Lecture 7 - Java Database

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

CSC-1004: Computational Laboratory Using Java
Course Page: [Click]

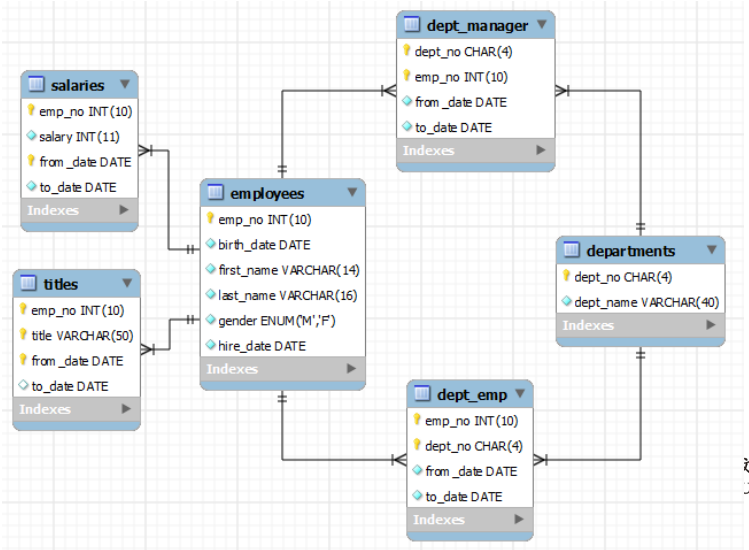
Outline

- Basic Structured Query Language (SQL)
- Java Database Connectivity (JDBC)

What is SQL?

- **SQL (Structured Query Language)** performs operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.
- **SQL is not a database but a query language**. To use it, you must install a database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc.

MySQL Database Example



SQL Database

- Syntax of **Create Use and Drop** Database statement in MySQL.

```
CREATE DATABASE Database_Name;
```

```
USE database_name;
```

```
DROP DATABASE Database_Name;
```

- When this query is executed successfully, then it will show "Database created successfully". You can verify whether your database is created in SQL by:

```
SHOW DATABASE ;
```

- Syntax of **Use and Rename** statement in MySQL

```
RENAME DATABASE old_database_name TO new_database_name;
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL Table

Table is a collection of data, organized in terms of rows and columns.

- Create a table in MySQL database.

Without using a PRIMARY KEY.

```
CREATE TABLE Employee
(
EmployeeID int,
FirstName varchar(255),
LastName varchar(255),
Email varchar(255),
AddressLine varchar(255),
City varchar(255)
);
```

With a PRIMARY KEY.

```
CREATE TABLE Employee(
EmployeeID NOT NULL,
FirstName varchar(255) NOT NULL,
LastName varchar(255),
City varchar(255),
PRIMARY KEY (EmployeeID)
);
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL Table

- MySQL Data Types (other types like ENUM, SET and BLOB are less popular).

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.
LONGTEXT	It holds a string with a maximum length of 4,294,967,295 characters.

SQL Table

- MySQL Numeric Data Types (other types like DOUBLE, DECIMAL, and BOOL are less popular).

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().

SQL Table

- MySQL Date and Time Data Types.

DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

SQL Table

- A SQL DROP TABLE statement is used to delete a table definition and all data from a table.

```
DROP TABLE table_name;
```

- The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

```
DELETE FROM table_name [WHERE condition];
```

- A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause.

```
TRUNCATE TABLE table_name;
```

SQL Insert

- Inserting data directly into a table.

```
INSERT INTO table_name (column1, column2, column3....)
VALUES (value1, value2, value3.....);
```

Let's take an example of a table that has five records within it.

```
INSERT INTO STUDENTS (ROLL_NO, NAME, AGE, CITY)
VALUES (1, ABHIRAM, 22, ALLAHABAD);
INSERT INTO STUDENTS (ROLL_NO, NAME, AGE, CITY)
VALUES (2, ALKA, 20, GHAZIABAD);
INSERT INTO STUDENTS (ROLL_NO, NAME, AGE, CITY)
VALUES (3, DISHA, 21, VARANASI);
INSERT INTO STUDENTS (ROLL_NO, NAME, AGE, CITY)
VALUES (4, ESHA, 21, DELHI);
INSERT INTO STUDENTS (ROLL_NO, NAME, AGE, CITY)
VALUES (5, MANMEET, 23, JALANDHAR);
```

SQL Insert

- Inserting data directly into a table.

```
INSERT INTO table_name (column1, column2, column3....)
VALUES (value1, value2, value3.....);
```

It will show the following table as the final result.

