Chapter: Database

Hà Nội

Learning objectives

- The limitations of a file-based approach to storage and retrieval of data
- The features and terminology of a relational database
- Entity-relationship (E-R) diagrams to document database design
- Normalisation to third normal form (3NF)
- Producing a normalised database design for a given set of data or tables
- The features provided by a database management system (DBMS)
- The creation and modification of a database structure using a database definition language (DDL)
- Queries and the maintenance of a database using a database manipulation language (DML)
- Using SQL as a DDL and as a DML
- Understand and write a SQL script

1 Database concepts

1. **Flat-file database**
2. **Relational database**

2: Database Management Systems (DBMSs)

1. **DBMSs address file based approach**
2. **Use and purpose of DBMS tools**

3: Data definition language (DDL) and data manipulation language (DML)

1. Data definition language (DDL)
2. Data manipulation language (DML)

Database concepts

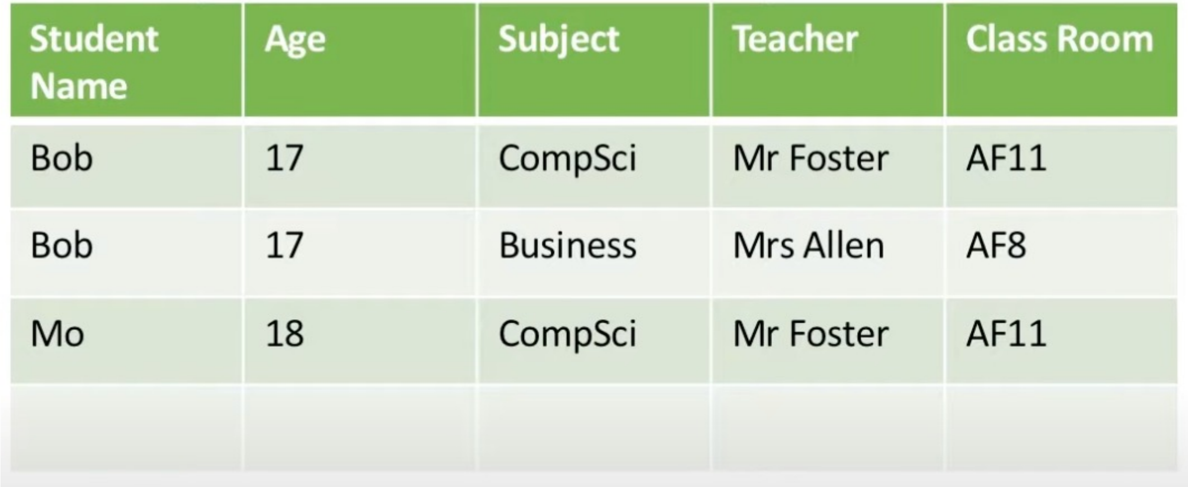
- File based approach
- Database
- Relational database terminology
  - Entity, Table, Records and Fields
  - Tuple, Attributes
  - Primary key, Candidate key, Secondary key, Foreign key
- Relationship
- Normalization process

Relational Database concepts

- File based approach
- Database definition
- Relation database concepts
- E-R diagram

File based approach

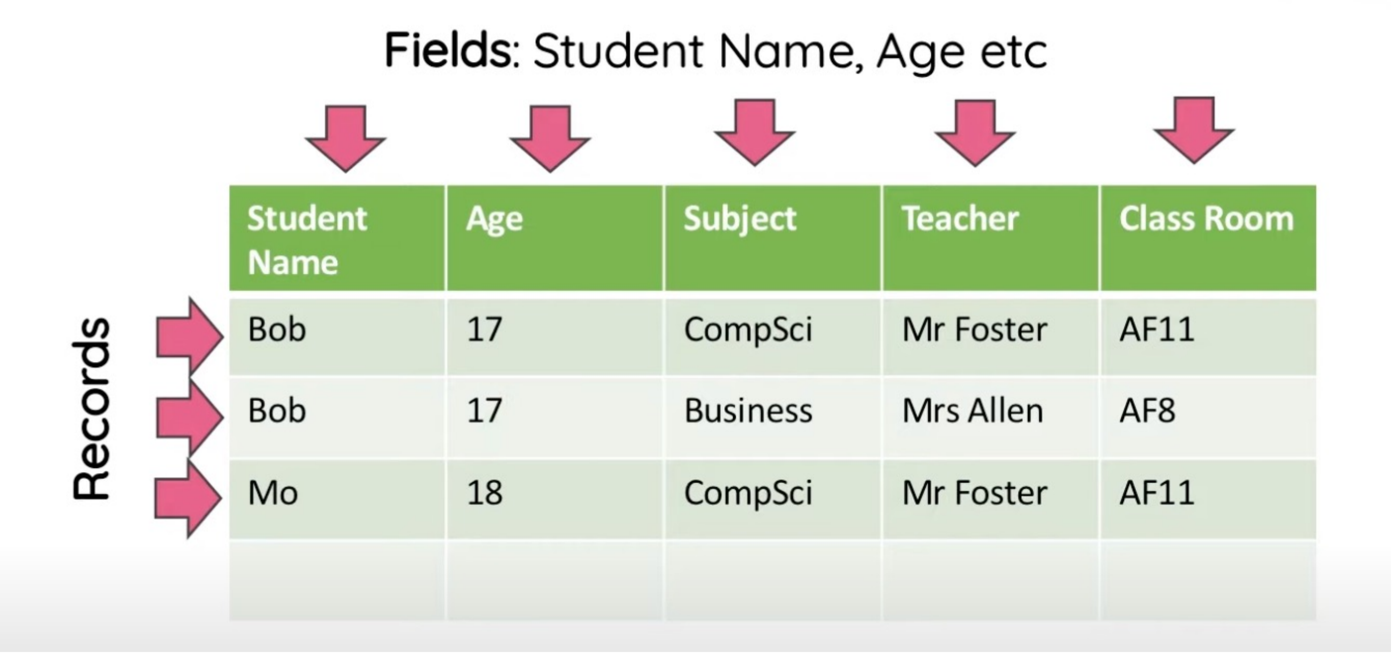
- Based on flat-file
- Flat file is a file consisting of a single table
- Individual elements of data can be called data items.
- Example:



Student Name	Age	Subject	Teacher	Class Room
Bob	17	CompSci	Mr Foster	AF11
Bob	17	Business	Mrs Allen	AF8
Mo	18	CompSci	Mr Foster	AF11

File based approach

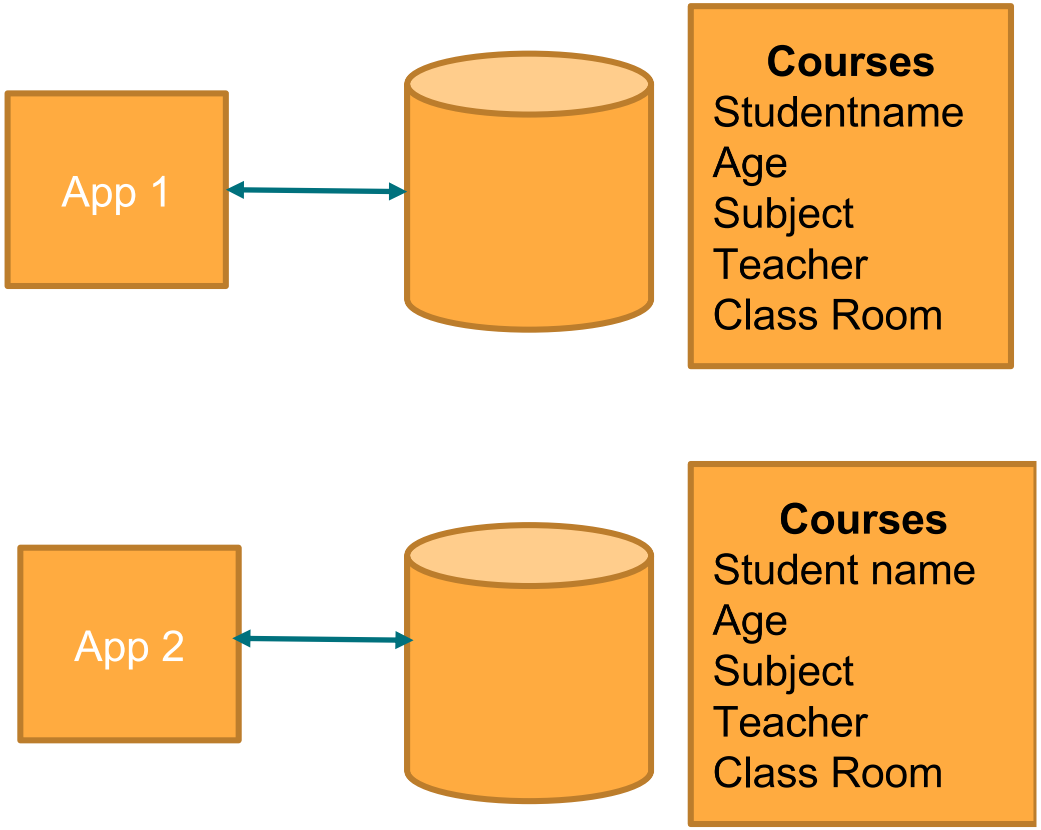
- Data is stored as a table:
  - Each column is a field
  - Each row is a record


