

# *Databases, DBMS and SQL*

*IICT Lecture 06*

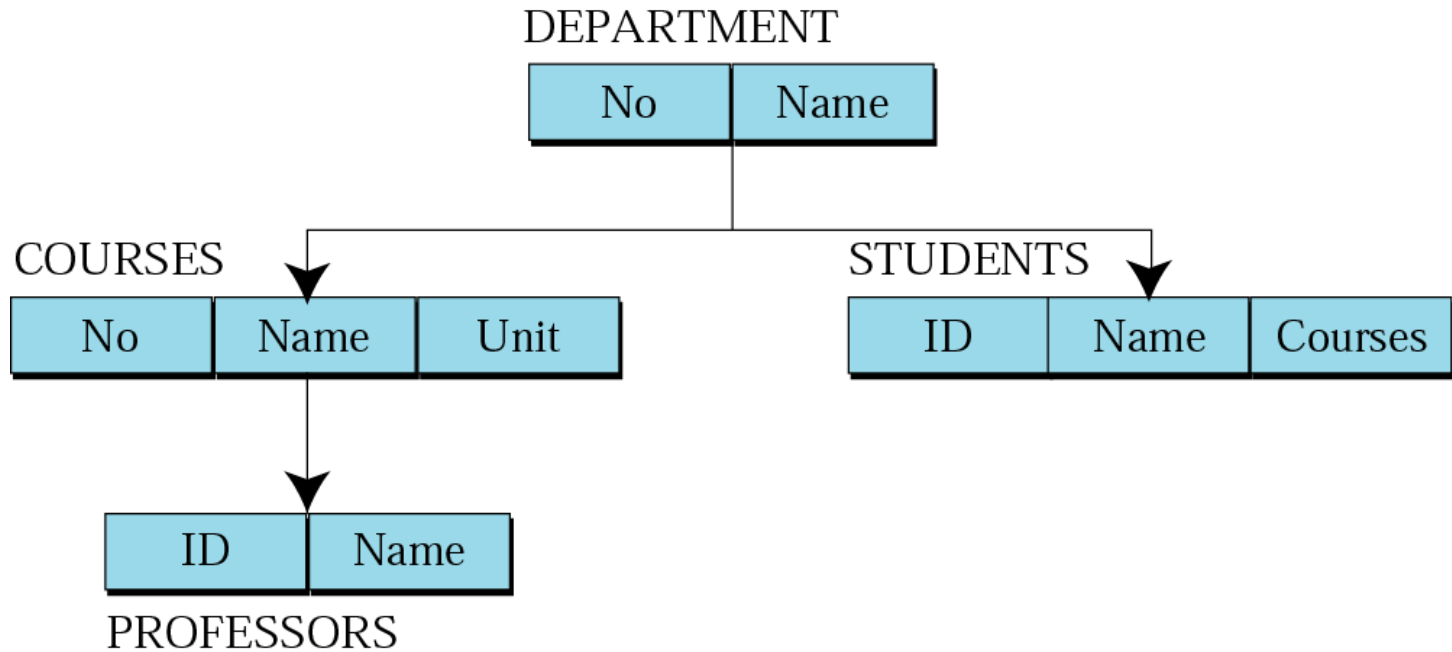
# What is a Database?

- An organized collection of Data
- A comprehensive collection of related data organized for convenient access, generally in a computer

