Chapter 1.1

Introduction to DBMS

Where do you Find DB’s in Real-life ?

- E-Commerce Platforms
- Banking and Finance
- Healthcare
- Transportation (Railway)
- National Id Systems
- Education

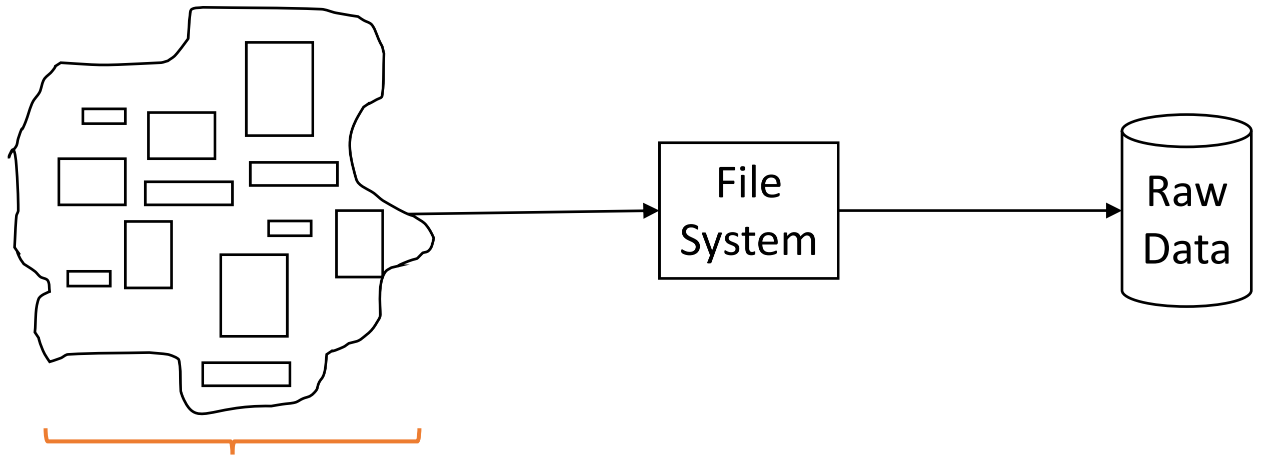
Any modern day computer application,
Be it a website, app or software, will have a
DB as one of its core components.

Life Just Before The Advent Of DBMS

Early Information Systems / Legacy Systems

Early Information System

Say **Bank**:

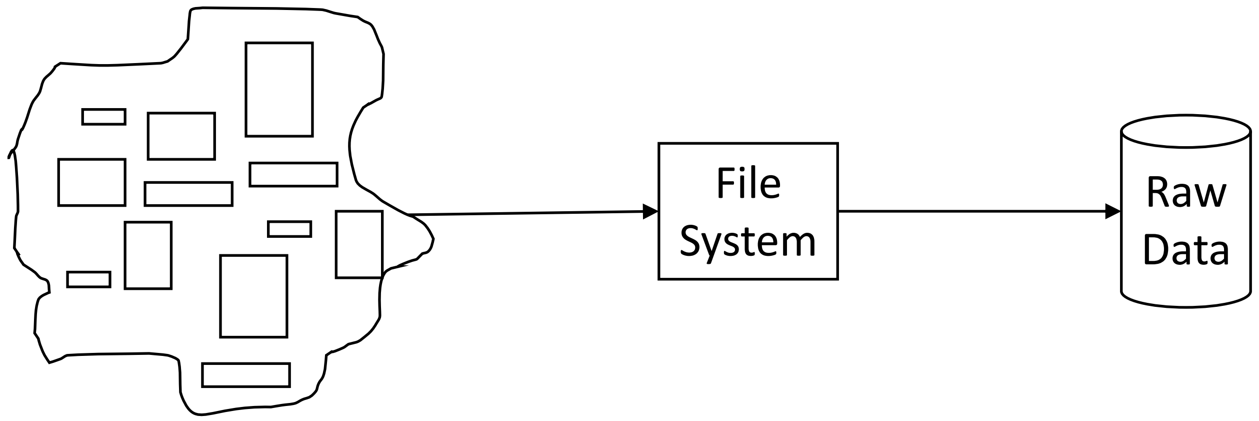


Different Appcn Programs: Say Debit, Loan, NEFT, FD

Programs directly access the file system

Note: Every operations like Loan, NEFT, is actually a separate program, operating upon the same data

Early Information System



What is the problem with this approach ?

