

Week 1: Course Overview

L1.1: Course Overview

- Refer to [this](#) video
-

L1.2: *Why DBMS? - Part 1*

Data Management

Physical

- Physical data formally known as **Book Keeping** was used.
- The most significant development happened when **Henry Brown** patented a "*receptacle for storing and preserving papers*" on Nov 2, 1886.
- **Herman Hollerith** adopted the punch cards used for weaving looms to act as the memory for a mechanical tabulating machine in 1890.

Electronic

Electronic Data or Records management moves with the advances in technology - especially of *memory*, *storage*, *computing* and *networking*.

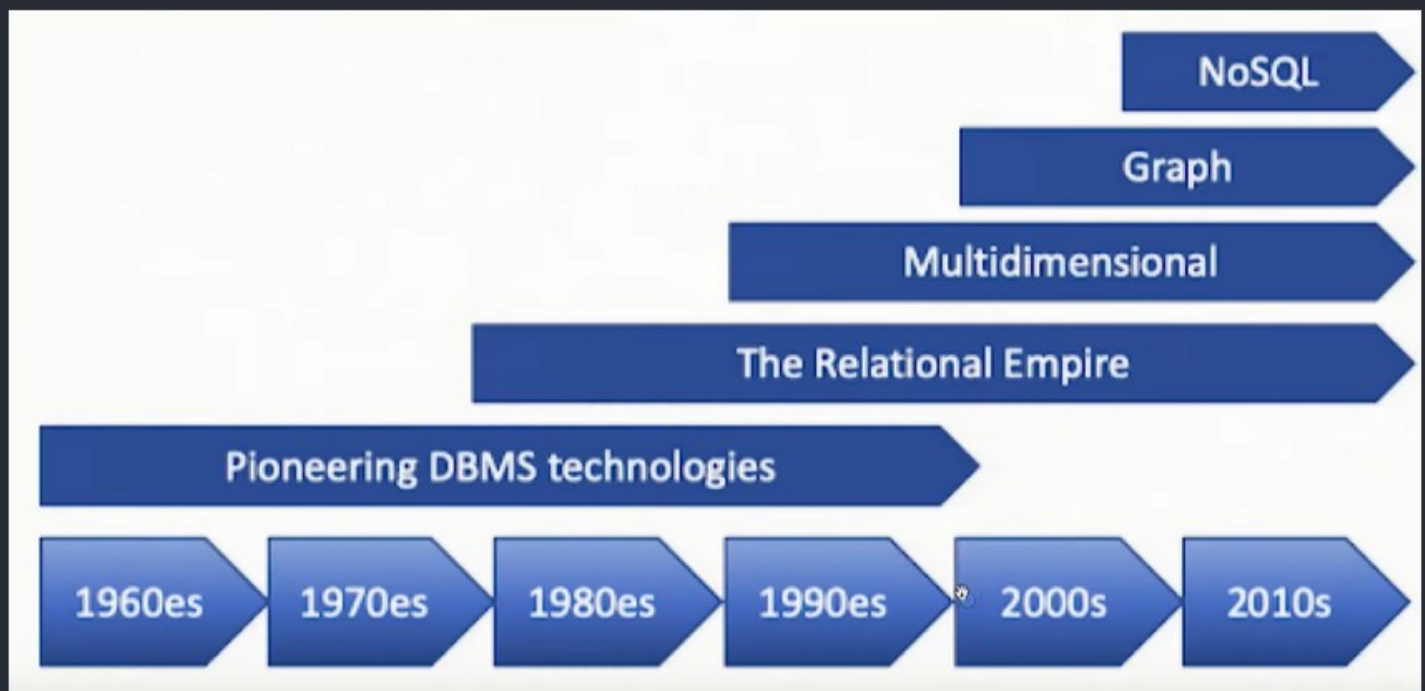
- 1950s: Computer Programming started
- 1960s: Data management with punch card / tapes and magnetic tapes.
- 1970s
 - **COBOL** and **CODASYL** approach was introduced in 1971.
 - On October 14 in 1979, Apple II platform shipped VisiCalc, marking the birth of the spreadsheet.
 - Magnetic disks became prevalent.
- 1980s: **RDBMS** changed the face of data management.
- 1990s: With Internet, data management started becoming global.
- 2000s: e-Commerce boomed, NoSQL was introduced for unstructured data management.
- 2010s: Data Science started riding high