# Database Model

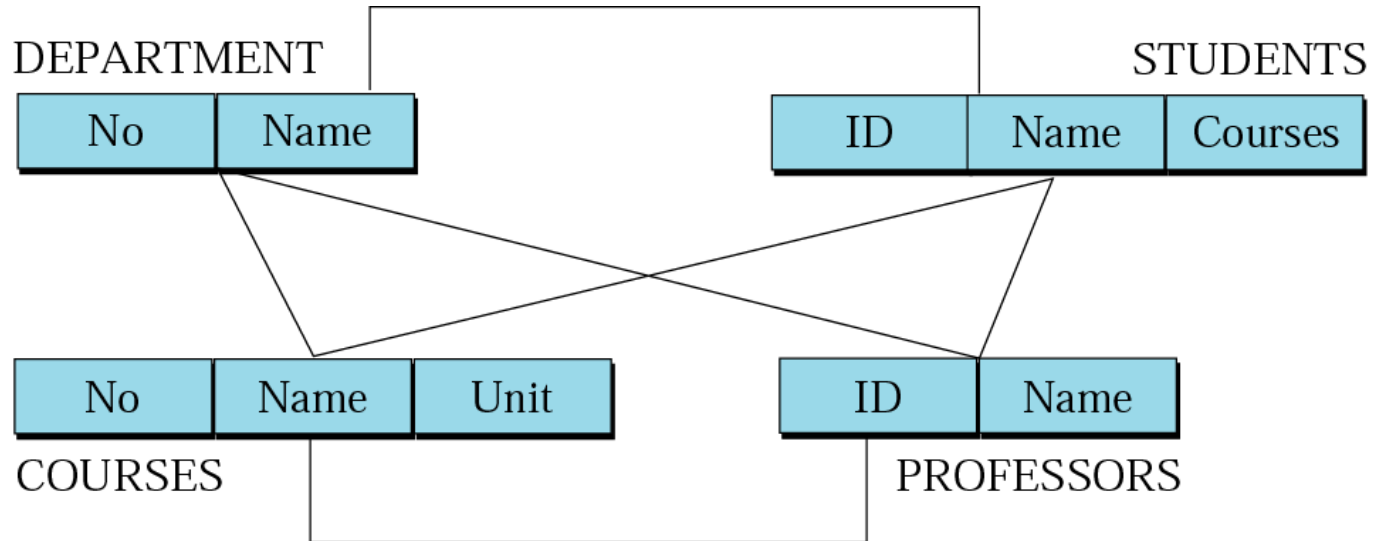
- Database model defines the logical design of data.
- Database model describes the relation between different parts of data.
- There are three database models:
  1. Hierarchical Model
  2. Network Model
  3. Relational Model

# Hierarchical model



- Data are organized in an upside down tree
- Each entity has one parent and many children
- Old and not used now

# Network model



- Entities are organized in a graph
- Entities can be accessed through several paths
- Old and not used

# Relational model

DEPARTMENT

No	Name
...	...
...	...
...	...

PROFESSORS

ID	Name	Dept-No	Courses
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

COURSES

No	Dept-No	Prof-ID	Unit
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

STUDENTS

ID	Name	Courses
...	...	...
...	...	...
...	...	...
...	...	...

