TEMA 2.
MODELOS DE DATOS

Material para revisar en clase.

1. EJEMPLO – ELEMENTOS DE UN MODELO DE DATOS.

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:

A. Entidades
B. Reglas de negocio
C. 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 10,000 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.

2. EJEMPLOS DE MODELOS DE DATOS.

Sistemas de archivos.

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.

Almacenamiento empleando Sistemas Computacionales.

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. Olowski	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

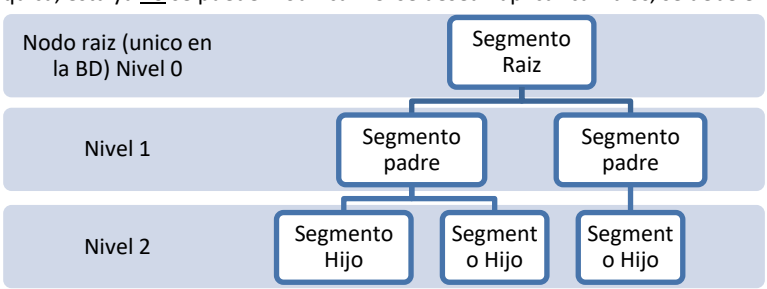
- La generación de reportes empleando sistemas de archivos manuales como los explicados anteriormente tomaba semanas en desarrollarlos.
- Surge entonces un nuevo rol llamado "*Especialista de procesamiento de datos*" (DP) encargado de crear un sistema basado en computadora encargado de almacenar los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

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 representa 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 1:

(Revisar en pizarrón).

- 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*.

Problemas del modelo Jerárquico

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

Ejemplo 2:

(Revisar en pizarrón).

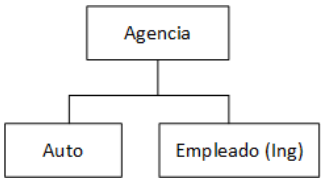
- Cualquier nodo a excepción de nodo raíz, debe tener un padre. *Por ejemplo*, no se puede registrar a un agente sin su correspondiente agencia en la que labora.
- Eliminar un nodo provoca eliminar a sus nodos hijos. ¿Qué pasa si se elimina a la agencia "Central"? Todos los datos asociados a la agencia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

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).

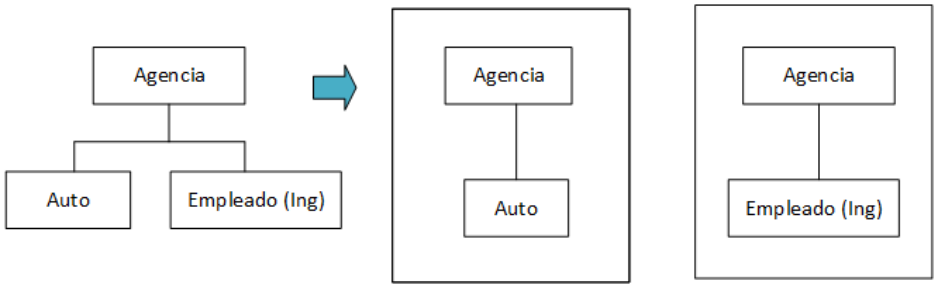
Conceptos del modelo de red.

- Las relaciones entre registros 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**.
- Un mismo registro puede ser **dueño** de varias listas.
- Cada lista debe tener un nombre asignado.

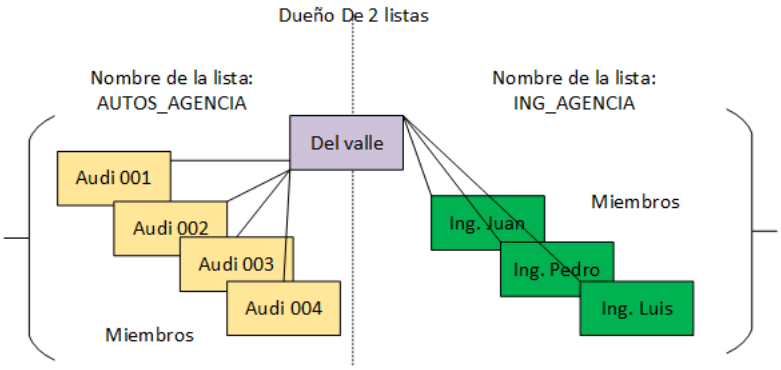


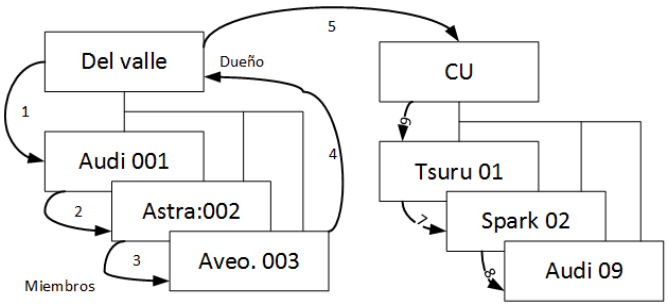
Ejemplo:

- En una sucursal pueden existir varios autos, en una sucursal pueden laborar hasta 3 Ing. Mecánicos (empleados). El modelo jerárquico para estas 2 reglas es el siguiente:
- La descomposición en 2 nuevas listas se muestra a continuación.
  - 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.





