Levels of Abstraction

- **Physical level:** describes how a record (e.g., instructor) is stored.

- how are they stored on the secondary storage-block of continuous storage locations (DBA may know))

- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
    ID : string;
    name : string;
    dept_name : string;
    salary : integer;
end;
```

((((COURSE, STUDENT DATA

SNO	NAME	PHNO
1	A	1234
2	B	5678

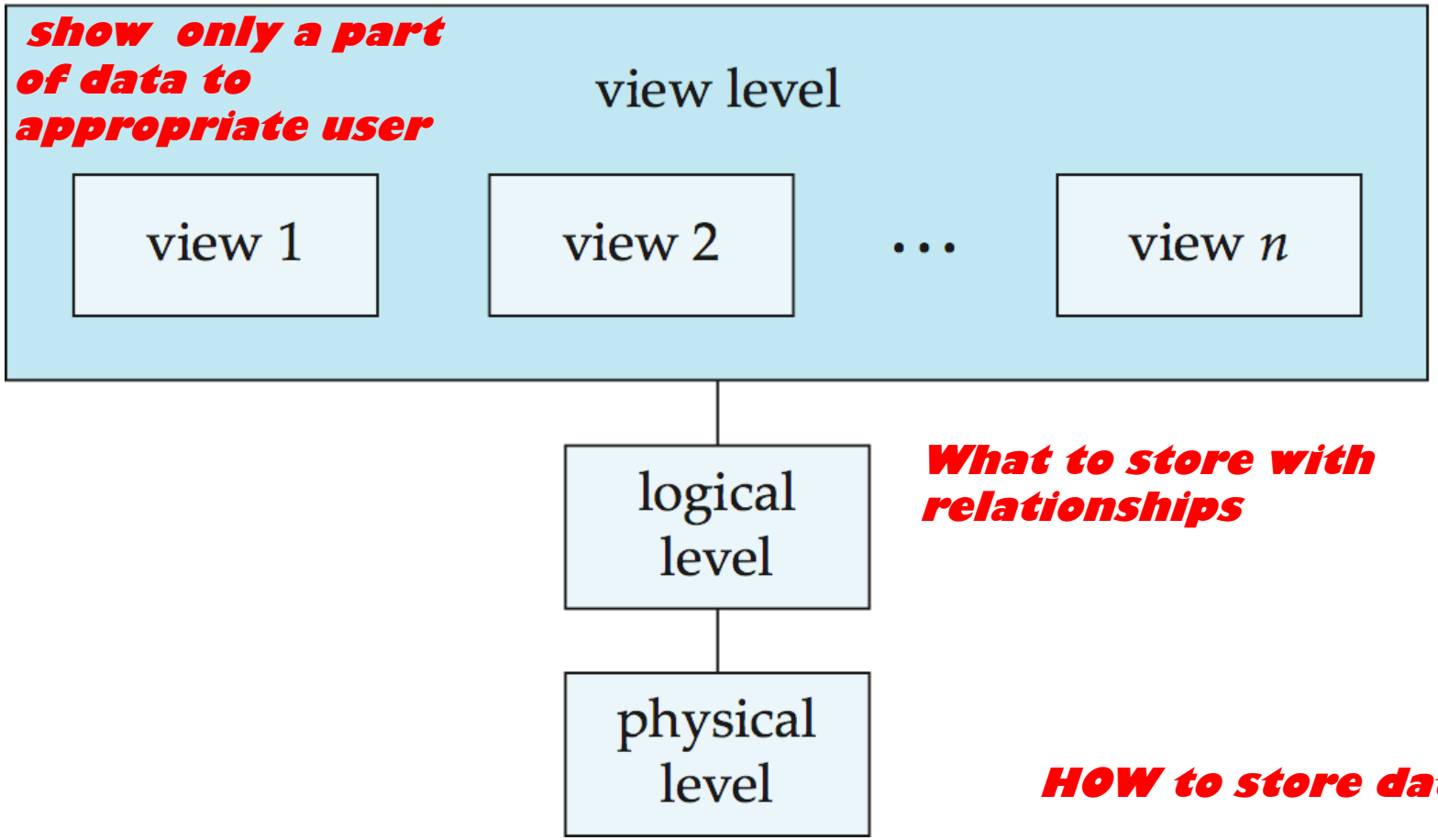
- Logically the structure that's easy to retrieve(table -structure)

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

- Showing only a part of the table

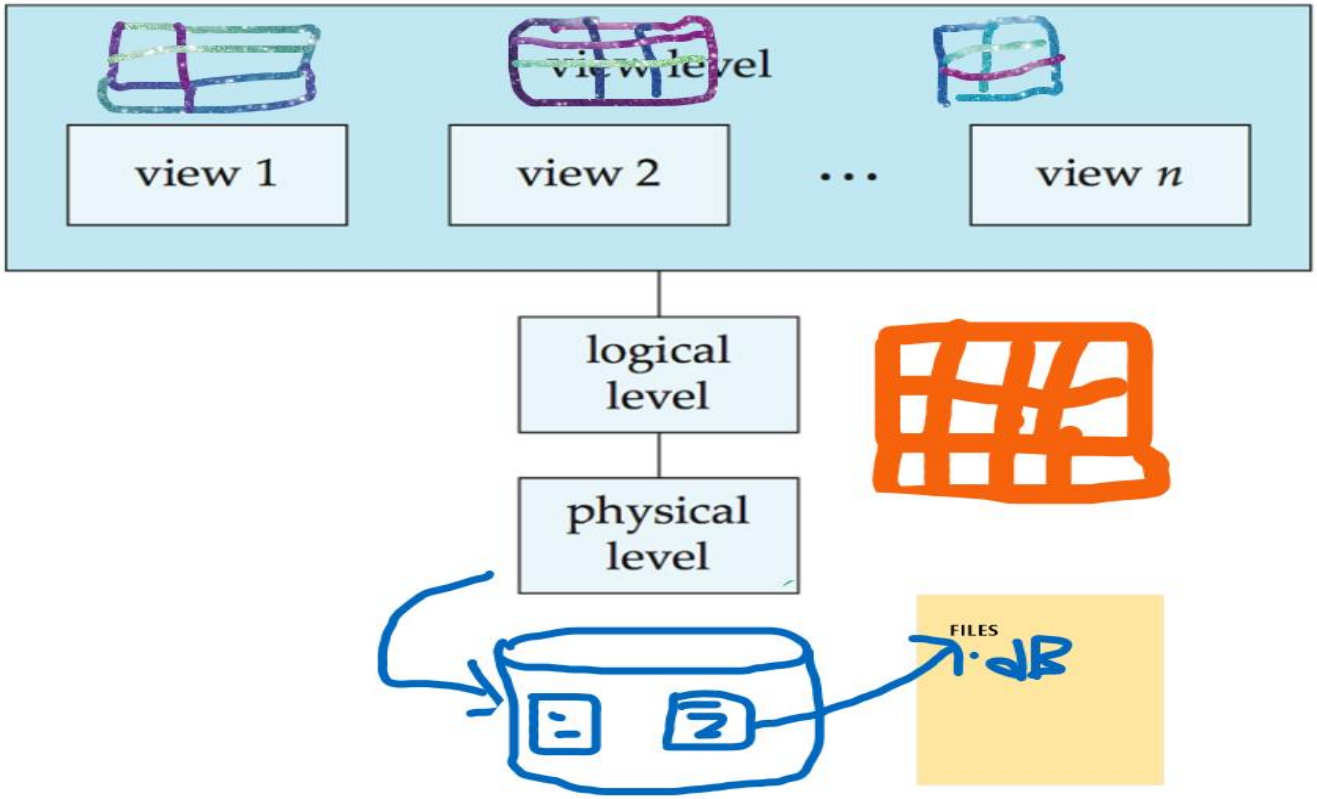
View of Data

An architecture for a database system



View of Data

An architecture for a database system



Instances and Schemas

- Databases change over time
- Database Instance is collection of information stored in database at a particular moment

  int i;

  i=20;

  ---

  i=30;

- Several schemas partitioned according to the levels of abstraction

  Physical schema

  Logical schema

  Sub schemas

Interfaces need to be specified clearly so that changes can be made to any level without affecting other .

Instances and Schemas

- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between the → (table structure)
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time

→ (same table may have 50 instructors on MAY 25th , 150 instructors on NOV 30th) 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.

Review of previous class
Disadvantages of FPS
Advantages of DBMS
Any vendors of DBMS
How are these dbms different ?

Data Models

- A collection of conceptual tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Consistency constraints

DATA MODEL

A way to describe database design at physical,logical and view level

- Relational model(collection of tables)
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model : **network database** is similar to a hierarchical database, except records have a many-to-many rather than a one-to-many relationship
  - Hierarchical model: **hierarchical database** organizes data in a tree structure. Each parent record has one or more child records, similar to the structure of a file system.

stored data in rigid, predetermined relationships, Because no data definition language existed, changing the structure of the data was difficult. Also, these systems lacked a simple query language, which hindered application development.

Object-Relational Data Models

- Relational model: flat, “atomic” values
- 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.
  - Oracle Database implements the object-type model as an extension of the relational model, while continuing to support standard relational database functionality, such as queries, fast commits, backup and recovery, scalable connectivity, row-level locking, read consistency, and more.
  - SQL and various programmatic interfaces and languages, including PL/SQL, Java, Oracle Call Interface, Pro*C/C++, and C# have been enhanced with extensions to support Oracle objects. The result is an object-relational model that offers the intuitiveness and economy of an object interface while preserving the high concurrency and throughput of a relational database.
