Chapter 1: Introduction

Database System Concepts, 6th Ed.

Outline

- The Need for Databases
- Data Models
- Relational Databases
- Database Design
- Storage Manager
- Query Processing
- Transaction Manager

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access 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

Purpose of Database management systems

Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating 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

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

Database Users

Users are differentiated by the way they expect to interact with the system.

Application programmers: interact with system through DML calls.

Application Programmers are responsible for writing application programs that use the database. These programs could be written in General Purpose Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc. to manipulate the database. These application programs operate on the data to perform various operations such as retaining information, creating new information, deleting or changing existing information.

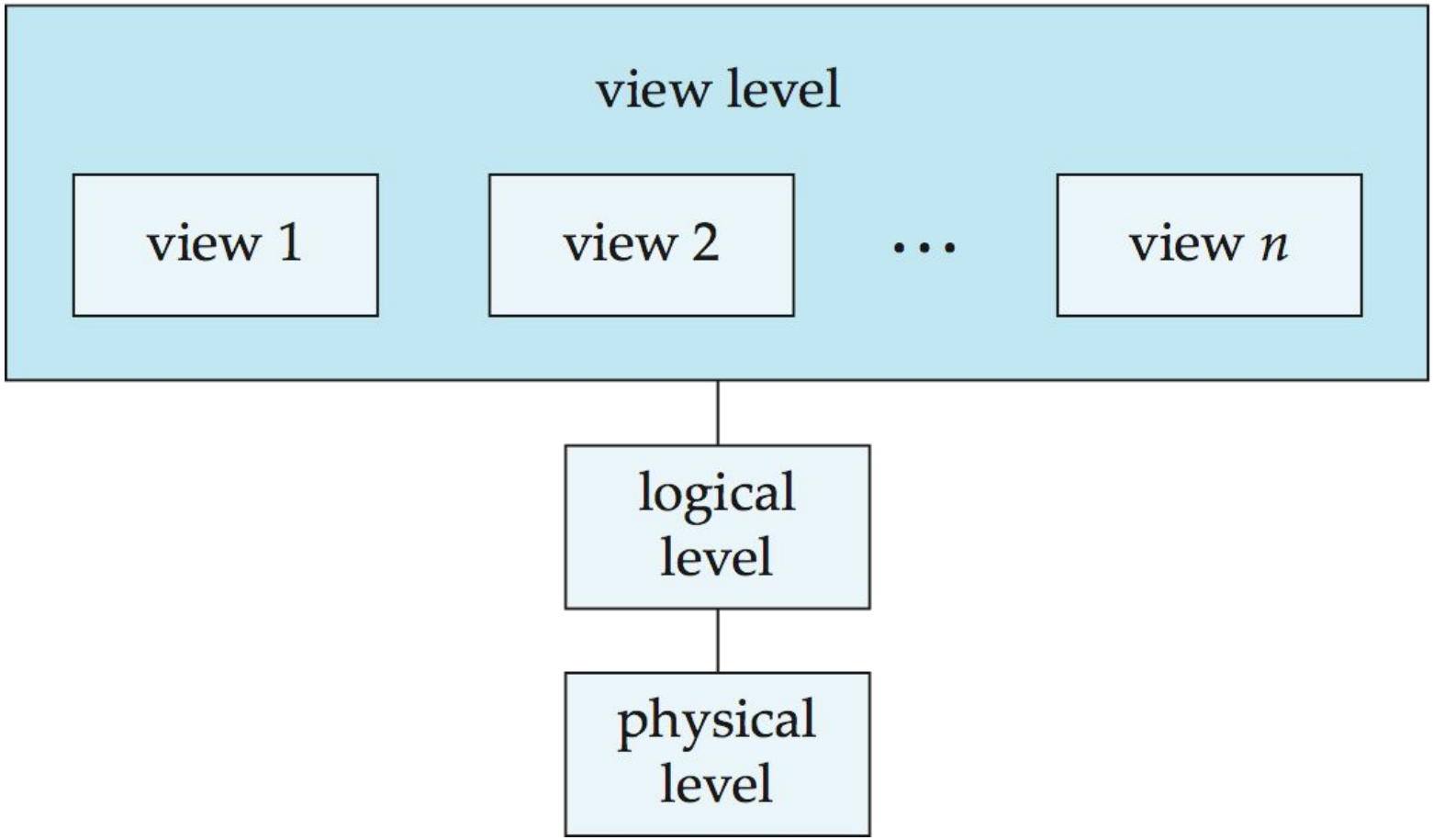
Specialized users: write specialized database applications that do not fit into the traditional data processing framework ,
Among these applications are computer aided-design systems, knowledge-base and expert systems etc.

