HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN

Lập trình mạng

Giảng viên: TS. Nguyễn Trọng Khánh

Điện thoại/E-mail: khanhnt82@gmail.com

Bộ môn: CNPM- Khoa CNTT1

Học kỳ/Năm biên soạn: August 2018

Kết nối cơ sở dữ liệu trong Java

Nội dung

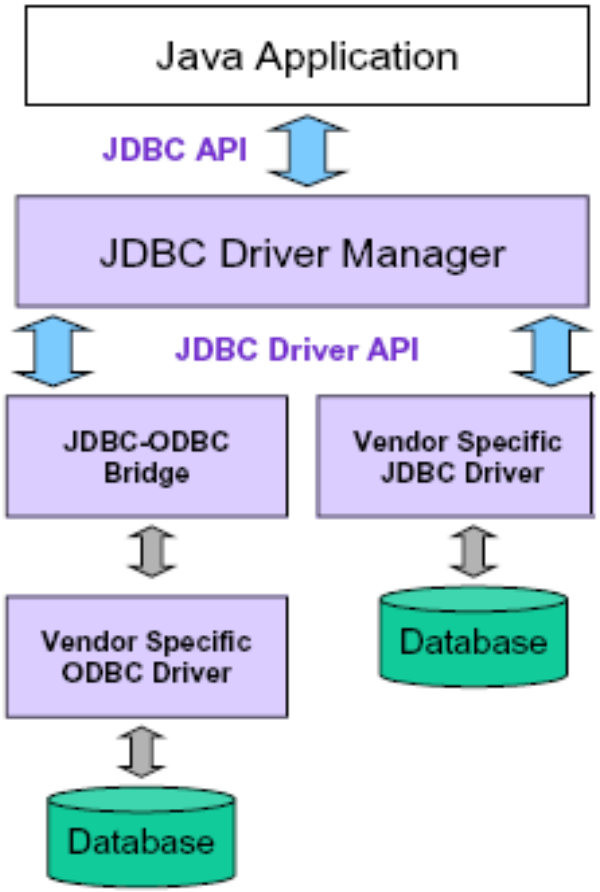
- Tổng quan JDBC
- JDBC Drivers
- 7 bước sử dụng JDBC
- Lấy dữ liệu từ ResultSet

Tổng quan JDBC

- Thư viện chuẩn để truy cập cơ sở dữ liệu quan hệ
  - Các API chuẩn hoá
    - Để thiết lập kết nối tới cơ sở dữ liệu
    - Khởi tạo truy vấn
    - Phương thức để tạo truy vấn lưu trữ
    - Cấu trúc dữ liệu cho kết quả truy vấn
      - Xác định số lượng cột
      - Tìm kiếm metadata etc.
  - API không chuẩn hoá cú pháp SQL
    - JDBC không nhúng SQL
  - Lớp JDBC nằm trong gói java.sql package

