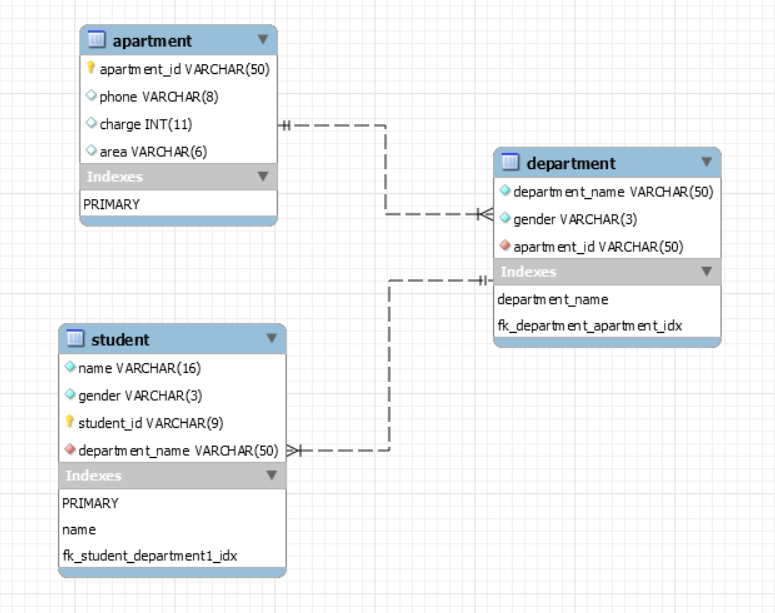
### 一、

## 1.



## 2.

String create\_apartment;  
create\_apartment = **"CREATE TABLE IF NOT EXISTS `apartment`.`apartment` (\n"** +  
 **" `apartment\_id` VARCHAR(50) NOT NULL,\n"** +  
 **" `phone` VARCHAR(8) NULL,\n"** +  
 **" `charge` INT NULL,\n"** +  
 **" PRIMARY KEY (`apartment\_id`)\n"** +  
 **");"**;  
String create\_student;  
create\_student = **"CREATE TABLE IF NOT EXISTS `apartment`.`student` (\n"** +  
 **" `student\_id` VARCHAR(9) NOT NULL,\n"** +  
 **" `name` VARCHAR(16) NOT NULL,\n"** +  
 **" `gender` VARCHAR(3) NOT NULL,\n"** +  
 **" `department\_name` VARCHAR(50) NOT NULL,\n"** +  
 **" PRIMARY KEY (`student\_id`),\n"** +  
 **" INDEX (`name`)\n"** +  
 **");"**;  
String create\_department;  
create\_department = **"CREATE TABLE IF NOT EXISTS `apartment`.`department` (\n"** +  
 **" `department\_name` VARCHAR(50) NOT NULL,\n"** +  
 **" `gender` VARCHAR(3) NOT NULL,\n"** +  
 **" `apartment\_id` VARCHAR(50) NOT NULL,\n"** +  
 **" `area` VARCHAR(8) NOT NULL,\n"** +  
 **" unique(`department\_name`,`gender`)\n"** +  
 **");"**;

## 3.

## 类ExcelReader：读入Excel并插入数据库

**package** apartment;  
  
**import** org.apache.poi.hssf.usermodel.HSSFWorkbook;  
**import** org.apache.poi.ss.usermodel.Cell;  
**import** org.apache.poi.ss.usermodel.Row;  
**import** org.apache.poi.ss.usermodel.Sheet;  
**import** org.apache.poi.ss.usermodel.Workbook;  
  
**import** java.io.FileInputStream;  
**import** java.io.FileNotFoundException;  
**import** java.io.InputStream;  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
*/\*\*  
 \* Created by LWY on 2017/11/7.  
 \*/***public class** ExcelReader {  
 */\*\*  
 \* 适用于第一行是标题行的excel，例如  
 \* 姓名 年龄 性别 身高  
 \* 张三 25 男 175  
 \* 李四 22 女 160  
 \* 每一行构成一个map，key值是列标题，value是列值。没有值的单元格其value值为null  
 \* 返回结果最外层的list对应一个sheet页，map对应sheet页中的一行  
 \*  
 \** ***@throws*** *Exception  
 \*/* **private** Map<String, String> **apartmentPhone**;  
  
 **public** ExcelReader() {  
 String txtPath = **"C:\\Users\\Venric\\Desktop\\database\\Homework II\_design\\电话.txt"**;  
 **apartmentPhone** = **new** GetPhone().readTXT(txtPath);  
 }  
  