Sophisticated users: sophisticated users form their requests in a database query language and submit them to a query processor to make the storage manager understand the requests. Analysts who submit queries to explore data in the database fall in this category.

Naive users: This type of users generally interacts with the system through previously created programs. Such as, a user may use the system to transfer some balance from one account to another. This will be done a program, like ‘Transfer’. The user will only fill some required fields of the program like the balance, the accounts etc. The typical interfaces for these users are forms including some required fields to be filled.

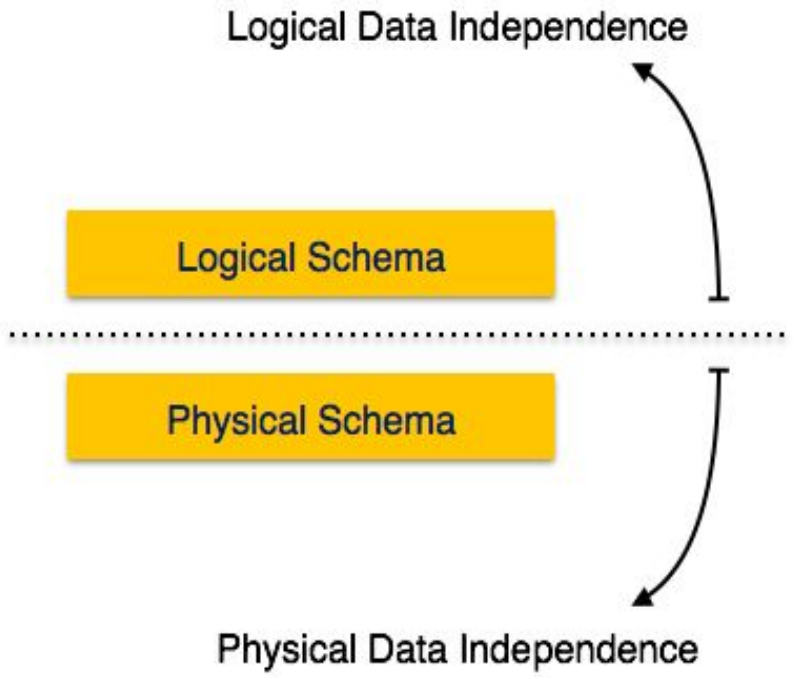
View of Data

An architecture for a database system



If a database system is not multi-layered, then it becomes difficult to make any changes in the database system

- Data Independence
- A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



Data Independence conti..

