Chapter 1: Introduction

Modificado: Sonia Ordoñez S.

Database System Concepts, 6th Ed.

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to management the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

Drawbacks of Using File Systems to Store Data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Files supported by OS

	Branch name	account		account		account	
0	Perryridge	A-102	400	A-201	900	A-218	700
1	Round Hill	A-305	350	⊥	⊥	⊥	⊥
2	Mianus	A-215	700	⊥	⊥	⊥	⊥
3	Downtown	A-101	500	A-110	600	⊥	⊥
4	Redwood	A-222	700	⊥	⊥	⊥	⊥
5	Brighton	A-217	750	⊥	⊥	⊥	⊥

custimer	Account number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Drawbacks of Using File Systems to Store Data (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Levels of Abstraction

View level

- This level corresponds to solving the question What views and reports are required for the application (s) that will be supported by this particular database?
- For example, in an airline application, the list of all the itineraries (round trip) that coincide with origin, arrival, departure date and return defined by the user must be displayed through one of its interfaces.

Logical level

According to the requirements defined at the view level, the data is molded in such a way that it complies with the chosen paradigm and standard. In the case of the airline:

- Information on origins and destinations, that is, countries, cities and airports
- The information of the different flight plans of the airline, that is, the planning of the flights (airline, origin, destination, stopovers, day and time)
- Seat availability information ().
- Information on ticket costs

Levels of Abstraction

- Passenger information (passport, name, surname, nationality, visa, contact, telephone, mail, residence address)
- Information for the payment of tickets, that is, form of payment, franchise, among others.

- The relationships between all the information.
- The data types that the data supports
- A model is used that expresses the data and its relationships
- The key in data modeling is balancing application needs, database engine performance characteristics, and data recovery patterns.

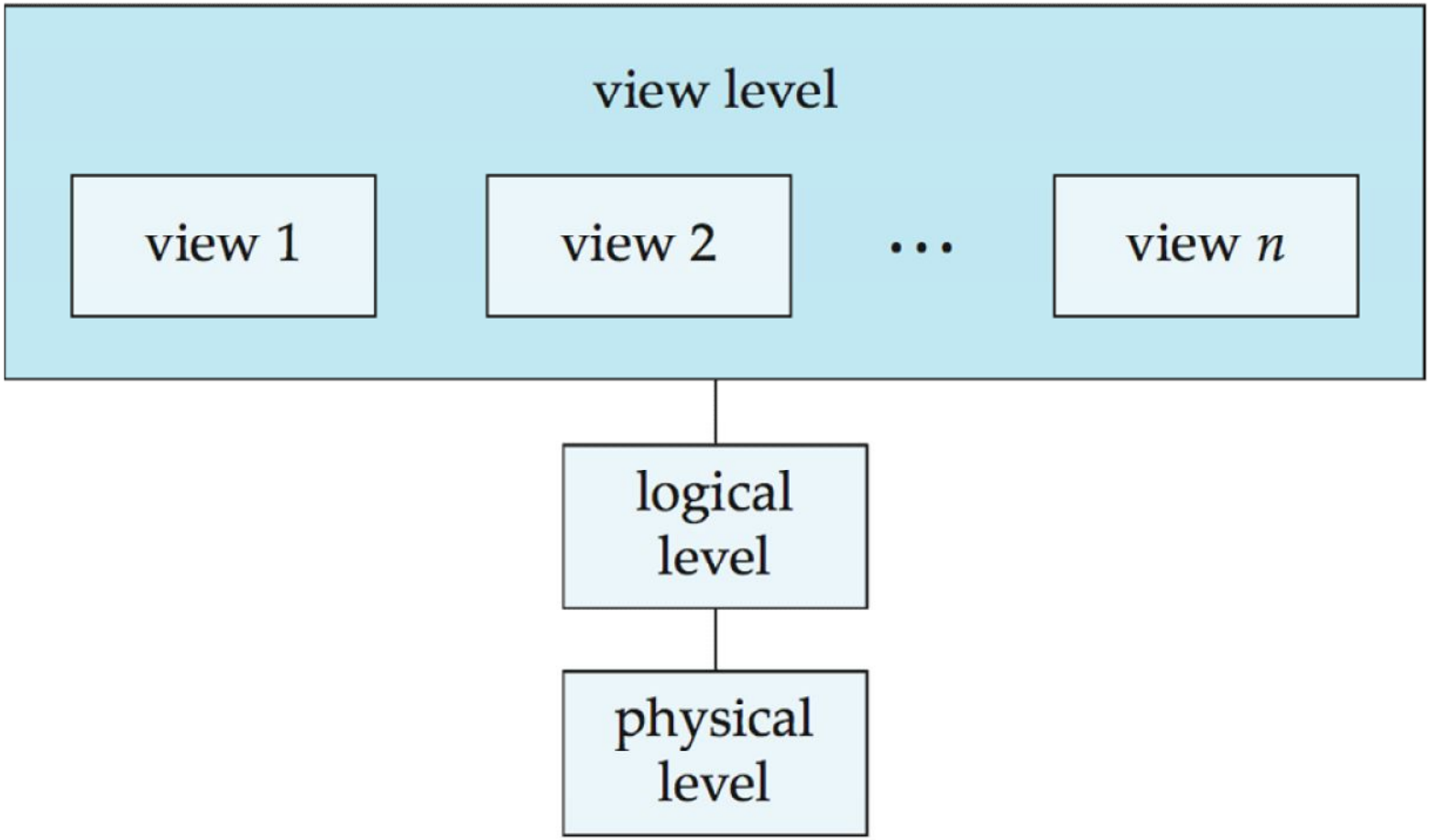
Levels of Abstraction

Physical level

- Each SMBD follows a paradigm and is therefore restricted to the type of data model and standards.
- From the logical level and the previous requirements, new elements are included so that:
  - structures
  - Information security
  - Information storage
  - Searches are fast
  - Relationships are guaranteed, among others.