 **public void** readExcelAndInsert(String filepath) **throws** Exception {  
 **long** a = System.*currentTimeMillis*();  
 String fileType = filepath.substring(filepath.lastIndexOf(**"."**) + 1, filepath.length());  
 InputStream is = **null**;  
 Workbook wb = **null**;  
 Connection con = MyConnection.*getConnection*();  
 **try** {  
 is = **new** FileInputStream(filepath);  
  
 **if** (fileType.equals(**"xls"**)) {  
 wb = **new** HSSFWorkbook(is);  
 } **else** {  
 **throw new** Exception(**"读取的不是excel文件"**);  
 }  
  
  
 Sheet sheet = wb.getSheetAt(0);  
  
 List<String> titles = **new** ArrayList<String>();*//放置所有的标题  
  
 // <id,<\*\*>* Map<String,Map<String,String>> apartment = **new** HashMap<String, Map<String, String>>();  
 Map<String,Map<String,String>> department = **new** HashMap<String, Map<String, String>>();  
  
 **int** rowSize = sheet.getLastRowNum() + 1;  
 PreparedStatement apartStmt = con.prepareStatement(**"INSERT INTO `apartment`.apartment (apartment\_id,phone,charge,area) VALUE(?,?,?,?)"**);  
 PreparedStatement departStmt = con.prepareStatement(**"INSERT INTO `apartment`.department (department\_name,gender,apartment\_id) VALUE(?,?,?)"**);  
 PreparedStatement studentStmt = con.prepareStatement(**"INSERT INTO `apartment`.student (student\_id,name,gender,department\_name) VALUE(?,?,?,?)"**);  
 **for** (**int** j = 0; j < rowSize; j++) { *//遍历行* Row row = sheet.getRow(j);  
 **if** (row == **null**) {*//略过空行* **continue**;  
 }  
 **int** cellSize = row.getLastCellNum();*//行中有多少个单元格，也就是有多少列* **if** (j == 0) {*//第一行是标题行* **for** (**int** k = 0; k < cellSize; k++) {  
 Cell cell = row.getCell(k);  
 titles.add(cell.toString());  
 }  
 } **else** {*//其他行是数据行* Map<String, String> rowMap = **new** HashMap<String, String>();*//对应一个数据行* **for** (**int** k = 0; k < titles.size(); k++) {  
 Cell cell = row.getCell(k);  
 String key = titles.get(k);  
 String value = **null**;  
 **if** (cell != **null**) {  
 value = cell.toString();  
 }  
 rowMap.put(key, value);  
 }  
 studentStmt.setString(1, rowMap.get(**"学号"**));  
 studentStmt.setString(2, rowMap.get(**"姓名"**));  
 studentStmt.setString(3, rowMap.get(**"性别"**));  
 studentStmt.setString(4, rowMap.get(**"院系"**));  
 studentStmt.executeUpdate();  
  
 **if**(!department.containsKey(rowMap.get(**"院系"**))){  
 Map<String,String> departValue = **new** HashMap<String, String>();  
 departValue.put(**"gender"**,rowMap.get(**"性别"**));  
 departValue.put(**"apartment\_id"**,rowMap.get(**"宿舍楼"**));  
 department.put(rowMap.get(**"院系"**),departValue);  
 }  
 **if**(!apartment.containsKey(rowMap.get(**"宿舍楼"**))){  
 Map<String,String> apartValue = **new** HashMap<String, String>();  
  
 apartValue.put(**"phone"**,**apartmentPhone**.get(rowMap.get(**"宿舍楼"**)));  
 apartValue.put(**"charge"**,rowMap.get(**"住宿费标准"**));  
 apartValue.put(**"area"**,rowMap.get(**"校区"**));  
 apartment.put(rowMap.get(**"宿舍楼"**),apartValue);  
 }  
 }  
 }  
  
 **for** (Map.Entry<String, Map<String,String>> entry : department.entrySet()) {  
 departStmt.setString(1,entry.getKey());  
 departStmt.setString(2,entry.getValue().get(**"gender"**));  
 departStmt.setString(3,entry.getValue().get(**"apartment\_id"**));  
 departStmt.executeUpdate();  
 }  
 **for** (Map.Entry<String, Map<String,String>> entry : apartment.entrySet()) {  
 apartStmt.setString(1,entry.getKey());  
 apartStmt.setString(2,entry.getValue().get(**"phone"**));  
 apartStmt.setInt(3,Integer.*valueOf*(entry.getValue().get(**"charge"**)));  
 apartStmt.setString(4,entry.getValue().get(**"area"**));  
 apartStmt.executeUpdate();  
 }  
  
