

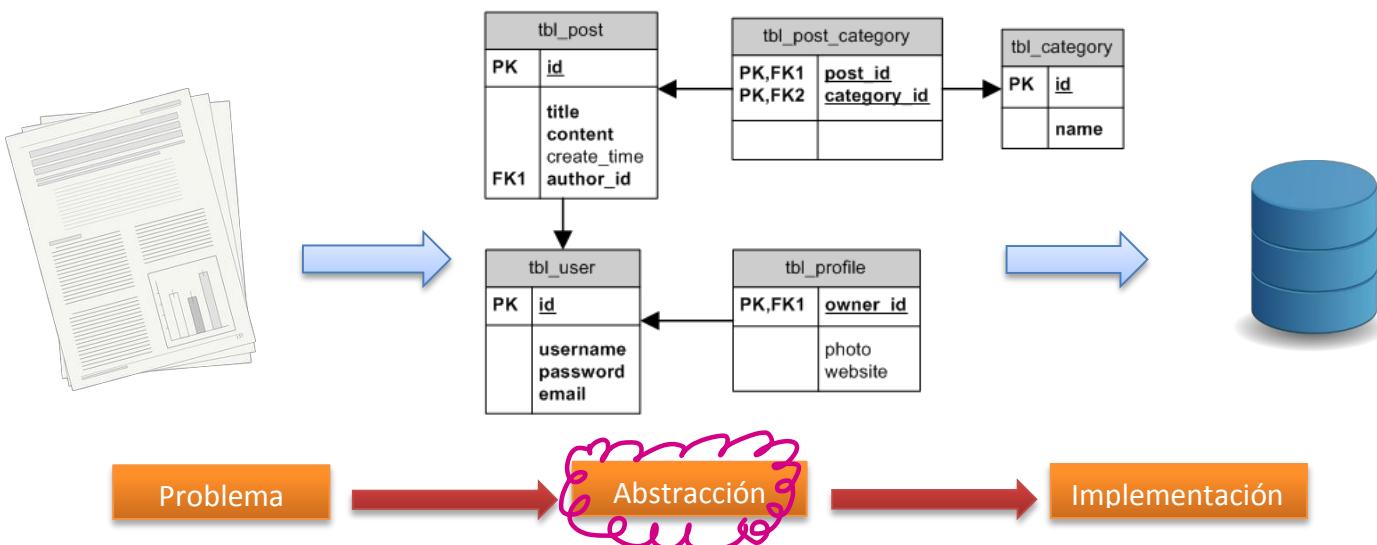
**TEMA 2**  
**MODELOS DE DATOS.**

**2.1. MODELOS DE DATOS.**

- Uno de los principales problemas en el diseño de base de datos es que tanto usuarios como ingenieros, programadores y diseñadores visualizan los datos de manera distinta.
- La construcción de un modelo de datos permite contar con una visión única y correcta de la estructura y comportamiento de los datos aplicada a un problema real: "Problema de dominio".

*Modelo de datos:*

- Es una representación generalmente gráfica que describe la estructura de los datos y sus características acerca de un problema real o caso de estudio.
- Representa una abstracción del problema a resolver.
- Ayuda a entender el contexto y el entorno de dicho problema.
- Representa una herramienta de comunicación entre los involucrados en el proyecto asociado con el diseño de una base de datos.



**2.2. ELEMENTOS DE UN MODELO DE DATOS (INDEPENDIENTE DE SU TIPO):**

Cualquier modelo de datos debe contar con los siguientes elementos básicos:

- Entidades
- Atributos
- Relaciones entre entidades.
- Restricciones

**2.2.1. Entidad**

Cualquier cosa a partir de la cual se realiza obtención de datos. Representa a un objeto del mundo real los cuales pueden ser de 2 tipos:

## TEMA 2

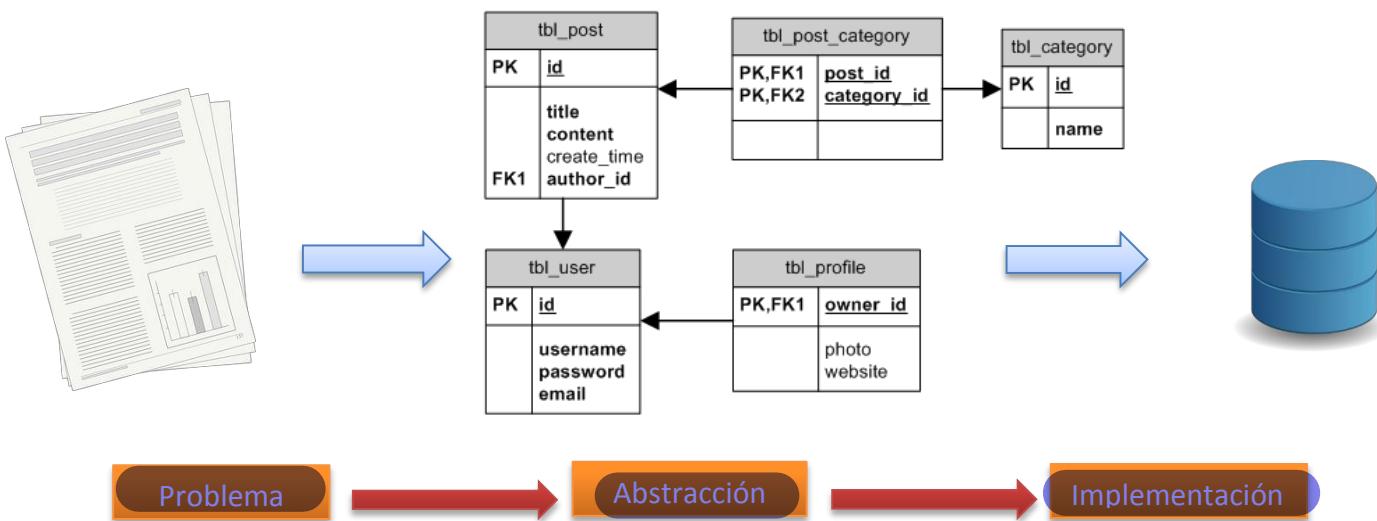
### MODELOS DE DATOS.

#### 2.1. MODELOS DE DATOS.

- Uno de los principales problemas en el diseño de base de datos es que tanto usuarios como ingenieros, programadores y diseñadores visualizan los datos de manera distinta.
- La construcción de un modelo de datos permite contar con una visión única y correcta de la estructura y comportamiento de los datos aplicada a un problema real: "Problema de dominio".

**Modelo de datos:**

- Es una representación generalmente gráfica que describe la estructura de los datos y sus características acerca de un problema real o caso de estudio.
- Representa una abstracción del problema a resolver.
- Ayuda a entender el contexto y el entorno de dicho problema.
- Representa una herramienta de comunicación entre los involucrados en el proyecto asociado con el diseño de una base de datos. **para entender y estandarizar**



#### 2.2. ELEMENTOS DE UN MODELO DE DATOS (INDEPENDIENTE DE SU TIPO):

Cualquier modelo de datos debe contar con los siguientes elementos básicos:

- Entidades
- Atributos
- Relaciones entre entidades.
- Restricciones

##### 2.2.1. Entidad

Cualquier cosa a partir de la cual se realiza obtención de datos. Representa a un objeto del mundo real los cuales pueden ser de 2 tipos:

- **Objetos físicos:** Aquellos que podemos ver y/o tocar.
  - Una silla, una escuela, un pizarrón, etc.
- **Abstracciones:** Aquellos objetos que no podemos ver ni tocar.
  - Un viaje, una reservación, una inscripción, un curso, un certificado, etc.

Generalmente se identifican a través de **sustantivos en singular**. Una estrategia sencilla para detectar las posibles entidades es **subrayar todos los sustantivos de un caso de estudio**.

### **2.2.2. Instancia de una entidad:**

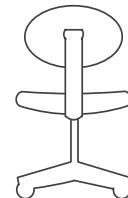
Mientras que una entidad se define a través de sus atributos, una **instancia** de dicha entidad, es representada por un **objeto en particular** en donde **se conoce el valor de cada uno de sus atributos**.

- Cada instancia debe **distinguirse de todas las demás**
- Todas las instancias **deben tener los mismos atributos**.

#### Ejemplos:

Entidad:

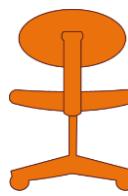
Silla (descripción, tipoMaterial, color, estilo, peso);



Instancias:

Silla 1 ('De comedor', 'De Madera', 'Café rustico', 5.6);

Silla 2 ('Reclinable', 'Metálica con piel', 'Negra', 6.4);



Entidad:

Viaje (nombre, origen, destino, fechaSalida);

Instancias:

Viaje 1 ('Excursión de graduación ', 'México DF', 'Acapulco', 1, '01/01/2011');

### **2.2.3. Atributo**

Un atributo es una **característica de una entidad**. Un atributo es similar a un “campo” (field) en un sistema de archivos. Normalmente las **entidades se definen en términos de su lista de atributos**:

#### Ejemplos:

- Silla (material, color, peso).
- Escuela (num\_salones, dirección, tipo).
- Curso (nombre, nivel, fecha de inicio, fecha fin, cupo máximo).
- Viaje (origen, destino, duración, fecha salida, itinerario).

Un error común que ocurre al momento de identificar los elementos de un modelo de datos es la posibilidad de **confundir a una entidad de un atributo o viceversa**. Los siguientes puntos pueden considerarse para facilitar y realizar la identificación de forma correcta:

¿Cómo identificar correctamente a una entidad?



