数据库原理 第五次实验报告

09019106 牟倪

目录

一、应用程序概述

1.1 应用场景

特展作为展品的全新组织形式，将具有某些相关性的展品（如同一年代、同一主题、同一形式）进行集中展览，一般由第三方主办，特定博物馆承办。一场精彩的特展，不仅聚集了丰富的展品，更聚集了大量领域爱好者甚至专家；人们不仅可以对展品大饱眼福，更可以了解领域的历史沿革，享受观点与思维的碰撞。

对特展从组织到展览的全过程进行考察，如下需求浮出水面：

- 在组织者的角度，举办一场特展需要确定展览地点和展品，需要统一的管理系统来记录特展的地点和展品信息。
- 在博物馆方的角度，如果没有统一的管理系统，很难考察文物在特定时间的所在地，进而很难确定文物在某段时间是否已经被特展借出、是否可以接受新的特展预定请求。
- 在游客的角度，游客可能希望参与特定主题的特展，希望参与位于特定城市/特定博物馆的特展，希望参与特定时间段的特展，希望去特展观看特定文物；建立统一的管理系统，就可以实现特展信息的多样化查询。同时，游客可能希望了解特展的口碑和人气，管理系统可以记录游客对特展的评价，并将评价进行展示以供其他游客参考。

因此，我们考虑建立统一的管理系统，对特展信息进行管理。

1.2 主要需求

- 特展组织者角度：
  - 高效管理（增删改查）特展信息（名称、地点、场所（博物馆）、起始时间、展品等）。
- 博物馆角度：
  - 高效管理馆藏文物信息，记录文物出展的时间表。
  - 对特展请求进行筛选，判断特展是否可以预定文物。
- 游客角度：
  - 对特展信息进行可视化展示。
  - 通过主题/城市/博物馆/日期/文物，查询特展信息。
  - 对特展进行评价。
  - 查看其他用户对特展的评价。

二、数据库设计

2.1 实体与关系设计

设计如下实体：

- 管理员：管理系统的管理员，具有 id、密码属性。
- 用户：普通用户，具有 id、账号、密码属性。
- 博物馆：具有博物馆 id、名称、简介、图片、所在城市属性。
- 文物：具有文物 id、名称、所属博物馆 id 属性。
- 特展：具有特展 id、名称、开始日期、结束日期、简介、图片、所在博物馆 id 属性。

设计如下关系：

- <博物馆>拥有<文物>：1 – N。
- <博物馆>举办<特展>：M – N。
- <特展>展出<文物>：1 – N，在给定时刻，文物最多被一个特展展出。
- <用户>喜欢<特展>：M – N。
- <管理员>管理<特展>：M – N。

2.2 数据库模式（database schema）

database schema 如下所示（标*的属性为主键）：

```
admin (
    *adminid: auto-increment unsigned int,
    password: non-null varchar(255)
)
exhibition (
    *eid: auto-increment unsigned int,
    ename: non-null varchar(255),
    start_date: sql date,
    end_date: sql date,
    eintro: non-null varchar(255),
    epicture: non-null varchar(255),
    mid: unsigned int
)
likelist (
    *uid: unsigned int,
    *eid: unsigned int
)
```