Electronic Data Management Parameters:

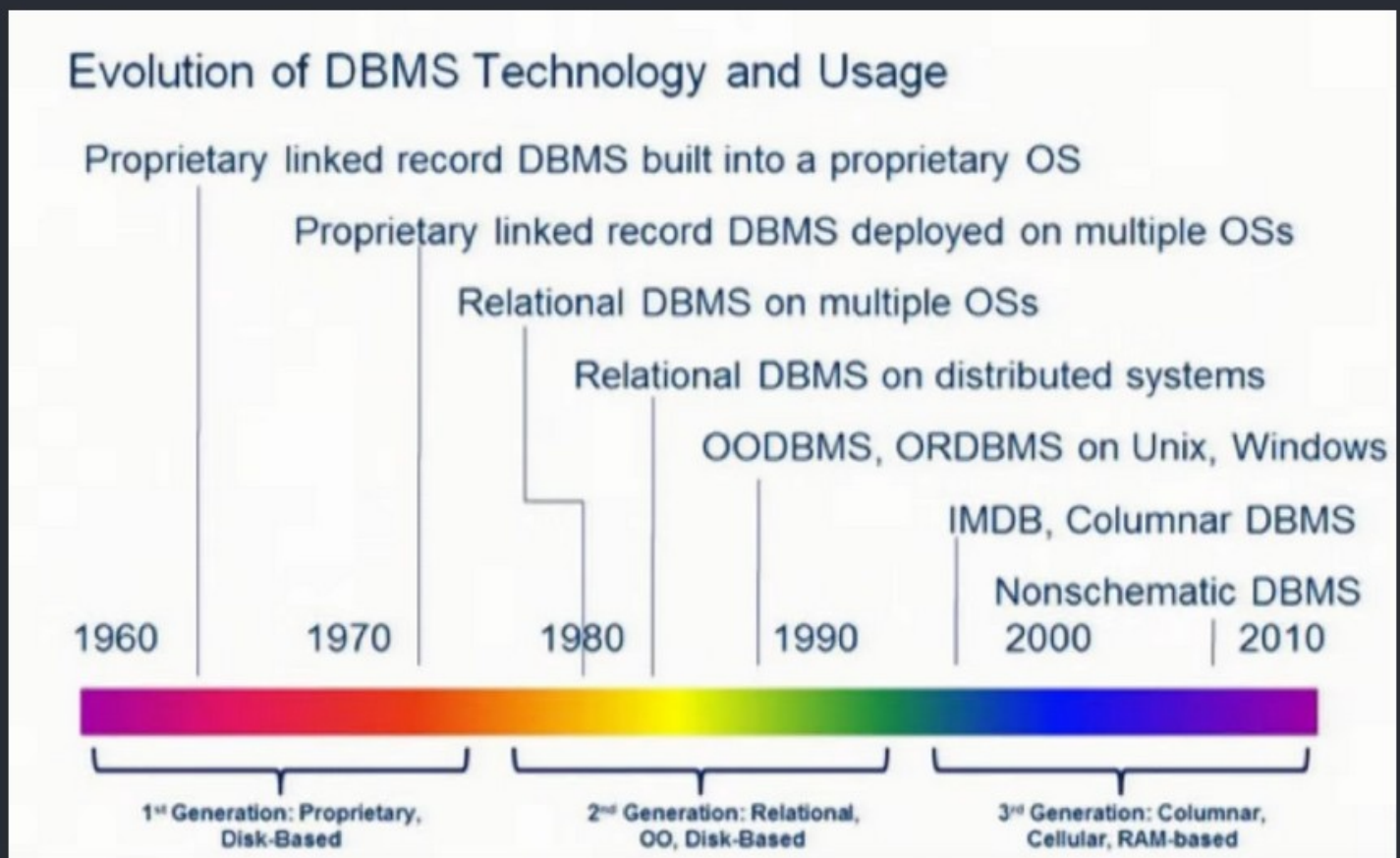
- Durability
- Scalability
- Security
- Retrieval
- Ease of Use

