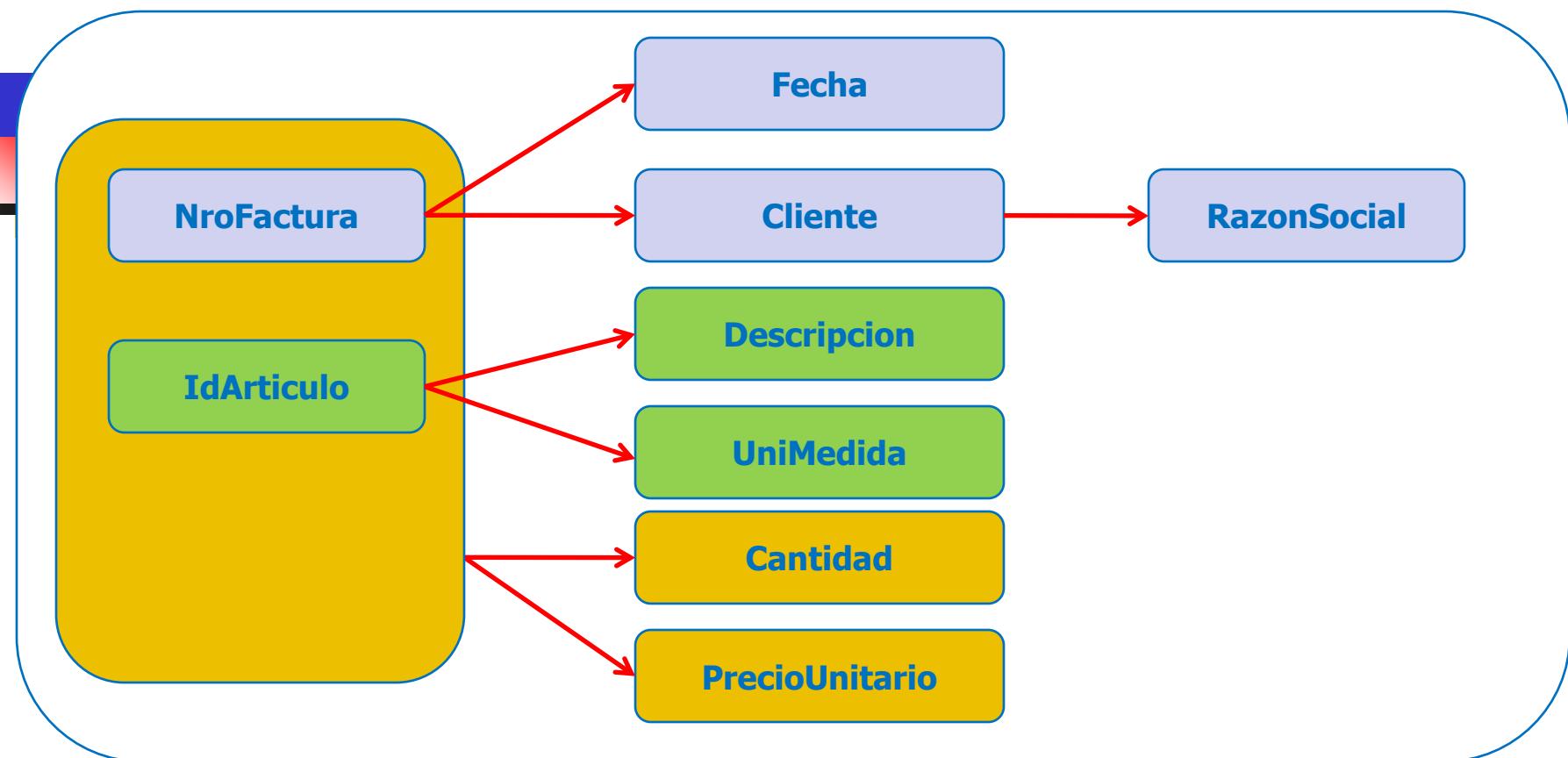
NroFactura	Fecha	Cliente	RazonSocial	IdArticulo	Descripcion	UniMedida	Cantidad	PrecioUnitario
124589	02/01/2010	4568	Distribuidora Omega	C01	Computador DeskTop	UNI	10	1200
124589	02/01/2010	4568	Distribuidora Omega	L01	Laptop	UNI	20	1850
124589	02/01/2010	4568	Distribuidora Omega	I01	Impresoras Laser	UNI	5	785
124589	02/01/2010	4568	Distribuidora Omega	D01	DVD	CJA	50	75
124590	03/02/2010	8750	Inka's Software	W01	Windows Vista	UNI	5	500
124590	03/02/2010	8750	Inka's Software	VS01	Visual Studio 2008	UNI	10	450
124591	10/02/2010	5531	Solución Total S.A.	C01	Computador DeskTop	UNI	3	1200
124591	10/02/2010	5531	Solución Total S.A.	VS01	Visual Studio 2008	UNI	2	480

NORMALIZACIÓN...

FORMA NORMAL DE BOYCE – CODD (FNBC)

R(NroFactura, Fecha, Cliente, RazonSocial, IdArticulo, Descripcion, UniMedida, Cantidad, PrecioUnitario)

Las dependencias funcionales son:



NORMALIZACIÓN...

FORMA NORMAL DE BOYCE – CODD (FNBC)

La relación :

R(NroFactura, Fecha, Cliente, RazonSocial, IdArticulo, Descripcion, UniMedida, Cantidad, PrecioUnitario)

Se debe descomponer entonces de modo que cada Determinante sea una clave candidata:

CLIENTE(Cliente, RazonSocial)

ARTICULO(IdArticulo, Descripcion, UniMedida)

FACTURA(NroFactura, Fecha, Cliente)

FACTURA_ARTICULO(NroFactura, IdArticulo, Cantidad, PrecioUnitario)

Estas relaciones están en la Forma Normal de Boyce – Codd, porque cada determinante es una clave candidata.

Sistemas de Base de Datos

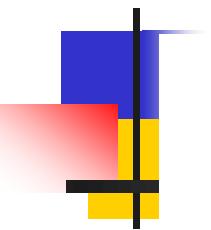
NORMALIZACIÓN...

EJERCICIOS

La universidad adquiere bienes, estos bienes pueden ser equipos o maquinarias, muebles y otros. Luego, estos bienes son asignados a las diferentes dependencias de la universidad y entregados a responsabilidad a los empleados de estas dependencias. La unidad de Patrimonio es la responsable de velar por estos bienes y solicita que se efectúe el diseño lógico de la base de datos, considerando la siguiente relación universal:

RU(CodDependencia, NombreDependencia, Ubicación, CodResponsable, NombreResponsable, DocIdentidadResponsable, CodBien, Descripción, TipoBien, NroSerie, Marca, Modelo, DocumentoAdquisición, FechaAdquisición, Precio, DocumentoAsignación, FechaAsignación, Observaciones).

NORMALIZARLO A LA FORMA NORMAL DE BOYCE – CODD (FNBC)



NORMALIZACIÓN

NORMALIZACIÓN... (Anomalías en el diseño)

Dada la siguiente relación:

Libro(Código, Título, Autor, Editorial, NombreEditorial)

Es está una relación diseñada apropiadamente?

Anomalías de inserción

- No es posible insertar los datos de un libro si no se conoce el nombre de la editorial.
- Se tiene que digitar redundantemente el valor del Nombre de la editorial, cada vez que se registre un libro de la misma editorial.

Anomalías de borrado

- Al borrar un libro y si éste es el único de una editorial, se pierde los datos de la editorial.
- Para borrar una editorial hay que borrar tantas tuplas como libros tenga.

Anomalías de actualización

- Si fuese necesario modificar el Nombre de una editorial, la actualización hay que realizarlo en tantas tuplas como libros tenga la editorial.
- Puede haber tuplas de libros de la misma editorial con distintos valores de nombre de editorial.

NORMALIZACIÓN

Es un proceso por fases que permite eliminar las dependencias entre atributos que originan anomalías en la inserción, borrado y actualización de datos en una base de datos relacional.

Se realiza con el propósito de:

- Optimizar la estructura de la base de datos.
- Eliminar situaciones no deseables, tales como:
 - . Redundancia innecesaria de datos
 - . Anomalías de inserción, borrado y actualización
 - . Atributos no atómicos
 - . Dependencias funcionales parciales
 - . Dependencias transitivas

Sistemas de Base de Datos

NORMALIZACIÓN...

Atributos Atómicos

Un atributo es atómico si los valores de su dominio son indivisibles

En la relación:

PERSONAL(Codigo, AP, AM, Nombres, Direccion, FechaNacimiento)

Atributos Atómicos:

Codigo, AP, AM, Nombres

Atributos no Atómicos:

Dirección y FechaNacimiento

Sistemas de Base de Datos

NORMALIZACIÓN...

Dependencias funcionales

Un atributo Y es funcionalmente dependiente de otro atributo X, si a cada valor de X le corresponde un único valor de Y.

Este hecho, también se puede expresar como: que X determina o implica a Y, X es el implicante o determinante e Y el implicado.

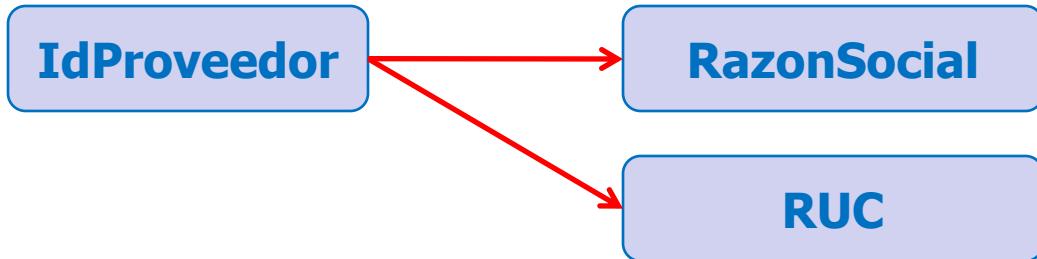
Gráficamente se representa como:



Por ejemplo, en la relación:

PROVEEDOR(IdProveedor, RazonSocial, RUC)

IdProveedor determina a RazonSocial y RUC



NORMALIZACIÓN...

Dependencias funcionales completas

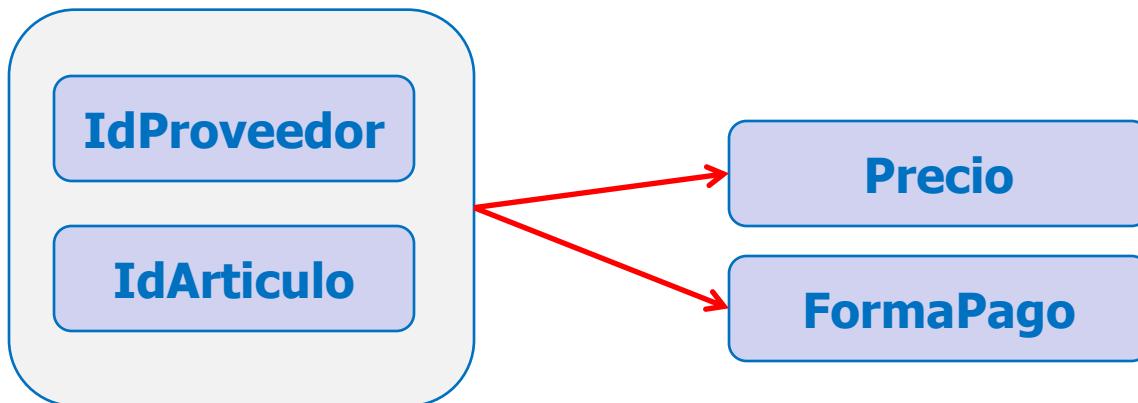
La dependencia funcional completa se da cuando un atributo Y tiene dependencia funcional de un grupo de atributos X, y no de una parte del grupo de atributos X.

El siguiente ejemplo ilustra este hecho:

En la relación:

PROVEEDOR_ARTICULO(IdProveedor, IdArticulo, Precio, FormaPago)

IdProveedor e IdArticulo (Ambos) determinan a Precio y FormaPago



Notar que **Precio** no es determinado sólo por **IdProveedor** o sólo por **IdArticulo**

NORMALIZACIÓN...

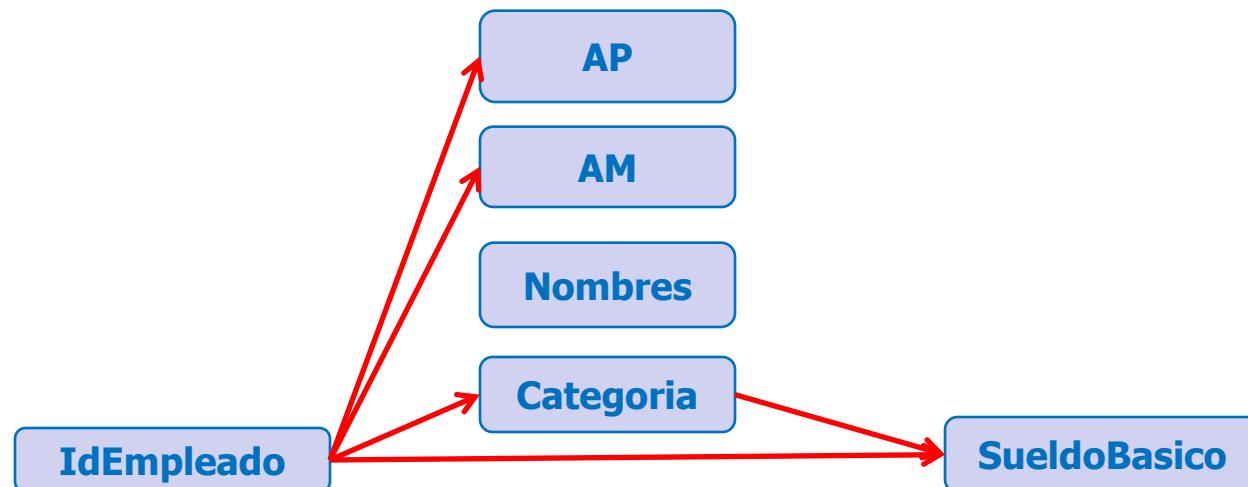
Dependencias funcionales transitivas

Si un atributo Z tiene dependencia funcional de otro atributo Y, y éste tiene dependencia funcional de otro atributo X, entonces el atributo Z tiene dependencia funcional transitiva del atributo X.

El siguiente ejemplo ilustra este hecho:

En la relación:

EMPLEADO(IdEmpleado, AP, AM, Nombres, SueldoBasico, Categoria)



SueldoBasico presenta dependencia funcional transitiva de IdEmpleado

NORMALIZACIÓN...

PRIMERA FORMA NORMAL (1FN)

Una relación está en PRIMERA FORMA NORMAL si todos sus atributos son atómicos

La relación:

PROVEEDOR(IdProveedor, RazonSocial, RUC)

Está en primera forma normal porque, todos sus atributos son atómicos

Mientras que la relación:

LIBRO (IdLibro, Titulo, Autor, Editorial)

No está en primera forma normal, porque un libro puede tener más de un autor, en consecuencia, el atributo autor no es atómico porque tiene múltiples valores.

Para que la relación cumpla con la primera forma normal se debe descomponer:

LIBRO (IdLibro, Titulo, Editorial)

LIBRO_AUTOR (IdLibro, Autor)

Ahora ambas relaciones están en primera forma normal.

NORMALIZACIÓN...

PRIMERA FORMA NORMAL (1FN)

También es deseable que una relación que esté en la Primera Forma Normal satisfaga lo siguiente:

- La tabla contiene una clave primaria.
- La clave primaria no contiene atributos nulos.
- No posee ciclos repetitivos.

NORMALIZACIÓN...

SEGUNDA FORMA NORMAL (2FN)

Una relación está en 2FN si está en 1FN y todos los atributos que no son parte de la clave tienen Dependencia funcional completa respecto de cada una de las claves, es decir, no hay dependencias parciales.

Consideremos que se desea almacenar la relación de los integrantes de proyectos de elaboración de software, para lo cual se tiene la siguiente relación:

INTEGRANTE(DNI, AP, AM, Nombres, Proyecto, NroHoras)

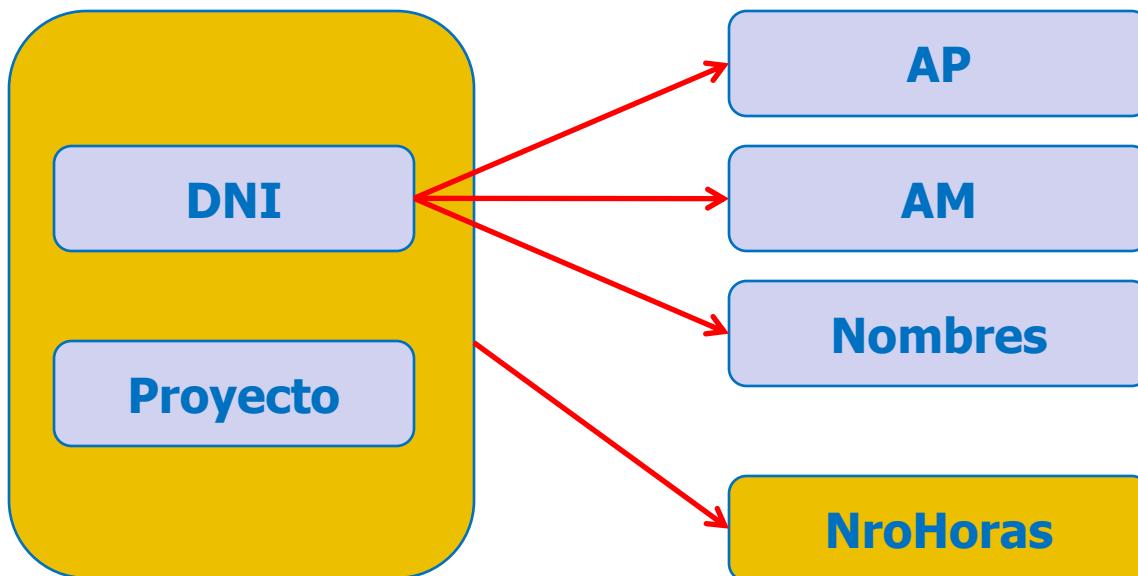
El atributo NroHoras hace referencia a las horas semanales que trabaja un integrante en un proyecto. Por ejemplo, el especialista en reportes puede trabajar alguna horas semanales en diferentes proyectos.

NORMALIZACIÓN...

SEGUNDA FORMA NORMAL (2FN)

INTEGRANTE(DNI, AP, AM, Nombres, Proyecto, NroHoras)

El gráfico correspondiente a las dependencias funcionales de la relación es:



El atributo **NroHoras** tiene dependencia funcional completa, mientras que los atributos: **AP, AM, Nombres** presentan dependencia funcional parcial. En consecuencia la relación no está en Segunda Forma Normal.

NORMALIZACIÓN...

SEGUNDA FORMA NORMAL (2FN)

La relación :

INTEGRANTE(DNI, AP, AM, Nombres, Proyecto, NroHoras)

Se debe descomponer en:

INTEGRANTE(DNI, AP, AM, Nombres)

INTEGRANTE_PROYECTO(DNI, Proyecto, NroHoras)

Sistemas de Base de Datos

NORMALIZACIÓN...

TERCERA FORMA NORMAL (3FN)

Una relación está en 3FN si está en 2FN y si no existe ninguna dependencia funcional transitiva.

Consideremos la siguiente relación:

PRESTAMO(NroPagare, Fecha, Cliente, RazonSocial, RUC, Importe, FechaVcto)

NORMALIZACIÓN...

TERCERA FORMA NORMAL (3FN)

PRESTAMO(NroPagare, Fecha, Cliente, RazonSocial, RUC, Importe, FechaVcto)

El gráfico correspondiente a las dependencias funcionales de la relación es:



Las dependencias transitivas hacen que la relación no esté en tercera forma normal.

NORMALIZACIÓN...

TERCERA FORMA NORMAL (3FN)

La relación :

PRESTAMO(NroPagare, Fecha, Cliente, RazonSocial, RUC, Importe, FechaVcto)

Se debe descomponer en:

PRESTAMO(NroPagare, Fecha, Cliente, Importe, FechaVcto)

CLIENTE(Cliente, RazonSocial, RUC)

Sistemas de Base de Datos

NORMALIZACIÓN...

FORMA NORMAL DE BOYCE-CODD(FNBC)

Una relación está en Formal Normal de Boyce/Codd (BCNF) si y sólo si cada determinante de la relación es una clave candidata.

Veamos la siguiente relación

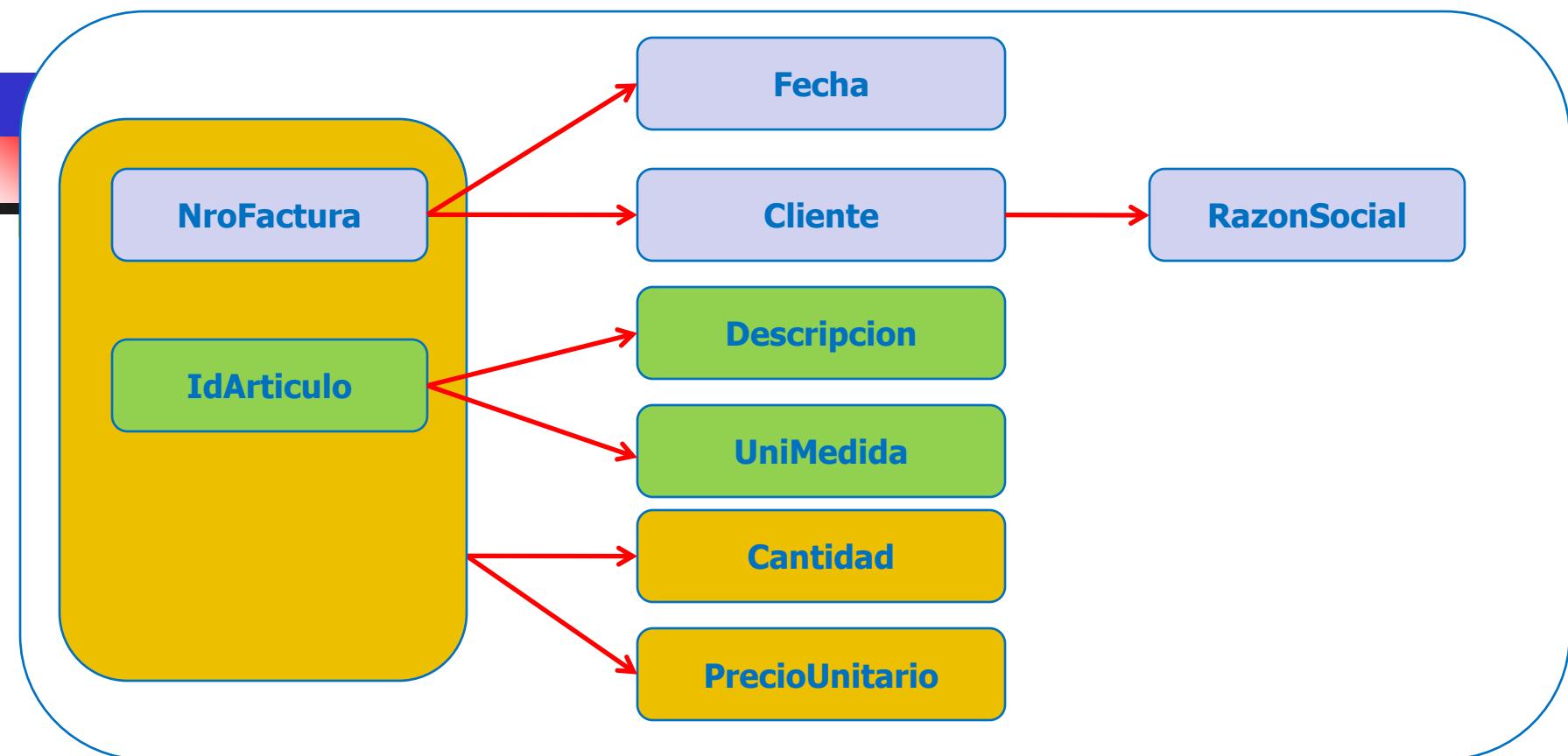
NroFactura	Fecha	Cliente	RazonSocial	IdArticulo	Descripcion	UniMedida	Cantidad	PrecioUnitario
124589	02/01/2010	4568	Distribuidora Omega	C01	Computador DeskTop	UNI	10	1200
124589	02/01/2010	4568	Distribuidora Omega	L01	Laptop	UNI	20	1850
124589	02/01/2010	4568	Distribuidora Omega	I01	Impresoras Laser	UNI	5	785
124589	02/01/2010	4568	Distribuidora Omega	D01	DVD	CJA	50	75
124590	03/02/2010	8750	Inka's Software	W01	Windows Vista	UNI	5	500
124590	03/02/2010	8750	Inka's Software	VS01	Visual Studio 2008	UNI	10	450
124591	10/02/2010	5531	Solución Total S.A.	C01	Computador DeskTop	UNI	3	1200
124591	10/02/2010	5531	Solución Total S.A.	VS01	Visual Studio 2008	UNI	2	480

NORMALIZACIÓN...

FORMA NORMAL DE BOYCE – CODD (FNBC)

R(NroFactura, Fecha, Cliente, RazonSocial, IdArticulo, Descripcion, UniMedida, Cantidad, PrecioUnitario)

Las dependencias funcionales son:



NORMALIZACIÓN...

FORMA NORMAL DE BOYCE – CODD (FNBC)

La relación :

R(NroFactura, Fecha, Cliente, RazonSocial, IdArticulo, Descripcion, UniMedida, Cantidad, PrecioUnitario)

Se debe descomponer entonces de modo que cada Determinante sea una clave candidata:

CLIENTE(Cliente, RazonSocial)

ARTICULO(IdArticulo, Descripcion, UniMedida)

FACTURA(NroFactura, Fecha, Cliente)

FACTURA_ARTICULO(NroFactura, IdArticulo, Cantidad, PrecioUnitario)

Estas relaciones están en la Forma Normal de Boyce – Codd, porque cada determinante es una clave candidata.

Sistemas de Base de Datos

NORMALIZACIÓN...

EJERCICIOS

La universidad adquiere bienes, estos bienes pueden ser equipos o maquinarias, muebles y otros. Luego, estos bienes son asignados a las diferentes dependencias de la universidad y entregados a responsabilidad a los empleados de estas dependencias. La unidad de Patrimonio es la responsable de velar por estos bienes y solicita que se efectúe el diseño lógico de la base de datos, considerando la siguiente relación universal:

RU(CodDependencia, NombreDependencia, Ubicación, CodResponsable, NombreResponsable, DocIdentidadResponsable, CodBien, Descripción, TipoBien, NroSerie, Marca, Modelo, DocumentoAdquisición, FechaAdquisición, Precio, DocumentoAsignación, FechaAsignación, Observaciones).

NORMALIZARLO A LA FORMA NORMAL DE BOYCE – CODD (FNBC)

DISEÑO DE BASES DE DATOS – DISEÑO CONCEPTUAL

EJERCICIO 1: ALQUILER DE VEHÍCULOS

La empresa Inkas'car se dedica al alquiler de vehículos, para lo cual, cuenta con: Varios puntos de alquiler (agencias), varios garajes y una flota de vehículos. Los clientes de la empresa, pueden realizar varios procesos de alquiler a la vez; y en cada proceso de alquiler, alquilar varios vehículos. La empresa exige que cada cliente debe contar ser avalado por otro cliente de la empresa. En el proceso de alquiler se registra las fechas de inicio y fin del alquiler de cada vehículo, así como los galones de gasolina en el momento de realizar el alquiler. Los clientes recogen los vehículos de sus respectivos garajes. Cada vehículo está asignado a un solo garaje. De cada vehículo se debe registrar: la placa, la marca, el modelo, el color, el número de asientos y los accesorios con los que está equipado (GPS, radio, etc.). Para cada accesorio se debe registrar el identificador, la descripción y la marca.

Diseño Conceptual

Entidades



Agencia



Garaje



Vehículo



Cliente



Doc.
Alquiler



Accesorio

Diseño Conceptual

Entidades



Agencia



Garaje



Vehículo



Cliente



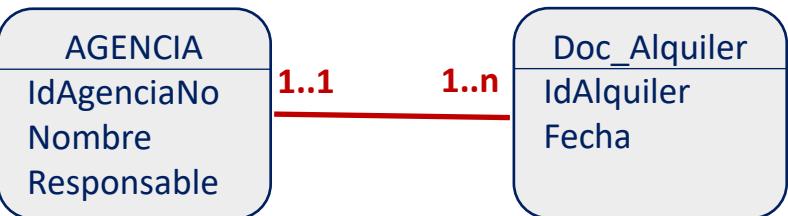
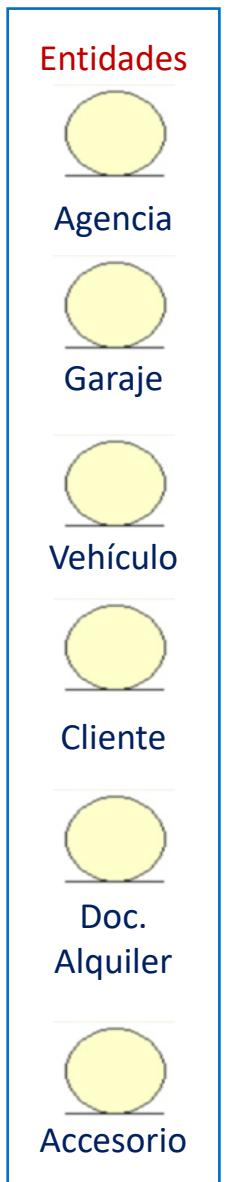
Doc.
Alquiler