Fields: Student Name, Age etc
Records

Student Name	Age	Subject	Teacher	Class Room
Bob	17	CompSci	Mr Foster	AF11
Bob	17	Business	Mrs Allen	AF8
Mo	18	CompSci	Mr Foster	AF11

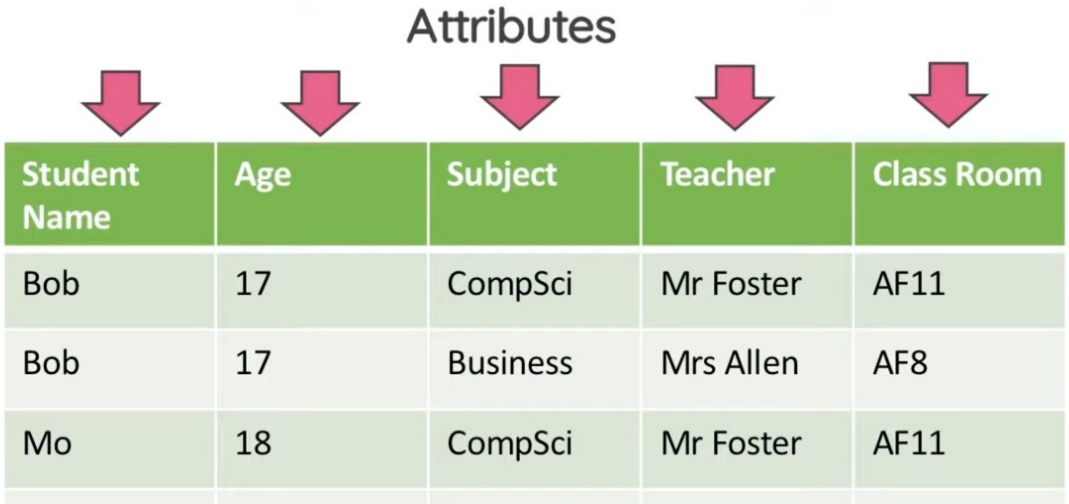
File based approach disadvantages

- Data items are duplicated by the separate applications and some data is redundant
- data can be altered by one application and not by another; it then becomes inconsistent
- enquiries available can depend on the structure of the data and the software used so the data is not independent.

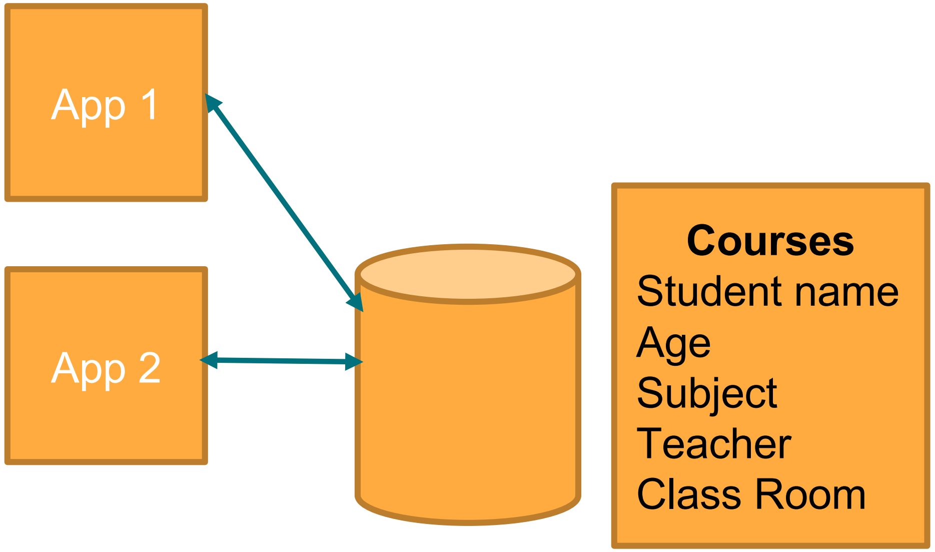


Database

- Database is structured collection of items of data that can be accessed by different applications programs
- It is based on entities and its attributes


Attributes

Student Name	Age	Subject	Teacher	Class Room
Bob	17	CompSci	Mr Foster	AF11
Bob	17	Business	Mrs Allen	AF8
Mo	18	CompSci	Mr Foster	AF11


App 1

App 2

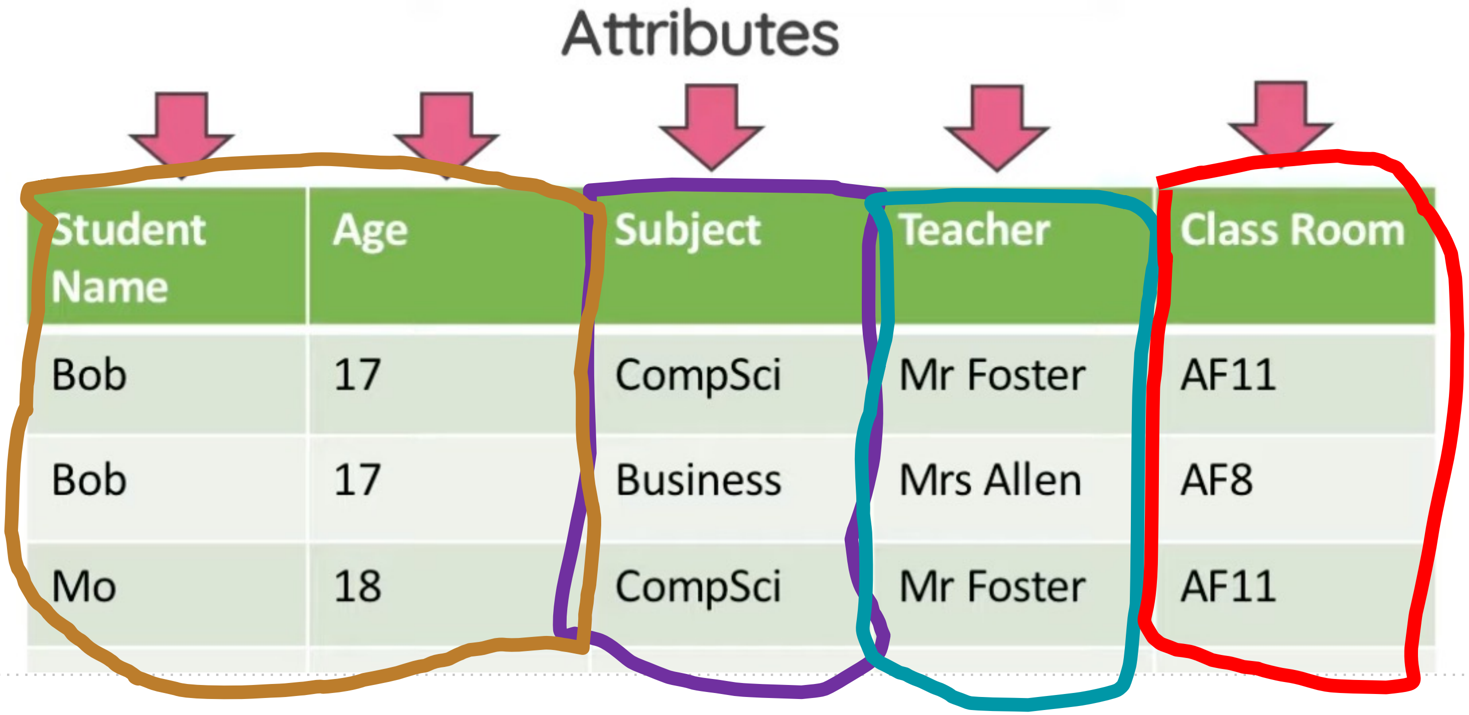
Courses
Student name
Age
Subject
Teacher
Class Room

Database

- What is an entity?
- What is an attribute?
- An entity is anything that can have data stored about it, such as a person, place, event or object.
- An attribute is an individual data item stored for an entity

Database

- How many entities are there in the database?
- How many attributes does each entity have?