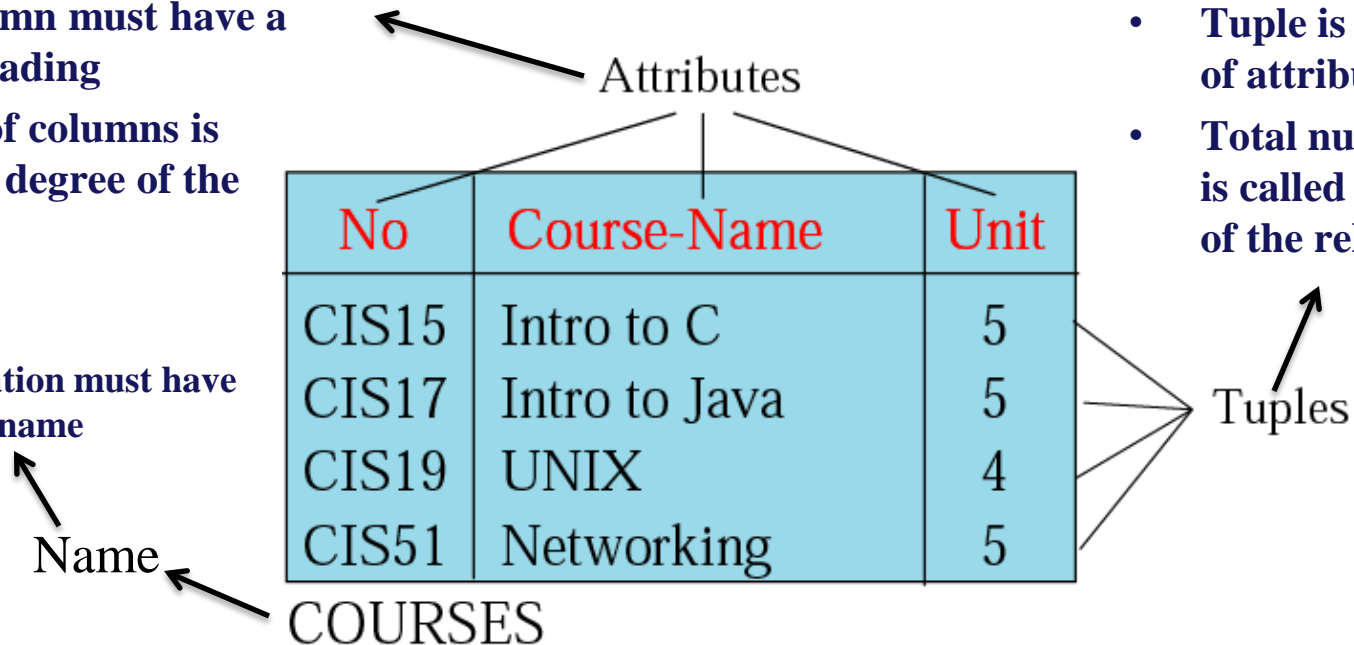
- **Data are organized in two dimensional tables (relations)**
- **Tables re related to each other**
- **Relational Database Management System (RDBMS) are more common model used today**

# Relation (Name, Attributes, Tuples)

- Attributes are the column heading
- Each column must have a unique heading
- Number of columns is called the degree of the relation

- Each relation must have a unique name

- Tuple is a collection of attribute value
- Total number of rows is called Cardinality of the relation



- Relation appears in 2 dimensional table
- That doesn't mean data stored as table; the physical storage of data is independent of the logical organization of data

*OPERATIONS*  
*ON*  
*RELATIONS*



# Insert operation

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5

Insert

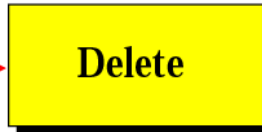
No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
<i>CIS52</i>	<i>TCP/IP Protocols</i>	<i>6</i>

- **Unary operation**
- **Insert Operation: Inserts new tuple into the relation**

# Delete operation

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS51	Networking	5
CIS52	TCP/IP Protocols	6

- **Unary operation**
- **Delete Operation: Deletes tuple from the relation**

# Update operation

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



**Update**



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	<b>6</b>
CIS52	TCP/IP Protocols	6

- **Unary operation**
- **Update Operation: Changes the values of some attributes of a tuple**

# Select operation

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS51	Networking	5

- **Unary operation**
- **Select Operation: Uses some criteria to select some tuples from the original relation**

# Project operation

## COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



No	Unit
CIS15	5
CIS17	5
CIS19	4
CIS51	5
CIS52	6

- **Unary operation**
- **Project Operation: Creates relation in which each tuple has fewer attributes**

# Join operation

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6

TAUGHT-BY

No	Professor
CIS15	Lee
CIS17	Lu
CIS19	Walter
CIS51	Lu
CIS52	Lee

Join

No	Course-Name	Unit	Professor
CIS15	Intro to C	5	Lee
CIS17	Intro to Java	5	Lu
CIS19	UNIX	4	Walter
CIS51	Networking	5	Lu
CIS52	TCP/IP Protocols	6	Lee

- **Binary operation**
- **Join Operation: Takes two relation and combine them based on common attribute**

# Union operation

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

CIS52-Roster

Student-ID	F-Name	L-Name
342-88-9999	Rich	White
145-67-6754	John	Brown
232-56-5690	George	Yellow