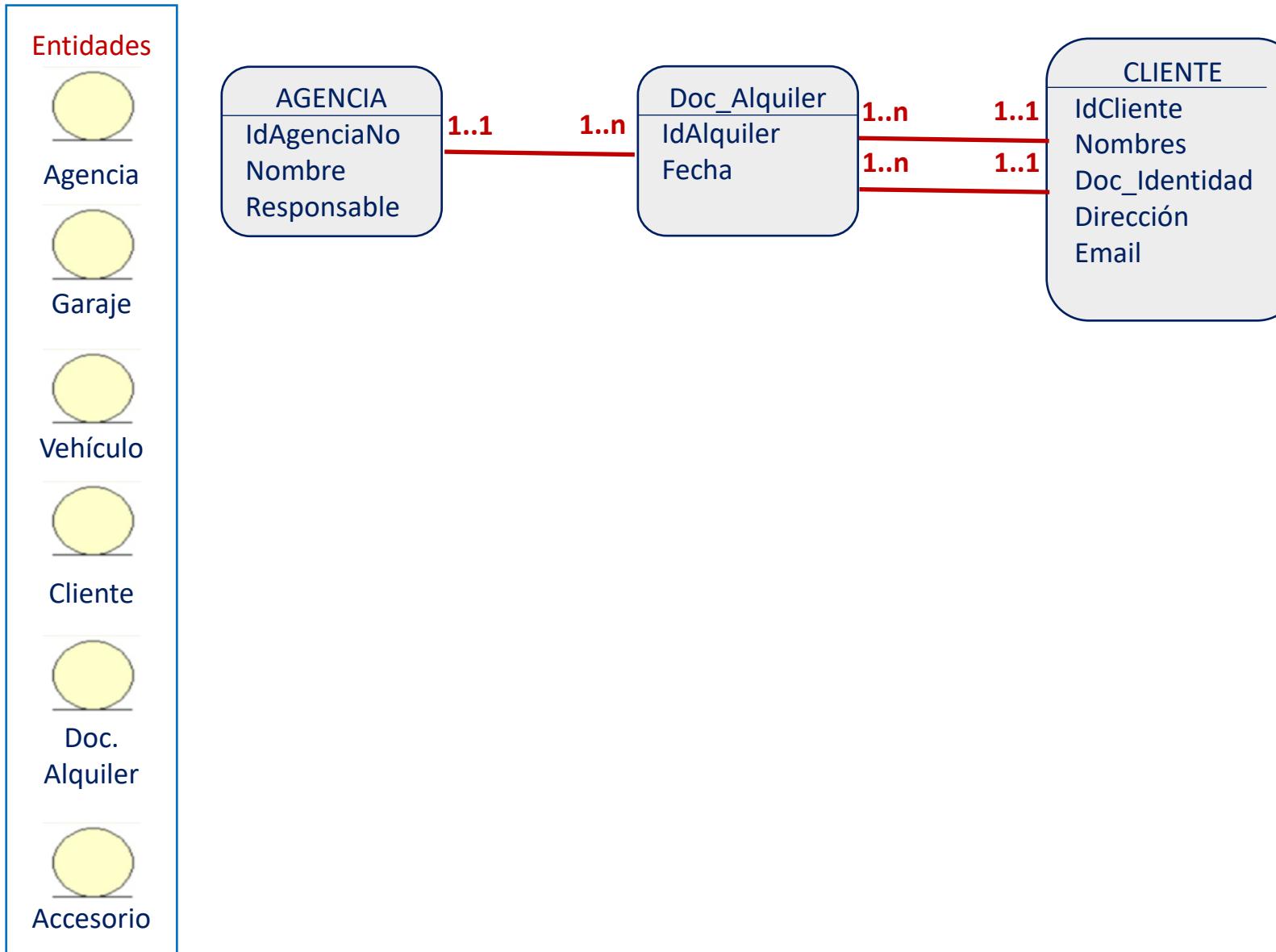
Accesorio



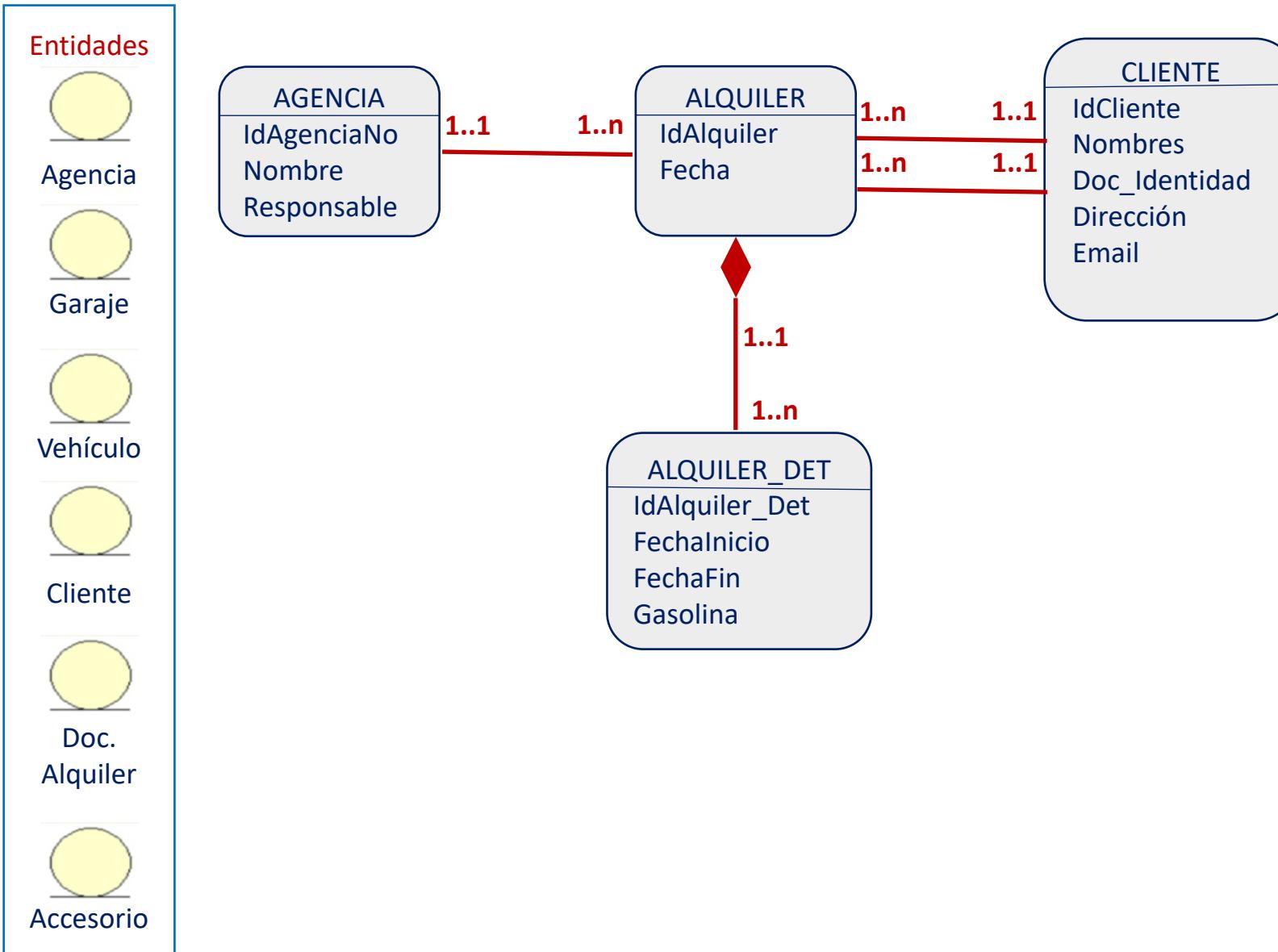
Diseño Conceptual



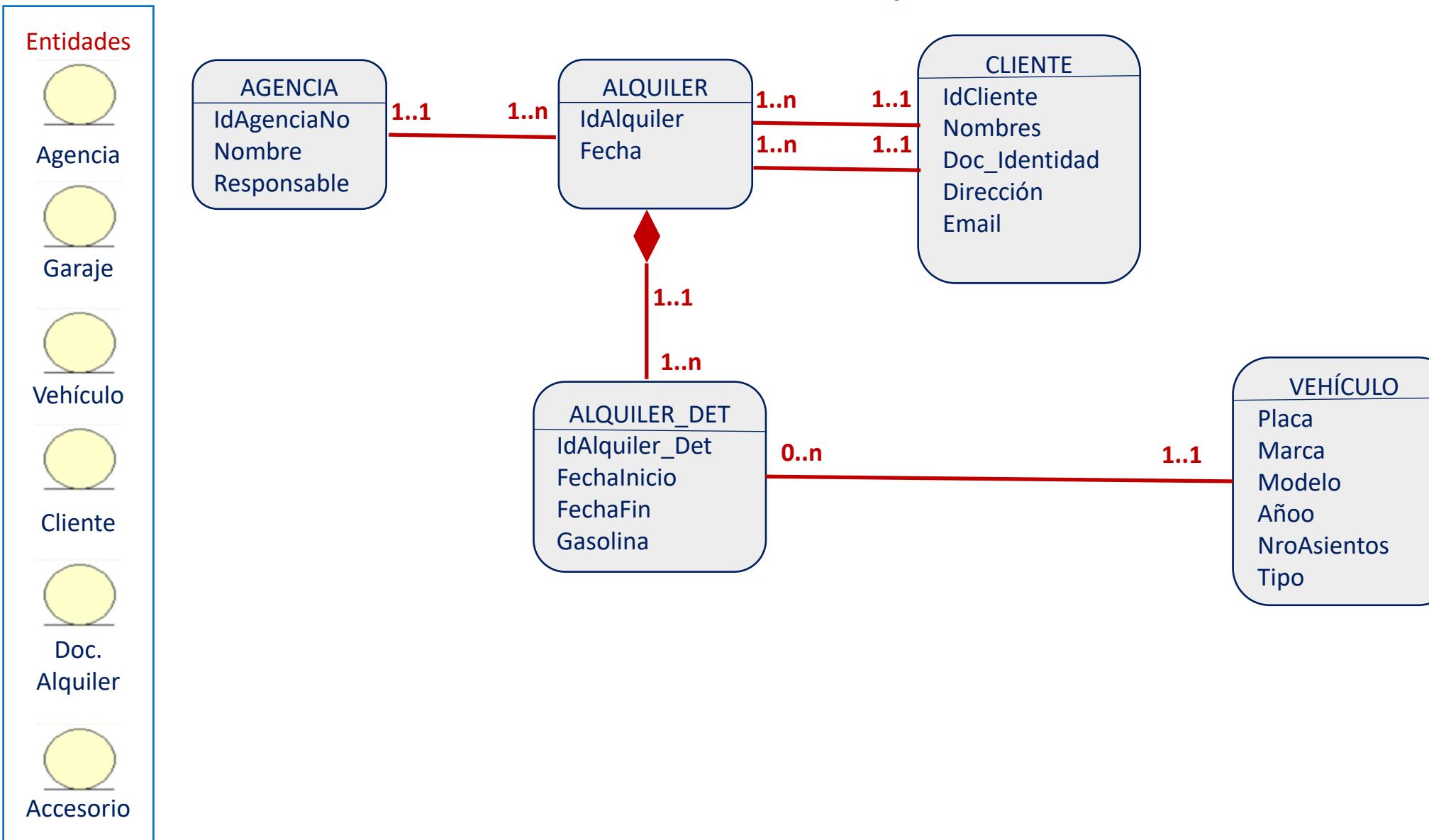
Diseño Conceptual



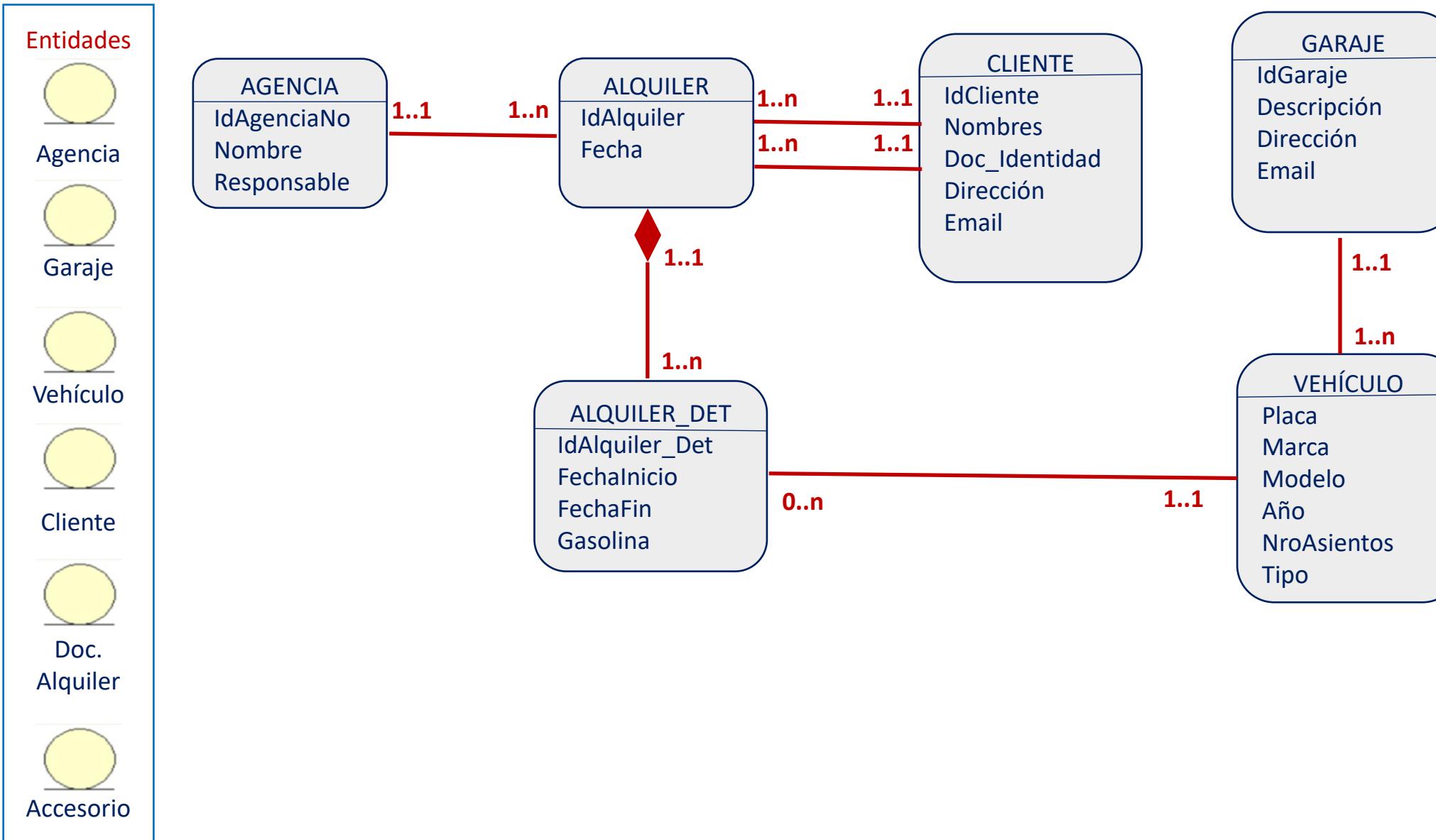
Diseño Conceptual



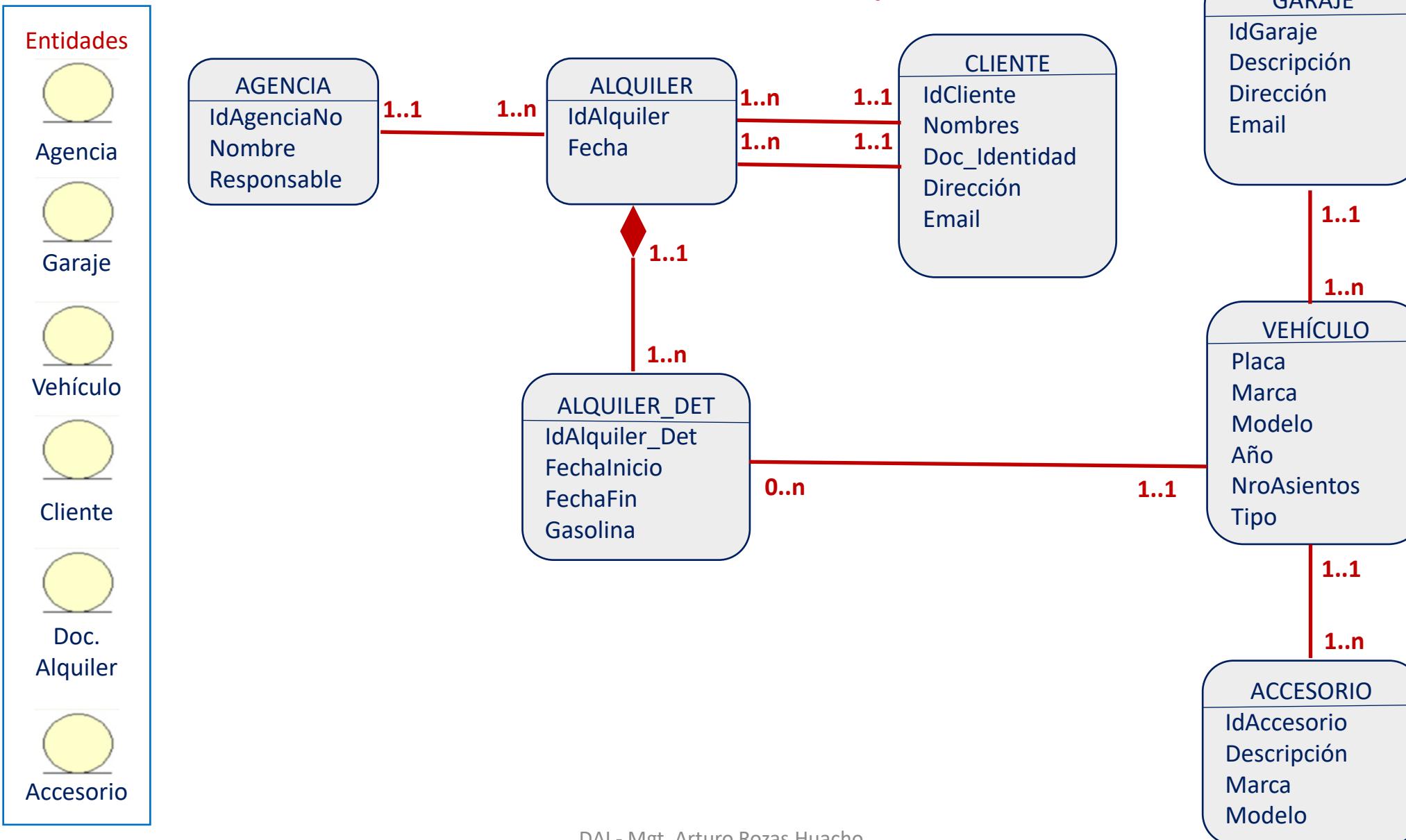
Diseño Conceptual



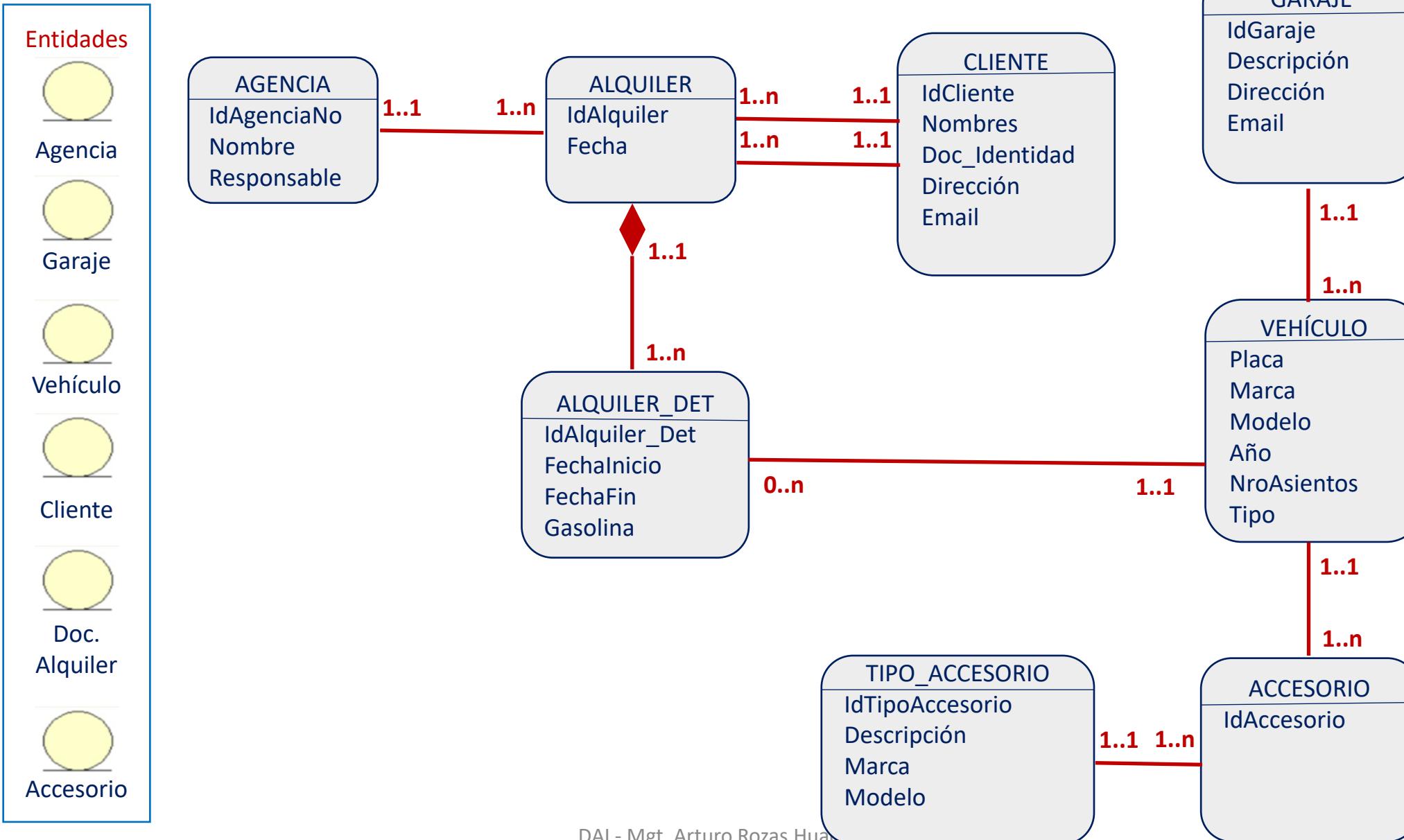
Diseño Conceptual



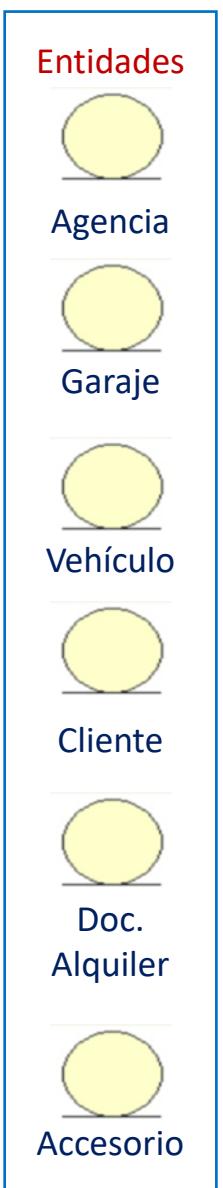
Diseño Conceptual



Diseño Conceptual



Diseño Conceptual



AGENCIA
IdAgenciaNo
Nombre
Responsable

1..1

1..n

ALQUILER
IdAlquiler
Fecha

1..n

1..n

CLIENTE
IdCliente
Nombres
Doc_Identidad
Dirección
Email

1..1

1..1

GARAJE

IdGaraje
Descripción
Dirección
Email

1..1

1..n

VEHÍCULO

Placa
Marca
Modelo
Año
NroAsientos
Tipo

1..1

1..1

1..n

TIPO_ACCESORIO

IdTipoAccesorio
Descripción
Marca
Modelo

1..1 1..n

ACCESORIO

IdAccesorio

1..n

Diseño Lógico

AGENCIA(IdAgenciaNo, Nombre, Responsable)

CLIENTE(IdCliente, Nombres, Doc_Identidad, Dirección, Email)

GARAJE(IdGaraje, Descripción, Dirección, Email)

VEHÍCULO(Placa, Marca, Modelo, Año, NroAsientos, Tipo, **IdGaraje**)

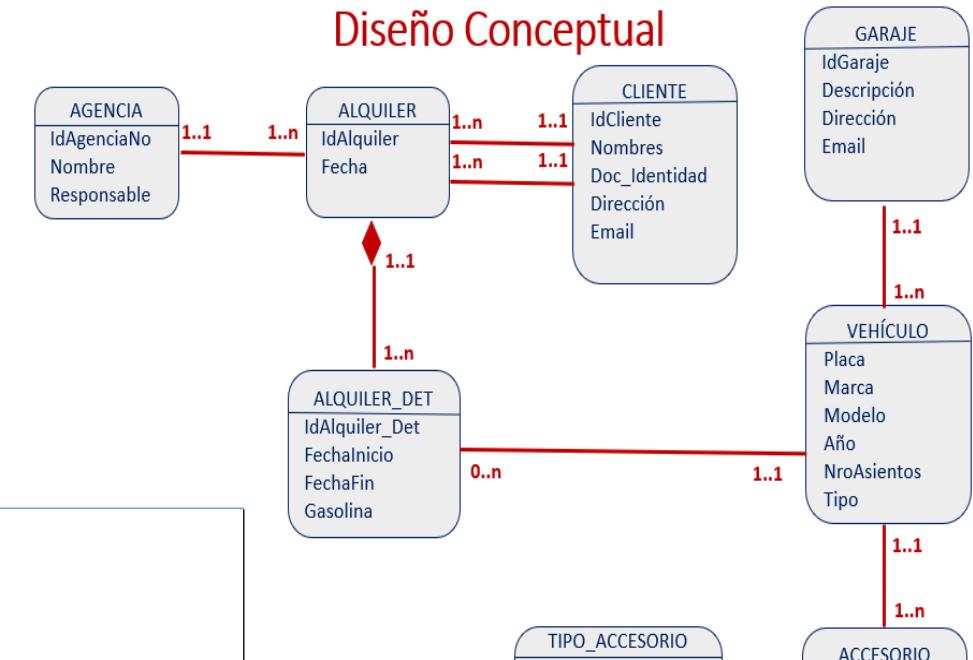
TIPO_ACCESORIO(IdTipoAccesorio, Descripción, Marca, Modelo)

ACCESORIO(IdAccesorio, Placa, **IdTipoAccesorio**)

ALQUILER(IdAlquiler, Fecha, **IdAGenciaNo**, **IdCliente**, **IdAval**)

ALQUILER_DET(IdAlquiler, IdAlquiler_Det, FechInicio, FechaFin, Gasolina, **IdVehículo**)

Diseño Conceptual



4.1.7.- Tablas Cruzadas

Para entender el concepto de tablas cruzadas, veamos el siguiente ejemplo:

Supongamos que se desea Analizar las ventas por tipo de prenda y color.

En un proceso previo se obtuvo la siguiente tabla.

CrossTable1

Prenda	Color	Cantidad
Camisa	Blanco	5
Camisa	Oscuro	13
Camisa	Pastel	12
Chompa	Blanco	6
Chompa	Claro	9
Chompa	Oscuro	11
Chompa	Pastel	2
Pantalón	Claro	1
Pantalón	Oscuro	21
Pantalón	Pastel	7
Polo	Blanco	3
Polo	Claro	8
Polo	Oscuro	1

4.1.7.- Tablas Cruzadas

CrossTable1

Prenda	Color	Cantidad
Camisa	Blanco	5
Camisa	Oscuro	13
Camisa	Pastel	12
Chompa	Blanco	6
Chompa	Claro	9
Chompa	Oscuro	11
Chompa	Pastel	2
Pantalon	Claro	1
Pantalon	Oscuro	21
Pantalon	Pastel	7
Polo	Blanco	3
Polo	Claro	8
Polo	Oscuro	1

Atributos de Medición

Atributos de Dimensión

4.1.7.- Tablas Cruzadas

CrossTable1

Prenda	Color	Cantidad
Camisa	Blanco	5
Camisa	Oscuro	13
Camisa	Pastel	12
Chompa	Blanco	6
Chompa	Claro	9
Chompa	Oscuro	11
Chompa	Pastel	2
Pantalon	Claro	1
Pantalon	Oscuro	21
Pantalon	Pastel	7
Polo	Blanco	3
Polo	Claro	8
Polo	Oscuro	1

Prenda	Blanco	Claro	Oscuro	Pastel
Camisa	5	0	13	12
Chompa	6	9	11	2
Pantalon	0	1	21	7
Polo	3	8	1	0



Los valores del atributo de la dimensión Color se utilizan como columnas en la nueva tabla.

Este tipo de resúmenes es el primer nivel de análisis de la información y es una necesidad frecuente en el apoyo a la toma de decisiones.

4.1.7.- Tablas Cruzadas...

Paso 1.- Supongamos que ya se tiene la siguiente tabla:

CrossTable1 (Prenda, Color, Cantidad)

A partir de ésta, generar una tabla temporal con las siguientes Columnas:

Resultado(Prenda, Blanco, Claro, Oscuro)

```
select Prenda,  
       "Blanco" = case when Color = 'Blanco' then Cantidad else 0 end,  
       "Claro"  = case when Color = 'Claro'  then Cantidad else 0 end,  
       "Oscuro" = case when Color = 'Oscuro' then Cantidad else 0 end  
  into #Prenda  
  from CrossTable1
```

4.1.7.- Tablas Cruzadas...

```
select Prenda,
       "Blanco" = case when Color = 'Blanco' then Cantidad else 0 end,
       "Claro"   = case when Color = 'Claro'   then Cantidad else 0 end,
       "Oscuro"  = case when Color = 'Oscuro'  then Cantidad else 0 end
  into #Prenda
 from CrossTable1
```

CrossTable1

Prenda	Color	Cantidad
Camisa	Blanco	5
Camisa	Oscuro	13
Camisa	Pastel	12
Chompa	Blanco	6
Chompa	Claro	9
Chompa	Oscuro	11
Chompa	Pastel	2
Pantalon	Claro	1
Pantalon	Oscuro	21
Pantalon	Pastel	7
Polo	Blanco	3
Polo	Claro	8
Polo	Oscuro	1

#Prenda

Prenda	Blanco	Claro	Oscuro
Camisa	5	0	0
Camisa	0	0	13
Camisa	0	0	0
Chompa	6	0	0
Chompa	0	9	0
Chompa	0	0	11
Chompa	0	0	0
Pantalon	0	1	0
Pantalon	0	0	21
Pantalon	0	0	0
Polo	3	0	0
Polo	0	8	0
Polo	0	0	1

4.1.7.- Tablas Cruzadas...

Paso 2.- Totalizar cada columna, considerando la prenda como criterio de grupo.

```
select Prenda,  
       "Blanco" = sum(Blanco),  
       "Claro"  = sum(Claro),  
       "Oscuro" = sum(Oscuro)  
  from #Prenda  
 group by Prenda
```

#Prenda

Prenda	Blanco	Claro	Oscuro
Camisa	5	0	0
Camisa	0	0	13
Camisa	0	0	0
Chompa	6	0	0
Chompa	0	9	0
Chompa	0	0	11
Chompa	0	0	0
Pantalon	0	1	0
Pantalon	0	0	21
Pantalon	0	0	0
Polo	3	0	0
Polo	0	8	0
Polo	0	0	1

Resultado

Prenda	Blanco	Claro	Oscuro
Camisa	5	0	13
Chompa	6	9	11
Pantalon	0	1	21
Polo	3	8	1



4.1.7.- Tablas Cruzadas...

Algoritmo genérico en un solo paso.

```
select Prenda,  
      "Blanco" = sum(case when Color = 'Blanco' then Cantidad else 0 end),  
      "Claro"   = sum(case when Color = 'Claro'   then Cantidad else 0 end),  
      "Oscuro"  = sum(case when Color = 'Oscuro'  then Cantidad else 0 end)  
from CrossTable1  
group by Prenda
```

CrossTable1

Prenda	Color	Cantidad
Camisa	Blanco	5
Camisa	Oscuro	13
Camisa	Pastel	12
Chompa	Blanco	6
Chompa	Claro	9
Chompa	Oscuro	11
Chompa	Pastel	2
Pantalon	Claro	1
Pantalon	Oscuro	21
Pantalon	Pastel	7
Polo	Blanco	3
Polo	Claro	8
Polo	Oscuro	1

Resultado

Prenda	Blanco	Claro	Oscuro
Camisa	5	0	13
Chompa	6	9	11
Pantalon	0	1	21
Polo	3	8	1

4.1.7.- Tablas Cruzadas Dinámicas...

En este tipo de tablas cruzadas no se conoce de antemano el número de columnas. En consecuencia no se puede aplicar el algoritmo anterior.

CrossTable1

Prenda	Color	Cantidad
Camisa	Blanco	5
Camisa	Oscuro	13
Camisa	Pastel	12
Chompa	Blanco	6
Chompa	Claro	9
Chompa	Oscuro	11
Chompa	Pastel	2
Pantalon	Claro	1
Pantalon	Oscuro	21
Pantalon	Pastel	7
Polo	Blanco	3
Polo	Claro	8
Polo	Oscuro	1

Por ejemplo, de la anterior tabla se desea analizar las unidades Vendidas de todos los tipos de Prenda y de todos los colores.

Prenda	Blanco	Claro	Oscuro	Pastel
Camisa	5	0	13	12
Chompa	6	9	11	2
Pantalon	0	1	21	7
Polo	3	8	1	0

4.1.7.- Tablas Cruzadas Dinámicas...

```
-- Generar tabla temporal con solo Color
select distinct cast(Color as varchar(2000)) Color
into #Color
from CrossTable1
order by Color;

-- Concatenar los Colores en la variable @Colores
declare @Colores varchar(2000);
set @Colores = '';
update #Color
    set @Colores = Color = @Colores+', '+Color;
-- Quitar primera coma
set @Colores = SubString(@Colores,2,2000);

-- Construir sentencia SQL que permita generar el Cross Table
declare @TextoSQL varchar(2000);
declare @Posicion int;
declare @Color varchar(12);
```

4.1.7.- Tablas Cruzadas Dinámicas...

```
set @TextoSQL = 'SELECT Prenda '
while @Colores <> ''
begin
    -- Extraer Color
    set @Posicion = CharIndex(',',,@Colores);
    if @Posicion > 0
        begin
            set @Color = SubString(@Colores,1,@Posicion-1);
            set @Colores = SubString(@Colores,@Posicion+1,2000);
        end
    else
        begin
            set @Color = @Colores;
            set @Colores = '';
        end;
    -- Crear sentencia SQL para Color
    set @TextoSQL = @TextoSQL + ',''''+@Color+'''=SUM(case when Color =
                    ''''+@Color+''' then Cantidad else '0' end)';
end; -- while
```

4.1.7.- Tablas Cruzadas Dinámicas...

```
set @TextoSQL = @TextoSQL + ' FROM CrossTable1 GROUP BY Prenda';  
-- Ejecutar SQL  
exec(@TextoSQL)
```

El contenido de la variable @TextoSQL es:

```
select Prenda,  
    "Blanco" = sum(case when Color = 'Blanco' then Cantidad else 0 end),  
    "Claro"   = sum(case when Color = 'Claro'   then Cantidad else 0 end),  
    "Oscuro"  = sum(case when Color = 'Oscuro'  then Cantidad else 0 end)  
    "Pastel"  = sum(case when Color = 'Pastel'  then Cantidad else 0 end)  
from CrossTable1  
group by Prenda
```

4.3- Cursos

Las operaciones en una base de datos relacional se realizan sobre conjuntos de tuplas. La sentencia SELECT actúa sobre tablas y da como resultado un conjunto de tuplas (otra tabla).

En muchas aplicaciones no siempre es posible trabajar adecuadamente sobre conjuntos de tuplas, sino se necesitan de mecanismos para trabajar con una tupla a la vez.

Los CURSORES son una extensión del SQL que permiten procesar las tuplas de una en una.

Nota.- Se recomienda que sean utilizados como último recurso para la generación de consultas, en vista que consumen recursos y tiempo en el servidor.

4.3- Cursores...(Sintaxis)

-- Declarar el cursor

```
DECLARE nombre-cursor CURSOR FOR (sentencia SELECT)
```

-- Abrir el cursor

```
OPEN nombre-cursor
```

-- Recuperar la primera tupla

```
FECTH NEXT FROM nombre_cursor INTO @Variable1, @Variable2, ...
```

-- Estructura repetitiva para procesar cada tupla

```
WHILE @@Fetch_Status = 0
```

```
begin
```

instrucciones que procesen cada Tupla

-- Recuperar siguiente tupla

```
FECTH NEXT FROM nombre_cursor INTO @Variable1, @Variable2, ...
```

```
end; -- while
```

-- Cerrar y liberar la memoria utilizada por el cursor

```
CLOSE nombre_cursor
```

```
DEALLOCATE nombre_cursor
```

4.3- Cursores...(Ejemplo)

Dada la siguiente base de datos:

Carrera_Profesional(Cod_CP, Nombre_CP)

Alumno(Cod_Alumno, Paterno, Materno, Nombres, Cod_CP)

Docente(Cod_Docente, Paterno, Materno, Nombres, Categoria,Regimen)

Asignatura(Cod_Asignatura, Cod_CP, Nombre_Asignatura,Categoría,Creditos)

Catalogo(Semestre, Cod_Asignatura, Cod_CP, Grupo, Cod_Docente)

Matricula(Semestre, Cod_Asignatura, Cod_CP, Grupo, Cod_Alumno, Nota)

4.3- Cursores...(Ejemplo)

Determinar la relación de asignaturas aprobadas por el alumno de código '980115', con su respectivo creditaje y el acumulado de los créditos.

R(Cod_Asignatura, Creditos, Cred_Acum)

4.3- Cursores...(Ejemplo 1, Variante 1)

```
-- Crear tabla de resultados
create table #CreditosAcumulados
( Cod_Asignatura  varchar(5) ,
  Creditos        int,
  Acum_Creditos   int);
-- Declarar el cursor
DECLARE cu_CreditosAcumulados CURSOR
  FOR SELECT M.Cod_Asignatura, A.Creditos
    FROM Matricula M inner join Asignatura A
    ON (M.Cod_Asignatura = A.Cod_Asignatura) and
       (M.Cod_CP = A.Cod_CP)
    WHERE (M.Cod_Alumno = '980115') AND
          (Nota in ('11','12','13','14','15',
                    '16','17','18','19','20'))
-- Declarar variables para utilizar en el cursor
DECLARE  @Cod_Asignatura varchar(5),
         @Creditos int,
         @Acum_Creditos int
-- Inicializar la variable @Acum_Creditos
SET @Acum_Creditos = 0
```

4.3- Cursores...(Ejemplo 1, Variante 1)

```
-- Abrir el cursor
OPEN cu_CreditosAcumulados
-- Recuperar el primer registro en las variables
FETCH NEXT FROM cu_CreditosAcumulados
    INTO @Cod_Asignatura, @Creditos
-- Procesar cada registro
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Acumular creditaje
    set @Acum_Creditos = @Acum_Creditos + @Creditos;
    -- Insertar en tabla de resultados
    INSERT INTO #CreditosAcumulados
        values(@Cod_Asignatura, @Creditos, @Acum_Creditos)
    -- Recuperar siguiente registro
    FETCH NEXT FROM cu_CreditosAcumulados
        INTO @Cod_Asignatura, @Creditos
END
-- Cerrar el cursor
CLOSE cu_CreditosAcumulados
DEALLOCATE cu_CreditosAcumulados
-- Mostrar en la consulta los datos solicitados
SELECT * FROM #CreditosAcumulados
```

4.3- Cursores...(Ejemplo 1, Variante 2)

```
-- Crear tabla de resultados
declare @CreditosAcumulados table( Cod_Asignatura      varchar(5) ,
                                      Creditos            int,
                                      Acum_Creditos       int);

-- Declarar el cursor
DECLARE cu_CreditosAcumulados CURSOR
    FOR SELECT M.Cod_Asignatura, A.Creditos
        FROM Matricula M inner join Asignatura A
        ON (M.Cod_Asignatura = A.Cod_Asignatura) and
           (M.Cod_CP = A.Cod_CP)
        WHERE (M.Cod_Alumno = '980115') AND
              (Nota in ('11','12','13','14','15',
                        '16','17','18','19','20'))
-- Declarar variables para utilizar en el cursor
DECLARE @Cod_Asignatura varchar(5),
        @Creditos int,
        @Acum_Creditos int
-- Inicializar la variable @Acum_Creditos
SET @Acum_Creditos = 0
```

4.3- Cursores...(Ejemplo 1, Variante 2)

```
-- Abrir el cursor
OPEN cu_CreditosAcumulados
-- Recuperar los valores de la primera tupla
FETCH NEXT FROM cu_CreditosAcumulados INTO @Cod_Asignatura,@Creditos
-- Procesar cada tupla
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Acumular creditaje
    set @Acum_Creditos = @Acum_Creditos + @Creditos;
    -- Insertar en tabla de resultados
    INSERT INTO @CreditosAcumulados
        values (@Cod_Asignatura, @Creditos, @Acum_Creditos)
    -- Recuperar valores de la siguiente tupla
    FETCH NEXT FROM cu_CreditosAcumulados
        INTO @Cod_Asignatura, @Creditos
END
-- Cerrar el cursor
CLOSE cu_CreditosAcumulados
DEALLOCATE cu_CreditosAcumulados
-- Mostrar en la consulta los datos solicitados
SELECT * FROM @CreditosAcumulados
```

4.3- Cursores...(Ejemplo 2)

Asumiendo que en procesos previos se obtuvo la siguiente Tabla:

#Alumno(Cod_Alumno, Nombres, Cod_CP, Promedio)

Determinar los dos mejores estudiantes de cada carrera profesional:

4.3- Cursores...(Ejemplo 2)

```
-- Crear tabla de resultados
declare @Mejores table( Nro int,
                        Cod_Alumno varchar(6),
                        Nombres      varchar(40),
                        Cod_CP       varchar(2),
                        Promedio     Numeric(15,2));

-- Declarar el cursor
DECLARE cu_MejoresCURSOR
    FOR SELECT Cod_Alumno, Nombres, Cod_CP, Promedio
        FROM #Alumno
        SORT BY Cod_CP, Promedio DESC

-- Declarar variables para utilizar en el cursor
DECLARE @Nro          int,
        @Cod_Alumno   varchar(6),
        @Nombres      varchar(40),
        @Cod_CP       varchar(2),
        @Promedio     Numeric(15,2),
        @Cod_CP_Aux  varchar(2)
```

Sistemas de Base de Datos

4.3- Cursores...(Ejemplo 2)

```
-- Inicializar la variable @Acum_Creditos
SET @Nro = 0; SET @Cod_CP_Aux = '';
-- Abrir el cursor
OPEN cu_Mejores
-- Recuperar los valores de la primera tupla
FETCH NEXT FROM cu_Mejores
    INTO @Cod_Alumno, @Nombres, @Cod_CP, @Promedio
-- Procesar cada tupla
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Incrementar o Inicializar contador
    IF (@Cod_CP_AUX = @Cod_CP)
        set @Nro = @Nro + 1;
    ELSE
        set @Nro = 1;
    -- Insertar en tabla de resultados
    IF (@Nro <= 2)
        INSERT INTO @Mejores
            values(@Nro, @Cod_Alumno, @Nombres, @Cod_CP, @Promedio);
    set @Cod_CP_AUX = @Cod_CP;
    -- Recuperar valores de la siguiente tupla
    FETCH NEXT FROM cu_Mejores
        INTO @Cod_Alumno, @Nombres, @Cod_CP, @Promedio
END
```

4.3- Cursores...(Ejemplo 2)

```
-- Cerrar el cursor
CLOSE cu_Mejores
DEALLOCATE cu_Mejores
-- Mostrar en la consulta los datos solicitados
SELECT *
FROM @Mejores
```

4.3- Cursores...(Ejemplo 3)

Se tienen las siguientes tablas

#Calificacion (Cod_CP, Cod_Postulante, Nota)

Vacantes (Cod_CP, Nro_Vacantes)

Determinar la relación de ingresantes

#Ingresantes (Nro, Cod_CP, Cod_Postulante, Nota)

4.3- Cursores...(Ejemplo 3)

```
-- Crear tabla de ingresantes
SELECT TOP 0 Nro = 0, *
    INTO #Ingresantes
    FROM #CALIFICACION

-- Declarar variables para el cursor
DECLARE @Nro int,
        @Cod_Postulante varchar(8),
        @Cod_CP varchar(2),
        @Nota numeric(15,6),
        @Nro_Vacantes int,
        @Cod_CP_Aux varchar(2),
        @NotaUltimoIngresante numeric(15,6)
-- Inicializar Variables auxiliares
set @Cod_CP_Aux = '';
```

4.3- Cursores...(Ejemplo 3)

```
-- Crear Cursor
DECLARE cu_Ingresantes CURSOR
    FOR SELECT Cod_CP, Cod_Postulante, Nota
        FROM #CALIFICACION
        ORDER BY Cod_CP, Nota desc
-- Abrir Cursor
OPEN cu_Ingresantes
-- Recuperar la primera tupla
FETCH NEXT FROM cu_Ingresantes
    INTO @Cod_CP, @Cod_Postulante, @Nota,
```

4.3- Cursores...(Ejemplo 3)

```
-- Procesar cada tupla
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Determinar si se cambia de carrera
    IF @Cod_CP_Aux <> @Cod_CP
        BEGIN
            -- Recuperar Nro de vacantes de la carrera
            SELECT @Nro_Vacantes = Nro_Vacantes
                FROM VACANTES
                WHERE Cod_CP = @Cod_CP
            -- Inicializar contador
            set @Nro = 1;
            -- Actualizar variables auxiliares
            set @Cod_CP_Aux = @Cod_CP
            set @NotaUltimoIngresante = 0;
        END
    ELSE
        set @Nro = @Nro + 1;
```

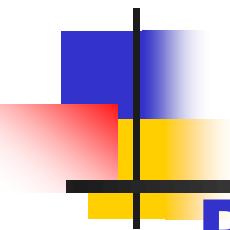
4.3- Cursores...(Ejemplo 3)

```
-- Verificar si es o no ingresante
IF (@Nro <= @Nro_Vacantes) OR
    (@NOTA = @NotaUltimoIngresante)
BEGIN
    -- Agregar a la tabla de ingresantes
    INSERT INTO #INGRESANTES
        VALUE(@Nro, @Cod_CP, @Cod_Postulante, @Nota)
    -- Guardar la nota del ultimo ingresante
    set @NotaUltimoIngresante = @Nota;
END;
-- Actualizar Cod_CP
set @Cod_CP_Aux = @Cod_CP;
-- Recuperar la siguiente tupla
FETCH NEXT FROM cu_Ingresantes
    INTO @Cod_CP, @Cod_Postulante, @Nota,
END
```

4.3- Cursores...(Ejemplo 3)

```
-- Cerrar el cursor  
CLOSE cu_Ingresantes  
DEALLOCATE cu_Ingresantes  
-- Retornar la información solicitada  
SELECT *  
    FROM #Ingresantes
```

Modelo Relacional - SQL

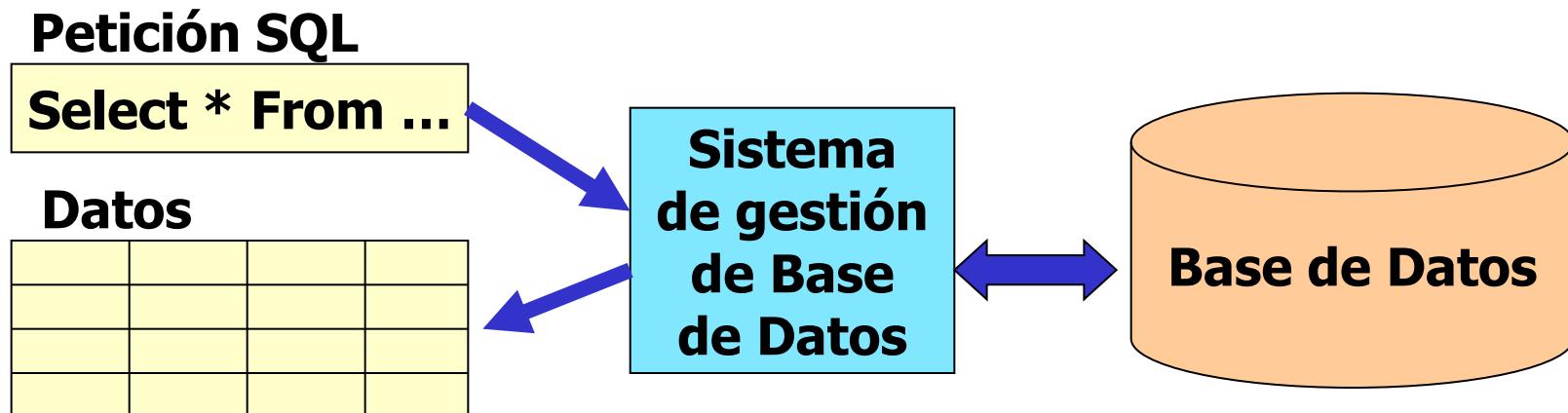


**USO INTERACTIVO Y
PROGRAMACION DE APLICACIONES**

4.1.1- El Lenguaje Estructurado de Consulta (SQL)

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos.

**SQL = Structured Query Language
(Lenguaje estructurado de Consulta)**



4.1.1- El Lenguaje Estructurado de Consulta (SQL)

El nombre SQL es en cierta medida un nombre inapropiado, porque es mucho más que un lenguaje de consulta, es una herramienta que permite controlar todas las funciones de un SGBD.

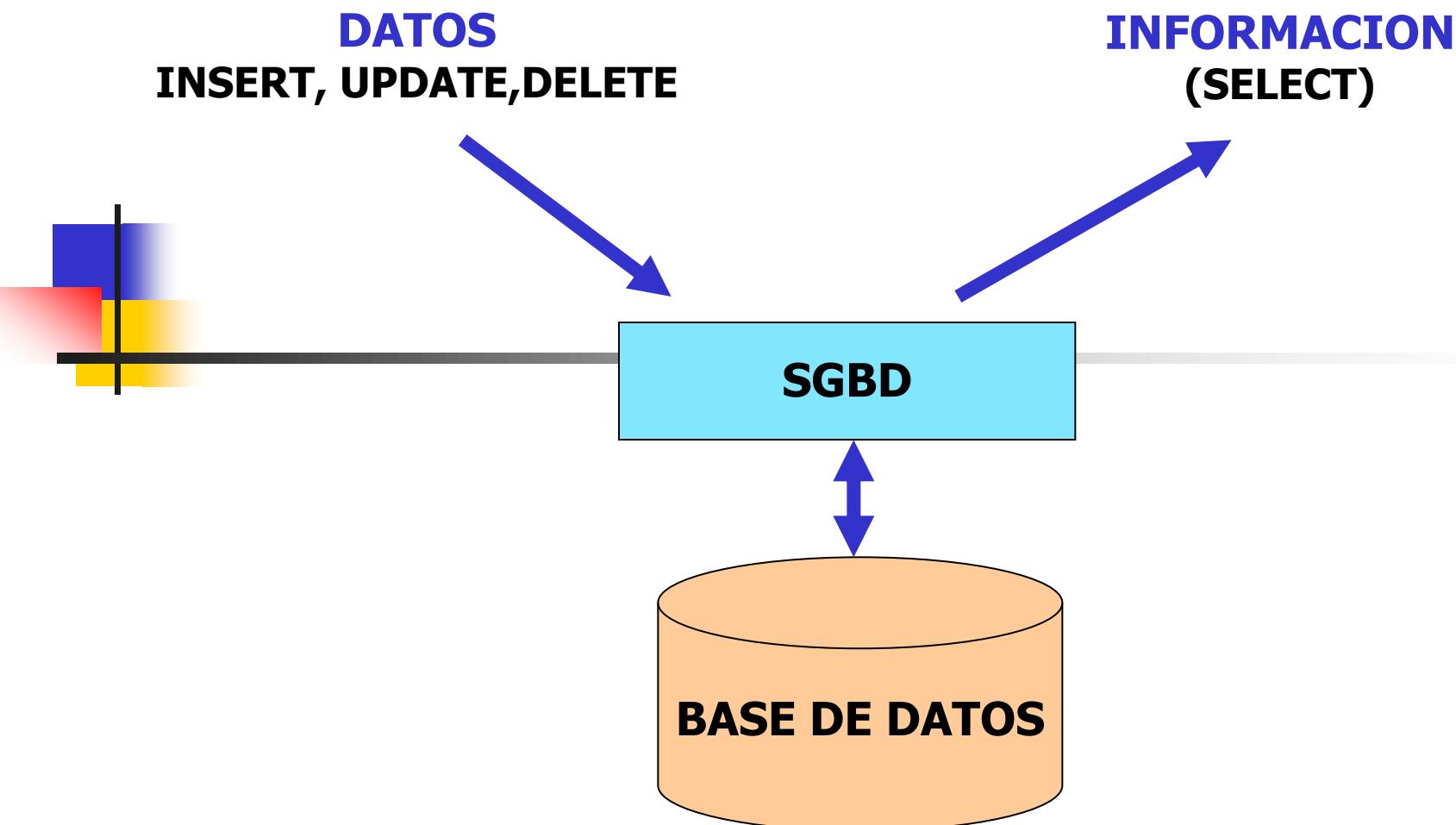
- **Definición de datos**
- **Recuperación de datos**
- **Manipulación de datos**
- **Control de acceso**
- **Compartición de datos**
- **Integridad de datos**

4.1.1- El Lenguaje Estructurado de Consulta (SQL)

SQL es un Lenguaje de:

- **Programación de Base de datos**
- **Administración de Base de Datos**
- **Cliente/Servidor**
- **Base de datos distribuida**
- **Pasarela de Base de Datos**

4.1.2- Recuperación de datos



4.1.3- Conceptos básicos

Principales grupos de sentencias

■ Definición de Datos

- **CREATE**
- **DROP**
- **ALTER**

■ Manipulación de Datos

- **SELECT**
- **INSERT**
- **DELETE**
- **UPDATE**

Control de Acceso

- **GRANT**
- **REVOKE**

■ Control de Transacciones

- **COMMIT**
- **ROLLBACK**

4.1.4- Consultas simples

Estructura de la sentencia SELECT

SELECT Atributos

FROM Tablas

WHERE Condición

GROUP BY Atributos_Grupo

HAVING Condición_Grupo

ORDER BY Atributos_Ordenamiento

4.1.4- Consultas simples...