The data in this case will be used, updated & potentially overwritten by various users / programs at various point of time.

Nobody really knows what the other person / code has done.

$\downarrow$

To bypass this, people started to replicate data.
Have your own copy so that your program is safe.

1. Unnecessary Redundancy (Space wastage ?)
2. Inconsistency

Redundancy & Inconsistency

Data is replicated to avoid Collison
What problems arise due to replication ?

Say address is in 5 different places (due to copies made by 5 programs).

If 1 program updates the address, then every other copy becomes inconsistent. $\longrightarrow$ Update needs to be propagated

Imagine you order something in Amazon, and then update your address, but order goes to old address.

$$\text{Redundancy} \xrightarrow{\text{Causes}} \text{Inconsistency}$$

Next Issue with EIS

Querying

Difficulty in Querying (Life before SQL)

What is Querying ? → It is the process of requesting specific information from the Database.

Say, you choose to maintain your DB as a flat file system [Like Excel / CSV]

Movie Database:
Kutramae Thandanai, 2016, Tamil, Manikandan. Split, 2016, English, M.Shyamalan.

Say you want to find out the director of Split movie.

```
For line in File:
        record = PARSE(Line)
If "Split" == record[0]
        print (string (record[3]))
```

If it is DBMS based,
SELECT Director FROM imdb WHERE title = "Split"

For each query you need to write your own program.

Typically the language used to create the DB itself is used to manipulate & retrieve info from it.
You have to anticipate/guess what type of queries may arise for this table before hand.

Other issues in EIS

Consider applications that are multi-threaded. What if two threads update same file ? (**Concurrency**)

The developer has to resolve these collisions by himself

Say your Hardisk failed. How do you recover ?. **(Availability & Recovery)**
How do you make your system available in case of failure ?

The data needs to be replicated for the sake of availability
(Note: This replication is different)

What if a new application wants to use the same DB ?

Replicate the entire DB. Unnecessary complications

Other issues in EIS

SECURITY:
Should the complete Database be visible to all the programs and users ?

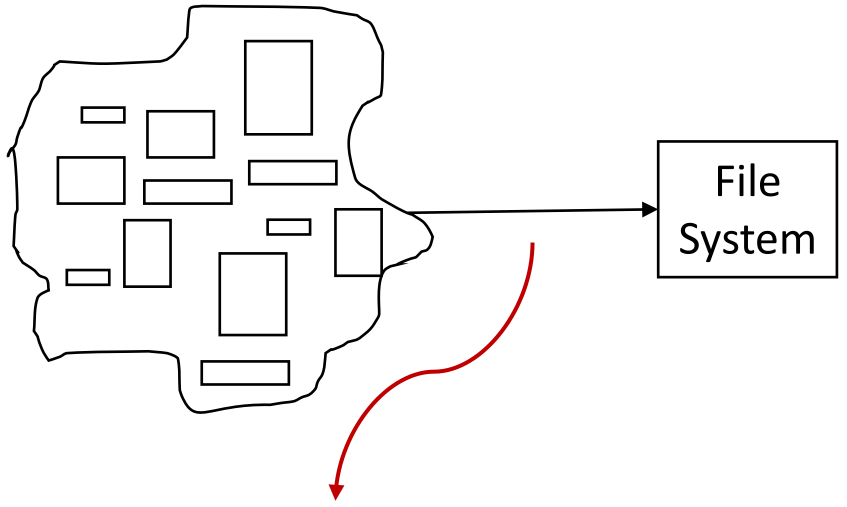
Programs that do not use certain part of the DB should have no access to it
VIEWS in DBMS helps to achieve this

INTEGRITY:
Accurate, consistent, complete data at any point in time. How do you achieve it ?