 } **catch** (FileNotFoundException e) {  
 **throw** e;  
 } **finally** {  
 **long** b=System.*currentTimeMillis*();  
 System.***out***.println(**"用时"**+ (b-a)+**" ms"**);  
 **if** (wb != **null**) {  
 wb.close();  
 }  
 **if** (is != **null**) {  
 is.close();  
 }  
 }  
 }  
  
 **public static void** main(String[] args) {  
 String excelPath = **"C:\\Users\\Venric\\Desktop\\database\\Homework II\_design\\分配方案.xls"**;  
  
 **try** {  
 **new** ExcelReader().readExcelAndInsert(excelPath);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

## 类GetPhone 读入txt文件

**package** apartment;  
  
**import** java.io.\*;  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
*/\*\*  
 \* Created by LWY on 2017/11/7.  
 \*/***public class** GetPhone {  
  
 **private** Map<String, String> **apartmentPhone** = **new** HashMap<String, String>();  
 **private** String **filePath** = **""**;  
  
 **public** Map<String, String> readTXT(String filePath) {  
 **if** (!**apartmentPhone**.isEmpty() && **this**.**filePath**.equals(filePath)) {  
 **return apartmentPhone**;  
 }  
  
 **try** {  
 File fileView = **new** File(filePath);  
 **this**.**filePath** = filePath;  
  
 BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fileView), **"UTF-8"**));  
 String line = in.readLine(); *// desert the first line* String apartmentID;  
 String phone;  
 **while** ((line = in.readLine()) != **null**) {  
 apartmentID = line.split(**";"**)[0];  
 phone = line.split(**";"**)[1];  
 **apartmentPhone**.put(apartmentID, phone);  
 }  
  
 in.close();  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 } **catch** (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return apartmentPhone**;  
 }  
}

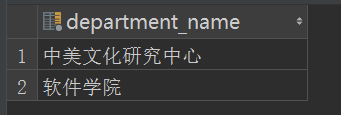
## 运行时间截图

## 

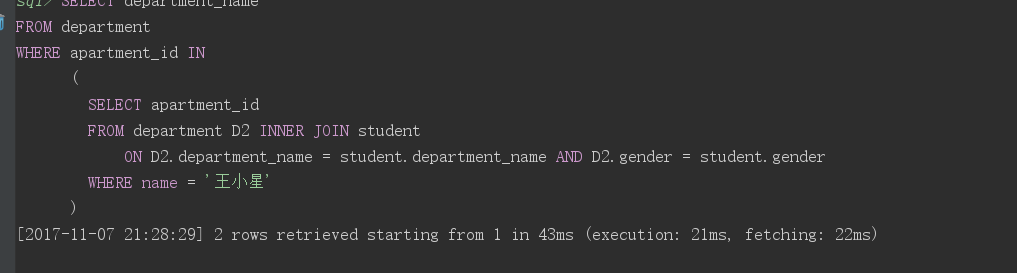
## 4.

SELECT department\_name  
FROM department  
WHERE apartment\_id IN  
 (  
 SELECT apartment\_id  
 FROM department D2 INNER JOIN student  
 ON D2.department\_name = student.department\_name AND D2.gender = student.gender  
 WHERE name = '王小星'  
 )