Ejemplos:

ALUMNO(Codigo, Apellido_Paterno, Apellido_Materno, Nombres, CP)

```
SELECT Apellido_Paterno, Apellido_Materno, Nombres  
1   FROM Alumno
```

2

```
SELECT Apellido_Paterno, Apellido_Materno, Nombres  
1   FROM Alumno  
2   WHERE CP='CO'
```

3

4.1.4- Consultas simples...

Condiciones:

- Test de Comparación

=, <>, <, <=, >, >=

- Test de Rango

Between

- Test de pertenencia a conjunto

IN

- Test de correspondencia con patrón

Like

- Test de valor nulo

IS NULL

4.1.4- Consultas simples...

Funciones de resumen:

SUM: Calcula la suma de una columna de datos.
(Columna de tipo numérico).

AVG: Calcula el promedio de una columna de datos.
(Columna de tipo numérico).

MIN: Retorna el valor mínimo de una columna

MAX: Retorna el valor máximo de una columna

COUNT: Cuenta el número de registros. El resultado es un valor entero.

4.1.4- Consultas simples...

En base a la siguiente tabla:

ALUMNO(Codigo, Apellido_Paterno, Apellido_Materno, Nombres, CP)

Determinar el número de estudiantes de la universidad

```
SELECT Count(*)  
      FROM Alumno
```

Determinar el número total de estudiantes de las carreras profesionales de Ingeniería de la universidad

```
SELECT Count(*) 3  
1   FROM Alumno  
2 WHERE CP like 'I%'
```

4.1.4- Consultas simples...

Determinar el número de estudiantes de cada carrera profesionales de Ingeniería de la universidad

SELECT CP, Count(*) as NroAlumnos

4

1 FROM Alumno

2 WHERE CP like 'I%'

3 GROUP BY CP

4.1.4- Consultas simples...

Determinar el número de estudiantes de cada carrera profesionales de Ingeniería de la universidad y que tengan más de 200 alumnos. La relación resultante que muestre ordenado por número de alumnos y en orden descendente.

SELECT CP, Count(*) as NroAlumnos

5

1 FROM Alumno

2 WHERE CP like 'I%'

3 GROUP BY CP

4 HAVING Count(*) > 200

6 ORDER BY NroAlumnos Desc

Sistemas de Base de Datos

4.1.5- Composición (Consultas multitable)

```
SELECT *  
FROM ALUMNO inner join CP
```

TABLA ALUMNO

Codigo	Nombres	ID_Carrera
121	Pedro	TU
234	Ana	AG
148	Eva	TU

TABLA CP

ID_Carrera	Nombre
TU	Turismo
AG	Agronomía

FROM ALUMNO INNER JOIN CP

ALUMNO.Codigo	ALUMNO.Nombres	ALUMNO.ID_Carrera	CP.ID_Carrera	CP.Nombre
121	Pedro	TU	TU	Turismo
121	Pedro	TU	AG	Agronomía
234	Ana	AG	TU	Turismo
234	Ana	AG	AG	Agronomía
148	Eva	TU	TU	Turismo
148	Eva	TU	AG	Agronomía

Sistemas de Base de Datos

4.1.5- Composición (Consultas multitablea)...

```
SELECT *  
FROM ALUMNO INNER JOIN CP  
ON ALUMNO.ID_Carrera = CP.ID_Carrera
```

Tabla resultante antes de aplicar la cláusula ON

ALUMNO.Codigo	ALUMNO.Nombres	ALUMNO.ID_Carrera	CP.ID_Carrera	CP.Nombre
121	Pedro	TU	TU	Turismo
121	Pedro	TU	AG	Agronomía
234	Ana	AG	TU	Turismo
234	Ana	AG	AG	Agronomía
148	Eva	TU	TU	Turismo
148	Eva	TU	AG	Agronomía

Tabla resultante después de aplicar la cláusula ON

ALUMNO.Codigo	ALUMNO.Nombres	ALUMNO.ID_Carrera	CP.ID_Carrera	CP.Nombre
121	Pedro	TU	TU	Turismo
234	Ana	AG	AG	Agronomía
148	Eva	TU	TU	Turismo

4.1.5.1.- Composición Interna

Este tipo de composición se utiliza cuando las cardinalidades mínimas son 1 (Obligatorios).



```
SELECT *  
FROM ALUMNO A inner join CARRERA C  
ON A.ID_Carrera = C.ID_Carrera
```

Ejercicios. Dada las siguientes tablas:

Carrera_Profesional(Cod_CP, Nombre_CP)

Alumno(Cod_Alumno, Paterno, Materno, Nombres, Cod_CP)

Docente(Cod_Docente, Paterno, Materno, Nombres,
Categoría, Regimen)

Asignatura(Cod_Asignatura,Cod_CP,Nombre_Asignatura,
Categoría, Creditos)

Catalogo(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Docente)

Matricula(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Alumno, Nota)

Ejercicios resueltos

Prob 1. Determinar el número de alumnos matriculados por semestre y por Carrera Profesional

-- Agregar el código de la carrera a la tabla matricula

```
SELECT M.* , Cod_CP  
      INTO #matriculaCP  
    FROM Matricula M inner join Alumno A  
      ON M.Cod_Alumno = A.Cod_Alumno
```

-- Contar matriculados por semestre de cada carrera profesional

```
SELECT Semestre, Cod_CP,  
      Nro_Matriculados = count(distinct Cod_Alumno)  
    INTO #alumnoSemestre  
  FROM #matriculaCP  
 GROUP BY Semestre, Cod_CP  
 ORDER BY Semestre, Cod_CP
```

-- Agregar datos de la Carrera profesional

```
SELECT T1.Semestre, T1.Cod_CP, Nombre_CP,  
      Nro_Matriculados  
    FROM #alumnoSemestre T1,Carrera T2  
   WHERE T1.Cod_CP = T2.Cod_CP  
 ORDER BY Semestre, T1.Cod_CP
```

Ejercicios resueltos



Prob 2. **Determinar** la relación de asignaturas que en el último semestre tengan menos de 7 alumnos matriculados, con el objetivo de desactivar dichas asignaturas en el semestre.

PRIMER ALGORITMO

-- Determinar ultimo semestre

```
SELECT ultimoSemestre = max(Semestre)
      INTO #ultimoSemestre
      FROM Matricula
```

-- Seleccionar las asignaturas que se dictan en este último Semestre

```
SELECT M.*
      INTO #AsignaturasUltimoSemestre
      FROM Matricula M inner join #ultimoSemestre U
      ON M.Semestre = U.ultimoSemestre
```

-- Determinar las asignaturas que tengan menos de 7 alumnos

```
SELECT Semestre, Cod_Asignatura, Cod_CP, Grupo,
      Nro_Alumnos = count(Cod_Alumno)
      INTO #asignaturasAlumnos
      FROM #AsignaturasUltimoSemestre
      GROUP BY Semestre, Cod_Asignatura, Cod_CP, Grupo
      HAVING count(Cod_Alumno) < 7
```

-- Agregar datos de la asignatura

```
SELECT A.* , Nro_Alumnos
      FROM #asignaturasAlumnos T1 inner join Asignatura A
      ON T1.Cod_Asignatura=A.Cod_Asignatura and T1.Cod_CP=A.Cod_CP
```

SEGUNDO ALGORITMO

-- Determinar ultimo semestre

```
Declare @UltimoSemestre varchar(7);
SELECT @UltimoSemestre = max(Semestre)
FROM Matricula
```

-- Determinar las asignaturas del ultimo semestre que tengan
menos de 7 alumnos

```
SELECT Cod_Asignatura, Cod_CP, Grupo,
       Nro_Alumnos = count(Cod_Alumno)
  INTO #asignaturasAlumnos
     FROM Matricula M
    WHERE Semestre = @UltimoSemestre
  GROUP BY Cod_Asignatura, Cod_CP, Grupo
 HAVING count(Cod_Alumno) < 7
```

-- Agregar datos de la asignatura

```
SELECT A.* , Nro_Alumnos
  FROM #asignaturasAlumnos T1 inner join Asignatura A
    ON T1.Cod_Asignatura=A.Cod_Asignatura and T1.Cod_CP=A.Cod_CP
```

Ejercicios resueltos



Prob 3. Obtener la ficha de seguimiento
del estudiante de código 980115.

PRIMER ALGORITMO

-- Obtener la información de las tablas Matrícula y Asignatura

```
select M.Semestre, M.Cod_Asignatura, A.Categoría, A.Creditos, M.Nota
from MATRICULA M inner join ASIGNATURA A
on (M.Cod_Asignatura = A.Cod_Asignatura) AND
(M.Cod_CP = A.Cod_CP)
where Cod_Alumno = '980115' and
Nota in ('11','12','13','14','15','16','17','18','19','20')
```

SEGUNDO ALGORITMO

-- Recuperar matriculas del estudiante 980115

```
select Semestre, Cod_Asignatura, Cod_CP, Nota  
    into #MATRICULA  
    from MATRICULA  
   where Cod_Alumno = '980115' and  
         Nota in('11','12','13','14','15','16','17','18','19','20')
```

-- Completar información de asignatura

```
select M.Semestre, M.Cod_Asignatura, A.Nombre_Asignatura,  
      A.Categoría, A.Creditos, M.Nota  
  from #MATRICULA M inner join ASIGNATURA A  
  on (M.Cod_Asignatura = A.Cod_Asignatura) and  
      (M.Cod_CP = A.Cod_CP)
```

TERCER ALGORITMO

-- Declarar variable

```
declare @Cod_Alumno varchar(6);
set @Cod_Alumno = '980115';
```

-- Recuperar matriculas del estudiante 980115

```
select Semestre, Cod_Asignatura, Cod_CP, Nota
into #MATRICULA1
from MATRICULA
where (Cod_Alumno = @Cod_Alumno) and
      (Nota in('11','12','13','14','15','16','17','18','19','20'))
```

-- Completar información de asignatura

```
select M.Semestre, M.Cod_Asignatura, A.Nombre_Asignatura,
       A.Categoría, A.Creditos, M.Nota
  from #MATRICULA1 M inner join ASIGNATURA A
    on (M.Cod_Asignatura = A.Cod_Asignatura) and
       (M.Cod_CP = A.Cod_CP)
```

4.1.5.2.- Composición Externa

Dadas las siguientes tablas, determinar la relación de préstamos con sus respectivos saldos

TABLA PRESTAMO

ID_Prestamo	Importe
PA-134	600
PA-345	1200
PA-456	850

TABLA AMORTIZACION

ID_Amortizacion	Importe	Id_Prestamo
RC-234	450	PA-134
RC-241	850	PA-456

¿Es la siguiente sentencia la solución?

```
SELECT P.* , (P.Importe - A.Importe) as Saldo  
FROM PRESTAMO P inner join AMORTIZACION A  
ON P.ID_Prestamo = A.ID_Prestamo
```

4.1.5.2.- Composición Externa...

La sentencia: FROM PRESTAMO P inner join AMORTIZACION A
Genera la siguiente tabla

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

La sentencia: ON P.ID_Prestamo = A.ID_Prestamo
selecciona las tuplas que no están subrayadas

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

4.1.5.2.- Composición Externa...

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

En Consecuencia la relación resultante es:

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

El préstamo PA-345 ha sido ignorado debido a que no empareja con ninguna de las tuplas de la tabla de AMORTIZACION

4.1.5.2.- Composición Externa...

Este tipo de composición se utiliza cuando las cardinalidades mínimas son 0 (No son obligatorios).



Como el modelo anterior indica que hay PRESTAMOS que no necesariamente están relacionados con AMORTIZACIONES, entonces estas instancias desaparecerán cuando se hace una Composición Interna, en su lugar se debe realizar una Composición Externa.

4.1.5.2.- Composición Externa...

Sintaxis:

```
SELECT atributos
      FROM tabla1 (LEFT/RIGHT) OUTER JOIN tabla 2
           ON tabla1.atributo=tabla2.atributo
```

OUTER JOIN
LEFT OUTER JOIN
RIGHT OUTER JOIN

Considera los registros de ambas tablas
Considera todos los registros de la tabla de la izquierda
Considera todos los registros de la tabla de la derecha

4.1.5.2.- Composición Externa...

La solución al problema propuesto es:

```
SELECT P.* , (P.Importe - A.Importe) as Saldo  
FROM PRESTAMO P left outer join AMORTIZACION A  
ON P.ID_Prestamo = A.ID_Prestamo
```

La relación resultante es:

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-345	1200	null	null	null
PA-456	850	RC-241	850	PA-456

Ejercicios. Dada las siguientes tablas:

Carrera_Profesional(Cod_CP, Nombre_CP)

Alumno(Cod_Alumno, Paterno, Materno, Nombres, Cod_CP)

Docente(Cod_Docente, Paterno, Materno, Nombres,
Categoría, Regimen)

Asignatura(Cod_Asignatura,Cod_CP,Nombre_Asignatura,
Categoría, Creditos)

Catalogo(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Docente)

Matricula(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Alumno, Nota)

Ejercicios resueltos

Prob 1. Determinar el número de créditos aprobados por los alumnos de ingeniería informática

-- Recuperar los alumnos de Ingeniería informática

```
SELECT Cod_Alumno, Paterno, Materno, Nombres  
      INTO #AlumnosIN  
     FROM Alumno
```

-- Seleccionar matriculas con asignaturas aprobadas y de Informática

```
SELECT Cod_Asignatura, Cod_Alumno, Nota  
      INTO #MatriculaIN  
     FROM Matricula  
    WHERE (Cod_CP = 'IN') and (Nota in (11,12,13,14,15,16,17,18,19,20))
```

-- Agregar datos de la Carrera profesional

```
SELECT T1.Semestre, T1.Cod_CP, Nombre_CP,  
          Nro_Matriculados  
     FROM #alumnoSemestre T1,Carrera T2  
    WHERE T1.Cod_CP = T2.Cod_CP  
ORDER BY Semestre, T1.Cod_CP
```

4.1.6.- Sub consultas



Una Sub consulta es una consulta que está anidada dentro de una sentencia SELECT, INSERT, UPDATE o DELETE, o dentro de otra sub consulta.

4.1.6.- Sub consultas...

- Como parte de las cláusulas Where y Having
 - Con operadores de comparación
 - Con IN o NOT IN
 - Con EXISTS o NOT EXISTS
 - Con ANY, SOME o ALL
- Como Tablas en composiciones
- En lugar de expresiones

4.1.6.- Sub consultas...

Las sub consultas se pueden utilizar en las siguientes partes de una sentencia SELECT

```
Select Atributo1, ..., (sub consulta)
      from Tabla1, (sub consulta)
     where (sub consulta)
   group by Atributo_Grupo1, ...
  having (sub consulta)
 order by Atributo_Ordenamiento
```

4.1.6.- Sub consultas...

Sub Consultas... con operadores de comparación

Seleccionar los préstamos de mayor importe

```
Select *  
  from Prestamo  
where Importe = ( Select max(Importe)  
                      from Prestamo  
                  )
```

Nota.- Una sub consulta referenciada mediante un operador de comparación debe retornar un único valor.

4.1.6.- Sub consultas...

Sub Consultas... con IN o NOT IN

Seleccionar los prestatarios que tengan más de 5 préstamos

```
Select *  
from Prestatario  
where Cod_Prestatario IN  
      ( Select Cod_Prestatario  
            from Prestamo  
            group by Cod_Prestatario  
            having count(Doc_Prestamo) > 5  
      )
```

Nota.- Una sub consulta referenciada mediante IN o NOT IN debe retornar una tabla con una sola columna (Una lista).

4.1.6.- Sub consultas...

Sub Consultas... con EXISTS y NOT EXISTS

Seleccionar la relación de alumnos que en el semestre 2005-I no se hayan presentado a una o más asignaturas

```
Select *  
  from Alumno A  
 where EXISTS (Select *  
                 from (select *  
                           from Matricula  
                         where (M.Nota = 'NSP') and  
                               (Semestre = '2005-I')  
                     ) M  
               where (A.Cod_Alumno = M.Cod_Alumno)  
             )
```

Nota.- Esta sub consulta es también un ejemplo de una sub consulta correlacionada o repetida, pues en la sub consulta se utiliza valores de la consulta principal. Este tipo de sub consultas se ejecutan por cada fila de la consulta principal, en consecuencia son muy lentos.

4.1.6.- Sub consultas...

Sub Consultas... con ANY, SOME o ALL

Seleccionar la relación de asignaturas dictadas en el semestre 2005-I

```
Select *  
from Asignatura  
where Cod_Asignatura=ANY (Select Cod_Asignatura  
                           from Matricula  
                           where Semestre = '2005-I'  
)
```

4.1.6.- Sub consultas...

Sub Consultas... Como tablas en composiciones

Seleccionar la relación de préstamos con sus respectivos saldos

```
▪ Select P.* ,  
      (P.Importe-IsNull(C.Cancelado,0)) as Saldo  
  from Prestamo P left outer join  
    (Select Doc_Prestamo, sum(Importe) Cancelado  
     from Cancelacion  
     group by Doc_Prestamo) C  
   on P.Doc_Prestamo = C.Doc_Prestamo
```

4.1.6.- Sub consultas...

Sub Consultas... En lugar de expresiones

Seleccionar la relación de prestatarios con el número de préstamos realizados por cada uno de ellos.

```
■ Select T1.* ,  
      (Select count(T2.Doc_Prestamo)  
       from Prestamo T2  
      where T1.Cod_Prestatario=T2.Cod_Prestatario  
      ) as NroPrestamos  
  from Prestatario T1
```

Seleccionar la relación de prestatarios con el número de préstamos realizados por cada uno de ellos.

-- Contar nro. De préstamos por prestatario

WITH T1 (CodPrestatario, NroPrestamos)

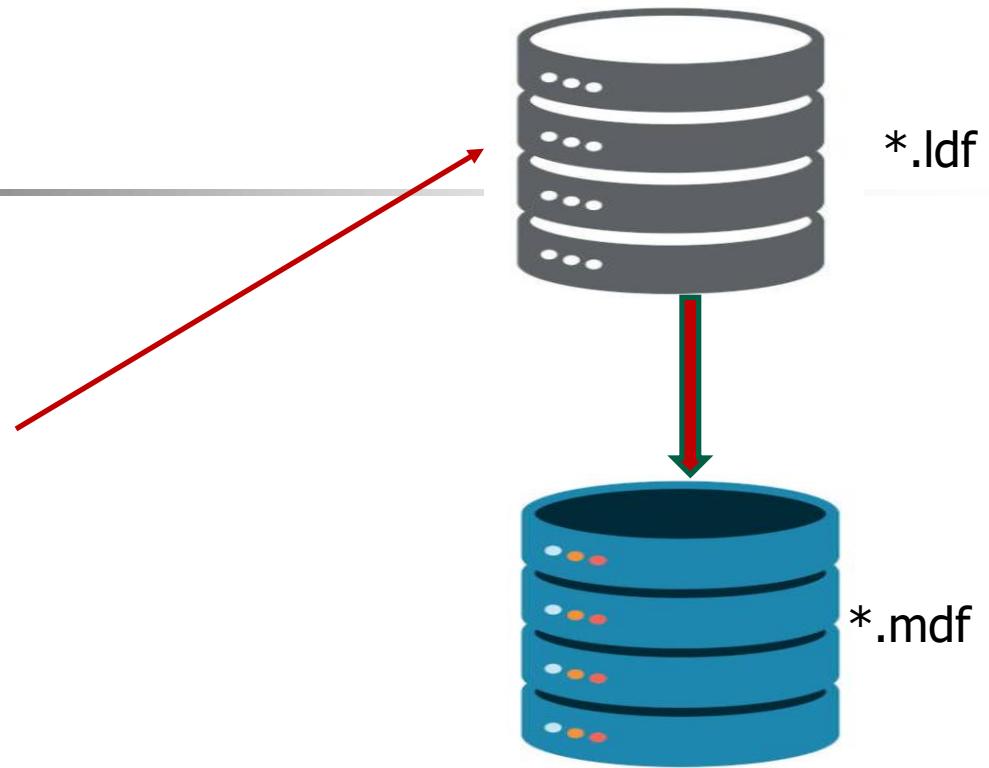
AS (

```
    Select CodPrestatario, count(DocPrestamo) as NroPrestamos  
        from Prestamo  
        group by CodPrestatario)
```

-- Relacionar con tabla de prestatario

```
Select P.* , IsNull(NroPrestamos, 0) as NroPrestamos  
    from Prestatario P left outer join T1 T  
    on P.CodPrestatario = T.CodPrestatario
```

TRANSACCIONES



Transacciones

- **Unidad lógica de procesamiento**

Secuencia de operaciones que implican accesos a la base de datos

- **Pero también se considera...**

- **Unidad lógica de integridad**

- **Unidad lógica de concurrencia**

- **Unidad lógica de recuperación**

- **Una transacción es atómica**

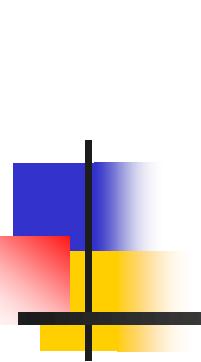
O se ejecutan todas las operaciones que componen la transacción, o no se realiza ninguna

- **ejemplo:**

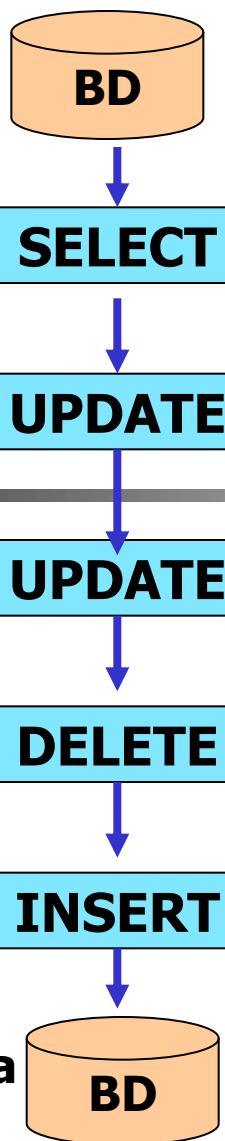
transferencia de dinero entre dos cuentas bancarias

Transacciones...

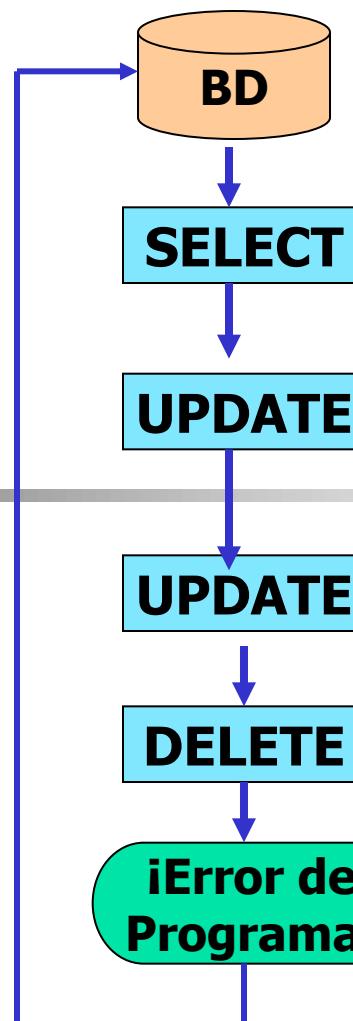
Estado
antes de la
Transacción



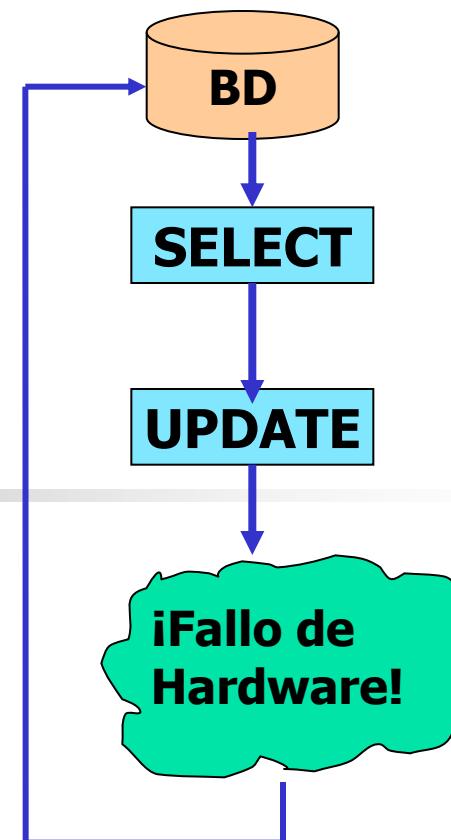
TRANSAKCION



Estado
después de la
Transacción



El SGBD deshace
Todos los cambios



El SGBD deshace
Todos los cambios

Transacciones

- **Tipos de fallos**
 - **Fallo del computador (hardware)**
 - **Errores de software. Desbordamiento, división por cero, etc.**
 - **Condiciones de excepción detectadas por la transacción.
(Por ejemplo no se tiene el saldo para cubrir un retiro)**
 - **Imposición del control de concurrencia
(Transacciones en bloqueo mortal)**
 - **Fallo del disco**
 - **Problemas y catástrofes físicos**

Transacciones...

Propiedades ACID

- **Atomicidad**

- Todo o nada



SubSistema de Recuperación

- Conservación de **consistencia**

- T lleva la BD de un estado de consistencia a otro
 - No necesariamente se mantiene la consistencia "a mitad de T"

SS de Integridad + Programadores

- **Aislamiento (isolation)**

- T no muestra los cambios que produce hasta que finaliza
 - Puede no imponerse de forma estricta (niveles de aislamiento)

SS de Control de Concurrencia

- **Durabilidad**

- Una vez que T finaliza con éxito y es confirmada, los cambios perduran aunque el sistema falle después

SubSistema de Recuperación

Transacciones

- **Sea Ti una transacción para transferir S/. 500.00 de la cuenta A a la cuenta B. Se puede definir dicha transacción como:**

Ti: leer(A)

 A \leftarrow A – 500

 escribir(A)

 leer(B)

 B \leftarrow B + 500

 escribir(B)

Transacciones

Atomicidad

- **Todas o ninguna de las operaciones de la transacción son ejecutadas.**
- **Si la transacción falla, sus resultados parciales deben ser deshechos.**
- **La actividad de preservar la atomicidad de la transacción en la presencia de abortos de la transacción debido a errores de entrada, interbloqueos es llamada recuperación de la transacción.**
- **El sub sistema de recuperación es el responsable de asegurar la atomicidad en la presencia de fallos.**

Transacciones

Consistencia

- **Consistencia Interna**
 - Una transacción que se ejecuta al terminar deja la base de datos en estado consistente.
 - Las transacciones no violan las restricciones de integridad de las bases de datos.

Transacciones

Aislamiento

- **Serialización**
 - Si varias transacciones son ejecutadas concurrentemente, los resultados deben ser los mismos como si ellas fueran ejecutadas serialmente en algún orden.
- **Resultados incompletos**
 - Una transacción incompleta no puede revelar sus resultados a otras transacciones antes de su compromiso.

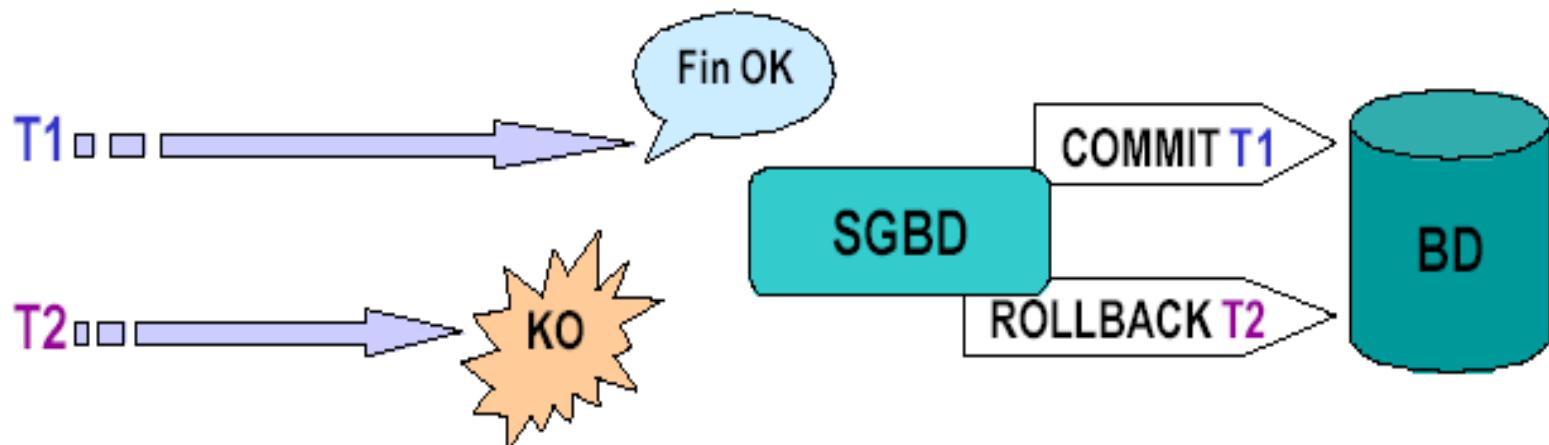
Transacciones

Durabilidad

- **Una vez que la transacción se compromete, el sistema debe garantizar que los resultados de sus operaciones nunca serán perdidos, a pesar de fallas subsecuentes.**
- **Recuperación de la base de datos**

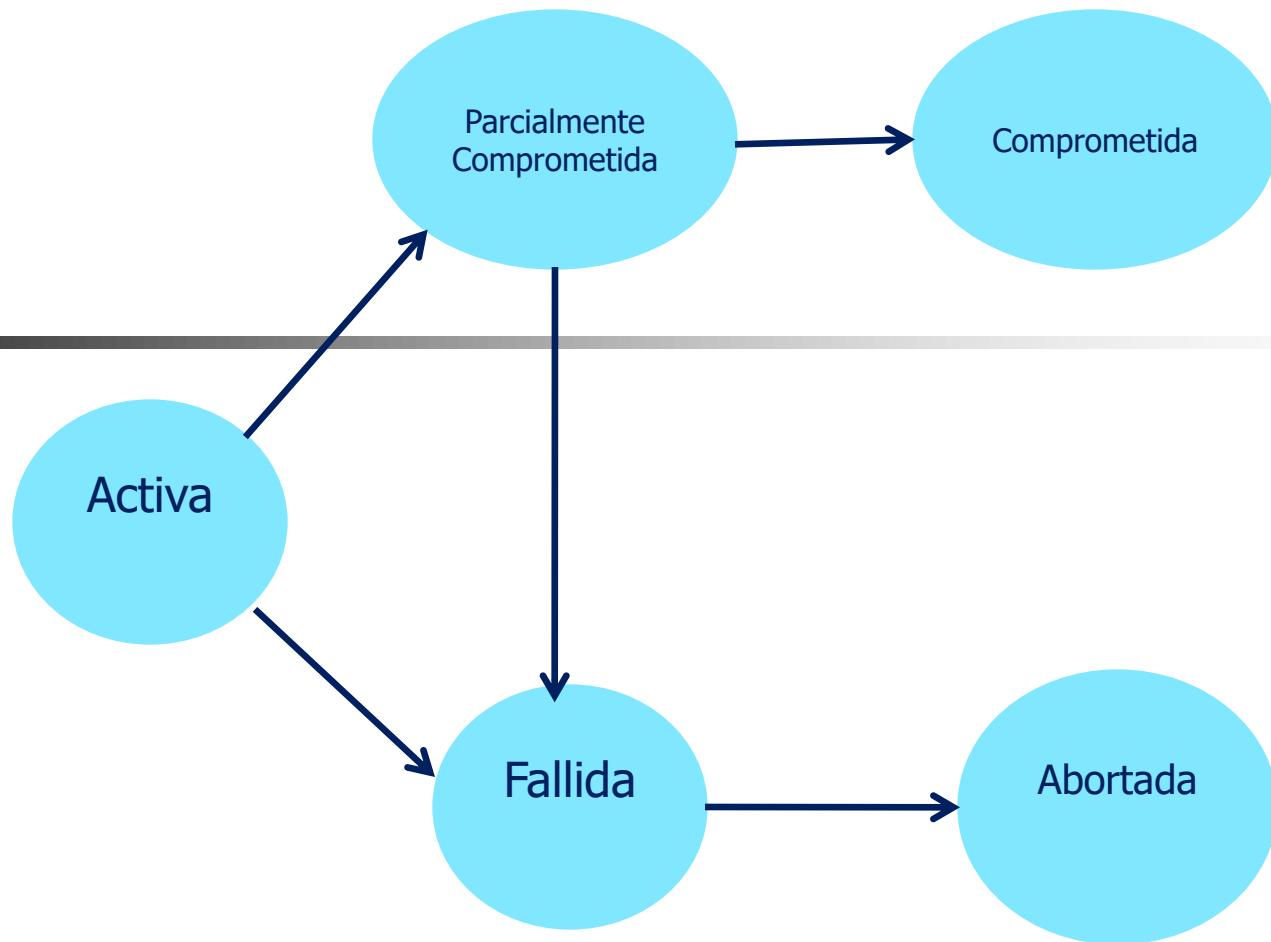
Transacciones...

- **Inicio** de una transacción
 - Sentencia SQL (LLD o LMD) interactiva
 - Sentencia SQL incluida en un programa (si no tiene ya transacción en progreso)
- **Fin** de una transacción
 - Confirmar (COMMIT) *xor* Anular (ROLLBACK)
 - Ambas operaciones pueden ser de tipo explícito o implícito



Transacciones...

Estados de una transacción



Transacciones... (SQL Server)

Sentencias

- **Begin transaction.**- Inicia una transacción
- **Commit.**- Confirma la transacción y se convierte en una parte permanente de la base de datos.
- **Rollback.**- Cancela o revierte la transacción, borrando todas las modificaciones de la base de datos.

BEGIN DISTRIBUTED TRANSACTION	ROLLBACK TRANSACTION
BEGIN TRANSACTION	ROLLBACK WORK
COMMIT TRANSACTION	SAVE TRANSACTION
COMMIT WORK	

BEGIN TRY

Begin transaction

{ sql_statement | statement_block }

Commit

END TRY

BEGIN CATCH

[{ sql_statement | statement_block }]

Rollback

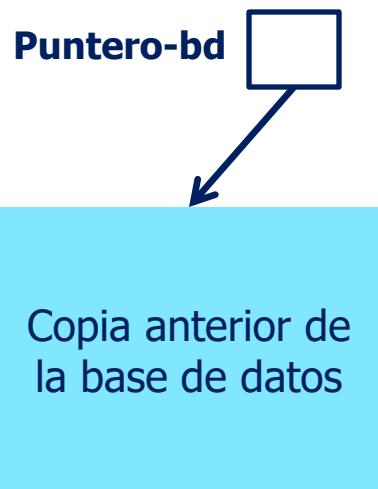
END CATCH

[;]

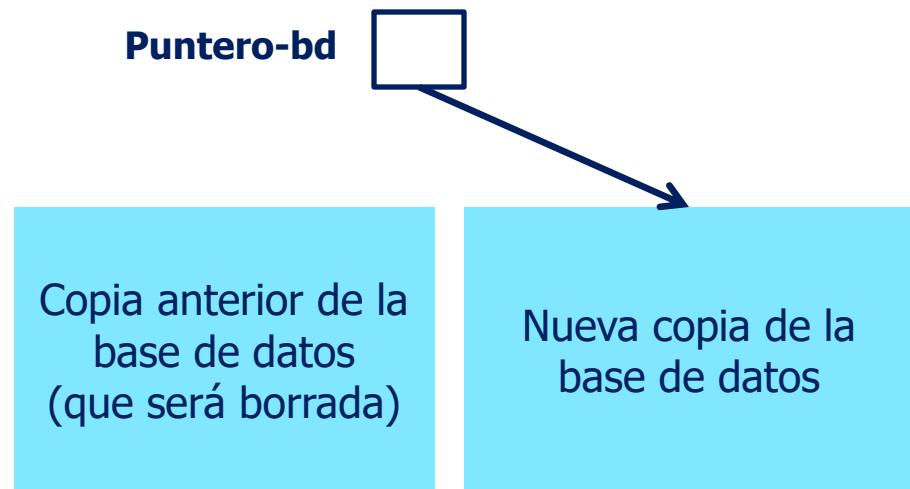
Transacciones...

Implementación de la atomicidad y la durabilidad

Copia en la sombra. Esquema simple pero extremadamente ineficiente. Asume que sólo una transacción está activa en cada momento.



(a) Antes de la actualización



(b) Despues de la actualización

Si el sistema falla antes de que la transacción haya sido confirmada, entonces solamente se borra la nueva copia.

Sistema de Recuperación

Clasificación de los fallos

Fallo en la transacción.

- Error lógico (desbordamiento).
- Error del sistema (Inter bloqueo)

Caída del sistema

Fallo del disco

Se pueden proponer algoritmos para garantizar la consistencia de la base de datos y la atomicidad de las transacciones a pesar de los fallos.

Sistema de Recuperación

Estructura del almacenamiento

Para entender cómo se pueden garantizar las propiedades de atomicidad y durabilidad de una transacción, se deben comprender las estructuras de almacenamiento y sus métodos de acceso.

Tipos de almacenamiento

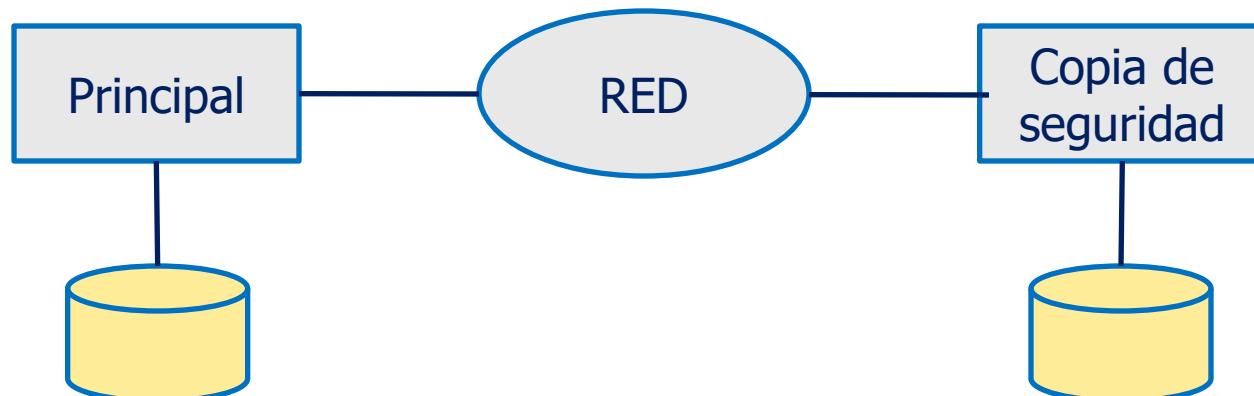
- **Almacenamiento volátil.**- La información no suele sobrevivir a las caídas del sistema. Ejemplos: Memoria principal y memoria cache.
- **Almacenamiento no volátil.**- Sobrevive a las caídas del sistema. Ejemplos: Discos, CD's, DVD's.
- **Almacenamiento estable.**- La información que reside en almacenamiento estable “**nunca**” se pierde.

Sistema de Recuperación

Implementación del almacenamiento estable

Para implementar almacenamiento estable se debe replicar la información necesaria en varios medios de almacenamiento no volátil (normalmente discos) con modos de fallo independientes.

- **RAID.**- (Disposición redundante de discos independientes)
Discos con imagen.
(No protegen contra la pérdida de datos debida a desastres naturales)
- **Almacenamiento estable remoto.**- Copias de datos en el sistema de discos locales y a través de una red de computadoras.



Sistema de Recuperación

RAID (Redundant Array of Independent Disks)

Son técnicas de organización de los discos, denominadas colectivamente disposición redundante de discos independientes; con el propósito de conseguir mayor rendimiento y fiabilidad.