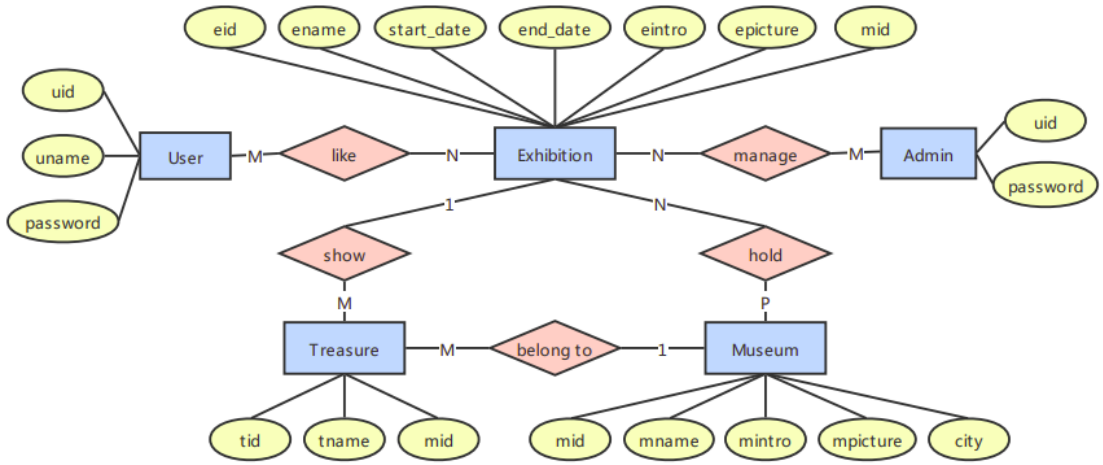
```
museum (
    *mid: auto-increment unsigned int,
    mname: non-null varchar(255),
    mintro: non-null varchar(255),
    mpicture: non-null varchar(255),
    city: non-null varchar(255)
)
showlist (
    *eid: unsigned int,
    *tid: unsigned int
)
treasure (
    *tid: auto-increment unsigned int,
    tname: non-null varchar(255),
    mid: unsigned int
)
user (
    *uid: auto-increment unsigned int,
    uname: non-null varchar(255),
    password: non-null varchar(255)
)
```

其中，

- `showlist` 表：`eid` 属性为关联到 `exhibition` 表的外键、`tid` 属性为关联到 `treasure` 表的外键。
- `likelist` 表：`uid` 属性为关联到 `user` 表的外键、`eid` 属性为关联到 `exhibition` 表的外键。

该 data schema 依据“一事一地”原则设计，不存在部分函数依赖，不存在属性对主键的传递依赖，属于关系数据库的第三范式。

ER 图如下所示：



三、功能设计

应用程序与数据库交互的功能，大致分为以下 3 部分：

- 管理员 & 用户相关：
    - 通过管理员 id，查询密码。
    - 通过用户 id，查询密码。
    - 增删改 exhibition。
    - 增删改 showlist。
    - 增删 likelist。
    - 通过用户 id，查询用户所有喜欢特展的 id。
- 特展相关：
    - 查询所有特展 id+名称+开始时间+结束时间。
    - 通过特展名字，查询特展 id+名称+开始时间+结束时间。
    - 通过日期，查询特展 id+名称+开始时间+结束时间。
    - 通过城市，查询特展 id+名称+开始时间+结束时间。
    - 通过文物名称，查询特展 id+名称+开始时间+结束时间。
    - 通过博物馆名称，查询特展 id+名称+开始时间+结束时间。
    - 通过特展 id，查询博物馆 id+名称。
    - 通过特展 id，查询文物 id+名称。
- 博物馆相关：
    - 查询所有博物馆 id+名称。
    - 通过博物馆名字，查询博物馆 id+名称。
    - 通过城市，查询博物馆 id+名称。
    - 通过特展名称，查询博物馆 id+名称。
    - 通过文物名称，查询博物馆 id+名称。
    - 通过博物馆 id，查询文物 id+名称。
    - 通过博物馆 id，查询特展 id+名称+开始时间+结束时间。