ROLL_NO	NAME	AGE	CITY
1	ABHIRAM	22	ALLAHABAD
2	ALKA	20	GHAZIABAD
3	DISHA	21	VARANASI
4	ESHA	21	DELHI
5	MANMEET	23	JALANDHAR

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL Insert

- Inserting data through SELECT Statement

```
INSERT INTO table_name
[(column1, column2, .... column)]
SELECT column1, column2, .... Column N
FROM table_name [WHERE condition];
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL Insert

- Inserting multiple rows into a single table in a single statement. Please check the following example

  Step 1: Select the database in which we want to create a table.

```
mysql> USE dbs;
```

  Step 2: Create a table named student in the selected database 'dbs'.

```
mysql> CREATE TABLE student(ID INT, Name VARCHAR(20), Percentage INT, Location VARCHAR(20), DateOfBirth DATE);
```

  Step 3: write a single query to insert multiple records in the student table:

```
mysql> INSERT INTO student(ID, Name, Percentage, Location, DateOfBirth) VALUES(1, "Manthan Koli", 79, "Delhi", "2003-08
20"), (2, "Dev Dixit", 75, "Pune", "1999-06-17"), (3, "Aakash Deshmukh", 87, "Mumbai", "1997-09-12"), (4, "Aaryan Jaiswal", 90, "Chennai", "2005
10-02"), (5, "Rahul Khanna", 92, "Ambala", "1996-03-04"), (6, "Pankaj Deshmukh", 67, "Kanpur", "2000-02
02"), (7, "Gaurav Kumar", 84, "Chandigarh", "1998-07-06"), (8, "Sanket Jain", 61, "Shimla", "1990-09-08"), (9, "Sahil Wagh", 90, "Kolkata", "1968
04-03"), (10, "Saurabh Singh", 54, "Kashmir", "1989-01-06");
```

SQL Insert

- Inserting multiple rows into a single table in a single statement. Please check the following example

ID	Name	Percentage	Location	DateOfBirth
1	Manthan Koli	79	Delhi	2003-08-20
2	Dev Dixit	75	Pune	1999-06-17
3	Aakash Deshmukh	87	Mumbai	1997-09-12
4	Aaryan Jaiswal	90	Chennai	2005-10-02
5	Rahul Khanna	92	Ambala	1996-03-04
6	Pankaj Deshmukh	67	Kanpur	2000-02-02
7	Gaurav Kumar	84	Chandigarh	1998-07-06
8	Sanket Jain	61	Shimla	1990-09-08
9	Sahil Wagh	90	Kolkata	1968-04-03
10	Saurabh Singh	54	Kashmir	1989-01-06

SQL Select

- SELECT Statement in SQL.

```
SELECT Column_Name_1, Column_Name_2, ....., Column_Name_N FROM Table_Name;
```

If you want to access all rows from all fields of the table, use the following SQL SELECT syntax with * asterisk sign.

SQL Select

- SELECT Statement with **WHERE** clause. The syntax is:

```
SELECT * FROM Name_of_Table WHERE [condition];
```

Example: `SELECT * FROM Employee_Details;`

Employee_Id	Emp_Name	Emp_City	Emp_Salary	Emp_Panelty
101	Anuj	Ghaziabad	25000	500
102	Tushar	Lucknow	29000	1000
103	Vivek	Kolkata	35000	500
104	Shivam	Goa	22000	500

SQL Select

- SELECT Statement with **WHERE** clause. The syntax is:

```
SELECT * FROM Name_of_Table WHERE [condition];
```

Example:

```
SELECT * FROM Employee_Details WHERE Emp_Panelty = 500;
```

Employee_Id	Emp_Name	Emp_City	Emp_Salary	Emp_Panelty
101	Anuj	Ghaziabad	25000	500
103	Vivek	Kolkata	35000	500
104	Shivam	Goa	22000	500

SQL Select

- SQL SELECT Statement with **GROUP** BY clause. The syntax is:

```
SELECT column_Name_1, column_Name_2, ....., column_Name_N aggregate_function_name(column_Name2)
FROM table_name GROUP BY column_Name1;
```

Example: `SELECT * FROM Cars_Details;`

Car_Number	Car_Name	Car_Amount	Car_Price
2578	Creta	3	1000000
9258	Audi	2	900000
8233	Venue	6	900000
6214	Nexon	7	1000000

SQL Select

- SQL SELECT Statement with **GROUP** BY clause. The syntax is:

```
SELECT column_Name_1, column_Name_2, ....., column_Name_N aggregate_function_name(column_Name2)
FROM table_name GROUP BY column_Name1;
```

Example:

```
SELECT COUNT (Car_Name), Car_Price FROM Cars_Details GROUP BY Car_Price;
```

Count (Car_Name)	Car_Price
2	1000000
2	900000

SQL Select

- SQL SELECT Statement with **HAVING** clause. The syntax is:

```
SELECT column_Name_1, column_Name_2, ....., column_Name_N aggregate_function_name(column_Name_2)
FROM table_name GROUP BY column_Name1 HAVING ;
```

Example: `SELECT * FROM Employee_Having;`

Employee_Id	Employee_Name	Employee_Salary	Employee_City
201	Jone	20000	Goa
202	Basant	40000	Delhi
203	Rashet	80000	Jaipur
204	Anuj	20000	Goa
205	Sumit	50000	Delhi

SQL Select

- SQL SELECT Statement with **HAVING** clause. The syntax is:

```
SELECT column_Name_1, column_Name_2, ....., column_Name_N aggregate_function_name(column_Name_2)
FROM table_name GROUP BY column_Name1 HAVING ;
```

Example:

```
SELECT SUM (Employee_Salary), Employee_City FROM Employee_Having
GROUP BY Employee_City HAVING SUM(Employee_Salary)>5000;
```

SUM (Employee_Salary)	Employee_City
90000	Delhi
80000	Jaipur

SQL Select

- SELECT Statement with **ORDER BY** clause. The syntax is:

```
SELECT Column_Name_1, Column_Name_2, ....., column_Name_N FROM table_name
WHERE [Condition] ORDER BY[column_Name_1, column_Name_2, ....., column_Name_N asc | desc ];
```

Example: `SELECT * FROM Employee_Order;`

Id	FirstName	Salary	City
201	Jone	20000	Goa
202	Basant	15000	Delhi
203	Rashet	80000	Jaipur
204	Anuj	90000	Goa
205	Sumit	50000	Delhi

SQL Select

- SELECT Statement with **ORDER BY** clause. The syntax is:

```
SELECT Column_Name_1, Column_Name_2, ....., column_Name_N FROM table_name
WHERE [Condition] ORDER BY[column_Name_1, column_Name_2, ....., column_Name_N asc | desc ];
```

Example: `SELECT * FROM Employee_Order ORDER BY Emp_Salary DESC;`

Emp_Id	Emp_Name	Emp_Salary	Emp_City
204	Anuj	90000	Goa
203	Rashet	80000	Jaipur
205	Sumit	50000	Delhi
201	Jone	20000	Goa
202	Basant	15000	Delhi

SQL Key

- A column is called **primary key** that *uniquely* identifies each row in the table. When multiple columns are used as a primary key, it is known as **composite primary key**.

SQL Key

- SQL primary key for **one column**:

```
CREATE TABLE students
(
S_Id int NOT NULL,
LastName varchar (255) NOT NULL,
FirstName varchar (255),
Address varchar (255),
City varchar (255),
PRIMARY KEY (S_Id)
)
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL Key

- SQL primary key for **multiple columns**:

```
CREATE TABLE students
(
S_Id int NOT NULL,
LastName varchar (255) NOT NULL,
FirstName varchar (255),
Address varchar (255),
City varchar (255),
CONSTRAINT pk_StudentID PRIMARY KEY (S_Id, LastName)
)
```

SQL Key

- **Alternate key** is a secondary key it can be simple to understand by an example. Let's take an example of a student it can contain NAME, ROLL NO., ID, and CLASS. Here ROLL NO. is the primary key and the rest of the columns like NAME, ID, and CLASS are alternate keys.

SQL View

- SQL provides the concept of **VIEW**, which hides the complexity of the data and restricts unnecessary access to the database. It permits the users to access only a particular column rather than the whole data of the table.
- The View in the SQL is considered as the **virtual table**, which depends on the result-set of the predefined SQL statement.
- Like the SQL tables, Views also store data in rows and columns, but the rows **do not have any physical existence** in the database.

