|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **□ 수행평가-빅데이터를 활용한 IoT 시스템 개발(feat.커넥티드카)** | | | | | | |
|  |  |  | |  | |  |
| **과정명** | | 빅데이터를 활용한 IoT 시스템 개발(feat.커넥티드카) | | | | |
| **교과목명** | | IoT 운영시스템 구축 기반기술 | | **훈련교사** | | 이진만 |
| **과정명** | | SW기초기술이해  응용SW기술이해  Database 구문이해 | | | | |
| **수행날짜** | | 2019.06.10 | 훈련생명 | | 김소희 | |
| **과제개요** | | | | | | |
| 1. **ERD**      1. **DDL, DML**   **--DDL**  CREATE TABLE FACTORY  (FACTNO VARCHAR2(5),  FACNAME VARCHAR2(14),  FACLOC VARCHAR2(13))  CREATE TABLE PRODUCTS  (PDNO NUMBER,  PDNAME VARCHAR2(10),  PDSUBNAME VARCHAR2(10),  FACTNO VARCHAR2(5),  PDDATE DATE,  PDCOST NUMBER,  PDPRICE NUMBER,  PDAMOUNT NUMBER)  **--DML**  ALTER TABLE FACTORY ADD PRIMARY KEY (FACTNO)  INSERT INTO FACTORY VALUES ('10', 'A', 'SEOUL')  INSERT INTO FACTORY VALUES ('20', 'B', 'LONDON')  INSERT INTO FACTORY VALUES ('30', 'C', 'PARIS')  ALTER TABLE PRODUCTS ADD PRIMARY KEY (PDNO)  ALTER TABLE PRODUCTS ADD CONSTRAINT FACTNO FOREIGN KEY (FACTNO) REFERENCES FACTORY (FACTNO)  INSERT INTO PRODUCTS VALUES (001, 'PC', 'PC01', 10, '06/10/2019', 10, 100, 2)  INSERT INTO PRODUCTS VALUES (002, 'TV', 'TV01', 20, '06/11/2019', 15, 150, 3)  INSERT INTO PRODUCTS VALUES (003, 'REF', 'REF01', 30, '06/12/2019', 20, 200, 1)  INSERT INTO PRODUCTS VALUES (004, 'VAC', 'VAC01', 30, '06/13/2019', 25, 250, 1)   1. **UML**      1. **작성 코드** 2. **SQL**   **package** frame;  **public** **class** Sql {  **public** **static** String *insertFactory* = "INSERT INTO FACTORY VALUES (?,?,?)";  **public** **static** String *deleteFactory* = "DELETE FROM FACTORY WHERE FACTNO = ?";  **public** **static** String *updateFactory* = "UPDATE FACTORY SET FACNAME = ?, FACLOC = ? WHERE FACTNO = ?";  **public** **static** String *selectFactory* = "SELECT \* FROM FACTORY WHERE FACTNO = ?";  **public** **static** String *selectAllFactory* = "SELECT \* FROM FACTORY";    **public** **static** String *insertProducts* = "INSERT INTO PRODUCTS VALUES (?,?,?,?,?,?,?,?)";  **public** **static** String *deleteProducts* = "DELETE FROM PRODUCTS WHERE PDNO = ?";  **public** **static** String *updateProducts* = "UPDATE PRODUCTS SET PDNAME =?, PDSUBNAME =?, FACTNO =?, PDDATE=?, PDCOST =?, PDPRICE =?, PDAMOUNT =? WHERE PDNO =?";  **public** **static** String *selectProducts* = "SELECT \* FROM PRODUCTS WHERE PDNO = ?";  **public** **static** String *selectAllUProducts* = "SELECT \* FROM PRODUCTS";  }   1. **Factory**   package vo;  public class Factory {  String FactNo ;  String FactName;  String FacLoc;  public Factory(String factNo, String factName, String facLoc) {  FactNo = factNo;  FactName = factName;  FacLoc = facLoc;  }  public String getFactNo() {  return FactNo;  }  public void setFactNo(String factNo) {  FactNo = factNo;  }  public String getFactName() {  return FactName;  }  public void setFactName(String factName) {  FactName = factName;  }  public String getFacLoc() {  return FacLoc;  }  public void setFacLoc(String facLoc) {  FacLoc = facLoc;  }  @Override  public String toString() {  return "Factory [FactNo=" + FactNo + ", FactName=" + FactName + ", FacLoc=" + FacLoc + "]";  }  }   1. **Products**   package vo;  import java.util.Date;  public class Products {  int PdNo;  String PdName;  String PdsUBName;  String FactNo;  Date PdDate;  int PdCost;  int PdPrice;  int PdAmount;  public Products(int pdNo, String pdName, String pdsUBName, String factNo, Date pdDate, int pdCost, int pdPrice,  int pdAmount) {  PdNo = pdNo;  PdName = pdName;  PdsUBName = pdsUBName;  FactNo = factNo;  PdDate = pdDate;  PdCost = pdCost;  PdPrice = pdPrice;  PdAmount = pdAmount;  }  public int getPdNo() {  return PdNo;  }  public void setPdNo(int pdNo) {  PdNo = pdNo;  }  public String getPdName() {  return PdName;  }  public void setPdName(String pdName) {  PdName = pdName;  }  public String getPdsUBName() {  return PdsUBName;  }  public void setPdsUBName(String pdsUBName) {  PdsUBName = pdsUBName;  }  public String getFactNo() {  return FactNo;  }  public void setFactNo(String factNo) {  FactNo = factNo;  }  public Date getPdDate() {  return PdDate;  }  public void setPdDate(Date pdDate) {  PdDate = pdDate;  }  public int getPdCost() {  return PdCost;  }  public void setPdCost(int pdCost) {  PdCost = pdCost;  }  public int getPdPrice() {  return PdPrice;  }  public void setPdPrice(int pdPrice) {  PdPrice = pdPrice;  }  public int getPdAmount() {  return PdAmount;  }  public void setPdAmount(int pdAmount) {  PdAmount = pdAmount;  }  @Override  public String toString() {  return "Products [PdNo=" + PdNo + ", PdName=" + PdName + ", PdsUBName=" + PdsUBName + ", FactNo=" + FactNo  + ", PdDate=" + PdDate + ", PdCost=" + PdCost + ", PdPrice=" + PdPrice + ", PdAmount=" + PdAmount + "]";  }  }   1. **Dao**   package frame;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  public abstract class Dao<K, V> {  public void close(PreparedStatement con) {  if(con !=null) {  try {  con.close();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }  public void close(ResultSet con) {  if(con !=null) {//null 이면 어떠한 함수도 호출 할 수 없어서 오류!  try {  con.close();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }  public abstract void insert(V v,Connection con) throws Exception;  public abstract void delete(K k,Connection con) throws Exception;  public abstract void update(V v,Connection con) throws Exception;  public abstract V select(K k,Connection con) throws Exception;  public abstract ArrayList<V> select(Connection con) throws Exception;  }   1. **Biz**   package frame;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  import java.util.ArrayList;  public abstract class Biz<K,V> {  String id;  String pwd;  String url;  public Biz() {  try {  Class.forName("oracle.jdbc.driver.OracleDriver");  } catch (ClassNotFoundException e) {  System.out.println("Driver Loading Error...");  }  id = "db";  pwd = "db";  url = "jdbc:oracle:thin:@70.12.50.229:1521:xe";  }  public Connection getCon() {  Connection con = null;  try {  con = DriverManager.getConnection(url, id, pwd);  con.setAutoCommit(false);//transcation 내가 관리 할거야 ~(Commit , Rollback)  } catch (SQLException e) {  e.printStackTrace();  }  return con;  }  public void close(Connection con) {  if (con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  public abstract void register(V v) throws Exception;  public abstract void remove(K k) throws Exception;  public abstract void modify(V v) throws Exception;  public abstract V get(K k) throws Exception;  public abstract ArrayList<V> get() throws Exception;  }   1. **FactoryBiz**   package com;  import java.sql.Connection;  import java.util.ArrayList;  import frame.Biz;  import frame.Dao;  import vo.Factory;  public class FactoryBiz extends Biz<String, Factory> {  Dao<String, Factory> dao = new FactoryDao();    @Override  public void register(Factory v) throws Exception {    Connection con = null;  try {  con = getCon();  dao.insert(v,con);  //dao.insert(v,con);  con.commit();//위에 둘다 정상이면 commit  } catch (Exception e) {  con.rollback();//비정상이면 rollback  throw e;  } finally {  close(con);  }    }  @Override  public void remove(String k) throws Exception {  Connection con = null;  try {  con = getCon();  dao.delete(k, con);  con.commit();  }catch(Exception e){  con.rollback();  throw e;  }finally {  close(con);  }    }  @Override  public void modify(Factory v) throws Exception {  Connection con = null;  try {  con = getCon();  dao.update(v, con);  con.commit();  }catch(Exception e){  con.rollback();  throw e;  }finally {  close(con);  }    }  @Override  public Factory get(String k) throws Exception {  Connection con = null;  Factory u = null;  try {  con = getCon();  u = dao.select(k,con);  } catch (Exception e) {  throw e;  } finally {  close(con);  }  return u;  }  @Override  public ArrayList<Factory> get() throws Exception {    Connection con = null;  ArrayList<Factory> list = new ArrayList<>() ;    try {  con = getCon();  list = dao.select(con);    }catch(Exception e) {  