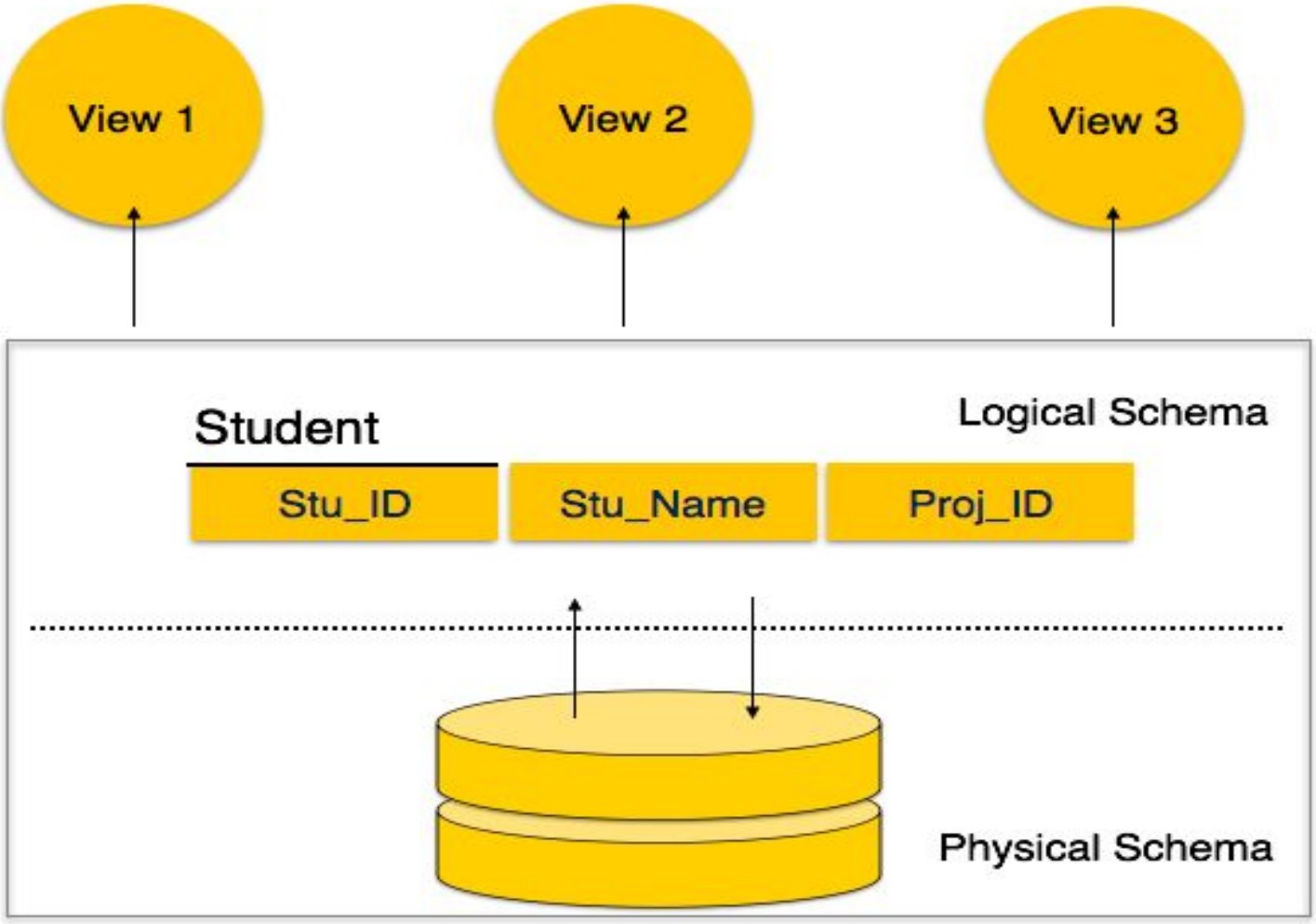
- Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.
- Logical Data Independence
- Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.
- Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

- All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.
- For example, in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas.

Database Schema

- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.


View 1
View 2
View 3
Student
Logical Schema
Stu_ID
Stu_Name
Proj_ID
Physical Schema

A database schema can be divided broadly into two categories –

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Languages

- DDL
- DDLs are used to define the metadata of the database. i.e.; using this, we create schema, tables, constraints, indexes in the database. DDLs are also used to modify Schema, tables index etc. Basically, using DDL statements, we create skeleton of the database. It helps to store the metadata information like number of schemas and tables, their names, columns in each table, indexes, constraints etc in the database.
- CREATE – to create database and its objects like (table, index, views, store procedure, function and triggers)
- ALTER – alters the structure of the existing database
- DROP – delete objects from the database
- TRUNCATE – remove all records from a table, including all spaces allocated for the records are removed
- COMMENT – add comments to the data dictionary
- RENAME – rename an object

- DML
- When we have to insert records into table or get specific record from the table, or need to change some record, or delete some record or perform any other actions on records in the database, we need to have some media to perform it. DML helps to handle user requests. It helps to insert, delete, update, and retrieve the data from the database.
- SELECT – retrieve data from the a database
- INSERT – insert data into a table
- UPDATE – updates existing data within a table
- DELETE – Delete all records from a database table

- DCL
- languages are used to control the user access to the database, tables, views, procedures, functions and packages. They give different levels of access to the objects in the database
- Suppose we have created a table STUDENT in the database. Now who can view or access these tables? No one other than who has created it! We need to explicitly tell the database, who can view this table and what kind of access like read, write, delete should be given to the other users.
- Similarly, some of the users who were accessing this table have left the organization. But still they have access to this table, which is not acceptable. Hence we need to remove their access on such tables and database.
- GRANT – allow users access privileges to database
- REVOKE – withdraw users access privileges given by using the GRANT command

- TCL is short name of Transaction Control Language which deals with transaction within a database.
- Suppose we have inserted some records in to Employee table. Now we need to save them. How? Similarly, we have updated something wrong on the table. After updating we realized that its wrong. Now we need to unsave the changes that have been done. How? We have deleted something or inserted something which is not correct. It has to be undone. All these
- COMMIT – commits a Transaction
- ROLLBACK – rollback a transaction in case of any error occurs
- SAVEPOINT – to rollback the transaction making points within groups
- SET TRANSACTION – specify characteristics for the transaction
- saving and undoing the tasks can be done by TCL.