Example:
Invalid year
Wrongly formatted phone numbers.
Negative number in age
Different address for a same person

To enforce these constraints,
we need to specify it for every single app program.
Whereas, in DBMS you can specify all these constraints at one single place.

How DBMS Solves all these issues:



DBMS is a layer introduced at this level which tries to:

- Organize the data
- Gives a comprehensive view of the entire DB
- You place your request to this DBMS layer, the actual execution is upto it

What does the layer of DBMS Provide ?

Separation of Data & Metadata:

Meta data sits in-between the DBMS S/W and the actual Data, provides flexibility.
Helps to achieve Program-Data Independence.
The notion of "**Metadata**" is introduced by DBMS only.

$\downarrow$

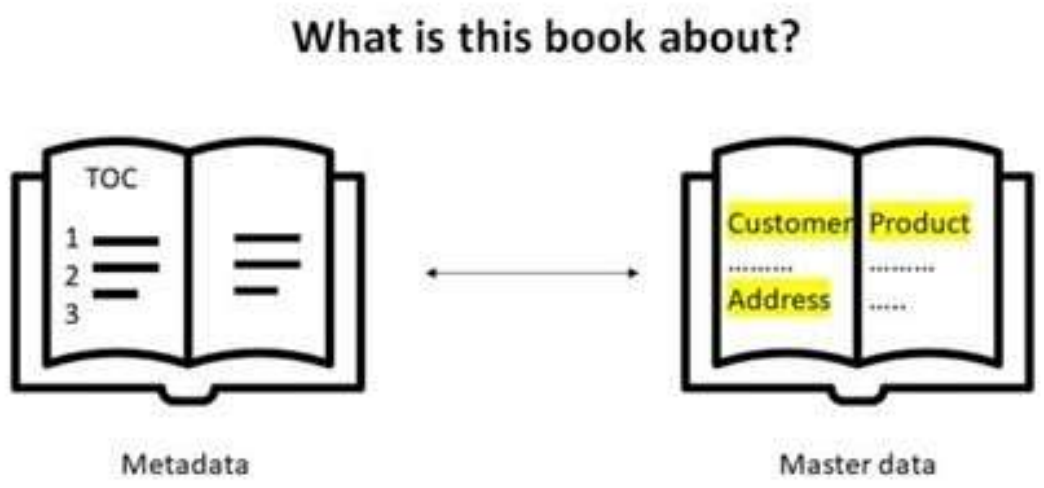
It describes how (i.e., Its structure & organization) the raw data is stored.

Remember:

$$\begin{tikzcd} \text{Structure of the Data} \arrow[r, leftrightarrow, "\text{Hardcoded}", "\text{into}"'] & \text{App Programs} \end{tikzcd}$$

Metadata helps to remove this dependency by storing the structure information separately.

Metadata – Data about the Data



Other Names:

- Catalogue
- Data Dictionary
- Schema

What does the layer of DBMS Provide ?

Programmatic Convenience:

- Less effort for system development.

- Concentrate only on logical design [No physical level implementation worries].

- Concurrency control, recovery, access/authoraziation control, security integrity check.

And many more things will be taken by the DBMS S/W. You need not worry about it.

What Makes DBMS Generic ?

We know that in DBMS Systems, Data is organized and structured in a **standard** way.

i.e. $\boxed{\text{Data \& Schema}}$ are stored separately

Schema $\longrightarrow$ Description

This defines the structure in which data
Is to be stored.

It sets up the skeleton structure of the DB

int a;

Instance

It is the actual data stored in a format as
prescribed by the schema

a = 10;

Schema

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Figure 2.1
Schema diagram for the database in Figure 1.2.