- En la figura del lado izquierdo se muestra un ejemplo de las instancias de las entidades Auto y Agencia con sus respectivas relaciones.
- El usuario del BD podrá navegar entre estos 2 conjuntos a través del lenguaje DML (Data Manipulation Language) proporcionado por la implementación de este tipo de DBMS. Empleando para ello punteros que indican el registro actual que se está accediendo (figura del lado derecho).

Modelo Relacional (RDBMS).

- Desarrollado en 1970 por Edgar Frank Codd.
- A diferencia de los 2 modelos anteriores, este no hace uso de punteros para asociar a 2 registros.
- Los registros se relacionan cuando se necesitan, el modelo no impone jerarquías ni estructuras predefinidas para realizar el acceso a los datos.

Este modelo se revisará a detalle en el siguiente capítulo.

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).

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/

Tarea:
Versant Object Database

Ejemplo:

Guardando objetos en Versant Object Database (antes DB4O):

```
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);
```

Modelos Objeto – Relacional (ORDBMS).

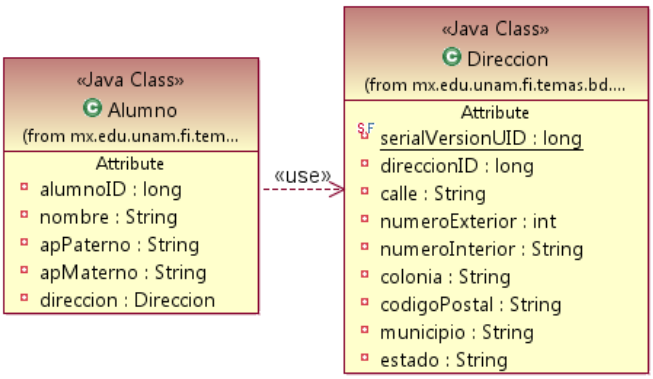
- Representan una extensión de un RDBMS agregando funcionalidades asociadas con conceptos de la programación orientada a objetos (POO).
- Tipos de datos personalizados: `create type Persona (nombre varchar(20), dirección varchar(20))`
- Herencia entre tablas (no todos los manejadores la implementan)
  `create type Estudiante under Persona (curso varchar(20),departamento varchar(20))`
- Funciones definidas por el usuario.

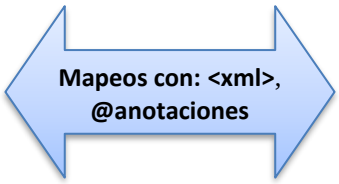
`create function myFunction(p1,P2) BEGIN …. END;`

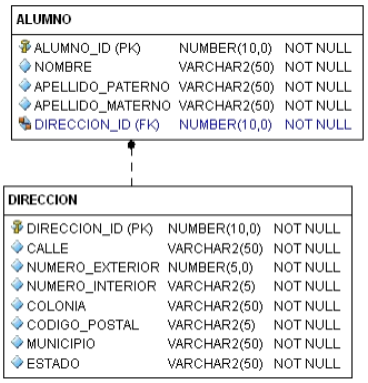
- En un ORDBMS todos los elementos de la base de datos son considerados como objetos con sus 2 elementos similar a un objeto en POO
  - Características -> atributos.
  - Comportamiento -> métodos.
- Una tabla, registro, columna, vista, índice, usuario, todos son considerados como objetos.

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.