Attributes

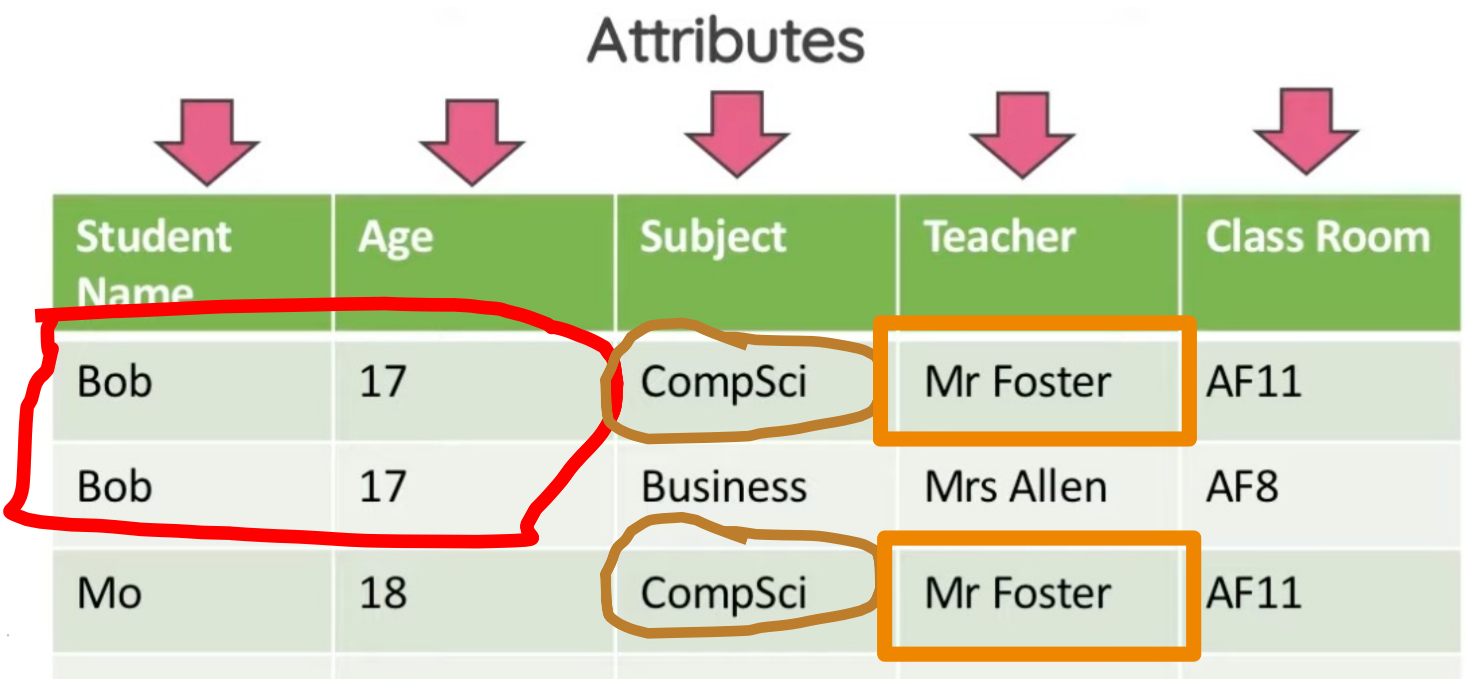
Student Name	Age	Subject	Teacher	Class Room
Bob	17	CompSci	Mr Foster	AF11
Bob	17	Business	Mrs Allen	AF8
Mo	18	CompSci	Mr Foster	AF11

Relational Database

- One record of an entity possibly is duplicated many times in many records

=> If an attribute changes, all records containing that attribute need updating.

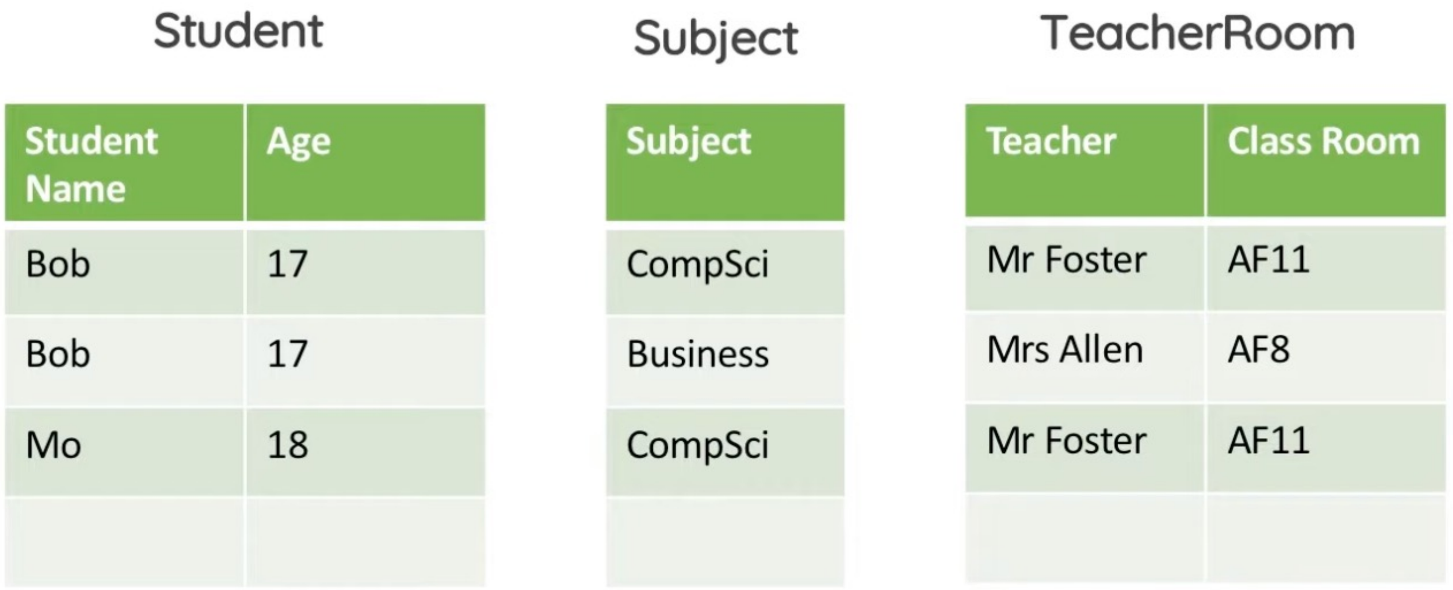
=> May lead to data inconsistency


Attributes

Student Name	Age	Subject	Teacher	Class Room
Bob	17	CompSci	Mr Foster	AF11
Bob	17	Business	Mrs Allen	AF8
Mo	18	CompSci	Mr Foster	AF11

Relational Database

- **Relational database** is a database in which the data items are linked by internal pointers.



Student

Student Name	Age
Bob	17
Bob	17
Mo	18

Subject

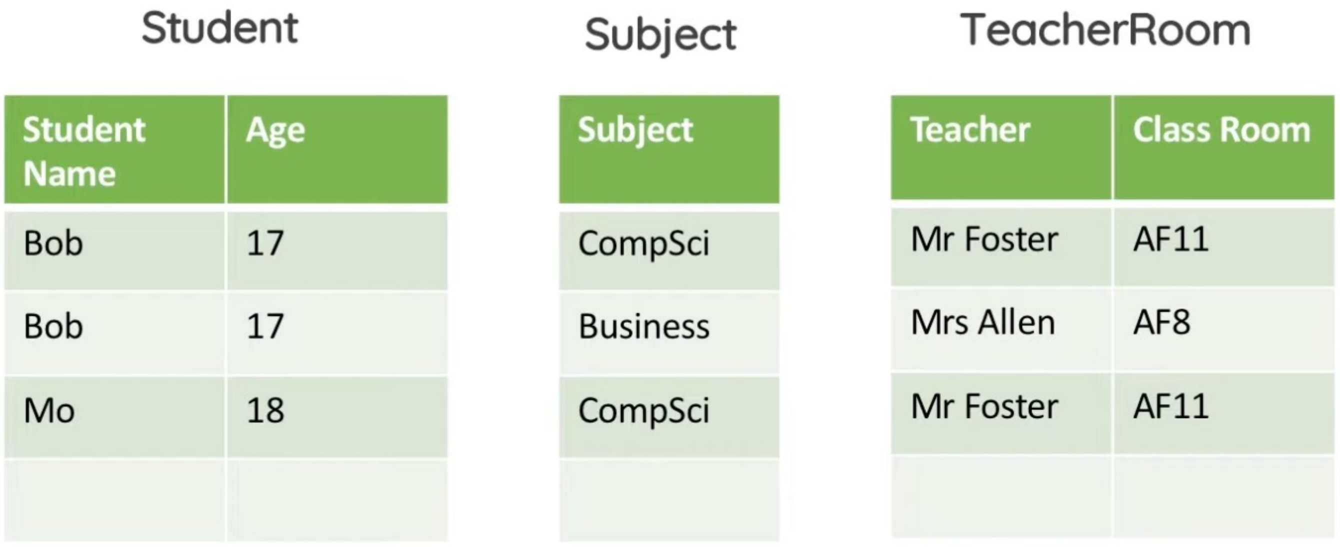
Subject
CompSci
Business
CompSci

TeacherRoom

Teacher	Class Room
Mr Foster	AF11
Mrs Allen	AF8
Mr Foster	AF11

Relational Database

- **Problems:**
  - **Duplicate data**
  - **How are the entity related to each other?**



