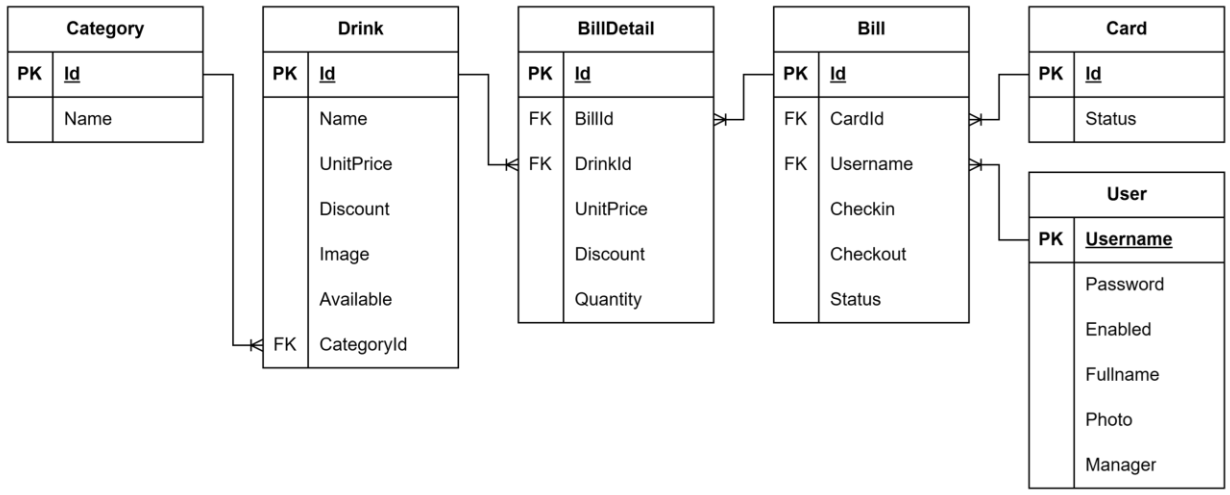
BÀI 1: DATABASE

Tạo CSDL PolyCafe có sơ đồ quan hệ như hình sau




```
CREATE TABLE Categories(
   Id NVARCHAR(20) NOT NULL,
   Name NVARCHAR(50) NOT NULL,
   PRIMARY KEY(Id)
)

CREATE TABLE Drinks(
   Id NVARCHAR(20) NOT NULL,
   Name NVARCHAR(50) NOT NULL,
   UnitPrice FLOAT NOT NULL,
   Discount FLOAT NOT NULL,
   Image NVARCHAR(50) NOT NULL,
   Available BIT NOT NULL,
   CategoryId NVARCHAR(20) NOT NULL,
   PRIMARY KEY(Id),
   FOREIGN KEY(CategoryId) REFERENCES Categories(Id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)

CREATE TABLE Cards(
   Id INT NOT NULL,
   Status INT NOT NULL,
   PRIMARY KEY(Id)
)

CREATE TABLE Users(
   Username NVARCHAR(20) NOT NULL,
```