- This level includes the computational implications of the logical layer
- This level is implemented through the own commands of the selected SMBD

View of Data



Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
  - Standard
  - paradigm

Database Classification by Data Models

- **Entity-Relationship data model (mainly for database design)**
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

Entity-Relationship Data Model

- The Entity - Relationship (ER) model is presented as part of the methodology published through an article “The Entity Relationship Model - Toward A Unified View of Data” (1976) by Peter P. Chen.
- This article:
  - ES considers a pioneer in modeling computational requirements.
  - It has been awarded as one of the 38 most influential articles for computer science.
  - It is one of the most cited works in the field of computer science
  - It has been and continues to be the basis not only for methodologies for the development of systems but also for work aimed at developing CASE (Computer-Aided Software Engineering) tools.

Entity-Relationship Data Model

- The E-R model was used by IBM and was later adopted by the American National Standards Institute (ANSI) as a standard meta-model.
- It is the basis of methodologies such as Oracle, Microsoft, Unified Modeling Language (UML), among others.
- The original entity-relationship (ER) model proposed by Peter P. Chen has undergone changes and an extended model is currently used.
- Currently it is used as a previous step to the relational model

Entity-Relationship Data Model

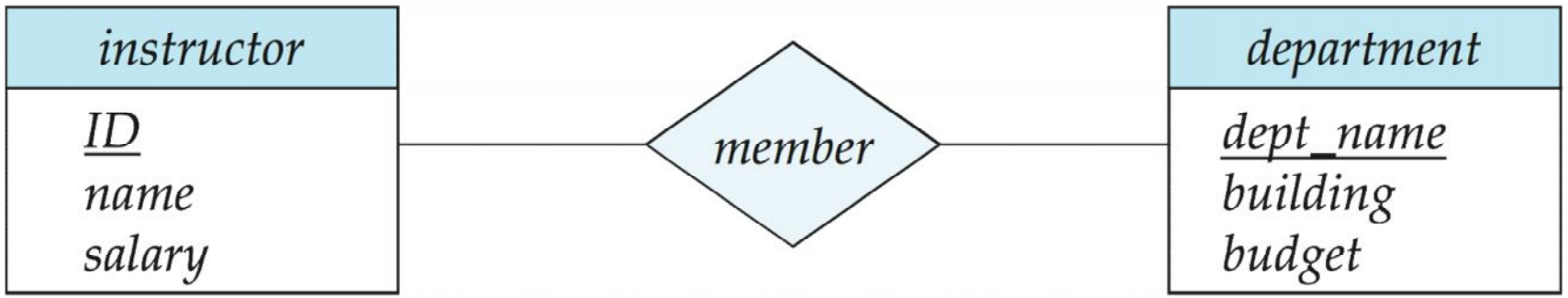
ADVANTAGE

- It allows to find new entities from the existing ones.
- Transform multi-dimensional relationships to binary.
- Transform attributes to relationships and vice versa (Normalization)
- Convert the model to an "implementable relational model" in a database engine.
- It is very widespread at the academy level.
- It is supported by many of the tools for database design.

DISADVANTAGES

- Exclusive to relational data
- inadequate for unstructured data
- It does not allow an integration with an existing database engine.

E-R Data Model



Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- **Relational model**
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

Relational Data Model

- Much of what the relational model is today was published in 1970 in the article “A relational model of data for large shared data banks” by the English mathematician Edgar Frank Codd of the IBM laboratories in San José California.
- With this publication Codd created a new paradigm and directs the data model towards the needs of users, that is, towards queries.
- Codd's ideas had no impact and later Michael Stonebreaker of the University of Berkeley in California, search funding to develop the Ingres system (based on Codd's publication) whose first version was presented in 1974 as the first relational database manager data.
- Codd also developed the first relational language called ALPHA.

Relational Data Model

CHARACTERISTICS OF THE MODEL

- The model is based on set theory and predicate logic.
- Define relationships as a set of tuples, where each tuple is an unordered collection of different elements.
- Includes the foreign key to combine domains or relationships.
- Uses the concept of primary key, to distinguish one tuple from another, that is, a set of elements of the tuple that allows to univocally distinguish one tuple from the other.
- Codd originally proposed eight operations: five fundamental (restriction, projection, Cartesian product, union, difference), and three that can be expressed from the above: (concatenation (join), intersection and division). The constraint and the projection are unary operations (they operate on a single relation), while the others are binary because they work on pairs of relations

Relational Data Model

CHARACTERISTICS OF THE MODEL

- Codd's original article defines an array, which represents an R n-ary relationship with the following properties:
  - Each row represents a tuple of R.
  - The order of the rows is immaterial.
  - All the rows are different
  - The order of the columns is significant and corresponds to the order S1, S2,. . . , Sz of the domains in which R is defined
  - The semantics of each column is partially carried by the label with the name of the corresponding domain.

Relational Data Model

- IBM reacts and builds the relational engine R, with multi-user features and a structured query language, SEQUEL, which would later be called SQL (Structured Query Language).
- Around the same time and based on Codd's research, a businessman from Valle del Silicón developed the Oracle engine.
- From the relational model proposed by Codd, variations such as that of the British Richard Barker, Ian Palmer, Harry Ellis and others appear. This notation was developed for a consultancy.
- When Barker worked with Oracle he adopted it and published the book "Entity Relationship Modeling" as part of the CASE Method book series.
- This notation was and continues to be used by the Oracle CASE modeling tools.

Relational Data Model

ADVANTAGE

- It provides tools that guarantee to avoid duplication of records.
- It guarantees referential integrity, thus, deleting a record implies deleting all related records.
- It favors standardization by being more understandable and applicable.