由于该项目是我们小组选择的最终项目，因此小组成员对所有功能进行分工实现。队友叶若天负责的一部分，汤雯婧负责第三部分，我负责第二部分。

四、软件接口实现

4.1 SQL 语句编写

查询所有 特展 id+名称+开始时间+结束时间。	SELECT E.eid, E.ename, E.start_date, E.end_date from exhibition E
通过特展名字， 查询特展 id+名称+开始时间+结束时间。	SELECT E.eid, E.ename, E.start_date, E.end_date from exhibition E WHERE E.ename LIKE 'tempname '
通过日期， 查询特展 id+名称+开始时间+结束时间。	SELECT E.eid, E.ename, E.start_date, E.end_date from exhibition E WHERE 'date' BETWEEN E.start_date AND E.end_date
通过城市， 查询特展 id+名称+开始时间+结束时间。	SELECT E.eid, E.ename, E.start_date, E.end_date from exhibition E WHERE E.mid IN (SELECT M.mid FROM museum M WHERE M.city LIKE 'tempname')
通过文物名称， 查询特展 id+名称+开始时间+结束时间。	SELECT E.eid, E.ename, E.start_date, E.end_date FROM exhibition E WHERE E.eid IN (SELECT S.eid FROM show S WHERE S.tid IN (SELECT T.tid FROM treasure T WHERE T.tname LIKE 'tempname'))
通过博物馆名称， 查询特展 id+名称+开始时间+结束时间。	SELECT E.eid, E.ename, E.start_date, E.end_date FROM exhibition E WHERE E.mid IN (SELECT M.mid FROM museum M WHERE M.mname LIKE 'tempname')
通过特展 id， 查询博物馆 id+名称。	SELECT M.mid, M.mname FROM museum M WHERE M.mid IN (SELECT M.mid FROM exhibition E WHERE E.eid = eid)
通过特展 id， 查询文物 id+名称。	SELECT T.tid, T.tname, T.mid FROM treasure T WHERE T.tid IN (SELECT S.tid FROM showlist S WHERE S.eid = eid)

4.2 接口代码编写思路

4.2.1 建立数据库连接

与数据库交互，首先需要建立数据库连接。关键代码如下：

```
static { // 只做一次
    try {
        Class.forName(driver);
        con = DriverManager.getConnection(url, user, password);
        if (!con.isClosed()) {
//                System.out.println("数据库连接成功");
        }
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
        System.out.println("数据库驱动没有安装");
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("数据库连接或操作失败");
    }
}
```

执行 SQL 查询并获得执行结果的关键代码如下：

```
public ResultSet search(String sql) { // 数据库查询操作，sql 为数据库操作指令
    ResultSet resultSet = null;
    try {
        resultSet = this.statement.executeQuery(sql);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return resultSet;
}
```

执行 SQL 增删改的关键代码如下：

```
public void adm(String sql) { //adm 为add delete modify 增删改的意思
    try {
        statement.executeUpdate(sql);
    } catch (SQLException e) {
        e.printStackTrace();
```

```
6.      }
7. }
```

这些代码被封装在 DbOperation 类中，DbOperation 类用来与数据库建立连接、交换信息。

4.2.2 与数据库交换信息

利用 DbOperation 类，与数据库交换信息。基本流程如下：

- 声明新的数据库连接。
- 执行 SQL 语句，接收返回结果。
- 遍历并解析 SQL 语句的返回结果。

以查询所有特展的 id+名称+开始日期+结束日期为例：

```
1. public static List<Exhi_info3> getAllExhibition() throws SQLExcep
   tion {
2.     DbOperation db = new DbOperation();
3.     List<Exhi_info3> lst = new ArrayList<>();
4.     ResultSet res = db.search("SELECT E.eid, E.ename, E.start_dat
   e, E.end_date from exhibition E");
5.     String eid = null, ename = null, start_date = null, end_date
   = null;
6.     while (res.next()) {
7.         eid = res.getString("eid");
8.         ename = res.getString("ename");
9.         start_date = res.getString("start_date");
10.        end_date = res.getString("end_date");
11.        Exhi_info3 temp = new Exhi_info3();
12.        temp.setEid(eid);
13.        temp.setEname(ename);
14.        temp.setStart_date(start_date);
15.        temp.setEnd_date(end_date);
16.        lst.add(temp);
17.    }
18.    return lst;
19.}
```

4.3 实现模糊搜索

当通过特展名称/文物名称查询特展时，模糊搜索（而非精确查找）显然更符合实际应用场景。

可以利用SQL的`LIKE`运算符实现模糊搜索。得到查询关键词后，在字符串首尾、每个字与字之间都插入一个'%'（如 "123" → "%1%2%3" ），以实现正则表达式的匹配。

关键代码如下：

```
1. String tempname = "%";
2. for (int i = 0; i < ename.length(); ++i) {
3.     tempname = tempname + origin.charAt(i) + "%";
4. }
```