Característica a verificar	Descripción
Duplicados	Revisar si existen sustantivos que representan <b>sinónimos</b> . Estos deberán <b>descartarse</b> . Por ejemplo, <b>Empleado</b> , <b>Trabajador</b> , etc.
Cardinalidad	En algunas ocasiones se <b>detecta</b> que <b>existirá una sola instancia</b> de la <b>entidad</b> propuesta. No tiene sentido contar con una entidad que solo contará con una instancia por lo que puede ser <b>descartada</b> .
Roles	<p></p> <p>En algunas ocasiones se detecta la ocurrencia de <b>roles asociados a una misma entidad</b>. Por ejemplo, la entidad Empleado cuenta con 3 roles: Administrador, Investigador, Profesor. En el tema 5 se revisará este escenario. Por <b>ahora</b> es posible aplicar el siguiente criterio:</p> <ul style="list-style-type: none"> <li>Si los roles cuentan con <b>atributos particulares</b> se conservan, de lo contrario se eliminan.</li> <li>En el tema 5 se revisa la forma en la que se realiza el modelado de estos roles particularmente cuando definen sus propios atributos.</li> </ul>
Ambigüedades	Evitar la definición de entidades que <b>no representen a un objeto</b> (sustantivo) o que el sustantivo <b>no defina a un objeto de interés</b> para el caso de estudio. Ejemplos de palabras ambiguas: "catálogo", "revisar", "histórico", etc.

- ¿Cómo evitar confundir a un atributo de una entidad?



Característica a verificar	Descripción
Número de atributos.	<ul style="list-style-type: none"> <li>Si se identifican <b>al menos 2 atributos asociados</b> a un mismo objeto, es factible proponer una nueva entidad.</li> <li>Si solo se identifica <b>un solo atributo</b>, <small>depende</small> en algunas situaciones puede ser factible proponer una nueva entidad a pesar de tener un solo atributo cuando ocurre alguna de las siguientes situaciones. <ul style="list-style-type: none"> <li><b>El atributo aparece en varias entidades.</b> Esto significa que sus valores pudieran repetirse varias veces en distintos lugares. En este caso se crea una entidad y se le asigna un identificador. En lugar de repetir N veces los valores del atributo, se crea una relación entre la nueva entidad con las N entidades donde aparece el atributo y se propaga dicho identificador.</li> <li><b>El atributo aparece en una sola entidad.</b> Si los valores del atributo se aparecen con alta frecuencia, es posible crear una nueva entidad y relacionarlas a través de un identificador.</li> <li><b>Los valores del atributo cambian con alta frecuencia.</b> Al tener múltiples copias del mismo valor ya sea en la misma o en diferentes entidades, un cambio de valor implica cambiar todas las copias. Esto implica complejidad y aumenta la posibilidad de inconsistencias. Por lo tanto, se justifica crear una entidad con un identificador y su atributo.</li> <li><b>Los valores del atributo son de longitud relativamente grande.</b> A mayor longitud o tamaño de los valores de un atributo requiere mayor almacenamiento y mayor procesamiento para almacenar un mismo valor de forma repetida aumentando la posibilidad de inconsistencias si sus valores son susceptibles a cambios.</li> </ul> </li> </ul>

Objeto  
obj\_id  
nombre Atributo

### Estudiante

Lugar Nacimiento : 12

- ① aguascal
- ② nevoleon
- ③ colmx
- ④ zac

### Profesor

Lugar Nacimiento 8

### Escuela

Entidad 16

### Auto

color

15

blanco  
negro  
azul marino  
azul agua

Se repiten  
demasiado y  
creamos entidad

normalizar  
a alto nivel      vs      desempeño

Ejemplos:

Suponer las siguientes entidades que fueron detectadas para realizar la administración de una empresa que cuenta con una cadena de agencias de autos distribuidas en todo el mundo. Existen 3 tipos de agencias A, B y C.

*a*      *e*      *e*      *e*      *a*

Auto (num\_serie, marca, modelo, pais\_manufactura, precio)  
 Cliente (RFC, CURP, email, calle, colonia, municipio, estado)  
 Agencia (clave, calle, colonia, municipio, estado, pais, tipo)

¿Qué atributos podrían ser considerados como entidades para evitar los problemas mencionados anteriormente?

R: Marca, Modelo, País, Estado, Municipio, Colonia, tal vez Calle.

Los siguientes puntos permiten decidir si se consideran como entidades.

- Los valores de estos atributos pueden repetirse con gran frecuencia.
- Sus valores son cadenas relativamente grandes.
- Los valores de estos atributos se modifican con muy baja frecuencia. Esto podría minimizar la posibilidad de inconsistencia.
- Si los requerimientos indican la necesidad de contar con un control estricto de la dirección y es crítico minimizar inconsistencias, la separación y el uso de entidades (catálogos) puede ser opción. Como se mencionó anteriormente, esta decisión puede tener impactos en desempeño.

#### 2.2.4. Relaciones entre entidades

Generalmente entidades relacionadas

- Prácticamente en cualquier problema o caso de estudio a partir del cual se requiere generar un modelo de datos, las entidades involucradas siempre estarán relacionadas entre sí.
- Para determinar la forma en la que se relacionan las entidades, es necesario determinar el tipo de relación que existe entre ambas.

#### Tipos de relaciones:

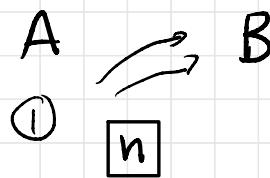
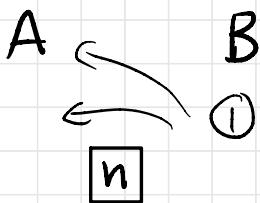
Tipo de relación	Notación
Uno a Uno (one to one)	1:1 ó 1..1
Uno a Muchos (one to many)	1:M ó 1..*
Muchos a Muchos (many to many)	M:N ó *..*

#### ¿Cómo identificar el tipo de relación entre 2 entidades?

- Para cada par de entidades A y B, se realiza las siguientes verificaciones:
  - ¿Cuántas instancias de A se asocian con una instancia de B?
  - ¿Cuántas instancias de B se asocian con una instancia de A?
- Las respuestas a estas preguntas, determinan el tipo de relación:



Respuestas:	Tipo de relación existente entre A y B
1:1 , 1:1	Uno a Uno
1:1 , 1:M	Uno a Muchos
1:M , 1:M	Muchos a Muchos



cuantos de A con B | cuantos de B con A

Caso 1

$$\begin{array}{ccc} 1:1 & 1:1 & \Rightarrow 1:1 \\ \text{Caso 2} \\ 1:1 & 1:M & \Rightarrow 1:M \end{array}$$

Caso 3

$$1:M \quad 1:1 \quad \Rightarrow 1:M$$

Caso 4

$$1:M \quad 1:M \quad \Rightarrow M:N$$

**Ejemplos:**

- Relación Uno a Muchos:

Entidades Artículo, Periodista.

- Un periodista escribe muchos artículos  $\rightarrow$  1:M
- Un artículo es escrito por un periodista  $\rightarrow$  1:1

- Relación Muchos a Muchos

Entidades Curso, Estudiante

- A un curso se inscriben muchos estudiantes.  $\rightarrow$  1:M
- Un estudiante puede inscribirse a varios cursos.  $\rightarrow$  1:M

M:N

- Relación Uno a Uno

Entidades Credencial, Estudiante.

- Un estudiante cuenta con una credencial vigente.  $\rightarrow$  1:1
- Una credencial vigente pertenece a un solo estudiante.  $\rightarrow$  1:1

1:1

Como se puede observar, para determinar cada pareja de enunciados, estos deben obtenerse de la naturaleza y de las reglas de negocio del caso de estudio a modelar.

**2.2.5. Restricciones (Constraints).**

- Es una condición que se aplica a una entidad o a sus atributos.
- Su objetivo es mantener la integridad de los datos.
- Se identifican a través del análisis del caso de estudio apoyando se en las reglas de negocio.
- Algunas restricciones pueden ser implementadas y verificadas de forma automática por un DBMS al realizar alguna operación sobre los datos.
- Otras restricciones se pueden programar ya sea empleando un sistema de software, o empleando una extensión o lenguaje que soporte la propia BD. Ejemplo: PL/SQL.

**2.3. REGLAS DE NEGOCIO (RN).**

- Son enunciados breves, concisos, sin ambigüedades que describen una política, regla, hecho, principio o procedimiento dentro de una compañía u organización.
- Son de vital importancia para lograr los objetivos de dicha organización.
- Se encuentran inmersas en los procesos de negocio de la organización.
- Los enunciados deben ser escritos con un lenguaje natural, claramente entendibles para los involucrados.
- Los enunciados deben ser atómicos, es decir, lo más simple posible, de tal forma que un enunciado ya no pueda ser dividido en sub-enunciados.

¿Por qué son tan importantes las RN en el diseño de bases de datos?

- Ayudan a definir los elementos de un modelo de datos: Entidades, restricciones y relaciones entre entidades.
- Ayudan a estandarizar la visión de la organización con respecto a la administración de sus datos.
- Ayudan al diseñador a entender la naturaleza de los datos.
- Permiten la creación de modelos de datos adecuados.

**Ejemplo:**

Considere el siguiente caso de estudio asociado con un sistema de renta de automóviles. Realice una lectura cuidadosa del caso de estudio, determinar los elementos del modelo de datos:

- Entidades
- Reglas de negocio
- Relaciones entre entidades

#### **Caso de estudio, EU-Rent.**

##### *Modelo de negocio de EU-Rent:*

EU-Rent es una compañía que se dedica a la renta de autos.



##### Sucursales:

La compañía cuenta con más de 1000 sucursales en ciudades del país. Las sucursales están distribuidas en 3 diferentes regiones: región A, región B y región C. Una región está integrada por varias ciudades, y en una ciudad pueden existir varias sucursales.