Union

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple
342-88-9999	Rich	White

- **Binary operation**
- **Union Operation: Creates new relation in which each tuple is either in the first relation, the second relation or in both**

# Intersection operation

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

CIS52-Roster

Student-ID	F-Name	L-Name
342-88-9999	Rich	White
145-67-6754	John	Brown
232-56-5690	George	Yellow

Intersection

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow

- **Binary operation**
- **Intersection Operation: Creates new relation in which each tuple is in both relations.**



# Difference operation

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

CIS52-Roster

Student-ID	F-Name	L-Name
342-88-9999	Rich	White
145-67-6754	John	Brown
232-56-5690	George	Yellow

Difference

Student-ID	F-Name	L-Name
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

- **Binary Operation**
- **Difference Operation:** Creates new relation where the new tuples are in the first relation but not in the second.

# Database Management System

- A database management system (DBMS) is system software for creating and managing Database.
- The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.
- DBMS allow all the operations on database discussed in previous slides
  - Inserte, Delete, retrieve, Union, Join etc...

***STRUCTURED  
QUERY  
LANGUAGE***

# SQL

- SQL is the standard language used for relational databases.
- It is declarative language where users declare what they want without having to write a step by step procedure.
- It was first implemented by Oracle Corporation

# 1. Insert

- SQL Insert Operation format

**insert into**    **RELATION-NAME**  
**values**        **(..., ..., ...)**

# Insert (Example)

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5



**Insert**



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
<i>CIS52</i>	<i>TCP/IP Protocols</i>	<i>6</i>

**insert into** COURSES  
**values** ("CIS52", "TCP/IP Protocols", 6)

## 2. Delete

- SQL Delete Operation format

**delete from**    **RELATION-NAME**  
**where**            **criteria**

# Delete (Example)

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



**Delete**



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS51	Networking	5
CIS52	TCP/IP Protocols	6

**Delete from** COURSES  
**where** No = "CIS19"



# 3. Update

- SQL Update Operation format

**update**    **RELATION-NAME**  
**set**        attribute1 = value1    attribute 2 = value2 ...  
**where**    criteria

# Update (Example)

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



**Update**



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	<b>6</b>
CIS52	TCP/IP Protocols	6

**update** COURSES  
**set** unit = 6  
**where** No = "CIS51"

# 4. Select

- SQL Select Operation format

```
select *  
from RELATION-NAME  
where criteria
```

# Select (Example)

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



Select



No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS51	Networking	5

```
select *  
from   COURSES  
where  Unit = 5
```

# 5. Project

- SQL Project Operation format

**select**      attribute-list  
**from**      RELATION-NAME

# Project (Example)

## COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6



No	Unit
CIS15	5
CIS17	5
CIS19	4
CIS51	5
CIS52	6

**select** No, Unit  
**from** COURSES

# 6. Join

- SQL Join Operation format

**select**    attribute-list  
**from**     RELATION NO1, RELATION NO2  
**where**    criteria

# Join (Example)

COURSES

No	Course-Name	Unit
CIS15	Intro to C	5
CIS17	Intro to Java	5
CIS19	UNIX	4
CIS51	Networking	5
CIS52	TCP/IP Protocols	6

TAUGHT-BY

No	Professor
CIS15	Lee
CIS17	Lu
CIS19	Walter
CIS51	Lu
CIS52	Lee



No	Course-Name	Unit	Professor
CIS15	Intro to C	5	Lee
CIS17	Intro to Java	5	Lu
CIS19	UNIX	4	Walter
CIS51	Networking	5	Lu
CIS52	TCP/IP Protocols	6	Lee

**select** No, Course-Name, Unit, Professor  
**from** COURSES, TAUGHT-BY  
**where** COURSES.No = TAUGHT-BY.No;



# 7. Union

- SQL Union Operation format

```
select    *  
from      RELATION NO1  
union  
select    *  
from      RELATION NO2
```