- Consistency
- Efficiency
- Cost

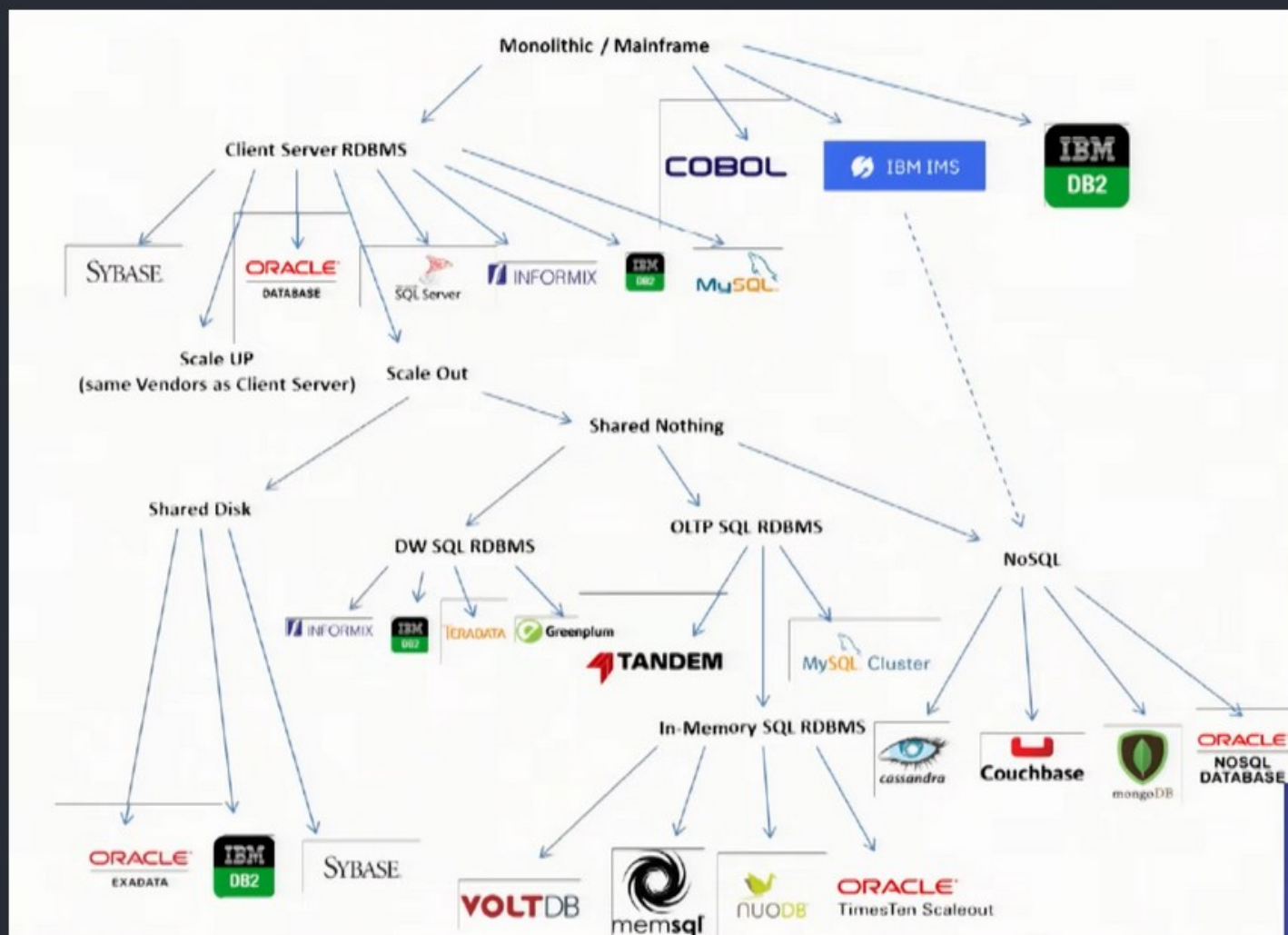
Evolution of Data Models



Evolution of DB Technology



Evolution of DB Architecture



L1.3: Why DBMS? - Part 2

- Refer to [this](#) video

Parameter	File I/O in Python	DBMS
Scalability (amount of data)	Good for small to medium-sized datasets. Can become inefficient for large datasets.	Excellent for large datasets.
Scalability (changes in structure)	Difficult to change the structure of a file once it has been created.	Easy to change the structure of a database.
Time of execution	Can be slow for large datasets.	Typically faster than file I/O for large datasets.
