**CMU (Intro to database)**

**Lec1**

………………………………………………………………………………………………………………………………………………………………

🡪**Database**

organized collection of inter-related data that models some aspects of the real

world . (ex : Employees in a company)

**🡪Database Management Studio(DBMS)**

It’s the software package / program that manages database.

Managing database means provide the facility to create ,manipulate ,delete and

retrieve data from database in the best performance way. It also provide security

level which allows only authorized users with specific permissions to deal with

database**.**

**🡪Flat File Issues**

1.Data Integration

There’s no guarantee that the data is consistent as name of objects in an entity differs from another.it can also store other data-types in a column

2.Implementaion

\*searching for a specific record is so difficult

\*accessing file from two developers ‘d cause some problems such inconsistency and concurrence control

3.Durabilty

When manipulating data while the machine crashes down.

**🡪Data Model**

high level collection of concepts that describe the data stored in the database.

Example :

\*Relational Model

\*Key/Value

\*graph

\*Column-family

\*Array/Matrix

\*Hierarchical

\*Network

\*Multi-Value

**🡪Schema**

Description of particular collection of data, using a data model.

**🡺Relational Model**

A type of data model introduced by Ted Codd in 1970 as he noticed that people were rewriting DBMSs every time they want to change the physical layer .

The relational model has three keys :

* store database in a simple data structure called relation

relation 🡪 an unordered set that contains the relationship of attributes that

represent entities.

* Access database through high-level language.

Rather than telling DBMS explicitly **how** to retrieve data from database ,just tell it **what** you need and DBMS will choose the best way to retrieve it.

* Physical storage of the data should be left up to DBMS for implementation

The relational data model defines three concepts:

• **Structure**: The definition of relations and their contents. This is the attributes the relations have and the values that those attributes can hold.

• **Integrity**: Ensure the database’s contents satisfy constraints. An example constraint would be that any value for the year attribute has to be a number.

• **Manipulation**: How to access and modify a database’s contents.

**🡺primary key**

An attribute that uniquely identifies each tuple in the relation .

Some DBMS automatically generate this attribute

🡺**Foreign Key**

specifies that an attribute from one relation has to map to a tuple in another relation.

🡺**Data manipulation Languages(DML)**

**Methods to store and retrieve information from the database**

* **Procedural :** the query specifies the strategy that the DBMS should use to retrieve data from database(**how** to retrieve).
* **Non-procedural (Daclarative) :**

the query only tell the DBMS what you want and DBMS determines the best way to retrieve it(**what** you need).

**🡺Relational Algebra**

* set of fundamental operations to retrieve and manipulate tuples in a relation
* each operator take a relation or more than one as an input and retrieve a new relation as an output
* in complex query ,you can use more than one operator as a chain

1. **Select**

Retrieve a subset of the tuples that satisfy a certain predicate. Maps to **where** clause in SQL

1. **Projection**

Generate a relation with tuples that contain only specified attributes which can rearrange the attributes order and manipulating its values.

Maps to **select** clause in SQL.

1. **Union**

Generate new relation contains all tuples that appears in only one or more input relations. Maps to **Union All** in SQL

1. **Intersection**

Generate a new relation contains only tuples that appear in all input relations. Maps to **Intersect** clause in SQL.

1. **Difference**

Generate a new relation contains tuples that only appear in the first input relation and not in the second input relation. (R - S)

Maps to **Except** clause in SQL.

1. **Product**

Generate a new relation contains all possible combinations of tuples from the input relations. (R \* S)

Maps to **Cross Join** in SQL

1. **Join**

Generate a new relation contains all tuples that are a combination of two tuples (one from each input relation).

Maps to **Join** Clause in SQL