JDBC Drivers

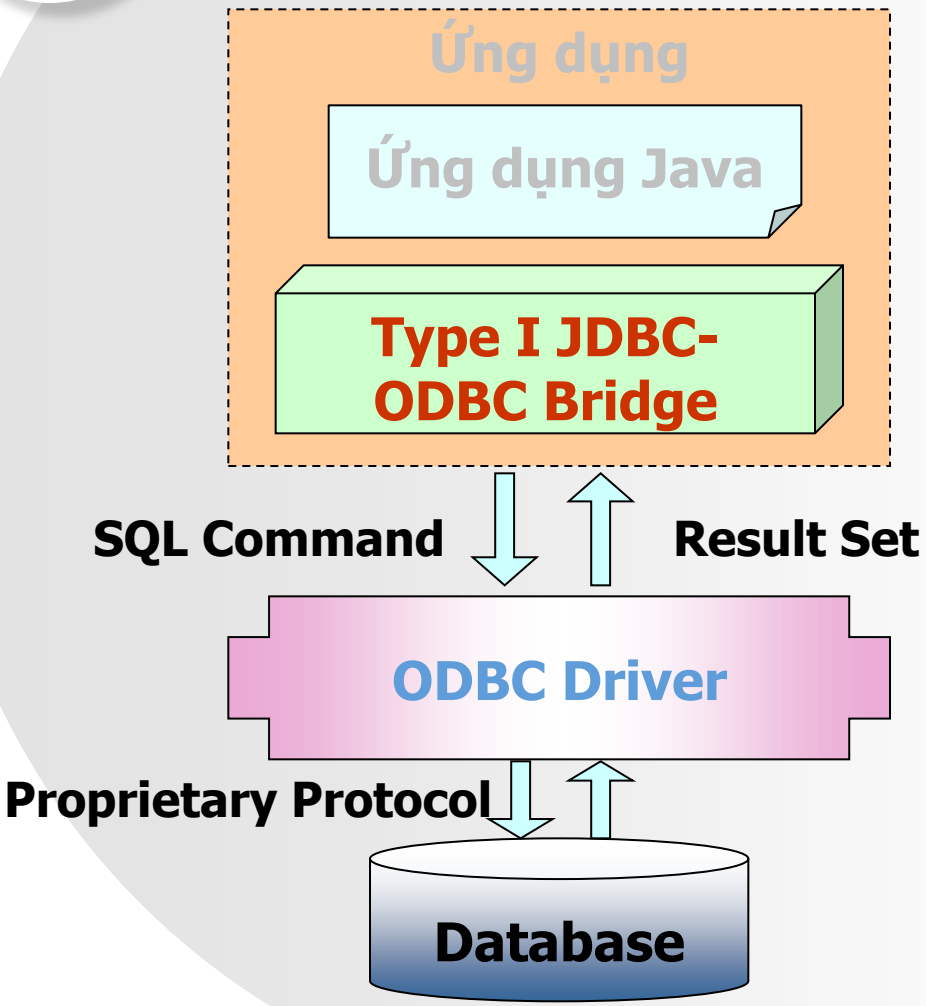
- JDBC bao gồm 2 thành phần :
  - JDBC API
  - JDBC Driver Manager : liên kết với các driver cụ thể của nhà cung cấp
    - 4 type : loại 1 – JDBC-ODBC và loại 4 thường dùng nhất.



JDBC Driver

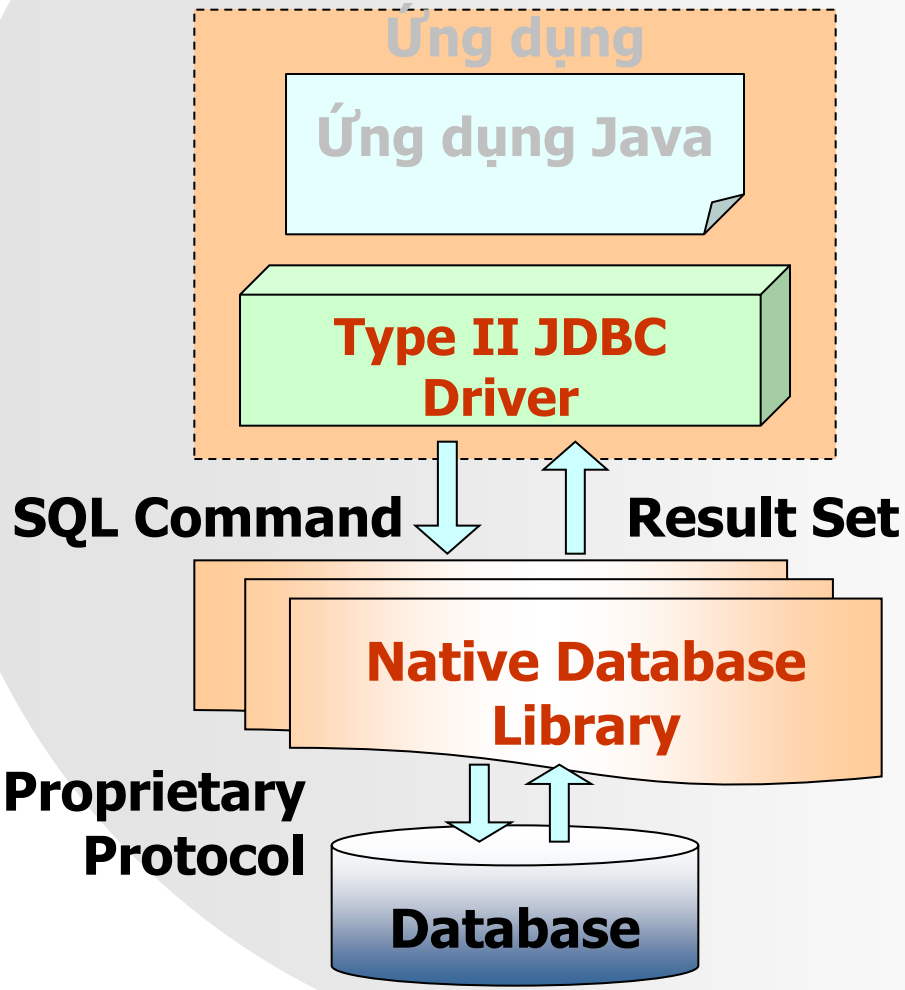
- 4 loại JDBC Driver
  - Loại 1: JDBC/ODBC
  - Loại 2: Native-API
  - Loại 3: Open Protocol-Net
  - Loại 4: Proprietary-Protocol-Net
- Loại 2,3,4 nói chung được viết bởi nhà cung cấp csdl, hiệu quả hơn loại 1 nhưng thực hiện phức tạp hơn.