Persistence	Data processed using temporary data structures have to 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 productivity.
Arithmetic operations	Easy to do arithmetic operations.	Limited set of arithmetic operations are available.
Costs	Low costs for hardware, software and human resources.	High costs for hardware, software and human resources.

L1.4: Introduction to DBMS - Part 1

Levels of Abstraction

Physical Level

Describes how a record is stored.

- For example: Instructor.

Logical Level

Describes data stored in database, and the relationships among the data fields.

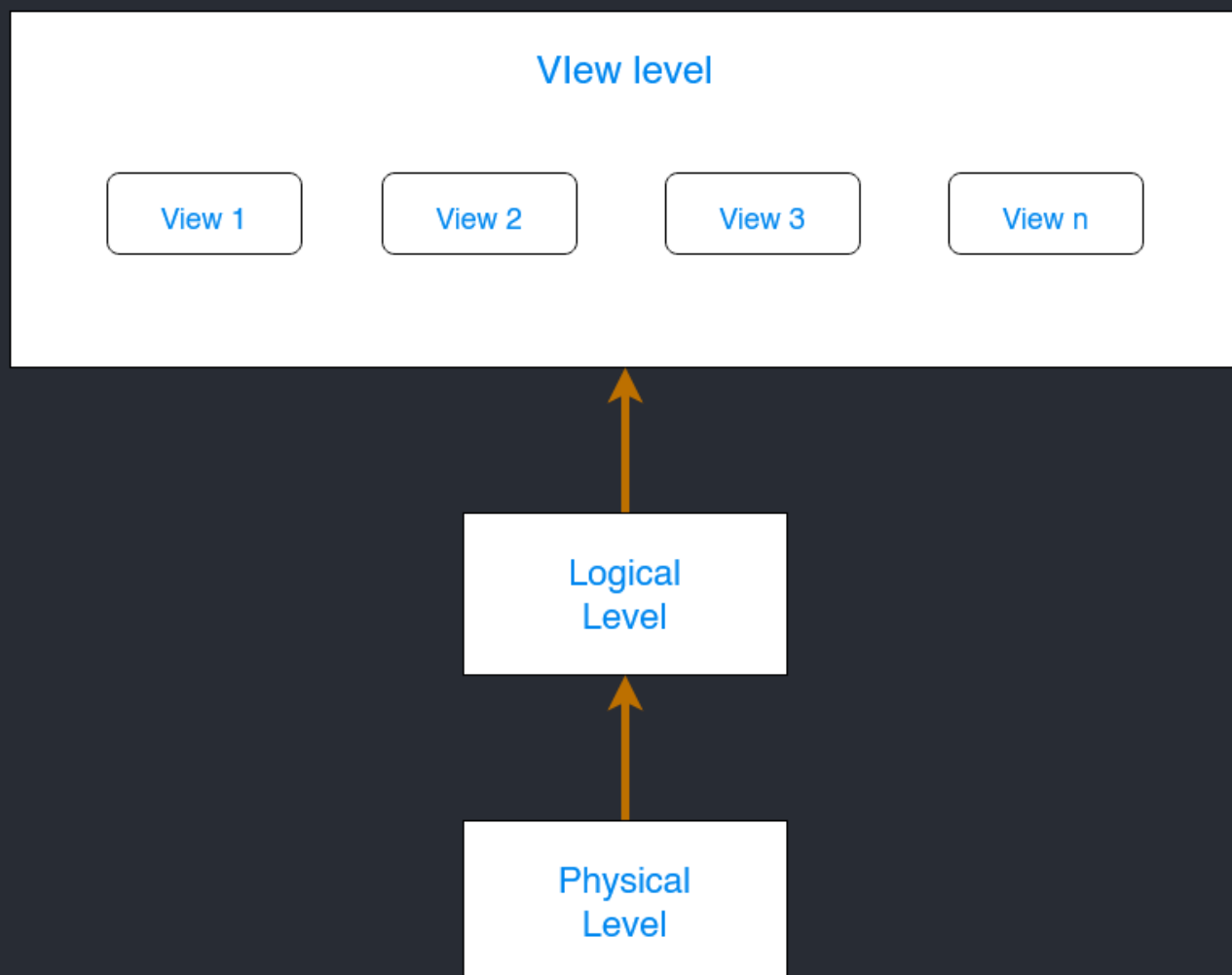
```
type instructor = record
  ID: string;
  name: string;
  dept_name: string;
  salary: integer;
end;
```