DATABASE ADMINISTRATOR

- The life cycle of database starts from designing, implementing to administration of it.
- A database for any kind of requirement needs to be designed perfectly so that it should work without any issues.
- Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database.
- The database grows as the data grows in the database.
- When the database becomes huge, its performance comes down. Also accessing the data from the database becomes challenge.
- There will be unused memory in database, making the memory inevitably huge. These administration and maintenance of database is taken care by database Administrator
- A DBA has many responsibilities. A good performing database is in the hands of DBA.

- **Installing and upgrading the DBMS Servers: -** DBA is responsible for installing a new DBMS server for the new projects. He is also responsible for upgrading these servers as there are new versions comes in the market or requirement. If there is any failure in upgradation of the existing servers, he should be able revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hot fixes/ patches to the DBMS servers.
- **Design and implementation: -** Designing the database and implementing is also DBA's responsibility. He should be able to decide proper memory management, file organizations, error handling, log maintenance etc for the database.
- **Performance tuning: -** Since database is huge and it will have lots of tables, data, constraints and indices, there will be variations in the performance from time to time. Also, because of some designing issues or data growth, the database will not work as expected. It is responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs works in fraction of seconds

- **Migrate database servers: -** Sometimes, users using oracle would like to shift to SQL server or Netezza. It is the responsibility of DBA to make sure that migration happens without any failure, and there is no data loss.
- **Backup and Recovery: -** Proper backup and recovery programs needs to be developed by DBA and has to be maintained him. This is one of the main responsibilities of DBA. Data/objects should be backed up regularly so that if there is any crash, it should be recovered without much effort and data loss.
- **Security: -** DBA is responsible for creating various database users and roles, and giving them different levels of access rights.
- **Documentation: -** DBA should be properly documenting all his activities so that if he quits or any new DBA comes in, he should be able to understand the database without any effort. He should basically maintain all his installation, backup, recovery, security methods. He should keep various reports about database performance.
- **There are different kinds of DBA depending on the responsibility that he owns**

Levels of Abstraction

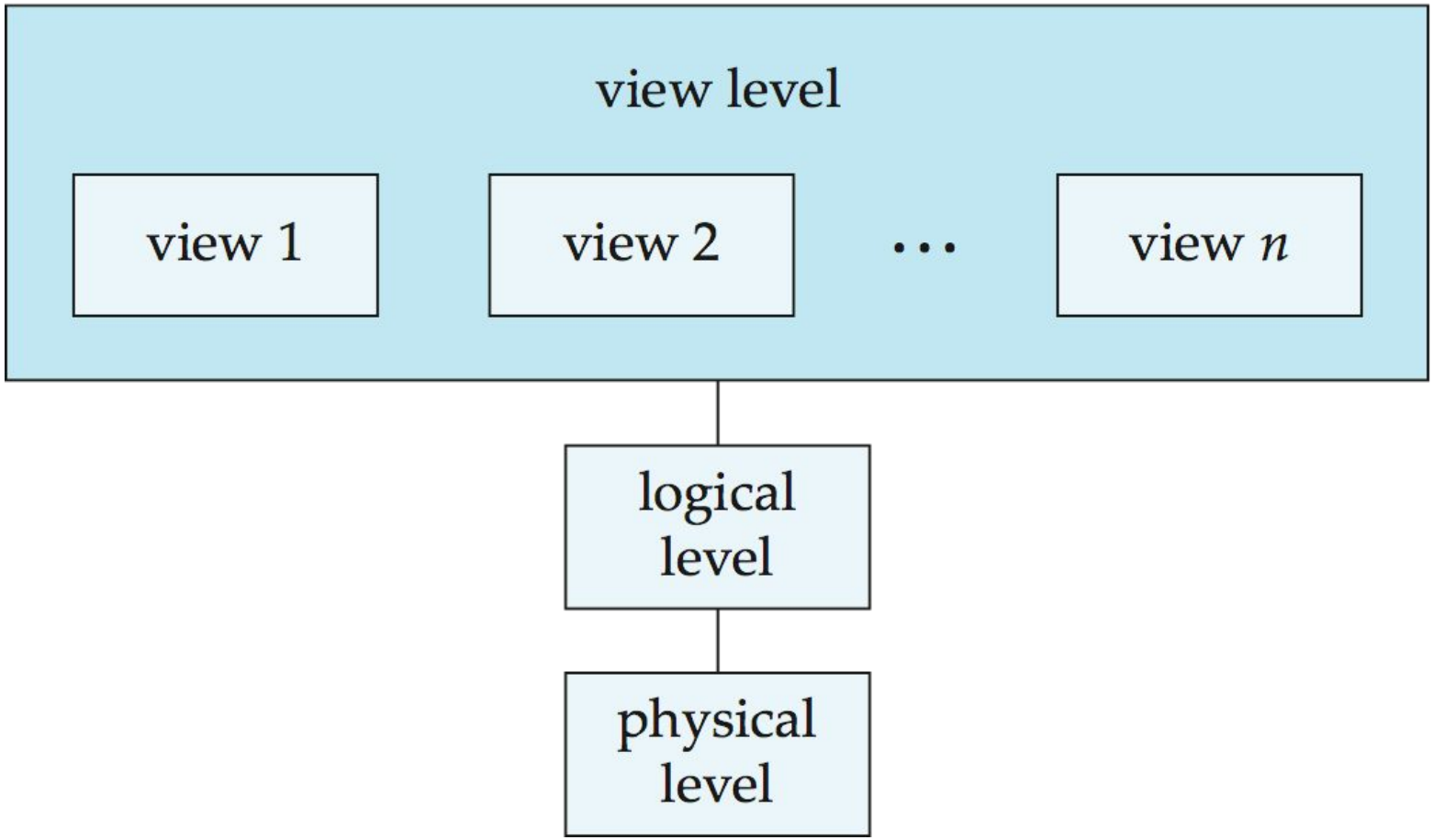
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