## Result

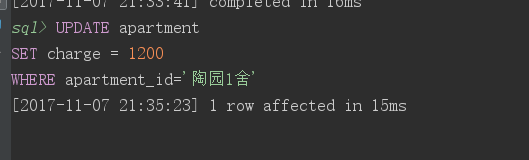


## Efficiency



## 5.

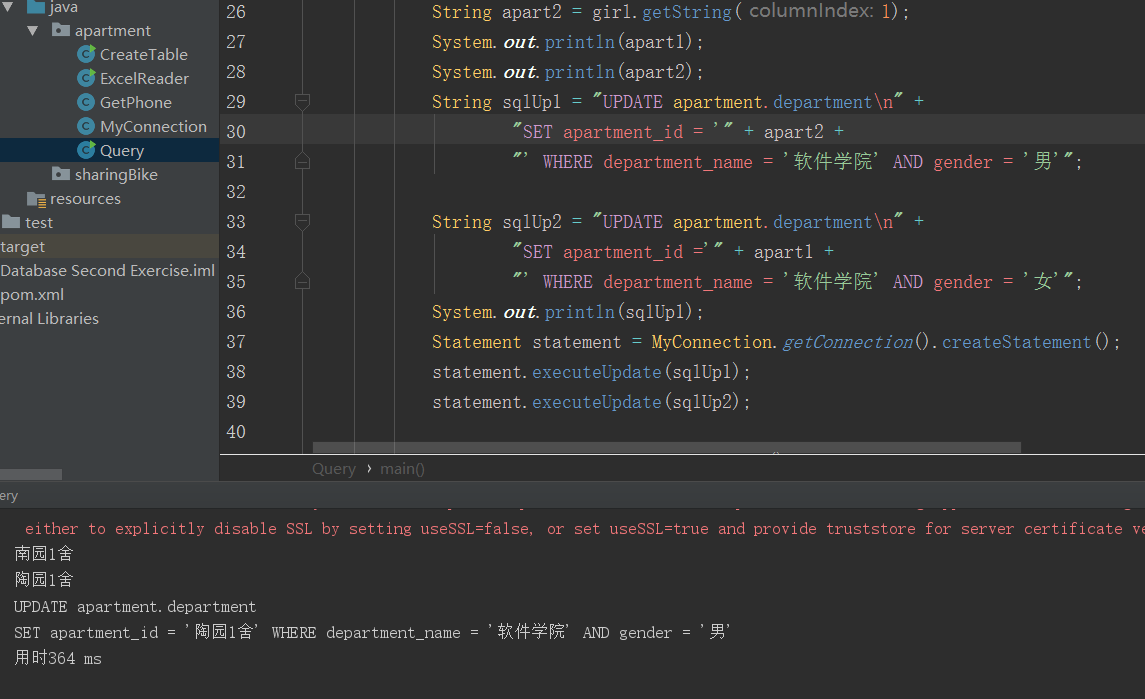
UPDATE apartment  
SET charge = 1200  
WHERE apartment\_id='陶园1舍';



## 6.

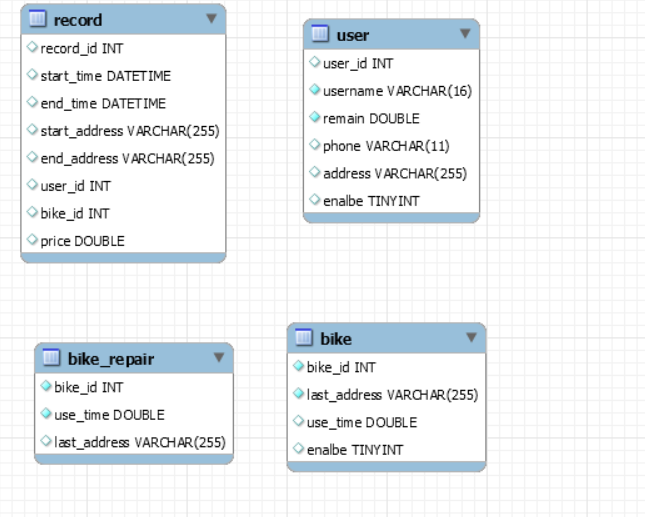
**package** apartment;  
  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
  
*/\*\*  
 \* Created by LWY on 2017/11/7.  
 \*/***public class** Query {  
 **public static void** main(String[] args) {  
 **long** a = System.*currentTimeMillis*();  
  
 *//language=MySQL* String queryBoy = **"SELECT apartment\_id FROM apartment.department WHERE department\_name = '软件学院' AND gender='男'"**;  
 String queryGirl = **"SELECT apartment\_id FROM apartment.department WHERE department\_name = '软件学院' and gender='女'"**;  
  
  
 **try** {  
  
 ResultSet boy = MyConnection.*myQuery*(queryBoy);  
 ResultSet girl = MyConnection.*myQuery*(queryGirl);  
 boy.next();  
 girl.next();  
 String apart1 = boy.getString(1);  
 String apart2 = girl.getString(1);  
 System.***out***.println(apart1);  
 System.***out***.println(apart2);  
 String sqlUp1 = **"UPDATE apartment.department\n"** +  
 **"SET apartment\_id = '"** + apart2 +  
 **"' WHERE department\_name = '软件学院' AND gender = '男'"**;  
  
 String sqlUp2 = **"UPDATE apartment.department\n"** +  
 **"SET apartment\_id ='"** + apart1 +  
 **"' WHERE department\_name = '软件学院' AND gender = '女'"**;  
 System.***out***.println(sqlUp1);  
 Statement statement = MyConnection.*getConnection*().createStatement();  
 statement.executeUpdate(sqlUp1);  
 statement.executeUpdate(sqlUp2);  
  
 **long** b = System.*currentTimeMillis*();  
 System.***out***.println(**"用时"** + (b - a) + **" ms"**);  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
}