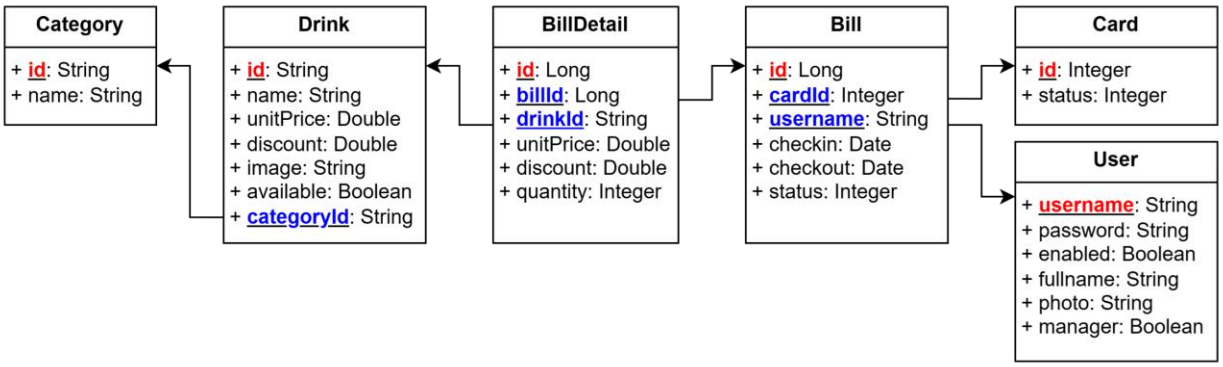
```
    Password NVARCHAR(50) NOT NULL,
    Enabled BIT NOT NULL,
    Fullname NVARCHAR(50) NOT NULL,
    Photo NVARCHAR(50) NOT NULL,
    Manager BIT NOT NULL,
    PRIMARY KEY(Username)
)

CREATE TABLE Bills(
    Id BIGINT NOT NULL IDENTITY(10000, 1),
    Username NVARCHAR(20) NOT NULL,
    CardId INT NOT NULL,
    Checkin DATETIME NOT NULL,
    Checkout DATETIME NULL,
    Status INT NOT NULL,
    PRIMARY KEY(Id),
    FOREIGN KEY(Username) REFERENCES Users(Username)
        ON UPDATE CASCADE,
    FOREIGN KEY(CardId) REFERENCES Cards(Id)
        ON UPDATE CASCADE
)

CREATE TABLE BillDetails(
    Id BIGINT NOT NULL IDENTITY(100000, 1),
    BillId BIGINT NOT NULL,
    DrinkId NVARCHAR(20) NOT NULL,
    UnitPrice FLOAT NOT NULL,
    Discount FLOAT NOT NULL,
    Quantity INT NOT NULL,
    PRIMARY KEY(Id),
    FOREIGN KEY(BillId) REFERENCES Bills(Id)
        ON DELETE CASCADE,
    FOREIGN KEY(DrinkId) REFERENCES Drinks(Id)
        ON UPDATE CASCADE
)
```

BÀI 2: ENTITY CLASS

Tạo các lớp thực thể trong package poly.entity để mô tả cấu trung dữ liệu của CSDL theo sơ đồ tổ chức như sau




```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class Category {
   private String id;
   private String name;
}
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class Drink {
   private String id;
   private String name;
   @Builder.Default
   private String image = "product.png";
   private double unitPrice;
   private double discount;
   private boolean available;
   private String categoryId;
}
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class User {
   private String username;
   private String password;
   private boolean enabled;
   private String fullname;
   @Builder.Default
   private String photo = "photo.png";
   private boolean manager;
```

```
}
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class Card {
   private Integer id;
   private int status;
}
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class Bill {
   private Long id;
   private String username;
   private Integer cardId;
   @Builder.Default
   private Date checkin = new Date();
   private Date checkout;
   private int status;
}
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class BillDetail {
   private Long id;
   private Long billId;
   private String drinkId;
   private double unitPrice;
   private double discount;
   private int quantity;
}
```

BÀI 3: SỬ DỤNG CÁC LỚP TIỆN ÍCH XJDBC VÀ XQUERY

3.1 – Khai báo thông số kết nối đến CSDL trong XJdbc.openConnection()

```
public static Connection openConnection() {
      var driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
      var dburl =
"jdbc:sqlserver://localhost;database=PolyCafe;encrypt=true;trustServerCertificate=true;";
      var username = "sa";
      var password = "******";
      …
   }
```

3.2 – Sử dụng XJdbc để thực hiện truy xuất dữ liệu

Tạo lớp Test chứa phương thức main() và thực hiện các đoạn mã sau để truy xuất dữ liệu từ bảng Categories

- Thêm mới

```
String sql = "INSERT INTO Categories (Id, Name) VALUES(?, ?)";
XJdbc.executeUpdate(sql, "C01", "Loại 1");
XJdbc.executeUpdate(sql, "C02", "Loại 2");
```

- Truy vấn nhiều bản ghi

```
String sql = "SELECT * FROM Categories WHERE Name LIKE ?";
ResultSet rs = XJdbc.executeQuery(sql, "%Loại%");
```

- Truy xuất nhiều bản ghi và chuyển đổi sang List<Bean>

```
String sql = "SELECT * FROM Categories WHERE Name LIKE ?";
List<Category> beans = XJdbc.getBeanList(Category.class, sql, "%Loại%");
```