# Union (Example)

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

CIS52-Roster

Student-ID	F-Name	L-Name
342-88-9999	Rich	White
145-67-6754	John	Brown
232-56-5690	George	Yellow

Union

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple
342-88-9999	Rich	White

```
select *  
from CIS15-Roster  
union  
select *  
from CIS52-Roster;
```

# 8. Intersection

- SQL Intersection Operation format

```
select    *  
from      RELATION NO1  
intersection  
select    *  
from      RELATION NO2
```

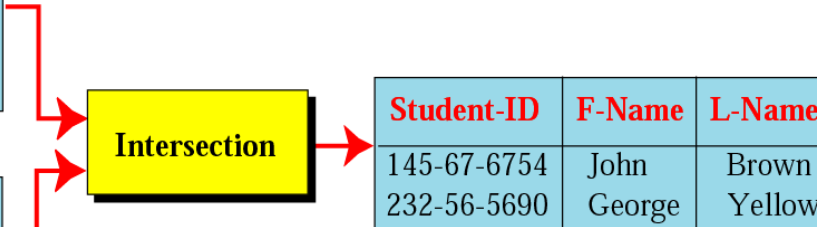
# Intersection (Example)

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

CIS52-Roster

Student-ID	F-Name	L-Name
342-88-9999	Rich	White
145-67-6754	John	Brown
232-56-5690	George	Yellow



```
select      *
from        CIS15-Roster
intersection
select      *
from        CIS52-Roster;
```

# 9. Difference

- SQL Difference Operation format

```
select *  
from RELATION NO1  
minus  
select *  
from RELATION NO2
```

# Intersection (Example)

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

CIS52-Roster

Student-ID	F-Name	L-Name
342-88-9999	Rich	White
145-67-6754	John	Brown
232-56-5690	George	Yellow

Difference

Student-ID	F-Name	L-Name
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

```
select *  
from CIS15-Roster  
minus  
select *  
from CIS52-Roster;
```

***OTHER  
DATABASE  
MODELS***

# The levels of Data

<b>Database</b>	<b>One or more tables</b>
<b>Table (relation)</b>	<b>A collection of Records</b>
<b>Record/Tuple</b>	<b>A group of related fields</b>
<b>Field</b>	<b>One or more character</b>
<b>Character</b>	<b>At least 8 bits</b>
<b>Bit</b>	<b>0 or 1</b>

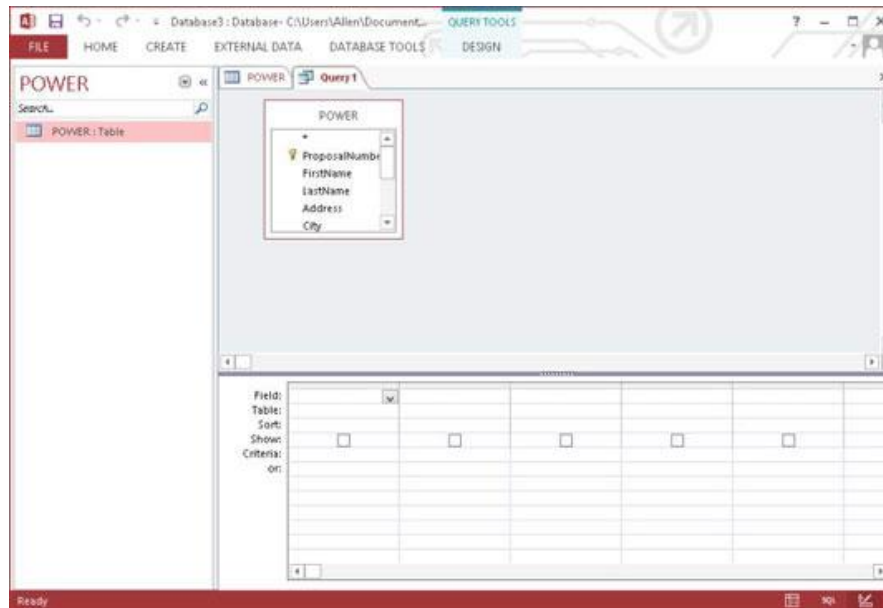


# 5 characteristics of Good Database

<b>Data Integrity</b>	Ensuring data is valid
<b>Data Independence</b>	Data is separated from software
<b>Avoiding data Redundancy</b>	Repetition of input data is avoided
<b>Data Security</b>	Data is not accessible to unauthorized users
<b>Data Maintenance</b>	Set procedures for adding ,deleting ... records for the purpose of optimization