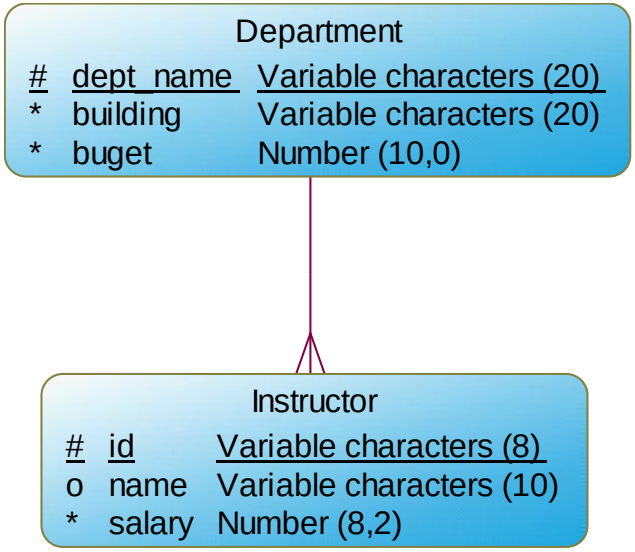
DISADVANTAGES

- Exclusive to relational data
- Not suitable for unstructured data

STANDARDS

“ANSI-American National Standards Institute” specifies the standards that define the elements and characteristics of the relational model

Relational Data Model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

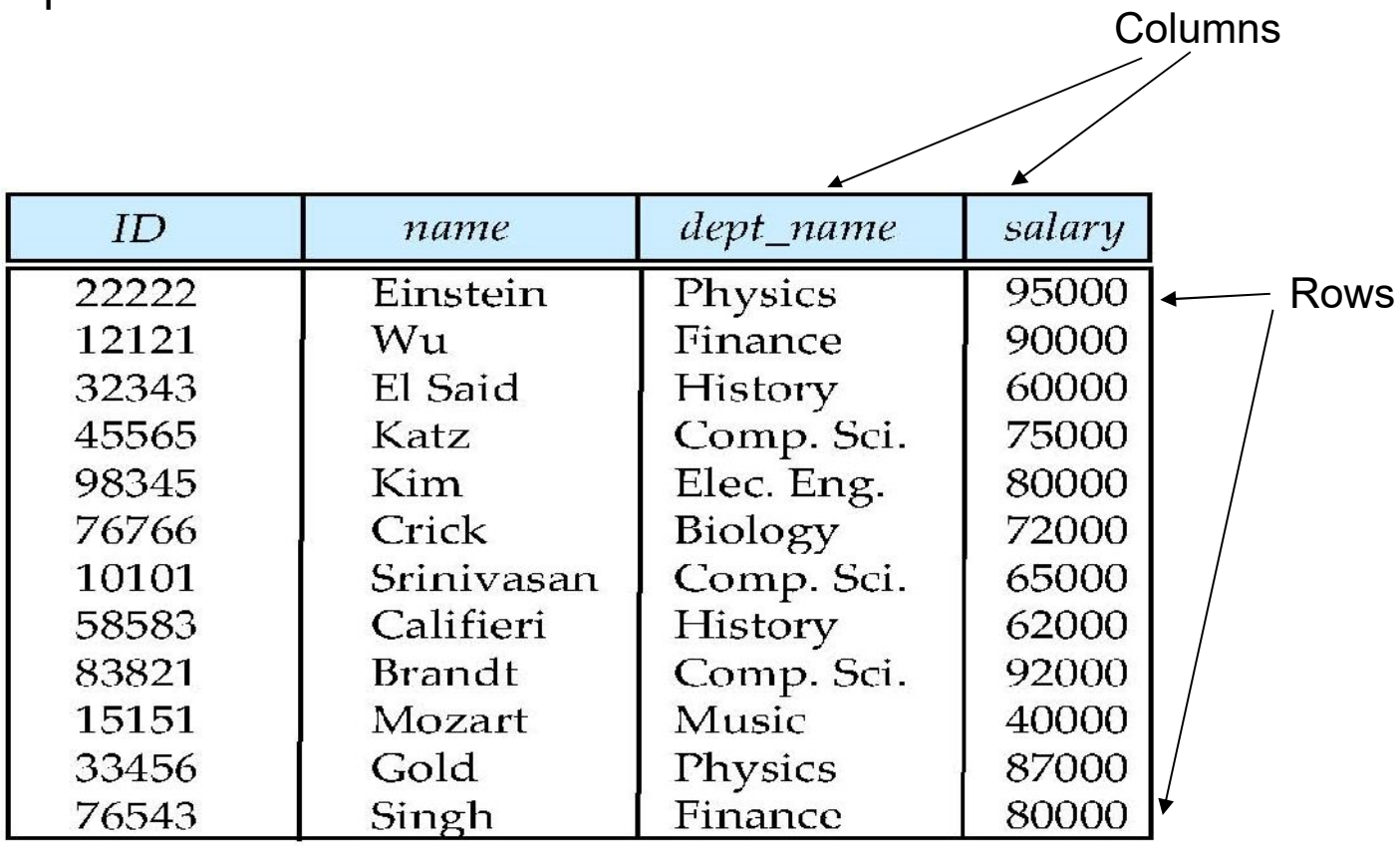
(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Relational Model

- Example of tabular data in the relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- **Object-oriented model**
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

Object-Oriented Data Model

- Object-oriented database management systems (ODBMS) appeared in the 1980s with the growth of object-oriented languages such as C ++.
- ODBMS extends object-oriented languages with persistent data transparently, concurrency control, data retrieval, associative queries, and other capabilities.
- From the moment object-oriented languages start, many experimental prototypes and commercial object-oriented database systems appear, among these O2, GEMSTONE, ORION, OpenOODB, IRIS, ONTOS, ObjecStore.
- On the other hand, the database community observed that the relational model was unsuitable for domains addressed to knowledge bases, geo-referential applications and in general non-conventional databases, which involve complex data objects and their interactions such as software. CAD / CAM, computer graphics and information retrieval.

Object-Oriented Data Model

- In an object-oriented database, data is represented as a collection of objects that are organized into classes and have complex values and methods.
- Although object-oriented database models allow much richer structures than the relational base model, they require that all data conform to a predefined schema.
- Object-oriented data models are similar to semantic models in that they provide mechanisms for building complex data. They differ from semantic models in that they support local forms of behavior in a way similar to object-oriented programming languages.
- In object-oriented bases it is modeled the same as in object-oriented programming
- The Object Database Management Group (ODMG) specifies the standards and elements that define the characteristics, such as achieving persistence in object-oriented databases.

Object Database Management Group (ODMG)

ODMG specifies the standards that define Object Definition Language (ODL) and Object Query Language (OQL).