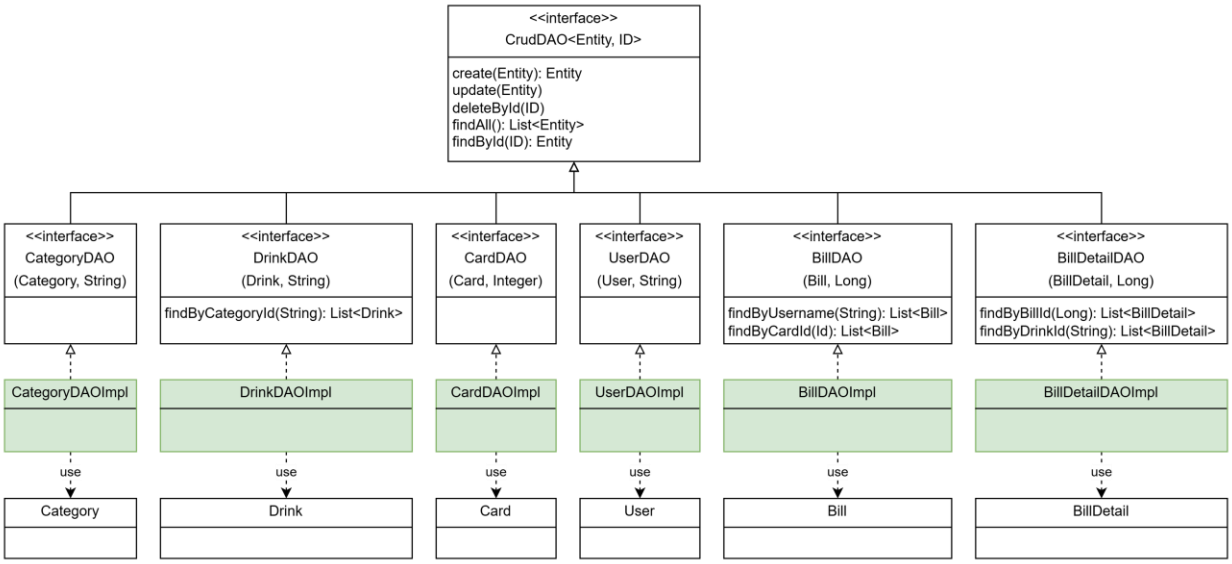
- Truy xuất một bản ghi và chuyển đổi sang Bean

```
String sql = "SELECT * FROM Categories WHERE Id=?";
Category bean = XJdbc.getSingleBean(Category.class, sql, "C02");
```

- Truy vấn 1 giá trị

```
String sql = "SELECT max(Id) FROM Categories WHERE Name LIKE ?";
String id = XJdbc.getValue(sql, "%Loại%");
```

BÀI 4: TẠO CÁC LỚP TRUY XUẤT DỮ LIỆU CƠ BẢN ĐƯỢC TỔ CHỨC THEO SƠ ĐỒ LỚP SAU:



4.1 – Tạo các interface khai báo các phương thức truy xuất dữ liệu

- Generic DAO: Truy xuất dữ liệu tổng quát

```
public interface CrudDAO<T, ID> {
   T create(T entity);
   void update(T entity);
   void deleteById(ID id);
   List<T> findAll();
   T findById(ID id);
}
```

- Specific DAO: Truy xuất dữ liệu cụ thể

```
public interface CategoryDAO extends CrudDAO<Category, String>{
}
```

```
public interface DrinkDAO extends CrudDAO<Drink, String>{
   List<Drink> findByCategoryId(String categoryId);
}
```

```
public interface UserDAO extends CrudDAO<User, String>{
}
```

```
public interface CardDAO extends CrudDAO<Card, Integer>{
}
```

```
public interface BillDAO extends CrudDAO<Bill, Long>{
   List<Bill> findByUsername(String username);
   List<Bill> findByCardId(Integer cardId);
}
```

```
public interface BillDetailDAO extends CrudDAO<BillDetail, Long>{
   List<BillDetail> findByBillId(Long billId);
   List<BillDetail> findByDrinkId(String drinkId);
```

```
}
```