Loại 1: JDBC-ODBC Bridge



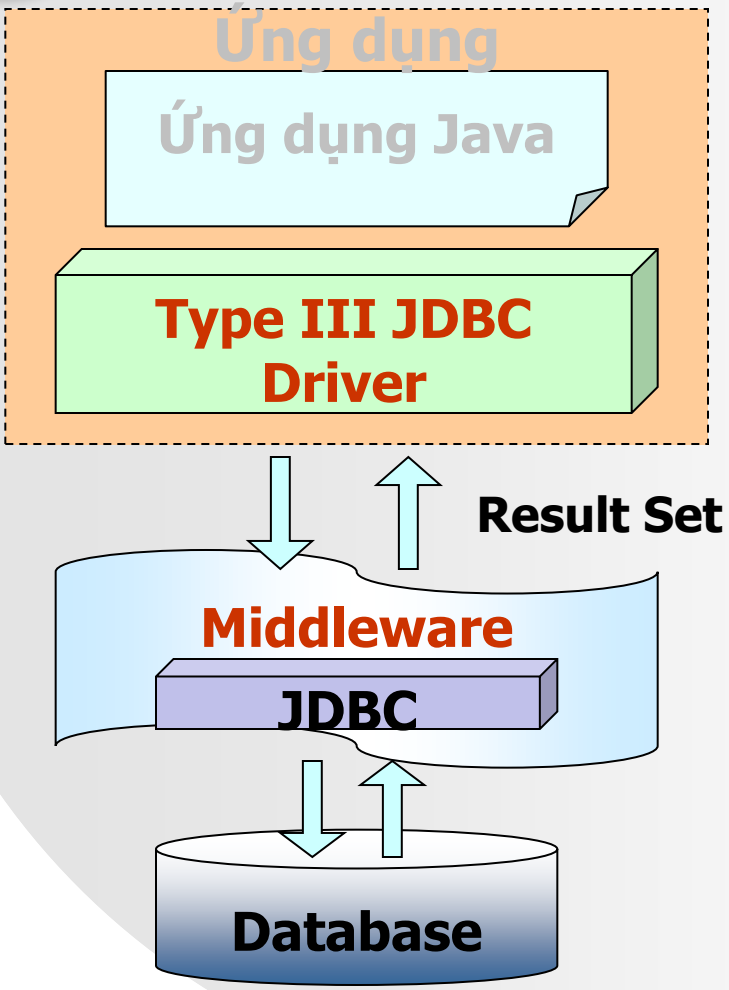
- jdk hỗ trợ cầu nối jdbc-odbc (jdbc-odbc bridge).
- Mềm dẻo nhưng không hiệu quả.
- Tích hợp trong jdk (rt.jar)
- Java 8: Không dùng
- Ví dụ: Microsoft Access