View Level

Application programs hide details of data types.

- Views can also hide information for security purposes.
- For example: employee's salary.

View of Data



Schemas and Instances

Similar to type of a variable and value of the variable at run-time in programming languages.

Schema

Logical Schema

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 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

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 #
Pawan Laha	6728	9172325	123467967919	9823854721
Ankit Kumar	6729	9172326	123467967920	7865872349
Nandini	6730	9172327	122357367921	8923678692

- **Account Instance**

Account #	Account Type	Interest Rate	Min. Bal.	Balance
9172325	Savings	4.5	1000	10000
9172326	Current	3.5	5000	20000
8271683	Term Deposit	6.75	1000	100000

Physical Data Independence

The ability to modify the physical schema without changing the logical schema.

- Analogous to independence of Interface and Implementation in OOP.
- 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.

Physical Data Independence refers to the modification of the physical level without affecting the logical and view level.

Logical Data Independence refers to the modification of the logical level without affecting the view level.

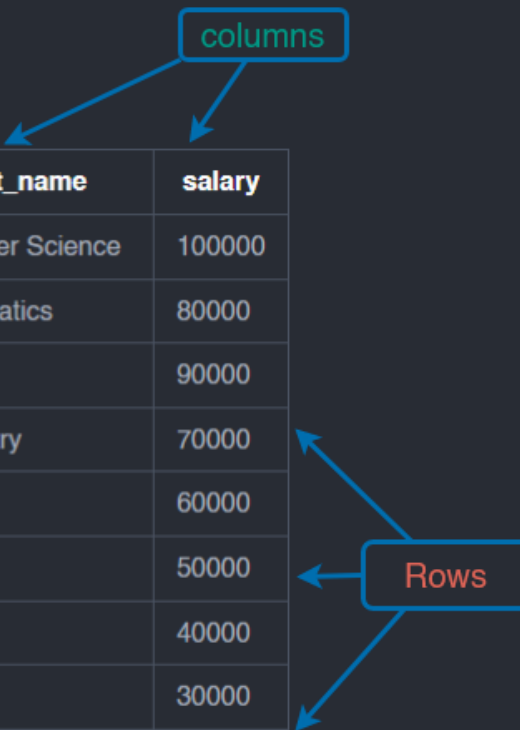
Data Models

- A collection of tools for describing:
 - Data
 - Data relationships
 - Data semantics
 - Data constraints
- **Relational model**
- **Entity-Relationship 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 managable formats
 - Content Addressable Storage (CAS) with metadata descriptors
 - XML format
 - RDBMS which supports BLOBs