##### Personal de las sucursales:

Cada sucursal cuenta con un gerente y con varios agentes de ventas encargados de atender a los clientes. Adicionalmente, cada sucursal cuenta con hasta 3 ingenieros mecánicos para dar mantenimiento a los autos. Los gerentes solo pueden administrar a una sucursal, aunque los agentes, de así solicitarlo pueden trabajar en diferentes sucursales. Los empleados no pueden realizar rentas de autos.

##### Autos:

Cada sucursal cuenta con un catálogo de autos para rentar. Un auto solo pertenece a una sucursal.

Cada 3 meses o cada 10000 Km, el auto debe recibir servicio de mantenimiento. El auto puede tener varios servicios a lo largo de su vida útil.

##### Clientes:

Se requiere almacenar los siguientes datos del cliente: nombre, apellido paterno, apellido materno, email, RFC, dirección y teléfono.

##### Rentas:

Un cliente puede rentar hasta 3 autos al mismo tiempo (máximo 3 por renta). Se registran los datos de la renta: fecha de solicitud, periodo de tiempo de la renta el cual puede ser hasta por un mes, el cliente que la solicita, y se asocian los autos que el cliente selecciona. Si el auto esta en servicio, este no podrá ser rentado.

##### Facturas:

Cada vez que el cliente realice una renta, se genera una factura, se requiere almacenar los siguientes datos: monto total, renta asociada y la fecha de elaboración.

##### Solución:

#### **A. Detección de entidades.**

Observando los sustantivos que aparecen en el caso de estudio, podemos listar las siguientes entidades candidatas:

- Sucursal
- Ciudad
- Región (aparentemente, esta entidad podría confundirse con un atributo, sin embargo no lo es, ya que por lo menos tiene 2 atributos: la clave de la zona A,B, etc., y las ciudades que pertenecen a ella).
- Empleado (Gerente)
- Empleado (Agente de venta)
- Cliente

# I detección entidades

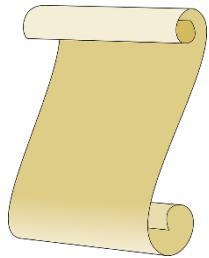
- e sucursal \* duplicado
- a ~~región~~ Major se asocia con ciudad
- e auto
- e cliente
- e serv. marr
- empleado** ✓✓
- e ciudad — region - nombre - id
- e factura
- e renta
- a dirección
- ~~gerente~~
- ~~agente~~
- ~~ingeniero~~
- ~~reconico~~ \* > ingeniero mecanico rol //
- empleado** cardinalidad 1
- ~~país~~



- Empleado (Ingeniero).
- Auto
- Renta
- Servicio de mantenimiento
- Factura
- ¿Compañía?, Puede ser una entidad, sin embargo, ¿Cuántas instancias tendría esta entidad?, solo estamos hablando de una compañía, por lo tanto, solo existiría una instancia, o lo que es lo mismo, un registro en la base de datos. Por esta razón no tiene mucho sentido considerarla a menos de que existiera algún requerimiento explícito que nos lo solicitara.

### B. Reglas de negocio:

- i. Las sucursales se distribuyen en varias regiones.
- ii. Un gerente solo puede administrar una agencia.
- iii. Un agente de ventas puede trabajar en una o más sucursales.
- iv. Un empleado no puede rentar autos.
- v. Cada sucursal puede tener hasta 3 ingenieros mecánicos.
- vi. Un auto es asignado a una sola sucursal.
- vii. Un servicio de mantenimiento se debe realizar cada 3 meses o cada 10000 km.
- viii. Un cliente puede rentar máximo 3 autos a la vez.
- ix. Un auto en servicio no puede ser rentado.
- x. Un auto puede ser rentado por un periodo no mayor a un mes.
- xi. Se genera una factura por cada renta.



### C. Relaciones entre entidades:

Considerando la lista de entidades y las reglas de negocio obtenidas en los puntos anteriores, se pueden determinar las siguientes relaciones:

- i. **Sucursal y Región (1:M)**
  - **Una** sucursal se ubica en **una** región. (1:1)
  - En **una** región pueden existir **varias** sucursales. (1:M)
- ii. **Ciudad y Región (1:M)**
  - **Una** ciudad pertenece a **una** región (1:1)
  - **Una** región está formada por **varias** ciudades (1:M)
- iii. **Sucursal y Empleado(Gerente) (1:1)**
  - **Una** sucursal es administrada por **un** gerente. (1:1)
  - **Un** gerente solo puede administrar **una** sucursal. (1:1)
- iv. **Sucursal y Empleado (Ingeniero) (1:M)**
  - **Una** sucursal puede tener hasta **3** ingenieros mecánicos (1:M)
  - **Un** ingeniero trabaja para **una** sucursal (1:1). Como se puede observar, esta última regla no se especifica de forma clara en el caso de estudio. Esta situación es común en la práctica. En esta situación, se emplea el sentido común, y posteriormente se revisa con el experto que conoce las reglas de negocio (usuario final). Aquí, consideramos que un ingeniero, normalmente trabaja para una sola sucursal.

v. **Sucursal y Empleado (Agente) (M:N)**

- **Un** agente de ventas trabaja en una o **varias** sucursales. (1:M)
- **Una** sucursal cuenta con **varios** agentes de ventas (1:M)

vi. **Sucursal y Auto (1:M)**

- **Una** sucursal tiene **varios** autos asignados. (1:M)
- **Un** auto es asignado a **una** sucursal. (1:1)

vii. **Servicio y Auto (1:M)**

- **Un** auto puede recibir **varios** servicios de mantenimiento. (1:M)
- **Un** servicio de mantenimiento es aplicado a **un** auto (1:1)

viii. **Renta y Auto (M:N)**

- **Una** renta puede incluir hasta 3 autos. (1:M)
- **Un** auto puede ser rentado varias veces. (1:M)

ix. **Cliente y Renta. (1:M)**

- **Un** cliente puede hacer **varias** rentas (1:M)
- **Una** renta es solicitada por **un** cliente (1:1)

x. **Factura y Renta (1:1)**

- **Una** Factura es generada para cada renta. (1:1)
- **Una** renta se asocia con una factura. (1:1)



Algunos tips:

- Escribir los enunciados tratando de escribir el lado “uno” o lado “one” del lado izquierdo, invirtiendo solo a las entidades.
- Todos los enunciados deben tener un lado “one”.
- No mezclar más de 2 entidades al invertir el orden de las entidades, por ejemplo, el siguiente caso es incorrecto:
  - **Un cliente** puede solicitar **varias rentas** (1:M)
  - **Una renta** requiere la generación de **una factura** (1:1)
- No confundir atributos con entidades al momento de escribir los enunciados. Por ejemplo, los siguientes enunciados son incorrectos:
  - Un cliente tiene un nombre.
  - Un nombre puede pertenecer a varios clientes.

## 2.4. EVOLUCIÓN DE LOS MODELOS DE DATOS.

Los conceptos explicados hasta el momento, son **independientes al tipo de modelo a emplear**. La siguiente tabla muestra un resumen de los tipos de modelos de datos existentes a lo largo del tiempo.

Tiempo	Nombre
50s < t < 70s	Sistemas de archivos
50s < t < 80s	Modelo Jerárquico
60s < t < 90s	Modelo de Red
70s < t	Modelo Relacional => <b>(RDBMS)</b>
75 < t	Modelo Entidad - Relación
80s < t	Modelo orientado a objetos => <b>(OODBMS)</b>

Tiempo	Nombre
90s < t	<ul style="list-style-type: none"> <li>• <b>Modelo orientado a objeto / Relacional (ORDBMS)</b> <ul style="list-style-type: none"> <li>○ ORMs (Object Relational Mapping)</li> <li>○ Soporte para procesamiento de documentos XML.</li> <li>○ Soporte para procesamiento de documentos JSON</li> <li>○ Bases de datos Multidimensionales</li> </ul> </li> </ul>
2000 < t	<ul style="list-style-type: none"> <li>• Modelos NoSQL</li> </ul>

#### 2.4.1. Sistemas de archivos.

##### 2.4.1.1. Almacenamiento manual.

- Con la finalidad de tener éxito, empresas en general deben desarrollar sistemas para manejar sus tareas de negocio de forma correcta. Históricamente no existían sistemas, dichas tareas se realizaban de forma manual. Los datos de las empresas se escribían en **papel** y se almacenaban en **folder o carpetas almacenadas en grandes estantes**. Esta estrategia se vuelve problemática conforme la empresa crece y por lo tanto la cantidad de datos a mantener. Por lo anterior se opta por hacer uso de **tecnologías computacionales**.



##### 2.4.1.2. Almacenamiento empleando Sistemas Computacionales.

- La generación de reportes empleando sistemas de archivos manuales como los explicados anteriormente tomaba semanas en desarrollarlos.
- Surge entonces un nuevo rol llamado "**Especialista de procesamiento de datos**" (DP) encargado de crear un sistema basado en computadora encargado de almacenar los archivos y generar los reportes.