throw e;  }finally {  close(con);  }  return list;  }  }   1. **FactoryDao**   package com;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  import frame.Dao;  import frame.Sql;  import vo.Factory;  public class FactoryDao extends Dao<String, Factory> {  @Override  public void insert(Factory v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.insertFactory);  pstmt.setString(1,v.getFactNo());  pstmt.setString(2,v.getFactName());  pstmt.setString(3,v.getFacLoc());    pstmt.executeUpdate();    }  catch(Exception e) {  throw e;  }  finally {  close(pstmt);  }    }  @Override  public void delete(String k, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.deleteFactory);  pstmt.setString(1,k);  pstmt.executeUpdate();    }  catch(Exception e) {  throw e;  }  finally {  close(pstmt);  }    }  @Override  public void update(Factory v, Connection con) throws Exception {    PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.updateFactory);  pstmt.setString(1,v.getFactName());  pstmt.setString(2,v.getFacLoc());  pstmt.setString(3,v.getFactNo());    }  catch(Exception e) {  throw e;  }  finally {  close(pstmt);  }    }  @Override  public Factory select(String k, Connection con) throws Exception {      PreparedStatement pstmt = null;  ResultSet rset = null;  Factory facotry = null;    try {  pstmt = con.prepareStatement(Sql.selectFactory);  pstmt.setString(1, k);  rset = pstmt.executeQuery();    rset.next();  String uid = rset.getString("FACTNO");  String upwd = rset.getString("FACNAME");  String uname = rset.getString("FACLOC");    facotry = new Factory(uid,upwd,uname);    } catch (Exception e) {  throw e;  } finally {  close(pstmt);  close(rset);  }  return facotry;  }  @Override  public ArrayList<Factory> select(Connection con) throws Exception {    PreparedStatement pstmt = null;  ResultSet rset = null;  ArrayList<Factory> list = new ArrayList<>();    try {  pstmt = con.prepareStatement(Sql.selectAllFactory);  rset = pstmt.executeQuery();    while (rset.next()) {  String uid = rset.getString("FACTNO");  String upwd = rset.getString("FACNAME");  String uname = rset.getString("FACLOC");    list.add(new Factory(uid,upwd,uname));  }  } catch (Exception e) {  throw e;  } finally {  close(pstmt);  close(rset);  }  return list;  }  }   1. **ProductsBiz**   package com;  import java.sql.Connection;  import java.util.ArrayList;  import javax.xml.bind.ParseConversionEvent;  import frame.Biz;  import frame.Dao;  import vo.Factory;  import vo.Products;  public class ProductsBiz extends Biz<Integer, Products> {  Dao<Integer, Products> dao;  FactoryBiz fb = new FactoryBiz();  public ProductsBiz() {  dao = new ProductsDao();  }  @Override  public void register(Products v) throws Exception {  Connection con = null;  ArrayList<Factory> rset = new ArrayList<>();  rset = fb.get();  boolean notFactory = false;  try {  con = getCon();  dao.insert(v, con);  for (int i = 0; i < rset.size(); i++) {  if (rset.get(i).getFactNo() == v.getFactNo()) {  notFactory = true;  con.commit();  }  }  } catch (Exception e) {  System.out.println("없는 공장을 입력했습니다.");  con.rollback();// 비정상이면 rollback  throw e;  } finally {  close(con);  }  }  @Override  public void remove(Integer k) throws Exception {  Connection con = null;  try {  con = getCon();  dao.delete(k, con);  con.commit();  } catch (Exception e) {  con.rollback();  throw e;  } finally {  close(con);  }  }  @Override  public void modify(Products v) throws Exception {  Connection con = null;  ArrayList<Factory> rset = new ArrayList<>();  rset = fb.get();  boolean notFactory = false;  try {  con = getCon();  dao.update(v, con);  for (int i = 0; i < rset.size(); i++) {  if (rset.get(i).getFactNo() == v.getFactNo()) {    notFactory = true;  con.commit();  }  }      } catch (Exception e) {  System.out.println("없는 공장을 입력했습니다.");  con.rollback();  throw e;  } finally {  close(con);  }  }  @Override  public Products get(Integer k) throws Exception {  Connection con = null;  Products u = null;  