ODL

- ODL is an object definition language, which allows you to define object types for ODMG-compliant systems (ODL is the equivalent of the DDL data definition language for DBMS).
- Defines attributes and relationships between types, and specifies operations.
- The ODL syntax extends the Common Object Request Broker Architecture (CORBA) interface definition language (IDL).

Object Database Management Group (ODMG)

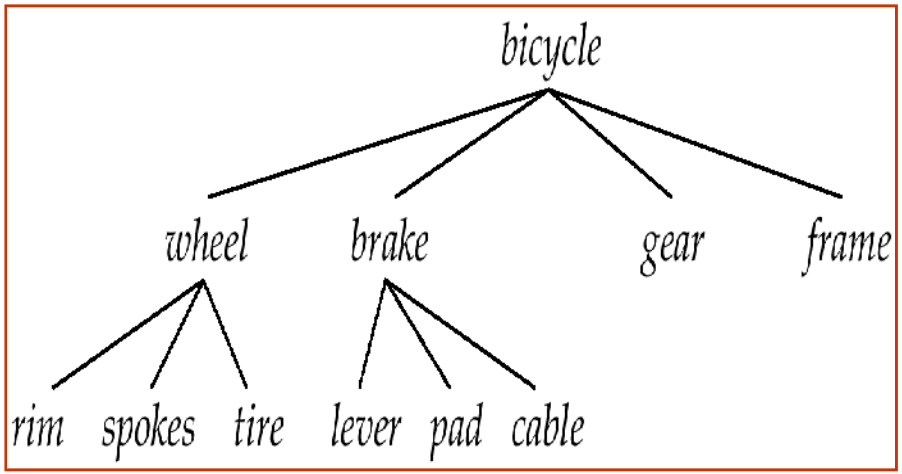
ODMG specifies the standards that define Object Definition Language (ODL) and Object Query Language (OQL).

OQL

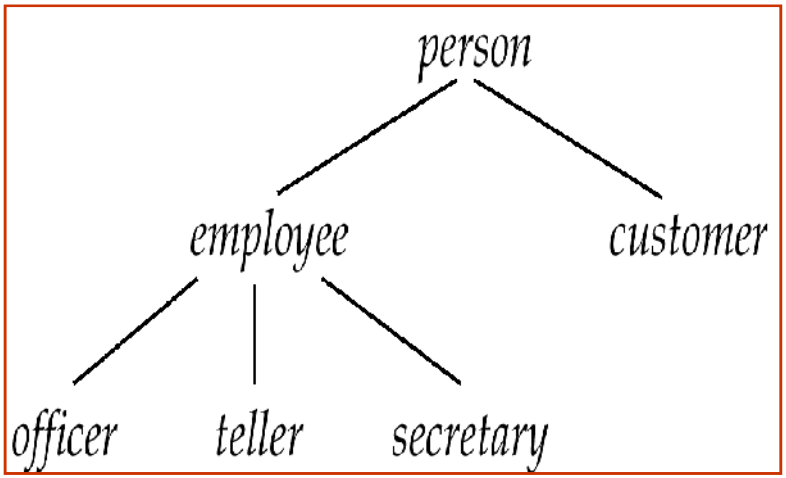
- OQL is a declarative SQL-type language that enables efficient queries on object-oriented databases, including high-level primitives for object sets and structures.
- It is based on SQL-92, providing a superset of the syntax of the SELECT statement.
- OQL does not have primitives to modify the state of the objects since the modifications can be carried out by means of the methods they possess.

Object-Oriented Data Model

name	street	city	amount
Lowerly	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	SideHill	Brooklyn	801
Hodges	SideHill	Brooklyn	647



Is-part-of relationship



ISA relationship

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- **Object-relational model**
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

Object-Relational Data Models

- Extend the relational data model by including object orientation and constructs to deal with added data types.
- Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
- Provide upward compatibility with existing relational languages.
- Ejemplos PostgreSql, Oracle

STANDARDS

"ANSI-American National Standards Institute" specifies the standards that define the elements and characteristics of the Object - Relational model

OR Data Model

name	street	city	amount
Lowerly	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	SideHill	Brooklyn	801
Hodges	SideHill	Brooklyn	647

title	*author-set*	*publisher*	*keyword-set*
		(*name*, *branch*)	
Compilers	{Smith, Jones}	(McGraw-Hill, New York)	{parsing, analysis}
Networks	{Jones, Frick}	(Oxford, London)	{Internet, Web}

OR Data Model

```
CREATE TYPE t AS OBJECT (
    list of attributes and methods
);
/
```

- Note the slash at the end, needed to get Oracle to process the type definition.

For example here is a definition of a point type consisting of two numbers:

```
CREATE TYPE PointType AS OBJECT (
    x NUMBER,
    y NUMBER
);
/
```

An object type can be used like any other type in further declarations of object-types or table-types. For instance, we might define a line type by:

```
CREATE TYPE LineType AS OBJECT (
    end1 PointType,
    end2 PointType
);
/
```

Then, we could create a relation that is a set of lines with ``line ID's" as:

```
CREATE TABLE Lines (
    lineID INT,
    line   LineType
);
```