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

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

View of Data

An architecture for a database system



Instances and Schemas

- Similar to types and variables in programming languages
- **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 them
    - 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
  - 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
- Relational model
- 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
  - Hierarchical model

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model

Columns

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

Rows

(a) The *instructor* table

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

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***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

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
- SQL is NOT a Turing machine equivalent language
- 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

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

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 (Chapter 7)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Chapter 8)
    - Formalize what designs are bad, and test for them

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.

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 Engine

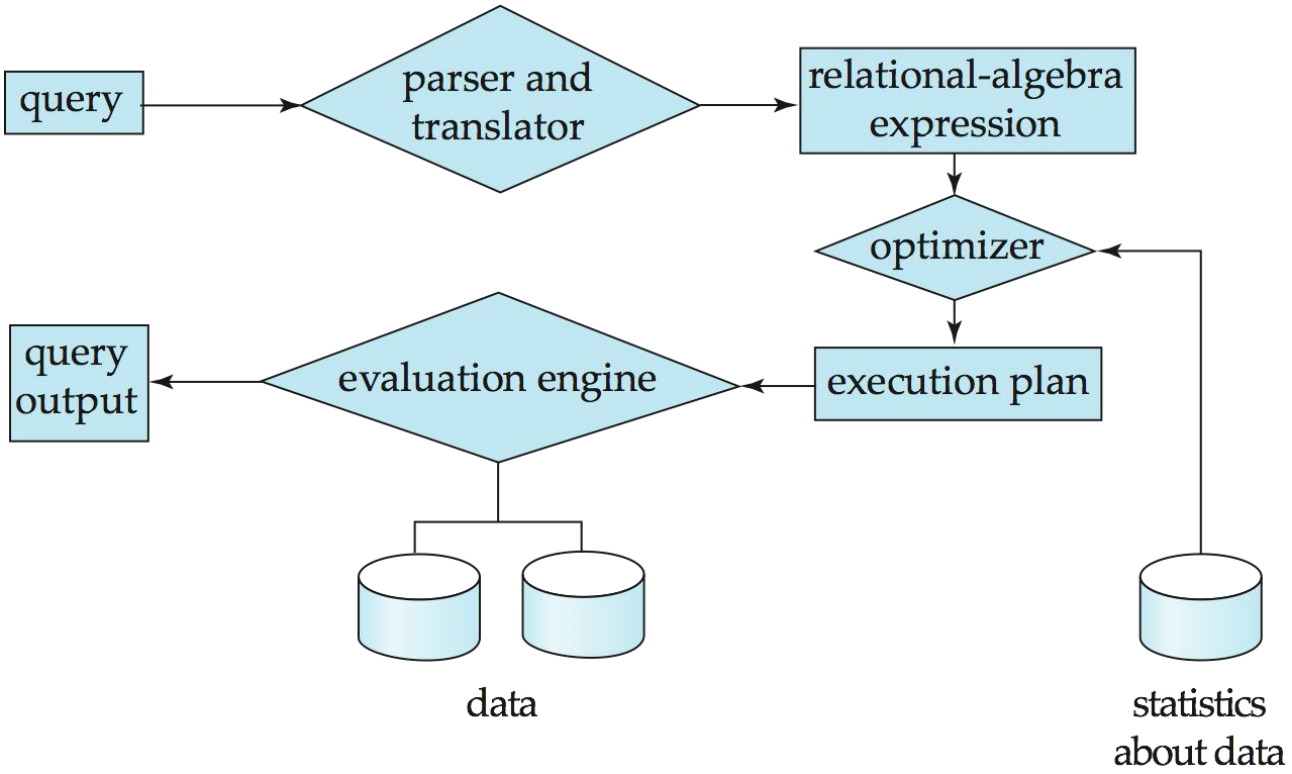
- Storage manager
- Query processing
- Transaction manager