Factory d = null;      ArrayList<Factory> rset = new ArrayList<>();  rset = fb.get();    boolean notFactory = false;  try {    con = getCon();  u = dao.select(k, con);  con.commit();  } catch (Exception e) {  System.out.println("없는 품목을 입력했습니다.");  con.rollback();  throw e;  } finally {  close(con);  }  return u;  }  @Override  public ArrayList<Products> get() throws Exception {  Connection con = null;  Products u = null;  ArrayList<Products> list = new ArrayList<>();  try {  con = getCon();  list = dao.select(con);  } catch (Exception e) {  throw e;  } finally {  close(con);  }  return list;  }  }   1. **ProductsDao**   package com;  import java.sql.Connection;  import java.sql.Date;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  import frame.Dao;  import frame.Sql;  import vo.Factory;  import vo.Products;  public class ProductsDao extends Dao<Integer, Products> {  @Override  public void insert(Products v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.insertProducts);  pstmt.setInt(1, v.getPdNo());  pstmt.setString(2, v.getPdName());  pstmt.setString(3, v.getPdsUBName());  pstmt.setString(4, v.getFactNo());  pstmt.setDate(5, (Date) v.getPdDate());  pstmt.setInt(6, v.getPdCost());  pstmt.setInt(7, v.getPdPrice());  pstmt.setInt(8, v.getPdAmount());  pstmt.executeUpdate();  } catch (Exception e) {  throw e;// Exception으로 던짐  } finally {  close(pstmt);// 반드시 실행 될 수 있도록 ~  }  }  @Override  public void delete(Integer k, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.deleteProducts);  pstmt.setInt(1, k);  pstmt.executeUpdate();  } catch (Exception e) {  throw e;// Exception으로 던짐  } finally {  close(pstmt);// 반드시 실행 될 수 있도록 ~  }  }  @Override  public void update(Products v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.updateProducts);  pstmt.setString(1, v.getPdName());  pstmt.setString(2, v.getPdsUBName());  pstmt.setString(3, v.getFactNo());  pstmt.setDate(4, (Date)v.getPdDate());  pstmt.setInt(5, v.getPdCost());  pstmt.setInt(6, v.getPdPrice());  pstmt.setInt(7, v.getPdAmount());  pstmt.setInt(8, v.getPdNo());  pstmt.executeUpdate();  } catch (Exception e) {  throw e;  } finally {  close(pstmt);  }    }  @Override  public Products select(Integer k, Connection con) throws Exception {  PreparedStatement pstmt = null;  ResultSet rset = null;  Products list = null;  try {  pstmt = con.prepareStatement(Sql.selectProducts);  pstmt.setInt(1, k);  rset = pstmt.executeQuery();// ORACLE 로직에 접근  rset.next();// 한 칸 옮겨야함 ,빈공간을 처음에 가르키고 있어서  int uid = rset.getInt("PDNO");  String upwd = rset.getString("PDNAME");  String uname = rset.getString("PDSUBNAME");  String uno = rset.getString("FACTNO");  Date upddate = rset.getDate("PDDATE");  int updcost = rset.getInt("PDCOST");  int updprice = rset.getInt("PDPRICE");  int updamount = rset.getInt("PDAMOUNT");  list = new Products(uid, upwd, uname, uno, upddate, updcost, updprice, updamount);  System.out.println(list);  } catch (Exception e) {  throw e;  } finally {  close(pstmt);  close(rset);  }  return list;  }  @Override  public ArrayList<Products> select(Connection con) throws Exception {  PreparedStatement pstmt = null;  ResultSet rset = null;  ArrayList<Products> list = new ArrayList<>();  try {  pstmt = con.prepareStatement(Sql.selectAllUProducts);  rset = pstmt.executeQuery();  while (rset.next()) {  int uid = rset.getInt("PDNO");  String upwd = rset.getString("PDNAME");  String uname = rset.getString("PDSUBNAME");  String uno = rset.getString("FACTNO");  Date upddate = rset.getDate("PDDATE");  int updcost = rset.getInt("PDCOST");  int updprice = rset.getInt("PDPRICE");  int updamount = rset.getInt("PDAMOUNT");  list.add(new Products(uid, upwd, uname, uno, upddate, updcost, updprice, updamount));  System.out.println("["+ uid + " " + upwd + " " + uname + " " + uno + " " + upddate+ " "+ updcost + " " + updprice +" "+ updamount+"]");  }        } catch (Exception e) {  throw e;  } finally {  close(pstmt);  close(rset);  }  return list;  }  } | | | | | | |