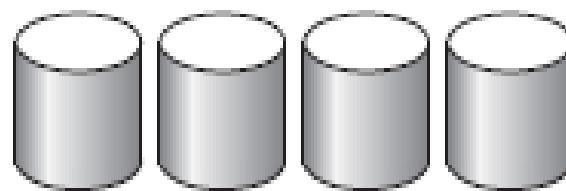


La solución al problema de la fiabilidad es introducir la redundancia; es decir, se guarda información adicional que normalmente no se necesita pero que se puede utilizar en caso de fallo de un disco para reconstruir la información perdida. Por tanto, aunque falle un disco, no se pierden datos

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 0: Distribución no redundante



(a) RAID 0: distribución no redundante

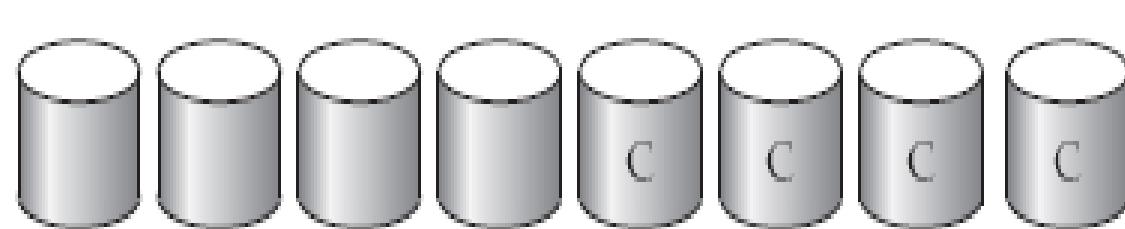
Hace referencia a disposiciones de discos con distribución en el nivel de bloques, pero sin redundancia.

Mejora rendimiento pero no fiabilidad.

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 1: Discos con imagen



(b) RAID 1: discos con imagen

Hace referencia a la creación de imágenes del disco con distribución de bloques.

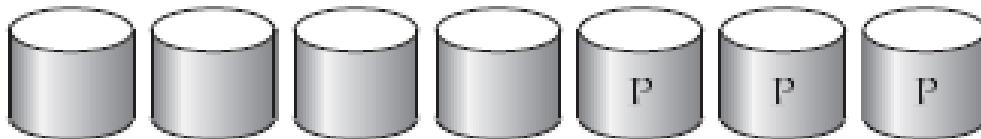
La figura muestra una organización con imagen que guarda cuatro discos de datos.

Nota.- la C indica una segunda copia de los datos.

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 2: Códigos de corrección de errores tipo memoria.



(c) RAID 2: códigos de corrección de errores tipo memoria

También conocido como organización de códigos de corrección de errores tipo memoria memory-style-error-correcting-code organization, ECC), emplea bits de paridad. Hace tiempo que los sistemas de memoria utilizan los bits de paridad para la detección y corrección de errores. Cada byte del sistema de memoria puede tener asociado un bit de paridad que registra si el número de bits del byte que valen uno es par (paridad = 0) o impar (paridad = 1). Si alguno de los bits del byte se deteriora (un uno se transforma en cero y viceversa) la paridad del byte se modifica y, por tanto, no coincide con la paridad guardada. Análogamente, si el bit de paridad guardado se deteriora no coincide con la paridad calculada. Por tanto, el sistema de memoria detecta todos los errores que afectan a un número impar de bits de un mismo byte. Los esquemas de corrección de errores guardan dos o más bits adicionales y pueden reconstruir los datos si se deteriora un solo bit.

Nota.- la P indica los bits para la corrección de errores.

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 3: Paridad con bits entrelazados



(d) RAID 3: paridad con bits entrelazados

Organización de paridad con bits entrelazados, mejora respecto al nivel 2 al aprovechar que los controladores de disco, a diferencia de los sistemas de memoria, pueden detectar si los sectores se han leído correctamente, por lo que se puede utilizar un solo bit de paridad para la corrección y para la detección de los errores. La idea es la siguiente: si uno de los sectores se deteriora, se sabe exactamente el sector que es y, para cada uno de sus bits, se puede determinar si es un uno o un cero calculando la paridad de los bits correspondientes de los sectores de los demás discos. Si la paridad de los bits restantes es igual que la paridad guardada, el bit perdido es un cero; en caso contrario, es un uno.

El nivel 3 de RAID es tan bueno como el nivel 2, pero resulta menos costoso en cuanto al número de discos adicionales (sólo tiene un disco adicional)

Nota.- la P indica los bits para la corrección de errores.

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 4: Paridad con bloques entrelazados



(e) RAID 4: paridad con bloques entrelazados

Organización de paridad con bloques entrelazados, emplea la distribución del nivel de bloques, como RAID 0, y, además, guarda un bloque de paridad en un disco independiente para los bloques correspondientes de los otros N discos. Si falla uno de los discos, se puede utilizar el bloque de paridad con los bloques correspondientes de los demás discos para restaurar los bloques del disco averiado.

La lectura de un bloque sólo accede a un disco, lo que permite que los demás discos procesen otras solicitudes. Por tanto, la velocidad de transferencia de datos de cada acceso es menor, pero se pueden ejecutar en paralelo varios accesos de lectura, lo que produce una mayor velocidad global de E/S.

Nota.- la P indica los bloques de paridad.

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 5: Paridad distribuida con bloques entrelazados



(f) RAID 5: paridad distribuida con bloques entrelazados

Paridad distribuida con bloques entrelazados, mejora respecto al nivel 4 al dividir los datos y la paridad entre todos los $N + 1$ discos, en vez de guardar los datos en N discos y la paridad en uno solo. En el nivel 5 todos los discos pueden atender a las solicitudes de lectura, a diferencia del nivel 4 de RAID, en el que el disco de paridad no puede participar, por lo que el nivel 5 aumenta el número total de solicitudes que pueden atenderse en una cantidad de tiempo dada. En cada conjunto de N bloques lógicos, uno de los discos guarda la paridad y los otros N guardan los bloques.

Nota.- la P indica los bloques de paridad.

Sistema de Recuperación

NIVELES DE RAID

(a) RAID nivel 5: Redundancia P + Q



(g) RAID 6: redundancia P + Q

También denominado esquema de redundancia P+Q, es muy parecido al nivel 5, pero guarda información redundante adicional para la protección contra los fallos de disco múltiples. En lugar de utilizar la paridad, el nivel 6 utiliza códigos para la corrección de errores como los de Reed–Solomon. En el esquema de la Figura se guardan dos bits de datos redundantes por cada cuatro bits de datos (a diferencia del único bit de paridad del nivel 5) y el sistema puede tolerar dos fallos del disco.

Nota.- la P indica los bloques de paridad.

Sistema de Recuperación

RAID, consideraciones adicionales

- Los diferentes fabricantes han propuesto varias modificaciones a los esquemas RAID básicos descritos, empleando su propia terminología para las variantes.
- Se tiene también sistemas **RAID hardware**, que permiten obtener ventajas significativas. Las implementaciones RAID hardware pueden utilizar RAM no volátil para registrar las operaciones de escritura antes de llevarlas a cabo. En caso de fallo de corriente, cuando el sistema se restaura, recupera la información relativa a las operaciones de escritura incompletas de la memoria RAM no volátil y completa esas operaciones.

Sistema de Recuperación

RAID, consideraciones adicionales

- ❑ Algunas implementaciones RAID permiten el **intercambio en caliente**; esto es, se pueden retirar los discos averiados y sustituirlos por otros nuevos sin desconectar la corriente. El intercambio en caliente reduce el tiempo medio de reparación, ya que los cambios de disco no necesitan esperar a que se pueda apagar el sistema.
- ❑ Además, muchas implementaciones RAID asignan un disco de repuesto para cada disposición (o para un conjunto de disposiciones de disco). Si falla algún disco, el disco de repuesto se utiliza como sustituto de manera inmediata. En consecuencia, el tiempo medio de reparación se reduce notablemente, lo que minimiza la posibilidad de pérdida de datos. El disco averiado se puede reemplazar cuando resulte más conveniente.

Sistema de Recuperación

Implementación del almacenamiento estable

Es necesario que, si se produce un fallo durante una transferencia de datos, el sistema lo detecte e invoque a un procedimiento de recuperación para restaurar el bloque a un estado estable. Para hacer esto, el sistema debe mantener dos bloques físicos por cada bloque lógico de la base de datos.

Una operación de salida se ejecuta de la siguiente manera:

1. Se escribe la información en el primer bloque físico
2. Cuando la primera escritura se completa con éxito, se escribe la misma información en el segundo bloque físico.
3. La salida está completada sólo después de que la segunda escritura finalice con éxito.

Sistema de Recuperación

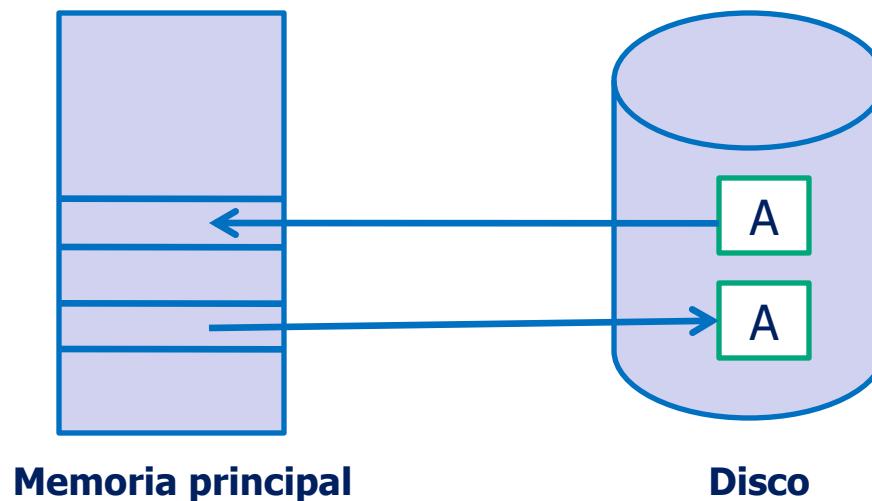
Acceso a los datos

Las transacciones llevan información del disco hacia la memoria principal y luego devuelven la información al disco.

Las operaciones de entrada y salida se realizan en unidades de bloque

Operaciones:

1. **entrada(B)** transfiere el bloque físico B a la memoria principal
2. **salida(B)** transfiere el bloque de memoria intermedia B al disco y reemplaza allí al correspondiente bloque físico.



Sistema de Recuperación

Acceso a los datos

Cada transacción T_i posee un área de trabajo privado en la cual se guardan copias de todos los elementos de datos accedidos y actualizados por T_i . Cada elemento de datos X almacenado en el área de trabajo de la transacción T_i se denominará como x_i .

Operaciones de transferencia de datos:

1. leer(X) asigna el valor del elemento de datos X a la variable local x_i .
 - a) Si el bloque B_x en el que reside X no está en la memoria principal, entonces se emite entrada(B_x).
b) Asignar a x_i el valor de X en el bloque de memoria intermedia.
2. escribir(X) asigna el valor de la variable local x_i al elemento de datos X en el bloque de memoria intermedia.
 - a) Si el bloque B_x en el que reside X no está en la memoria principal, entonces se lanza entrada(B_x).
b) Asignar el valor de x_i a X en la memoria intermedia B_x .

Sistema de Recuperación

RECUPERACION Y ATOMICIDAD

Saldos Iniciales

Cuenta A
S/. 1000

Cuenta B
S/. 2000

Supóngase que se desea transferir S/. 50 de la cuenta A a la B y que el sistema cae durante la ejecución de T_i .

- Después de ejecutarse **salida(B_A)**, pero antes de ejecutarse **salida (B_B)** (Donde B_A y B_B denotan los bloques de memoria intermedia en los que residen A y B.)

Estados inconsistentes con dos posibles procedimientos de recuperación

Volver a ejecutar T_i

Cuenta A
S/. 900

Cuenta B
S/. 2050

No volver a ejecutar T_i

Cuenta A
S/. 950

Cuenta B
S/. 2000

Sistema de Recuperación

RECUPERACIÓN BASADA EN EL REGISTRO HISTÓRICO

El registro histórico es una secuencia de registros que mantiene un registro de todas las actualizaciones de la base de datos.

Campos del registro de actualización del registro histórico:

- El identificador de la transacción
- El identificador del elemento de datos.
- El valor anterior
- El valor nuevo

<T_i, X_j, Valor anterior, Valor nuevo>

Sistema de Recuperación

RECUPERACIÓN BASADA EN EL REGISTRO HISTÓRICO

Tipos de registros del registro histórico:

- **<T_i, iniciada>**. La transacción T_i ha comenzado.
- **<T_i, X_j, V₁, V₂>**. La transacción T_i ha realizado una escritura sobre el elemento de datos X_j. X_j tenía el valor V₁ antes de la escritura y tendrá el valor V₂ después de la escritura.
- **<T_i, comprometida>**. La transacción T_i se ha comprometido.
- **<T_i, abortada>** La transacción T_i ha sido abortada.

Sistema de Recuperación

Modificación diferida de la base de datos

Garantiza la atomicidad mediante el almacenamiento de todas las modificaciones de la base de datos en el registro histórico, pero retardando la ejecución de todas las operaciones escribir de una transacción hasta que la transacción se compromete parcialmente.

Cuando una transacción se compromete parcialmente, la información del registro histórico asociada a esa transacción se utiliza para la ejecución de las escrituras diferidas.

Si el sistema cae antes de que la transacción complete su ejecución o si la transacción aborta, la información del registro histórico simplemente se ignora.

Sistema de Recuperación

Modificación diferida de la base de datos

El esquema de recuperación usa un procedimiento de recuperación:

- **rehacer(T_i)** fija el valor de todos los elementos de datos actualizados por la transacción T_i a los nuevos valores.

La operación **rehacer** debe ser idempotente, esto es, el resultado de ejecutarla varias veces debe ser equivalente al resultado de ejecutarla una sola vez.

Después de ocurrir un fallo, el subsistema de recuperación consulta el registro histórico para determinar las transacciones que deben rehacerse.

Una transacción T_i debe rehacerse si y sólo si el registro histórico contiene los registros **< T_i , iniciada>** y **< T_i , comprometida>**.

Sistema de Recuperación

Modificación diferida de la base de datos

Sea T_0 una transacción que transfiere S/. 50 desde la cuenta A a la cuenta B. Y sea T_1 una transacción que retira S/. 100 de la cuenta C.
(Saldos iniciales: A = 1000, B = 2000 y C = 700)

T_0 : leer(A)
A := A – 50
escribir(A)
leer(B)
B := B + 50
escribir(B)

T_1 : leer(C)
C := C – 100
escribir(C)

Registro histórico
< T_0 , iniciada>
< T_0 , A, 950>
< T_0 , B, 2050>
< T_0 , comprometida>

< T_1 , iniciada>
< T_1 , C, 600>
< T_1 , comprometida>

Registro histórico
< T_0 , iniciada>
< T_0 , A, 950>
< T_0 , B, 2050>
< T_0 , comprometida>

< T_1 , iniciada>
< T_1 , C, 600>
< T_1 , comprometida>

Base de datos

A = 950
B = 2050

C = 600

Sistema de Recuperación

Modificación diferida de la base de datos

Tres situaciones de caída del sistema:

T_0 : leer(A)
A := A - 50
escribir(A)
leer(B)
B := B + 50
escribir(B)

T_1 : leer(C)
C := C - 100
escribir(C)

T_0 : leer(A)
A := A - 50
escribir(A)
leer(B)
B := B + 50
escribir(B)

T_1 : leer(C)
C := C - 100
escribir(C)

T_0 : leer(A)
A := A - 50
escribir(A)
leer(B)
B := B + 50
escribir(B)

T_1 : leer(C)
C := C - 100
escribir(C)

< T_0 iniciada>
< T_0 , A, 950>
< T_0 , B, 2050>

< T_0 iniciada>
< T_0 , A, 950>
< T_0 , B, 2050>
< T_0 , comprometida>
< T_1 iniciada>
< T_1 , C, 600>

< T_0 iniciada>
< T_0 , A, 950>
< T_0 , B, 2050>
< T_0 , comprometida>
< T_1 iniciada>
< T_1 , C, 600>
< T_1 , comprometida>

Sistema de Recuperación

Modificación diferida de la base de datos

Tres situaciones de caída del sistema:



<T₀ iniciada>
<T₀, A, 950>
<T₀, B, 2050>

<T₀ iniciada>
<T₀, A, 950>
<T₀, B, 2050>
<T₀, comprometida>
<T₁ iniciada>
<T₁, C, 600>

<T₀ iniciada>
<T₀, A, 950>
<T₀, B, 2050>
<T₀, comprometida>
<T₁ iniciada>
<T₁, C, 600>
<T₁, comprometida>

No es necesario llevar a cabo ninguna acción rehacer , ya que no aparece el registro de comprometido.

**Se realiza la operación
rehacer(T₀)
ya que el registro
<T₀, comprometida>
Aparece en el registro
histórico.**

**Se realiza las operaciones
rehacer(T₀)
rehacer(T₁)
ya que los registros
<T₀, comprometida>
<T₁, comprometida>
Aparecen en el registro
histórico.**

Sistema de Recuperación

Modificación inmediata de la base de datos

Permite realizar la salida de las modificaciones de la base de datos a la propia base de datos mientras que la transacción está todavía en estado activo.

En caso de una caída o de un fallo en la transacción, el sistema debe utilizar el campo para el valor anterior de los registros del registro histórico.

El esquema de recuperación usa dos procedimientos de recuperación:

- **deshacer(T_i)** restaura el valor de todos los elementos de datos actualizados por la transacción T_i a los valores anteriores.
- **rehacer(T_i)** fija el valor de todos los elementos de datos actualizados por la transacción T_i a los nuevos valores.

Sistema de Recuperación

Modificación inmediata de la base de datos

Sea T_0 una transacción que transfiere S/. 50 desde la cuenta A a la cuenta B. Y sea T_1 una transacción que retira S/. 100 de la cuenta C.
(Saldos iniciales: A = 1000, B = 2000 y C = 700)

T ₀ : leer(A) A := A – 50 escribir(A) leer(B) B := B + 50 escribir(B)	Registro histórico <T ₀ iniciada> <T ₀ , A, 1000, 950> <T ₀ , B, 2000, 2050> <T ₀ , comprometida> <T ₁ iniciada> <T ₁ , C, 700, 600> <T ₁ , comprometida>	Registro histórico <T ₀ iniciada> <T ₀ , A, 1000, 950> <T ₀ , B, 2000, 2050> <T ₀ , comprometida> <T ₁ iniciada> <T ₁ , C, 600> <T ₁ , comprometida>	Base de datos A = 950 B = 2050 C = 600
---	---	--	---

Sistema de Recuperación

Modificación diferida de la base de datos

Tres situaciones de caída del sistema:

```
T0: leer(A)  
      A := A - 50  
      escribir(A)  
      leer(B)  
      B := B + 50  
      escribir(B)  
T1: leer(C)  
      C := C - 100  
      escribir(C)
```

```
T0: leer(A)  
      A := A - 50  
      escribir(A)  
      leer(B)  
      B := B + 50  
      escribir(B)  
T1: leer(C)  
      C := C - 100  
      escribir(C)
```

```
T0: leer(A)  
      A := A - 50  
      escribir(A)  
      leer(B)  
      B := B + 50  
      escribir(B)  
T1: leer(C)  
      C := C - 100  
      escribir(C)
```

```
<T0 iniciada>  
<T0, A, 1000, 950>  
<T0, B, 2000, 2050>
```

```
<T0 iniciada>  
<T0, A, 1000, 950>  
<T0, B, 2000, 2050>  
<T0, comprometida>  
<T1 iniciada>  
<T1, C, 700, 600>
```

```
<T0 iniciada>  
<T0, A, 1000, 950>  
<T0, B, 2000, 2050>  
<T0, comprometida>  
<T1 iniciada>  
<T1, C, 700, 600>  
<T1, comprometida>
```

Sistema de Recuperación

Modificación inmediata de la base de datos

Tres situaciones de caída del sistema:



```
<T0 iniciada>
<T0, A, 1000, 950>
<T0, B, 2000, 2050>
```

**Se realiza la operación
deshacer(T₀)**

```
<T0 iniciada>
<T0, A, 1000, 950>
<T0, B, 2000, 2050>
<T0, comprometida>
<T1 iniciada>
<T1, C, 700, 600>
```

**Se realiza la operación
rehacer(T₀)
ya que el registro
<T₀, comprometida>
Aparece en el registro
histórico.**

**Se realiza la operación
deshacer(T₁)**

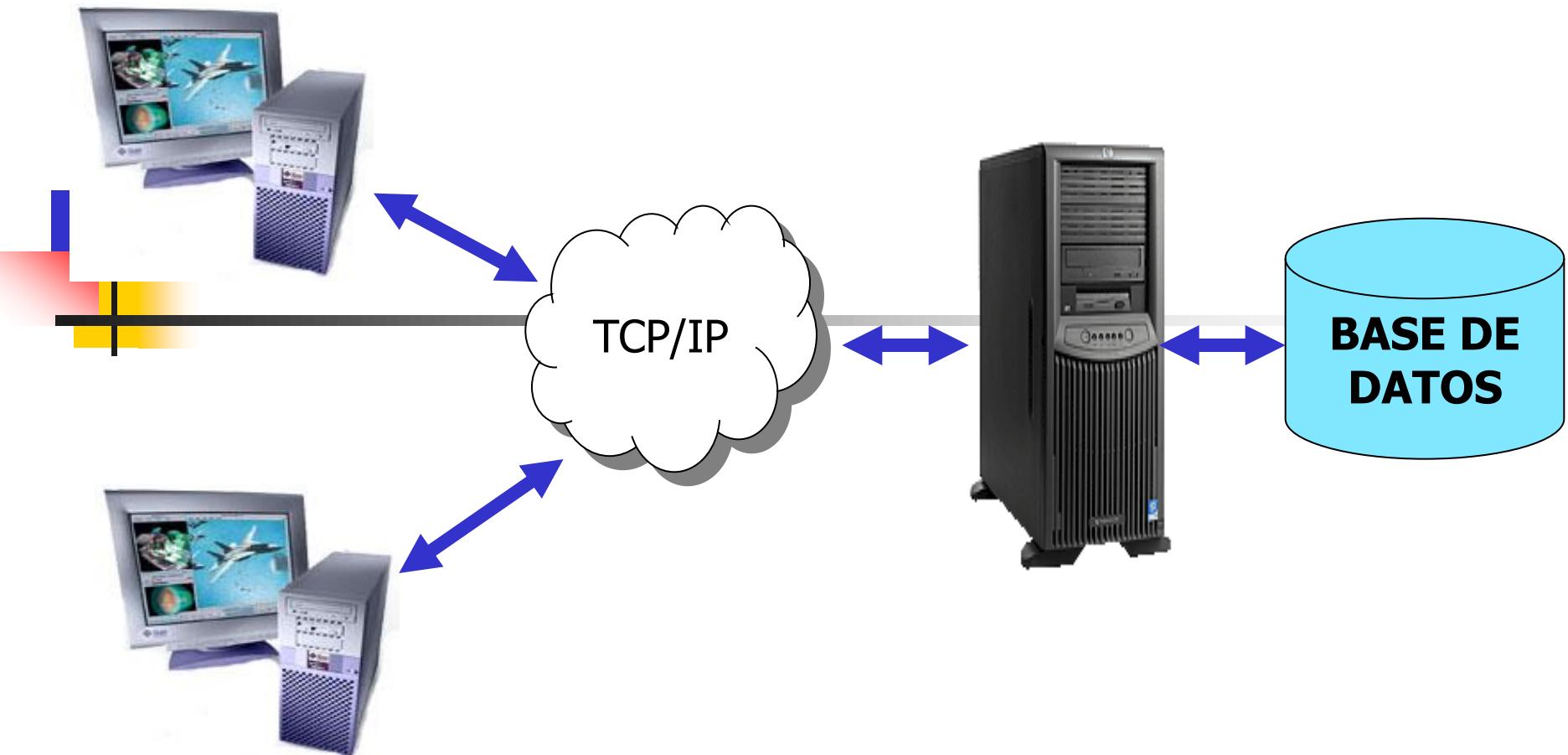
```
<T0 iniciada>
<T0, A, 1000, 950>
<T0, B, 2000, 2050>
<T0, comprometida>
<T1 iniciada>
<T1, C, 700, 600>
<T1, comprometida>
```

**Se realiza las operaciones
rehacer(T₀)
rehacer(T₁)
ya que los registros
<T₀, comprometida>
<T₁, comprometida>
Aparecen en el registro
histórico.**

Servicios de Bases de Datos

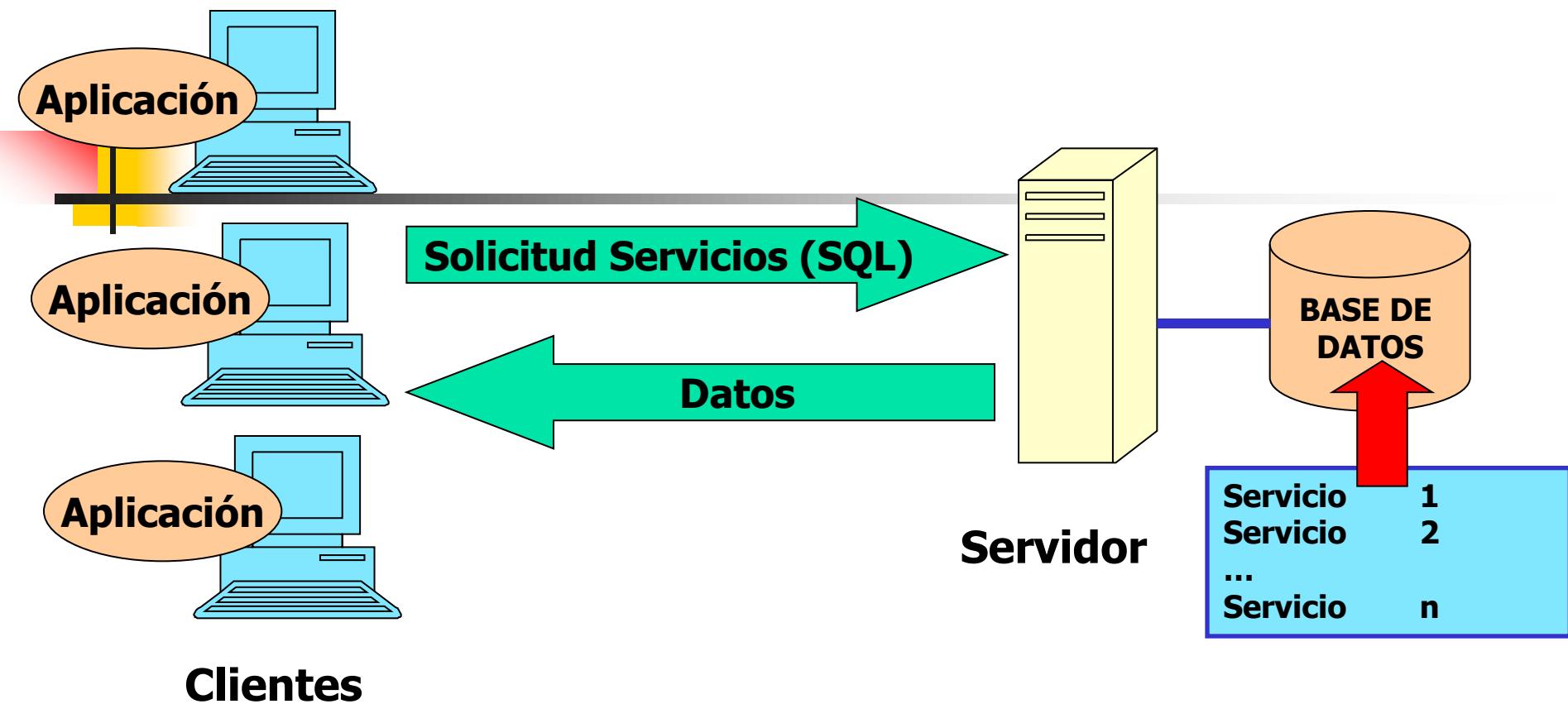
Funciones y procedimientos

Servicios de Bases de Datos



Servicios de Base de Datos

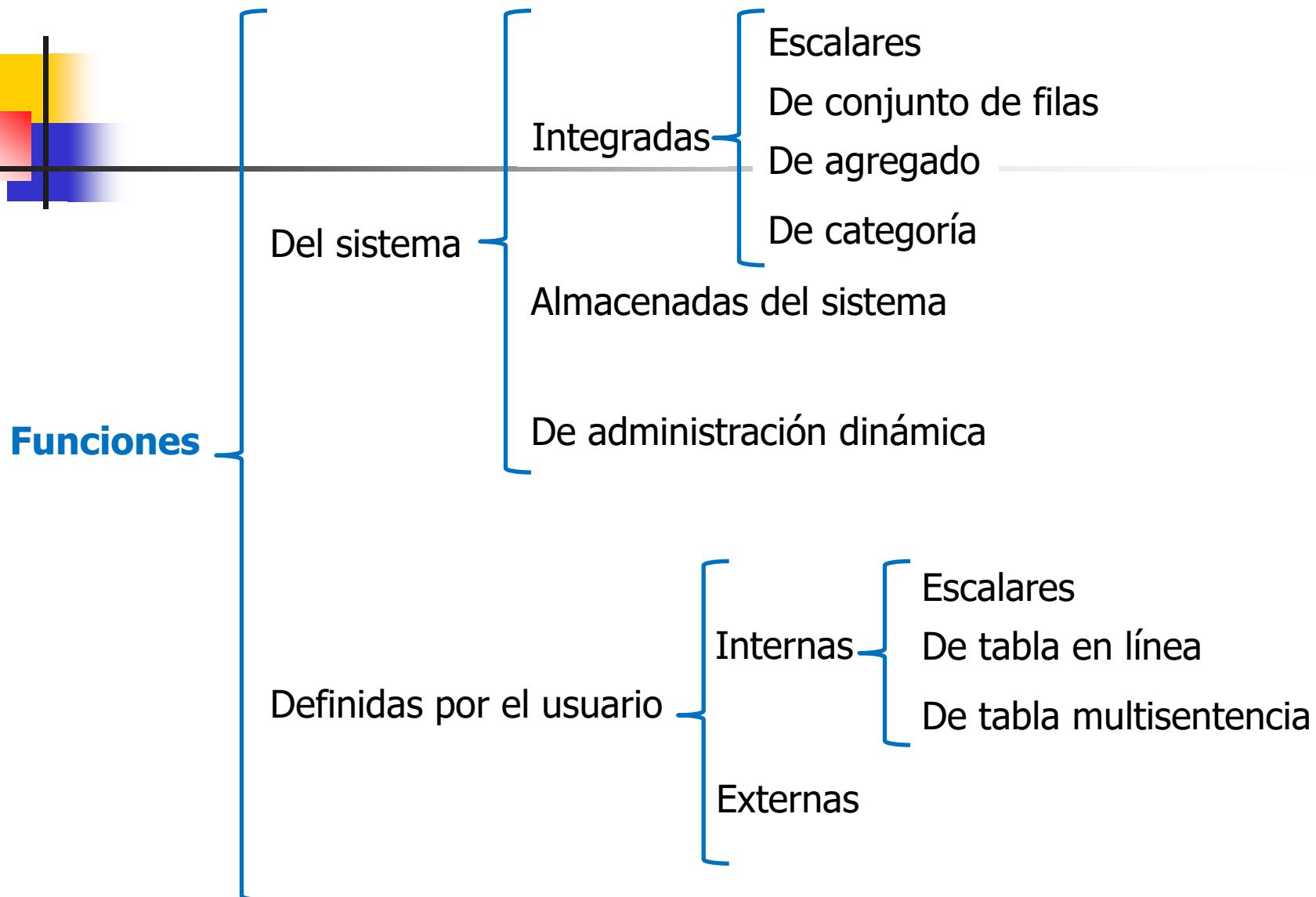
Los servicios de BD se almacenan en la base de datos y pueden ser funciones y procedimientos.



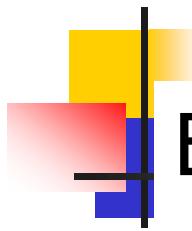
Servicios de BD

- Los servicios de BD pueden ser: **vistas, de tipo función o de tipo procedimiento (stored procedure)**
- Son un grupo de sentencias SQL que es compilado una vez, y luego puede ser ejecutado muchas veces.
- Las funciones y los procedimientos almacenados se almacenan en la base de datos y constituyen los servicios que ofrece la base de datos (Cliente/Servidor).
- Las funciones y los procedimientos almacenados cuando se ejecutan no tienen que ser recompilados cada vez, este hecho hace que sean muy eficientes.

Funciones en SQL SERVER



Funciones en SQL SERVER



Ejemplos de funciones:

De administración dinámica

```
select *  
from sys.dm_tran_active_transactions
```

Funciones internas definidas por el usuario

Se tienen tres tipos:

- **Escalares:** Retornan un tipo de los datos como int, money, varchar, real, etc. Pueden ser utilizadas dentro de sentencias SQL.
- **De tabla en Linea:** Retornan la salida de una simple declaración SELECT. La salida se puede utilizar dentro de composiciones o consultas como si fuera una tabla de estándar.
- **De multisentencia:** Dan como resultado una tabla, puede estar constituido por varias sentencias SQL, y puede llamar a otras funciones. La salida también se puede utilizar en operaciones de tabla.

Funciones Definidas por el Usuario... Funciones escalares:

```
/* *****
   Function que convierte una nota de tipo texto a número
***** */
CREATE FUNCTION fnNota (@NotaTexto varchar(3))
RETURNS int
AS
BEGIN
    declare @Nota int;
    set @Nota = case
        when @NotaTexto = 'NSP' then 0
        else Cast(@NotaTexto as int)
    end;
    return (@Nota);
END
```

Funciones Definidas por el Usuario... Funciones escalares:

```
/* ****
Function que convierte una nota a número
***** */
CREATE FUNCTION fnRecuperarNota (@Semestre varchar(7),
                                  @Cod_Asignatura varchar(5),
                                  @Cod_Alumno varchar(6))
RETURNS int
AS
BEGIN
    declare @Nota int;
    select @Nota = dbo.fnNota(Nota)
        from MATRICULA
       where (Semestre = @Semestre) and
             (Cod_Asignatura = @Cod_Asignatura) and
             (Cod_Alumno = @Cod_Alumno)
    return(@Nota)
END
```

Funciones Definidas por el Usuario... Funciones de tabla en Línea:

```
/* ****
Function que devuelve las asignaturas aprobadas
***** */
ALTER FUNCTION fnAsignaturasAprobadas (@Cod_CP varchar(2))
RETURNS TABLE
AS
RETURN (SELECT M.Semestre, M.Cod_CP, M.Cod_Alumno,
M.Cod_Asignatura,
A.Nombre_Asignatura, A.Creditos, A.Categoría,
Nota = dbo.fnNota(M.Nota)
FROM MATRICULA M inner join ASIGNATURA A
ON (M.Cod_Asignatura = A.Cod_Asignatura) and
(M.Cod_CP = A.Cod_CP)
WHERE ((M.Cod_CP = @Cod_CP) or (@Cod_Cp = '*')) and
(dbo.fnNota(M.Nota) > 10))
```

Funciones Definidas por el Usuario... Funciones de multisentencia:

```
/* ****
Function que devuelve los creditos acumulados semestre a
semestre de los alumnos de una carrera o en general
***** */
CREATE FUNCTION fnCreditosAcumuladosSemestre(@Cod_CP varchar(2))
RETURNS @taCreditosAcumuladosSemestre TABLE (Semestre varchar(7),
                                              Cod_Alumno varchar(6),
                                              Paterno varchar(15),
                                              Materno varchar(15),
                                              Nombres varchar(15),
                                              Cod_Cp varchar(2),
                                              Creditos int,
                                              Creditos_Acum int)
AS
BEGIN
    -- Crear tabla temporal para acumular creditaje
    declare @taCreditosAcumulados table(Semestre varchar(7),
                                          Cod_Alumno varchar(6),
                                          Creditos int,
                                          Creditos_Acum int)
```

Funciones Definidas por el Usuario...

Funciones de multisentencia:

```
-- Recuperar información a la tabla temporal
insert into @taCreditosAcumulados
select Semestre, Cod_Alumno, sum(Creditos), sum(Creditos)
from dbo.fnAsignaturasAprobadas (@Cod_CP)
group by Semestre, Cod_Alumno
order by Cod_Alumno, Semestre

-- Acumular los creditos por semestre
-- Declarar e inicializar variables
declare @Cod_Alumno varchar(6),
        @Creditos_Acum int;
set @Cod_Alumno = '';
set @Creditos_Acum = 0;
```

Funciones Definidas por el Usuario...

Funciones de multisentencia:

```
-- Acumular creditos
update @taCreditosAcumulados
    set @Creditos_Acum = Creditos_Acum = case
                                                when @Cod_Alumno <>
Cod_Alumno then Creditos
                                                else @Creditos_Acum +
Creditos
                                            end,
    @Cod_Alumno = Cod_Alumno;

-- Devolver resultado
INSERT INTO @taCreditosAcumuladosSemestre
SELECT C.Semestre, A.Cod_Alumno, A.Paterno, A.Materno, A.Nombres,
       A.Cod_CP, C.Creditos, C.Creditos_Acum
  from @taCreditosAcumulados C inner join ALUMNO A
    ON C.Cod_Alumno = A.Cod_Alumno
RETURN
END
```

Procedimientos Almacenados

Un procedimiento almacenado (stored procedure) es un grupo de sentencias SQL que es compilado una vez, y luego puede ser ejecutado muchas veces; se almacena en la base de datos y constituye los servicios que ofrece la base de datos (Cliente/Servidor).

Los procedimientos almacenados cuando se ejecutan no tienen que ser recompilados cada vez, este hecho hace que sean muy eficientes.

Procedimientos Almacenados...

- **Se escriben utilizando el lenguaje de programación de la base de Datos. Permiten declaración de variables, sentencias de asignación, estructuras selectivas y repetitivas.**
- **Aceptan parámetros de entrada y salida**
- **Pueden retornar conjunto de datos así como también un solo valor.**
- **Permiten la programación modular y puede hacer referencia a otros procedimientos almacenados, con lo que se puede simplificar una serie de instrucciones complejas.**
- **Pueden ser ejecutados desde otros procedimientos almacenados, vistas, disparadores y consultas en general.**

Procedimientos Almacenados...

Sintaxis :

```
CREATE PROCEDURE nombre procedimiento
    @parametros tipo de dato [= valor],
    @parametros_salida tipo de dato OUTPUT
AS
    [DECLARE @Variables tipos]
BEGIN
    sentencias sql
END
```

Procedimientos Almacenados...

Escribir procedimientos almacenados

Escribir un procedimiento para determinar la relación de alumnos de una determinada carrera

```
CREATE PROCEDURE spu_AlumnosPorCarrera  
    @Cod_Carrera varchar(2)  
AS  
BEGIN  
    -- Seleccionar alumnos de la carrera indicada  
    SELECT *  
        FROM ALUMNO  
        WHERE Cod_Carrera = @Cod_Carrera  
END
```

Constituye el código fuente, se debe almacenar como un archivo de extensión SQL.

Este código se debe compilar y almacenar en la base de datos, para que pueda ser invocado como un servicio.

Procedimientos Almacenados...

Ejecución de un procedimiento.

Sintaxis :

```
[DECLARE @variables tipo de variable]
EXEC/EXECUTE nombre_procedimiento [@parametros], [@variables]
```

Ejemplo de ejecución:

```
exec sp_AlumnosPorCarrera 'IN'
```

Procedimientos Almacenados...

Ejemplo: Generador de Stored procedures

```
IF OBJECT_ID ( 'spu_GeneradorStoredProcedure', 'P' ) IS NOT NULL
    DROP PROCEDURE spu_GeneradorStoredProcedure;
GO
CREATE PROCEDURE spu_GeneradorStoredProcedure @NomTabla varchar(256)
AS
begin
    -- Crear sentencia de parametros del procedimiento almacenado y la sentencia insert
    declare @TextoParametros varchar(8000);
    declare @TextoInsert varchar(8000);
    set @TextoParametros = "";
    set @TextoInsert = 'insert into '+ @NomTabla + ' values(';
```

Stored procedures. Generador de Stored procedures

-- Mediante un cursor construir dinámicamente ambos textos

-- Declarar el cursor

```
declare cu_Atributos cursor
```

```
for select c.name As Atributo, t.name as Tipo, c.max_length as Longitud, c.precision as  
Precision, c.scale as Decimales
```

```
from sys.columns c inner join sys.types AS t ON c.system_type_id=t.user_type_id
```

```
where Object_Id = (select Object_Id from sys.tables where Name = @NomTabla)
```

-- Declarar variables para utilizar en el cursor

```
declare @Atributo varchar(256), @Tipo varchar(24), @Longitud int, @Precision int,  
@Decimales int;
```

-- Abrir el cursor

```
open cu_Atributos
```

-- Recuperar el primer registro en las variables

```
fetch next from cu_Atributos into @Atributo, @Tipo, @Longitud, @Precision, @Decimales
```

-- Procesar cada atributo

```
declare @Separador varchar(1);
```

```
set @Separador = ";
```

Stored procedures. Generador de Stored procedures

```
while @@FETCH_STATUS = 0
begin
    -- Construir textos
    set @TextoParametros = @TextoParametros + @Separador + '@' + @Atributo + ' ' + @Tipo +
    case Upper(@Tipo)
        when 'VARCHAR' then '('+Cast(@Longitud as varchar(4))+)'
        when 'DATETIME' then "
        when 'INT' then "
        when 'NUMERIC' then '(' + Cast(@Precision as varchar(4))+','+Cast(@Decimales as varchar(4))+)'
        when 'DECIMAL' then '(' + Cast(@Precision as varchar(4))+','+Cast(@Decimales as varchar(4))+'
        else "
    end;

    set @TextoInsert = @TextoInsert + @Separador + '@' + @Atributo;
    set @Separador = ',';
    -- Recuperar siguiente registro
    fetch next from cu_Atributos into @Atributo, @Tipo, @Longitud, @Precision, @Decimales
end
-- Cerrar el cursor
close cu_Atributos
deallocate cu_Atributos
```

Stored procedures. Generador de Stored procedures

-- Completar TextoInsert

```
set @TextoInsert = @TextoInsert + ') ';
```

-- Declarar variable para generar sentencia SQL dinámicamente

```
declare @TextoSQL varchar(8000);
```

-- Borrar stored procedure si existiese

```
set @TextoSQL = 'if OBJECT_ID ( "spu_'+@NomTabla+'", "P" ) IS NOT NULL '+  
    'drop procedure spu_'+@NomTabla+'; ';
```

```
exec(@TextoSQL)
```

-- Generar stored procedure

```
set @TextoSQL = 'create procedure spu_ ' + @NomTabla + ' ' + @TextoParametros+' '+  
    'as '+  
    'begin '+
```

```
@TextoInsert+ ' '+
```

```
'end; ';
```

```
exec(@TextoSQL)
```

```
end;
```

```
GO
```

```
-- exec spu_GeneradorStoredProcedure 'Entrada_Detalle'
```

Sistemas de bases de datos

TRIGGERS (DISPARADORES)

Disparadores (Triggers)

¿Qué es un disparador?