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



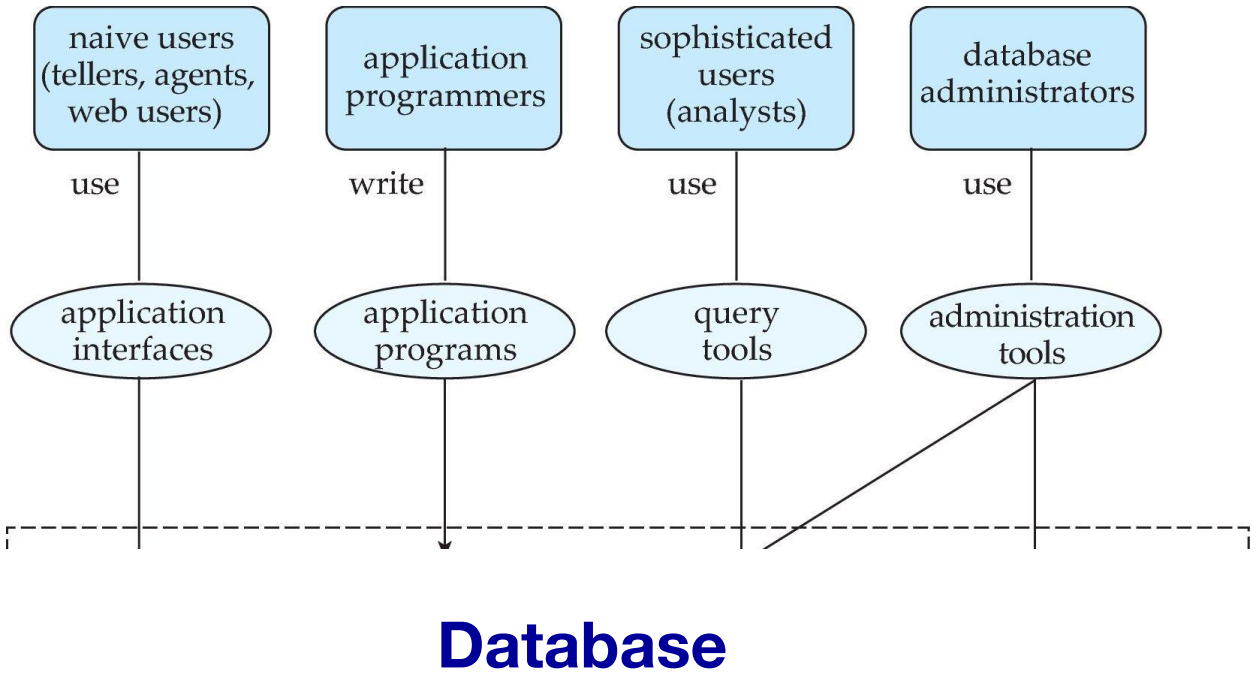
Query Processing (Cont.)

