Introduction To Java

LESSON 6

Learning Outlines : Lesson 6

- Database
- MySql
- JDBC

Database

What is the Database (DB)?

Database is any collection of related information.

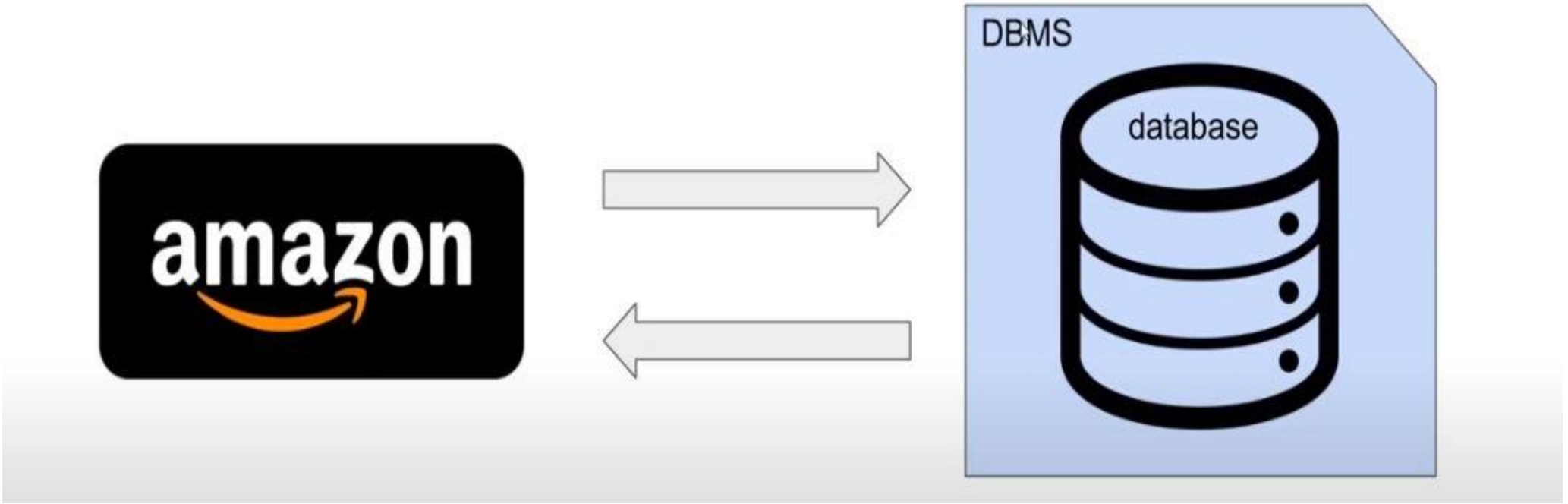
- My Phone book
- Shopping List
- Todo List
- List of students in Conrner Stone College

Database Management System (DBMS)

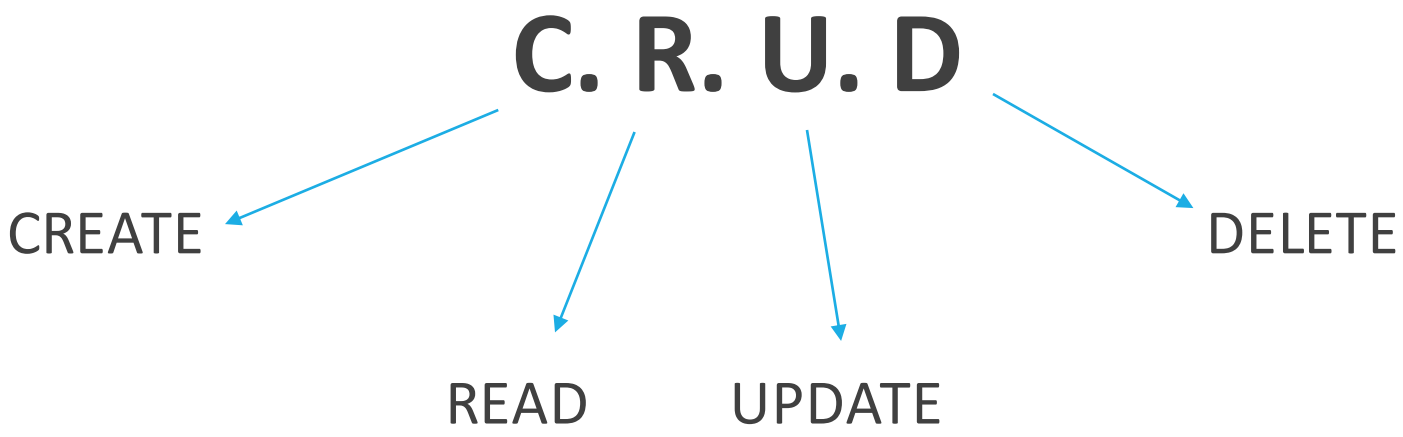
A special software program that help user to create and maintain the database