C_NAME	C_PHONE	C_ADDRESS	C_ZIP	A_NAME	A_PHONE	TP	AMT	REN
Alfred A. Ramas	615-844-2573	218 Fork Rd., Babs, TN	36123	Leah F. Hahn	615-882-1244	T1	100.00	05-Apr-2014
Leona K. Dunne	713-894-1238	Box 12A, Fox, KY	25246	Alex B. Alby	713-228-1249	T1	250.00	16-Jun-2014
Kathy W. Smith	615-894-2285	125 Oak Ln., Babs, TN	36123	Leah F. Hahn	615-882-2144	S2	150.00	29-Jan-2015
Paul F. Olowksi	615-894-2180	217 Lee Ln., Babs, TN	36123	Leah F. Hahn	615-882-1244	S1	300.00	14-Oct-2014
Myron Orlando	615-222-1672	Box 111, New, TN	36155	Alex B. Alby	713-228-1249	T1	100.00	28-Dec-2014
Amy B. O'Brian	713-442-3381	387 Troll Dr., Fox, KY	25246	John T. Okon	615-123-5589	T2	850.00	22-Sep-2014
James G. Brown	615-297-1228	21 Tye Rd., Nash, TN	37118	Leah F. Hahn	615-882-1244	S1	120.00	25-Mar-2015
George Williams	615-290-2556	155 Maple, Nash, TN	37119	John T. Okon	615-123-5589	S1	250.00	17-Jul-2014
Anne G. Farriss	713-382-7185	2119 Elm, Crew, KY	25432	Alex B. Alby	713-228-1249	T2	100.00	03-Dec-2014
Olette K. Smith	615-297-3809	2782 Main, Nash, TN	37118	John T. Okon	615-123-5589	S2	500.00	14-Mar-2015

- La figura anterior muestra un ejemplo de un archivo almacenado en un sistema de cómputo.
- Se establecía un **lenguaje de traducción** para entender el contenido de cada columna. Por ejemplo:
  - C\_NAME = Nombre del cliente.
  - TP = Tipo de seguro, etc.
- Cuando el usuario final requiere datos de dichos archivos, **este se los solicita al DP**.
- Con base a los requerimientos del reporte solicitado, el **DP crea una serie de programas** para obtener los datos de los archivos, manipularlos y presentarlos en el formato solicitado.
- Si el reporte se vuelve a solicitar, el **DP reutiliza** los programas, pero si el usuario cambia requerimientos solicitando formatos más complejos, el **DP deberá crear nuevas versiones** de sus programas.
- Estos programas **definen todas las características y estructura de los archivos**, en un principio esta técnica puede ser buena por la rapidez de los programas al estar hechos totalmente "a la medida" con respecto a las características de un archivo.

- En general, el uso de sistemas de cómputo para almacenar archivos de datos representó un avance significativo, en especial por la decisión de incorporar el uso de tecnologías computacionales para procesar datos, pero el principal problema fue que no se disponía de las herramientas necesarias para convertir los datos en la información que un usuario final necesitaba.
- Lo anterior puede traer diversos inconvenientes como los siguientes:
  - Dificultad para crear nuevos programas los cuales deben adaptarse a archivos y programas existentes, su ejecución se puede volver lenta.
  - Al crear nuevos archivos, fácilmente puede generar duplicación de datos sobre todo al momento de relacionarlos y como consecuencia, el riesgo de inconsistencias.
  - Al existir varias aplicaciones o programas que manejan sus propios archivos, fácilmente pueden caer en una situación de duplicidad al ser dichas aplicaciones independientes entre sí.
  - Control deficiente de datos. No existe un control centralizado de los datos, un mismo dato podría aparecer en diversos archivos. Por ejemplo, el nombre de un empleado podría aparecer en los archivos del departamento de RH, en el de ventas, o en el de finanzas.
  - Dificultad para acceder a los datos. Cada consulta de datos o búsquedas requiere la creación de más programas encargados de realizar la búsqueda muy particular. Una búsqueda puede implicar procesar varios archivos que pudieran tener diferente formato y estructura lo que complica su lectura.
  - Dificultad para administrar los archivos tan pronto como su cantidad aumenta considerablemente, adicional a las operaciones de mantenimiento que requieren: inserciones, actualizaciones, eliminación de registros.
  - Falta de medidas de seguridad y control de accesos, en especial con datos sensibles y con datos que deben ser compartidos inclusive con usuarios geográficamente dispersos.
- Las bases de datos permiten en buena medida que los “programas” sean independientes a la definición de la estructura de los datos, esto debido principalmente a la capacidad de proporcionar una fuente común y centralizada de los datos, así como el soporte de lenguajes encargados de realizar el acceso y manipulación de los datos.

#### 2.4.2. Modelo de datos Jerárquico.

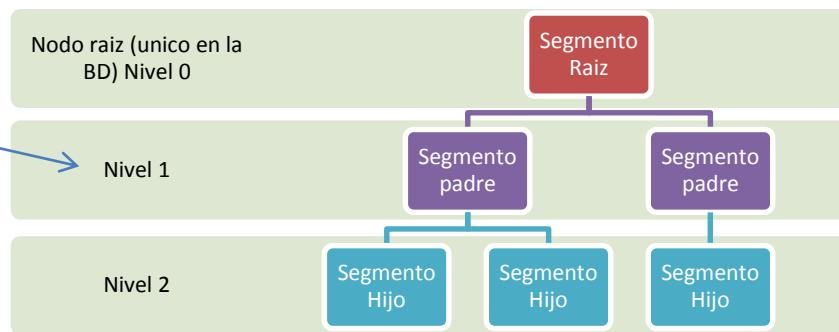
- Desarrollado en la década de los 60s para administrar grandes cantidades de datos para proyectos complejos como fue el proyecto del Cohete Apollo Rocket que aterrizó en la luna en 1969.
- Representa el primer modelo de datos empleado en un DBMS comercial formado por una estructura de árbol.
- No existen estándares ni formalizaciones del modelo jerárquico como en el caso del relacional.
- Sistema principal que implementa a este modelo es IMS (Information Management System) de IBM el cual emplea un lenguaje de datos DL/I
- El modelo hace uso principalmente de punteros y fue desarrollado especialmente para representar relaciones 1:M. Se emplea una estructura PADRE/HIJO donde el nodo PADRE puede tener a varios nodos HIJO, mientras que un nodo HIJO solo puede tener a un nodo PADRE.
- A las entidades se les conoce como SEGMENTOS, a sus instancias se les conoce como OCURRENCIAS, y a los atributos como CAMPOS.
- Solo permite representar relaciones 1:1, 1:M

Una vez creada la estructura jerárquica, está ya no se puede modificar. Si se desean aplicar cambios, se debe eliminar.



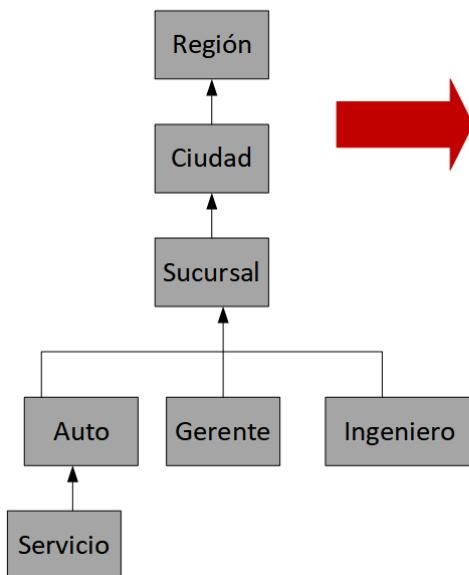
La liga entre nodos determina el camino único de acceso a los datos llamado **CAMINO SECUENCIA JERARQUICA**.

Se realiza un recorrido arriba, abajo, de izquierda a derecha, adelante y atrás.

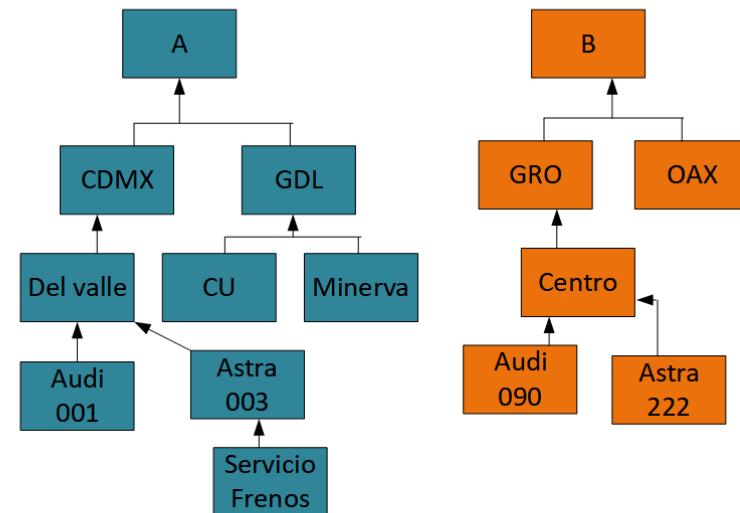


### Ejemplo:

Diseño conceptual  
(Definición de Entidades y Relaciones)



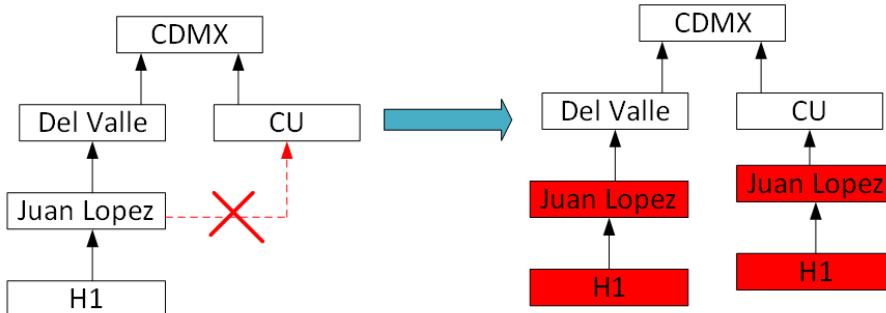
Instancias de entidades



- Un registro en una BD jerárquica representa la ocurrencia del nodo raíz, más la ocurrencia de todos los nodos que dependan jerárquicamente del nodo raíz, es decir, los registros en la BD representan una colección o bosque de árboles disjuntos.