Student

Student Name	Age
Bob	17
Bob	17
Mo	18

Subject

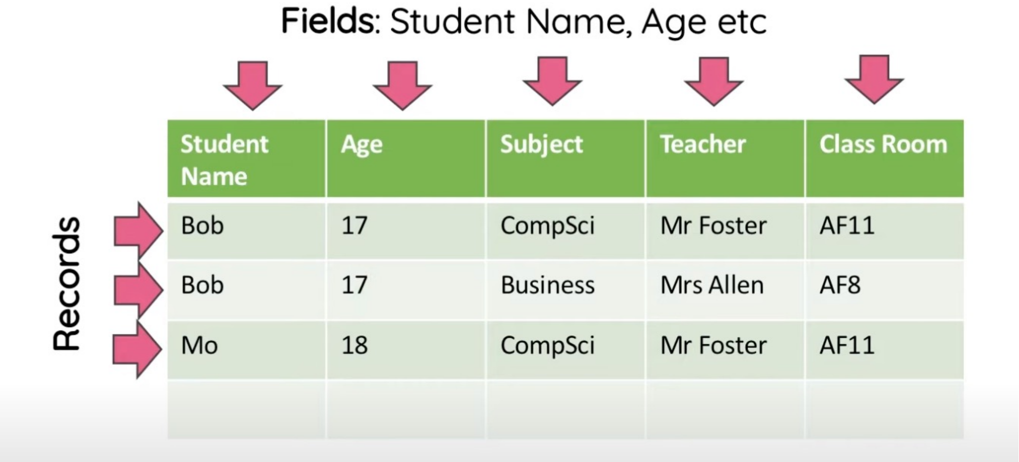
Subject
CompSci
Business
CompSci

TeacherRoom

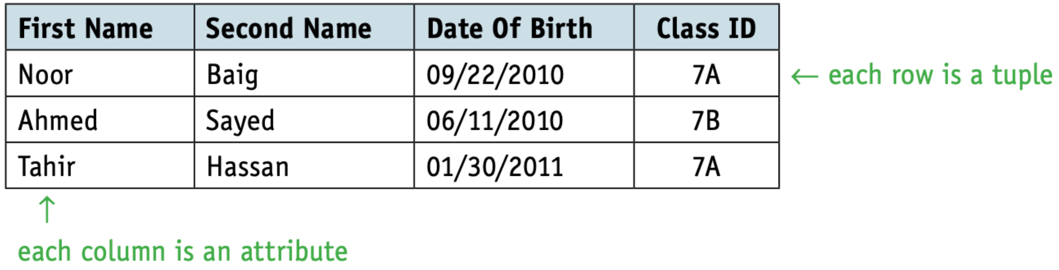
Teacher	Class Room
Mr Foster	AF11
Mrs Allen	AF8
Mr Foster	AF11

Relational Database

- **Table?**
  - A **table** is a group of similar data, in a database, with rows for each instance of an entity and columns for each attribute.
- **Record/tuple?**
  - A **record** is a row in a table in a database
- **Field?**
  - A **field** is a column in a table in a database


Fields: Student Name, Age etc
Records

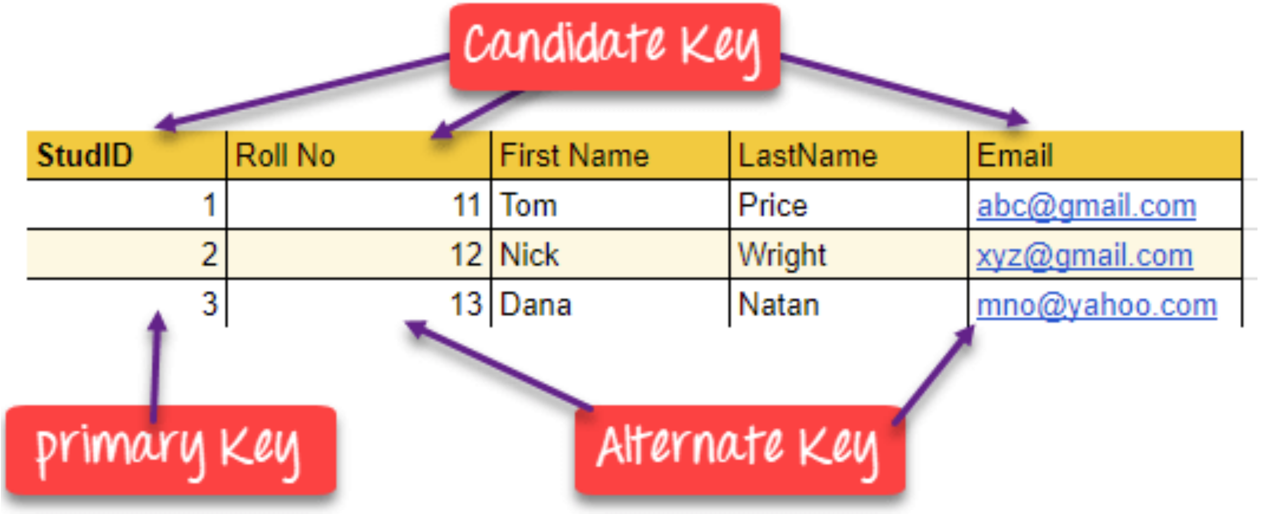
Student Name	Age	Subject	Teacher	Class Room
Bob	17	CompSci	Mr Foster	AF11
Bob	17	Business	Mrs Allen	AF8
Mo	18	CompSci	Mr Foster	AF11


← each row is a tuple
↑
each column is an attribute

First Name	Second Name	Date Of Birth	Class ID
Noor	Baig	09/22/2010	7A
Ahmed	Sayed	06/11/2010	7B
Tahir	Hassan	01/30/2011	7A

Relational Database - Keys

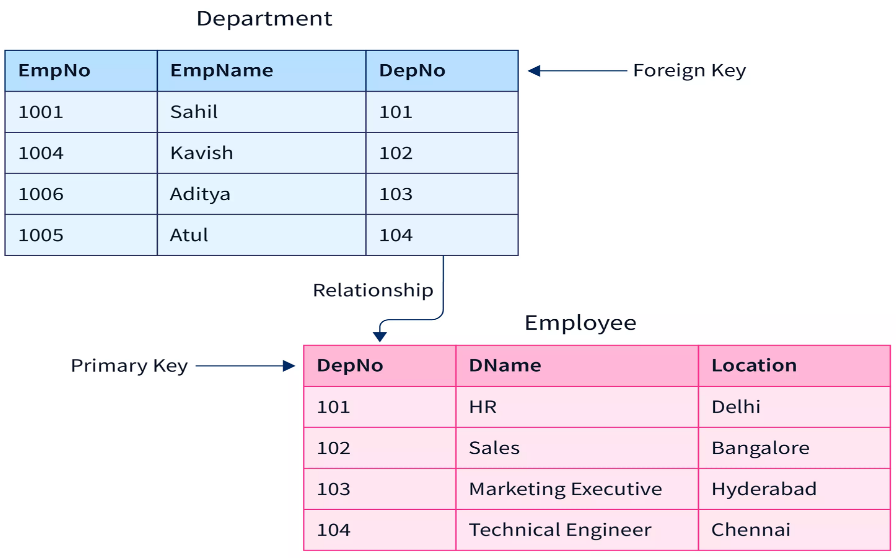
- Candidate key?
- Primary key?
- Secondary/alternate key?



StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

Relational Database - Keys

- **Foreign key?**



Department

EmpNo	EmpName	DepNo
1001	Sahil	101
1004	Kavish	102
1006	Aditya	103
1005	Atul	104

Foreign Key

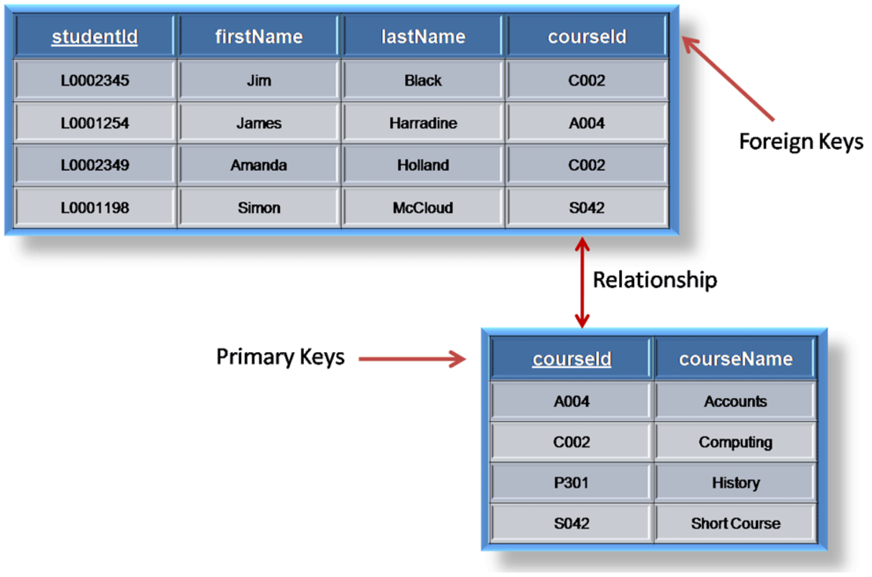
Relationship

Employee

Primary Key

DepNo	DName	Location
101	HR	Delhi
102	Sales	Bangalore
103	Marketing Executive	Hyderabad
104	Technical Engineer	Chennai

SCALER Topics



studentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

Foreign Keys

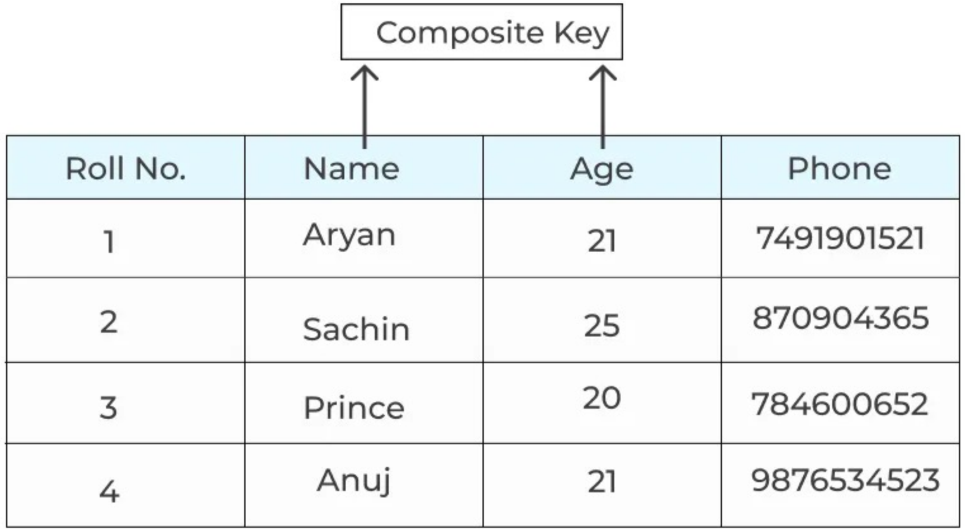
Relationship

Primary Keys

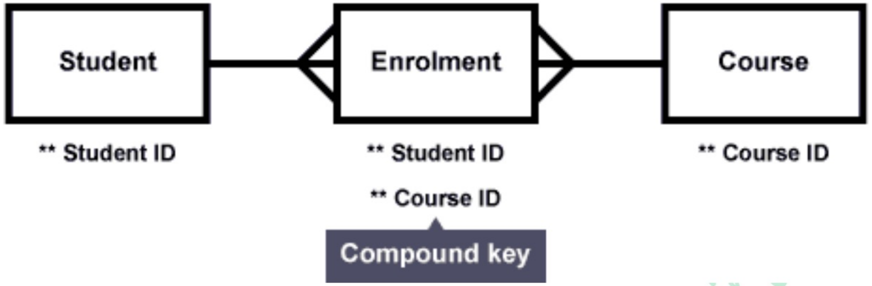
courseId	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

Relational Database - Keys

- **Composite key?**
  - ○ Primary key that composes of two or more fields.
- **Compound key?**
  - ○ Is composite key created by two or more fields that are primary keys of other tables.

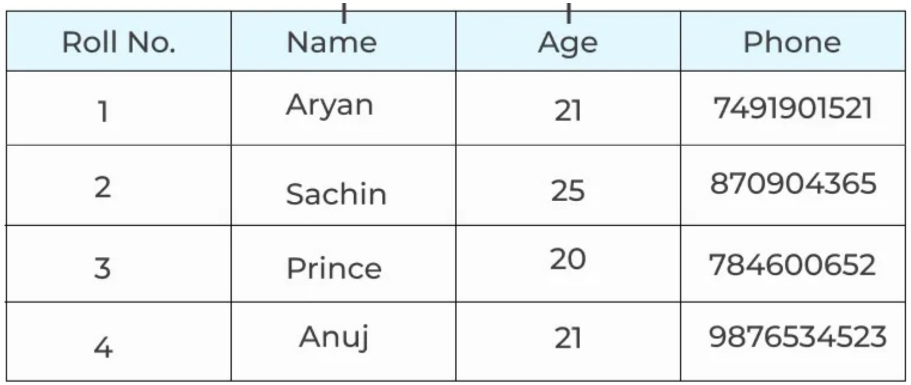

Composite Key

Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523



Relational Database – Table Schema

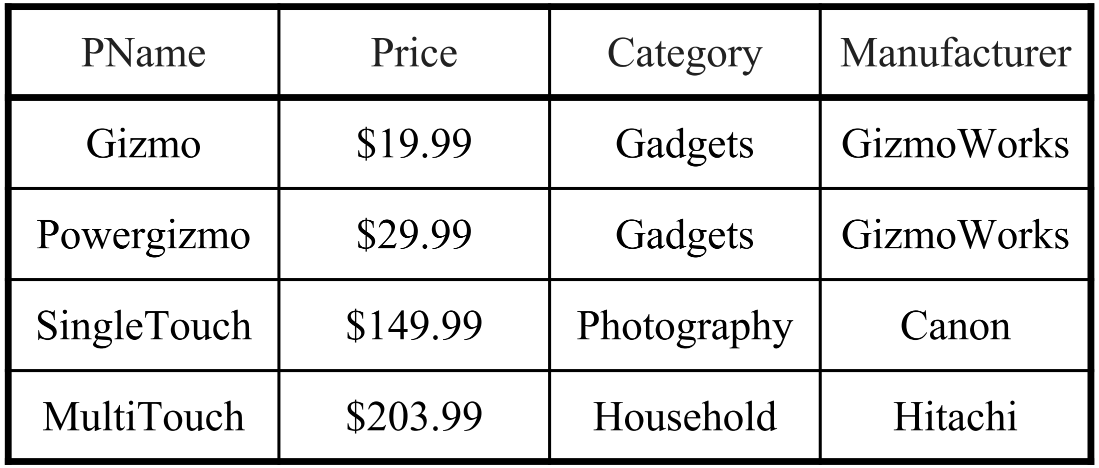
- The *schema* of a table is the table name and its attributes:
- Primary key is underlined



Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

Roll(RollNo,Name, Age, Phone)

Product

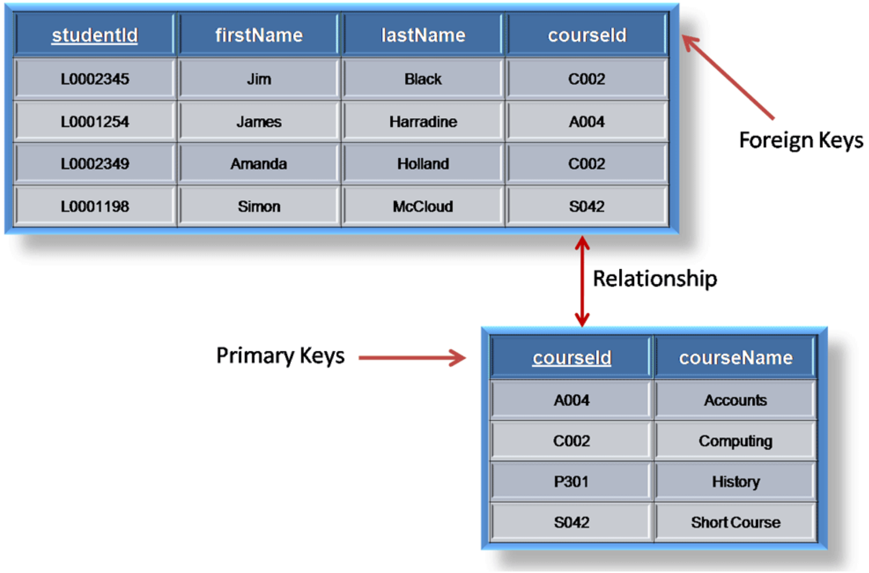


PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Product(PName, Price, Category, Manfacturer)