- Makes it easy to manage large amounts of information
- Handle Security
- Backups
- Importing / exporting data
- Interacts with software application

Amazon.com Database Diagram



Function Of Database



Two Types of Database

Relational Database (SQL)

- Organize data into one or more tables
- Each table has columns and rows
- A unique key identifies each rows

Non- Relational (NoSQL)

- Organize data in anything but a traditional table
- Flexible Tables

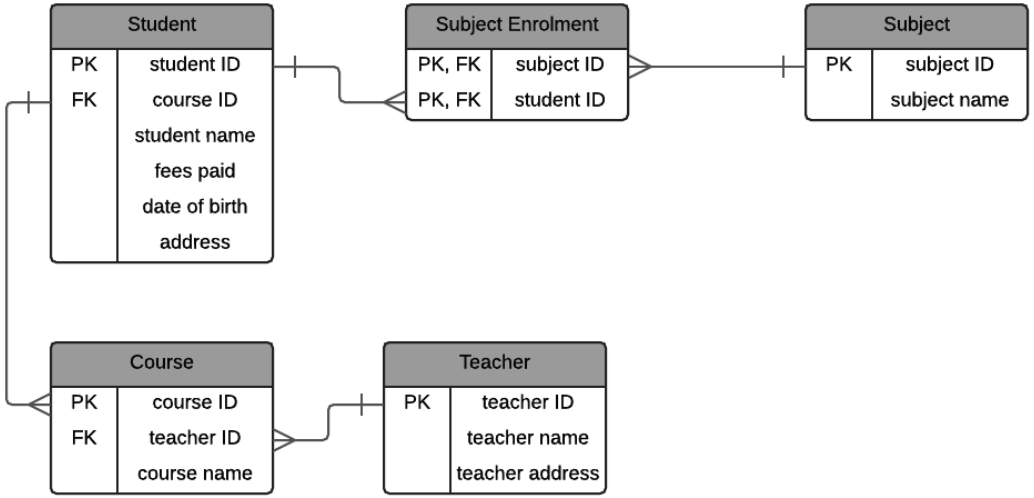
SQL

- Structured Query Language (SQL) is the language for defining tables and integrity constraints, and for accessing and manipulating data.
- SQL (pronounced “S-Q-L” or “sequel”) is the universal language for accessing relational database systems.
- SQL can be used on MySQL, Oracle, SQL Server Management Studio, MS Access, or any other relational database system.

Core Concept of Relational Database

- Entities / Tables: A logical grouping of data in rows and columns
- Attribute / Column: A characteristic that describes data like id, name
- Tuple / Row: A single row of a table, which contains a single record
- Relation: An association between entities
- Schema: A graphical representation of entities and their relations

Schema



Table

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	English
4	Jack	Biology
5	Mike	Comp. Sci

```java
public class Student {
    private int studentId;
    private String name;
    private String major;

    public Student(int studentId, String name, String major) {
        this.studentId = studentId;
        this.name = name;
        this.major = major;
    }

}
```

```java
public class ApplicationDriver {

    public static void main(String[] args) {
        Student student1 = new Student(1, "Jack", "Biology");
        Student student2 = new Student(2, "Kate", "Sociology");
        Student student3 = new Student(3, "Claire", "English");

    }

}
```

Primary Key

A primary key is a column that uniquely identifies each row in a table.

- The primary key must not be null.
- Each primary key must be unique.
- Only one column can be designed to store the primary key per table
- Primary key must be in minimal form

Foreign Key

A foreign key is an attribute in one table that references the primary key of another Table.

- Foreign key can be null.
- Foreign key defines how table are related to each other.
- Each table may have multiple foreign key

Employee

emp_id	first_name	last_name	birth_date	sex	salary	branch_id
100	Jan	Levinson	1961-05-11	F	110,000	1
101	Michael	Scott	1964-03-15	M	75,000	2
102	Josh	Porter	1969-09-05	M	78,000	3
103	Angela	Martin	1971-06-25	F	63,000	2
104	Andy	Bernard	1973-07-22	M	65,000	3

Branch

branch_id	branch_name	mgr_id
2	Scranton	101
3	Stamford	102
1	Corporate	108

Naming

- Entity names are nouns that describe the table.
- Each word in entity name starts with an upper case letter eg. BankAccount, Employee
- Column names usually start with a lowercase letter. er. Column names comprised of two or more words may be separated with underscore eg. first_name or student_id

Design the database for a college in which students can enroll in only one course for a semester and each course is taught by an instructor.

Students :

student_id	student_name	course_id
1	John	1
2	Jack	1
3	Jane	2

Instructor:

instructor_id	Instructor_name	Course_id
1	Allen	1
2	Michael	2

Course:

course_id	course_name	instructor_id
1	Java	1
2	HTML & CSS	2

Student can have only one course.