#### 2.4.2.1. Problemas del modelo Jerárquico

- Redundancia** al no permitir que un nodo tenga más de un parente, por ejemplo, representación de relaciones M:N Una agencia tiene varios agentes de ventas , un agente puede trabajar en varias agencias:



- Cualquier nodo a excepción de nodo raíz, debe tener un parente. Por ejemplo, en la estructura anterior, no se puede registrar a un agente sin su correspondiente agencia en la que labora.
- Eliminar un nodo provoca eliminar a sus nodos hijos. ¿Qué pasa si se elimina a la agencia “Central”? Todos los datos asociados a la agencia como son empleados, autos, etc., también se eliminan.
- El modelo jerárquico no ofrece herramienta alguna para implementar restricciones a los datos.

#### 2.4.3. Modelo de red.

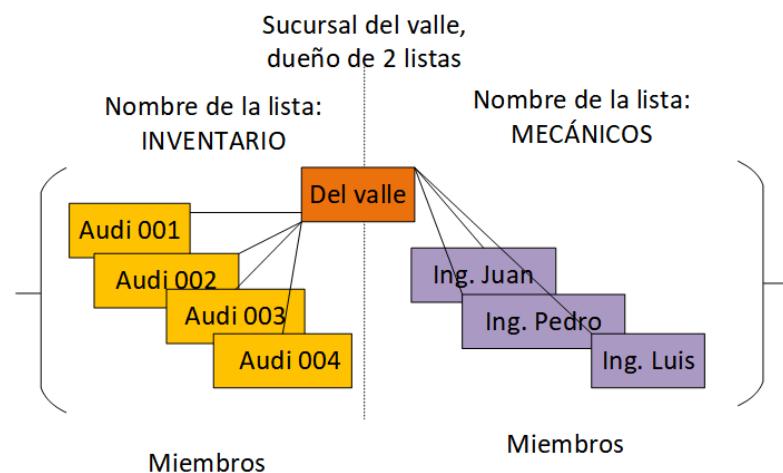
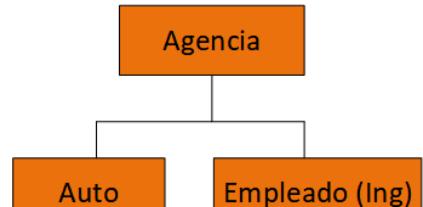
- Creado para representar relaciones complejas de una forma más efectiva con respecto al modelo jerárquico principalmente para mejorar desempeño.
- Similar al jerárquico, las entidades se representan como nodos y sus relaciones son las líneas que los unen.
- El usuario final percibe a los datos como una **colección de relaciones 1:M**
- Un registro puede tener **varios padres**.
- Ejemplos de este modelo: CODASYL. fue desarrollado en 1971 por un grupo conocido como **CODASYL** (Conference on Data System Languages, Data Base Task Group).

##### 2.4.3.1. Conceptos del modelo de red.

- Las relaciones entre instancias de una entidad deben ser descompuestas en un conjunto de **listas**.
- Cada lista está formada por:
  - Una única instancia de la entidad padre llamada **Dueño**.
  - Un conjunto de instancias hijas, todas ellas asociadas a la instancia padre llamadas **miembros**.
- Una misma instancia puede ser **dueño** de varias listas.
- Cada lista debe tener un nombre asignado.

Ejemplo:

- En una sucursal pueden existir varios autos, en una sucursal pueden laborar hasta 3 Ing. Mecánicos (ver imagen del lado derecho).
- La descomposición en 2 listas para una instancia de la entidad Sucursal (Sucursal del valle) y sus instancias asociadas se muestra en la siguiente imagen.
  - Lista 1: *Inventario*. Corresponde a la lista de autos que pertenecen a una Agencia en particular.
  - Lista 2: *Mecánicos*. Conjunto de Ingenieros que trabajan en una Agencia en particular.



En la figura anterior se tiene que una instancia de Agencia llamada **Dueño** (Owner) se asocie con un conjunto de Autos llamados **Miembros**. Lo mismo ocurre para los Ing. Mecánicos.

El modelo de red prácticamente no se emplea actualmente, aunque algunos de los conceptos que surgieron con él, se siguen empleando en modelos modernos:

- **Esquema:** Concepto empleado para organizar o agrupar a los objetos que integra a una base de datos.
- **Data Manipulation Language (DML):** Define el lenguaje empleado para realizar la manipulación de los datos: inserciones, actualizaciones, eliminaciones.
- **Data Language Definition (DDL):** Define el lenguaje empleado para definir los elementos y objetos que integran a un esquema.

#### **2.4.4. Modelo Entidad - Relación (E/R)**

- Propuesto por Peter P. Chen en 1976
- Empleado para construir una vista unificada de los datos empleando un enfoque “natural” fácil de entender para un usuario final e independiente al tipo de modelo de datos.
- En 1988 ANSI lo seleccionó como modelo estándar para los sistemas de diccionarios de recursos de información.
- Comúnmente empleado para realizar el diseño conceptual de una base de datos (se revisa a detalle más adelante).

#### **2.4.5. Bases de datos orientadas a objetos (OODBMS).**

- En estos manejadores ya no se requiere realizar un mapeo entre el modelo de objetos y el modelo relacional.
- Los objetos de una aplicación se guardan como tal en la base de datos.
- No existen tablas, ni SQL relacional.
- Existe un **solo modelo**, que en este caso es el modelo de objetos.
- Implementaciones:

Versant Object Database (antes DB4O)

<http://www.actian.com/products/operational-databases/versant/>

ObjectDatabase++ (ODBPP)

<http://www.ekkysoftware.com/ODBPP>

ObjectDB

<http://www.objectdb.com/>

Objectivity/DB

<http://www.objectivity.com/products/objectivitydb/>

Ejemplo:



## Guardando objetos:

```
Piloto piloto =
    new Piloto("Schumacher", 100);
db.store(piloto);
System.out.println("Stored " + piloto);
```

## Recuperando objetos:

```
Piloto pilotoEjemplo =
    new Piloto("Juan", 2);
ObjectSet result =
db.queryByExample(pilotoEjemplo);
listResult(result);
```

**2.4.6. Modelos Objeto – Relacional (ORDBMS).**

Los ORDBMS son RDBMS agregando funcionalidades asociadas con conceptos de la programación orientada a objetos (POO). Entre sus principales características se mencionan las siguientes:

- Tipos de datos personalizados

```
create type Persona (nombre varchar(20), dirección varchar(20))
```

- Herencia entre tablas (no todos los manejadores la implementan)

```
create type Estudiante under Persona (curso varchar(20), departamento
varchar(20))
```

- Funciones definidas por el usuario.

```
create function myFunction(p1, P2) BEGIN .... END;
```

- En un ORDBMS todos los elementos de la base de datos **son considerados como objetos** con sus 2 elementos similar a un objeto en POO

- Características -> atributos.
- Comportamiento -> métodos.

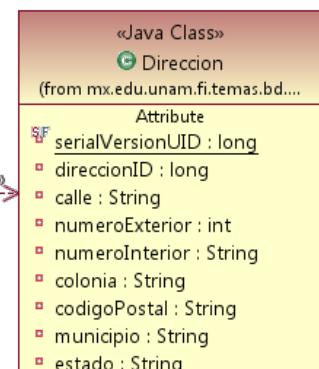
- Una tabla, registro, columna, vista, índice, usuario, todos son considerados como objetos.

**2.4.6.1. Mapeos Objeto – Relacional ORMs**

- Para poder sincronizar el estado de un conjunto de **objetos** en memoria (valores de sus atributos) con los valores de las columnas de un conjunto de **tablas** en el modelo relacional se requiere programar y ejecutar sentencias SQL que permitan comunicar a ambos modelos: POO y RDBMS.
- Un ORM permite realizar esta actividad sin la necesidad de realizar programación de sentencias SQL. El ORM las genera a partir de mapeos especificados ya sea en XML o por anotaciones.
- Ejemplos de implementación de ORMs: Hibernate, JPA.

ALUMNO	
ALUMNO_ID (PK)	NUMBER(10,0) NOT NULL
NOMBRE	VARCHAR2(50) NOT NULL
APELLIDO_PATERNO	VARCHAR2(50) NOT NULL
APPELLIDO_MATERNO	VARCHAR2(50) NOT NULL
DIRECCION_ID (FK)	NUMBER(10,0) NOT NULL

DIRECCION	
DIRECCION_ID (PK)	NUMBER(10,0) NOT NULL
CALLE	VARCHAR2(50) NOT NULL
NUMERO_EXTERIOR	NUMBER(5,0) NOT NULL
NUMERO_INTERIOR	VARCHAR2(5) NOT NULL
COLONIA	VARCHAR2(50) NOT NULL
CODIGO_POSTAL	VARCHAR2(5) NOT NULL
MUNICIPIO	VARCHAR2(50) NOT NULL
ESTADO	VARCHAR2(50) NOT NULL



## 2.4.6.2. Mapeo con XML y anotaciones.

```

1 <hibernate-mapping package="mx.edu.unam.fi.entidades">
2   <class name="Alumno" table="ALUMNO" dynamic-update="true" >
3     <id name="alumnoID" column="alumno_id">
4       <generator class="sequence" name="ALUMNO_SEQ"/>
5     </id>
6     <property name="nombre" type="string" column="nombre" />
7     <property name="apPaterno" column="ap_paterno" />
8     <property name="apMaterno" column="ap_materno" />
9     <many-to-one name="direccion" class="Direccion"
10       column="direccion_id" cascade="save-update"/>
11   </class>
12 </hibernate-mapping>

```

**Tarea:**  
Hibernate, JPA

```

1   @Entity
2   @Table(name="ALUMNO")
3   @SequenceGenerator(name="seq" sequenceName="ALUMNO_SEQ")
4   public class Alumno {
5
6     @Id
7     @Column(name="ALUMNO_ID")
8     @GeneratedValue(strategy=SEQUENCE, generator="seq")
9     private long alumnoID;
10
11     @Column(name="NOMBRE")
12     private String nombre;
13
14     @Column(name="APELLIDO_PATERNO")
15     private String apPaterno;
16
17     @Column(name="APELLIDO_MATERO")
18     private String apMaterno;
19
20     @ManyToOne(fetch=FetchType.LAZY)
21     @JoinColumn(name="direccion_id")
22     private Direccion direccion;
23
24   }

```

## 2.4.7. Soporte de los ORDBMs para procesar documentos XML

- Este soporte está representado por extensiones implementadas como parte de las características de un ORDBMS para almacenar documentos XML en columnas con tipo de dato XML.
- La gran mayoría de los manejadores relacionales actuales soportan tipos de datos XML.