## efficiency



### 二、

## 1. 表结构设计

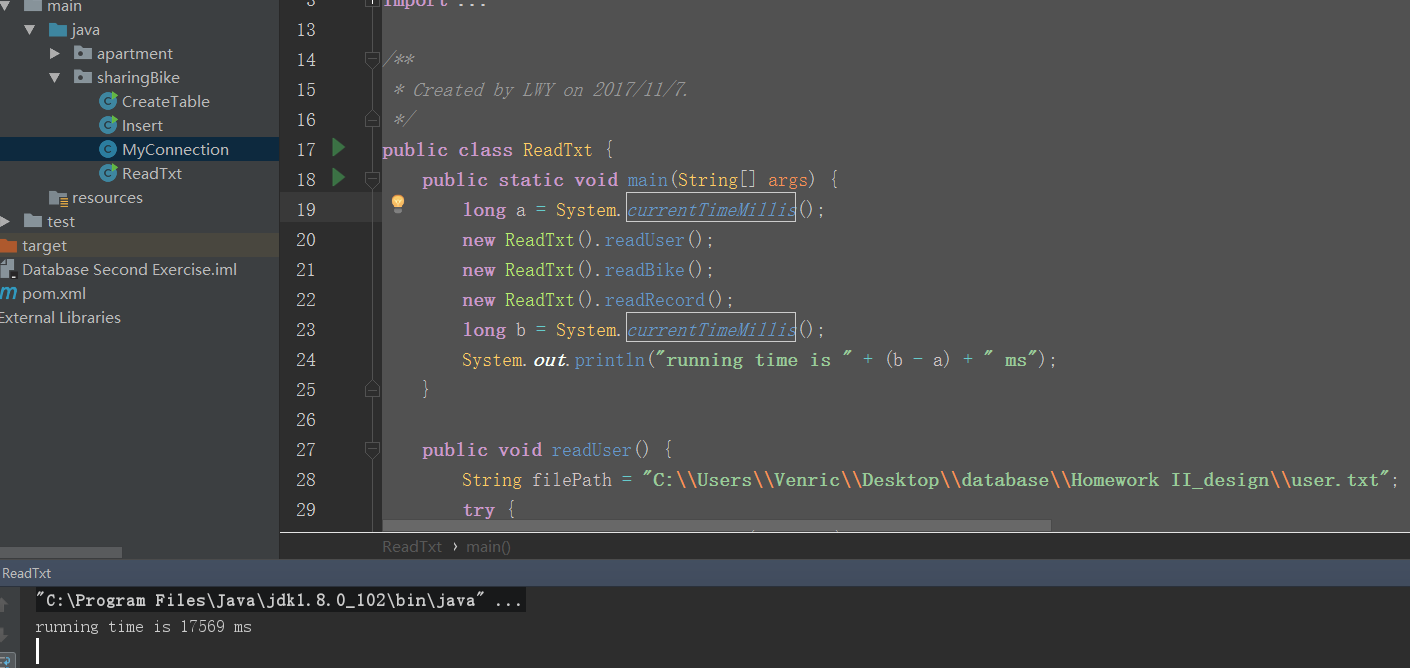


**package** sharingBike;  
  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
  
*/\*\*  
 \* Created by LWY on 2017/11/6.  
 \*/***public class** CreateTable {  
  
  
 **public static void** main(String[] args) {  
  
  
 String create\_user;  
 create\_user = **"CREATE TABLE IF NOT EXISTS `bike`.`user` (\n"** +  
 **" `user\_id` INT NULL,\n"** +  
 **" `username` VARCHAR(16) NOT NULL,\n"** +  
 **" `remain` DOUBLE NOT NULL,\n"** +  
 **" `phone` VARCHAR(11) NULL,\n"** +  
 **" `address` VARCHAR(255) NULL,\n"** +  
 **" `enalbe` TINYINT NULL);"**;  
 String create\_record;  
 create\_record = **"CREATE TABLE IF NOT EXISTS `bike`.`record` (\n"** +  
 **" `record\_id` INT NULL,\n"** +  
 **" `start\_time` DATETIME NULL,\n"** +  
 **" `end\_time` DATETIME NULL,\n"** +  
 **" `start\_address` VARCHAR(255) NULL,\n"** +  
 **" `end\_address` VARCHAR(255) NULL,\n"** +  
 **" `user\_id` INT NULL,\n"** +  
 **" `bike\_id` INT NULL,\n"** +  
 **" `price` DOUBLE NULL);\n"**;  
 String create\_bike;  
 create\_bike = **"CREATE TABLE IF NOT EXISTS `bike`.`bike` (\n"** +  
 **" `bike\_id` INT NOT NULL,\n"** +  
 **" `last\_address` VARCHAR(255) NOT NULL,\n"** +  
 **" `use\_time` DOUBLE NULL,\n"** +  
 **" `enalbe` TINYINT NULL);"**;  
 String create\_bike\_repair;  
 create\_bike\_repair = **"CREATE TABLE IF NOT EXISTS `bike`.`bike\_repair` (\n"** +  
 **" `bike\_id` INT NOT NULL,\n"** +  
 **" `use\_time` DOUBLE NOT NULL,\n"** +  
 **" `last\_address` VARCHAR(255) NULL);"**;  
  