Una clase especial de procedimiento almacenando (un grupo de sentencias SQL), que se ejecutan automáticamente cuando se produce un evento en el servidor de la base de datos.

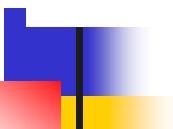
En SQL Server, los disparadores se pueden clasificar en:

- Disparadores DDL**
- Disparadores DML**
- Disparadores Log-on**

Disparadores (Triggers)

Disparadores DDL

Los disparadores DDL se ejecutan automáticamente cuando se modifican los metadatos, como consecuencia de la ejecución de una instrucción del lenguaje de definición de datos (DDL), instrucciones como: CREATE, ALTER o DROP.



Disparadores DML

Los disparadores DML se ejecutan automáticamente cuando se modifican los datos, como consecuencia de la ejecución de una instrucción del lenguaje de manipulación de datos (DML), instrucciones como: INSERT, UPDATE O DELETE.

Disparadores Log-on

Los disparadores Log-on se ejecutan automáticamente cuando se establece una sesión de un usuario.

Disparadores DDL

Se ejecutan automáticamente en respuesta a un variedad de eventos del Lenguaje de Definición de Datos, sobre todo en las instrucciones: CREATE, ALTER, DROP, GRANT, DENY o REVOKE.

Se aplican al servidor o a la base en la que han sido implementadas.

Se puede utilizar los disparadores DDL, por ejemplo en situaciones como:

- **Impedir determinados cambios en los metadatos o esquema de la base de datos.**
- **Efectuar algo en la base de datos como respuesta a un cambio realizado en los metadatos o esquema de la base de datos.**
- **Registrar cambios o eventos en el esquema de la base de datos.**

Disparadores DDL

Ejemplo: Implementar un disparador DDL, para evitar que se elimine o modifique accidentalmente alguna tabla de la Base de datos.

```
CREATE TRIGGER tr_ImpedirBorrarModificarTablas  
ON DATABASE  
FOR DROP_TABLE, ALTER_TABLE  
AS  
Begin  
    PRINT 'No está permitido eliminar o modificar la estructura de las tablas'  
    PRINT 'En caso de ser necesario, primero debe desactivar el disparador:'  
    PRINT 'tr_ImpedirBorrarModificarTablas'  
    ROLLBACK;  
End;  
GO
```

Disparadores DDL

Ejemplo: Implementar un disparador DDL, para evitar que se cree una tabla en la base de datos.

```
IF EXISTS (SELECT * FROM sys.server_triggers  
          WHERE name = 'tr_ProhibidoCrearTablas')  
DROP TRIGGER ddl_trig_database  
ON DATABASE;  
GO
```

```
CREATE TRIGGER tr_ProhibidoCrearTablas  
ON DATABASE  
FOR CREATE_TABLE  
AS  
begin  
    PRINT 'iii ERROR !!!'  
    PRINT 'No está permitido crear más tablas en esta base de datos.'  
    ROLLBACK  
end;  
GO
```

Disparadores DML

La información sobre un evento que activa un disparador DDL se captura mediante la función EVENTDATA. Esta función devuelve un valor **xml** . El esquema XML incluye información sobre lo siguiente:

- La hora del evento.
- El tipo de evento que activó el disparador.

Para recuperar la sentencia DDL (y la información asociada a ella) que activó el disparador, se hace uso de XQuery (del SQL Server) contra los datos xml que genera EVENTDATA y recuperando el elemento <CommandText>.

Disparadores DML

Tablas especiales **INSERTED** y **DELETED**

- Son tablas asociadas a los disparadores.
- Cuando se ejecuta una sentencia **INSERT**, **UPDATE** o **DELETE** se efectúa una copia de los datos insertados, modificados o eliminados en las tablas especiales **INSERTED** o **DELETED** según corresponda.
- Sólo se puede consultar el contenido de las tablas **INSERTED** y **DELETE**, no se puede modificar su contenido

Disparadores DML...

Tablas especiales INSERTED y DELETED

Cuando se ejecuta una sentencia	Tabla INSERTED	Tabla DELETED
INSERT	Contiene una copia del registro insertado	
UPDATE	Contiene una copia del registro modificado con los datos modificados	Contiene una copia del registro modificado con los datos anteriores a la modificación
DELETE		Contiene una copia del registro eliminado

Disparadores DML...

Utilidad de los disparadores

- **Implementación de referencias de Integridad
(Actualización o eliminación en cascada)**
- **Implementación de Reglas de Negocio (restricciones)**
- **Implementación de Históricos (Auditoría)**
- **Implementación de actualización de campos
calculados**
- **Implementación de ejecución de procesos especiales.
(Bases de datos activas)**

Disparadores DML...

Implementación de referencias de Integridad

Dadas las siguientes tablas

PRESTATARIO_RURAL (CodPrestatario, Nombres, Comunidad,
LimitePrestamo)

PRESTATARIO_CIUDAD (CodPrestatario, Nombres, Direccion,
Telefono, LimitePrestamo)

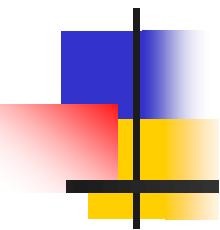
PRESTAMO (DocPrestamo, FechaPrestamo, ImportePrestamo,
FechaVcto, CodPrestatario)

CANCELACION (DocCancelacion, FechaCancelacion,
ImporteCancelacion, DocPrestamo)

Disparadores DML...

Implementación de referencias de Integridad...

Problema:



Al crear la tabla PRESTAMO se presenta un problema, el prestatario que efectúa el préstamo puede ser uno de la tabla PRESTATARIO_RURAL o de la tabla PRESTATARIO_CIUDAD, entonces ¿Cómo implementar una referencia de integridad del atributo *CodPrestatario* de modo que en su dominio sólo acepte un código de prestatario existente en una de las tablas de prestatario?

Disparadores DML...

Implementación de referencias de Integridad...

```
create trigger tr_Prestamo_ReferenciaIntegridad
  on PRESTAMO
  for INSERT
as
begin
  -- Recuperar el código del prestatario del registro insertado
  declare @CodPrestatario varchar(15);
  select @CodPrestatario = CodPrestatario
    from INSERTED;
  -- Verificar si CodPrestatario existe en tabla PRESTATARIO_RURAL.
  if not EXISTS (select CodPrestatario
                  from PRESTATARIO_RURAL
                  where CodPrestatario = @CodPrestatario)
  begin
    -- Verificar si CodPrestatario existe en tabla PRESTATARIO_CIUDAD.
    if not EXISTS (select CodPrestatario
                  from PRESTATARIO_CIUDAD
                  where CodPrestatario = @CodPrestatario)
      -- Deshacer o cancelar inserción en la tabla PRESTAMO
      ROLLBACK;
  end;
end;
```

Disparadores DML...

Operaciones en cascada de referencias de Integridad...

Cuando se modifica el código de un prestatario en la tabla PRESTATARIO_RURAL, lo correcto es que automáticamente todos los préstamos de este prestatario se actualicen al nuevo código del prestatario, para lo cual, se debe modificar el atributo CodPrestatario de la tabla PRESTAMO.

```
create trigger tr_Prestatario_Rural_ModificarCodPrestatario
  on PRESTATARIO_RURAL
  for UPDATE
as
begin
  -- Recuperar código de prestatario antes y después de la
  -- modificación.
  declare @CodPrestatarioAntes varchar(15);
  select @CodPrestatarioAntes = CodPrestatario
    from DELETED;
  declare @CodPrestatarioDespues varchar(15);
  select @CodPrestatarioDespues = CodPrestatario
    from INSERTED;
```

Disparadores DML...

Operaciones en cascada de referencias de Integridad...

```
-- Verificar si se modificó el código de prestatario
if @CodPrestatarioAntes <> @CodPrestatarioDespues
    -- Verificar si existen préstamos para @CodPrestatarioAntes
    if EXISTS (select CodPrestatario
                from PRESTAMO
                where CodPrestatario = @CodPrestatarioAntes)
        -- Existen prestamos, luego actualizar CodPrestatario
        UpDate PRESTAMO
            set CodPrestatario = @CodPrestatarioDespues
            where CodPrestatario = @CodPrestatarioAntes;
end;
```

Disparadores DML...

Implementación de restricciones

Los responsables de analizar las solicitudes de préstamo, determinan el monto máximo que se puede prestar a cada prestatario, considerando la capacidad de pago que puedan tener éstos. Entonces, para este propósito se tiene un atributo *LmitePrestamo* en ambas tablas de prestatarios.

```
create trigger tr_Prestamo_Restriccion_LimitePrestamo
  on PRESTAMO
  for INSERT
as
begin
  -- Recuperar el código del prestatario y el importe insertado
  declare @CodPrestatario varchar(15);
  declare @ImportePrestamo numeric(15,2);
  select @CodPrestatario = CodPrestatario,
         @ImportePrestamo = ImportePrestamo
  from INSERTED;
```

Disparadores DML...

Implementación de restricciones

```
-- Recuperar "LimitePrestamo" del prestatario @CodPrestatario,
-- haciendo uso del procedimiento almacenado sp_LimitePrestamo
declare @LimitePrestamo numeric(15,2);
exec spu_LimitePrestamo @CodPrestatario, @LimitePrestamo OUTPUT;
-- Verificar si importe de préstamo es permitido o no
if @ImportePrestamo > @LimitePrestamo
    -- Deshacer o cancelar el préstamo
    ROLLBACK;
end;
```

En este disparador se hace uso de un “stored procedure” para determinar el Límite de préstamo del prestatario.

Disparadores DML...

Implementación de restricciones...

Procedimiento almacenado para determinar el límite de crédito de un prestatario.

```
create procedure spu_LimitePrestamo
    @CodPrestatario varchar(15),
    @LimitePrestamo Numeric(15,2) OUTPUT
as
begin
    -- Inicializar @LimitePrestamo
    set @LimitePrestamo = 0;
    -- Recuperar Límite de préstamo asumiendo que @CodPrestatario
    -- pertenece a la tabla PRESTATARIO_RURAL
    select @LimitePrestamo = LimitePrestamo
        from PRESTATARIO_RURAL
        where CodPrestatario = @CodPrestatario;
    -- Si @LimitePrestamo = 0, entonces @CodPrestatario posiblemente
    -- pertenece a la tabla PRESTATARIO_CIUDAD
    if @LimitePrestamo = 0
        select @LimitePrestamo = LimitePrestamo
            from PRESTATARIO_CIUDAD
            where CodPrestatario = @CodPrestatario;
end;
```

Disparadores DML...

Implementación de campos calculados

El stock de un artículo se calcula realizando una suma de todas las entradas menos la suma de todas las salidas de dicho artículo. Este cálculo puede implicar un proceso reiterativo y mucho tiempo que podría ralentizar la aplicación. Para subsanar este inconveniente, se puede incluir un campo calculado (Stock), el cual se actualizará mediante disparadores. Considerando la siguiente base de datos, implementar los disparadores que actualicen el campo "Stock"

Tablas de la base de datos de Almacenes

ARTICULO (CodArticulo, Descripcion, UnidadMedida, Stock)

A0123	Fierro	PZA	80
A0132	Calamina	PZA	0

ENTRADA (DocEntrada, Fecha, Proveedor)

ENTRADA_DETALLE (DocEntrada, CodArticulo, Cantidad, PrecioUnitario)

FC-2345	A0123	100	21
---------	-------	-----	----

SALIDA (DocSalidada, Fecha, Cliente, Vendedor)

SALIDA_DETALLE (DocSalida, CodArticulo, Cantidad, PrecioUnitario)

Disparadores DML...

Implementación de Campos calculados...

```
create trigger tr_EEntrada_Detalle_ActualizarStock
  on Entrada_Detalle
  for INSERT
as
begin
  -- Recuperar el código y cantidad del artículo
  declare @CodArticulo varchar(15);
  declare @Cantidad numeric(15,4);
  select @CodArticulo = CodArticulo,
         @Cantidad = Cantidad
    from INSERTED;
  -- Actualizar en la tabla de artículos
  update ARTICULO
    set Stock = Stock + @Cantidad
   where @CodArticulo = CodArticulo
end;
```

Disparadores DML...

Implementación de campos calculados

```
create trigger tr_Actualizacion_Stock
  on ENTRADA_DETALLE
  for INSERT, UPDATE, DELETE
as
Begin
  declare @CodArticulo varchar(12);
  declare @Cantidad numeric(15,2);
  -- Revertir actualización debido a la operación UPDATE o DELETE, con valores anteriores
if Exists(select * from DELETED)
begin
  -- Recuperar los anteriores valores de código del artículo y cantidad
  select @CodArticulo = CodArticulo, @Cantidad = Cantidad
    from DELETED;
  -- revertir el valor de stock
  update ARTICULO
    set Stock = Stock - @Cantidad
      where CodArticulo = @CodArticulo;
end;
  -- Actualizar Stock debido a la operación INSERT o UPDATE, con los nuevos valores
if Exists(select * from INSERTED)
begin
  -- Recuperar los valores de código del artículo y cantidad
  select @CodArticulo = CodArticulo, @Cantidad = Cantidad
    from INSERTED;
  -- Actualizar Stock con los nuevos valores
  update ARTICULO
    set Stock = Stock + @Cantidad
      where CodArticulo = @CodArticulo;
end
end;
```

Disparadores DML...

Implementación de campos calculados...

```
create trigger tr_Actualizacion_Stock
  on SALIDA_DETALLE
  for INSERT, UPDATE, DELETE
as
Begin
  declare @CodArticulo varchar(12);
  declare @Cantidad numeric(15,2);
  -- Revertir actualización debido a la operación UPDATE o DELETE, con valores anteriores
if Exists(select * from DELETED)
begin
  -- Recuperar los anteriores valores de código del artículo y cantidad
  select @CodArticulo = CodArticulo, @Cantidad = Cantidad
    from DELETED;
  -- revertir el valor de stock
  update ARTICULO
    set Stock = Stock + @Cantidad
      where CodArticulo = @CodArticulo;
end;
  -- Actualizar Stock debido a la operación INSERT o UPDATE, con los nuevos valores
if Exists(select * from INSERTED)
begin
  -- Recuperar los valores de código del artículo y cantidad
  select @CodArticulo = CodArticulo, @Cantidad = Cantidad
    from INSERTED;
  -- Actualizar Stock con los nuevos valores
  update ARTICULO
    set Stock = Stock - @Cantidad
      where CodArticulo = @CodArticulo;
end
end;
```

Modelo de datos: No SQL

MongoDB

Modelo de datos NoSQL

Las bases de datos NoSQL son modelos de almacenamiento de información que no cumplen con el esquema relacional. Esto es, que no utilizan una estructura de datos en forma de relación o tabla para almacenar los datos; sino, que para el almacenamiento hacen uso de otros tipos de estructuras.



Ventajas del modelo de datos NoSQL

- ❑ **Aplicaciones de big data:** Permite manejar grandes volúmenes de datos.
- ❑ **Administración de la base de datos:** Requieren menos administración práctica, cuenta con capacidades de distribución de datos y reparación automática.
- ❑ **Versatilidad:** Se adapta fácilmente a un entorno de alto dinamismo; tanto al crecimiento en el volumen de datos o la posibilidad de incluir cambios en la estructura de los datos
- ❑ **Crecimiento Horizontal:** Son altamente escalables, se puede instalar más nodos, sin interrumpir la usabilidad.
- ❑ **Economía:** La adaptabilidad y flexibilidad permiten empezar con bajos niveles de inversión en equipos e ir ampliando la capacidad a medida de las necesidades.
- ❑ **Open source:** tecnología disponible como open source y libre de costo en muchos casos
- ❑ **Sprints:** puede acomodarse a iteraciones rápidas de sprints y entregas de código



Desventajas del modelo de datos NoSQL

- Menos maduro:** Hay muchas características importantes que aún no se han implementado
- Atomicidad:** Algunas de estas bases de datos no consideran la atomicidad. Puede implicar que la información no sea consistente entre nodos.
- Software poco documentado:** En algunos casos, Información escasa sobre las herramientas y sus características.
- Baja estandarización:** El lenguaje tiende a variar según el tipo de base de datos que se vaya a utilizar.
- Herramientas GUI:** La mayoría de los gestores no tienen una interfaz gráfica.
- Automatización:** No todas son buenas para automatizar la fragmentación o para distribuir la base de datos en varios nodos.
- Transacciones:** No son compatibles con las transacciones ACID.



Características adicionales del modelo de datos NoSQL

- Se ejecutan en máquinas con pocos recursos: Estos sistemas, a diferencia de los sistemas basados en SQL, no requieren de apenas computación, por lo que se pueden montar en máquinas de un coste más reducido.
- Pueden manejar gran cantidad de datos: Esto es debido a que utiliza una estructura distribuida, en muchos casos mediante tablas Hash.
- Escalabilidad horizontal: Para mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos, con la única operación de indicar al sistema cuáles son los nodos que están disponibles.
- No genera cuellos de botella: El principal problema de los sistemas SQL es que necesitan transcribir cada sentencia para poder ser ejecutada, y cada sentencia compleja requiere además de un nivel de ejecución aún más complejo, lo que constituye un punto de entrada en común, que ante muchas peticiones puede ralentizar el sistema.



Modelos de datos NoSQL

Es complicado categorizar los modelos de datos NOSQL, pues existe una diversidad de modelos y gestores implementados en estos modelos.

Sin embargo, de manera general, se puede indicar las siguientes:

- Bases de datos clave-valor: Riak, Redis, Dynamo, Voldemort.
- Bases de datos orientadas a documento: MongoDB, CouchDB.
- Bases de datos basadas en columnas: Cassandra, Hypertable, HBase, SimpleDB
- Bases de datos de grafos: Neo4J, Infinite Graph.



Para una clasificación más detalle, remitirse a <http://nosql-database.org>

MongoDB

MongoDB es un **gestor de bases de datos NoSQL orientada a documentos de código abierto**, creada por la compañía 10gen en el año. 2007. El nombre mongoDB viene del inglés (humongous, "gigantesco"). Es un software **escrito en C++**; los datos se almacenan como colecciones de documentos en formato JSON con un esquema dinámico denominado BSON.

Un aspecto relevante de MongoDB es que al ser un proyecto de código abierto, sus ejecutables están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris.}

Es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.



Para una clasificación más detalle, remitirse a <http://nosql-database.org>

MongoDB: JSON y BSON

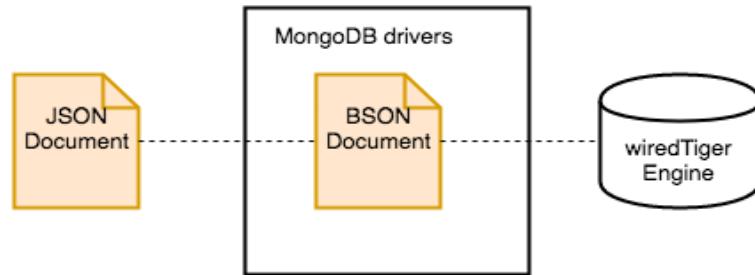
JSON

JSON (Notación de Objetos de JavaScript), es un formato de intercambio de datos legible por humanos, especificado a principios de la década de 2000.

Se basa en un subconjunto del estándar del lenguaje de programación JavaScript, es completamente independiente del lenguaje.

Los objetos JSON son contenedores asociativos, en los que una clave de cadena se asocia a un valor (número, texto, booleano, matriz, un valor nulo o incluso otro objeto).

```
{  
  "nombre": "Juan",  
  "edad": 25  
  "dirección":  
    {  
      "ciudad": "Barcelona"  
    },  
  "aficiones": [  
    {"nombre": "Fútbol"},  
    {"nombre": "Esquí"}  
  ]  
}
```

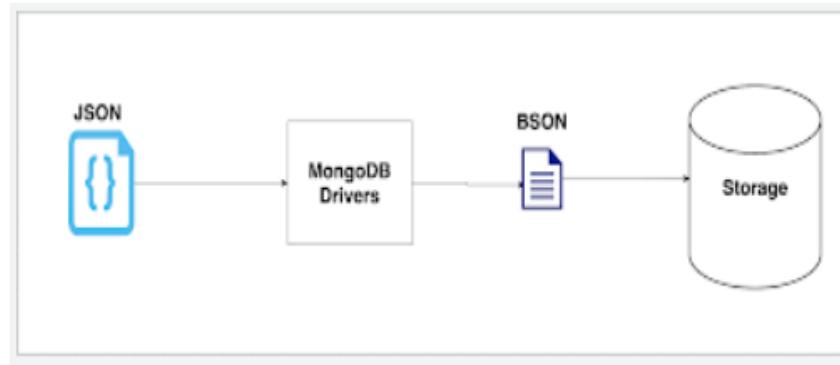


BSON

Para que MongoDB JSON sea de alto rendimiento y propósito general, se inventó BSON, una representación binaria para almacenar datos en formato JSON, optimizada para velocidad, espacio y eficiencia.

BSON significa "JSON binario". Corresponde a la estructura binaria de JSON. BSON codifica información de tipo y longitud, lo que permite recorrerla mucho más rápido en comparación con JSON.

BSON agrega algunos tipos de datos que no son nativos de JSON, como fechas y datos binarios, sin los cuales a MongoDB le habría faltado un soporte valioso.



```
\x16\x00\x00\x00  
\x02  
name\x00  
\x06\x00\x00\x00Devang\x00  
\x00
```

MongoDB

Comparativa de MongoDB con SQL

MongoDB

Base de Datos

Colecciones

Documentos

Campos

SQL

Base de Datos

Tablas

Filas (registros)

Columnas

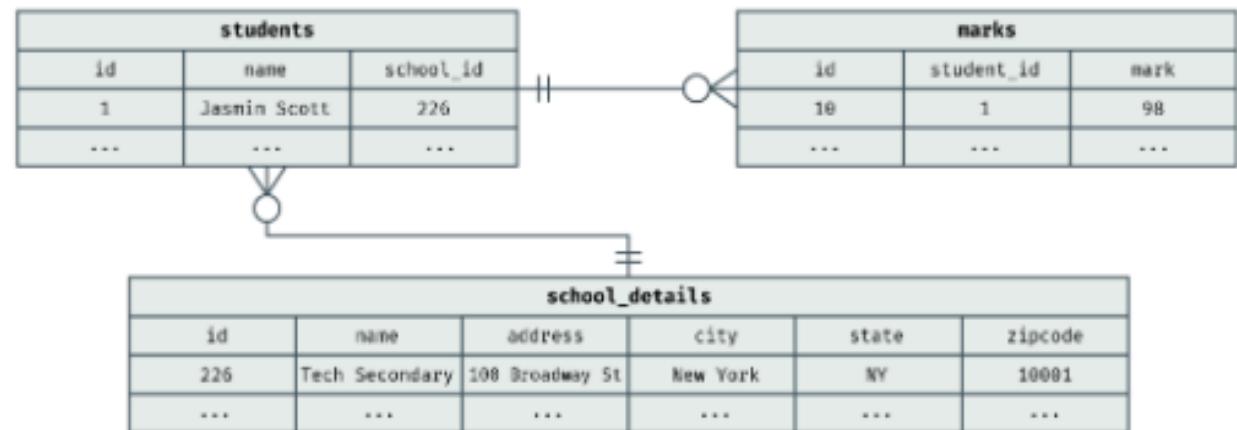
MongoDB

Comparativa de MongoDB con SQL

MongoDB

```
{  
  "_id": 1,  
  "student_name": "Jasmin Scott",  
  "school": {  
    "school_id": 226,  
    "name": "Tech Secondary",  
    "address": "100 Broadway St",  
    "city": "New York",  
    "state": "NY",  
    "zipcode": "10001"  
  },  
  "marks": [98, 93, 95, 88, 100],  
}
```

SQL



Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...

MongoDB

Comparativa de MongoDB con SQL

mongo

```
> db.students.find({"student_name":  
    "Jasmin Scott"})
```

sql

```
SELECT s.name, m.mark, d.name as "school name",  
d.city  
FROM students s  
INNER JOIN marks m ON s.id = m.student_id  
INNER JOIN school_details d ON s.school_id = d.id  
WHERE s.name = "Jasmin Scott";
```

MongoDB

Comparativa de MongoDB con SQL

```
db.users.insert ( ← collection
  {
    name: "sue", ← field: value
    age: 26, ← field: value
    status: "A" ← field: value
  }
)
```

```
INSERT INTO users ← table
  ( name, age, status ) ← columns
VALUES ( "sue", 26, "A" ) ← values/row
```

```
db.users.update(
  { age: { $gt: 18 } }, ← collection
  { $set: { status: "A" } }, ← update criteria
  { multi: true } ← update action
)
```

```
UPDATE users ← table
SET status = 'A' ← update action
WHERE age > 18 ← update criteria
```

```
db.users.remove(
  { status: "D" } ← collection
)
```

```
DELETE FROM users ← table
WHERE status = 'D' ← delete criteria
```

Por qué MongoDB es rápido

Documento de PAZ GUERRA ANGEL

```
db.clientes.insertOne(
{
  _id: 1,
  razon_social: 'PAZ GUERRA ANGEL',
  ruc: '12457898',
  email: 'paz.guerra@gmail.com',
  cuentas:
  [
    {
      _id : "CA125",
      tipo : "AHORRO",
      moneda : "PEN",
      movimientos :
      [
        {
          _id : 1457,
          documento : "RC1234",
          tipo : "DEPOSITO",
          importe : 2750.00
        },
        {
          _id : 3469,
          documento : "V2318",
          tipo : "RETIRO",
          importe : 560.00
        }
      ]
    },
    {
      _id : "CC873",
      tipo : "CTA CTE",
      moneda : "PEN",
      movimientos :
      [
        {
          _id : 2387,
          documento : "RC4564",
          tipo : "DEPOSITO",
          importe : 13450.00
        },
        {
          _id : 7461,
          documento : "V2398",
          tipo : "RETIRO",
          importe : 2000.00
        }
      ]
    }
  ]
},
```

Documento de LAZO PINO MELINA

```
db.clientes.insertOne(
{
  _id : 2,
  razon_social: 'LAZO PINO MELINA',
  ruc: '78894556',
  email: 'lazo.pino@gmail.com',
  cuentas:
  [
    {
      _id : "CA322",
      tipo : "AHORRO",
      moneda : "PEN",
      movimientos :
      [
        {
          _id : 2247,
          documento : "RC4413",
          tipo : "DEPOSITO",
          importe : 5000.00
        },
        {
          _id : 1536,
          documento : "V1878",
          tipo : "RETIRO",
          importe : 1600.00
        }
      ]
    },
    {
      _id : "CA323",
      tipo : "AHORRO",
      moneda : "USD",
      movimientos :
      [
        {
          _id : 543,
          documento : "RC117",
          tipo : "DEPOSITO",
          importe : 3000.00
        },
        {
          _id : 615,
          documento : "V878",
          tipo : "RETIRO",
          importe : 500.00
        }
      ]
    }
  ]
},
```

Todos los datos de un cliente, es decir, las cuentas y los movimientos de cada cuenta, se almacena en un único documento; Por tanto, al recuperar los datos de un cliente, se recuperan toda su información, sin necesidad de efectuar “join’s”

MongoDB

Estructura de MongoDB



La naturaleza no estructurada de MongoDB, permite que el schema o número de campos por documento puede cambiar, dando mucha más flexibilidad a la hora de diseñar la base de datos.

MongoDB

Formato JSON

JavaScript Object Notation (JSON), es un formato para guardar e intercambiar información de fácil lectura; contiene solo texto y usan la extensión **.json**.

Un objeto JSON comienza y termina con llaves **{}**. Está constituido por pares de **key:value** (clave:valor) separados por comas.

- Una **key** es un texto que va entre comillas.
- El **Value** es un tipo de datos JSON válido. Puede ser un dato simple (número, texto, booleano), un arreglo (array) u otro objeto.

```
{ _id: '40125832',
  nombre: ['Ana', 'Guerra', 'Paz'],
  edad: 28,
  genero: 'femenino',
  gustos: ['música', 'lectura'],
  datos_generales: function () {
    print(this.nombre[0] +' '+this.nombre[1]+ ' '+this.nombre[2]+ '\nEdad '+this.edad+ ' años\nGustos:\n- '+this.gustos[0]+ '\n- '+ this.gustos[1]);
  },
  saludo: function(mensaje) {
    print('Hola, Soy '+ this.nombre[0] + '. '+mensaje);
  }
};
```

MongoDB

CRUD: Create, Read, Update y Delete

CRUD EN MONGODB

Operación	Comandos	Ejemplo
CREATE	<code>insertOne</code> <code>insertMany</code>	<code>db.escuela.insertOne({_id:'CC',nombre:'Ciencia de la computación'})</code>
READ	<code>find</code> <code>aggregate</code>	<code>db.escuelas.find()</code>
UPDATE	<code>updateOne</code> <code>updateMany</code> <code>replaceOne</code>	<code>db.escuelas.updateOne({_id : {\$eq:"CC}}, {\$set : {nombre:'Ciencias de la computación'}})</code>
DELETE	<code>deleteOne</code> <code>deleteMany</code>	<code>db.escuelas.deleteOne({_id: { \$eq : 'CC'}})</code>

Modelo de datos: NewSQL

MySQL

Modelo de datos NewSQL

NewSQL es un tipo de lenguaje de base de datos que incorpora y se basa en los conceptos y principios del lenguaje de consulta estructurado (SQL) y los lenguajes NoSQL. Al combinar la confiabilidad de SQL con la velocidad y el rendimiento de NoSQL, NewSQL proporciona una funcionalidad y servicios mejorados.



Ventajas del modelo de datos NewSQL

- NewSQL funciona bien para aplicaciones que tienen muchas transacciones cortas que acceden a una pequeña cantidad de datos indexados y se ejecutan de forma repetitiva.
- NewSQL usa SQL, el lenguaje de acceso a datos más común y, por lo tanto, brinda familiaridad a los desarrolladores y programadores.
- NewSQL admite la consistencia de transacciones ACID, lo que proporciona una mejor integridad de datos que la mayoría de las alternativas de NoSQL.
- NewSQL puede reducir la complejidad de las aplicaciones que no requieren todas las comodidades de un DBMS SQL tradicional.

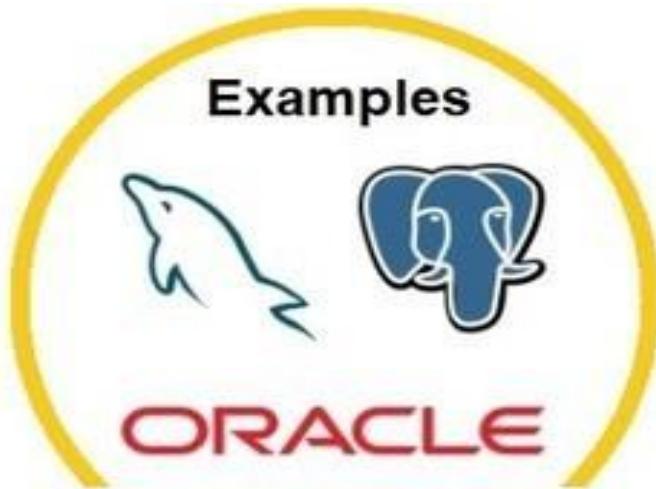


Desventajas del modelo de datos NewSQL

- NewSQL tiene menos funciones que el SQL tradicional y, por lo tanto, NewSQL no es tan capaz para la implementación de propósito general.
- La falta de herramientas para apoyar el desarrollo y la administración en comparación con los sistemas de bases de datos SQL tradicionales.
- Hay menos profesionales capacitados disponibles debido a la nueva tecnología de NewSQL.

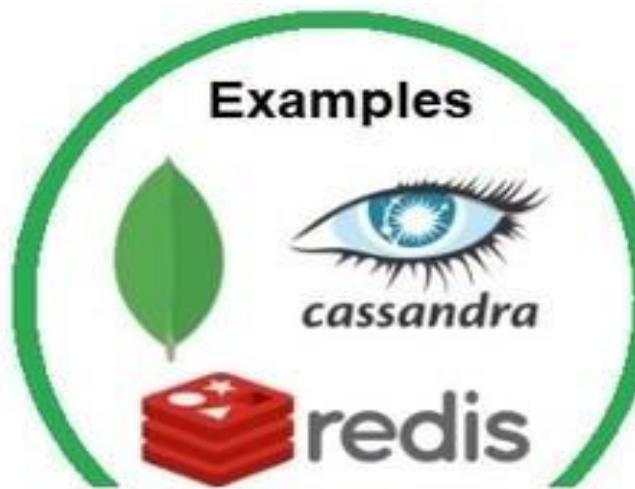


Comparativa de modelos de datos



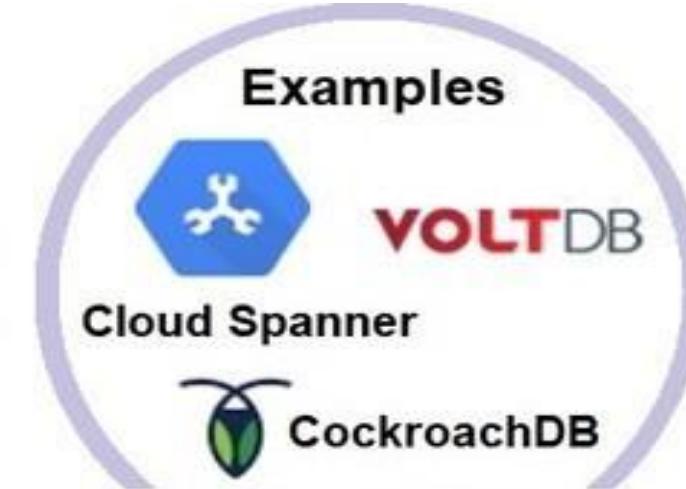
- ✓ ACID transactions
- ✓ SQL support
- ✓ Standardized
- ✗ Horizontal Scaling
- ✗ High Availability

RDBMS (SQL)



- ✓ Horizontal Scaling
- ✓ High Availability
- ✗ ACID transactions
- ✗ SQL support
- ✗ Standardized

NoSQL



- ✓ ACID transactions
- ✓ Horizontal Scaling
- ✓ High Availability
- ✓ SQL support
- ✗ Standardized

NewSQL

MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto ampliamente utilizado. Fue originalmente desarrollado por una empresa sueca llamada MySQL AB, fundada por Michael Widenius, David Axmark y Allan Larsson en 1995. Posteriormente, en 2008, Sun Microsystems adquirió MySQL AB. Finalmente, en 2010, Oracle Corporation adquirió Sun Microsystems, lo que llevó a que MySQL pasara a formar parte de Oracle. Es ampliamente utilizado en aplicaciones web y es una opción popular para desarrolladores y empresas que buscan una solución de base de datos confiable y escalable.

Características principales de MySQL:

- Relacional (Multi-modelo)
- Código abierto
- Multiplataforma
- Escalabilidad
- Veloz y eficiente
- Seguridad
- Soporte para el lenguaje SQL



MySQL

Tipos de datos

1. Numérico

TINYINT
SMALLINT
MEDIUMINT
INT
BIGINT
DECIMAL
FLOAT
DOUBLE

2. Cadena

CHAR
VARCHAR
TINYTEXT
TEXT
MEDIUMTEXT
LONGTEXT
BLOB
MEDIUMBLOB
LONGBLOB
ENUM
SET

3. Fecha y marca temporal

DATE
DATETIME
TIMESTAMP
TIME
YEAR

4. Espacial (*Especificación OpenGIS*)

GEOMETRY
POINT
LINESTRING
POLYGON
MULTIPOINT
MULTILINESTRING
MULTIPOLYGON
GEOMETRYCOLLECTION

5. JSON (*JavaScript Object Notation*)

MySQL

Funciones del Tipo de datos JSON

- [JSON Function Reference](#)
- [Functions That Create JSON Values](#)
- [Functions That Search JSON Values](#)
- [Functions That Modify JSON Values](#)
- [Functions That Return JSON Value](#)
- [Attributes](#)
- [JSON Table Functions](#)
- [JSON Schema Validation Functions](#)
- [JSON Utility Functions](#)



MySQL

Bases de datos con atributos estructurados y no estructurados

Alternativa 1.- Una tabla con atributos adicionales para la parte no estructurada (Esquema rígido)

ARTÍCULOS											
Código	Descripción	Unidad Medida	Precio Unitario	Disco	Marca	Modelo	Memoria	Pantalla	Tipo	Fecha Vcto	Gramaje
101	Computador	UNI	1500	1TB	HP	XYZ	12GB				
121	Laptop	UNI	2700	1TB	LENOVO	ZetaXYZ	12GB	12"			
142	Impresora	UNI	780		EPSON				Laser		
154	Tinta	PZA	120		EPSON					31/12/2023	
155	Papel	MIL	20								60Gr

Alternativa 2.- Dos tablas: para la parte estructurada y la parte no estructurada (Diseño vertical)

ARTÍCULOS			
Código	Descripción	Unidad Medida	Precio Unitario
101	Computador	UNI	1500
121	Laptop	UNI	2700
142	Impresora	UNI	780
154	Tinta	PZA	120
155	Papel	MIL	20

ARTÍCULOS_ATRIBUTOS		
Código	Atributo	Valor
101	Disco	1TB
101	Marca	HP
101	Modelo	XYZ
101	Memoria	12GB
121	Disco	1TB
121	Marca	LENOVO
121	Modelo	ZetaXYZ
121	Memoria	12GB
121	Pantalla	12"
142	Marca	EPSON
142	Tipo	Laser
154	Marca	EPSON
154	Fecha Vcto	31/12/2023
155	Gramaje	60gr

MySQL

Bases de datos con atributos estructurados y no estructurados

Alternativa 3.- Una tabla con columnas de tipo JSON

ARTÍCULO				
Código	Descripción	Unidad Medida	Precio Unitario	Datos
101	Computador	UNI	1500	{"HD": "1TB", "Marca": "HP", "Modelo": "XYZ", "Memoria": "12GB"}
121	Laptop	UNI	2700	{"HD": "1TB", "Marca": "LENOVO", "Modelo": "ZetaXYZ", "Memoria": "12GB", "Pantalla": "12"}
142	Impresora	UNI	780	{"Tipo": "Laser", "Marca": "EPSON", "Modelo": "L900"}
154	Tinta	PZA	120	{"Marca": "EPSON", "FechaVcto": "31/12/2023"}
155	Papel	MIL	20	{"Gramaje": "60gr"}



BUSINESS INTELLIGENCE

Mgt. Javier Arturo Rozas Huacho

Toma de Decisiones

Existen dos formas de tomar decisiones

Decisiones lentas, basadas sólo en la experiencia, prediciendo el futuro en base a la intuición.



Decisiones rápidas y oportunas, tomadas sobre cimientos firmes, prediciendo el futuro en base a información del pasado



Sistemas de Información

Sistemas de información de gestión



Sistemas de soporte
a la toma de
decisiones



Facturación,
planillas, Logística,
Finanzas, matriculas,
etc.