  - Example : PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics **applications**.

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

Cloud DBMS (CDBMS)

Cloud– platform, infrastructure,applications,database

- A **cloud DBMS** (CDBMS) is distributed database that delivers computing as a **service** instead of product, which allow sharing of resources and information between multiple devices over internet. CDBMS provide managed backup, restore and automated scheduling you may also able to pay little or nothing for unused time.
- Database Management with IBM **Cloud**.
  - MicrosoftAzure/SQLDatabase – A "full featured relational database-as-a-service," with "Tables" that offer NoSQL capabilities for storing large amounts of unstructured data, and "Blobs" (Binary Large Objects) for storing large amounts of unstructured text, video, audio and images.
  - AmazonWebServices/DynamoDB/RelationalDatabaseService – Amazon's offerings include NoSQL, MySQL, Oracle and MS SQL Server solutions. SimpleDB is Amazon's "highly available and flexible non-relational data store that [takes on] the work of database administration."
  - Xeround – A fully managed MySQL DBAAS that the vendor calls a "drop-in solution" because it "automates all configuration and ongoing DB operations."
  - GoogleCloudSQL/GoogleAppEngineDatastore – Google's solutions for storing structured and unstructured data.
  - ClearDB – This MySQL DBAAS boasts 100% uptime due to its "multi-regional read/write mirroring."
  - Database.com – A native cloud database service developed in house at Salesforce.com that became generally available in 2011. The vendor's website says it was "built with the needs of a social and mobile world at its core, not as an afterthought."

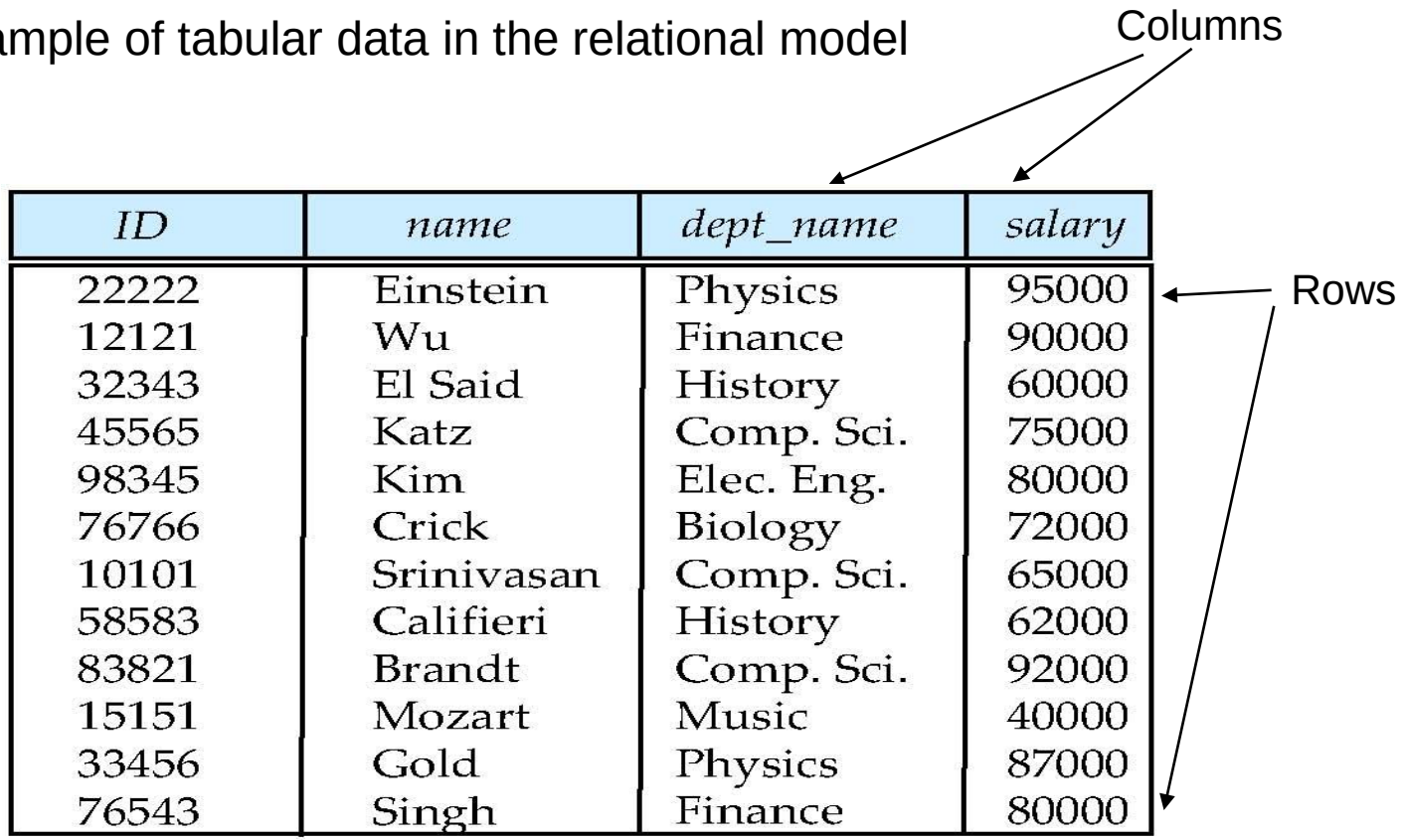
NO SQL DATA BASES

- NoSQL is a non-relational database management system. NoSQL was designed specifically to handle storing and retrieving large quantities of data without defined relationships (i.e., Big Data).