Para realizar operaciones sobre los datos que contiene el documento XML se emplea el estándar **SQL/XML** y herramientas como XPath, XQuery.

Ejemplo:

- Creación de una tabla con una columna tipo XML que permite almacenar documentos XML.

```
create table empleado (
  empleado_id number(3),
  datos_empleado xmltype
);
```

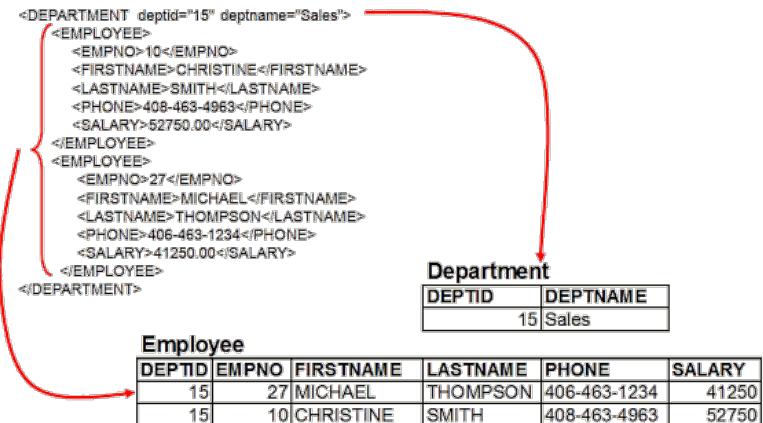
**Tarea:**  
Estándar SQL/XML

- Código empleado para insertar un registro en el que el valor de la columna datos\_empleado es un documento XML.

```
insert into empleado (empleado_id, datos_empleado)
values (
  1, xmltype(
<empleado>
  <nombre>daniel</nombre>
  <fecha_nacimiento>12/1/1951</fecha_nacimiento>
  <email>damorgan@u.washington.edu</email>
</empleado>)
);
```

- Código empleado para realizar consultas. Observar que es posible hacer referencia a los atributos y elementos del documento XML. El RDBMS analiza el documento, extrae los datos y los presenta en formato tabular.

```
select empleado_id, xmlquery(
  'for $i in /empleado
  where $i /nombre= "daniel"
  order by $i/nombre
  return $i/nombre'
  passing by value datos_empleado
  returning content) xmldata
from datos_empleado;
```



#### 2.4.8. Soporte de los ORDBMs para procesar documentos JSON

- De forma muy similar a documentos XML, la mayoría de los ORDBMs actuales implementan el estándar SQL/JSON que permite almacenar documentos JSON.

##### Ejemplo:

- Código empleado para almacenar un documento JSON en la columna productos

```
create table orden_detalle(
  id varchar2 (32) not null primary key,
  fecha_orden timestamp (6) with time zone,
  productos varchar2 (32767)
  constraint ensure_json check (productos is json)
);
```

- Código empleado para insertar un registro con un documento JSON

```
insert into detalle_orden(id,fecha_orden,productos) values (
  1,to_date('30-dec-2014'),
  '{ "numorden" : 1600,
    "referencia" : "abull-20140421",
    "solicitante" : "alexis bull",
    "usuario" : "abull",
    "centrocostos" : "a50",
    "instruccionesentrega" : {...},
    "instruccionesespeciales" : null,
    "habilitaentregaspaciales" : true,
    "productos" : [...]
  }'
);
```

- Código empleado para realizar una consulta SQL para obtener datos del documento JSON.

```
select do.productos.referencia from detalle_orden do;
```

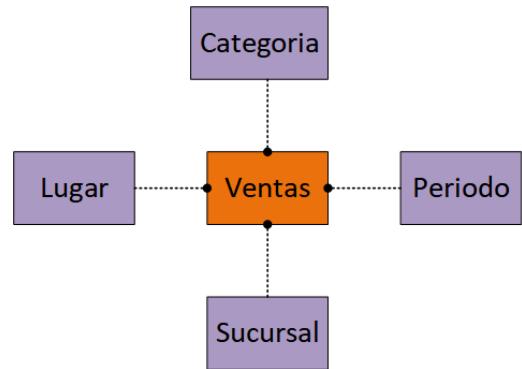
**Tarea:**



Estándar SQL/JSON

#### 2.4.9. Bases de datos multidimensionales

- Se emplean principalmente para aplicaciones OLAP como parte de las herramientas empleadas en el área de Business Intelligence.
- Procesan una gran cantidad y volumen de información con la finalidad de realizar análisis.
- Los resultados son vitales para realizar la toma de decisiones.
- Las dimensiones del BD multidimensional se implementan a través del concepto de Cubo OLAP.
- A partir de la construcción del cubo se generan tablas (generalmente cantidades mayores de columnas) con las que se realiza el análisis y consultas de forma rápida y eficiente.
- Los datos de la tabla resultante ya no se pueden modificar. Se debe regenerar o rediseñar el cubo.



#### Ejemplo:

Una empresa podría analizar algunos datos financieros por producto, por período, por ciudad, etc.

- Los parámetros en función de los cuales se realiza el análisis de datos se les conoce como dimensiones.
- Cada una de las dimensiones está formada por una Jerarquía de datos.

Dimensión: (Por Período, Por Producto)

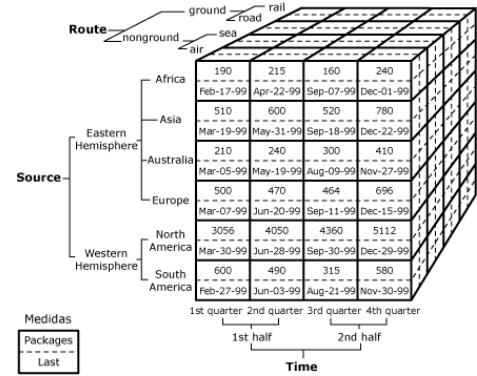
Jerarquía: (Año->Semestre->Mes->Semana),(Categoría->Línea->Marca)

Hechos: (Ventas, Inventario, Defectos, Devoluciones)

Métricas: (PD:=Devoluciones/Ventas, %Defectos)

Tabla resultante:

Tiempo	Productos	Ventas	Inventario	Defectos	Devoluciones	%Defectos
2006	Todos	1000	200	50	1/100	5
Enero06	Laptop	10	100	10	10/10	100%



#### 2.4.10. Big Data

La necesidad para administrar y ser capaces de atender las actuales demandas y tendencias en cuanto a la rapidez de crecimiento de los datos, desempeño, escalabilidad, y bajos costos ha disparado un fenómeno conocido como **Big Data**.

- Imaginar almacenar y analizar cada clic y actividad que realiza un cliente en un sitio web altamente concurrido para presentarle ofertas con base a sus gustos.
- Imaginar los requerimientos que implementa Facebook, YouTube, Twitter, o cualquier otra red social para almacenar el contenido producido en un día...
- Imaginar la capacidad de análisis de Amazon para ofrecer sugerencias a sus clientes en diferentes sitios web ...
- Imaginar los requisitos para que ideas como “Internet de las cosas” puedan ser una realidad...



La forma en la que los datos surgen y se administran hoy en día producen **requerimientos como los siguientes:**

- Incremento considerable del volumen de información a almacenar: **Terabytes por día o por horas.**
- **Facilidad para consultar grandes cantidades de datos** de una forma simple y eficiente.
- Necesidad de **distribuir el procesamiento a través de clusters** para mejorar el desempeño (**escalamiento horizontal**).
- Posibilidad de **almacenar, analizar y procesar datos** que no cuentan con una estructura definida (**datos no estructurados**), por ejemplo, **bitácoras**, datos que provienen de **dispositivos, imágenes, videos, etc.**
- Al hablar de este término, lo **primero** que se viene a la mente es pensar en **volumen de datos.**
- Volúmenes grandes de datos no es la única característica de BigData. Existen **4 principales características** que definen el término: **4V**
  - **Volumen**
  - **Velocidad**
  - **Variedad o diversidad**
  - **Valor.**

A nivel general, Big Data se refiere a un **movimiento que surge para encontrar nuevas y mejores formas para administrar grandes cantidades de datos y al mismo tiempo ofrecer un alto desempeño, escalabilidad a costos razonables.**

#### Volumen:

Grandes cantidades de datos son generados hoy en día, en especial por sistemas, sensores, etc. Un simple avión puede generar 10 TB en 30 min. Imaginar la cantidad total de datos por día de Una Aerolínea.

