Comparison

Parameter	File Handling via Python	DBMS
Scalability with respect to amount of data	Very difficult to handle insert, update and querying of records	In-built features to provide high scalability for a large number of records
Scalability with respect to changes in structure	Extremely difficult to change the structure of records as in the case of adding or removing attributes	Adding or removing attributes can be done seamlessly using simple SQL queries
Time of execution	In seconds	In milliseconds
Persistence	Data processed using temporary data structures have to be manually updated to the file	Data persistence is ensured via automatic, system induced mechanisms
Robustness	Ensuring robustness of data has to be done manually	Backup, recovery and restore need minimum manual intervention
Security	Difficult to implement in Python (Security at OS level)	User-specific access at database level
Programmer's productivity	Most file access operations involve extensive coding to ensure persistence, robustness and security of data	Standard and simple built-in queries reduce the effort involved in coding thereby increasing a programmer's throughput
Arithmetic operations	Easy to do arithmetic computations	Limited set of arithmetic operations are available
Costs	Low costs for hardware, software and human resources	High costs for hardware, software and human resources

Levels of Abstraction

- **Physical level:** describes how a record (for example, instructor) is stored
- **Logical level:** describes data stored in database, and the relationships among the data fields

  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

only 1 physical and logical level, multiple view levels possible

schema is the way data will be organized, instance is the actual value of that

- Similar to type of a variable and value of the variable at run-time in programming languages
- **Schema**
  - **Logical Schema** – the overall logical structure of the database
    - Analogous to type information of a variable in a program
    - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Customer Schema

Name	Customer ID	Account #	Aadhaar ID	Mobile #

    - Account Schema

Account #	Account Type	Interest Rate	Min. Bal.	Balance

  - **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
  - Customer Instance

Name	Customer ID	Account #	Aadhaar ID	Mobile #
Pavan Laha	6728	917322	182719289372	9830100291
Lata Kala	8912	827183	918291204829	7189203928
Nand Prabhu	6617	372912	127837291021	8892021892

  - Account Instance

Account #	Account Type	Interest Rate	Min. Bal.	Balance
917322	Savings	4.0%	5000	7812
372912	Current	0.0%	0	291820
827183	Term Deposit	6.75%	10000	100000

instance may be added/removed and the schema remains the same, vice versa is NOT true

Schema and Instances

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Analogous to independence of *Interface* and *Implementation* in Object-Oriented Systems
  - 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.

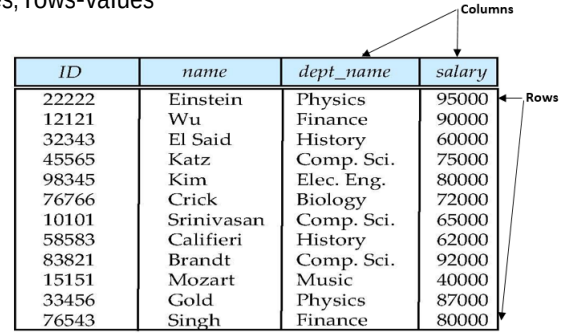
Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- **Relational model** (we focus in this course)
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Other older models
  - Network model
  - Hierarchical model
- Recent models for Semi-structured or Unstructured data
  - Converted to easily manageable formats
  - Content Addressable Storage (CAS) with metadata descriptors
  - XML format.
  - RDBMS which supports BLOBs

Relational Model

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

columns-attributes, rows-values



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

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 (that is, 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** (we focus in this course)
    - 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*
  - Cannot be used to solve all problems that a C program, for example, can solve
- 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 Programming Interface or API (for example, ODBC/JDBC) which allow SQL queries to be sent to a database

all programs you write in C you cannot write in SQL, HOWEVER vice versa is TRUE!

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 schema
  - 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

Design Approaches

- Need to come up with a methodology to ensure that each 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

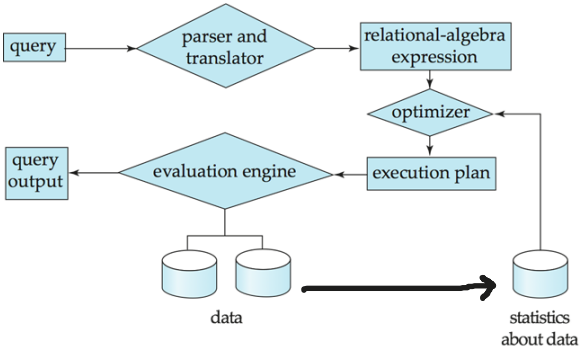
- Storage manager
- Query processing
- Transaction manager

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

a) Parsing and translation
b) Optimization
c) Evaluation



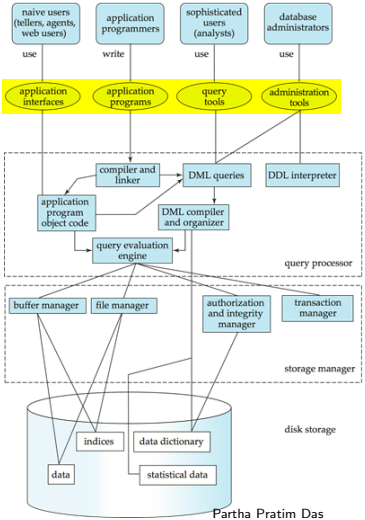
Query Processing (2)

- 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 System Internals



Database Architecture

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

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