Almost static, rarely changes

Instance

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Changes quite often, Reflects current situation

Chapter 1.2

Basic Terminologies

Get Comfortable with DBMS Parlance

Data:

Known facts that can be recorded and have an $\underbrace{\text{implicit meaning}}$.

By itself, data is just raw facts or figures,
but its **implicit meaning** comes from its relationship to
the real-world entities, events, or concepts it represents

For example:

1.Raw Data:

1. "100“ - Without context, this is just a number.

2.Implicit Meaning with Context:

1. "100" as **temperature in Fahrenheit** suggests it's a hot day.
2. "100" as **marks in an exam** indicates a perfect score.

DataBase (DB)

It is an *organized* collection of Inter-Related data that model some aspects of the real world.

Real World: DBMS is typically concerned with data associated with some company/organization/enterprise or business.

It typically has real world meaning. ➔ “Operational Data”

Given the data of an Enterprise or a Company:

The company may have current needs and requirements with it. (Different needs may also arise in the future)

↓

It is necessary to organize data in some fashion, so that it is easier to serve these needs.

While data should be organized robustly, such that future needs may also be accommodated.

In Simple Terms

A DB "**reflects the current state of affairs**" of the Enterprise/Company

The organizational hierarchy within the company is often reflected in the DB

Universe of Discourse (Set Theory Terminology)

Mini-World: Some part of the real world about which data is stored in a database.
For example, student grades and transcripts at a university

The data you saw in DS
Vs
The data you are going to see in DBMS

In traditional programming, the data and the code are tightly coupled.

If you change the type of the variable, or declare a new one, the entire code needs to be recompiled.

In DBMS the application code and the data are **loosely coupled**.

DBMS sits in b/w the app code and the data, eliminating any direct dependency.

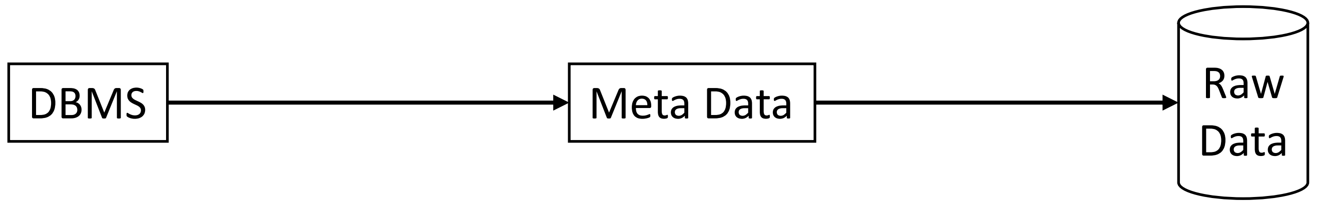
The DBMS S/W may rearrange the data underneath, while the Application may completely be unaware of it.

What is $\underbrace{\text{Loose Coupling}}_{\text{Program – Data Independence}}$ in the context of DBMS ?

Definition: The ability of a database system to separate the application programs from the physical storage and structure of the data.

Importance: Changes to the database structure do not require changes in the application programs.
Enables flexibility and scalability in database management.

What is $\underbrace{\text{Loose Coupling}}_{\text{Program – Data Independence}}$ Is achieved in DBMS ?



What is Metadata?
Metadata is data about data; it describes the structure, attributes, and constraints of the actual data in the database.

•**Examples:**
- Table names, column names, and data types.
- Relationships between tables, constraints like primary keys, and indexes.

What is a DBMS

It is a **general purpose** software package/ system/ tool to facilitate the creation and maintenance of a computerized database.

DBMS creates, organizes and continues to maintain a DB in a $\underbrace{\text{Standardized Manner.}}_{\textbf{Most commonly followed standard: SQL Standard}}$

$$\begin{tikzcd}
\text{Applications \& End User} & \boxed{\text{DBMS}} \arrow[l] \arrow[r] \arrow[d] & \text{DataBase} \\
 & \text{Developer \& Database Administrator} &
\end{tikzcd}$$

Note: A single DBMS S/W may manage multiple DB's at a time.

