

- Databases have been widely used to manage big data since the 70's
- The fundamental is **Relational Database** (Note when we say database we mean relational database)
- Many big data systems directly use relational databases or design principles of relational databases.(Amazon, Redshift)

WHAT IS A DATABASE?

- a collection of interrelated data items
- models real-world scenario (entities -> students, courses, instructors)
- **Database Management System** (DBMS) is a software that stores, queries, manages, and analyzes databases.

Problems of using a file system

- want to store the records of students in a file (sid: string, name: string, age: integer, gpa,real)
So suppose you want to find the students who have GPA of at least 3.5 we then must
- open file
- read all records
- extract the fields of each student to get the GPA
- print out if the GPA ≥ 3.5

THIS IS SLOW!!

Every records need to be scanned! Big OUCH!

HMMM

Data Independence

So what if we change file format and we add a new field, say the address, then all application/programs must change in order to access the file correctly.

Another ouch, every app needs to know the file format.

Concurrent Accesses

- If the file is accessed by multiple applications/programs/threads
- what if there is one thread updating the file while another thread is reading the file :(

- what if two thread are updating concurrently?

Durability*

- what if the file system is crashed or the machine is powered off while writing the data?

SOLUTION -> DBMS

- hides all the tedious implementation details from users/application details
- provides a simple query language (SQL)
- data independency, no need to change the applications and remember the metadata when the underlying data change
- fast search, query processing, optimization
- data storage, indexing
- concurrency control, crash recovery

Example

- only specify **what** you want but not **how**
- up to the DB system to figure out the most efficient way to answer the query
- known as **declarative language**

```
SELECT sid
FROM Students
WHERE GPA >= 3.5;
```

```
INSERT INTO Students
VALUES (100, "name", 20, 3.8);
```

Procedure Language

- Java/C++/Python
- must clearly specify steps to be taken to process a query

Declarative Language

- SQL (under the hood, it'll be compiled into relational algebra)
- only specify the query that the user needs answered and not how to answer it
- DBMS is responsible for query compilation and optimization for efficient evaluation

Data Model -> is the high-level abstraction and description of real-world scenarios and applications that hide low-level details.

- defines the way of how to store data

- how to represent relationship between real-world entities
- operations support

A **database** consists of a set of named relations (or tables)


- use tables to store data
- each table has a set of named **attributes** (fields, columns)
- each attribute has a **type** (integer, string)
- set of allowed values for each attribute is called the **domain** (special value is **NULL** is a valid value of any type or domain, unknown or undefined)
- the data is stored as a collection of **tuples** (records, rows)

RELATIONAL DATA MODEL

- attribute values are required to be **atomic**; that is, every element in the domain is **indivisible**
- can only be a single value not a set of values

EX:

sid	name	login	course-enrolled	
53666	Jones	jones@cs	CS448, CS440	not atomic



sid	name	login	course-enrolled
53666	Jones	jones@cs	CS448
53666	Jones	jones@cs	CS440

The **schema** of a table is the set of attributes and associated types, in other words, the structural description of the table.

Ex: (sid:string, name:string, login:string, age:integer, gpa:real)

Instance: the actual contents in the table at a given time.

Key: attribute whose value is unique in each tuple, or a set of attributes whose combined values are unique, e.g, {name, age}.

- why? -> to identify each tuple correctly
- can be used to reference other relationship from other tables
- A.K.A **candidate key** -> the minimal subset of attributes that uniquely identify a tuple in the table.

Primary key -> the candidate key that we choose to be "the key" for a given table

Super key -> a superset of a candidate key e.g. {sid, age}

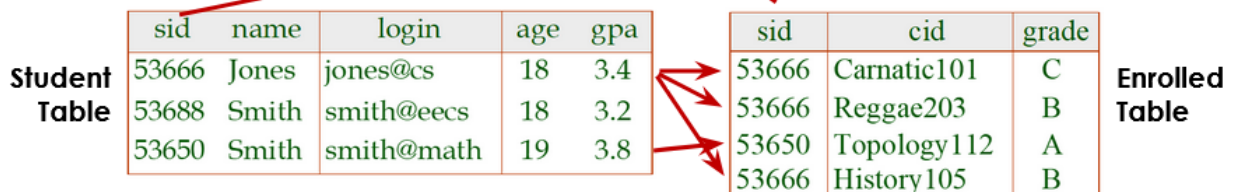
HOW TO CHOOSE A KEY

- Has to be unique like student id, chosen based on real-world application scenarios
- if for a given table no keys are identified, DBMS automatically generates unique keys for the table.

Foreign key -> set of attributes in relation A(enrolled table) that is used to refer to a tuple in relation B (student table) and it must correspond to primary key of the relation B.

- like a logical pointer
- used to represent relationship in real-world scenarios

Table → sid is a foreign key referring to Student



Student Table	sid	name	login	age	gpa
	53666	Jones	jones@cs	18	3.4
	53688	Smith	smith@eecs	18	3.2
	53650	Smith	smith@math	19	3.8

Enrolled Table	sid	cid	grade
	53666	Carnatic101	C
	53666	Reggae203	B
	53650	Topology112	A
	53666	History105	B

```
CREATE TABLE Enrolled
(
  sid CHAR(20), cid CHAR(20), grade CHAR(2),
  PRIMARY KEY (sid,cid),
  FOREIGN KEY (sid) REFERENCES Student(sid))
```

Referential integrity -> if F is a foreign key referring from A (enrolled) to B , then all the values appeared in the F column of A must appear in the F column of B , no dangling pointers.

Data integrity -> specify the constraints imposed on the databases to prevent errors or unexpected behaviors.

Entity integrity -> the primary key of a table cannot be NULL

User-defined integrity -> for example max gpa is 4.0 or name cannot be NULL (we define it)

Other data models with NoSQL:

- Key/Value
- Vector
- Graph
- Document
- Column-family
- Semi-structure (XML)

- Array/Matrix
- Hierarchical