Relational Model

Instructor Table

ID	name	dept_name	salary
1	John Doe	Computer Science	100000
2	Jane Doe	Mathematics	80000
3	Peter Smith	Physics	90000
4	Susan Jones	Chemistry	70000
5	David Brown	Biology	60000
6	Emily Green	English	50000
7	Michael White	History	40000
8	Sarah Black	Art	30000
9	Kevin Blue	Music	20000
10	Ashley Pink	Foreign Languages	10000



ID	name	dept_name	salary
1	John Doe	Computer Science	100000
2	Jane Doe	Mathematics	80000
3	Peter Smith	Physics	90000
4	Susan Jones	Chemistry	70000
5	David Brown	Biology	60000
6	Emily Green	English	50000
7	Michael White	History	40000
8	Sarah Black	Art	30000
9	Kevin Blue	Music	20000
10	Ashley Pink	Foreign Languages	10000

Department Table

dept_name	building	budget
Computer Science	101	1000000
Mathematics	102	800000
Physics	103	900000
Chemistry	104	700000
Biology	105	600000
English	106	500000
History	107	400000

DDL and DML

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
 - 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
- 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

- *SQL is NOT a Turing Machine equivalent language*
 - Cannot be used to solve all problems that a C program 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
 - API (example: ODBC/JDBC) which allow SQL queries to be sent to a database server.

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 we have and how should the attributes be distributed among the relation schemas?

Physical Design

Deciding on the physical layout of the database.

L1.5: Introduction to DBMS - Part 2

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 we have and how should the attributes be distributed among the relation schemas?

Physical Design

Deciding on the physical layout of the database.

Is this design good?

ID	name	dept_name	salary	building	budget
1	John Doe	Computer Science	100000	101	1000000
2	Jane Doe	Mathematics	80000	102	800000
3	Peter Smith	Physics	90000	103	900000
4	Susan Jones	Chemistry	70000	104	700000
5	David Brown	Biology	60000	105	600000
6	Emily Green	English	50000	106	500000
7	Michael White	History	40000	107	400000
8	Sarah Black	Computer Science	30000	101	1000000
9	Kevin Blue	Computer Science	20000	101	1000000
10	Ashley Pink	Mathematics	10000	102	800000

NO

- As there are duplicate rows present in the table, which is causing **data redundancy**.

A good design would be to separate the data into two tables, one for **instructors** and one for **departments**.

ID	name	dept_name	salary
1	John Doe	Computer Science	100000
2	Jane Doe	Mathematics	80000
3	Peter Smith	Physics	90000
4	Susan Jones	Chemistry	70000
5	David Brown	Biology	60000
6	Emily Green	English	50000
7	Michael White	History	40000
8	Sarah Black	Art	30000
9	Kevin Blue	Music	20000
10	Ashley Pink	Foreign Languages	10000

Department Table

dept_name	building	budget
Computer Science	101	1000000
Mathematics	102	800000
Physics	103	900000
Chemistry	104	700000
Biology	105	600000
English	106	500000
History	107	400000

Design Approaches

Need to come up with a methodology to ensure that each relation 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.
 - Widely used during the planning and designing phase of a database system
 - **Normalization**
 - Formalize what designs are bad, and test for them.

Object-Relational Data Models

- Relational model: flat, atomic values
- Extend the relational dat model by including object orientation and constructs to deal with added data types.
- Allow attributes of tuples to have complex types, including:
 - Structured types
 - Collection types
 - Inheritance
 - User-defined types
- Preserve relational foundations in particular the declarative access to data, while extending modeling power.
- Provide upwards 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 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 for:
 - 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