Mapeo con XML y anotaciones

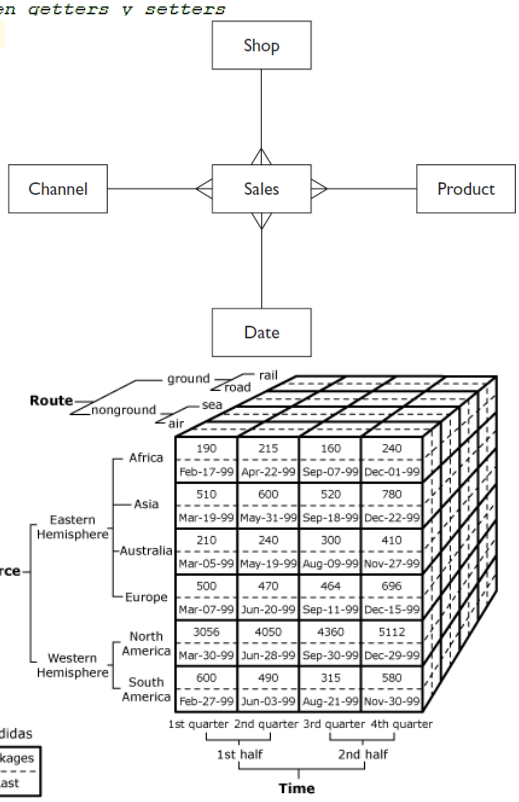
```xml
<hibernate-mapping package="mx.edu.unam.fi.entidades">
    <class name="Alumno" table="ALUMNO" dynamic-update="true" >
        <id name="alumnoID" column="alumno_id">
            <generator class="sequence" name="ALUMNO_SEQ"/>
        </id>
        <property name="nombre" type="string" column="nombre" />
        <property name="apPaterno" column="ap_paterno" />
        <property name="apMaterno" column="ap_materno" />
        <many-to-one name="direccion" class="Direccion"
            column="direccion_id" cascade="save-update"/>
    </class>
</hibernate-mapping>
```

```java
@Entity
@Table(name="ALUMNO")
@SequenceGenerator(name="seq" sequenceName="ALUMNO_SEQ")
public class Alumno {

    @Id
    @Colunm(name="ALUMNO_ID")
    @GeneratedValue(strategy=SEQUENCE,generator="seq")
    private long alumnoID;

    @Column(name="NOMBRE")
    private String nombre;

    @Column(name="APELLIDO_PATERNO")
    private String apPaterno;

    @Column(name="APELLIDO_MATERNO")
    private String apMaterno;

     @ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="direccion_id")
    private Direccion direccion;
    // se omiten getters y setters
}
```

Tarea:
Hibernate, JPA

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 Periodo, 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%

Bases de datos XML

- Son RDBMs con extensiones para almacenar documentos XML en columnas con tipo de dato XML.
- La gran mayoría de los manejadores 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:

```
CREATE TABLE empleado (
empleado_id   NUMBER(3),
datos_empleado XMLTYPE
);

INSERT INTO empleado (empleado_id, datos_empleado)
VALUES (1, XMLTYPE('
 <empleado>
   <nombre>Daniel</nombre>
   <fecha_nacimiento>12/1/51</fecha_nacimiento>
   <email>damorgan@u.washington.edu</email>
 </empleado>')
);

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;
```

```
<DEPARTMENT deptid="15" deptname="Sales">
  <EMPLOYEE>
    <EMPNO>10</EMPNO>
    <FIRSTNAME>CHRISTINE</FIRSTNAME>
    <LASTNAME>SMITH</LASTNAME>
    <PHONE>408-463-4963</PHONE>
    <SALARY>52750.00</SALARY>
  </EMPLOYEE>
  <EMPLOYEE>
    <EMPNO>27</EMPNO>
    <FIRSTNAME>MICHAEL</FIRSTNAME>
    <LASTNAME>THOMPSON</LASTNAME>
    <PHONE>406-463-1234</PHONE>
    <SALARY>41250.00</SALARY>
  </EMPLOYEE>
</DEPARTMENT>
```

Department

DEPTID	DEPTNAME
15	Sales

Employee

DEPTID	EMPNO	FIRSTNAME	LASTNAME	PHONE	SALARY
15	27	MICHAEL	THOMPSON	406-463-1234	41250
15	10	CHRISTINE	SMITH	408-463-4963	52750

Tarea:
XQuery, XPath

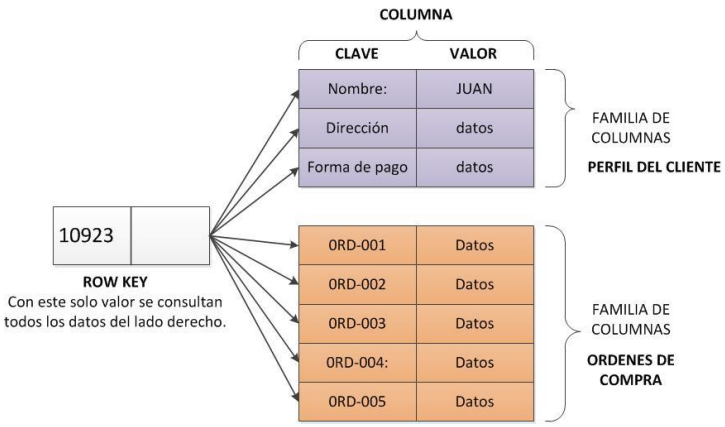
Modelos NoSQL.