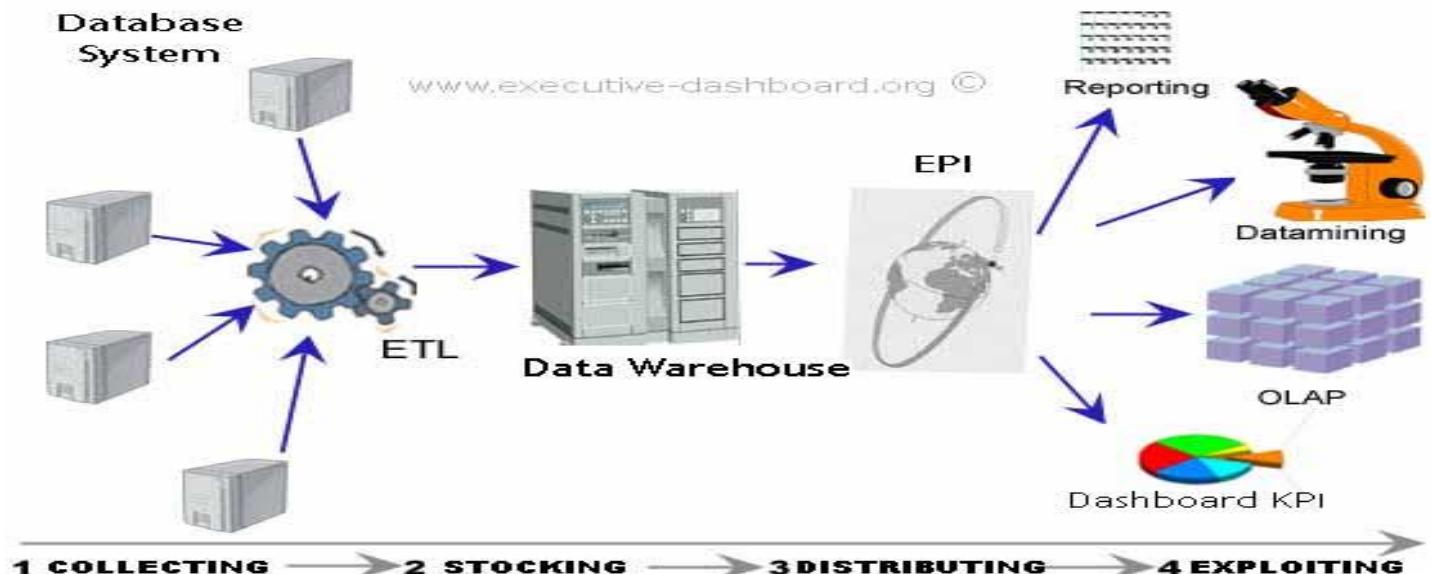
v10007b068 fotosearch.com

Inteligencia de Negocios

Los gerentes de las organizaciones requieren soluciones que les permitan:

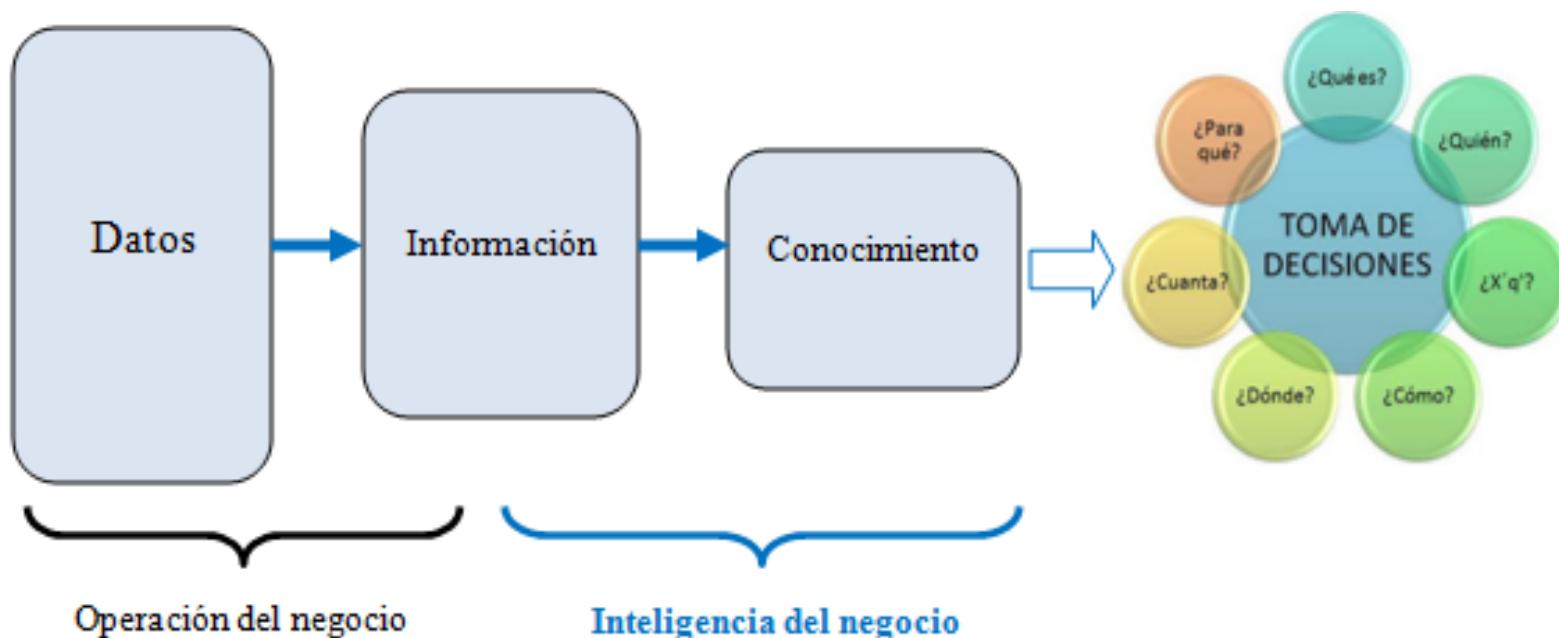
- ❑ Observar ¿qué está ocurriendo?
- ❑ Comprender ¿por qué ocurre?
- ❑ Predecir ¿qué ocurriría?
- ❑ Colaborar ¿qué debería hacer el equipo?
- ❑ Decidir ¿qué camino se debe seguir?

Inteligencia de negocios

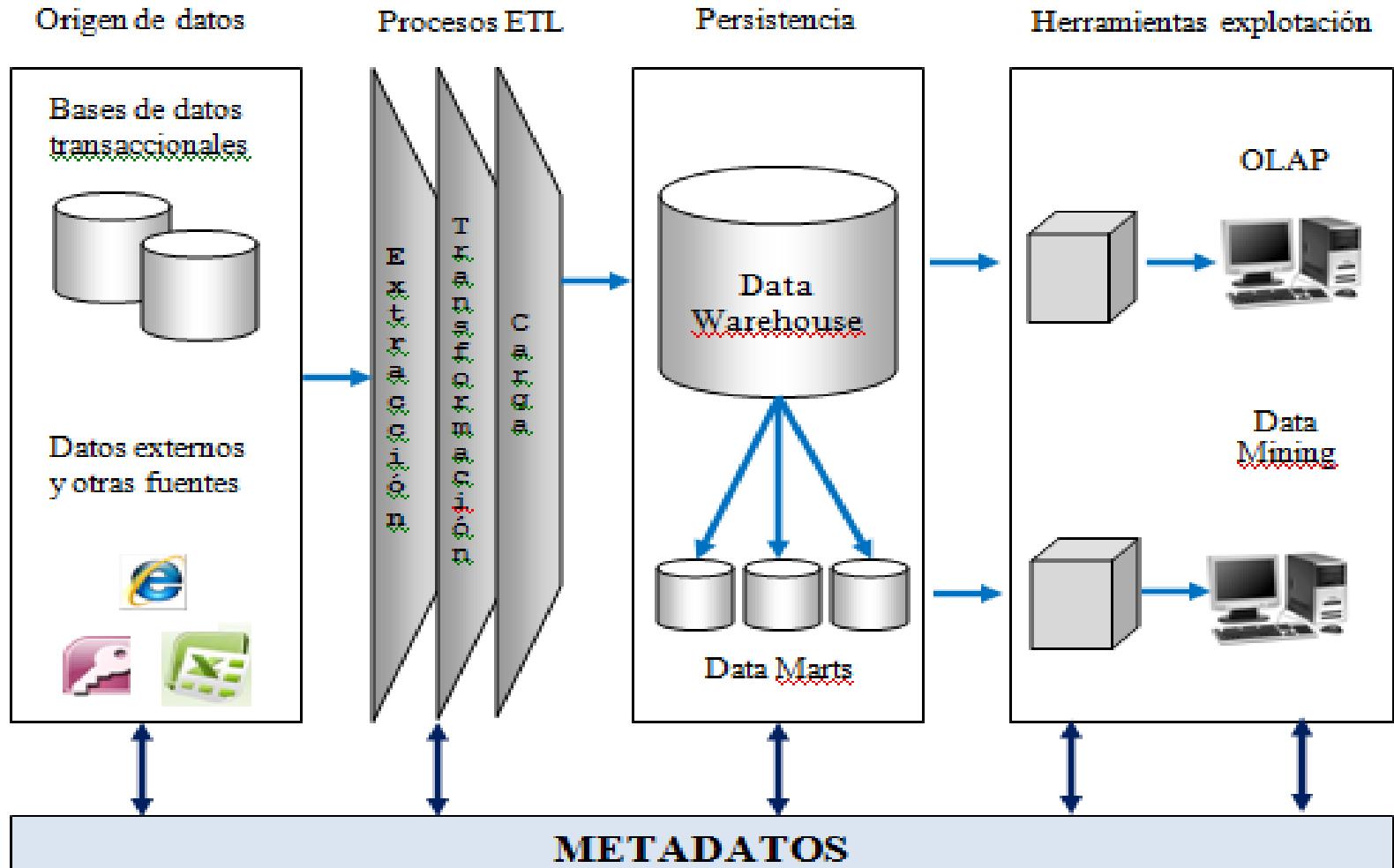


Inteligencia de negocios (Business Intelligence)

Es el proceso de integración y tratamiento de los datos para convertirlos en información, y la información en conocimiento, de forma que se pueda optimizar el proceso de toma de decisiones en los negocios y obtener ventajas competitivas.



Data warehousing



Data warehousing

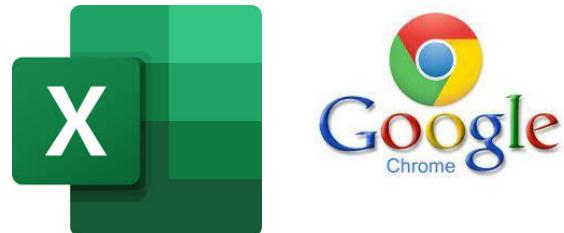
Origen de datos

Datos internos

Bases de datos transaccionales



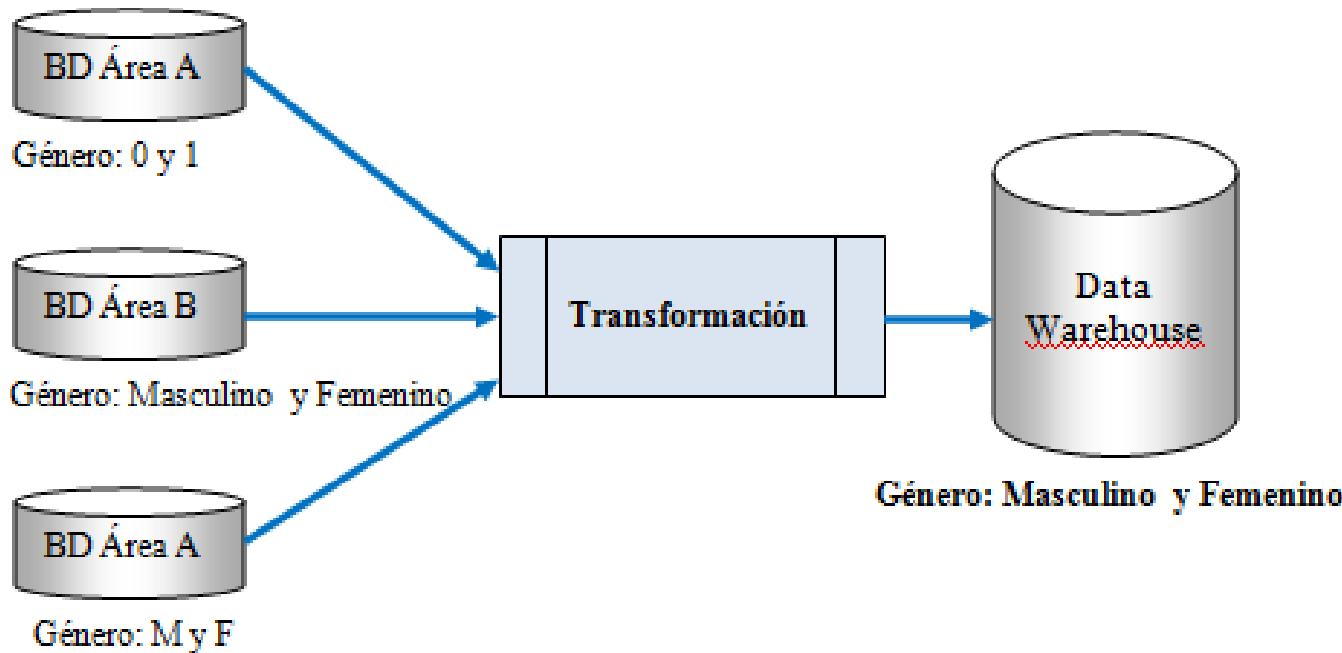
Datos externos



Data warehousing

Procesos ETL (Extracción, Transformación y carga (Load))

Es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos, depurarlos, limpiarlos y cargarlos en otra base de datos, denominada: data mart o data warehouse, con el propósito de analizarlos.



Data warehousing

Persistencia

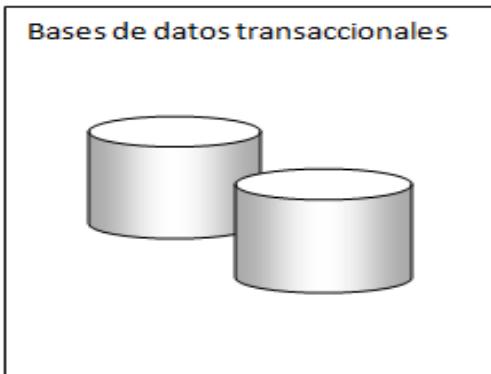
Lo constituye el almacén de datos (Data mart o Data warehouse)

Atributos de dimensión

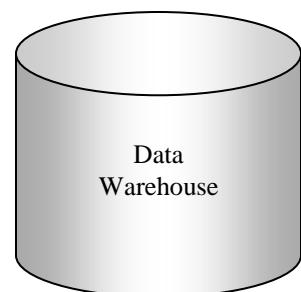
Cuándo	Dónde	Qué	Quién	Cuál
Año	Continente	Tipo producto	Vendedor	Tipo cliente
Semestre	País	Línea producto	Sucursal	Cliente
Trimestre	Región	Producto	Estudiante	
Mes	Ciudad	Asignatura	Docente	
Semana		Carrera Profesional		
Día				

Atributos de medición

Atributos de medición más comunes
Cantidad de productos vendidos
Ventas en soles o dólares
Número de piezas producidas
Notas obtenidas por un estudiante

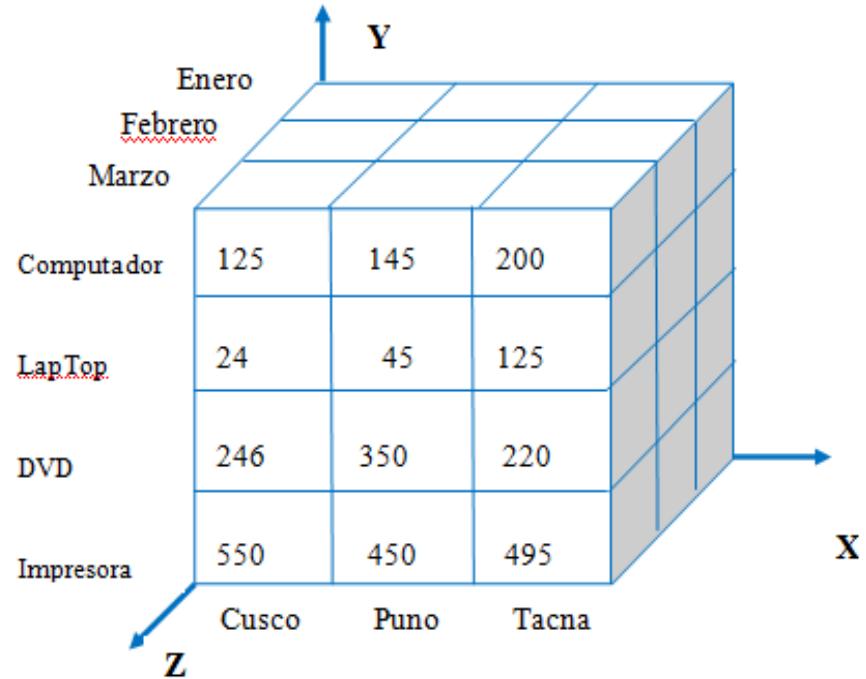
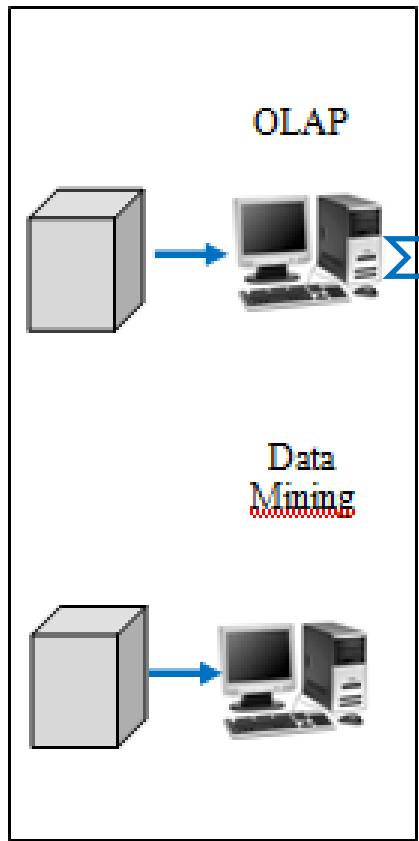


Criterios de análisis
(Atributos de Dimensión)
Medir o valorar
(Atributos de Medición)



Data warehousing

Herramientas de explotación



Modelos de Data warehouse

Modelo de Estrella

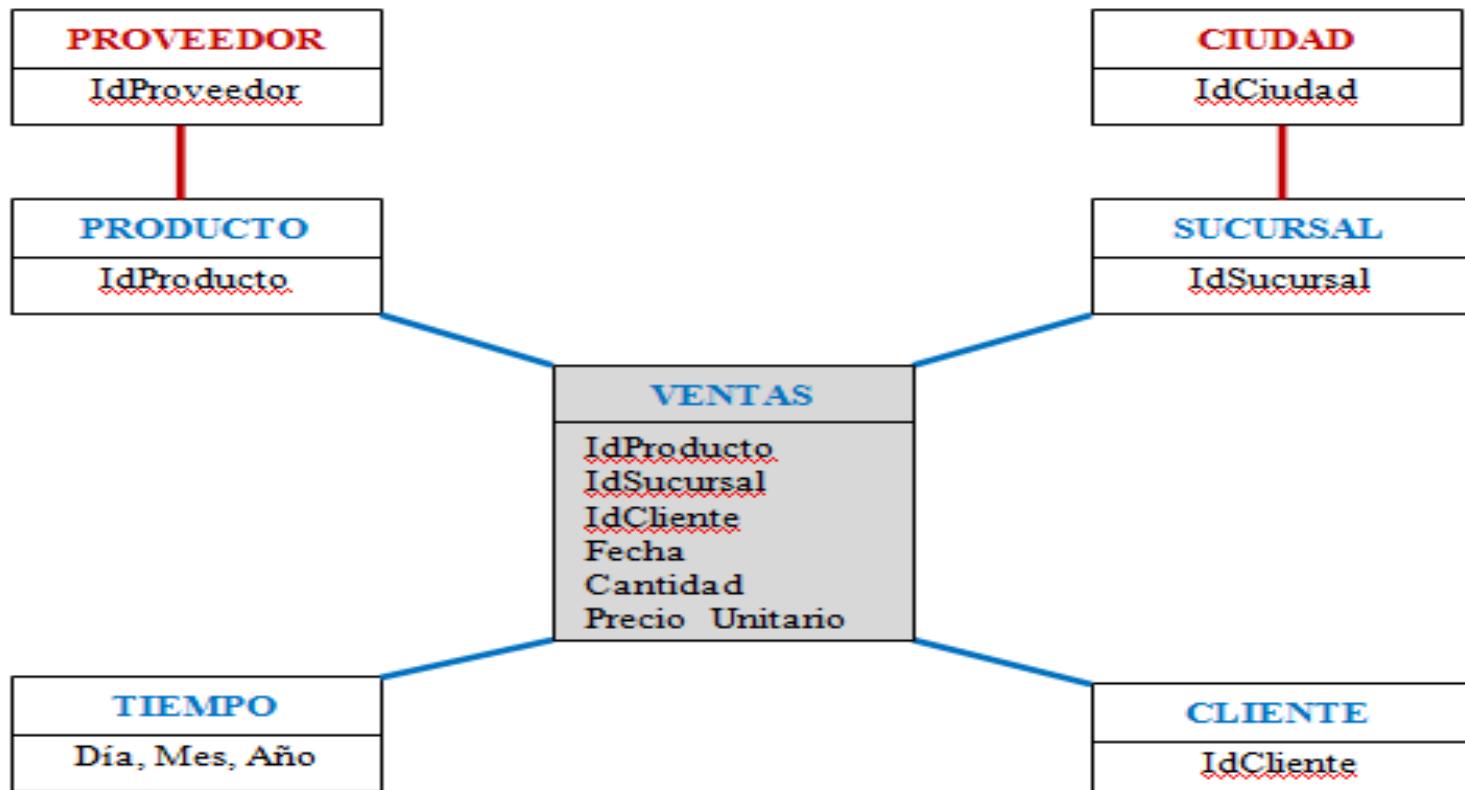
El centro del modelo estrella consta de una o más tablas de hechos que están constituidas por atributos de medición y las puntas de la estrella lo constituyen las tablas de dimensiones que están constituidas por atributos de dimensión que vienen a ser los criterios de análisis de información.



Modelos de Data warehouse

Modelo de copo de nieve (snowflake)

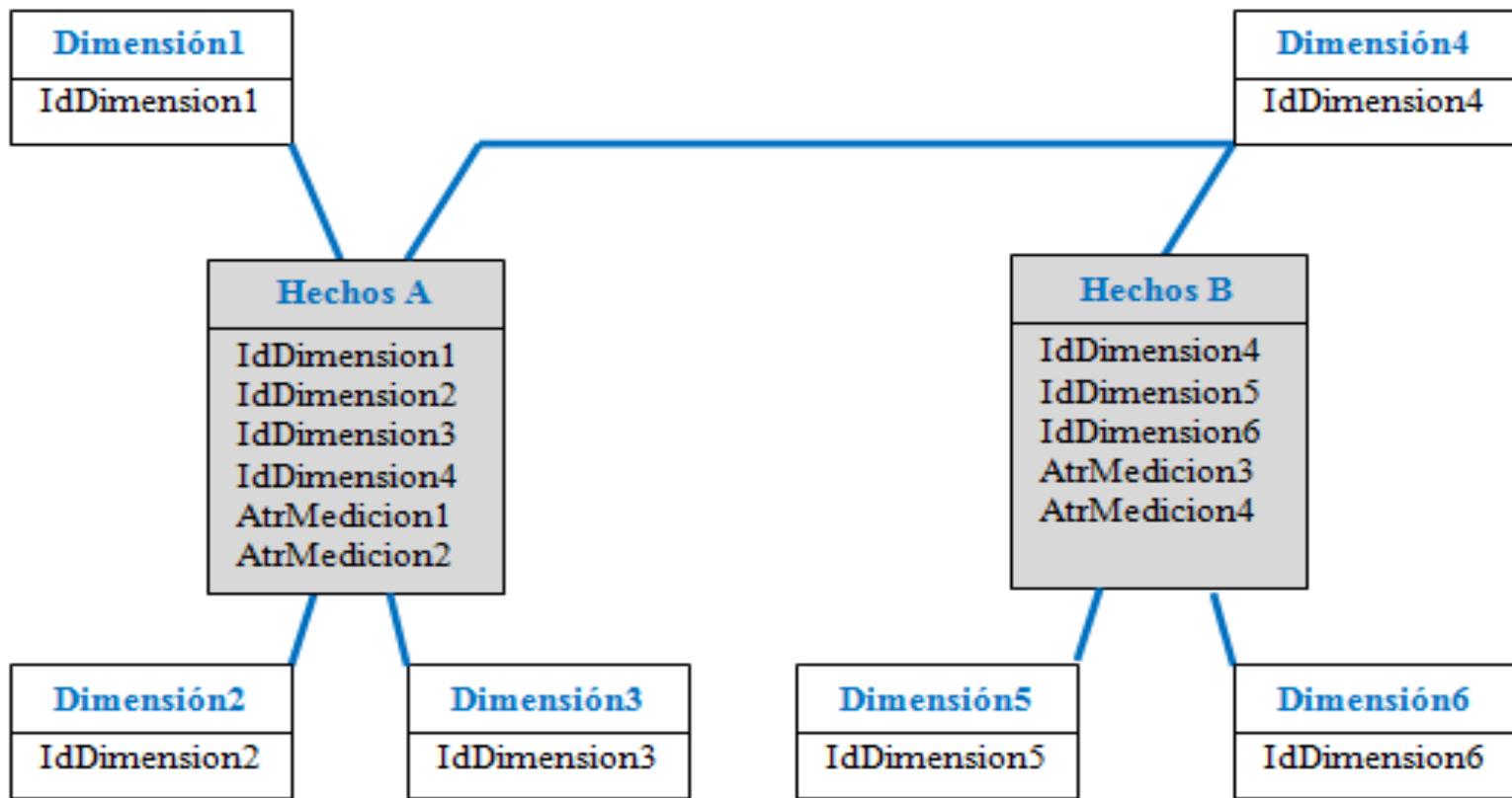
Es un derivado del modelo estrella, con la finalidad de optimizar las tablas de dimensiones mediante procesos de normalización que subdividen las tablas de dimensión en múltiples tablas, lo que permite disminuir la redundancia de datos.



Modelos de Data warehouse

Modelo de constelación

Está formado por dos o más esquemas de tipo estrella. Pueden existir varias tablas de hechos, donde una de ellas hace el papel de tabla de hechos principal y las otras de tabla de hechos auxiliares.

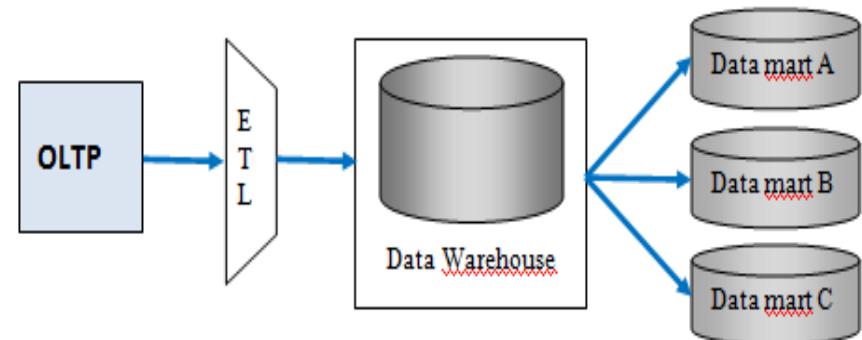


Metodologías de implementación de un Data warehouse

Existen varias metodologías, las más relevantes son:

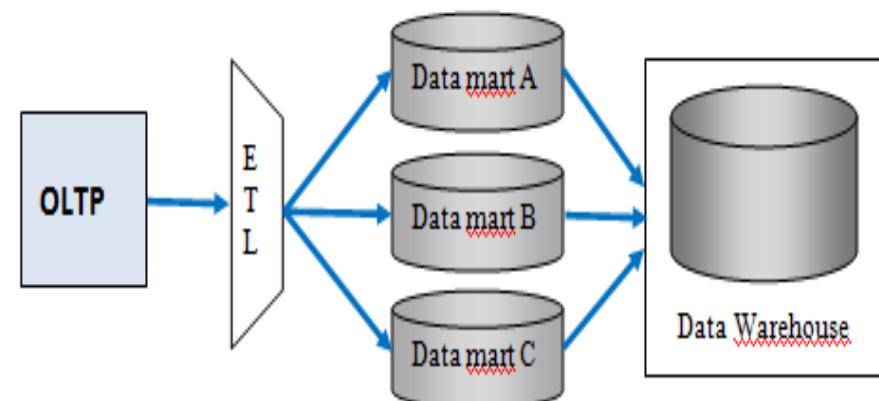
Metodologías Top-Down

- ❑ Metodología de Bill Inmon



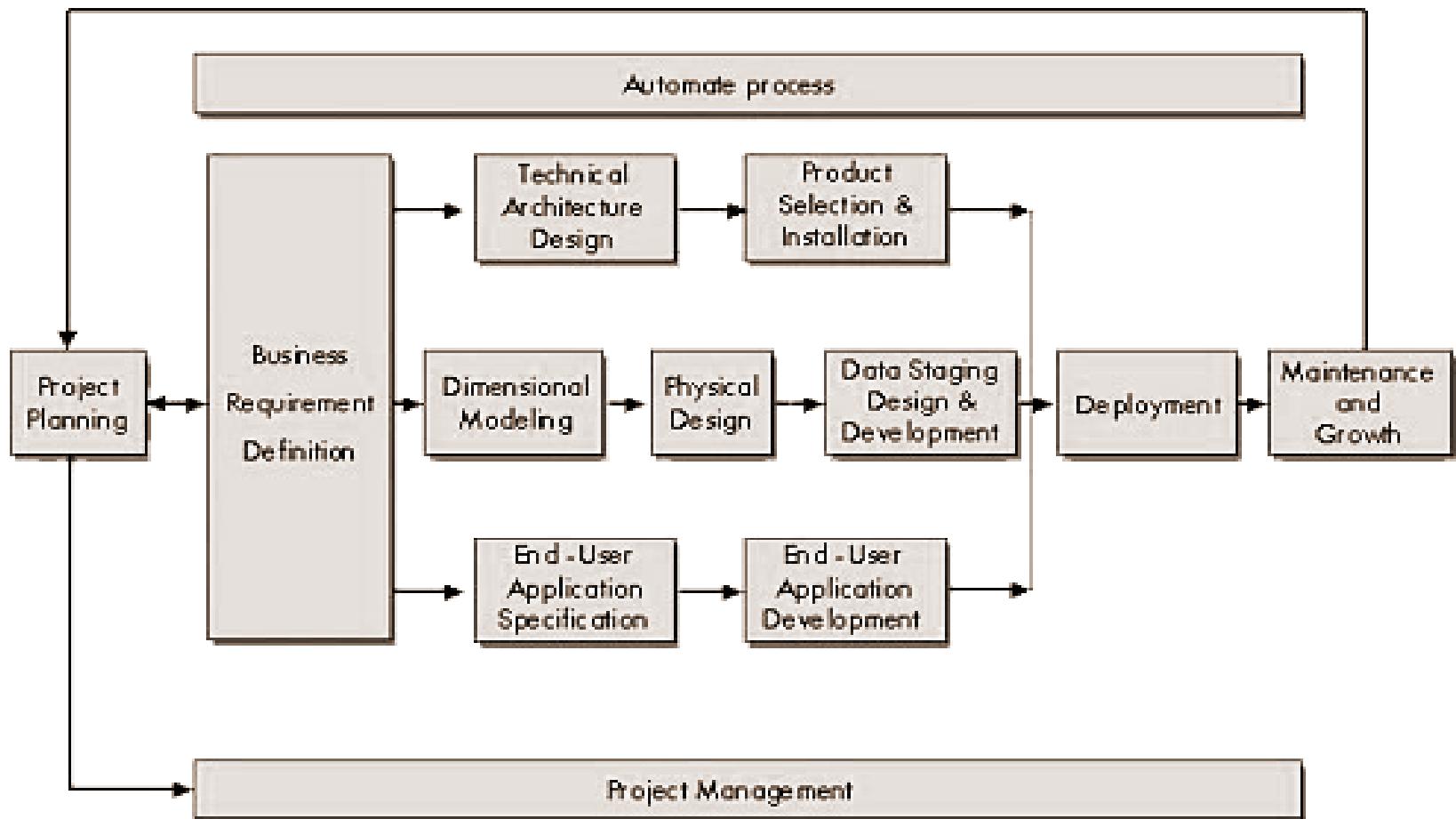
Metodologías Bottom-Up

- ❑ Metodología de Ralph Kimball
- ❑ Rapid Warehousing Methodology



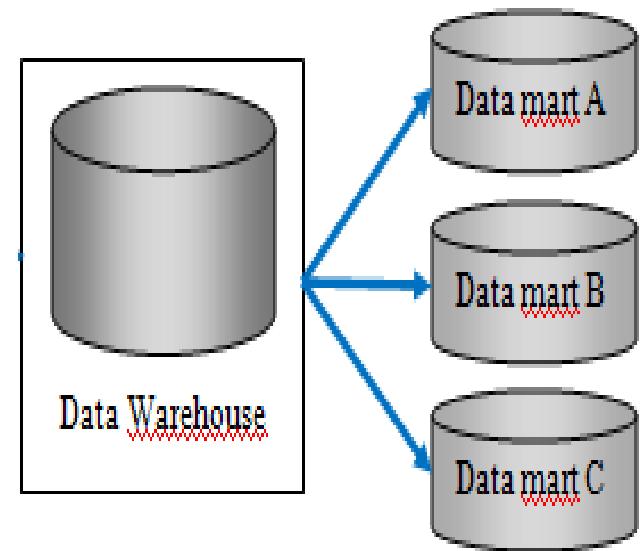
Metodologías de implementación de un Data warehouse

Metodología de Ralph Kimball



Data Mart

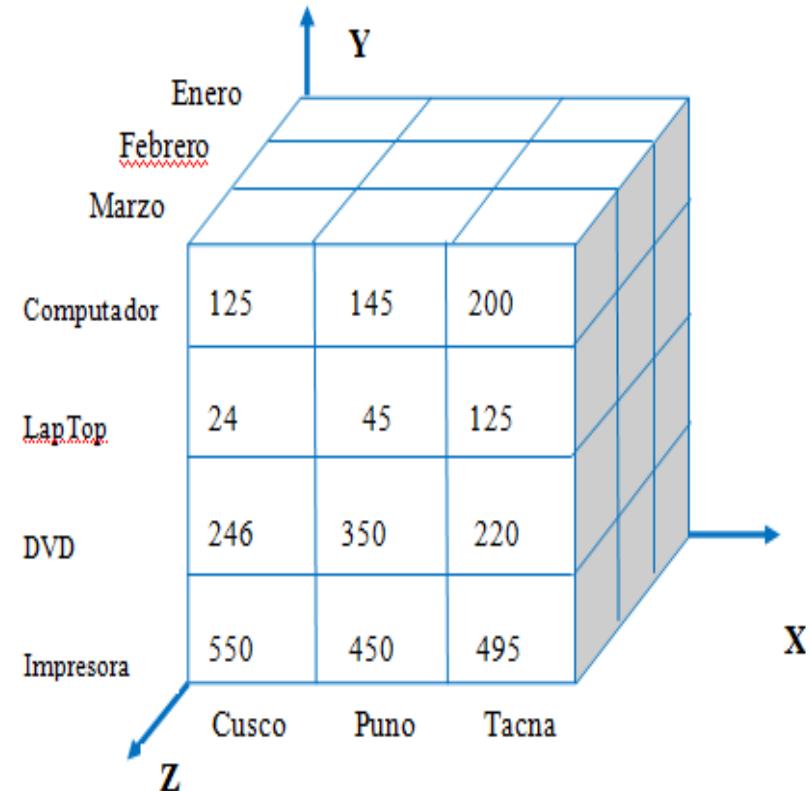
Un Data mart es un subconjunto de un Data Warehouse, orientado a un área o sección de la organización. Contiene los datos sólo de un área, en consecuencia da soporte a la toma de decisiones de dicha área, pudiendo los datos ser agrupados, explorados y propagados de diversas formas.



OLAP (*On-Line Analytical Processing*)

Es uno de los tipos de solución de la “*Inteligencia de negocios*”, cuyo objetivo fundamental es agilizar los procesos de extracción de información (consultas) de grandes volúmenes de datos, para dar soporte a la toma de decisiones.

Funciona en base a los “*Cubos Multidimensionales*”



Tipos de implementación de OLAP

❑ MOLAP

utilizaban arrays multidimensionales en memoria principal para almacenar los cubos de datos, de ahí el acrónimo MOLAP (Multidimensional OLAP).

La principal característica de este tipo de implementación es que permite optimizar los tiempos de respuesta.

❑ ROLAP

La implementación que se basa en los sistemas de gestión de bases de datos relacionales toma el nombre de ROLAP (Relational OLAP), por tanto, tiene todas las bondades de un SGBD relacionales.

La principal ventaja es que ofrece una estructura flexible y escalable

❑ HOLAP

Es un híbrido de los dos anteriores tipos (Hybrid OLAP), que combina sus principales ventajas e intenta minimizar o superar sus inconvenientes.

Operadores OLAP

Drill – Down

Roll – Up

Drill – Across

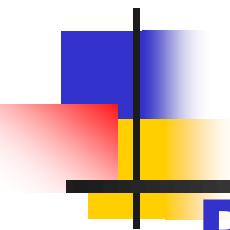
Roll – Across

Slice

Dice

Pivot

Modelo Relacional - SQL

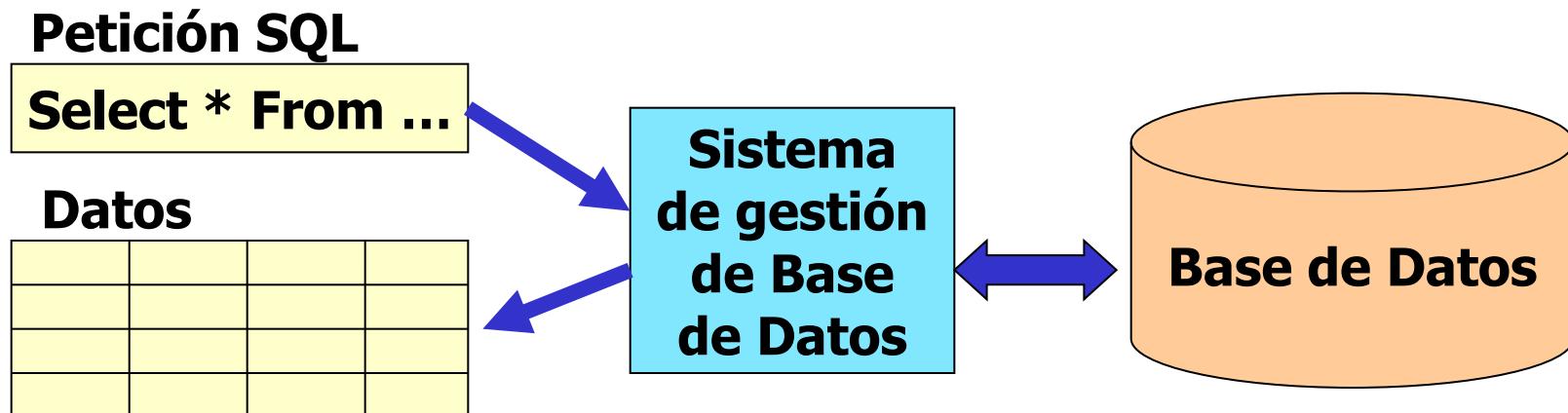


**USO INTERACTIVO Y
PROGRAMACION DE APLICACIONES**

4.1.1- El Lenguaje Estructurado de Consulta (SQL)

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos.

**SQL = Structured Query Language
(Lenguaje estructurado de Consulta)**



4.1.1- El Lenguaje Estructurado de Consulta (SQL)

El nombre SQL es en cierta medida un nombre inapropiado, porque es mucho más que un lenguaje de consulta, es una herramienta que permite controlar todas las funciones de un SGBD.

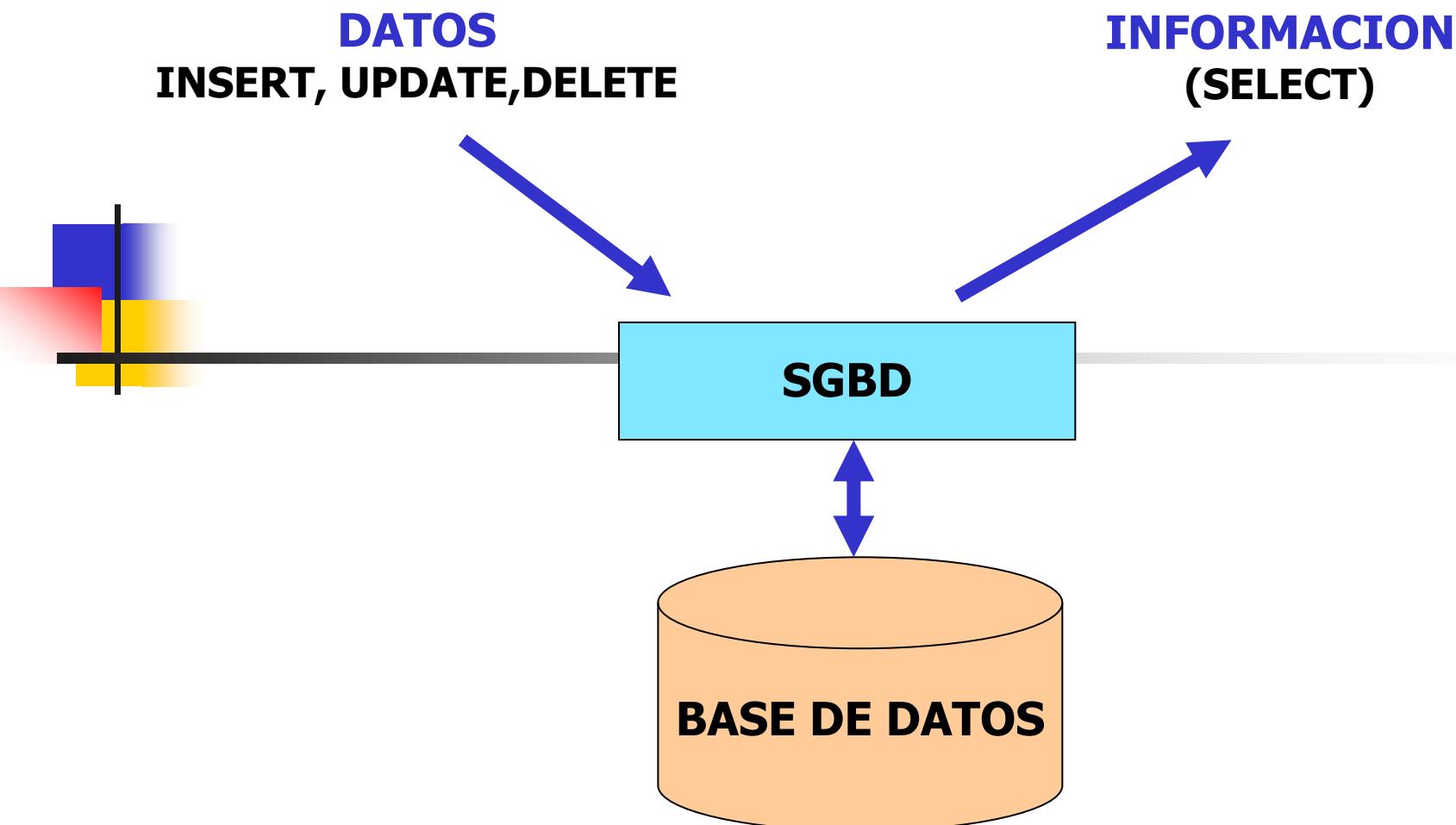
- **Definición de datos**
- **Recuperación de datos**
- **Manipulación de datos**
- **Control de acceso**
- **Compartición de datos**
- **Integridad de datos**

4.1.1- El Lenguaje Estructurado de Consulta (SQL)

SQL es un Lenguaje de:

- **Programación de Base de datos**
- **Administración de Base de Datos**
- **Cliente/Servidor**
- **Base de datos distribuida**
- **Pasarela de Base de Datos**

4.1.2- Recuperación de datos



4.1.3- Conceptos básicos

Principales grupos de sentencias

■ Definición de Datos

- CREATE
- DROP
- ALTER

■ Manipulación de Datos

- SELECT
- INSERT
- DELETE
- UPDATE

Control de Acceso

- GRANT
- REVOKE

■ Control de Transacciones

- COMMIT
- ROLLBACK

4.1.4- Consultas simples

Estructura de la sentencia SELECT

SELECT Atributos

FROM Tablas

WHERE Condición

GROUP BY Atributos_Grupo

HAVING Condición_Grupo

ORDER BY Atributos_Ordenamiento

4.1.4- Consultas simples...