#### *Medidas usadas:*

- Terabyte  $10^{12}$  bytes
- Petabyte  $10^{15}$  bytes
- Exabyte  $10^{18}$  bytes
- Zettabyte  $10^{21}$  bytes ( 1 billón de Tera bytes)
- Yottabyte  $10^{24}$  bytes



#### **Algunos datos interesantes:**

Cantidad de datos en el mundo:

- 800 Terabytes, 2000
- 160 Exabytes, 2006
- 500 Exabytes (Internet), 2009
- 4.4 Zettabytes, 2012
- 44 Zettabytes by 2020



#### *¿Cuántos datos en un dia?*

- 7 TB, Twitter
- 200 TB, Facebook ~ 7 Petabytes por mes (7000 teras)

**Velocidad**

Imaginar la velocidad que se requiere para poder intercambiar o compartir grandes cantidades de datos entre sistemas, redes sociales, etc. Se requiere prácticamente comunicación en tiempo real. Imaginar el flujo de información de Twitter en un día.

**Variedad o diversidad.**

Existen varios tipos o fuentes de datos que considera el concepto de BigData.

- **Datos Estructurados:**

- Tienden a ser relativamente bien definidos a través de un «esquema» (modelo relacional).
- Datos tradicionales generados comúnmente por sistemas transaccionales OLAP

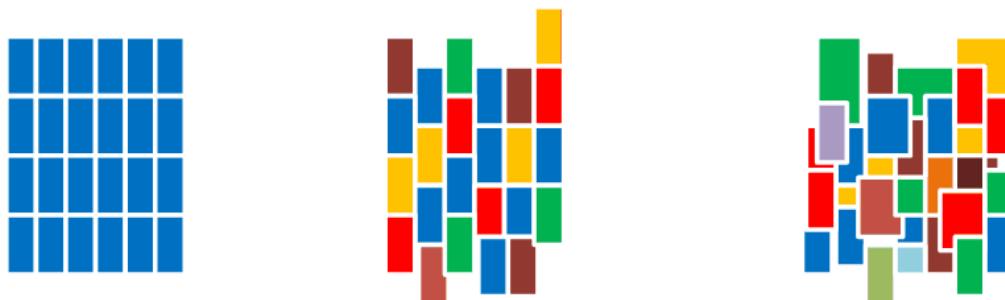
- **Datos Semi-estructurados**

- Datos generados por máquinas o sensores: bitácoras, Call Detail Records (Datos relativos a llamadas telefónicas).
- Datos sociales: Redes sociales, micro-blogging (Twitter).
- Típicamente: documentos JSON, XML

- **Datos No estructurados**

- Datos multimedia: imágenes, audio, bitácoras sin la existencia de un esquema.

*hadoop*


**Datos Estructurados**

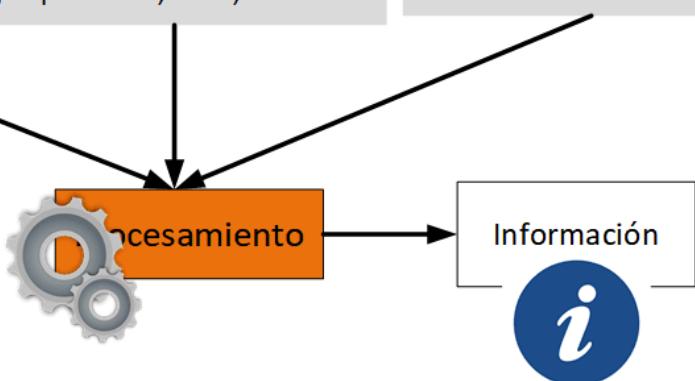
- Datos organizados con cierto formato
- Existe un modelo y esquema que define su estructura
- Ejemplos: RDBMs, excel

**Datos Semi - estructurados**

- Datos organizados de forma parcial
- Inexistencia de un modelo de datos que describe su estructura
- No siempre existe un esquema
- Ejemplos: XML, JSON, etc.

**Datos no estructurados**

- Datos no organizados
- No existe un modelo de datos
- No existe esquema
- Ejemplos: audio, vídeo, logs, imágenes.



El reto ahora es procesar estos 3 universos de datos.

**Valor.**

- Analizar y procesar ambos conjuntos de datos, revela un beneficio mayor para empresas.
- El conocimiento que se puede extraer de cada uno de estos universos es igual de importante que el conocimiento adquirido de a través del análisis típico con datos estructurados (sistemas OLAP).
- Empresas deben actualizar su infraestructura tecnológica para poder generar **valor** sobre estas nuevas formas de hacer análisis.

El análisis de datos basados en **herramientas y sistemas OLAP** son muy útiles y exitosas en ambientes con bases de datos relacionales. Sin embargo, realizar análisis sobre grandes cantidades de datos **semi y no estructurados** recolectados de diversas fuentes requieren estrategias diferentes.

Las siguientes tecnologías han permitido a las empresas procesar masivas cantidades de datos en múltiples formatos a costos aceptables.

Tecnología	Ejemplos.
Incorporación de <b>modelos y bases de datos NoSQL</b>	Basados en <b>documentos</b> , basadas en <b>clave llave valor</b> , etc.
Frameworks para procesar cantidades masivas de <b>datos distribuidos en clusters</b>	Apache <b>Hadoop</b> , <b>Spark</b>
Sistemas de <b>archivos distribuidos</b> que permitan <b>almacenar cantidades masivas de datos</b>	Hadoop Distributed File System ( <b>HDFS</b> )
Frameworks que permitan <b>comunicar sistemas distribuidos</b> a través de <b>colas de mensajes</b>	Apache <b>Kafka</b>
Nuevas <b>estrategias y técnicas</b> de programación que permitan el <b>procesamiento de cantidades masivas de datos</b> de forma distribuida.	<b>MapReduce</b>

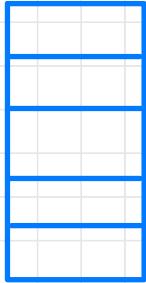
#### 2.4.11. **Bases de datos y modelos NoSQL.**

Surgen como respuesta para implementar nuevos requerimientos de una forma más eficiente y menos costosa de lo que un RDBMS puede normalmente realizar.

- Sistemas que no requieren un modelo con una estructura fija y rígida como lo es el relacional.
- Sistemas que no requieren un estricto control de concurrencia, por ejemplo, uso de las propiedades ACID de las transacciones.
- Sistemas que cuentan con una estructura de datos sencilla y a la medida de cada aplicación (El modelo relacional es un modelo genérico, no específico).
- La cantidad de datos a procesar la cual tiene a ser exponencial al transcurrir los días...
- Sistemas que requieren distribución de datos a través de un esquema de escalamiento horizontal, uso de clusters con múltiples servidores a bajo costo.
- La posibilidad de almacenar miles de registros con estructuras diferentes y variables (datos no estructurados).

##### 2.4.11.1. *Empresas con desarrollos NoSQL*

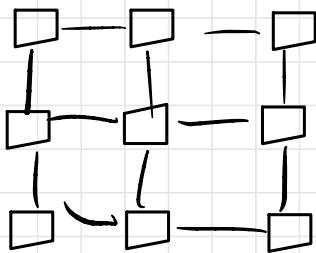
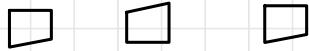
# ¿Qué es escalamiento Horizontal?



decrecer?

← agregarle más (antes)  
↑ caro :)

## Ahora



Divide y vencerás

capacidad de  
comparto grande  
uniendo

- Easy to move
- Located dif.
- Si se muere 1 quedan más

## Desventaja

• BD modelo relacional



• Datos divididos      BD distribuidas



- **BigTable** de Google
- **Dynamo** de Amazon
- **Cassandra** de Facebook. empleada para realizar búsquedas. Anteriormente, MySQL representaba el manejador favorito para manejar grandes cantidades de información en poco tiempo., pero, no dejaba de ser un modelo relacional.

Este tipo de soluciones, permite también implementar bases de datos distribuidas formadas por clusters baratos construidos de pc's comunes.

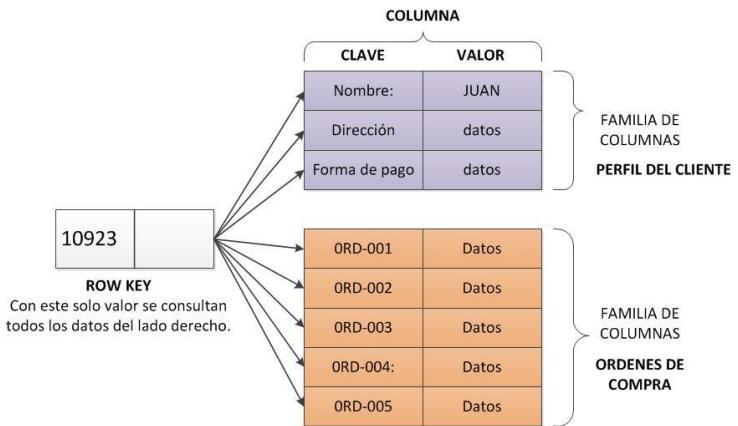
#### Ejemplos:

- **Cassandra** puede escribir hasta **50GB** de datos en disco en tan sólo **0.12** milisegundos, más de **2500** veces más rápido que MySQL.
- **BigTable** procesa 6 petabytes de datos por día empleando clusters formados por miles de servidores.
- Otro punto importante es el desempeño. Al no tener que realizar la traducción de datos hacia un formato amigable para SQL, las arquitecturas NoSQL son mucho más rápidas.