Each course has many students. ------> one to many

Each course has only one instructor.

Each instructor can have many courses. -------> one to many

Each student has only one instructor.

Each instructor can have many students -------> one to many

Database Diagram

The data diagram is a graphical representation of entities, attributes, and their relationships between entities. The diagram is usually drawn with one of the following style :

- Entity Relationship Diagram (ERD)
- Class Diagram (UML)

Why Do We Need Diagram?

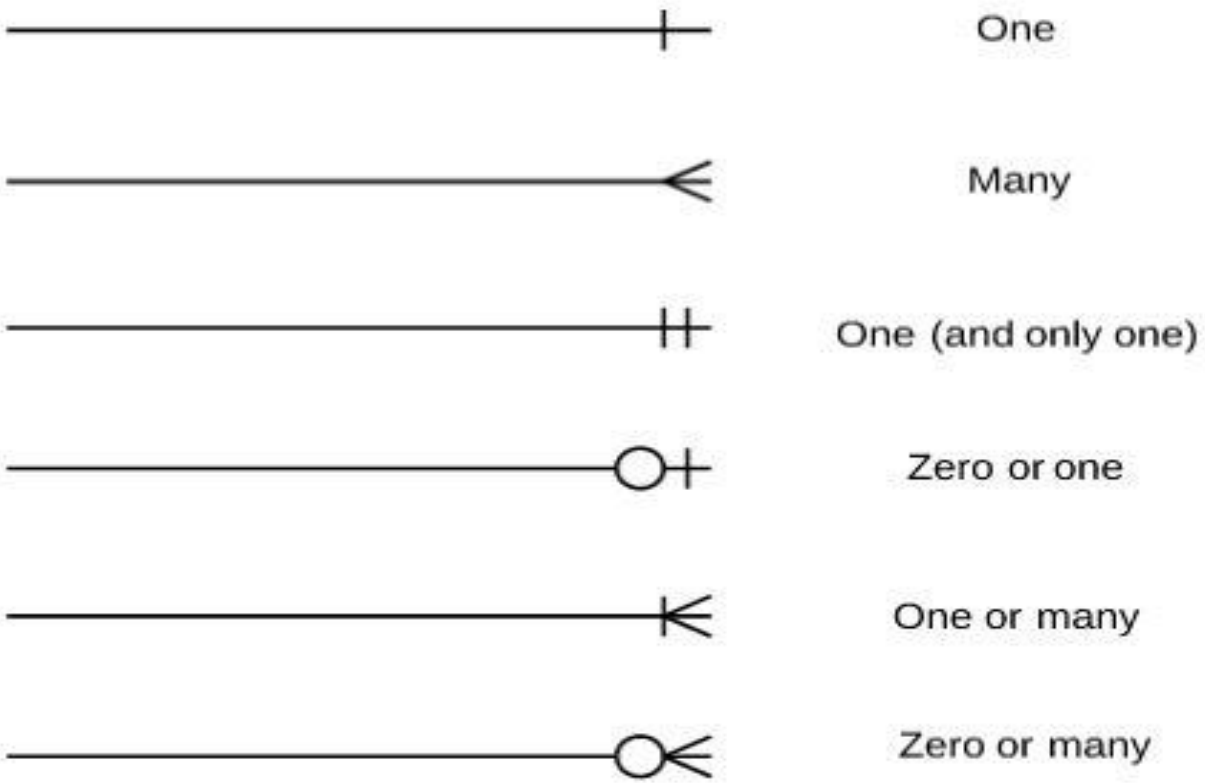
- Agree on structure of the database before deciding on a particular implementation.
- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs

Relations

Relationships are the glue that holds the tables together. They are used to connect related information between tables. Type of relationships are :

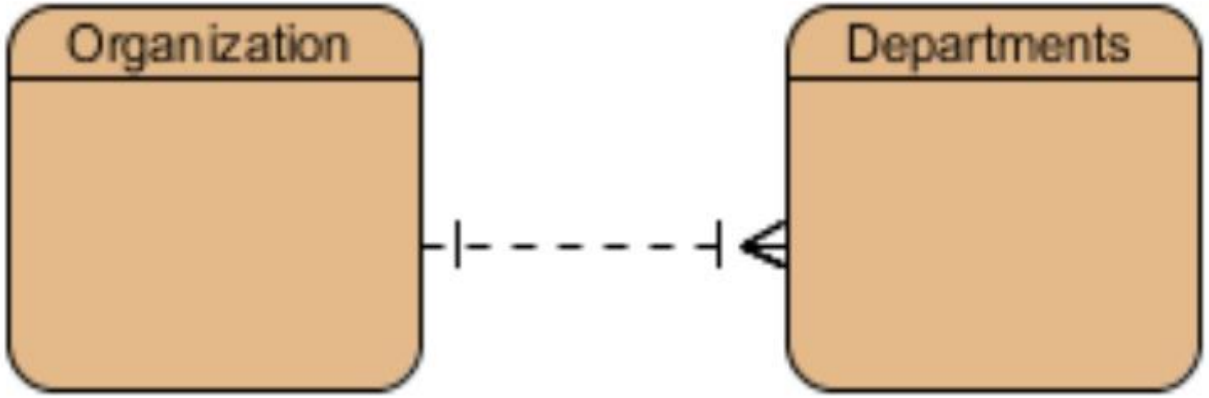
- One to Many (1:M)
- One to One (1:1)
- Many to Many (M:N)