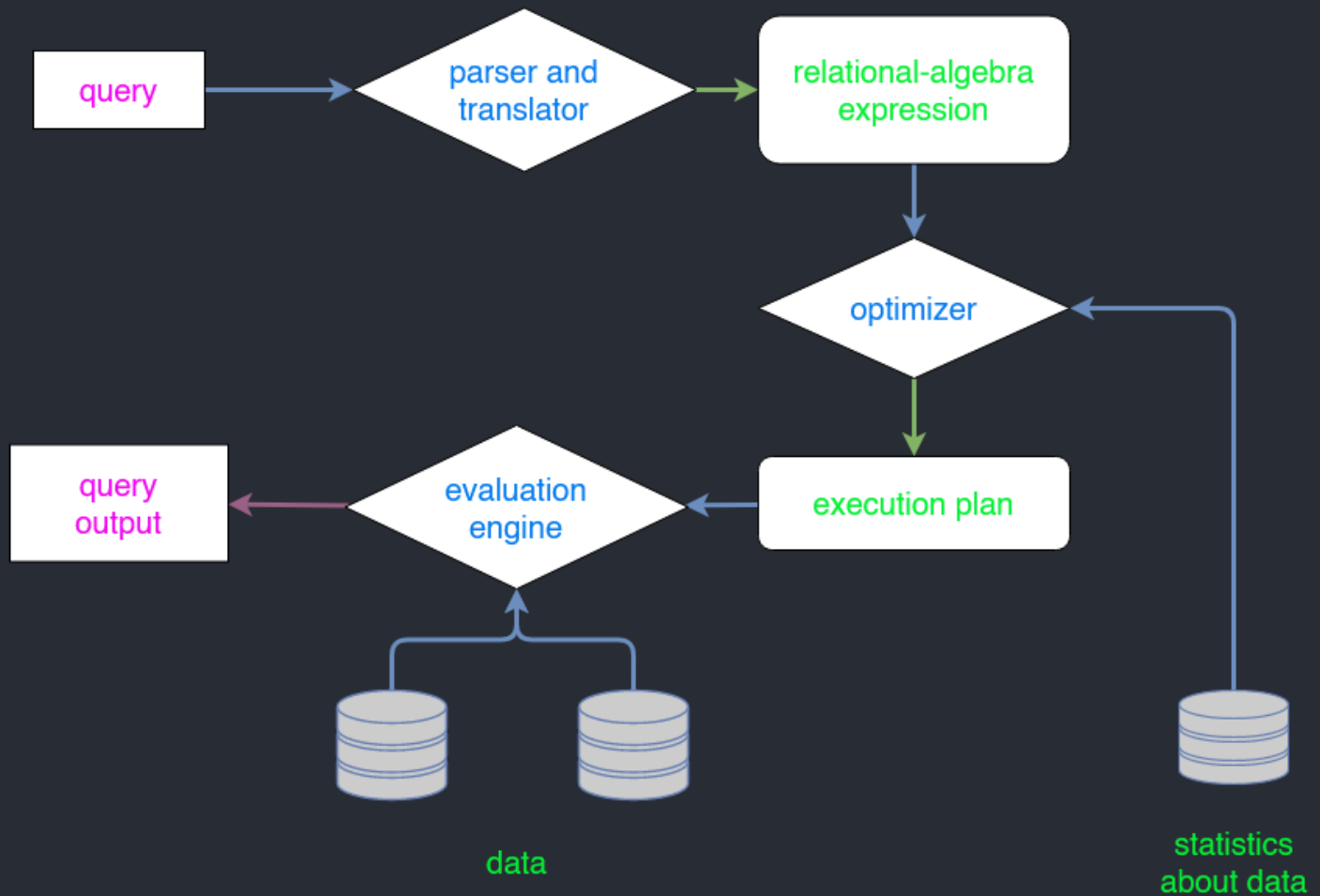
Analyzing the query's syntax and structure, and translating it into an internal representation.

2. Optimization

Finding the most efficient execution plan by estimating costs, generating alternative plans, and selecting the one with the lowest cost.