五、软件接口测试

对特展相关的软件接口进行测试。数据库数据如下：

Exhibition：

eid	ename	start_date	end_date	eintro	epicture	mid
1	国家宝藏-上海站	2021-12-06	2021-12-16	国家宝藏上海站	/null	1
2	国家宝藏-北京站	2021-12-17	2021-12-27	国家宝藏北京站	/null	3
3	国家宝藏-南京站	2022-01-01	2021-12-11	国家宝藏南京站	/null	2
4	戏曲金陵	2021-12-01	2022-01-01	南京的戏曲文化	/null	2

Museum：

mid	mname	mintro	mpicture	city
1	上海博物馆	上海博物馆	/null	上海
2	南京博物院	南京博物院	/null	南京
3	北京博物馆	北京博物馆	/null	北京

Treasure：

tid	tname	mid
1	金陵戏曲面具	2
2	清明上河图	3
3	京剧戏曲面具	3
4	上海的宝藏	1

Showlist：

eid	tid
1	2
1	3
1	4
2	2
2	3
2	4
3	2
3	3
3	4
4	1
4	2
4	3

调用软件接口，分别进行以下操作

- 搜索所有特展；
- 按特展名称“国家宝藏”搜索特展；
- 按日期“2021-12-13”搜索特展；
- 按城市“南京”搜索特展；
- 按文物名称“面具”搜索特展；
- 按博物馆名称“博物馆”搜索特展；
- 用 特展 id=1 搜索博物馆；
- 用 特展 id=1 搜索文物。

实验结果如下所示：

```
E:\Java\jdk-11.0.9\bin\java.exe "-javaagent:E:\JetBrains\IntelliJ IDEA 2020.1.4\lib\idea_rt.jar
搜索所有特展：
eid: 1, ename: 国家宝藏-上海站, start_date: 2021-12-06 , end_date: 2021-12-16
eid: 2, ename: 国家宝藏-北京站, start_date: 2021-12-17 , end_date: 2021-12-27
eid: 3, ename: 国家宝藏-南京站, start_date: 2022-01-01 , end_date: 2021-12-11
eid: 4, ename: 戏曲金陵, start_date: 2021-12-01 , end_date: 2022-01-01
按特展名称“国家宝藏”搜索特展：
eid: 1, ename: 国家宝藏-上海站, start_date: 2021-12-06 , end_date: 2021-12-16
eid: 2, ename: 国家宝藏-北京站, start_date: 2021-12-17 , end_date: 2021-12-27
eid: 3, ename: 国家宝藏-南京站, start_date: 2022-01-01 , end_date: 2021-12-11
按日期“2021-12-13”搜索特展：
eid: 1, ename: 国家宝藏-上海站, start_date: 2021-12-06 , end_date: 2021-12-16
eid: 4, ename: 戏曲金陵, start_date: 2021-12-01 , end_date: 2022-01-01
按城市“南京”搜索特展：
eid: 3, ename: 国家宝藏-南京站, start_date: 2022-01-01 , end_date: 2021-12-11
eid: 4, ename: 戏曲金陵, start_date: 2021-12-01 , end_date: 2022-01-01
按文物名称“面具”搜索特展：
按博物馆名称“博物馆”搜索特展：
用 特展id=1 搜索博物馆：
mid: 1, mname: 上海博物馆
mid: 2, mname: 南京博物院
mid: 3, mname: 北京博物馆
用 特展id=1 搜索文物：
tid: 2, tname: 清明上河图, mid: 3
tid: 3, tname: 京剧戏曲面具, mid: 3
tid: 4, tname: 上海的宝藏, mid: 1

Process finished with exit code 0
```

运行结果正确，代码编写合理。

六、实验体会

在本次实验中，我们小组根据“一事一地”原则，仔细设计 entity 属性和关系，确保在没有数据冗余的情况下表格数目最少，data schema 也达到了三范式。我们还初步接触了 SQL 与 java 的通信过程，非常感谢 jdbc 对 java 连接数据库的强大封装，使得我们通过几行代码即可实现两门语言间的交互。