Relationship Notation



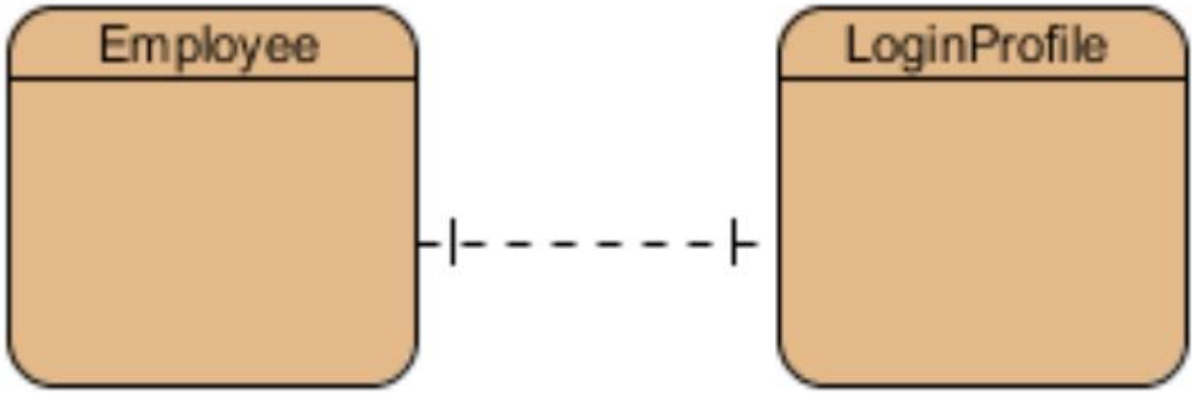
One to Many

- Each organization has many departments



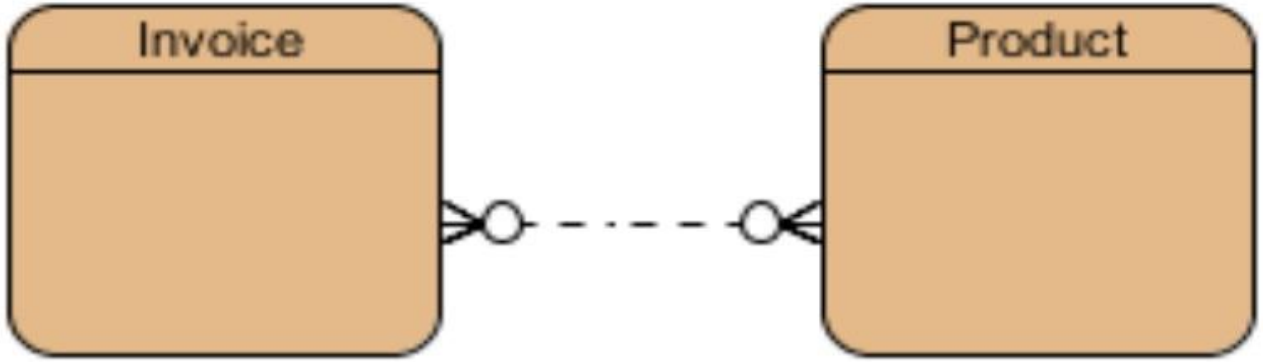
One to One

- Each employee has exactly one login profile. Each login profile belongs to exactly one employee



Many to Many

- product can be on zero or many invoices. An invoice can have zero or many products.



Imagin the website in which patients can book an appointment for different doctors

- List of Patients
- List of Doctors
- List of Appointments
- One doctor can have many patients
- One patient can have many doctors — the relation is many to many
- One doctor can have many appointments
- Each appointment belongs to one doctor. — the relation is many to one
- One patient can have many appointments
- Each appointment belongs to one patient — the relation is many to one