Ejemplos:

ALUMNO(Codigo, Apellido_Paterno, Apellido_Materno, Nombres, CP)

```
SELECT Apellido_Paterno, Apellido_Materno, Nombres  
1   FROM Alumno
```

2

```
SELECT Apellido_Paterno, Apellido_Materno, Nombres  
1   FROM Alumno  
2   WHERE CP='CO'
```

3

4.1.4- Consultas simples...

Condiciones:

- Test de Comparación

=, <>, <, <=, >, >=

- Test de Rango

Between

- Test de pertenencia a conjunto

IN

- Test de correspondencia con patrón

Like

- Test de valor nulo

IS NULL

4.1.4- Consultas simples...

Funciones de resumen:

SUM: Calcula la suma de una columna de datos.
(Columna de tipo numérico).

AVG: Calcula el promedio de una columna de datos.
(Columna de tipo numérico).

MIN: Retorna el valor mínimo de una columna

MAX: Retorna el valor máximo de una columna

COUNT: Cuenta el número de registros. El resultado es un valor entero.

4.1.4- Consultas simples...

En base a la siguiente tabla:

ALUMNO(Codigo, Apellido_Paterno, Apellido_Materno, Nombres, CP)

Determinar el número de estudiantes de la universidad

```
SELECT Count(*)  
      FROM Alumno
```

Determinar el número total de estudiantes de las carreras profesionales de Ingeniería de la universidad

```
SELECT Count(*) 3  
1   FROM Alumno  
2 WHERE CP like 'I%'
```

4.1.4- Consultas simples...

Determinar el número de estudiantes de cada carrera profesionales de Ingeniería de la universidad

SELECT CP, Count(*) as NroAlumnos

4

1 FROM Alumno

2 WHERE CP like 'I%'

3 GROUP BY CP

4.1.4- Consultas simples...

Determinar el número de estudiantes de cada carrera profesionales de Ingeniería de la universidad y que tengan más de 200 alumnos. La relación resultante que muestre ordenado por número de alumnos y en orden descendente.

SELECT CP, Count(*) as NroAlumnos

5

1 FROM Alumno

2 WHERE CP like 'I%'

3 GROUP BY CP

4 HAVING Count(*) > 200

6 ORDER BY NroAlumnos Desc

Sistemas de Base de Datos

4.1.5- Composición (Consultas multitable)

```
SELECT *  
FROM ALUMNO inner join CP
```

TABLA ALUMNO

Codigo	Nombres	ID_Carrera
121	Pedro	TU
234	Ana	AG
148	Eva	TU

TABLA CP

ID_Carrera	Nombre
TU	Turismo
AG	Agronomía

FROM ALUMNO INNER JOIN CP

ALUMNO.Codigo	ALUMNO.Nombres	ALUMNO.ID_Carrera	CP.ID_Carrera	CP.Nombre
121	Pedro	TU	TU	Turismo
121	Pedro	TU	AG	Agronomía
234	Ana	AG	TU	Turismo
234	Ana	AG	AG	Agronomía
148	Eva	TU	TU	Turismo
148	Eva	TU	AG	Agronomía

Sistemas de Base de Datos

4.1.5- Composición (Consultas multitablea)...

```
SELECT *  
FROM ALUMNO INNER JOIN CP  
ON ALUMNO.ID_Carrera = CP.ID_Carrera
```

Tabla resultante antes de aplicar la cláusula ON

ALUMNO.Codigo	ALUMNO.Nombres	ALUMNO.ID_Carrera	CP.ID_Carrera	CP.Nombre
121	Pedro	TU	TU	Turismo
121	Pedro	TU	AG	Agronomía
234	Ana	AG	TU	Turismo
234	Ana	AG	AG	Agronomía
148	Eva	TU	TU	Turismo
148	Eva	TU	AG	Agronomía

Tabla resultante después de aplicar la cláusula ON

ALUMNO.Codigo	ALUMNO.Nombres	ALUMNO.ID_Carrera	CP.ID_Carrera	CP.Nombre
121	Pedro	TU	TU	Turismo
234	Ana	AG	AG	Agronomía
148	Eva	TU	TU	Turismo

4.1.5.1.- Composición Interna

Este tipo de composición se utiliza cuando las cardinalidades mínimas son 1 (Obligatorios).



```
SELECT *  
FROM ALUMNO A inner join CARRERA C  
ON A.ID_Carrera = C.ID_Carrera
```

Ejercicios. Dada las siguientes tablas:

Carrera_Profesional(Cod_CP, Nombre_CP)

Alumno(Cod_Alumno, Paterno, Materno, Nombres, Cod_CP)

Docente(Cod_Docente, Paterno, Materno, Nombres,
Categoría, Regimen)

Asignatura(Cod_Asignatura,Cod_CP,Nombre_Asignatura,
Categoría, Creditos)

Catalogo(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Docente)

Matricula(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Alumno, Nota)

Ejercicios resueltos

Prob 1. Determinar el número de alumnos matriculados por semestre y por Carrera Profesional

-- Agregar el código de la carrera a la tabla matricula

```
SELECT M.* , Cod_CP  
      INTO #matriculaCP  
    FROM Matricula M inner join Alumno A  
      ON M.Cod_Alumno = A.Cod_Alumno
```

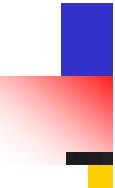
-- Contar matriculados por semestre de cada carrera profesional

```
SELECT Semestre, Cod_CP,  
      Nro_Matriculados = count(distinct Cod_Alumno)  
    INTO #alumnoSemestre  
  FROM #matriculaCP  
 GROUP BY Semestre, Cod_CP  
 ORDER BY Semestre, Cod_CP
```

-- Agregar datos de la Carrera profesional

```
SELECT T1.Semestre, T1.Cod_CP, Nombre_CP,  
      Nro_Matriculados  
    FROM #alumnoSemestre T1,Carrera T2  
   WHERE T1.Cod_CP = T2.Cod_CP  
 ORDER BY Semestre, T1.Cod_CP
```

Ejercicios resueltos



Prob 2. **Determinar** la relación de asignaturas que en el último semestre tengan menos de 7 alumnos matriculados, con el objetivo de desactivar dichas asignaturas en el semestre.

PRIMER ALGORITMO

-- Determinar ultimo semestre

```
SELECT ultimoSemestre = max(Semestre)
      INTO #ultimoSemestre
      FROM Matricula
```

-- Seleccionar las asignaturas que se dictan en este último Semestre

```
SELECT M.*
      INTO #AsignaturasUltimoSemestre
      FROM Matricula M inner join #ultimoSemestre U
      ON M.Semestre = U.ultimoSemestre
```

-- Determinar las asignaturas que tengan menos de 7 alumnos

```
SELECT Semestre, Cod_Asignatura, Cod_CP, Grupo,
      Nro_Alumnos = count(Cod_Alumno)
      INTO #asignaturasAlumnos
      FROM #AsignaturasUltimoSemestre
      GROUP BY Semestre, Cod_Asignatura, Cod_CP, Grupo
      HAVING count(Cod_Alumno) < 7
```

-- Agregar datos de la asignatura

```
SELECT A.* , Nro_Alumnos
      FROM #asignaturasAlumnos T1 inner join Asignatura A
      ON T1.Cod_Asignatura=A.Cod_Asignatura and T1.Cod_CP=A.Cod_CP
```

SEGUNDO ALGORITMO

-- Determinar ultimo semestre

```
Declare @UltimoSemestre varchar(7);
SELECT @UltimoSemestre = max(Semestre)
FROM Matricula
```

-- Determinar las asignaturas del ultimo semestre que tengan
menos de 7 alumnos

```
SELECT Cod_Asignatura, Cod_CP, Grupo,
       Nro_Alumnos = count(Cod_Alumno)
INTO #asignaturasAlumnos
FROM Matricula M
WHERE Semestre = @UltimoSemestre
GROUP BY Cod_Asignatura, Cod_CP, Grupo
HAVING count(Cod_Alumno) < 7
```

-- Agregar datos de la asignatura

```
SELECT A.* , Nro_Alumnos
FROM #asignaturasAlumnos T1 inner join Asignatura A
ON T1.Cod_Asignatura=A.Cod_Asignatura and T1.Cod_CP=A.Cod_CP
```

Ejercicios resueltos



Prob 3. Obtener la ficha de seguimiento
del estudiante de código 980115.

PRIMER ALGORITMO

-- Obtener la información de las tablas Matrícula y Asignatura

```
select M.Semestre, M.Cod_Asignatura, A.Categoría, A.Creditos, M.Nota
from MATRICULA M inner join ASIGNATURA A
on (M.Cod_Asignatura = A.Cod_Asignatura) AND
(M.Cod_CP = A.Cod_CP)
where Cod_Alumno = '980115' and
Nota in ('11','12','13','14','15','16','17','18','19','20')
```

SEGUNDO ALGORITMO

-- Recuperar matriculas del estudiante 980115

```
select Semestre, Cod_Asignatura, Cod_CP, Nota  
    into #MATRICULA  
    from MATRICULA  
   where Cod_Alumno = '980115' and  
         Nota in('11','12','13','14','15','16','17','18','19','20')
```

-- Completar información de asignatura

```
select M.Semestre, M.Cod_Asignatura, A.Nombre_Asignatura,  
      A.Categoría, A.Creditos, M.Nota  
  from #MATRICULA M inner join ASIGNATURA A  
  on (M.Cod_Asignatura = A.Cod_Asignatura) and  
      (M.Cod_CP = A.Cod_CP)
```

TERCER ALGORITMO

-- Declarar variable

```
declare @Cod_Alumno varchar(6);
set @Cod_Alumno = '980115';
```

-- Recuperar matriculas del estudiante 980115

```
select Semestre, Cod_Asignatura, Cod_CP, Nota
into #MATRICULA1
from MATRICULA
where (Cod_Alumno = @Cod_Alumno) and
      (Nota in('11','12','13','14','15','16','17','18','19','20'))
```

-- Completar información de asignatura

```
select M.Semestre, M.Cod_Asignatura, A.Nombre_Asignatura,
       A.Categoría, A.Creditos, M.Nota
  from #MATRICULA1 M inner join ASIGNATURA A
    on (M.Cod_Asignatura = A.Cod_Asignatura) and
       (M.Cod_CP = A.Cod_CP)
```

4.1.5.2.- Composición Externa

Dadas las siguientes tablas, determinar la relación de préstamos con sus respectivos saldos

TABLA PRESTAMO

ID_Prestamo	Importe
PA-134	600
PA-345	1200
PA-456	850

TABLA AMORTIZACION

ID_Amortizacion	Importe	Id_Prestamo
RC-234	450	PA-134
RC-241	850	PA-456

¿Es la siguiente sentencia la solución?

```
SELECT P.* , (P.Importe - A.Importe) as Saldo  
FROM PRESTAMO P inner join AMORTIZACION A  
ON P.ID_Prestamo = A.ID_Prestamo
```

4.1.5.2.- Composición Externa...

La sentencia: FROM PRESTAMO P inner join AMORTIZACION A
Genera la siguiente tabla

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

La sentencia: ON P.ID_Prestamo = A.ID_Prestamo
selecciona las tuplas que no están subrayadas

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

4.1.5.2.- Composición Externa...

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-134	600	RC-241	850	PA-456
PA-345	1200	RC-234	450	PA-134
PA-345	1200	RC-241	850	PA-456
PA-456	850	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

En Consecuencia la relación resultante es:

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-456	850	RC-241	850	PA-456

El préstamo PA-345 ha sido ignorado debido a que no empareja con ninguna de las tuplas de la tabla de AMORTIZACION

4.1.5.2.- Composición Externa...

Este tipo de composición se utiliza cuando las cardinalidades mínimas son 0 (No son obligatorios).



Como el modelo anterior indica que hay PRESTAMOS que no necesariamente están relacionados con AMORTIZACIONES, entonces estas instancias desaparecerán cuando se hace una Composición Interna, en su lugar se debe realizar una Composición Externa.

4.1.5.2.- Composición Externa...

Sintaxis:

```
SELECT atributos
      FROM tabla1 (LEFT/RIGHT) OUTER JOIN tabla 2
           ON tabla1.atributo=tabla2.atributo
```

OUTER JOIN
LEFT OUTER JOIN
RIGHT OUTER JOIN

Considera los registros de ambas tablas
Considera todos los registros de la tabla de la izquierda
Considera todos los registros de la tabla de la derecha

4.1.5.2.- Composición Externa...

La solución al problema propuesto es:

```
SELECT P.* , (P.Importe - A.Importe) as Saldo  
FROM PRESTAMO P left outer join AMORTIZACION A  
ON P.ID_Prestamo = A.ID_Prestamo
```

La relación resultante es:

P.ID_Prestamo	P.Importe	A.ID_Amortizacion	A.Importe	A.Id_Prestamo
PA-134	600	RC-234	450	PA-134
PA-345	1200	null	null	null
PA-456	850	RC-241	850	PA-456

Ejercicios. Dada las siguientes tablas:

Carrera_Profesional(Cod_CP, Nombre_CP)

Alumno(Cod_Alumno, Paterno, Materno, Nombres, Cod_CP)

Docente(Cod_Docente, Paterno, Materno, Nombres,
Categoría, Regimen)

Asignatura(Cod_Asignatura,Cod_CP,Nombre_Asignatura,
Categoría, Creditos)

Catalogo(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Docente)

Matricula(Semestre, Cod_Asignatura, Cod_CP, Grupo,
Cod_Alumno, Nota)

Ejercicios resueltos

Prob 1. Determinar el número de créditos aprobados por los alumnos de ingeniería informática

-- Recuperar los alumnos de Ingeniería informática

```
SELECT Cod_Alumno, Paterno, Materno, Nombres  
      INTO #AlumnosIN  
     FROM Alumno
```

-- Seleccionar matriculas con asignaturas aprobadas y de Informática

```
SELECT Cod_Asignatura, Cod_Alumno, Nota  
      INTO #MatriculaIN  
     FROM Matricula  
    WHERE (Cod_CP = 'IN') and (Nota in (11,12,13,14,15,16,17,18,19,20))
```

-- Agregar datos de la Carrera profesional

```
SELECT T1.Semestre, T1.Cod_CP, Nombre_CP,  
          Nro_Matriculados  
     FROM #alumnoSemestre T1,Carrera T2  
    WHERE T1.Cod_CP = T2.Cod_CP  
ORDER BY Semestre, T1.Cod_CP
```

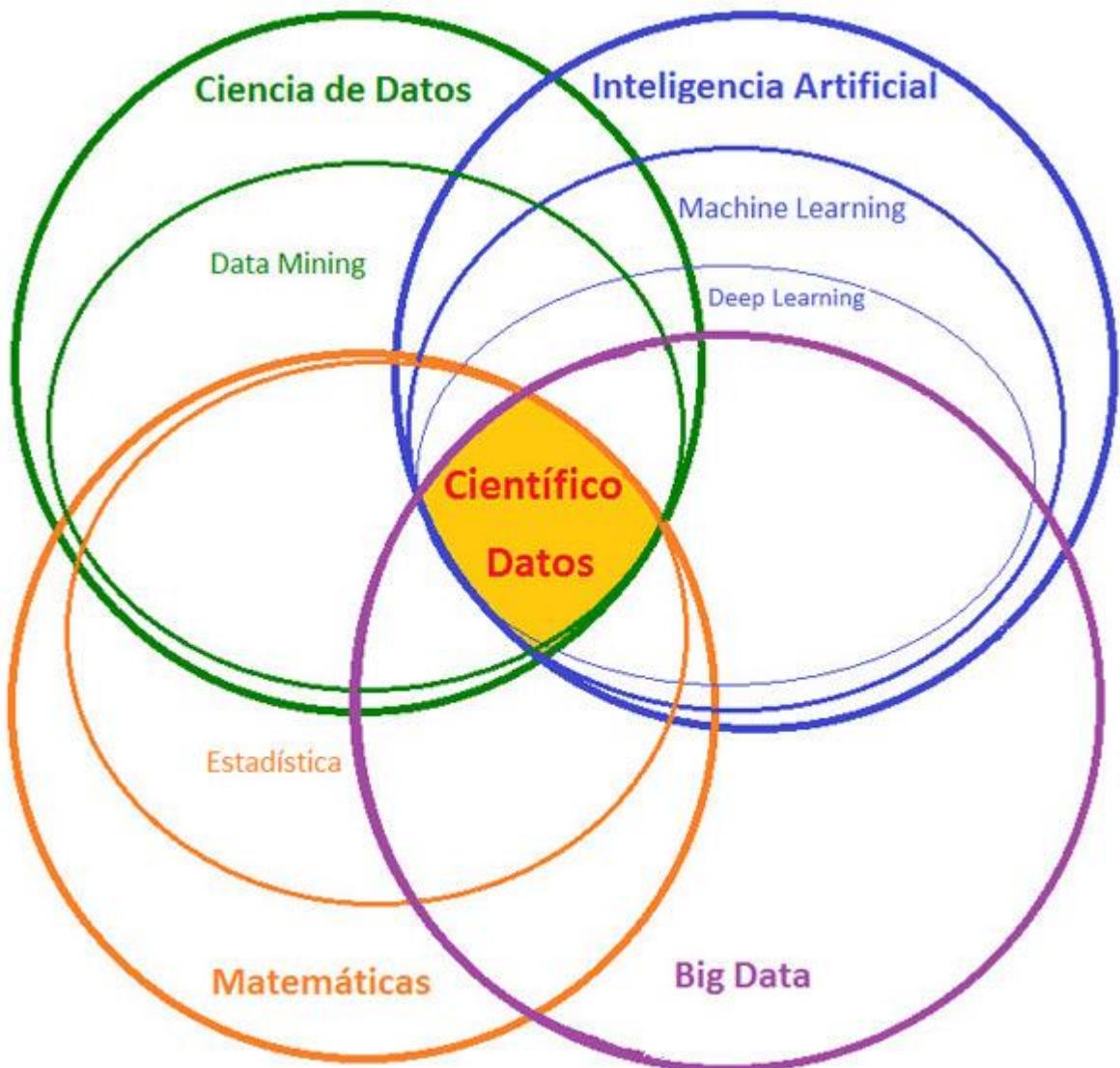
UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO



INTRODUCCIÓN A LA MINERÍA DE DATOS

AREA DE INTELIGENCIA ARTIFICIAL

Ciencia de datos y Minería de datos



La **ciencia de datos** es una disciplina que hace uso de diferentes métodos, tecnologías analíticas avanzadas y principios científicos para extraer información valiosa de los datos para la toma de decisiones

La **minería de datos** es el conjunto de técnicas y tecnologías que permiten explorar grandes **bases de datos**, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, correlaciones, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto; haciendo uso de la estadística, inteligencia artificial, etc.

Minería de datos

Minería de datos es un área de ciencias de la computación, tiene por objetivo descubrir patrones de comportamiento en grandes volúmenes de datos: Utiliza los métodos de:

- Inteligencia artificial (Aprendizaje automático)
 - Estadística
 - Bases de datos



“Es la extracción de patrones o información interesante (no trivial, implícita, previamente desconocida y potencialmente útil) de grandes volúmenes de datos”

Minería de datos

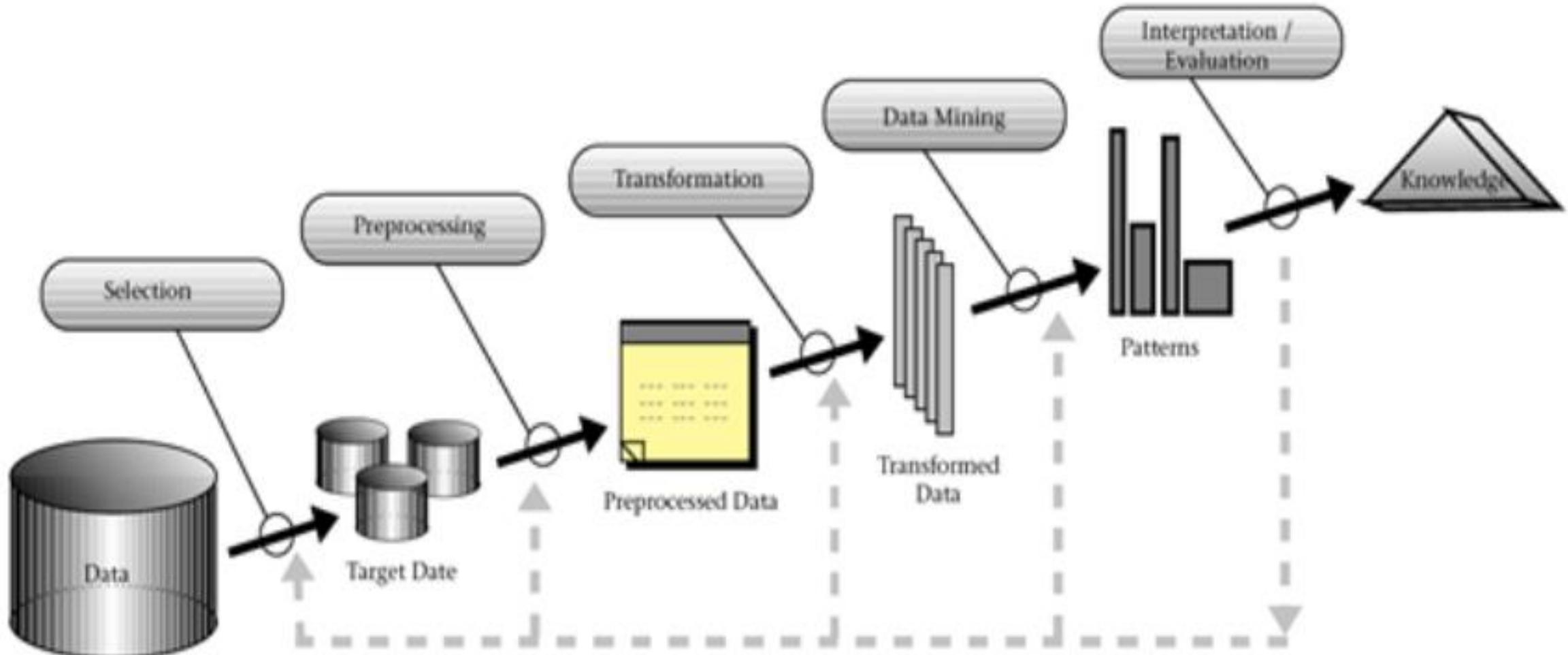
El proceso de KDD

El proceso de descubrimiento de conocimiento en los datos, KDD (Knowledge Discovery in Data), se representa de la siguiente forma:



Minería de datos

Procesos de la minería de datos



Minería de datos

Retos de la minería de datos

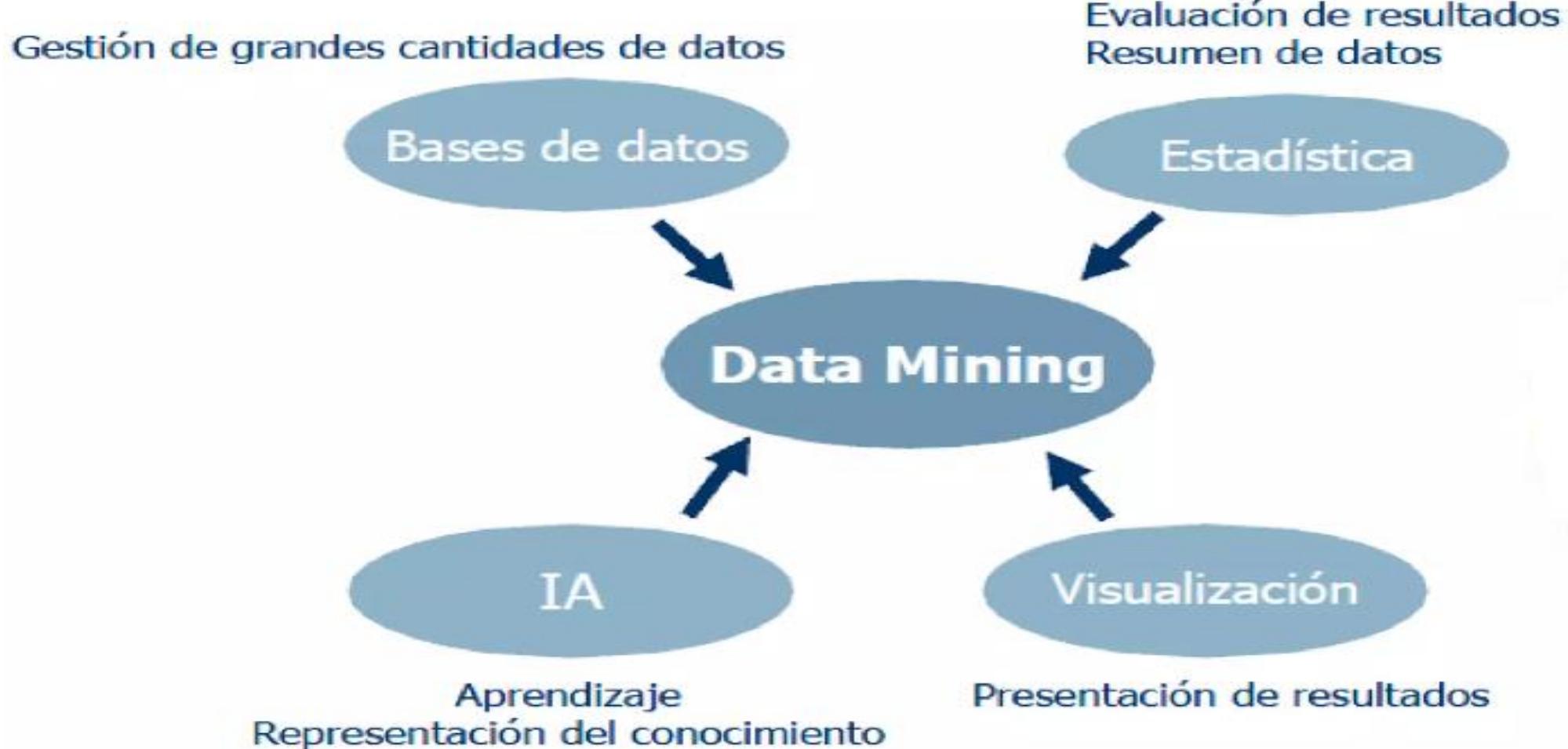
Es desarrollar una forma de pensar (teoría) y técnicas computacionales que permitan procesar y aprender de esta información:

- Reducción de dimensión
- Extracción de señales (filtrar el ruido)
- Visualización
- Aprender sobre problemas de interés (inferencia)
- Predecir (clasificar)
- Detectar anomalías

El marco conceptual de la minería de datos es la teoría de aprendizaje estadístico

Minería de datos

Áreas de interacción de la minería de datos



Minería de datos

Problemas típicos

La minería de datos típicamente se emplea en casos como:

- Predecir si un paciente va a ser hospitalizado con base en su historia clínica.
- Clasificación de dígitos (códigos) escritos a mano.
- Comprensión de información (imágenes).
- Determinantes de una enfermedad (síntomas y posología)
- Clasificación de clientes (instituciones financieras)
- Detección de anomalías y fraudes (sector financiero).

Minería de datos

Contextos de la minería de datos: Procesos determinísticos y estocásticos

Desde la perspectiva de la **minería de datos**, en el procesamiento y análisis de grandes volúmenes de datos, se plantea dos contextos

Procesos determinísticos

Utiliza herramientas **determinísticas** convencionales:

- Modelos matemáticos
- Estadística
- Lenguaje SQL*

Se aplica a problemas con relaciones entre atributos conocidos y reglas bien definidas (Obvias u obedecen a modelos matemáticos exactos).



Procesos estocásticos

Utiliza técnicas y métodos **estocásticos**:

- Estadística (Aleatoriedad)
- Aprendizaje automático*

Se aplica a problemas con relaciones entre atributos no lineales y complejos (No obvias).



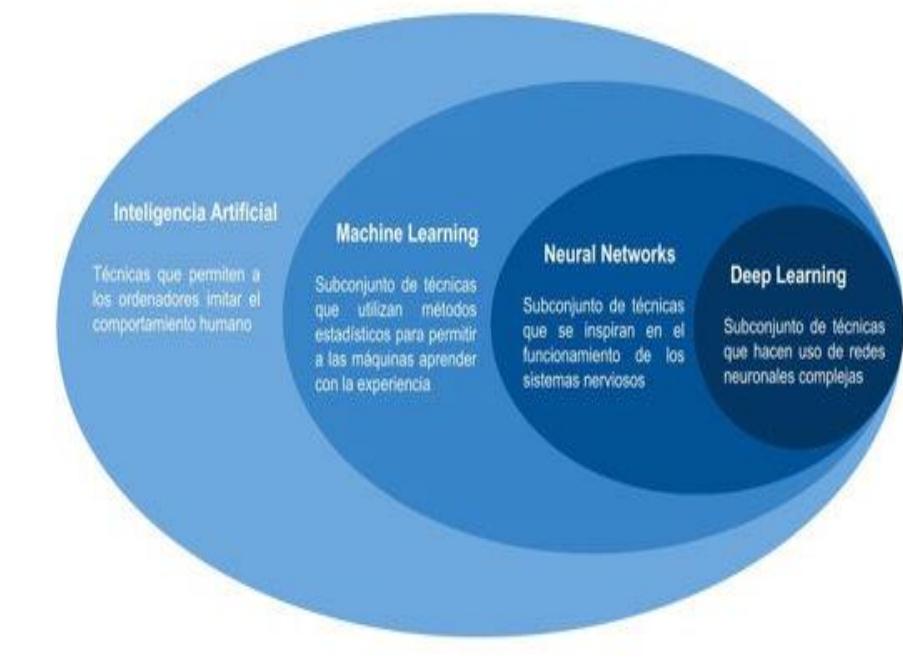
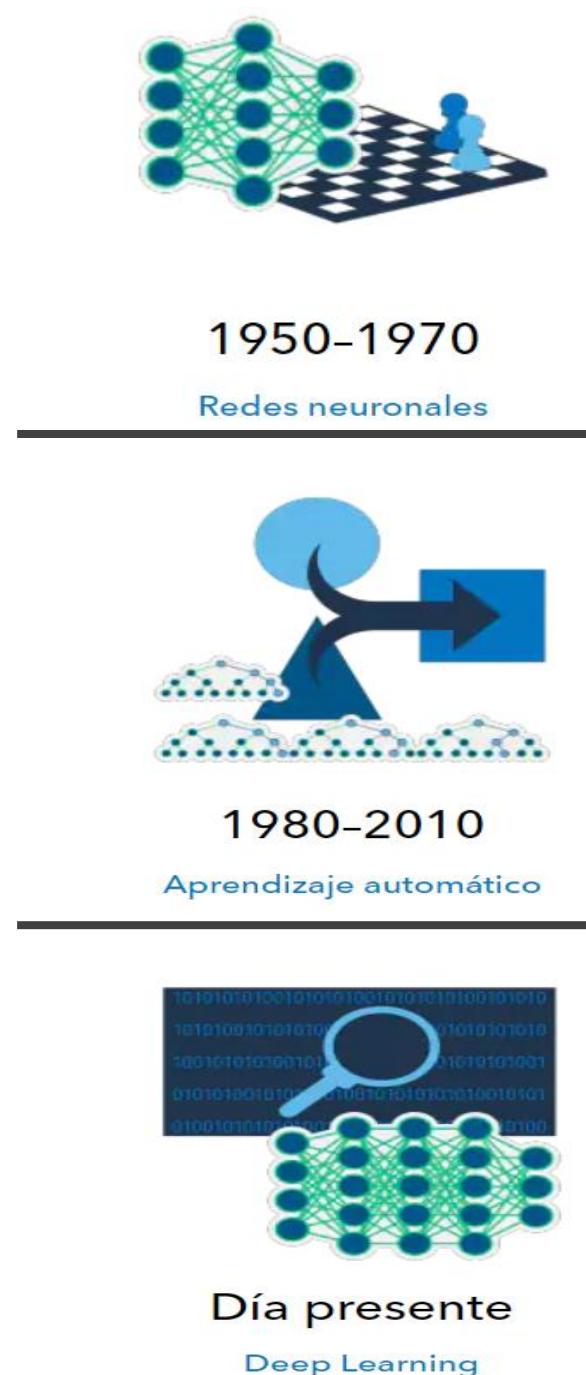
Técnicas de Minería de datos

Aprendizaje automático (Machine learning)

El **aprendizaje automático** es una área de la **inteligencia artificial** (IA) que **proporciona a los sistemas** la capacidad de **aprender y mejorar automáticamente a partir de la experiencia sin ser programado explícitamente**.

<https://expertsystem.com/machine-learning-definition/>

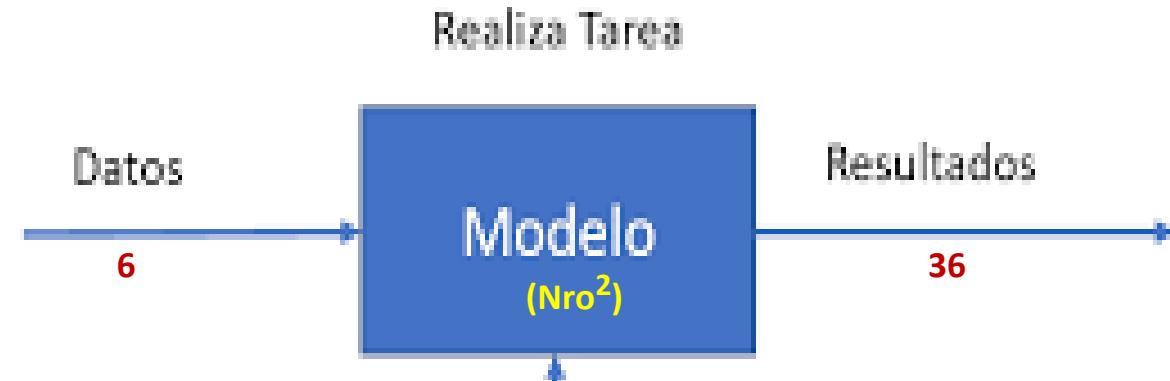
Es el proceso mediante el cual se usan modelos matemáticos de datos para ayudar a un equipo a aprender sin instrucciones directas. El aprendizaje automático usa algoritmos para identificar patrones en los datos, y esos patrones luego se usan para crear un modelo de datos que puede hacer predicciones. Con más experiencia y datos, los resultados del aprendizaje automático son más precisos, de forma muy similar a cómo los humanos mejoran con más práctica.



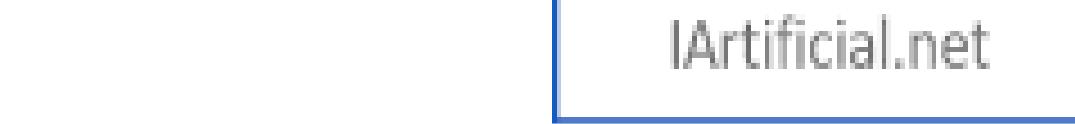


Aprendizaje automático

Modelo de aprendizaje



Fase de predicción o inferencia.- A la máquina se le da nuevos datos, ésta infiere los resultados asociados a estos datos, en base a lo aprendido.

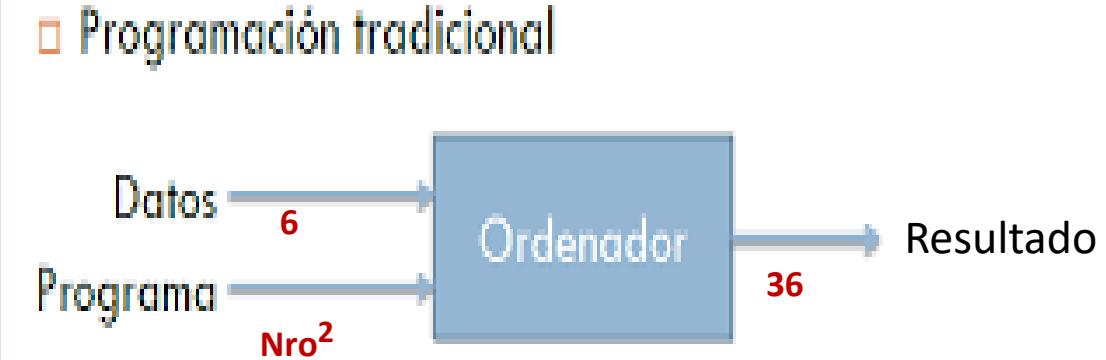
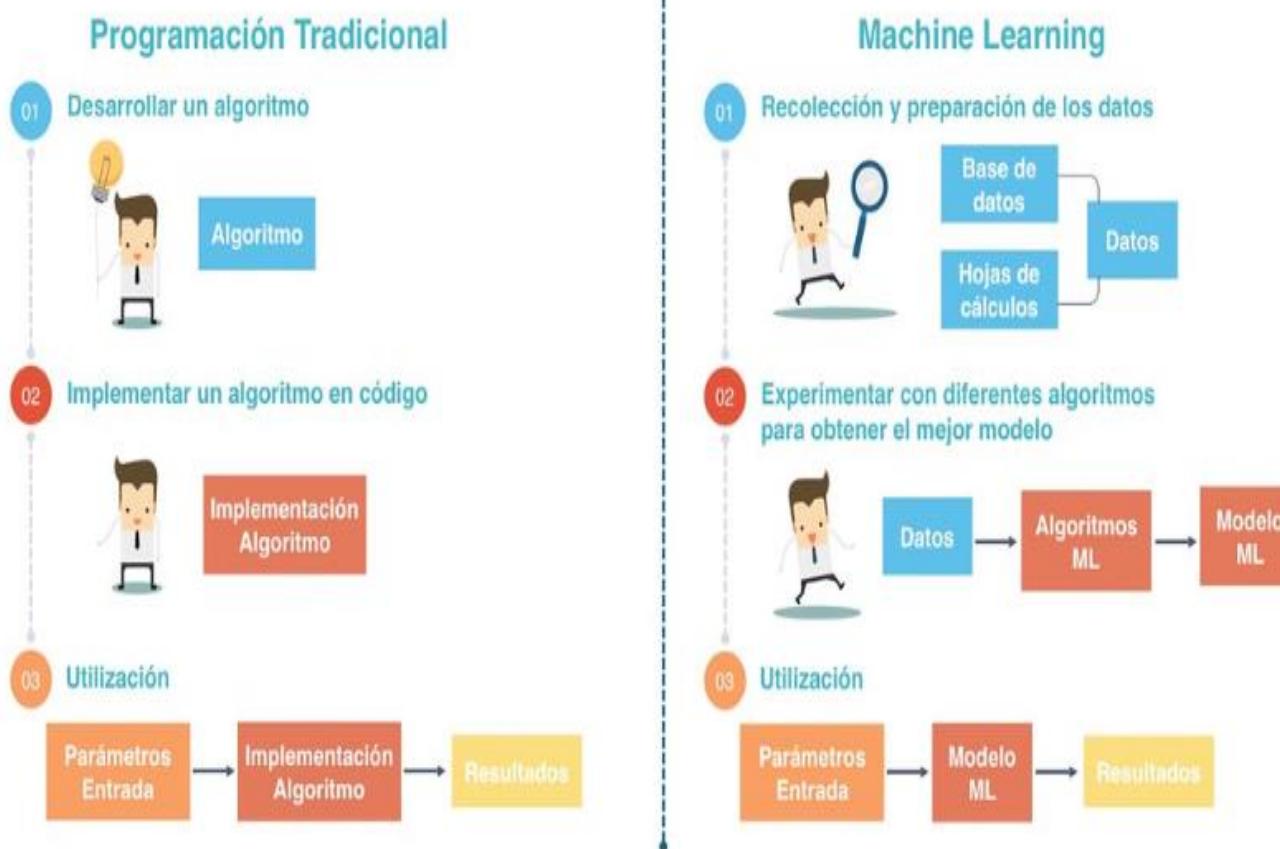


Fase de aprendizaje.- Se entrena la máquina para que aprenda, en base a los datos y los resultados asociados a esos datos.