4.2 – Cài đặt mã nguồn cho các lớp truy xuất dữ liệu

```
public class CategoryDAOImpl implements CategoryDAO {

   String createSql = "INSERT INTO Categories(Id, Name) VALUES(?, ?)";
   String updateSql = "UPDATE Categories SET Name=? WHERE Id=?";
   String deleteSql = "DELETE FROM Categories WHERE Id=?";
   String findAllSql = "SELECT * FROM Categories";
   String findByIdSql = "SELECT * FROM Categories WHERE Id=?";

   @Override
   public Category create(Category entity) {
      Object[] values = {
         entity.getId(),
         entity.getName()
      };
      XJdbc.executeUpdate(createSql, values);
      return entity;
   }
   @Override
   public void update(Category entity) {
      Object[] values = {
         entity.getName(),
         entity.getId()
      };
      XJdbc.executeUpdate(updateSql, values);
   }
   @Override
   public void deleteById(String id) {
      XJdbc.executeUpdate(deleteSql, id);
   }
   @Override
   public List<Category> findAll() {
      return XQuery.getEntityList(Category.class, findAllSql);
   }
   @Override
   public Category findById(String id) {
      return XQuery.getSingleBean(Category.class, findByIdSql, id);
   }
}
```

```
public class UserDAOImpl implements UserDAO {
   String createSql = "...";
   String updateSql = "...";
   String deleteSql = "...";
```

```
    String findAllSql = "...";
    String findByIdSql = "...";

    @Override
    public User create(User entity) {...}
    @Override
    public void update(User entity) {...}
    @Override
    public void deleteById(String id) {...}
    @Override
    public List< User> findAll() {...}
    @Override
    public User findById(String id) {...}
}
```

```
public class CardDAOImpl implements CardDAO {
    String createSql = "...";
    String updateSql = "...";
    String deleteSql = "...";
    String findAllSql = "...";
    String findByIdSql = "...";

    @Override
    public Card create(Card entity) {...}
    @Override
    public void update(Card entity) {...}
    @Override
    public void deleteById(String id) {...}
    @Override
    public List< Card> findAll() {...}
    @Override
    public Card findById(String id) {...}
}
```

```
public class DrinkDAOImpl implements DrinkDAO {
    String createSql = "...";
    String updateSql = "...";
    String deleteSql = "...";
    String findAllSql = "...";
    String findByIdSql = "...";

    String findByCategoryIdSql = "SELECT * FROM Drinks WHERE CategoryId=?";

    @Override
    public Drink create(Drink entity) {...}
    @Override
    public void update(Drink entity) {...}
```

```
    @Override
    public void deleteById(String id) {…}
    @Override
    public List< Drink> findAll() {…}
    @Override
    public Drink findById(String id) {…}

    @Override
    public List<Drink> findByCategoryId(String categoryId) {
        return XQuery.getBeanList(Drink.class, findByCategoryIdSql, categoryId);
    }
}
```

```
public class BillDAOImpl implements BillDAO {
    String createSql = "…";
    String updateSql = "…";
    String deleteSql = "…";
    String findAllSql = "…";
    String findByIdSql = "…";

    String findByUsernameSql = "…";
    String findByCardIdSql = "…";

    @Override
    public Bill create(Bill entity) {…}
    @Override
    public void update(Bill entity) {…}
    @Override
    public void deleteById(String id) {…}
    @Override
    public List< Bill> findAll() {…}
    @Override
    public Bill findById(String id) {…}

    @Override
    public List<Bill> findByUsername(String username) {…}
    @Override
    public List<Bill> findByCardId(Integer cardId) {…}
}
```

```
public class BillDetailDAOImpl implements BillDetailDAO {
    String createSql = "…";
    String updateSql = "…";
    String deleteSql = "…";
    String findAllSql = "…";
    String findByIdSql = "…";
```

```
    String findByBillIdSql = "…";
    String findByDrinkIdSql = "…";

    @Override
    public BillDetail create(BillDetail entity) {…}
    @Override
    public void update(BillDetail entity) {…}
    @Override
    public void deleteById(Long id) {…}
    @Override
    public List<BillDetail> findAll() {…}
    @Override
    public BillDetail findById(Long id) {…}

    @Override
    public List<BillDetail> findByBillId(Long billId) {…}
    @Override
    public List<BillDetail> findByDrinkId(String drinkId) {…}
}
```