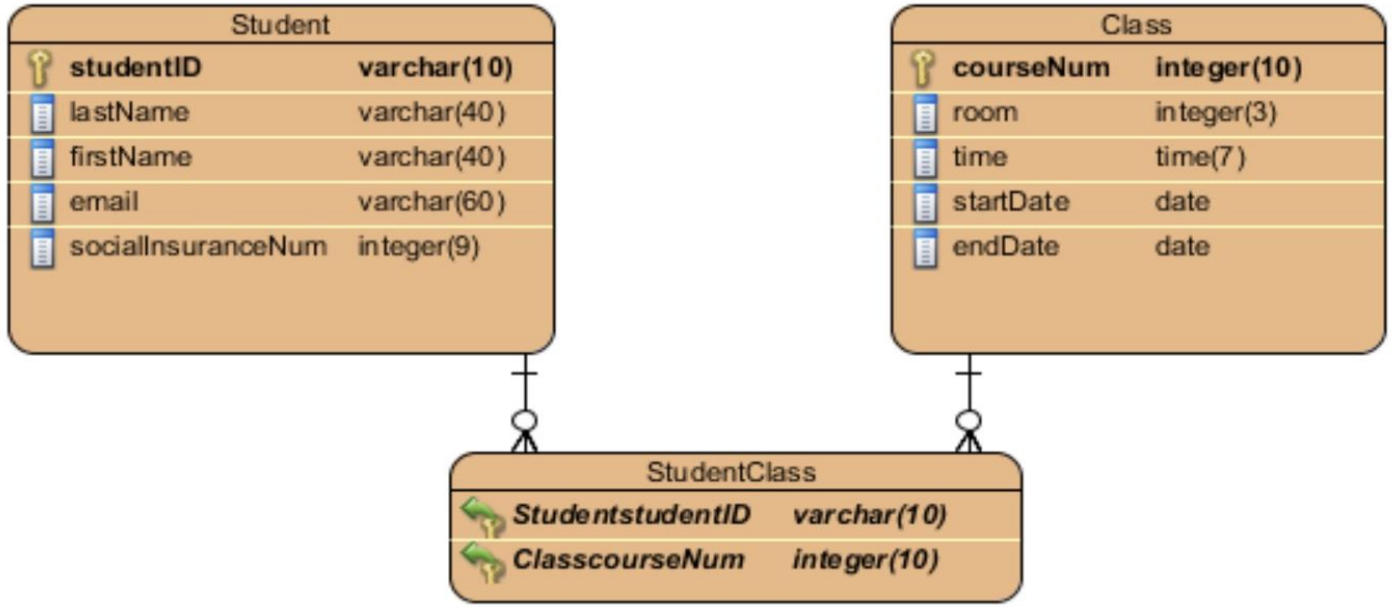
Fan Traps

- Relational database systems usually don't allow us to implement a direct many-to-many relationship between two tables. We couldn’t do this:

Student ID	Class ID	Student Name
1	3, 5, 9	John
2	1, 4, 5, 9	Debbie

Bridge Table

- A Bridge Table is a table that sits between the two other tables of a many-to-many relationship. Its purpose is to store a record for each of the combinations of these other two tables.



Imagine one company has many departments. And each department has many employees.

Company	each compeny can have many departments
Department	each department can have many employees.
Employee	

Student:

Student ID	Student name
1	John
2	Debbie

Class:

Class ID	Class name
1	English
2	Maths
3	Spanish
4	Biology
5	Science
6	Programming
7	Law
8	Commerce
9	Physical Education

Student ID	Class ID
1	3
1	5
1	9
2	1
2	4
2	5
2	9

Online Tools For Creating an ERD

- Lucidchart
- Visual Paradigm
- Draw.io

Imagine the relation between customer and order

List of customers. ----> customer_id (Primary key)

List of Orders. -------> order_id(Primary key)

One customer can have many orders

One order belongs to only one. The relation is many orders to one customer

Imagine database of one web creating invoice

List of Products ----------> Primary Key. (product_id).

List of Invoices ----------> Primary Key (invoice_id)

One product can belong to many invoices

One invoice can have many products.

the relation between product and invoice is many to many.

We need Bridge table (ProductInvoice) --------> Primary Key (product_invoice_id)

Remember the bridge table has two foreign keys. ---------> product_id

invoice_id

Customer

CustID	CustName	AccntNo.
100	Joe Smith	010839
101	Andy Blake	111248
102	Sue Brown	061544

BookOrders

OrderID	Title	CustID	Price
1001	The Dark Tower	102	12.00
1002	Incubus Dreams	101	19.99
1003	Song of Susannah	102	23.00
1004	The Time Traveler's Wife	100	21.00
1005	The Dark Tower	101	12.00
1006	Tanequil	102	15.00
1007	Song of Susannah	101	23.00

Imagine Amazon website. In Amazon website, we can create a USER(account), we can search for PRODUCT, we can write REVIEW for product.

USER --------> many to many (Bridge table)

USER to Product

PRODUCT. ---------> one to many Product to Review

REVIEW

MySql

SQL Data Types

- The following data types exist in SQL Server to regulate how each column is stored, to provide storage efficiency

- Don’t memorize it. Just remember you should choose suitable data types when designing your database. You can refer back to this or you can find a similar reference on the web whenever needed.