#### 2.4.11.2. Ejemplos de implementaciones NoSQL

- Orientadas al manejo de documentos:
  - CouchDB (de apache).
  - MongoDB (de empresa llamada 10gen).
  - SimpleDB (de amazon).
  - IBM Lotus Domino
  - Terrastore
- Orientadas al manejo de clave/valor
  - Cassandra (de apache).
  - BigTable (de google)
  - Dynamo (de amazon)
  - Project Voldemort (de LinkedIn)
- Orientadas al manejo de Grafos:
  - Neo4J
  - Allegro
  - Virtuoso



#### 2.4.11.3. Modelo de datos Cassandra:

**Columna:** Estructura de datos formada por un nombre y un valor.

**Familia de columnas:** Conjunto de columnas. Cada familia contiene una llave (key) la cual identifica al conjunto de columnas.

#### 2.4.11.4. Modelo de datos MongoDB

En este ejemplo se muestra la forma en la que se almacenan los datos de una entidad y los datos de las entidades con las que se relaciona empleando formato JSON

```

1  {"continente": {
2    "nombre": "América",
3    "habitado": "sí",
4    "paises": [
5      "país": [
6        {"nombre": "Costa Rica", "capital": "San José"},
7        {"nombre": "México", "capital": "DF"},
8        {"nombre": "Argentina", "capital": "Buenos Aires"}
9      ]
10    }
11  }}

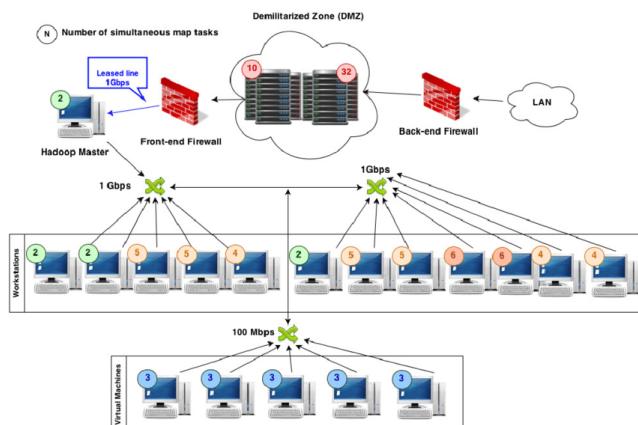
```

#### 2.4.11.5. Hadoop, HDFS.

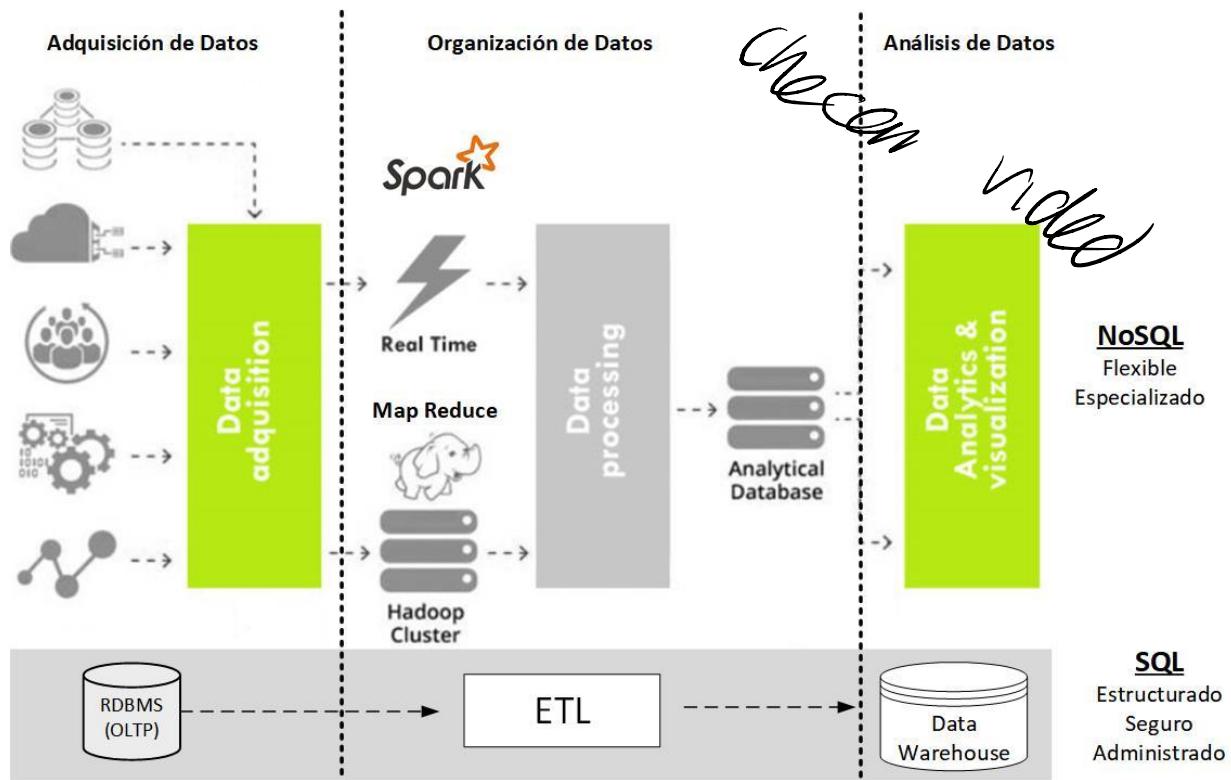
- Framework escrito en Java empleado para realizar procesamiento masivo de datos de forma distribuida a través del uso de clusters formados por máquinas que en su conjunto almacenan y procesan grandes volúmenes de datos.
- Formado por diversos módulos, 2 de ellos fundamentales: MapReduce y su sistema de archivos distribuido HDFS.
- Adicional a ser Open Source, permite implementar soluciones a bajo costo empleando clusters formados por máquinas simples.
- HDFS emplea el concepto de *Write one read many*. Los archivos se escriben en un nodo, y el framework se encarga de replicarlo para que pueda estar disponible en todos los nodos del cluster.
- HDFS hace uso de 3 tipos de nodos. *Name node*: encargado de guardar metadatos acerca del sistema de archivos y su contenido a lo largo del cluster. *Data node*: contiene una cierta cantidad de datos los cuales pueden ser replicados a otros nodos. *Client node*: actúa como interface o medio de acceso entre el cliente y HDFS.

#### 2.4.11.6. MapReduce

- Interfaz de programación (Application programming Interface - API) que permite la creación o programación de tareas las cuales pueden ser distribuidas y procesadas en paralelo en un cluster de Hadoop.
- El framework está formado por 2 principales funciones: La función `map` y la función `reduce`.
- La función `map` toma a la tarea y la divide o distribuye en unidades de trabajo pequeñas para ser procesadas en cada nodo. La función `reduce` recolecta las respuestas que se obtienen como resultado de la ejecución de la función `map`, las integra y genera un único resultado.



### Ejemplo de una arquitectura de solución.



- A nivel básico y en especial para realizar el análisis de datos estructurados, se requieren 3 principales etapas:
  - Adquisición u obtención de datos de diversas fuentes.
  - Organización y transformación de los datos para poder realizar el análisis.
  - Almacenamiento de los datos para su análisis en sistemas especializados.
- En un esquema tradicional estas 3 etapas típicamente se tienen los siguientes componentes (parte inferior de la imagen):
  - Los datos se encuentran en un RBDMS, en un sistema OLTP, generalmente con niveles altos de normalización.
  - Se requiere de un proceso ETL (Extracción - Transformación - Carga) principalmente para realizar la extracción de los datos del RDMBS, aplicar operaciones de trasformación, limpieza o denormalización que permitirá la carga de los datos en otro sistema para facilitar y optimizar su análisis.
  - Finalmente, los datos son almacenados en sistemas específicos o adecuados para análisis. Típicamente un Data Warehouse, listos para ser analizados por diversas herramientas entre otras, minería de datos.
- ¿Qué sucede con los datos semi – estructurados o estructurados?
  - Para realizar análisis con este tipo de información se emplean conceptos de BigData. Ahora los datos pueden localizarse en bases de datos NoSQL principalmente por sus características de ser semi o no estructurados, altos volúmenes, etc.
  - El proceso ETL que típicamente se realiza con datos estructurados se sustituye ahora por soluciones Map-Reduce, empleando frameworks y herramientas de procesamiento masivo y distribuido de datos como son: Hadoop, Spark, etc.

En el siguiente enlace se muestra como Oracle e IBM están implementando el concepto de Big Data así como los productos que ofrecen

<http://www.oracle.com/technetwork/es/articles/database-performance/big-data-oracle-hadoop-2813760-esa.html>

<https://developer.ibm.com/es/technologies/data-science/articles/que-es-big-data/>



**Tarea:**  
Resumen/opinión

#### 2.4.12. Ciencia de datos.

Una forma alternativa para realizar análisis de datos a las opciones vistas anteriormente es la llamada "Ciencia de los datos".

- La ciencia de los datos es un campo interdisciplinario que abarca el procesamiento y los sistemas para extraer conocimiento de datos que se pueden encontrar en varios formatos: Datos estructurados y datos no estructurados.
- En especial, los datos no estructurados representan la continuación de áreas de análisis de datos (Business Intelligence) como estadística, minería de datos, Análisis predictivo, Descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases - KDD).
- Principales áreas de conocimiento participan en Ciencia de datos:
  - Ingeniería de Software
  - Expertos en el dominio del problema (dominio de las reglas de negocio y operación asociadas con los datos).
  - Investigación de operaciones
  - Matemáticas
  - Estadística.
- Las 2 principales categorías de un científico de datos:
  - Analistas / Expertos en estadísticas
  - Ingenieros.

Los siguientes artículos muestran un panorama básico sobre esta área.

<https://dzone.com/articles/an-introduction-to-data-science>

[https://es.wikipedia.org/wiki/Ciencia\\_de\\_datos](https://es.wikipedia.org/wiki/Ciencia_de_datos)



**Tarea:**  
Resumen/opinión