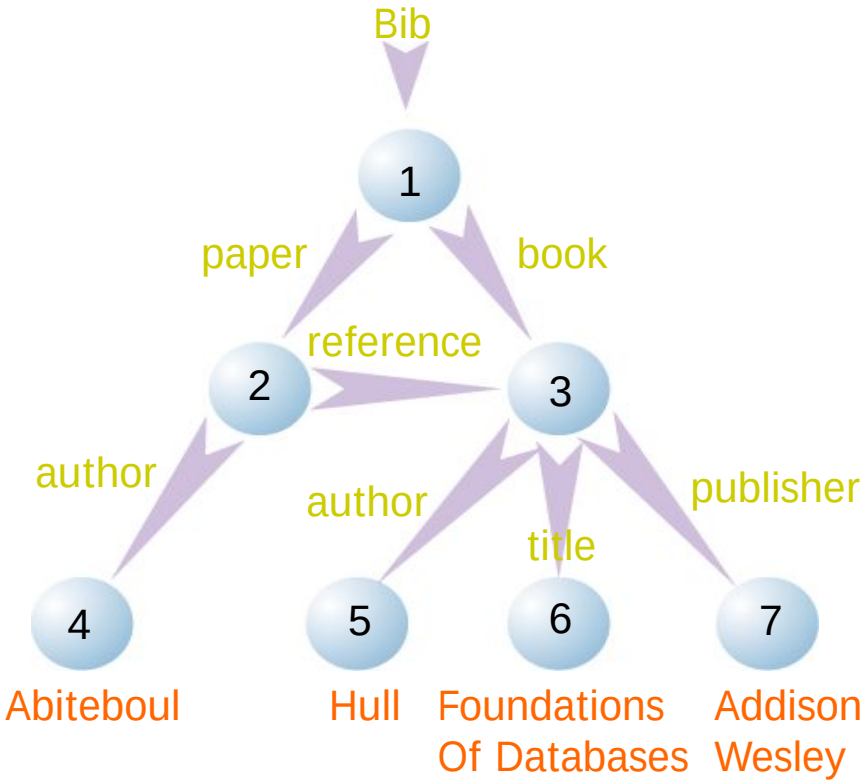
Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- **Semi structured data model (XML)**
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

XML Data Model

```
<Bib>
        <paper id="o2" references="o3">
            <author>Abiteboul </author>
        </paper>
        <book id="o3">
            <author> Hull </author>
            <title> Foundations of Data
                Bases </title>
            <publisher> Addison Wesley
        </publisher>
        </book>
</Bib>
```

XML data



OEM Model

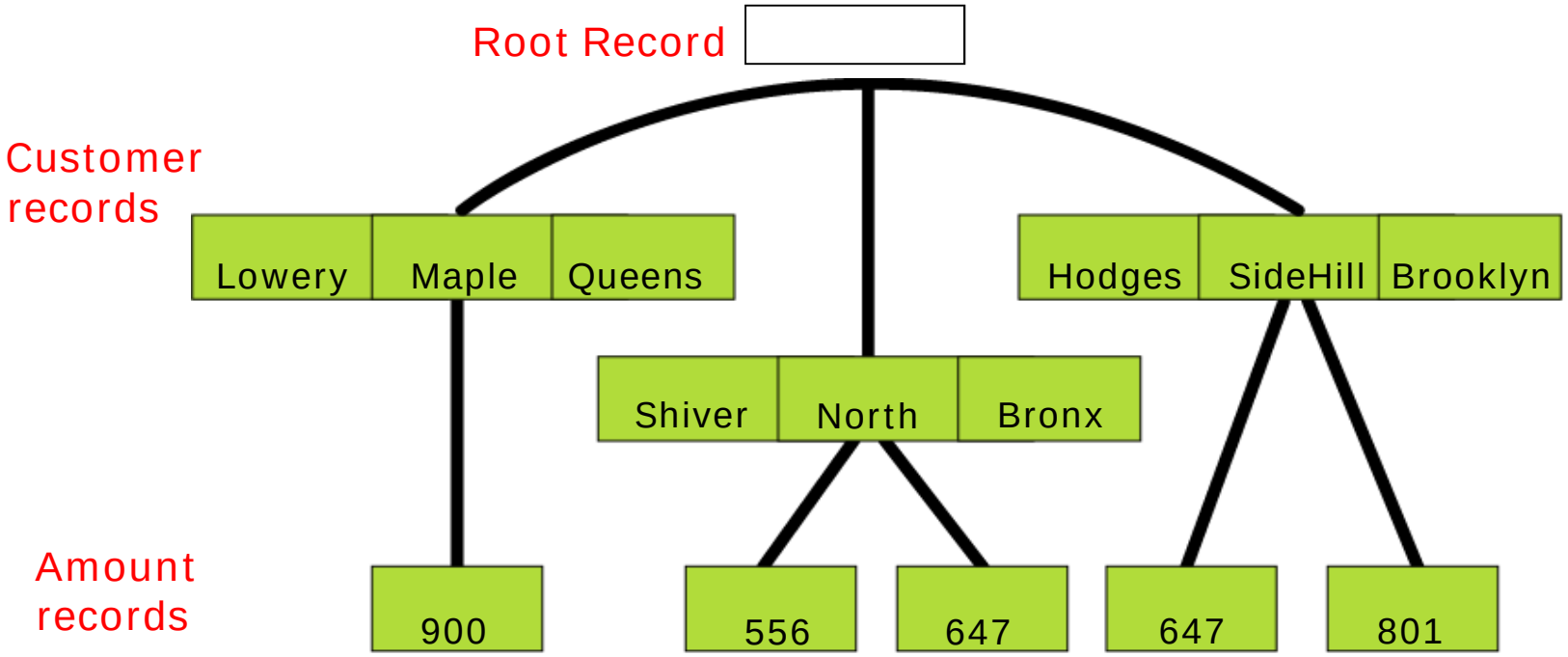
XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - **Network model**
  - **Hierarchical model**
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

Network Data Model



Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- **No SQL**
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - Graphs - oriented

NoSQL

- NoSQL (Not Only SQL) databases allow the use of both standard query languages and non-standard query languages called NoSQL.
- Databases solve unstructured, semi-structured and structured data storage problems.
- They are characterized by allowing them to scale horizontally by adding more equipment and increasing RAM, CPU, among other resources, unlike relational database management systems that only allow it to be done vertically.
- RDBMS ensure high reliability because transactions meet the ACID (Atomicity, Consistency, Insolation, and Durability) properties.
- NoSQL databases do not always comply with ACIDs. It could be said that they are between the ACID and the BASE (Availability as priority (Basic Availability), prioritize the propagation of data (Soft state) and eventual consistency (Eventually consistency))
- The main difference between RDBMS and NoSql databases is in the way data is modeled and the possibility of managing unstructured information, NoSQL databases use different storage structures than tables such as hash tables, graphs, columns, between others.