Aprendizaje automático

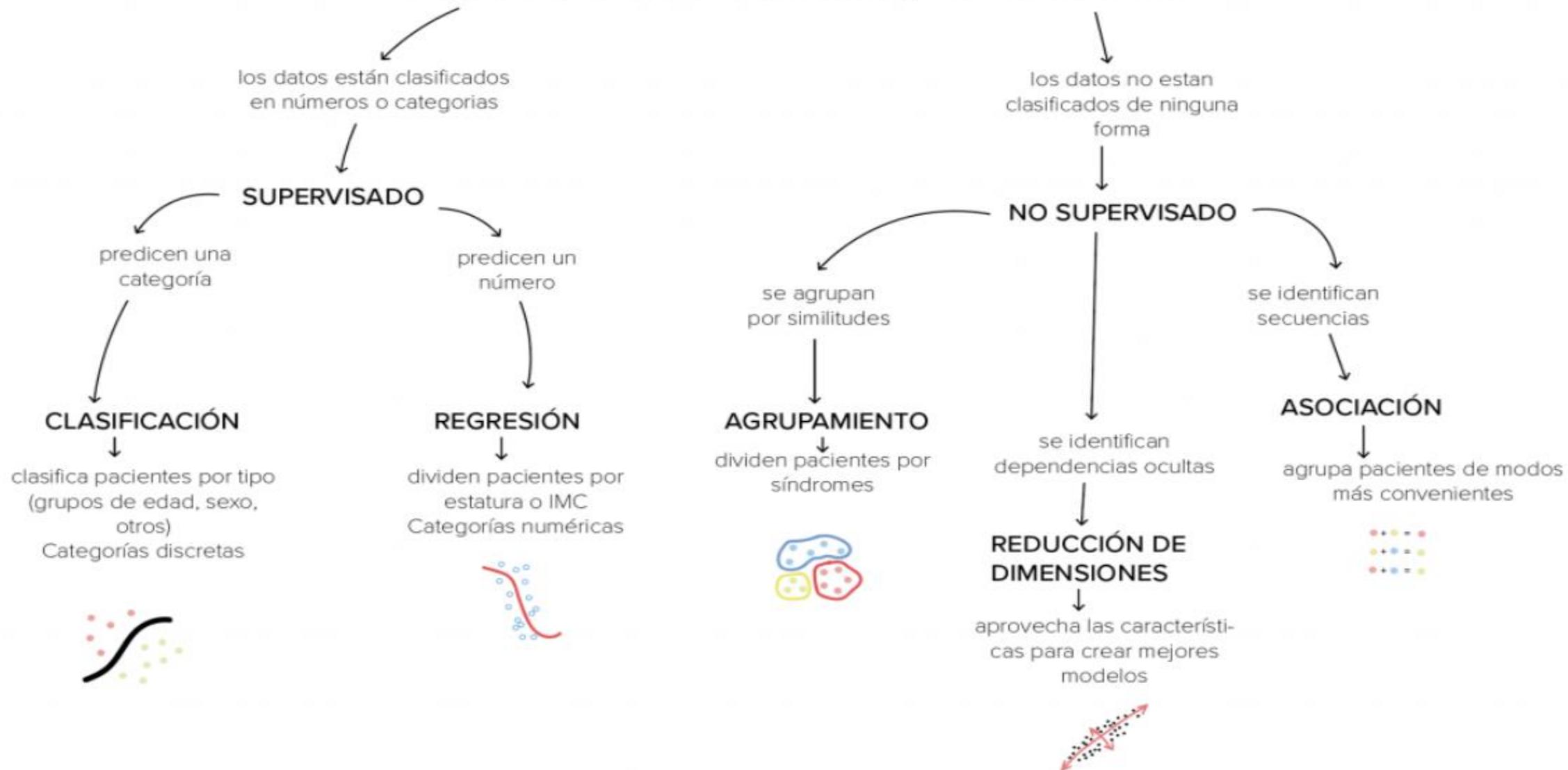
Programación tradicional vs. Machine Learning



<https://aprendeia.com/diferencia-entre-machine-learning-y-la-programacion-tradicional/#:~:text=Sin%20que%20nadie%20programe%20la%20l%C3%B3gica%2C%20en%20a%20programaci%C3%B3n%20tradicional,lo%20que%20es%20muy%20potente.>

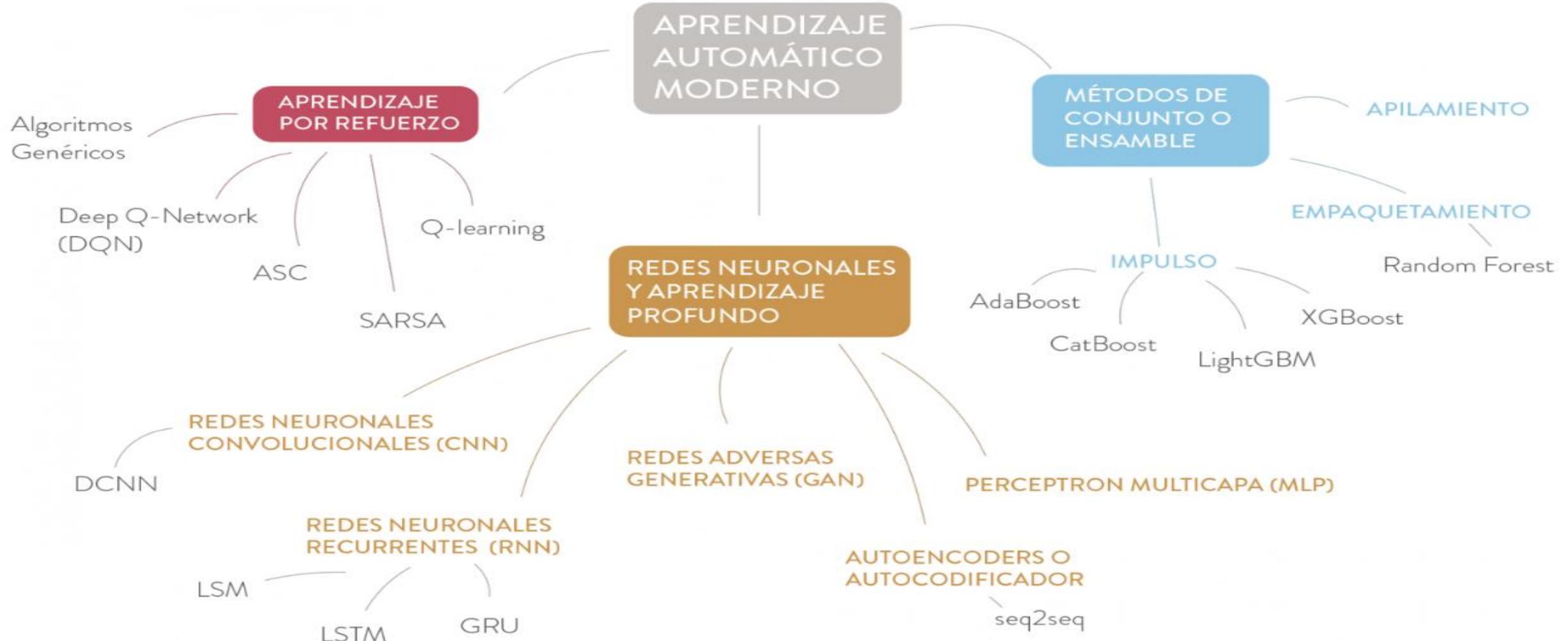
Tipos de aprendizaje automático

APRENDIZAJE AUTOMÁTICO CLÁSICO



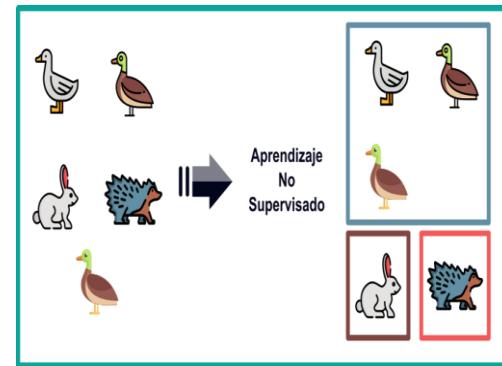
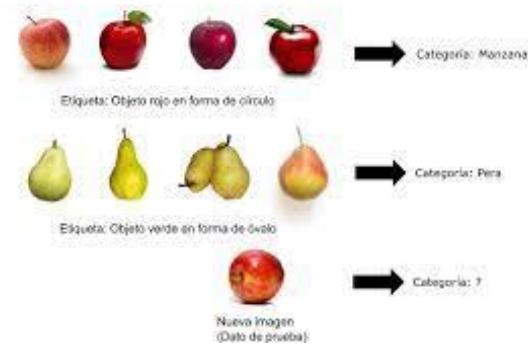
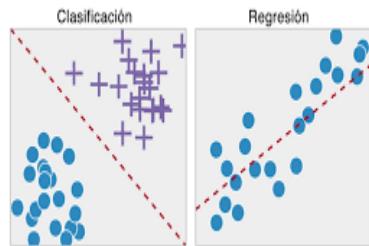
Tipos de aprendizaje automático

Se plantea una taxonomía de Aprendizaje automático Moderno con fines didácticos



A

Taxonomía convencional del aprendizaje automático



Supervisado

El aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento.

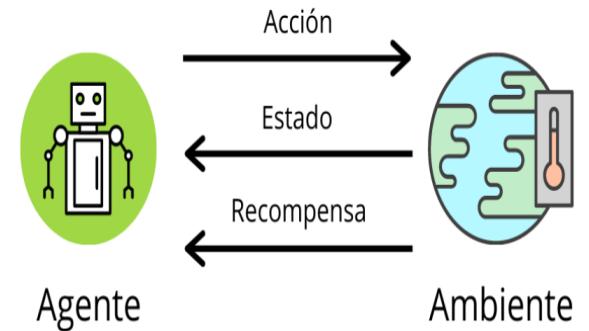
Los datos de entrenamiento consisten de pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados.

https://es.wikipedia.org/wiki/Aprendizaje_supervisado

No supervisado

Se distingue del Aprendizaje supervisado por el hecho de que no hay un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es tratado, de modo que, se establece grupos de objetos en base a patrones comportamiento similar.

https://es.wikipedia.org/wiki/Aprendizaje_no_supervisado



Reforzado

El aprendizaje reforzado intentará hacer aprender a la máquina basándose en un esquema de “premios y castigos” en un entorno en donde hay que tomar acciones y que está afectado por múltiples variables que cambian con el tiempo.

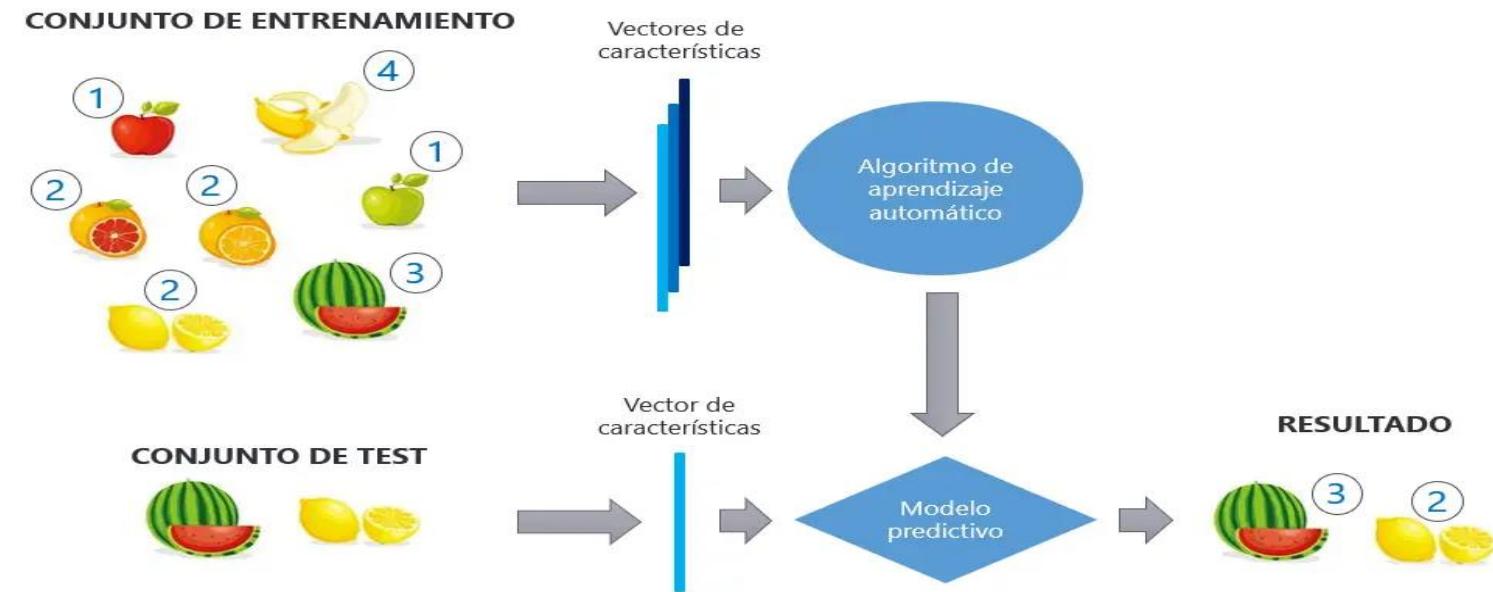
<https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/>

A Paradigmas del aprendizaje automático



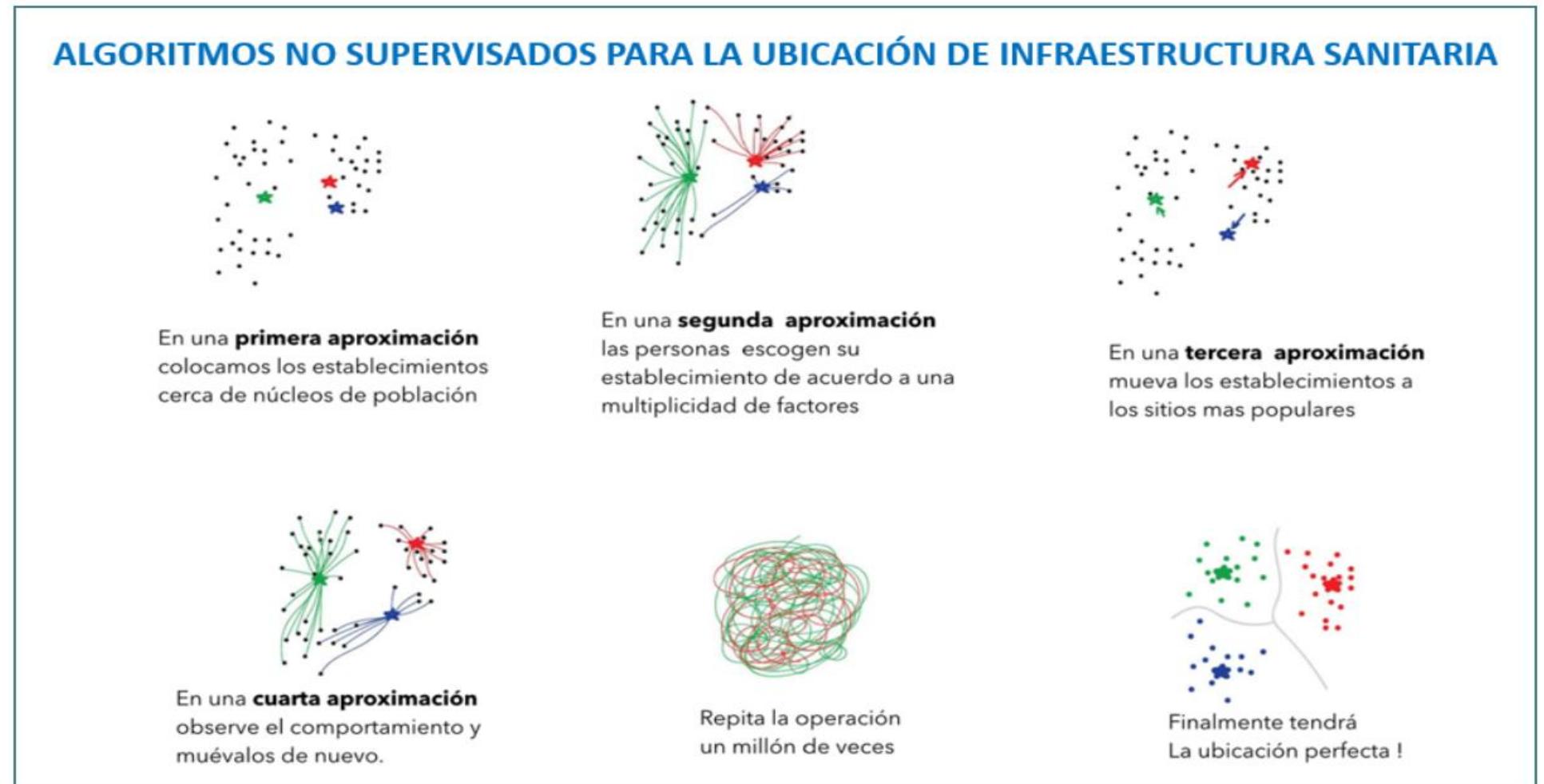
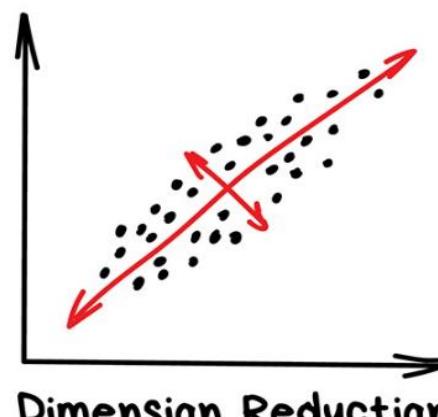
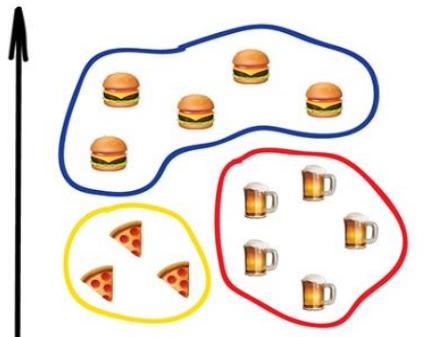
Aprendizaje supervisado

el aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados.



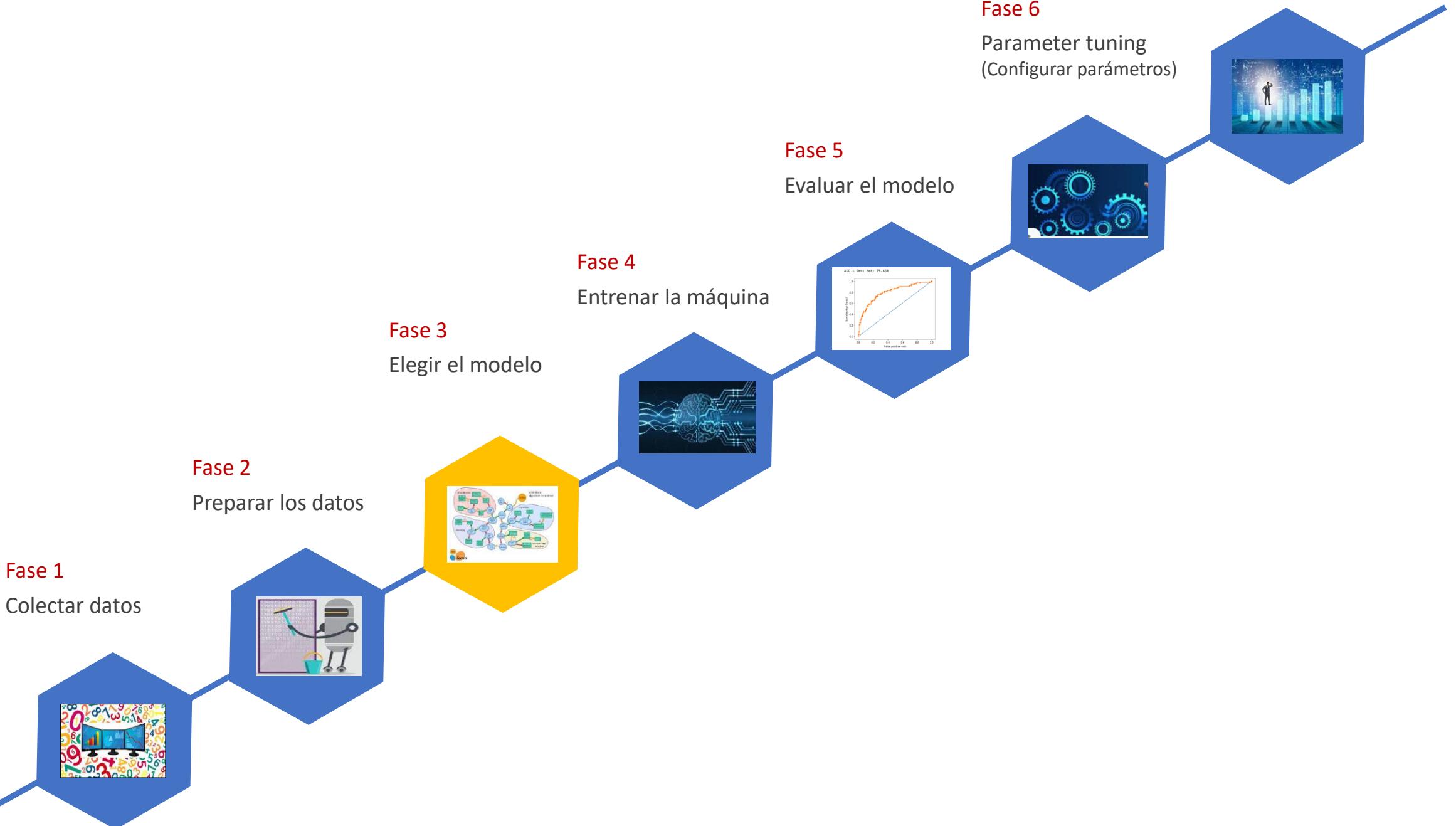
AI Algoritmos del aprendizaje automático no supervisado

Algoritmo no supervisado K-means para ubicar centroides de puntos geográficos y ubicar establecimientos de salud de forma eficiente.



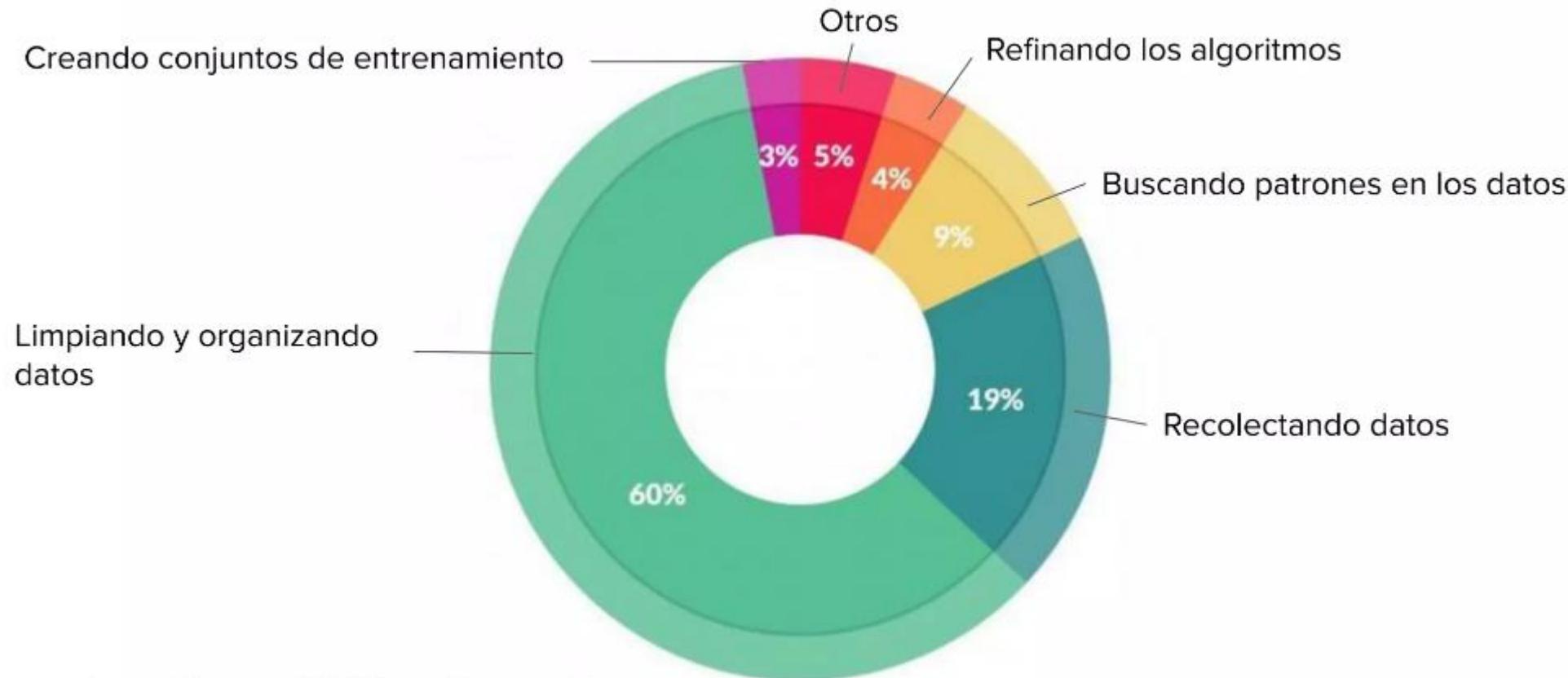
Uso de algoritmo K-means en el campo de la salud

FASES DEL APRENDIZAJE AUTOMÁTICO



Fases del aprendizaje automático: Perspectivas de tiempo

¿En qué se invierte más tiempo?



Origen: [CrowdFlower - 2016 Data Science Report](#)



Fases del aprendizaje automático

Fase 1

Conjunto de datos

Repositorios de datos para ML

brightdata.es

brightdata

Proxy Scraping Datos Web Documentación Precios ES Comuníquese con Ventas Iniciar sesión Prueba gratuita

Reciba conjuntos de datos confiables, completos y estructurados.

Obtenga conjuntos de datos estructurados y precisos para todos los casos de uso. Si aún no encuentra el conjunto de datos que necesita en nuestra tienda, generaremos un conjunto de datos personalizado.

Conjuntos de datos >



datosabiertos.gob.pe

gob.pe Plataforma Nacional de Datos Abiertos

Datos Abiertos

Marco de Gobernanza de Datos del Estado Peruano está constituido por instrumentos técnicos y normativos que establecen los requisitos mínimos que las entidades de la Administración Pública deben implementar conforme a su contexto legal, tecnológico y estratégico para asegurar un nivel básico y aceptable para la recopilación, procesamiento, publicación, almacenamiento y apertura de los datos que administre.

COVID-19 Exprésate Perú

12823 Distribución de Datos

Tipos de contenido

- Recurso (9227)
- Dataset (3360)
- Entidades (204)
- Harvest Source (27)
- Página (3)
- Data Dashboard (1)
- Data Story (1)

Categorías

- Economía y Finanzas (1203)

Search Ordenar por Pedido

Buscar Fecha cambiada Descendente Consultar Reiniciar

RENIEC - Certificados digitales emitidos y revocados de agente automatizado [Registro Nacional de Identificación y Estado Civil]

RENIEC - Registro Nacional de Identificación y Estado Civil
Gobernabilidad

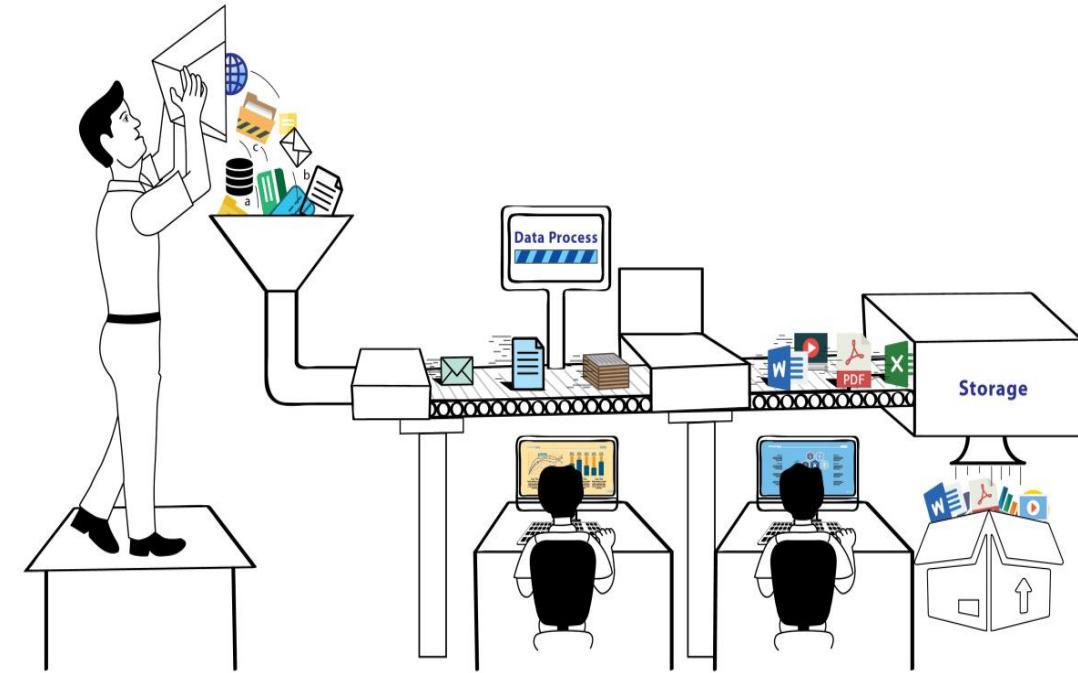
6x csv pdf



Fases del aprendizaje automático

Fase 2 Preprocesamiento de datos

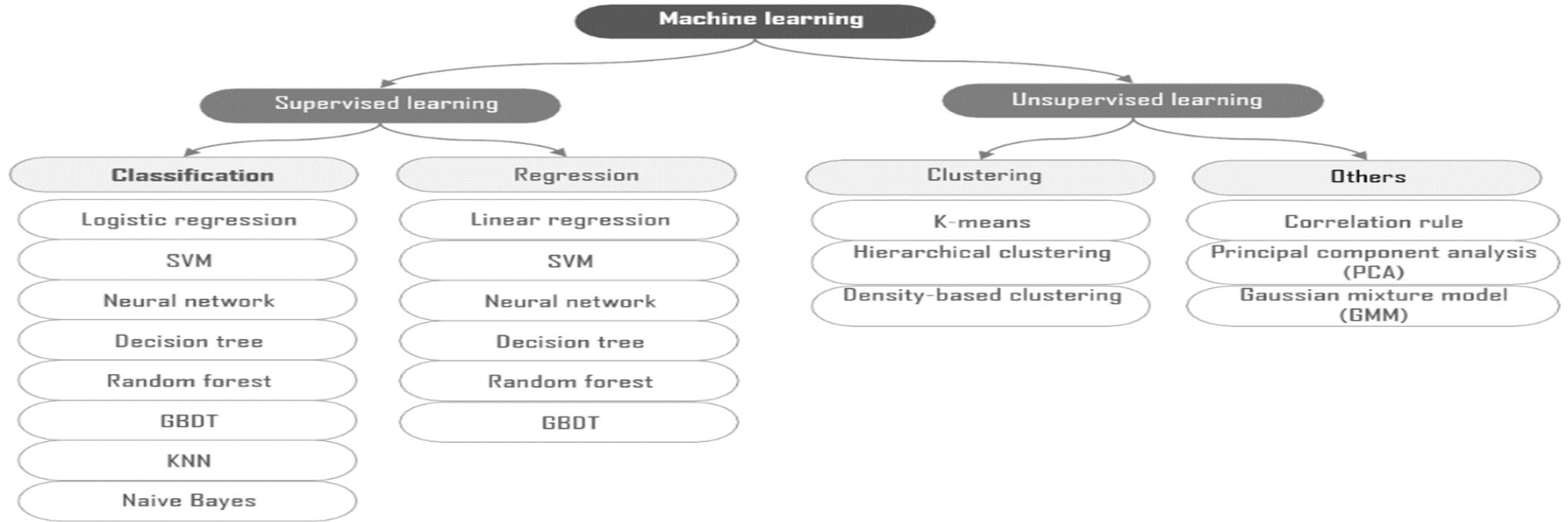
- Reducción de datos
- Limpieza y tratamiento de datos
 - Eliminación de datos duplicados
 - Tratamiento de Outliers
 - Detección de inconsistencia de datos
 - Tratamiento de valores faltantes
 - Transformación de datos categóricos
- Tratamiento de desbalance de datos (Under-sampling)
- Transformación de datos
 - Evaluación normalización
 - Estandarización
- Reducción de dimensionalidad
 - Aplicación de análisis de componentes principales (PCA)





Fases del aprendizaje automático

Fase 3 Elegir el modelo



Antes de elegir el modelo, se debe comprender completamente el problema; también, es importante conocer la semántica, la naturaleza y la distribución de los datos; así mismo, es imperativo identificar el tipo de problema, si éste es de clasificación, regresión, agrupamiento u otra tarea específica.

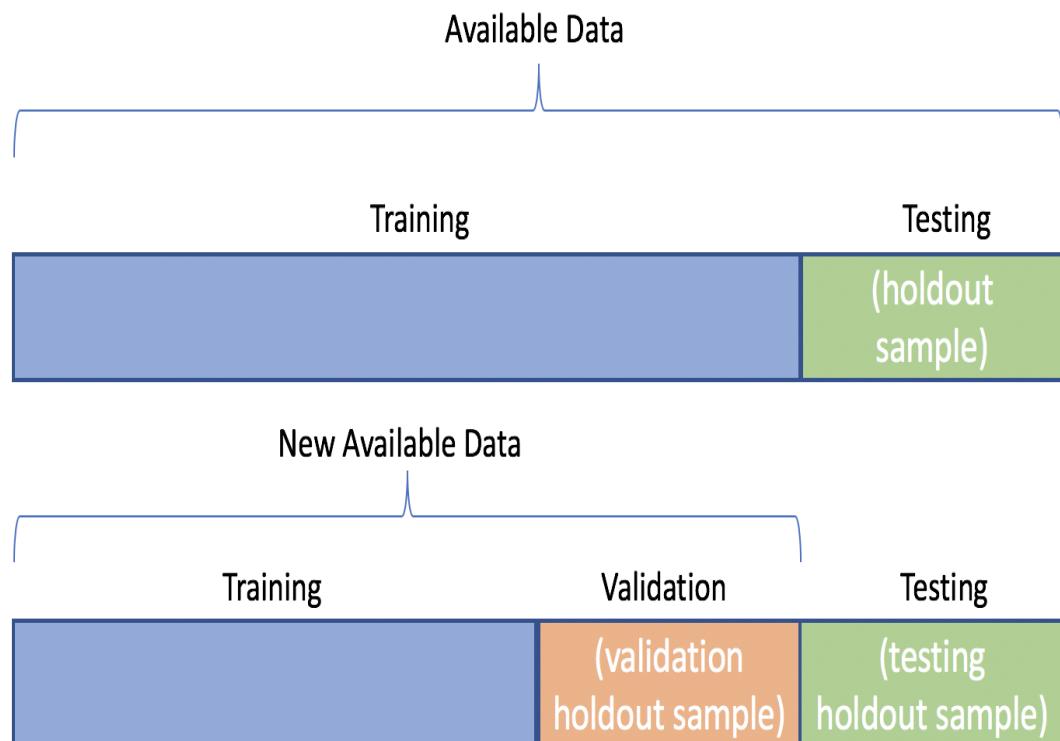


Fases del aprendizaje automático

Fase 4

Entrenar la máquina

El conjunto de datos se puede dividir en 2 o 3 partes. La elección dependerá de factores como el tamaño del conjunto de datos, la complejidad del modelo, la disponibilidad de recursos y la necesidad de ajustar hiperparámetros. La división en 3 partes suele ser preferible en proyectos más complejos donde se necesite un ajuste fino de hiperparámetros y una evaluación rigurosa del rendimiento del modelo.



- Conjunto de Entrenamiento (Training Set):**
Se utiliza para entrenar el modelo, ajustando sus parámetros y aprendiendo patrones de los datos. Los datos de entrenamiento son la base para que el modelo aprenda y ajuste sus parámetros durante el entrenamiento.
- Conjunto de Evaluación o Validación (Validation Set):**
Se utiliza para ajustar hiperparámetros y evaluar el rendimiento del modelo durante el entrenamiento. Después de cada iteración de entrenamiento, se evalúa el modelo en el conjunto de evaluación para evitar el sobreajuste a los datos de entrenamiento y ajustar hiperparámetros.
- Conjunto de Prueba (Test Set):**
Se utiliza para evaluar el rendimiento final del modelo después de que el entrenamiento ha concluido. El modelo no ha visto los datos del conjunto de prueba durante el entrenamiento ni la evaluación. Se utiliza para obtener una estimación no sesgada del rendimiento del modelo en datos no vistos.



Fases del aprendizaje automático

Fase 5

Evaluar el modelo en problemas de clasificación (Matriz de confusión)

		Valor real	
		Positivos	Negativos
Valor predicción	Positivos	Verdadero positivo	Falso positivo
	Negativos	Falso negativo	Verdadero negativo

- Verdadero positivo:** Instancia positiva que se clasificó correctamente como positiva.
- Verdadero negativo:** Instancia negativa que se clasificó correctamente como negativa.
- Falso positivo:** Instancia negativa que se clasificó incorrectamente como positiva.
- Falso negativo:** Instancia positiva que se clasificó incorrectamente como negativa.

		Valor real	
		Positivos	Negativos
Valor predicción	Positivos	76	28
	Negativos	18	78
		94	106



Fases del aprendizaje automático

Fase 5

Evaluar el modelo en problemas de clasificación (Matriz de confusión - Métricas)

		Valor real		
		Positivos	Negativos	
Valor predicción	Positivos	76	28	104
	Negativos	18	78	96
		94	106	

- **Precisión (precision):** Se calcula dividiendo el número de verdaderos positivos por la suma del número de verdaderos positivos y el número de falsos positivos

$$\text{Precision} = \frac{\text{VeraderosPositivos}}{\text{VerdaderosPositivos} + \text{FalsosPositivos}}$$

- **Exhaustividad (recall):** Se calcula dividiendo el número de verdaderos positivos por la suma del numero de verdaderos positivos y el número de falsos negativos

$$\text{Recall} = \frac{\text{VeraderosPositivos}}{\text{VerdaderosPositivos} + \text{FalsosNegativos}}$$



Fases del aprendizaje automático

Fase 5

Evaluar el modelo en problemas de clasificación (Matriz de confusión - Métricas)

		Valor real		
		Positivos	Negativos	
Valor predicción	Positivos	76	28	104
	Negativos	18	78	96
		94	106	

- **Exactitud (accuracy):** Suma del número de verdaderos positivos y el número de verdaderos negativos dividido por el número total de ejemplos
- **Tasa de error (error rate):** Se calcula restando la exactitud del valor 1
- **Ratio de Falsos Positivos:** Se calcula dividiendo el número de falsos positivos y el número de negativos reales



Fases del aprendizaje automático

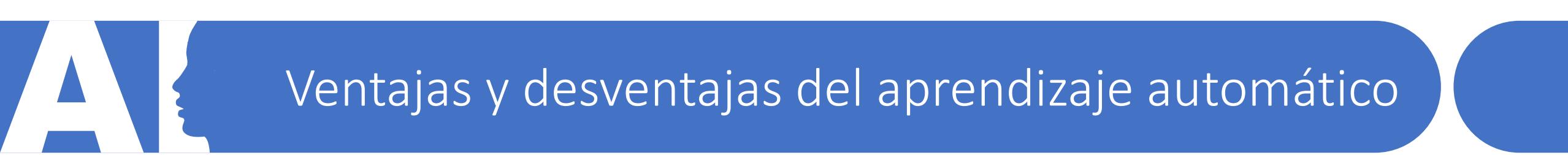
Fase 5

Evaluar el modelo en problemas de clasificación (Matriz de confusión - Métricas)

		Valor real		
		Positivos	Negativos	
Valor predicción	Positivos	76	28	104
	Negativos	18	78	96
		94	106	

- **F1-score** se puede interpretar como un promedio ponderado de la precisión y la exactitud (*recall*), donde el F1-score alcanza su mejor valor en 1 y el peor puntaje en 0.
- La contribución relativa de la precisión y la exactitud al F1-score es la misma [1].
- La fórmula para el puntaje F1-score es:

$$F1 = 2 * \frac{(precision * recall)}{(precision + recall)}$$



Ventajas y desventajas del aprendizaje automático

Lenguajes y librerías

- **Python:** numpy, pandas, SciPy, scikit-learn, Anaconda, TensorFlow, Keras, Theano, etc.
- **R, Octave, MatLab, etc:** CRAN, Rweka, nnet, rpart, tree, CORElearn, etc.
- **Java:** WEKA, rapid miner, ELKI, MOA, Java-ML, etc.
- **Javascript:** mljs, ConvNetJS, Synaptic, Mind, etc.
- **C, C++:** Shark, Shogun, libsvm, mlpack, Plearn, MLC++, etc.
- **C#:** Aforge.net, IKVM, infer.NET