 Statement stmt;  
 **try** {  
 stmt = MyConnection.*getConnection*().createStatement();  
 stmt.executeUpdate(create\_user);  
 stmt.executeUpdate(create\_bike);  
 stmt.executeUpdate(create\_bike\_repair);  
 stmt.executeUpdate(create\_record);  
  
 stmt.close();  
 } **catch** (SQLException e1) {  
 e1.printStackTrace();  
 }  
  
  
 }  
}

## 插入语句：使用prepareStatement的批量插入，同时将rewriteBatchedStatements设为true，大幅度提高插入效率，普通硬盘下17s完成插入

**package** sharingBike;  
  
**import** java.io.\*;  
**import** java.sql.PreparedStatement;  
**import** java.sql.SQLException;  
**import** java.sql.Timestamp;  
**import** java.text.DateFormat;  
**import** java.text.ParseException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Arrays;  
**import** java.util.Calendar;  
**import** java.util.Date;  
  
*/\*\*  
 \* Created by LWY on 2017/11/7.  
 \*/***public class** ReadTxt {  
 **public static void** main(String[] args) {  
 **long** a = System.*currentTimeMillis*();  
 **new** ReadTxt().readUser();  
 **new** ReadTxt().readBike();  
 **new** ReadTxt().readRecord();  
 **long** b = System.*currentTimeMillis*();  
 System.***out***.println(**"running time is "** + (b - a) + **" ms"**);  
 }  
  
 **public void** readUser() {  
 String filePath = **"C:\\Users\\Venric\\Desktop\\database\\Homework II\_design\\user.txt"**;  
 **try** {  
 File fileView = **new** File(filePath);  
  
 BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fileView), **"UTF-8"**));  
  
 String line;  
  
 String sql = **"INSERT INTO bike.user (user\_id, username, phone, remain) VALUES (?,?,?,?)"**;  
  
 PreparedStatement preparedStatement = MyConnection.*getConnection*().prepareStatement(sql);  
 **while** ((line = in.readLine()) != **null**) {  
 String[] lins = line.split(**";"**);  
  
 preparedStatement.setInt(1, Integer.*valueOf*(lins[0]));  
 preparedStatement.setString(2, lins[1]);  
 preparedStatement.setString(3, lins[2]);  
 preparedStatement.setDouble(4, Double.*valueOf*(lins[3]));  
 preparedStatement.executeUpdate();  
 }  
  
 in.close();  
 } **catch** (IOException | SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public void** readBike() {  
 String filePath = **"C:\\Users\\Venric\\Desktop\\database\\Homework II\_design\\bike.txt"**;  
 **try** {  
 File fileView = **new** File(filePath);  
  
 BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fileView), **"UTF-8"**));  
 String line;  
 String sql = **"INSERT INTO bike.bike(bike\_id) VALUES (?)"**;  
 PreparedStatement preparedStatement = MyConnection.*getConnection*().prepareStatement(sql);  
 **while** ((line = in.readLine()) != **null**) {  
 preparedStatement.setInt(1, Integer.*valueOf*(line));  
 preparedStatement.executeUpdate();  
 }  
 MyConnection.*getConnection*().commit();  
 in.close();  
 } **catch** (IOException | SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public void** readRecord() {  
 String filePath = **"C:\\Users\\Venric\\Desktop\\database\\Homework II\_design\\record.txt"**;  
 **try** {  
 File fileView = **new** File(filePath);  
  
 BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fileView), **"UTF-8"**));  
 String sql = **"INSERT INTO bike.record(user\_id, bike\_id,start\_time, end\_time, start\_address, end\_address,price) VALUES (?,?,?,?,?,?,?)"**;  
  
 PreparedStatement preparedStatement = MyConnection.*getConnection*().prepareStatement(sql);  
 String line;  
 **final int** batchSize = 1000;  
 **int** count = 0;  
 **while** ((line = in.readLine()) != **null**) {  
 String[] lins = line.split(**";"**);  
 **int** value;  
 **if** (count == 0) {  
 value = 11127;  
 } **else** {  
 value = Integer.*parseInt*(lins[0]);  
 }  
 preparedStatement.setInt(1, value);  
 preparedStatement.setInt(2, Integer.*valueOf*(lins[1]));  
  