Operational Database Model	Example DBMSs
Key-value store	Redis, MemcacheDB
Columnar database	Cassandra, Apache HBase
Document store	MongoDB, Couchbase
Graph database	OrientDB, Neo4j

Relational Model

- All the data is stored in various tables.
- 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

Relational Model is basis for RDBMS

In his seminal 1970 paper "A Relational Model of Data for Large Shared Data Banks," E. F. Codd defined a relational model based on mathematical set theory. Today, the most widely accepted database model is the relational model.

A **relational database** is a database that conforms to the relational model. The relational model has the following major aspects:

- **Structures:** Well-defined objects store or access the data of a database.
- **Operations:** Clearly defined actions enable applications to manipulate the data and structures of a database.
- **Integrity rules** : Integrity rules govern operations on the data and structures of a database.

A relational database stores data in a set of simple relations. A **relation** is a set of tuples. A **tuple** is an unordered set of **attribute** values.

A **table** is a two-dimensional representation of a relation in the form of rows (tuples) and columns (attributes). Each row in a table has the same set of columns. A relational database is a database that stores data in relations (tables). For example, a relational database could store information about company employees in an employee table, a department table, and a salary table.

http://portal.acm.org/citation.cfm?id=362685

CODD RULES

Rule Zero

•The system must qualify as relational, as a database, and as a management system. For a system to qualify as a relational database management system (RDBMS), that system must use its relational facilities (exclusively) to manage the database.

•The other 12 rules derive from this rule. The rules are as follows :

Rule 1 : The information rule: All information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.

Rule 2 : The guaranteed access rule: All data must be accessible. This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing row.

Rule 3 : Systematic treatment of null values: The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values (for example, "distinct from zero or any other number", in the case of numeric values), and independent of data type. It is also implied that such representations must be manipulated by the DBMS in a systematic way.

Rule 4 : Active online catalog based on the relational model: The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

Rule 5 : The comprehensive data sub language rule: The system must support at least one relational language that

1. Has a linear syntax
2. Can be used both interactively and within application programs,
3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).

CODD RULES

Rule 6 : The view updating rule: All views those can be updated theoretically, must be updated by the system.

Rule 7 : High-level insert, update, and delete: The system must support set-at-a-time insert, update, and delete operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8 : Physical data independence: Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

Rule 9 : Logical data independence: Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.

Rule 10 : Integrity independence: Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

Rule 11 : Distribution independence: The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully :

1. when a distributed version of the DBMS is first introduced; and
2. when existing distributed data are redistributed around the system.

Rule 12: The non-subversion rule: If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

A Sample Relational Database

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

Database Languages –

Data Definition Language (DDL)

- 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 compiler generates a set of table templates stored in a ***data dictionary***

→ ***It contains metadata(Metadata means data about data)***

- Data dictionary contains metadata (i.e., data about data)
  - Database schema (which columns, what type..)
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what (student can view marks , but not update,delete)
- Application programs may be written in host languages to communicate with database(to overcome limitations of SQL------ ODBC)

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

SQL

- The most widely used commercial language
- **users describe in SQL what they want to be done, and the SQL language compiler automatically generates a procedure to navigate the database and perform the desired task.**
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Dialects of SQL

- The following are the most popular dialects of SQL:
- PL/SQL stands for procedural language/SQL. It is developed by Oracle for the Oracle Database.
- Transact-SQL or T-SQL is developed by Microsoft for Microsoft SQL Server.
- PL/pgSQL stands for Procedural Language/PostgreSQL that consists of SQL dialect and extensions implemented in PostgreSQL
- MySQL has its own procedural language since version 5. Note that MySQL was acquired by Oracle.