The Workflow of Databases

The Workflow of Databases

- ***Define*** a particular database in terms of its data types, structures, and constraints
- *Construct* or **Load** the initial database contents on a secondary storage medium

- ***Manipulating*** the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing: the database through Web applications

- ***Sharing*** *of data* by a set of concurrent users and application programs – yet, keeping all data valid and consistent.

A quick example of a DB

College Database Example

College Database Example

What are some things or entities in the college about which we would like to store information ?

- STUDENTs
- COURSEs
- SECTIONs (of COURSEs)
- DEPARTMENTs
- INSTRUCTORs

Is it enough to store info about these things alone ?

What about their interactions ?

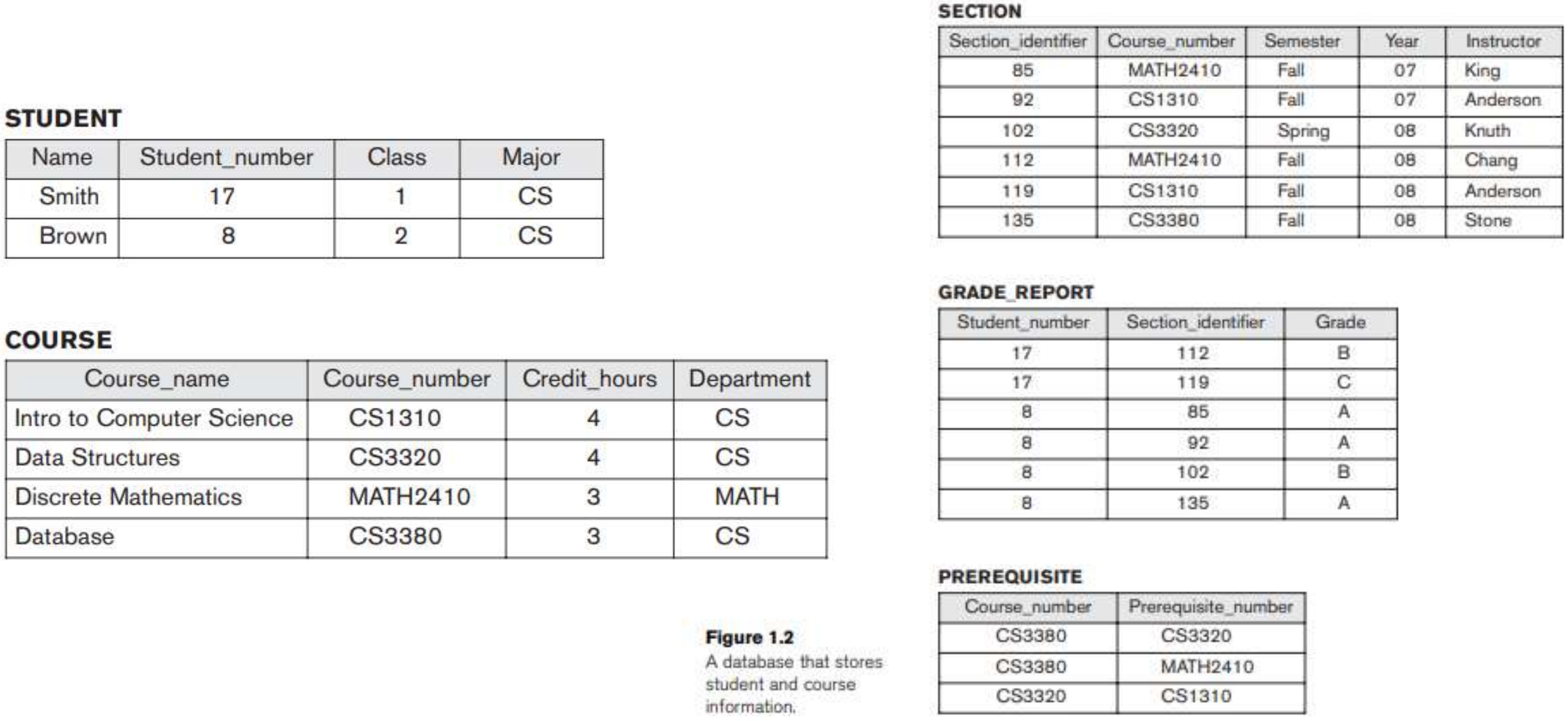
A single student may take/enroll in multiple courses, where is that information captured ?