NoSQL

- Because NoSQL systems generally sacrifice consistency, they often call themselves “eventually consistent” which means that updates eventually spread to all nodes
- Some SMBDs offer mechanisms for a certain degree of consistency, such as Multiversion Concurrency Control (MVCC).
- Supporters of NoSQL often cite Eric Brewer's theorem known as CAP (consistency - availability - fault tolerance) which states that a system can only have two out of three of its characteristics.
- With the advent of the web, especially Web 2.0, millions of users can both read and write data, the scalability of simple operations became more important.

- **SIMPLE OPERATION APPLICATIONS**: operations that involve reading or writing a small number of related records or not.
  - For example, apps looking to update email server databases, people profiles, web posts, likes, wikis, customer records, and many other types of data fit the definition of simple operations apps.

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - **Document - oriented**
  - Column - oriented
  - Key – values - oriented
  - Graphs - oriented

Document-Oriented Data Model

- Embedded documents capture the relationships between the data.
- The structure of documents differs in that in a collection documents do not need to have the same set of fields or the same type of data for a field.
- In practice, the documents in a collection share a similar structure, and you can apply validation rules for update and insert operations.
- References store relationships between data by including links or references from one document to another, and applications can resolve these references.
- Database engines like MongoDB y Cassandra use JSON to store and query data in records or documents using a JSON extension known as BSON or Binary JSON.
- JSON has query language (Xquery)

Document-Oriented Data Model

- Concepts that are handled:
  - table: row collection
  - Primary key: one or more columns. In MongoDB, the primary key is automatically set in the _id field.
  - The data model uses standardized data when:
    - The data is duplicated
    - To represent more complex many-to-many relationships
    - To model large hierarchical data sets.
  - Ids are unique, quick to generate and ordered. Ids values consist of 12 bytes, where the first four bytes are a time-tamp reflecting the creation of the ObjectId.

Document-Oriented Data Model


```
{
  "articles": [
    {
      "title": "title of the article",
      "articleID": 1,
      "body": "body of the article",
      "author": "Isaac Asimov",
      "comments": [
        {
            "username": "Fritz",
            "join date": "1/4/2014",
            "commentid": 1,
            "body": "this is a great article",
            "replies": [
                {
                    "username": "Freddy",
                    "join date": "11/12/2013",
                    "commentid": 2,
                    "body": "seriously? it's rubbish"
                }
            ]
        },
        {
            "username": "Stark",
            "join date": "19/06/2011",
            "commentid": 3,
            "body": "I don't agree with the conclusion"
        }
      ]
    }
  ]
}
```

Whereas relational databases chop up data, Document stores save documents as a single entity

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - **Column - oriented**
  - Graphs - oriented

Column-oriented Data Model

- In a column-oriented database it's easy to add another column because none of the existing columns are affected by it.
- Adding an entire record requires adapting all tables. This makes the row-oriented database preferable over the column-oriented database for online transaction processing (OLTP) because this implies adding or changing records constantly.
- The column-oriented database shines when performing analytics and reporting: summating values and counting entries.
- Overnight batch jobs bring the column-oriented database up to date, supporting fast speed lookups and aggregations using, for example, MapReduce algorithms for reports.
- Examples of column-family stores are Apache HBase, Facebook's Cassandra, Hypertable, and the grandfather of wide-column stores, Google BigTable.

Column-oriented Data Model

ROWID	Name	Birthday	Hobbies
1	Jos The Boss	11-12-1985	archery, conquering the world
2	Fritz von Braun	27-1-1978	building things, surfing
3	Freddy Stark		swordplay, lollygagging, archery
4	Delphine Thewiseone	16-9-1986	

Row-oriented databases layout. Every entity (person) is represented by a single row, spread over multiple columns.

Name	ROWID
Jos The Boss	1
Fritz Schneider	2
Freddy Stark	3
Delphine Thewiseone	4

Birthday	ROWID
11-12-1985	1
27-1-1978	2
16-9-1986	4

Hobbies	ROWID
archery	1, 3
conquering the world	1
building things	2
surfing	2
swordplay	3
lollygagging	3

A column-oriented database stores each column separately

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - **Key – values - oriented**
  - Column - oriented
  - Graphs - oriented

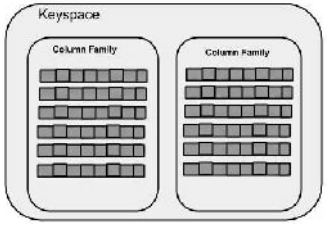
Key-value - oriented Data Model

- Key-Value databases use hash tables and pointers that direct to a set of data.
- The records can only be accessed by means of the key that identifies it.
- The information is condensed in a super list, which contains the indexes that allow finding the required information.
- Having the information stored in a super list allows you to vary the structure of the saved information without losing its grouping.
- The scalability for information retrieval makes it used by social networks such as Facebook and Amazon's “Shopping Cart”.
- Among the great advantages offered by this type of database is The ability to manage data at large scales Dynamic scaling, which allows adjusting the size of the system according to the data load (Number of accesses that have both day and night).

Key-value - oriented Data Model

- Key-value stores are the least complex of the NoSQL databases.
- They are, a collection of key-value pairs, and this simplicity makes them the most scalable of the NoSQL database types, and capable of storing huge amounts of data.
- The value in a key-value store can be anything: a string, a number, but also an entire new set of key-value pairs encapsulated in an object.
- Examples of key-value stores are Redis, Voldemort, Riak, and Amazon's Dynamo, Azure Store Table, Sophia, Casandra

Key-value - oriented Data Model



CASANDRA

- Keyspace is the container for data in Cassandra.
- The basic attributes of a Keyspace in Cassandra are:
  - Replication factor − It is the number of machines in the cluster that will receive copies of the same data.
  - Replica placement strategy: the strategy to place replicas in the ring. We have strategies such as simple strategy and network topology strategy.
  - **Column families**: A column family, is other a container of a collection of rows. Each row contains ordered columns. Column families represent the structure of your data. Each keyspace has at least one or many column families.