 SimpleDateFormat formatter = **new** SimpleDateFormat(**"yyyy/MM/dd-HH:mm:ss"**);  
 Timestamp start = **new** Timestamp(formatter.parse(lins[3]).getTime());  
 Timestamp end = **new** Timestamp(formatter.parse(lins[5]).getTime());  
 preparedStatement.setTimestamp(3, start);  
 preparedStatement.setTimestamp(4, end);  
 preparedStatement.setString(5, lins[2]);  
 preparedStatement.setString(6, lins[4]);  
 preparedStatement.setDouble(7, calculatePrice(start, end));  
 preparedStatement.addBatch();  
  
 **if** (++count % batchSize == 0) {  
 preparedStatement.executeBatch();  
 }  
 }  
 preparedStatement.executeBatch();  
 MyConnection.*getConnection*().commit();  
 in.close();  
 } **catch** (SQLException | IOException | ParseException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private double** calculatePrice(Timestamp start, Timestamp end) {  
 *// this code can be rewritten with table driven  
 //but for speed, following code was written.Shamed.* **double** minute = (end.getTime() - start.getTime()) / 1000.0 / 60;  
 **double** price = 0;  
 **if** (minute <= 30) {  
 price = 1;  
 } **else if** (minute > 30 && minute <= 60) {  
 price = 2;  
 } **else if** (minute > 60 && minute <= 90) {  
 price = 3;  
 } **else if** (minute > 90) {  
 price = 4;  
 }  
 **return** price;  
 }  
}

## Efficiency



## 2.

**package** sharingBike;  
  
**import** java.sql.\*;  
**import** java.util.Arrays;  
  
*/\*\*  
 \* Created by LWY on 2017/11/7.  
 \*/***public class** Insert {  
 **public static void** main(String[] args) {  
 **try** {  
 *insertAddress*();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 **public static void** insertAddress() **throws** SQLException {  
 String select = **"SELECT user\_id,end\_address\n"** +  
 **"FROM bike.record R1\n"** +  
 **"WHERE** *TIME***(R1.end\_time) BETWEEN '18:00:00' AND '23:59:59'\n"** +  
 **"GROUP BY user\_id\n"** +  
 **"HAVING** *count***(***\****) >= ALL (SELECT** *count***(***\****)\n"** +  
 **" FROM bike.record R2\n"** +  
 **" WHERE R1.user\_id = R2.user\_id AND** *TIME***(R2.end\_time) "** +  
 **"BETWEEN '18:00:00' AND '23:59:59')"**;  
 ResultSet myquery = MyConnection.*myQuery*(select);  
  
 String update = **"UPDATE bike.user SET address = ? WHERE user\_id = ?"**;  
 PreparedStatement preparedStatement = MyConnection.*getConnection*().prepareStatement(update);  
  
 **final int** batchSize = 1000;  
 **int** count = 0;  
 **while** (myquery.next()) {  
 preparedStatement.setString(2, myquery.getString(1));  
 preparedStatement.setString(1, myquery.getString(2));  
 preparedStatement.addBatch();  
  
 **if** (++count % batchSize == 0) {  
 preparedStatement.executeBatch();  
 }  
 }  
 preparedStatement.executeBatch();  
 MyConnection.*getConnection*().commit();  
 }  
  