College Database Example

Relationships between the Entities

- SECTIONs *are of specific* COURSEs
- STUDENTs *take* SECTIONs
- COURSEs *have prerequisite* COURSEs
- INSTRUCTORs *teach* SECTIONs
- COURSEs *are offered by* DEPARTMENTs
- STUDENTs *major in* DEPARTMENTs

College Database Example



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

College Database Example - Metadata

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Role of Metadata in Achieving PDI

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.
- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Role of Metadata in Achieving PDI

Abstracting Physical Details

•Metadata provides a layer of abstraction between application programs and physical data storage.
•Applications interact with the metadata rather than directly accessing raw data.

Managing Schema Evolution

•When the database schema changes (e.g., adding a new column), the metadata is updated.
•Applications continue to function without modification because they query the schema via metadata.

Simplifying Queries

•Metadata helps applications understand the structure of the database (e.g., which tables and columns to access).
•SQL queries rely on metadata to execute commands like SELECT, INSERT, etc.

Supporting Interoperability

•Different applications can interact with the same database without knowledge of its physical storage, thanks to metadata.

Users of the Database Systems

Users may be divided into

- Those who actually use and control the database content, and those who design, develop and maintain database applications (called “**Actors on the Scene**”)

- Those who design and develop the DBMS software and related tools, and the computer systems operators (called “**Workers Behind the Scene**”).

Actors on the Scene

- **Database Designers:**
  - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

- **Database administrators:**
  - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.

End Users

End-users: They use the data for queries, reports and some of them update the database content. End-users can be categorized into:

- **Casual**: access database occasionally when needed
- **Naïve** or Parametric: they make up a large section of the end-user population.

When Not To Use DBs

Overhead Costs of DBMS:

- High initial investment in hardware, software, and training
- The generality that a DBMS provides for defining and processing data
- Overhead for providing security, concurrency control, recovery, and integrity functions

Though these features are desirable, they incur significant cost to achieve.
Applications which do not demand these features, it is better not to use DBMS systems

When Not To Use DBs

Examples:

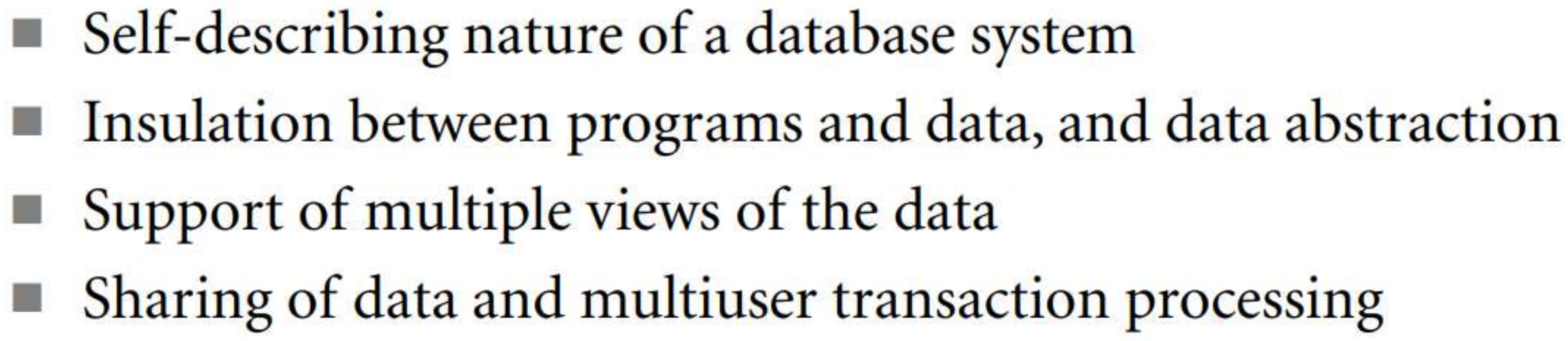
- Simple, well-defined database applications that are not expected to change at all
- Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead
- Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit