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two ways of doing so:
  - Entity Relationship Model
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory
    - Formalize what designs are bad, and test for them

Database Engine

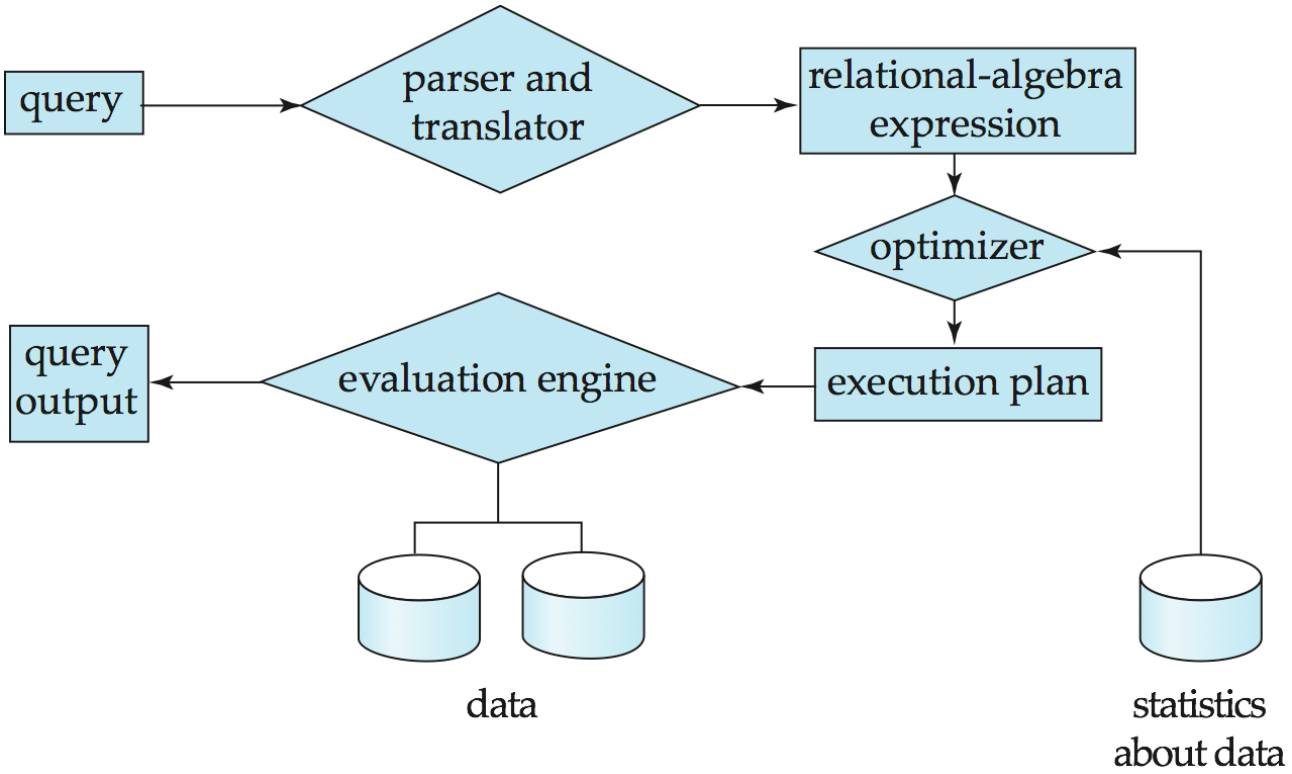