Key-value - oriented Data Model

CASANDRA

- **Column:** is the basic data structure of Cassandra with three values, namely key or column name, value, and a time stamp.
- **SuperColumn:** A super column is a special column, therefore, it is also a key-value pair. But a super column stores a map of sub-columns.
- Generally column families are stored on disk in individual files. Therefore, to optimize performance, it is important to keep columns that you are likely to query together in the same column family

Column		
name : byte[]	value : byte[]	clock : clock[]

Super Column	
name : byte[]	cols : map<byte[], column>

Database Classification by Data Models

- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-oriented model
- Object-relational model
- Semi structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
- No SQL
  - Document - oriented
  - Key – values - oriented
  - Column - oriented
  - **Graphs - oriented**

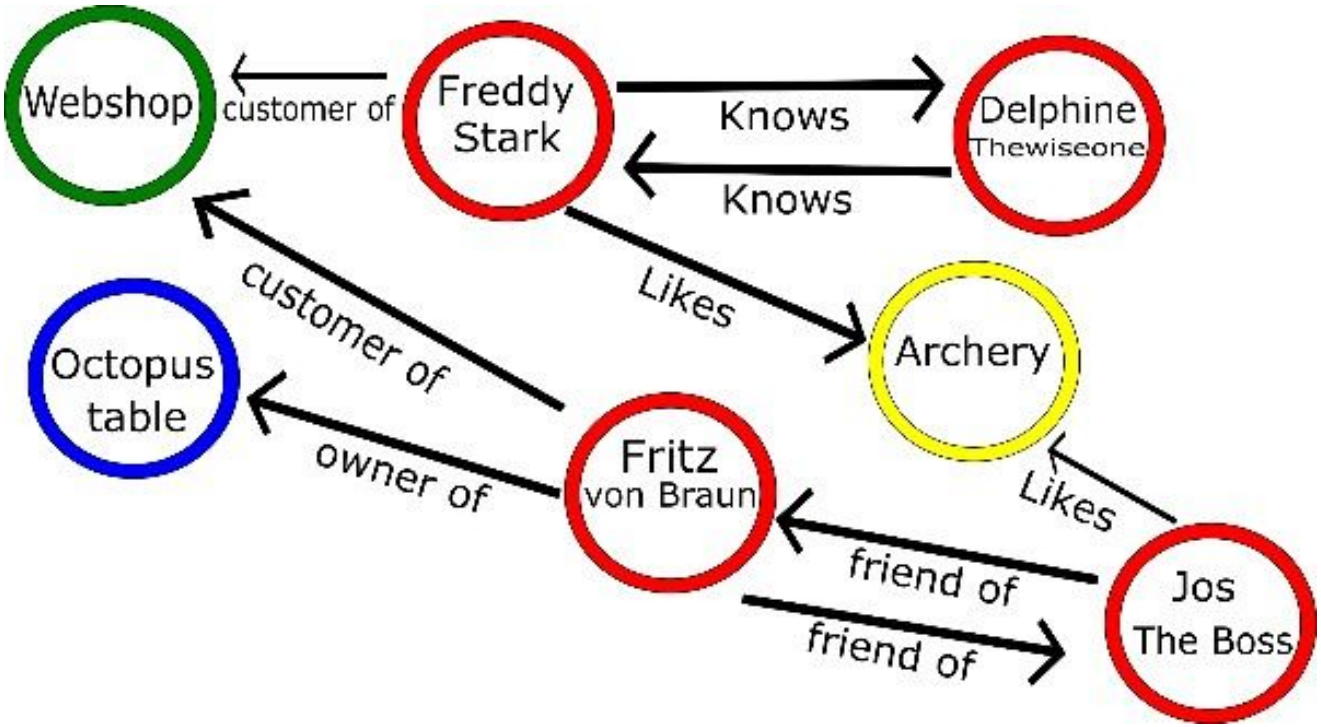
Graph - oriented Data Model

- Graph-oriented NoSQL databases use mathematical graph theory, abstracting the data into nodes, edges, and properties.
- They store information in multi-attribute tuples (nodes).
- Graph-oriented databases are useful for handling highly interconnected data and are therefore efficient in having multiple relationships between different entities.
- Provide an effective and flexible mechanism for managing interconnected information, because the relationship between the data is explicitly represented.
- Queries are based on existing relationships on nodes.

Graph - oriented Data Model

- Example. When the data is highly interconnected, such as for social networks or scientific paper citations.
- Graph data has two main components:
  - Node: The entities themselves. In a social network this could be people.
  - Edge: The relationship between two entities. This relationship is represented by a line and has its own properties. An edge can have a direction, for example, if the arrow indicates who is whose boss.
- Graphs can become incredibly complex given enough relation and entity types.
- Graph databases like Neo4j comply with ACID properties
- Examples: HyperGraphDB, AllegroGraph, ArangoDB

Graph - oriented Data Model



Graph data example with four entity types (person, hobby, company, and furniture) and their relations without extra edge or node information.

American National Standards Institute (ANSI)

HISTORY

- ANSI It was founded in 1918 by five engineering societies and three government agencies that banded together to form the American Engineering Standards Committee.
- The committee changed its name to the American Standards Association in 1928.
- In 1966 it reorganized and was renamed the United States of America Standards Institute.
- It took on its current moniker in 1969 (ANSI).

- ANSI's headquarters are in Washington, D.C

Fuente: https://www.investopedia.com/terms/a/ansi.asp

American National Standards Institute (ANSI)

PURPOSE

- The American National Standards Institute provides accreditation for standards developed by other organizations, companies, consumer groups, government agencies and other bodies.
- ANSI's membership is made up of more than 270,000 companies and organizations, and over 30 million professionals in government agencies, companies, organizations, academic and international bodies, and individuals.
- For more, see ANSI's website www.ansi.org.

RELATIONAL PARADIGM

SUMMARY DATES

- 1970: Edgar Frank Codd, a computer scientist working for IBM publishes his famous paper, “A Relational Model of Data for Large Shared Data Banks” in June 1970, in the Association of Computer Machinery (ACM) journal. This model remains the definitive model for relational database management systems (RDBMS).

- 1978: IBM Corporation develops the database system R (at their San Jose Research Center in California), and language, Structured English Query Language (SEQUEL) . This was based on Codd’s research.