Data Types	Description
int	Integer data from -2^31 through 2^31 - 1
bit	Integer data with either a 1 or 0 value
decimal	Fixed precision and scale numeric data from -10^38 +1 through 10^38 -1
money	Monetary data values from -2^63 through 2^63 - 1
float	Floating precision number data from -1.79E + 308 through 1.79E + 308
Datetime	Date and time data from January 1, 1753, through December 31, 9999, with an accuracy of 3.33 milliseconds
varchar	Variable-length data with a maximum of 8,000 characters
Nvarchar	Variable-length Unicode data with a maximum length of 4,000 characters
Uniqueidentifier	A globally unique identifier
varbinary	Variable-length binary data with a maximum length of 8,000 bytes
cursor	A reference to a cursor
table	A special data type used to store a result set for later processing

Create Table

- CREATE TABLE Inventory (

```
    `product_ID`        int(8),
    `product_name`      varchar(20),
    `vendor`            varchar(20),
    `quantity`          int(8),
    `price`             decimal(7,2),
    `stock_date`        date
);
```

Insert Data

- To add data to a table, use the INSERT command. The command usually starts with the INSERT INTO clause followed by the table name. Next the VALUES clause assigns the data for each column according to the pre-defined column sequence.

```
INSERT INTO `Inventory` VALUES (1, 'Ginger Tea', 'Yogi', 54, 5.48, '2020-03-05');
INSERT INTO `Inventory` VALUES (2, 'Green Tea', 'Stash', 48, 4.48, '2020-03-05');
INSERT INTO `Inventory` VALUES (3, 'Lemon Tea', 'Yogi', 45, 5.48, '2020-03-02');
INSERT INTO `Inventory` VALUES (4, 'Herbal Tea', 'Traditional Medicine', 60, 3.98, '2020-02-25');
INSERT INTO `Inventory` VALUES (5, 'Ginger Tea', 'Traditional Medicine', 50, 3.98, '2020-02-25');
```

Insert Data

- When adding data, you can also specify the columns with the following syntax:

```
INSERT INTO `Inventory` (`product_ID`, `product_name`, `quantity`, `price`) VALUES (6, 'Green Tea', 26, 4.35);
INSERT INTO `Inventory` (`product_ID`, `product_name`, `quantity`, `price`, `stock_date`) VALUES (7, 'Ginger Tea', 26, 4.35, '2020-03-05');
```

View Data

- To view data, use the SELECT command. SELECT * reads all columns from the

```
SELECT * FROM Inventory;
```

- You can also specify the columns in your query if preferred.

```
SELECT `product_name` FROM Inventory;
```

- If you need more than one column in your query, separate each column with a comma

```
SELECT `product_name`, `quantity` FROM Inventory;
```

Order Data

- The ORDER BY command beside an attribute forces the SELECT statement to retrieve data in a sorted sequence by the designated attribute.

  - Ascending Order:

```
SELECT * FROM inventory ORDER BY `quantity`;
```

  - Descending Order:

```
SELECT * FROM inventory ORDER BY `quantity` DESC;
```

Filter Data

- Adding a WHERE filter will search for rows that match a specific condition or set of conditions.

Operator	Example	Description
=	SELECT * FROM inventory WHERE `quantity`= 26;	Equals
>	SELECT * FROM inventory WHERE `quantity`> 40;	Greater than
<	SELECT * FROM inventory WHERE `quantity`< 40;	Less than
>=	SELECT * FROM inventory WHERE `quantity`>= 45;	Greater than or equal
<=	SELECT * FROM inventory WHERE `quantity`<= 45;	Less than or equal
AND	SELECT*FROM Inventory WHERE quantity > 45 AND vendor ='Yogi';	Both conditions must be true
OR	SELECT*FROM Inventory WHERE quantity > 45 OR vendor ='Yogi';	One condition must be true

Filter Data

- It is possible to check for NULL values in the WHERE clause.

```
SELECT*FROM Inventory WHERE vendor IS NULL;
SELECT*FROM Inventory WHERE vendor IS NOT NULL;
```

Update Data

- To update an existing record you can use the UPDATE and SET command

```
UPDATE Inventory SET quantity = 0 WHERE vendor IS NULL;
UPDATE Inventory SET stock_date = '2020-03-25'  WHERE stock_date='2020-03-05'
```

Drop Table

- To remove a table from your database and delete all data within it, the DROP command is used.

```
DROP TABLE Inventory;
```

Tips For Creating Table

- You can auto-generate the primary key if it is an integer with the AUTO_INCREMENT command.
- You can indicate whether a column should be NULL or NOT NULL. NULL indicates that an attribute may be left empty when other data is inserted into a row of the table.
- Attributes can be assigned DEFAULT values if no data is inserted into that column when creating a new row.