- Storage manager
- Query processing
- Transaction manager

Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



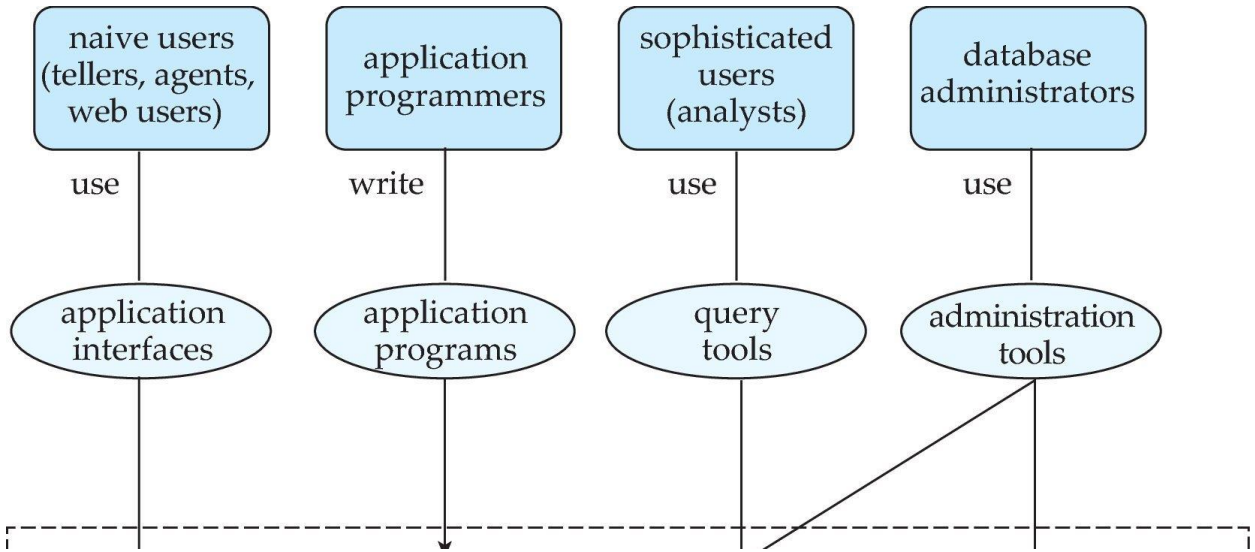
Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

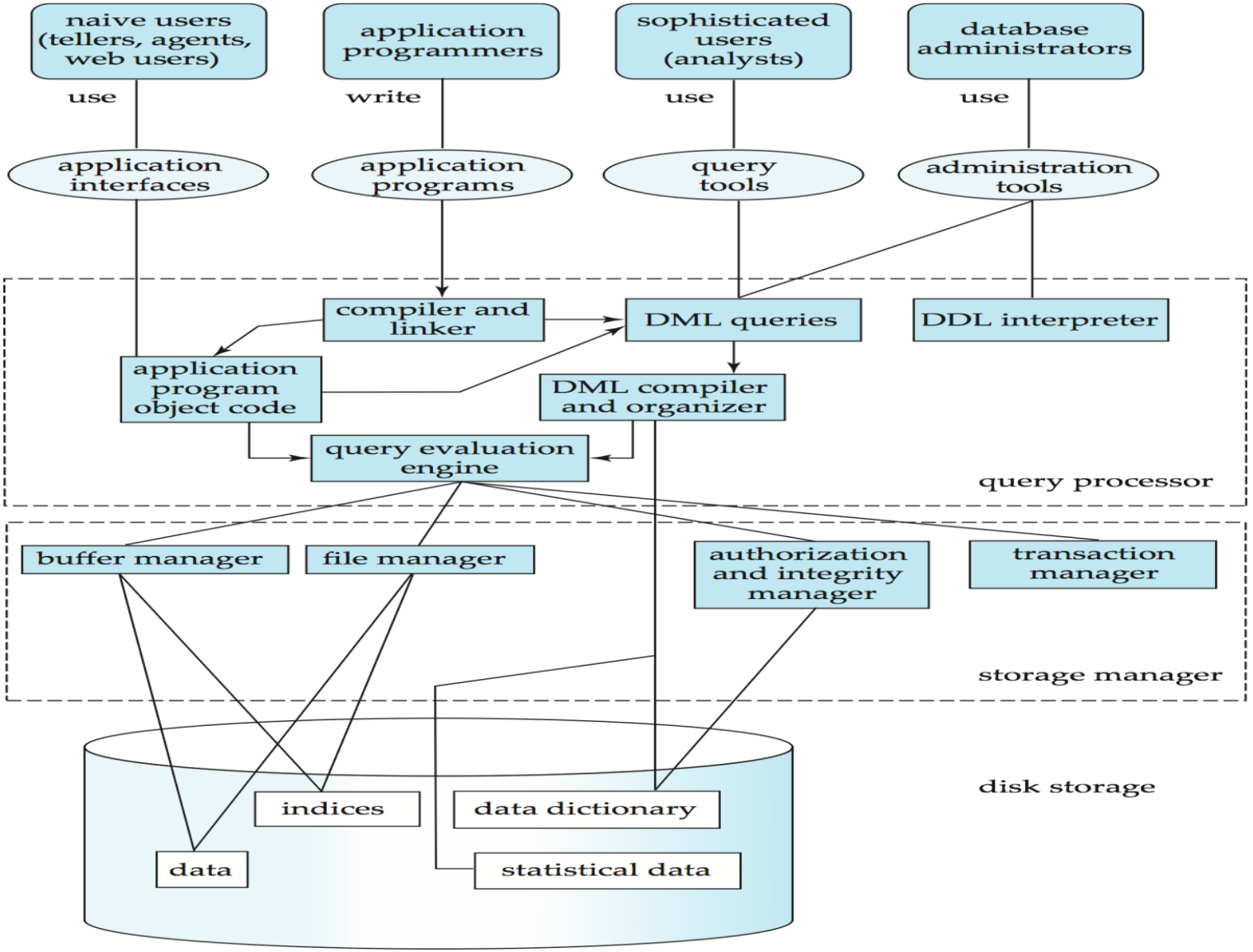
Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Users and Administrators



Database System Internals



Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

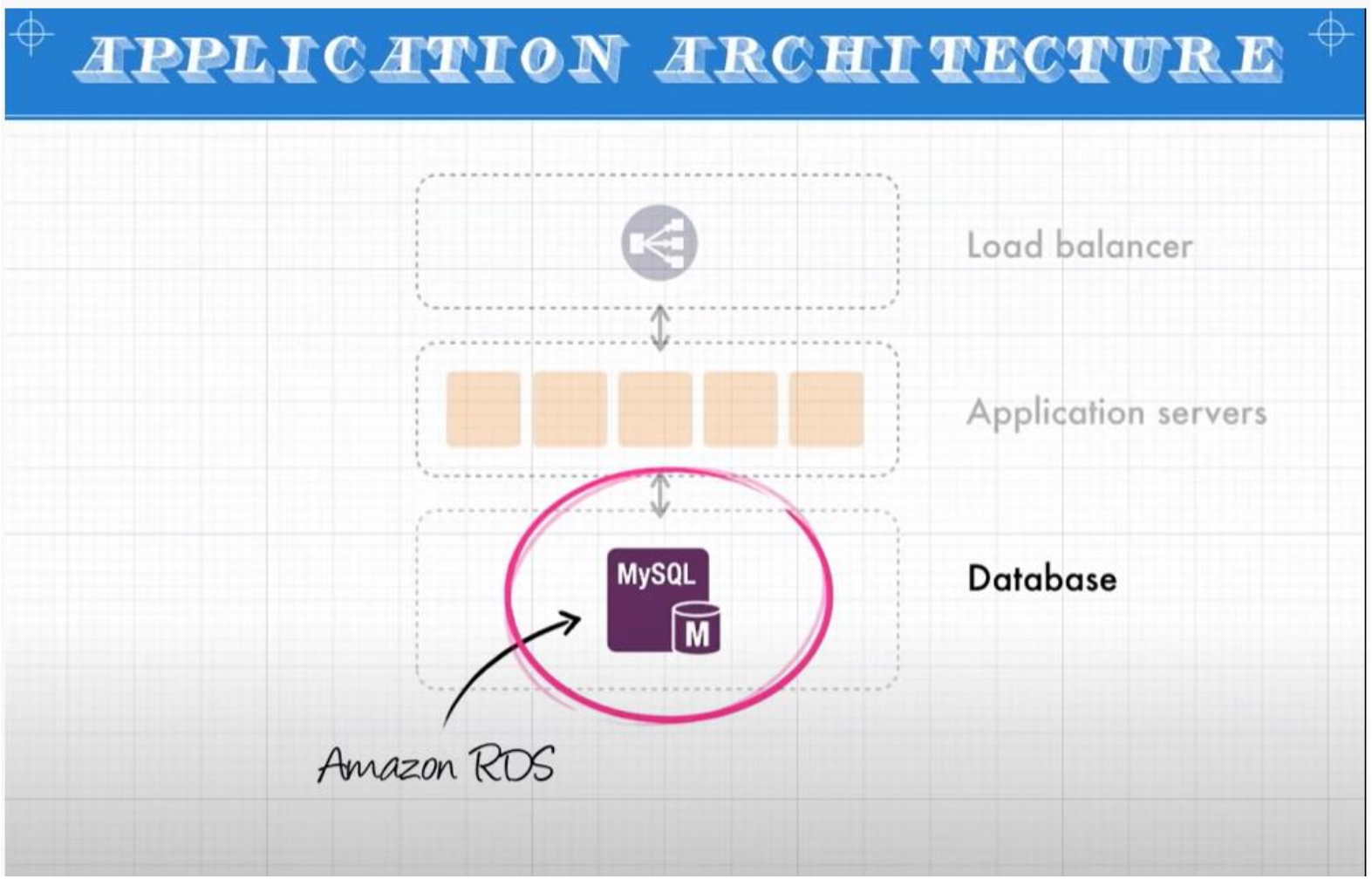
History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing

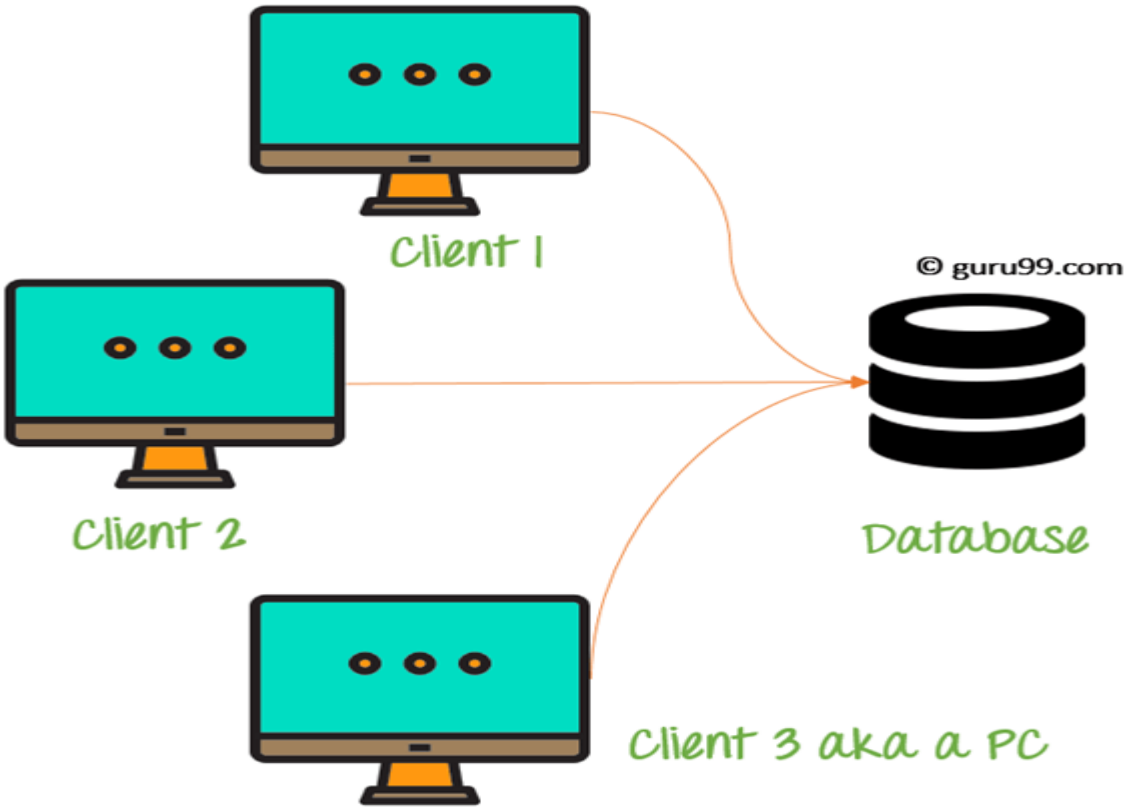
History (cont.)

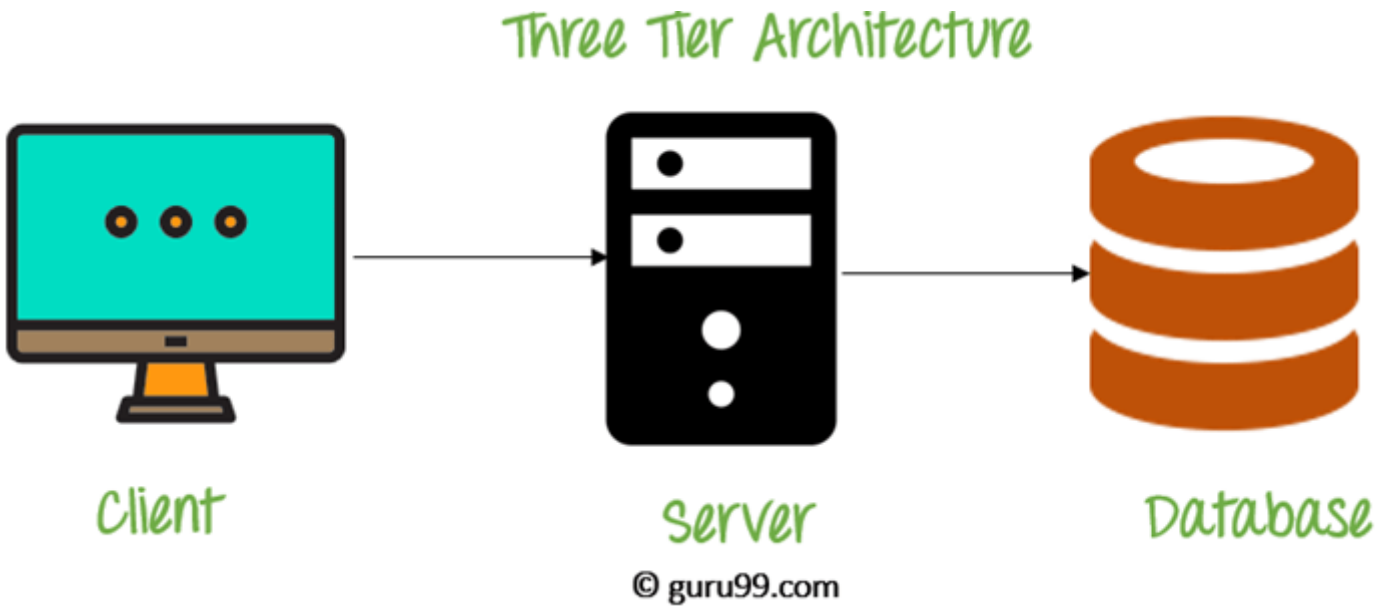
- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

AMAZON RELATIONAL DATABASE SERVICE ON AWS



Single Tier Architecture




Three Tier Architecture
Client
Server
Database
© guru99.com