RELATIONAL PARADIGM

SUMMARY DATES - RELATIONAL PARADIGM

- 1979: Relational Software, Inc., releases the first relational database management system. RDBMS functions on a minicomputer using SQL as the query language. The product becomes so popular, the company changes its name to Oracle.

- 1982: IBM releases its first commercial SQL-based RDBMS they name the SQL/Data System, or SQL/DS,
- In 1985 they release the Database 2 system, or DB2. Both systems run on an IBM mainframe computer.
- then IBM DB2 allow additional systems, including those running on Windows and UNIX operating systems.

RELATIONAL PARADIGM

ANSI STANDARDS YEAR BY YEAR

- 1986: SQL-87 was formalized by ANSI.
- 1989: The American National Standards Institute (ANSI) publishes the first set of standards for database query languages, known as SQL-89 or FIPS 127-1.
- 1992: ANSI publishes their revised standards, ANSI/ISO SQL-92 or SQL2, which were more stricter, adding some new features. These standards introduce levels of compliance that indicated the extent to which a dialect meets ANSI standards.
- 1999: ANSI publishes SQL3, or ANSI/ISO SQL: 1999, with new features, like support for objects. The replaced the levels of compliance with core specifications, as well as additional specifications for nine more packages.

RELATIONAL PARADIGM

ANSI STANDARDS YEAR BY YEAR

- 2003: ANSI publishes SQL: 2003, introducing standardized sequences, XML-related features and identity columns. The creator of the first RDBMS,
  - E.F.Codd, passes away on April 18 of the 2003 year.
- 2006: ANSI publishes SQL: 2006, defining how to use SQL with XML and enabling applications to integrate XQuery into their existing SQL code.
- 2008: ANSI publishes SQL: 2008, introducing triggers, as well as the TRUNCATE statement.
- 2011: ANSI publishes SQL: 2011 or ISO/IEC 9075:2011, the seventh revision of the ISO (1987) and ANSI (1986) standard for the SQL database query language

LANGUAGES OF THE SQL STANDARD

- Structured Query Language (SQL)
- Data Manipulation Language (DML)
- Data Definition Language (DDL)
- Transaction Control Language(TCL)
- Data Control Language (DCL)

Structured Query Language (SQL)

- SQL defines following ways to manipulate data stored in an RDBMS.
- widely used non-procedural language
  - Example: Find the name of the instructor with ID 22222

    select *name*
    from *instructor*
    where *instructor.ID* = '22222'
  - Example: Find the ID and building of instructors in the Physics dept.

    select *instructor.ID*, *department.building*
    from *instructor*, *department*
    where *instructor.dept name* = "physics"

Commands

select

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
- DML commands are used for manipulating the data stored in the table and not the table itself.
- DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.
- DML is also part of the SQL standard

Commands

insert	to insert a new row
update	to update existing row
delete	to delete a row
merge	merging two rows or two tables

Data Definition Language (DDL)

- This includes changes to the structure of the table like creation of table, altering table, deleting a table etc.
- All DDL commands are auto-committed. That means it saves all the changes permanently in the database.
- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL is also part of the SQL standard

Commands

Command	Description
create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table

Transaction Control Language(TCL)

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling the data back to its original state. It can also make any temporary change permanent.

TCL is also part of the SQL standard

Commands

Command	Description
commit	to permanently save
rollback	to undo change
savepoint	to save temporarily

Fuente: https://www.studytonight.com/dbms/introduction-to-sql.php

Data Control Language (DCL)

Data control language are the commands to grant and take back authority from any database user.

DCL is also part of the SQL standard

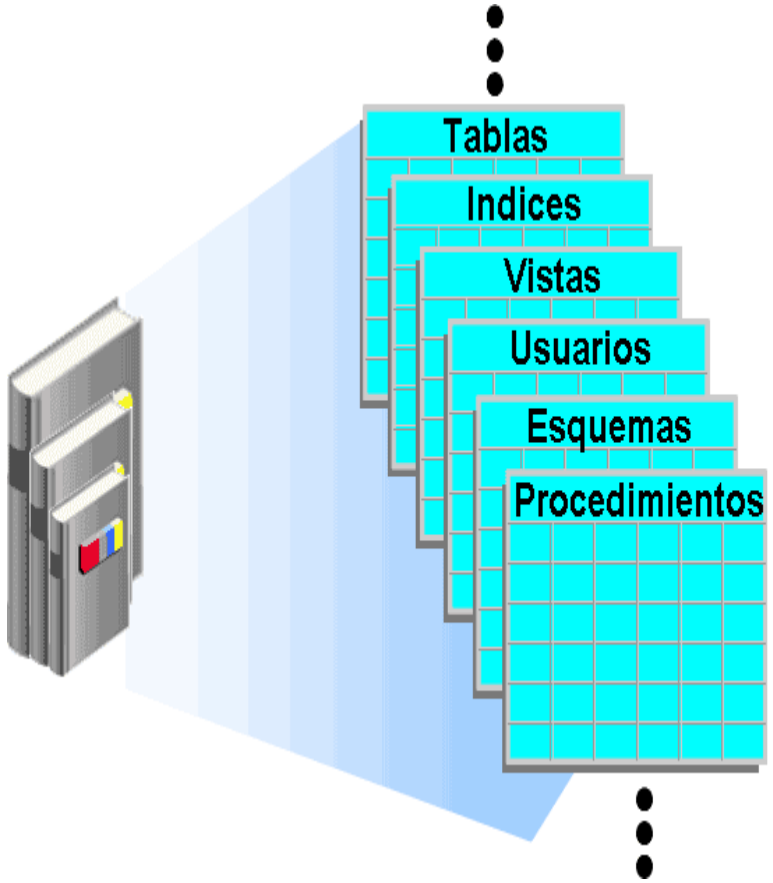
Commands

Command	Description
grant	grant permission of right
revoke	take back permission.

Fuente: https://www.studytonight.com/dbms/introduction-to-sql.php

Data Dictionary (DD)

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
    - Referential integrity (**references** constraint in SQL)
      - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation
  - Authorization



End of Chapter 1