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

- 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.
- Imaginar, analizar cada clic y actividad que realiza un cliente en un sitio web altamente concurrido para presentarle ofertas con base a sus gustos.
- Necesidad de distribuir el procesamiento a través de clusters para mejorar el desempeño (escalamiento horizontal).

Ejemplo:

Modelo de datos en Cassandra



Modelo de datos en MongoDB: Formato JSON

JSON

Ejemplos de implementaciones NoSQL

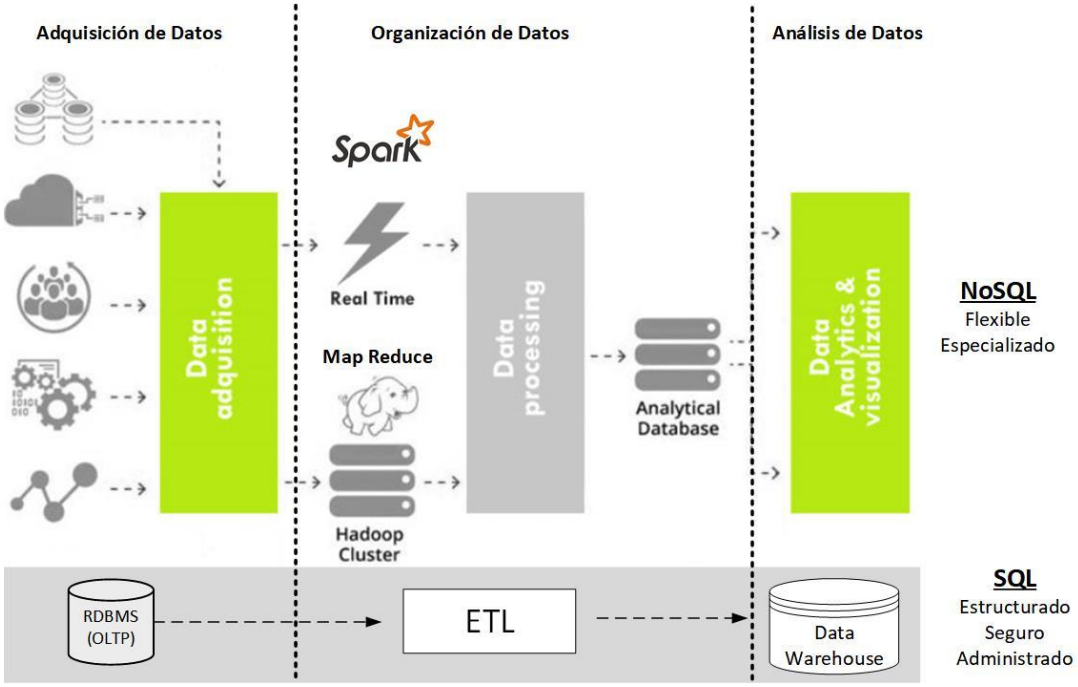
- Orientadas al manejo de documentos:
  - CouchDB (de apache).
  - MongoDB (de empresa llamada 10gen).
  - SimpleDB (de amazon).
  - IBM Lotus Domino
  - Terrastore
- Orientadas al manejo de clave/valor
  - Cassandra (de apache).
  - BigTable (de google)
  - Dynamo (de amazon)
  - Project Voldemort (de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

```json
{"continente": {
  "nombre": "América",
  "habitado": "si",
  "paises": {
    "pais": [
      {"nombre": "Costa Rica", "capital": "San José"},
      {"nombre": "México", "capital": "DF"},
      {"nombre": "Argentina", "capital": "Buenos Aires"}
    ]
  }
}}
```

BigData.

- Al hablar de este término, lo primero que se viene a la mente es pensar en «volumen», en gran cantidad de datos.
- Volúmenes grandes de datos no es la única característica de BigData. Existen 4 principales características que definen el término:
  - **V**olumen, **V**elocidad, **V**ariedad o diversidad, **V**alor.
- **Datos Estructurados:**
  - Tienden a ser relativamente bien definidos a través de un «*esquema*» (modelo relacional).
  - Datos tradicionales generados comúnmente por sistemas transaccionales
- **Datos Semi-estructurados**
  - Nuevas fuentes de información: documentos de texto no estructurados, videos, emails, audio, bitácoras, etc.
  - Datos generados por máquinas o sensores: bitácoras, Call Detail Records (Datos relativos a llamadas telefónicas).
  - Datos sociales: Redes sociales, micro-blogging (Twitter).

- **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 OLAP, 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 está implementando el concepto de Big Data así como los productos que ofrece: http://www.oracle.com/technetwork/es/articles/database-performance/big-data-oracle-hadoop-2813760-esa.html

Ciencia de los datos.

Tarea:
Resumen/opinión

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

Tarea:
Resumen/opinión