Loại 2: Native-API , partly Java driver



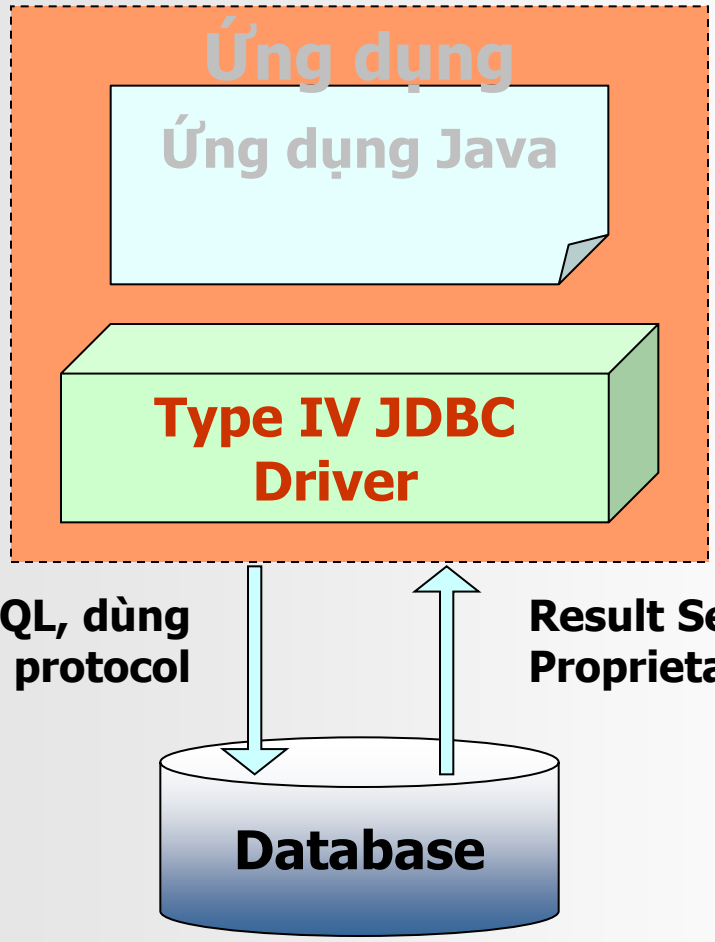
- JDBC API → native C/C++ API
- Cần có driver của DB
- DB thay đổi → thay đổi driver
- Nhanh hơn JDBC/ODBC
- Ví dụ: Oracle Call Interface

Loại 3: JDBC-net, pure Java driver



- 3 tầng
  - Ứng dụng: JDBC client → socket
  - Middleware: Socket → DBMS API (Loại 1, 2, hoặc 4)
  - DB
- Không phải của nhà cung cấp csdl
- Tất cả bằng mã java

Loại 4: Proprietary-Protocol Net



- 100% java
- Giao tiếp trực tiếp với hệ CSDL không cần chuyển đổi, thông qua socket
- Ví dụ: MySQL

Lựa chọn Driver

- Ứng dụng tương tác với 1 DBMS, ví dụ Oracle, Sybase, IBM, MySQL → loại 4
- Ứng dụng tương tác với nhiều loại DBMS → loại 3
- Với DBMS không hỗ trợ loại 3, 4 → loại 2
- Loại 1: đơn giản, không hỗ trợ từ Java 8

7 bước làm việc với JDBC (1)

1. Nạp driver
2. Định nghĩa Connection URL
3. Kết nối CSDL bằng đối tượng Connection
4. Tạo đối tượng Statement
5. Thi hành câu truy vấn
6. Xử lý kết quả
7. Đóng kết nối


7 bước làm việc với JDBC (2)

1. Nạp driver

```
String dbClass;
...
try {
  Class.forName(dbClass);
} catch { ClassNotFoundException cnfe) {
  System.out.println("Error loading driver: " cnfe);
}
```

- dbClass:
  - Microsoft: `"connect.microsoft.MicrosoftDriver";`
  - Oracle: `"oracle.jdbc.driver.OracleDriver";`
  - MySql: `"com.mysql.jdbc.Driver";`

7 bước làm việc với JDBC (3)

2. Connection URL

```
String host = "dbhost.yourcompany.com";
String dbName = "someName";
int port = 1234;
String oracleURL ="jdbc:oracle:thin:@"+host+":"+port+":"+dbName;
String sybaseURL ="jdbc:sybase:Tds:" + host  +
                  ":" + port + ":" + "?SERVICENAME=" + dbName;
String mysqlURL ="jdbc:mysql:"+host +":"+ port + "/" +dbName;
```

3. Thiết lập kết nối

```
String username = "jay_debesee";
String password = "secret";
Connection connection =
DriverManager.getConnection(oracleURL,username,
                            password);
```


7 bước làm việc với JDBC (4)

4. Tạo đối tượng Statement

```
Statement statement =
  connection.createStatement();
```

5. Chạy truy vấn

```
String query =
  "SELECT col1, col2, col3 FROM sometable";
ResultSet resultSet =
  statement.executeQuery(query);
```

- Để sửa cơ sở dữ liệu, sử dụng `executeUpdate`, với các string có `UPDATE`, `INSERT`, or `DELETE`
- Sử dụng `setQueryTimeout` để chỉ định thời gian delay lớn nhất để có kết quả

7 bước làm việc với JDBC (5)

6. Xử lý kết quả

```
while(resultSet.next()) {
   System.out.println(resultSet.getString(1) + " "
+ resultSet.getString(2) + " "
+ resultSet.getString(3));
}
```