Relational Database - Relationship

- **Relationship** between tables are formed when a table has a foreign key referring to primary key of another table
- Relationships can take several forms
  - one-to-one 1:1
  - one-to-many 1:m
  - many-to-one m:1
  - many-to-many, m:m



studentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

Foreign Keys

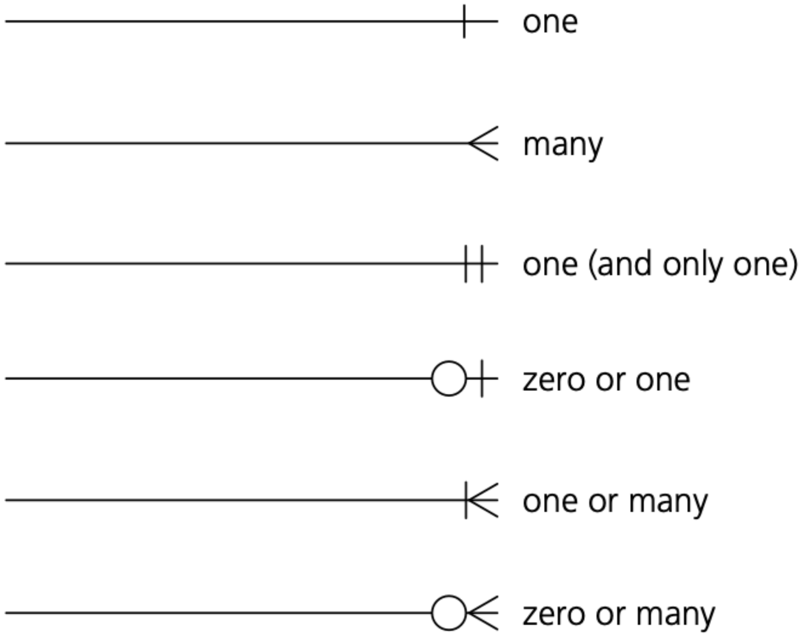
Relationship

Primary Keys

courseId	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

Entity – Relation Diagram

- An **E-R diagram** can be used to document the design of a database.
- Relationships and its cardinality (degree)



Normalisation process

- What is normalization?
- 1NF
- 2NF
- 3NF

Database Management System

Database Management System - DBMS

- A **DBMS is** system to **manage**, organise and maintain data in a database.
- Every DBMS has an already-defined data structure to set up and create the database
- Examples: MySQL, SQL Server…
- DBMS features:
  - **Data dictionary**
  - **Data modeling**
  - **Logical schema**
  - **Indexing capability**
  - **Control access rights for users**
  - **Backup procedures**

Database Management System - DBMS

- DBMS features:
  - **Data dictionary:** a set of data that contains metadata (data about other data) for a database.

    - A DBMS uses a data dictionary to store the metadata, including the definition of tables, attributes, relationships between tables and any indexing.

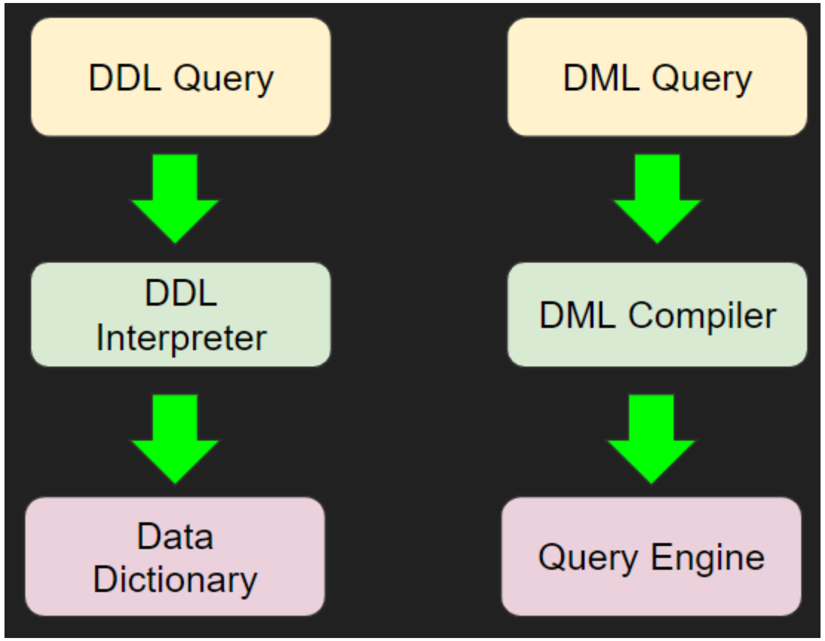
    - The data dictionary can also define the validation rules used for the entry of data and contain data about the physical storage of the data.
  - **Data modeling:** is an important tool used to show the data structure of a database. An E-R diagram is an example of a data model.
  - **Logical schema:** is a data model for a specific database that is independent of the DBMS used to build the database.

Database Management System - DBMS

- DBMS features:
  - **Indexing capability:** A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure.
  - **Control access rights for users** - Access rights: the permissions given to database users to access, modify or delete data.
  - **Backup procedures:** A DBMS can simplify the database backup process by providing a simple interface to manage backups.

DBMS software tools

- **Developer interface:** allows a developer to write queries in **structured query language (SQL)** rather than using query-by-example.
- **Query processor:** Process queries written in SQL. The query processor includes:
  - A DDL interpreter: Interpret DDL statements and record in the database's data dictionary
  - A DML compiler: compile statements into low level instructions and optimize the query
  - A query evaluation engine: Execute low level instructions



DBMS address limitations of file based approach

- **Data redundancy issue:** Reduce the duplication of data by storing data in separate linked tables where most items of data are only stored once.
- **Data inconsistency issue:**
  - ○ Storing most items of data only once, allowing updated items to be seen by all applications.
  - ○ Easier to maintain as an item of data will only be changed once, not multiple times, by different applications.
- **Data dependency issue:**
  - ○ Data is independent of the applications using the database, so changes made to the structure of the data will be managed by the DBMS
  - ○ Changes have little or no effect on the applications using the database.

Data Definition Language (DDL) and Data Manipulation Language (DML)

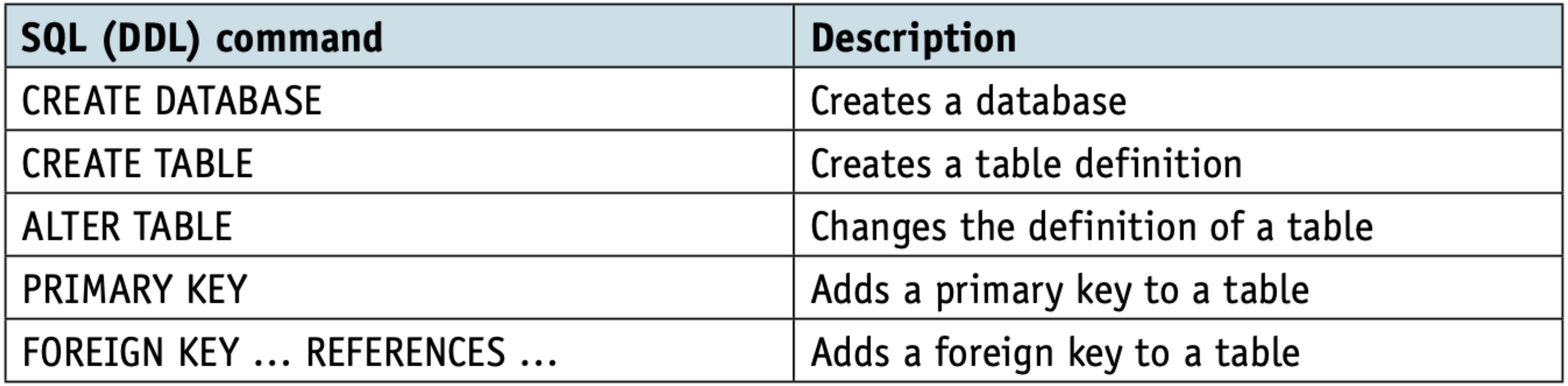
DDL and DML

- **Data Definition Language:** is used by DBMS to create, modify and remove the data structures that form a relational database.
- **Data Manipulation Language** is used to add, modify, delete and retrieve the data stored in a relational database.
- DDL and DML statements are written in a script that is similar to a computer program.
- DBMSs use structured query language (SQL) for both data definition and data manipulation.