Example

```
CREATE TABLE Employee(
   employee_ID    INT PRIMARY KEY AUTO_INCREMENT  ,
   first_name      VARCHAR(30) NULL ,
   last_name       VARCHAR(30) NOT NULL
);

INSERT INTO Employee(first_name, last_name) VALUES('John', 'Doe');
INSERT INTO Employee(first_name) VALUES('Jane');
INSERT INTO Employee(last_name) VALUES ('Davis');
```

Example

```
CREATE TABLE Employee(
    employee_ID   INT PRIMARY KEY AUTO_INCREMENT ,
    first_name       VARCHAR(30)  DEFAULT 'Jane',
    last_name        VARCHAR(30) NOT NULL
);

INSERT INTO Employee(first_name, last_name) VALUES('John', 'Doe');
INSERT INTO Employee(last_name) VALUES ('Davis');
```

Add Foreign Key

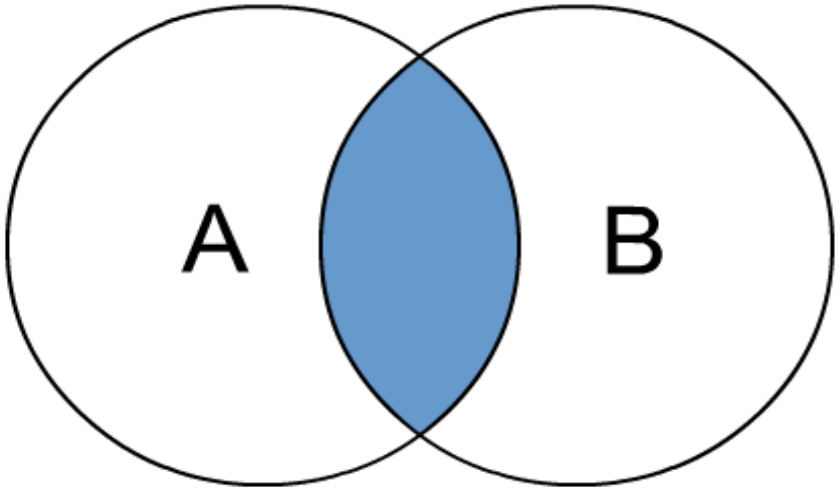
- You can only add a foreign key though if the table being referenced already exists. Also, the foreign key must reference part of the primary key in the related table.
- Here is an example:

Example

```
CREATE TABLE Department(
   dept_ID      INT PRIMARY KEY AUTO_INCREMENT,
   dept_name    VARCHAR(20)
 );
CREATE TABLE Employee(
   employee_ID   INT PRIMARY KEY AUTO_INCREMENT ,
   dept_ID         INT,
   first_name    VARCHAR(30) ,
   last_name     VARCHAR(30),
   FOREIGN KEY(dept_ID) REFERENCES Department(dept_ID)
);
```

Inner (Equal) Joins

- Inner joins return all fields where there is a match. The diagram below shows the selected rows where the related column matches.



Inner Join

- Here is an example of Inner Join:

```
SELECT Student.first_name, Student.last_name, Course.course_name
FROM Student, Course
WHERE Student.course_id = Course.course_id
```

Database Connection To Java

JDBC

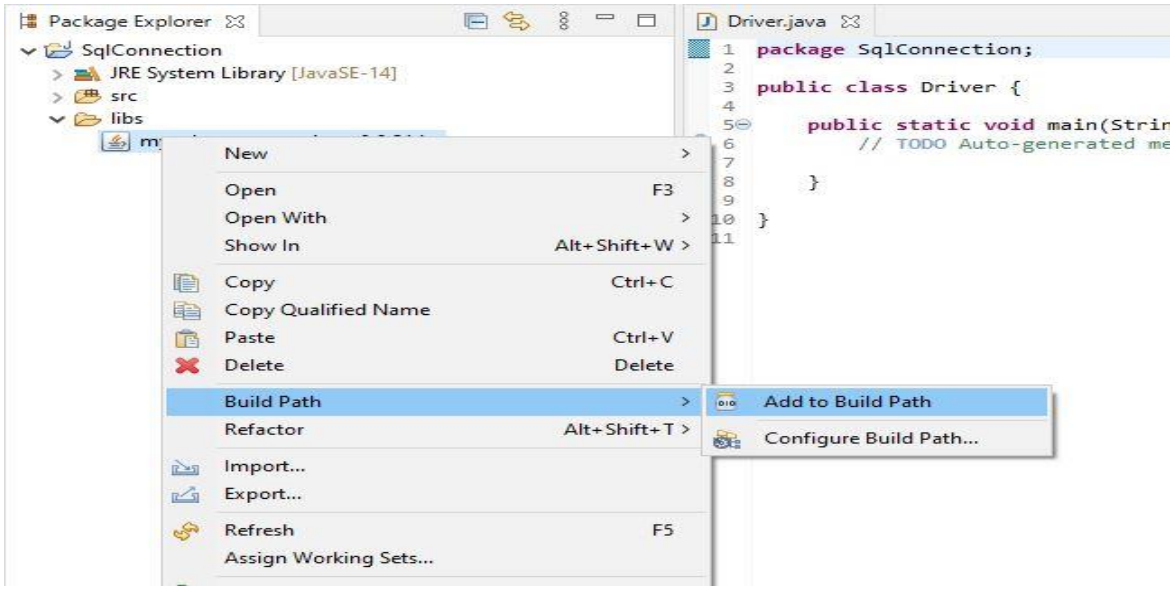
- JDBC is Java-Based API for connection to relational Database
- JDBC is originally known as the "Jave Database Connectivity"
- JDBC is commonly used in
  - Web-based applications hosted by Java Enterprise Edition (JEE) servers
  - Desktop applications working with local or remote database
  - Android applications