   - Cột đầu tiên có index 1, không phải 0
   - `ResultSet` cung cấp nhiều phương thức `getXxx` để lấy index hoặc tên cột và trả dữ liệu

7. Đóng kết nối

```
connection.close();
```

Các kiểu dữ liệu

JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT DOUBLE	double
BINARY VARBINARY LONGVARBINARY	byte[]
CHAR VARCHAR LONGVARCHAR	String

JDBC Type	Java Type
NUMERIC DECIMAL	BigDecimal
DATE	java.sql.Date
TIME TIMESTAMP	java.sql.Timestamp
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

*SQL3 data type supported in JDBC 2.0

Statement (1)

- Overview
  - Thông qua đối tượng the `Statement`, các lệnh SQL được gửi tới DB.
  - 3 kiểu đối tượng :
    - **`Statement`**
      - lệnh SQL đơn giản
    - **`PreparedStatement`**
      - lệnh SLQ dịch trước truyền tham số
    - **`CallableStatement`**
      - stored procedure


Statement (2)

- executeQuery
  - Chay truy vấn và trả về bảng dự liệu (ResultSet)
  - Không có kết quả null, có thể rỗng

```
ResultSet results =
    statement.executeQuery("SELECT a, b FROM table");
```

- executeUpdate
  - Sử dụng để thực thi truy vấn INSERT, UPDATE, DELETE
  - Kết quả: số dòng thêm, sửa hoặc xóa
  -

```
int rows =
    statement.executeUpdate("DELETE FROM EMPLOYEES" +
                            "WHERE STATUS=0");
```

Prepared Statements

Sử dụng cho trường hợp không biết trước giá trị tham số hoặc có nhiều giá trị trong câu lệnh truy vấn SQL

```
Connection connection =
  DriverManager.getConnection(url, user, password);
PreparedStatement statement =
  connection.prepareStatement("UPDATE employees "+
                              "SET salary = ? " +
                              "WHERE id = ?");
int[] newSalaries = getSalaries();
int[] employeeIDs = getIDs();
for(int i=0; i<employeeIDs.length; i++) {
  statement.setInt(1, newSalaries[i]);
  statement.setInt(2, employeeIDs[i]);
  statement.executeUpdate();
}
```

Transactions

- Mặc định: DB sẽ bị tác động khi thực thi lệnh SQL
- Transaction: nhóm nhiều lệnh trong một lần thực thi → cải thiện tốc độ; tắt chế độ mặc đinh

```
Connection connection =
  DriverManager.getConnection(url, username, passwd);
connection.setAutoCommit(false);
try {
  statement.executeUpdate(...);
  statement.executeUpdate(...);
  ...
  connection.commit();
} catch (Exception e) {
  try {
    connection.rollback();
  } catch (SQLException sqle) {}
} finally {
  try {
   connection.close();
  } catch (SQLException sqle) { }
}
```