Niche applications like: Medical systems, Avionics, Stock Trading systems, Simulations

Use optimized in-memory databases or direct data access through lightweight data structures.

A Quick Recap:

4 Characteristics of Database Approach



- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

Section 1.3, Chapter 1, Navathe

THE END

How do you start designing an DBMS Application ?

$$\begin{tikzcd}\textbf{Flowchart} \arrow[r] & \textbf{Algorithm} \arrow[r] & \textbf{Code}\end{tikzcd}$$

Flowchart	Algorithm	Code
It is a visual approach.	Step-by-step textual representation of logic	Ready to execute
Easier to interpret.	Easier to read, compared to code.	Difficult to read & interpret
Can be used to explain the logic even to non-technical users.	Used among the developers to discuss logic	

$$\begin{tikzcd}\textbf{Diagram} \arrow[r] & \textbf{Step-by-step-Logic} \arrow[r] & \textbf{Actual implementation}\end{tikzcd}$$

Data Model

Data Model:

It is a conceptual representation of how an enterprise's data is stored and organized in a DB. It provides a structured way to represent data, its relationships, and the rules governing it.

DM's helps to construct a comprehensive picture of an enterprise / company's data, which in turn can be used to create the actual database.

Data Model = Data Description (Its like a blueprint to what you're going to implement later)

Data Model – Its Importance

Provides a Blueprint: Helps in database design by visualizing the structure and relationships.

Ensures Data Integrity: Enforces rules to maintain consistency and accuracy.

Facilitates Communication: Acts as a bridge between developers, DBAs, and stakeholders.

Supports Query Optimization: Helps in structuring data for efficient access and manipulation.

Data Model – Its Importance

The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts.

Hence, the data model hides storage and implementation details that are not of interest to most database users.

– Chapter 1, Navathe, Pg 12

Data Model – Types

Data Model

High Level Model
Conceptual Model

Eg:
- **ER Model**
- OOPs Model
- UML Model

Low (Kind of Mid) Level Model
Representational Model

Eg:
- **Relational Model**

No SQL
- Key-Value (Dict)
- Graph
- Document

Legacy / Obsolete
- Hierarchal
- Network

AI/ML Workload
- Array/Matrix

Flowchart → **Algorithm**

Data Model – Types

$$\begin{tikzcd}
 & \text{Data Model} \arrow[dl] \arrow[dr] & \\
\textbf{High Level Model} & & \textbf{Representational Model}
\end{tikzcd}$$

High Level Model

High Level Description

Suitable for requirement collection and understanding

Representational Model

Describes data at a logical or a Representational level

‘Represents’ only the logical part of the DB

Details of physical implementation is avoided

Appcn Programs Views / Deals with the DB with this view.

This is how you imagine the information to reside in the Hard Disk

Also Known as “Record Based Models”

Physical Data Model (Storage Model)

Describes how data is actually stored in the storage medium.

- Deals with record, files, and its formats.
- Exact data types, Length, default values are specified.
- Exact location, file size, indexing, sorting are taken care at this level.
- Data partioninig, buffer management.

The previous two models were concerned with **Design,** this one is concerned with **Implementation.**

Sanity Check

Why do we need several DM's at different level of Abstraction ?

Remember, DB's deal with real world data (I.e., **Operational Data**) associated with enterprises.

So typically there are different kinds of people (**With varying technical skills**) are involved
In building the DBMS system.

$\boxed{\text{This is why, we need different Data Model's to cater to different stakeholders}}$