- 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

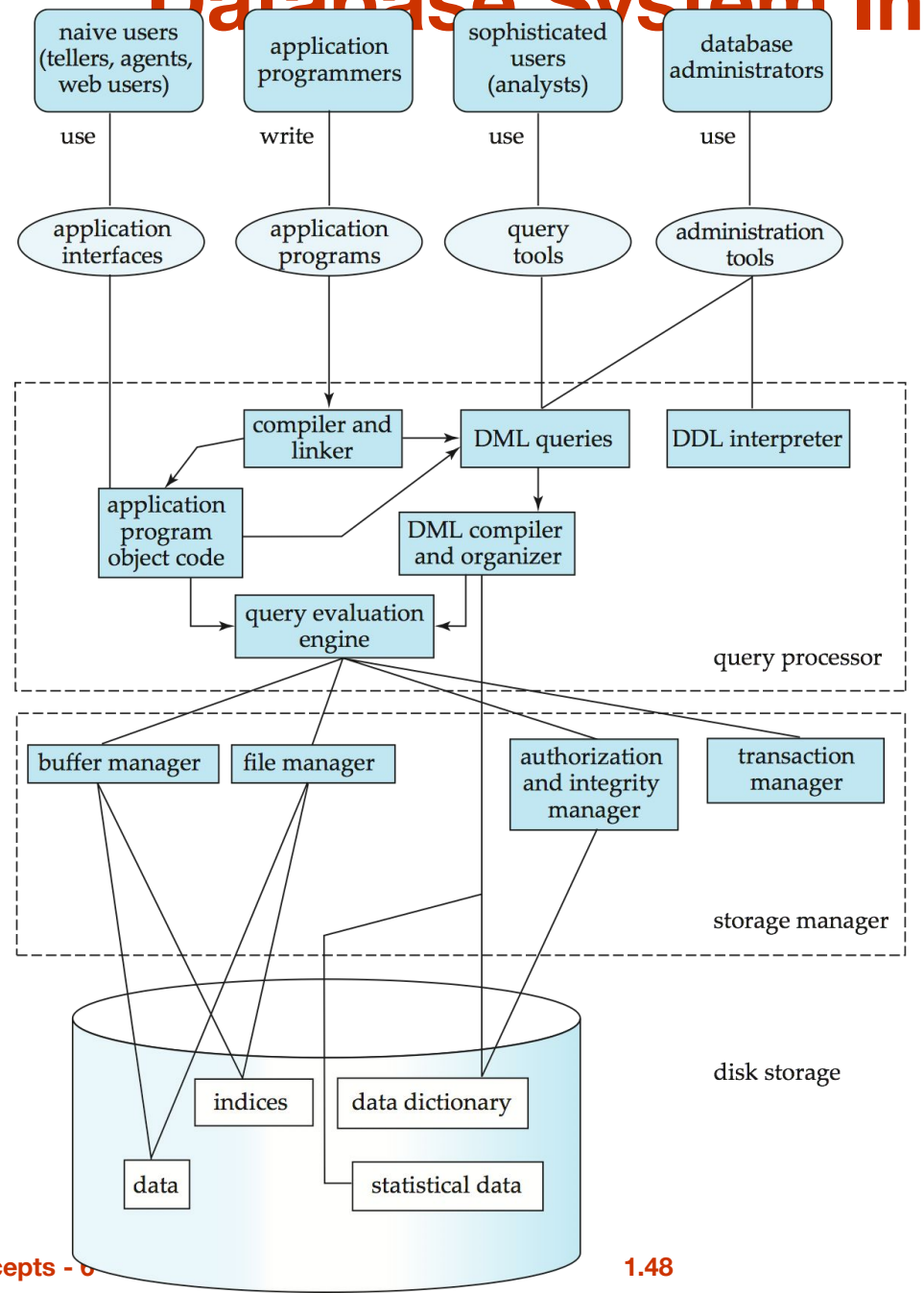
Transaction Management

- 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, ..