}

## 3.

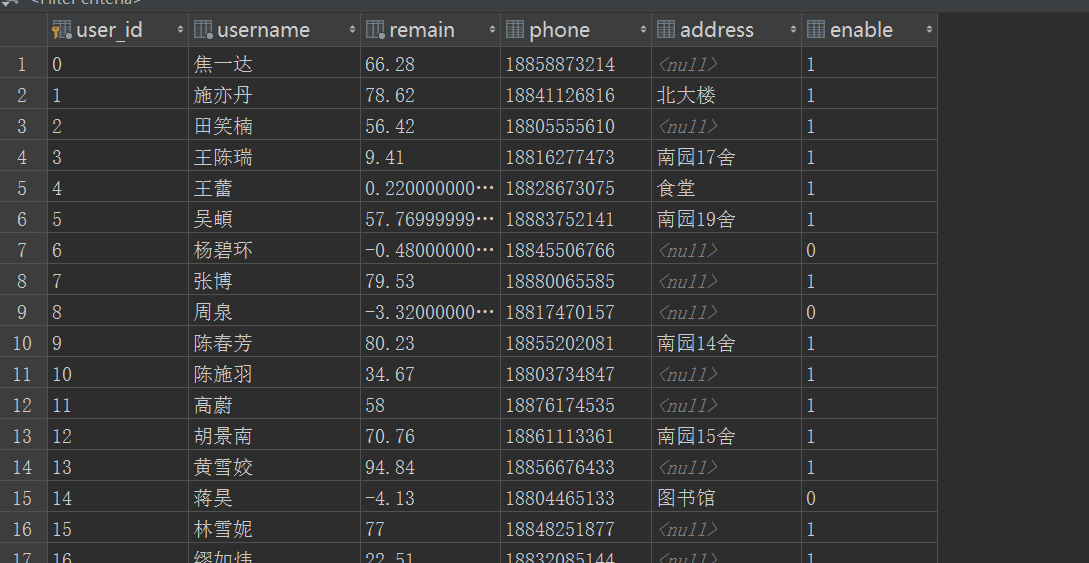
## 单个record价格计算

**private double** calculatePrice(Timestamp start, Timestamp end) {  
 *// this code can be rewritten with table driven  
 //but for speed, following code was written.Shamed.* **double** minute = (end.getTime() - start.getTime()) / 1000.0 / 60;  
 **double** price = 0;  
 **if** (minute <= 30) {  
 price = 1;  
 } **else if** (minute > 30 && minute <= 60) {  
 price = 2;  
 } **else if** (minute > 60 && minute <= 90) {  
 price = 3;  
 } **else if** (minute > 90) {  
 price = 4;  
 }  
 **return** price;  
}

## 使用触发器检测Update余额字段，小于零则将用户enable设为0

**public static void** changeChargeAndSetEnable() **throws** SQLException {  
 String changeCharge = **"UPDATE bike.user SET remain=remain-? WHERE bike.user.user\_id=?"**;  
 PreparedStatement preparedStatement = MyConnection.*getConnection*().prepareStatement(changeCharge);  
 String price = **"SELECT user\_id,***sum***(price) FROM bike.record GROUP BY user\_id"**;  
 ResultSet resultSet = MyConnection.*myQuery*(price);  
 **final int** batchSize = 1000;  
 **int** count = 0;  
 **while** (resultSet.next()) {  
 preparedStatement.setInt(2, resultSet.getInt(1));  
 preparedStatement.setDouble(1, resultSet.getDouble(2));  
 preparedStatement.addBatch();  
 **if** (++count % batchSize == 0) {  
 preparedStatement.executeBatch();  
 }  
 }  
 preparedStatement.executeBatch();  
 MyConnection.*getConnection*().commit();  
}  
  
**public static void** addTriggerToUser() {  
 String sql = **"CREATE TRIGGER user\_enable\n"** +  
 **"BEFORE UPDATE ON user\n"** +  
 **"FOR EACH ROW\n"** +  
 **" BEGIN\n"** +  
 **" IF NEW.remain < 0\n"** +  
 **" THEN SET NEW.enable = 0;\n"** +  
 **" ELSEIF NEW.remain >= 0\n"** +  
 **" THEN SET NEW.enable = 1;\n"** +  
 **" END IF;\n"** +  
 **" END;\n"**;  
 MyConnection.*update*(sql);  
  
}

## 结果



## 4.

## 使用trigger，每次插入record的时候都更新bike

CREATE TRIGGER record\_bike  
AFTER INSERT ON record  
FOR EACH ROW  
 BEGIN  
 UPDATE bike.bike  
 SET last\_address = NEW.end\_address AND use\_time = use\_time + (NEW.end\_time - NEW.start\_time) / 3600.0  
 WHERE bike\_id = NEW.bike\_id;  
 END;

## 每月初进行维护

**package** sharingBike;  
  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.time.LocalDate;  
  
*/\*\*  
 \* Created by LWY on 2017/11/9.  
 \*/***public class** Bike {  
 **public static void** main(String[] args) {  
 LocalDate localDate = LocalDate.*now*();  
 **if** (localDate.getDayOfMonth() == 1) {  
 **try** {  
 *bikeRepair*();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 **public static void** bikeRepair() **throws** SQLException {  
 String query = **"SELECT bike\_id,use\_time,last\_address FROM bike.bike WHERE use\_time>200.0"**;  
 ResultSet resultSet = MyConnection.*myQuery*(query);  
 String insert = **"INSERT INTO bike.bike\_repair(bike\_id, use\_time, last\_address) VALUES (?,?,?)"**;  
 PreparedStatement preparedStatement = MyConnection.*getConnection*().prepareStatement(insert);  
 **while** (resultSet.next()) {  
 preparedStatement.setInt(1,resultSet.getInt(1));  
 preparedStatement.setDouble(2,resultSet.getDouble(2));  
 preparedStatement.setString(3,resultSet.getString(3));  
 preparedStatement.executeUpdate();  
 }  
 MyConnection.*getConnection*().commit();  
 }  
}

## 5.

1. create view

2. add index 对where需要查询的字段添加索引，例如record的还车时间添加索引

3. 对sql查询语句进行优化，将条件更加严格的语句放在前面。