Case Study:
Quản lý nhân viên

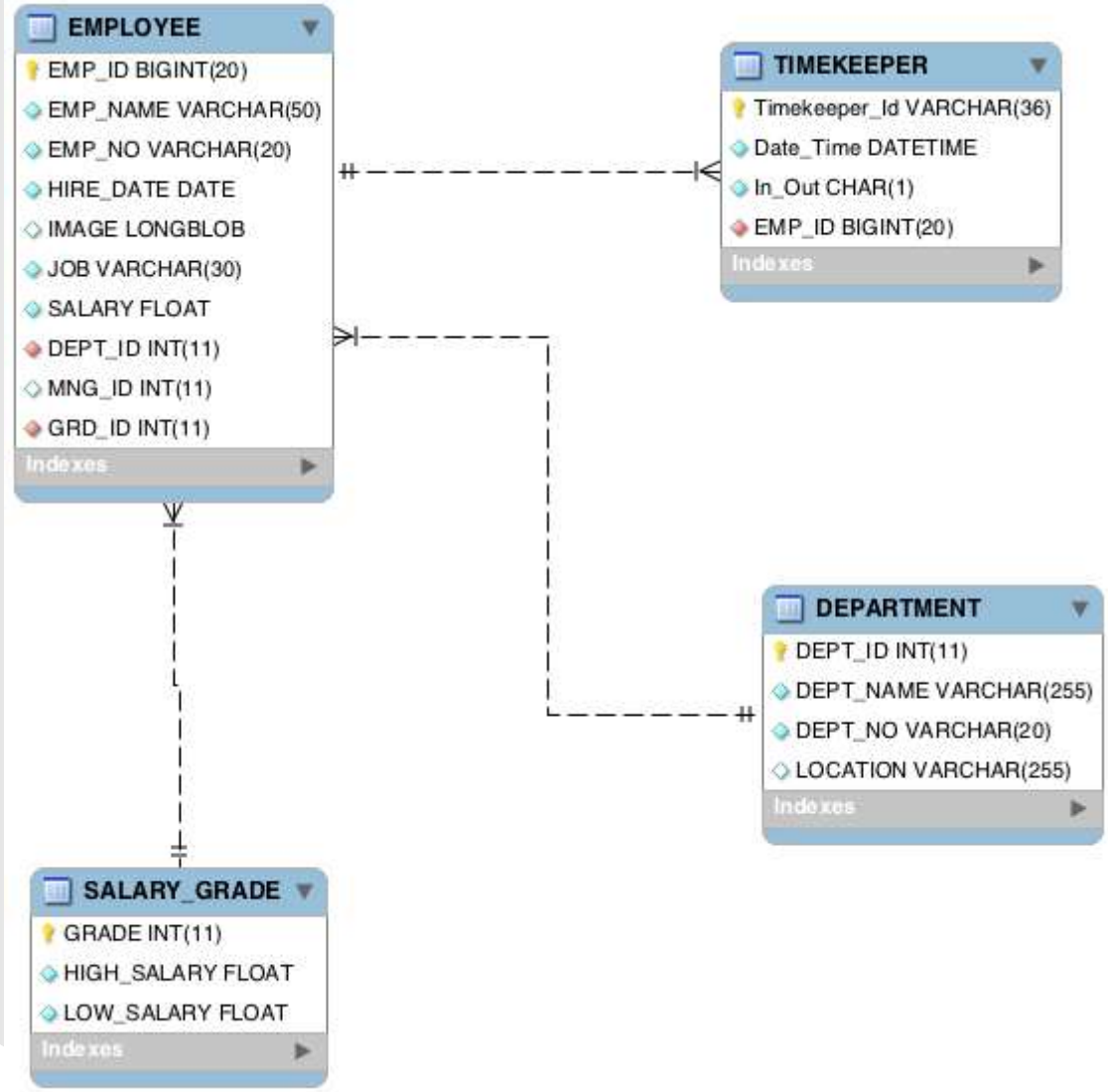
Bài toán

- Mỗi phòng ban DEPARTMENT(DEPT_ID, DEPT_NAME, DEPT_NO, LOCATION) có nhiều nhân viên EMPLOYEE(EMP_ID, EMP_NAME, EMP_NO, HIRE_DATE, IMAGE, JOB, SALARY, DEPT_ID, MNG_ID, GRD_ID)
- Mỗi nhân viên có một mức lương SALARY_GRADE(GRADE, HIGH_SALARY, LOW_SALARY), trong đó lương của nhân viên không được thấp hơn mức LOW_SALARY và cao hơn mức HIGH_SALARY
- Thông tin đi làm hay nghỉ của mỗi nhân viên được lưu trong bản ghi TIMEKEEPER(ID, DATE_TIME, IN_OUT, EMP_ID)

Yêu cầu

- Xây dựng ứng dụng chức năng cho phép liệt kê, thêm mới, sửa, xóa phòng ban, nhân viên, thông ti đi làm/nghỉ phép
  - Mô hình MVC
  - Phiên bản console
  - Phiên bản giao diện

CSDL


EMPLOYEE
EMP_ID BIGINT(20)
EMP_NAME VARCHAR(50)
EMP_NO VARCHAR(20)
HIRE_DATE DATE
IMAGE LONGBLOB
JOB VARCHAR(30)
SALARY FLOAT
DEPT_ID INT(11)
MNG_ID INT(11)
GRD_ID INT(11)
Indexes
TIMEKEEPER
Timekeeper_Id VARCHAR(36)
Date_Time DATETIME
In_Out CHAR(1)
EMP_ID BIGINT(20)
Indexes
DEPARTMENT
DEPT_ID INT(11)
DEPT_NAME VARCHAR(255)
DEPT_NO VARCHAR(20)
LOCATION VARCHAR(255)
Indexes
SALARY_GRADE
GRADE INT(11)
HIGH_SALARY FLOAT
LOW_SALARY FLOAT
Indexes

Demo bảng Employee

Bài tập

- Hoàn thành với các bảng còn lại
  - Console
  - GUI