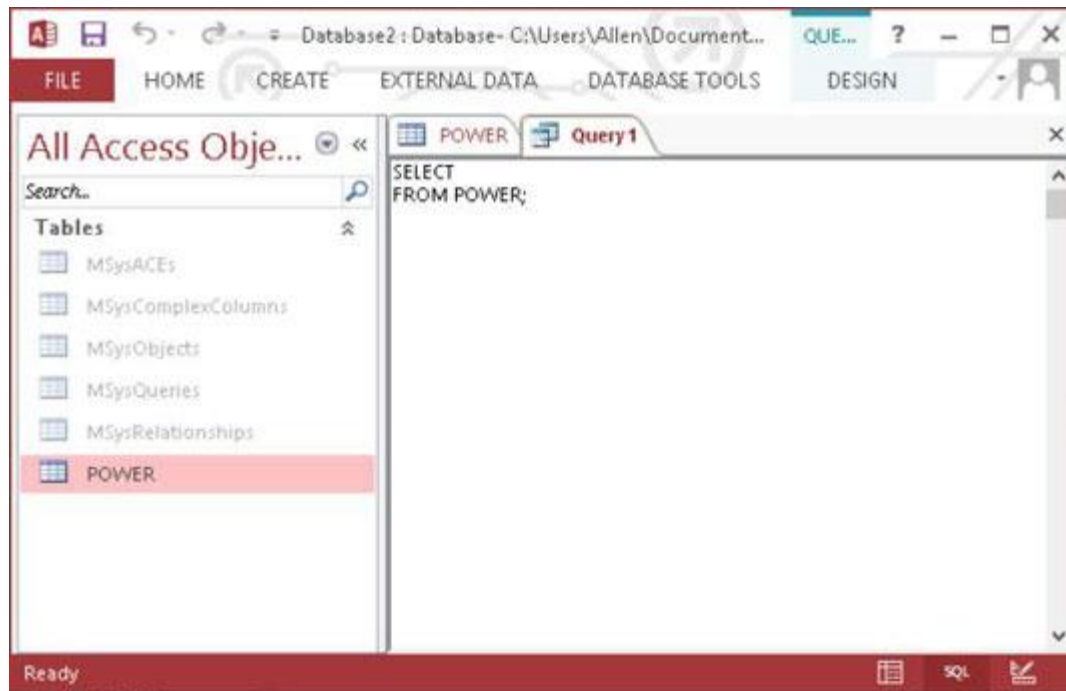
# How To Write And Run SQL Query

- **Open your database and click the CREATE tab.**
  - This will display the ribbon across the top of the window.
- **Click Query Design in the Queries section.**
  - The Show Table dialog box appears.
- **Select the POWER table. Click the Add button and then click the Close button to close the dialog box.**



# How To Write And Run SQL Query

- **Click the Home tab and then the View icon in the left corner of the Ribbon.**
  - A menu drops down, displaying the different views available to you in query mode. One of those views is SQL View.
- **Click SQL View to display the SQL View Object tab.**



# How To Write And Run SQL Query

- **Fill in an asterisk (\*) in the blank area in the first line and add a WHERE clause after the FROM line.**
  - If you had already entered some data into the POWER table, you could make a retrieval with something like:
  - `SELECT * FROM POWER WHERE LastName = 'Marx' ;`
- **Enter a name and then click OK.**
  - Your statement is saved and can be executed as a query later.