Java Driver

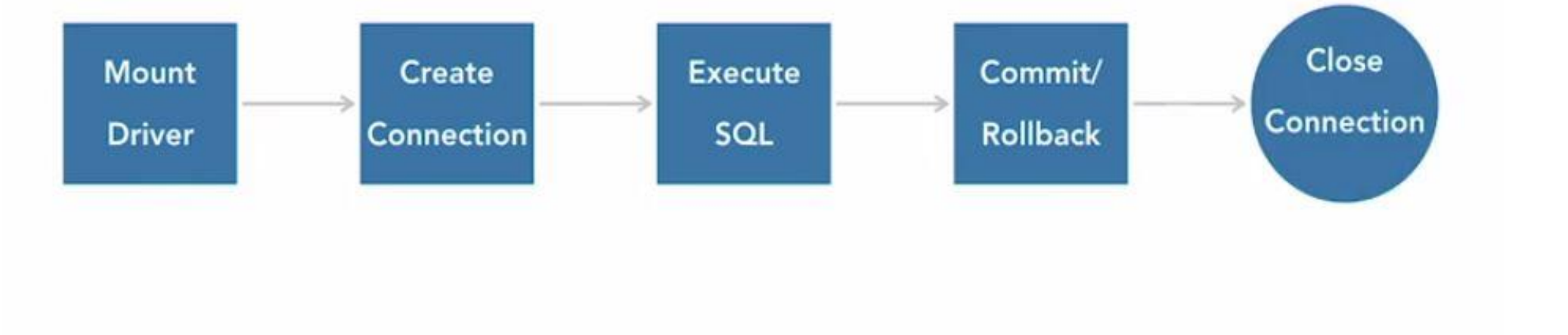
- JDBC Drives are software libraries that communicate between Java application and a database
- A driver library package contains specific implementations of these interfaces:
  - Connection
  - ResultSet
  - Statement
  - ...
- Go to https://dev.mysql.com/downloads/ and Download JConnector

Add MySql Driver To The Project

- Right click on the project in the Package Explorer and create new folder.
- Paste the jar file downloaded in the new folder
- Add the jar file to the project build path



Basic Flow



- For some queries, we are getting data back so after the execution, we are going to unpack the result set.

Connection

- DriverManager: A class to interact with driver for creating connections
- Connection: A class that the developer interacts with that manages the actual communication between the client and the server

Connection

```java
package SqlConnection;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Driver {

    private static final String USERNAME = "root";
    private static final String PASSWORD = "your password";
    private static final String CONN_STRING =
            "jdbc:mysql://localhost:3306/db name";
    public static void main(String[] args) throws SQLException {
        Connection conn = null;

        //1. Get a connection to database
        try {
            conn = DriverManager.getConnection(CONN_STRING, USERNAME, PASSWORD);
        } catch (SQLException e) {
            System.out.println(e);
        } finally {
            if (conn != null) {
                conn.close();
            }
        }

    }

}
```

Executions

- Statement: The representation SQL to be executed against the database
- PreparedStatement: An extension of statement that is used for precompiled statements (with inputs)
- ResultSet: The response from the database in a logical tabular form

Executions

```
        "jdbc:mysql://localhost:3306/db name";
public static void main(String[] args) throws SQLException {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;

    try {
        //1. Get a connection to database
        conn = DriverManager.getConnection(CONN_STRING, USERNAME, PASSWORD);

        //2. Create a statement
        stmt = conn.createStatement();

        //3. Execute SQL query
        rs = stmt.executeQuery("SELECT * FROM student");

        //4. Process the result set

    } catch (SQLException e) {
        System.out.println(e);
    } finally {
        if (rs != null) {
            rs.close();
        }
        if (stmt != null) {
            stmt.close();
        }
        if (conn != null) {
            conn.close();
        }
    }

}
```