3. Evaluation

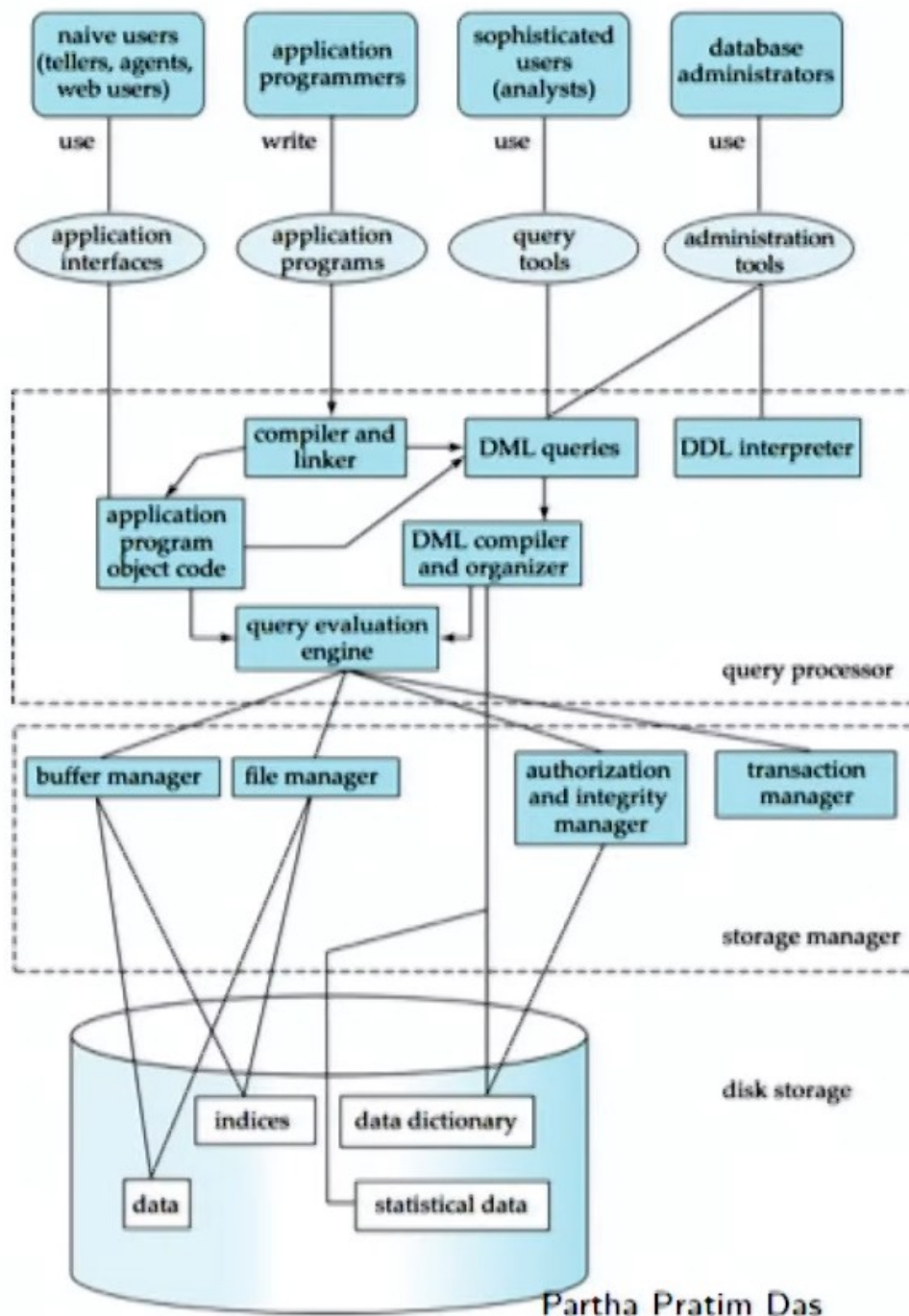
Retrieving data using selected access methods, joining tables, applying filters/aggregations, generating the final result, and delivering it to the user.



Transaction Management

- 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 state despite system failures 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