End of Chapter 1

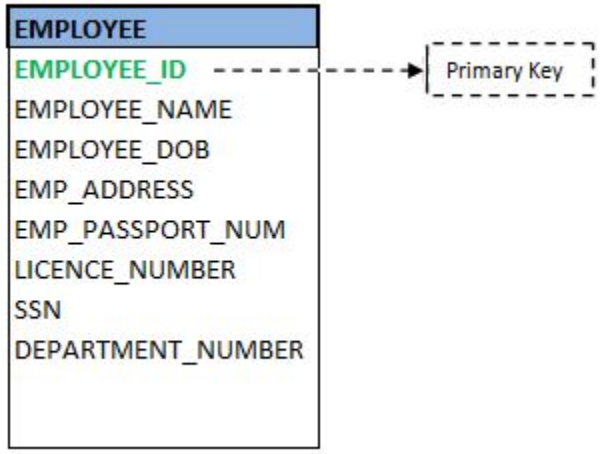
- Any attribute in the table which uniquely identifies each record in the table is called key. It can be a single attribute or a combination of attributes. For example, in STUDENT table, STUDENT_ID is a key, since it is unique for each student. In PERSON table, his passport number, driving license number, phone number, SSN, email address is keys since they are unique for each person.

STUDENT
STUDENT_ID
STUDENT_NAME
ADDRESS
DOB
COURSE

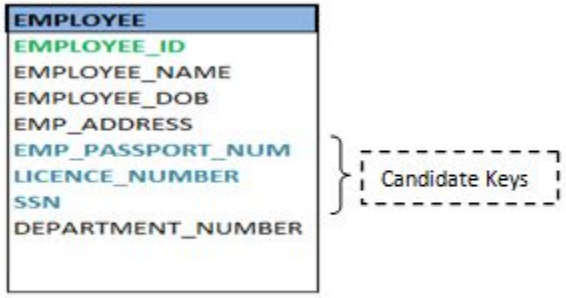
PERSON
EMPLOYEE_NAME
EMPLOYEE_DOB
EMP_ADDRESS
EMP_PASSPORT_NUM
LICENCE_NUMBER
SSN

- Primary Key

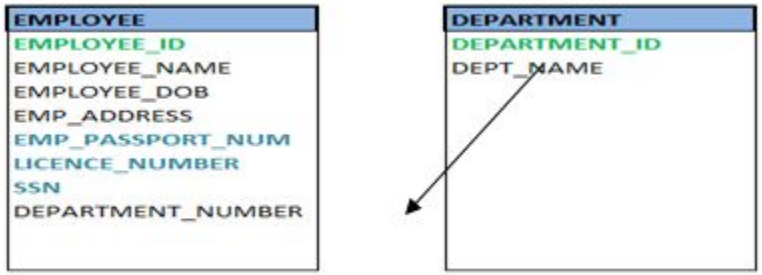
It is the first and foremost key which is used to uniquely identify a record. It can be a single attribute or a combination of attributes. For an entity, there could be multiple keys as we saw in PERSON table. Most suitable key from those lists becomes a primary key. In the Person table above, we can select SSN as primary key, since it is unique for each person. We can even select Passport Number or license number as primary key as they are also unique for a person. However, selection of primary key for each entity is based on requirement and d



- Candidate Key
- As we discussed above, an employee is identified by his ID in his office. Apart from his ID, does he have any other unique keys, so that he can be identified from others? Yes, he has passport number, PAN number, SSN number (if applicable), driving license number, email address etc. These are also identifies specific person uniquely. But we can choose any one of these unique attribute as primary key in the table. Rest of the attributes, which holds as strong as primary key are considered as Candidate key/secondary key. In our example of employee table, EMPLOYEE_ID is best suited for primary key as its from his own employer. Rest of the attributes like passport number, SSN, license Number etc are considered as candidate key.



- Foreign key
- In a company there would be different departments - Accounting, Human Resource (HR), development, Quality, etc. An employee, who works for that company, works in specific department. But we know that employee and department are two different entities. So we cannot store his department information in employee table. Instead what we do is we link these two tables by means of primary key of one of the table i.e.; in this case, we pick the primary key of department table - DEPARTMENT_ID and add it as a new attribute/column in the Employee table. Now DEPARTMENT_ID is a foreign key for Employee table, and both the tables are related!



- Constraints are the conditions forced on the columns of the table to meet the data integrity. We have seen above what types of data integrities exists in the database. Now let see what constraints can be applied on tables so that data integrity is met.
- NOT NULL
- This constraint forces the column to have non-null value. We cannot enter/update any NULL value into such columns. It must have valid value all the time. For example, each student in STUDENT table should have class specified. No student can exist without class. Hence class column in the STUDENT table can be made NOT NULL.
- CREATE TABLE STUDENT (STUDENT_ID NUMBER (10) **NOT NULL**
  STUDENT_NAME VARCHAR2 (50) NOT NULL,
  AGE NUMBER);

- UNIQUE