SQL View

- Create View from **Single** Table. The Syntax is:

```
CREATE VIEW View_Name AS
SELECT Column_Name1, Column_Name2, ....., Column_NameN
FROM Table_Name
WHERE condition;
```

Example: Let's consider the following Student_Details table

Student_ID	Stu_Name	Stu_Subject	Stu_Marks
1001	Akhil	Math	85
1002	Balram	Science	78
1003	Bheem	Math	87
1004	Chetan	English	95
1005	Diksha	Hindi	99
1006	Raman	Computer	90
1007	Sheetal	Science	68

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL View

- Create View from **Single** Table. The Syntax is:

```
CREATE VIEW View_Name AS
SELECT Column_Name1, Column_Name2, ....., Column_NameN
FROM Table_Name
WHERE condition;
```

Example: Let's create a view from the table and check the view.

```
CREATE VIEW Student_View AS
SELECT Student_ID, Stu_Subject, Stu_Marks
FROM Student_Details
WHERE Stu_Marks > 85;
```

```
Select * FROM Student_View;
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL View

- Create View from **Single** Table. The Syntax is:

```
CREATE VIEW View_Name AS
SELECT Column_Name1, Column_Name2, ....., Column_NameN
FROM Table_Name
WHERE condition;
```

Example: The result is:

Student_ID	Stu_Subject	Stu_Marks
1001	Math	85
1003	Math	87
1004	English	95
1005	Hindi	99
1006	Computer	90

SQL View

- Create View from **Multiple** Tables. The Syntax is:

```
REATE VIEW View_Name AS
SELECT Table_Name1.Column_Name1, Table_Name1.Column_Name2, Table_Name2.Column_Name2, .....,
FROM Table_Name1, Table_Name2, ....., Table_NameN
WHERE condition;
```

Example: Let's consider the Student_Details and Teacher_Details table.

Student_ID	Stu_Name	Stu_Subject	Stu_Marks
1001	Akhil	Math	85
1002	Balram	Science	78
1003	Bheem	Math	87
1004	Chetan	English	95
1005	Diksha	Hindi	99
1006	Raman	Computer	90
1007	Sheetal	Science	68

Teacher_ID	Teacher_Name	Teacher_Subject	Teacher_City
2001	Arun	Math	Gurgaon
2002	Manoj	Science	Delhi
2003	Reena	SST	Noida
2004	Parul	English	Gurgaon
2005	Nishi	Hindi	Noida
2006	Anuj	Computer	Delhi
2007	Ram	Physical Education	Delhi

)

SQL View

- Create View from **Multiple** Tables. The Syntax is:

```
REATE VIEW View_Name AS
SELECT Table_Name1.Column_Name1, Table_Name1.Column_Name2, Table_Name2.Column_Name2, .....,
FROM Table_Name1, Table_Name2, ....., Table_NameN
WHERE condition;
```

Example: Let's create a view from both tables and check the view.

```
CREATE VIEW Student_Teacher_View AS
SELECT Student_Details.Student_ID, Student_Details.Stu_Name,
Teacher_Details.Teacher_ID, Teacher_Details.Teacher_Subject
FROM Student_Details, Teacher_Details
WHERE Student_Details.Stu_Subject = Teacher_Details.Teacher_Subject;
Select * FROM Student_Teacher_View;
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SQL View

- Create View from **Multiple** Tables. The Syntax is:

```
REATE VIEW View_Name AS
SELECT Table_Name1.Column_Name1, Table_Name1.Column_Name2, Table_Name2.Column_Name2, .....,
FROM Table_Name1, Table_Name2, ....., Table_NameN
WHERE condition;
```

Example: The result is:

Student_ID	Stu_Name	Teacher_ID	Teacher_Subject
1001	Akhil	2001	Math
1002	Balram	2002	Science
1004	Chetan	2004	English
1005	Diksha	2005	Hindi
1006	Raman	2006	Computer

SQL View

- **Insert** the new row into the existing view. **Example**:

```
INSERT INTO Student_View (Student_ID, Stu_Subject, Stu_Marks) VALUES (1007, Hindi, 89);
```

- **Delete** the existing row from the view. **Example**:

```
DELETE FROM Student_View WHERE Stu_Subject = 'Math' ;
```

- **Drop** a View. **Example**:

```
DROP VIEW Student_View;
```

Question and Answering (Q&A)