SQL commands and scripts

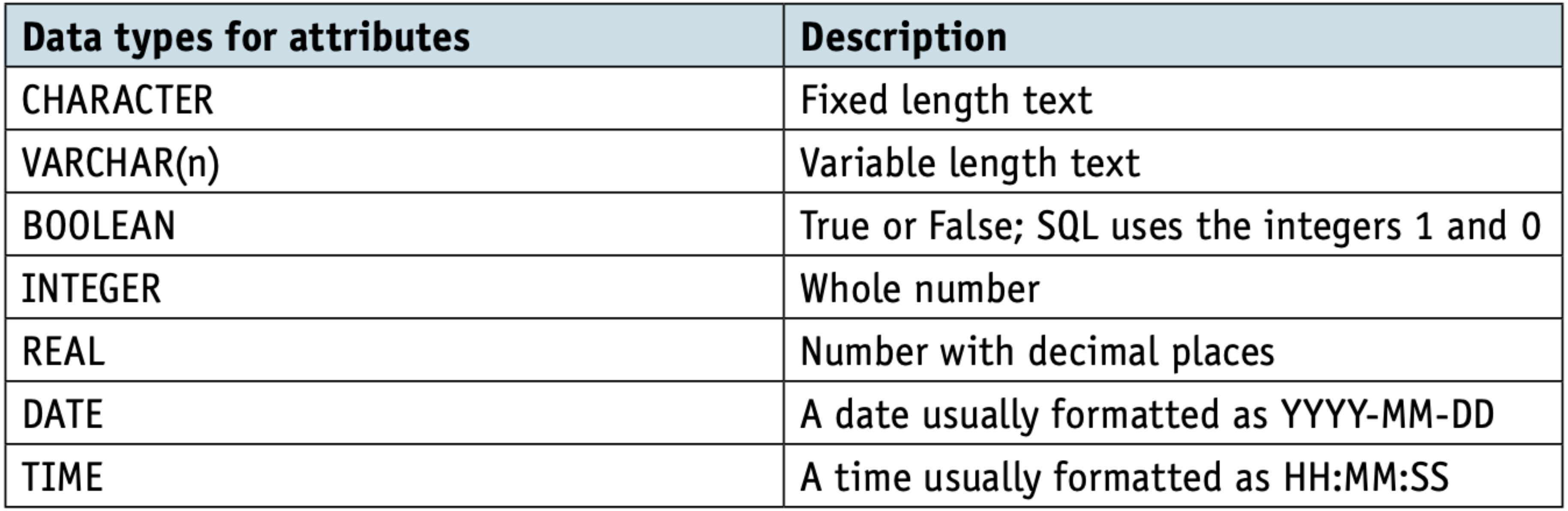
- SQL was developed in the 1970s and since then it has been adopted as an industry standard.
- There are many applications that allow you to write SQL commands, such as MySQL and SQLite
- SQL applications may differ slightly, so it is necessary to check it before using

SQL DDL commands



SQL (DDL) command	Description
CREATE DATABASE	Creates a database
CREATE TABLE	Creates a table definition
ALTER TABLE	Changes the definition of a table
PRIMARY KEY	Adds a primary key to a table
FOREIGN KEY … REFERENCES …	Adds a foreign key to a table

SQL DATA TYPES



Data types for attributes	Description
CHARACTER	Fixed length text
VARCHAR(n)	Variable length text
BOOLEAN	True or False; SQL uses the integers 1 and 0
INTEGER	Whole number
REAL	Number with decimal places
DATE	A date usually formatted as YYYY-MM-DD
TIME	A time usually formatted as HH:MM:SS

SQL DDL commands

Syntax:

- **CREATE database** Database_Name;
- **CREATE TABLE** table_name

```
(
column_Name1 data_type ( size of the column ) ,
column_Name2 data_type ( size of the column) ,
column_NameN data_type ( size of the column )
) ;
```

Example

- **Create Database** Books;
- **CREATE TABLE** Student

```
(    Roll_No.     Int ,
     First_Name    Varchar (20) ,
     Last_Name    Varchar (20) ,
     Age    Int ,
     Marks    Int
) ;
```

SQL DDL commands

Syntax:

```
INSERT INTO  table_name

(column1, column2, column3)

Values  (  value1, value2, value3) ;
```

Example

```
insert into Student

(Roll_No., First_Name, Last_Name, Age, Marks )

Values (     1, ”Tue Anh”, “Nguyen”,  16, 6) ;
```

SQL DDL commands

Syntax:

```
ALTER TABLE table_name
<SQL commands to change the table>
```

```
USE LibraryDB
ALTER TABLE Books
ADD ISBN INT NOT NULL;
```

```
USE LibraryDB
ALTER TABLE Books
ALTER COLUMN ISBN VARCHAR(50);
OR
MODIFY COLUMN ISBN VARCHAR(50);
```

SQL DDL commands

Syntax:

PRIMARY KEY (<litst of attributes>)

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID,LastName)
);
```

SQL DDL commands

Syntax:

```
FOREIGN KEY (<attr_name>) REFERENCES <table_name>(PK_name)
```

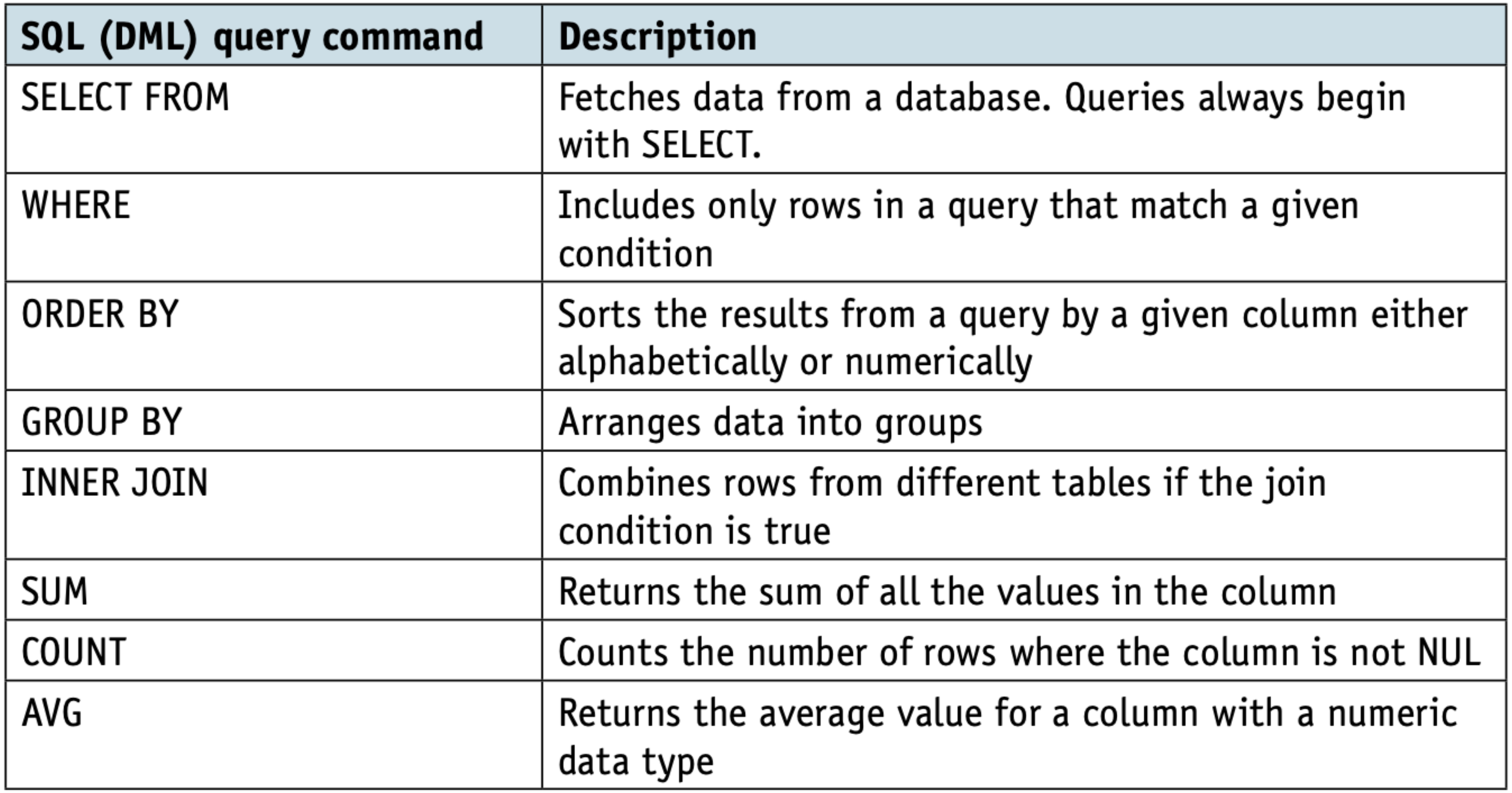
Orders table already exists

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Person
s(PersonID);
```

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

SQL DML commands

Query commands



SQL (DML) query command	Description
SELECT FROM	Fetches data from a database. Queries always begin with SELECT.
WHERE	Includes only rows in a query that match a given condition
ORDER BY	Sorts the results from a query by a given column either alphabetically or numerically
GROUP BY	Arranges data into groups
INNER JOIN	Combines rows from different tables if the join condition is true
SUM	Returns the sum of all the values in the column
COUNT	Counts the number of rows where the column is not NUL
AVG	Returns the average value for a column with a numeric data type

QUERY COMMAND - SELECT

Syntax:

```sql
SELECT column1, column2,… FROM table_name;
or
SELECT * FROM table_name;
```

```sql
SELECT CustomerName, City FROM Customers;
```

QUERY COMMAND - SELECT

Syntax:

```
SELECT column1, column2,… FROM table_name;
or
SELECT * FROM table_name;
```

```
SELECT CustomerName, City FROM Customers;

SELECT * FROM Customers;
```

QUERY COMMAND - SELECT

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   *
FROM     Product
WHERE    category=‘Gadgets’
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

“selection”

QUERY COMMAND - SELECT

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

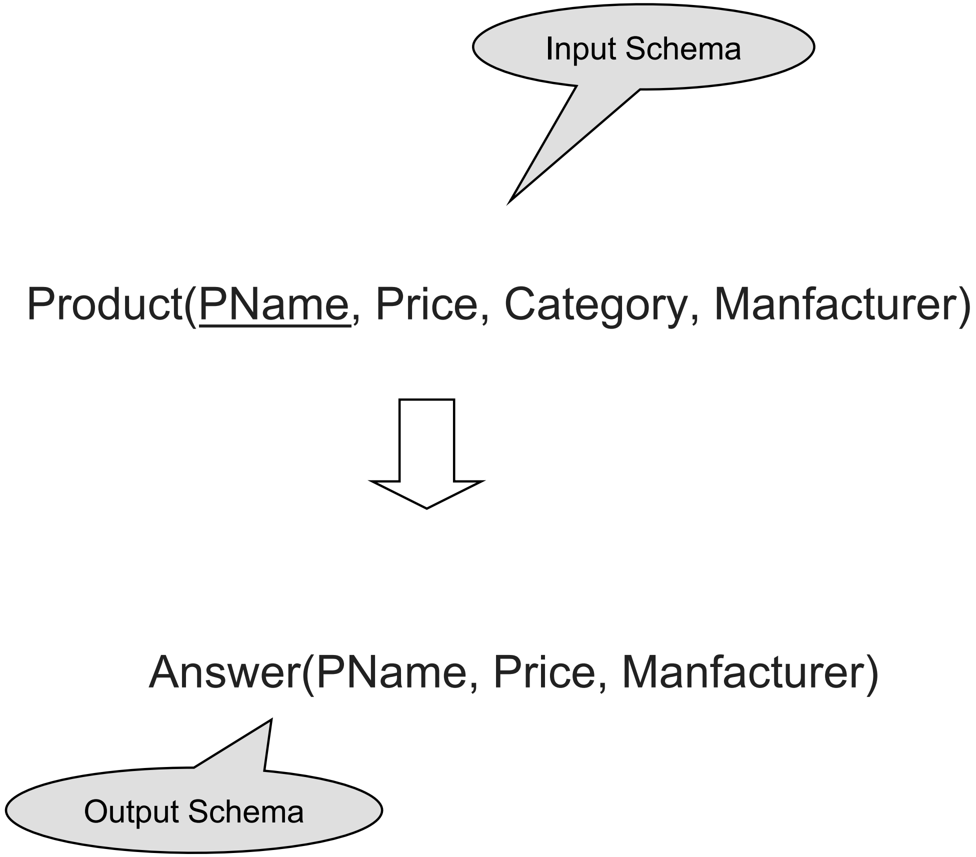
```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

“selection” and
“projection”

QUERY COMMAND - SELECT

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```



Input Schema

Product(PName, Price, Category, Manfacturer)

Answer(PName, Price, Manfacturer)

Output Schema

QUERY COMMAND - SELECT

- Case insensitive:
  - Same: SELECT Select select
  - Same: Product   product
  - Different: ‘Seattle’ ‘seattle’
- Constants:
  - ‘abc’ - yes
  - “abc” - no

QUERY COMMAND - The LIKE operator

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
  - % = any sequence of characters
  - _ = any single character

```
SELECT   *
FROM     Products
WHERE    PName LIKE ‘%gizmo%’
```

QUERY COMMAND - Eliminating Duplicates

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   DISTINCT category
FROM     Product
```

Category
Gadgets
Photography
Household

Compare to:

```
SELECT   category
FROM     Product
```

Category
Gadgets
Gadgets
Photography
Household

QUERY COMMAND - Ordering the Results

- Ties are broken by the second attribute on the ORDER BY list, etc.
- Ordering is ascending, unless you specify the DESC keyword.

```
SELECT   pname, price, manufacturer
FROM     Product
WHERE    category=‘gizmo’ AND price > 50
ORDER BY  price, pname
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   DISTINCT category
FROM     Product
ORDER BY category
```

⇨ ?

```
SELECT   Category
FROM     Product
ORDER BY  PName
```

⇨ ?

```
SELECT   DISTINCT category
FROM     Product
ORDER BY PName
```

⇨ ?

QUERY COMMAND - Keys and Foreign Keys

Company

Key

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Foreign key

QUERY COMMAND - Joins

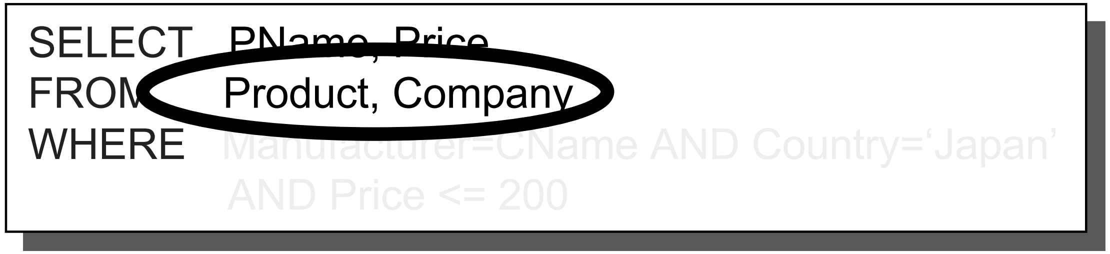
Product (pname, price, category, manufacturer)

Company (cname, stockPrice, country)

- Find all products under $200 manufactured in Japan; return their names and prices.

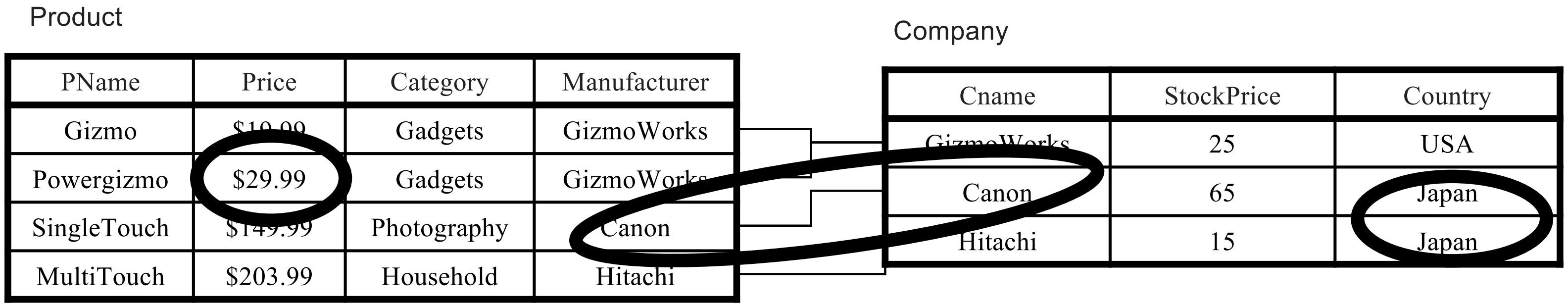
```
SELECT Pname, Price
FROM Product INNER JOIN Company ON
Product.manufacturer = Company.cname
```

Join
between Product
and Company



```
SELECT   PName, Price
FROM     Product, Company
WHERE    Manufacturer=CName AND Country='Japan'
         AND Price <= 200
```

QUERY COMMAND - Joins



Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

Cname	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT   PName, Price
FROM     Product, Company
WHERE    Manufacturer=CName AND Country=‘Japan’
         AND Price <= 200
```

PName	Price
SingleTouch	$149.99