# 第五周学习笔记

06月14日

## 一：SQL 基本语句

### 1. 基本概念

数据库：操作数据的集合；

SQL 语句：管理数据库的命令，大小写不敏感；

DDL：数据定义语言，对数据库内部的对象进行创建；

DML：数据操纵语言，实现对数据库进行增删改查等操作；

DCL：数据控制语言，定义数据库相应对象的访问权限和安全级别。

### 2. 数据库相关

| 语句 | 语义 |
|---|---|
| **数据库操作：** | |
| Create database | 创建数据库 |
| Show create database | 查看数据库 |
| Show database | 查看所有数据库 |
| Create database db character set ‘’ | 创建指定字符集 |
| Drop database | 删除数据库 |
| Use | 使用数据库 |
| **表相关：** | |
| create table t1(id,name) | 创建表 |
| show create table t1 | 查看单个表属性 |
| show tables | 查看所有表 |
| create table t1(id,name) engine=‘’ charset= ‘’ | 创建表指定引擎和字符集 |
| **修改表：** | |
| rename table t1 to t2 | 修改表名 |
| alter table t1 engine= ‘’ charset= ‘’ | 修改表属性 |
| alter table t1 add age int first/after xx | 添加表字段 |
| alter table t1 drop age | 删除表字段 |
| alter table t1 change age newAge int | 修改表字段名和类型 |
| alter table t1 modify age int first/after xx | 修改表类型和位置 |

| | |
|---|---|
| drop table t1 | 删除表 |
| **数据相关：** | |
| insert into t1 values() | 增 |
| insert into t1’’ values() | |
| insert into t1 values(),() | |
| insert into t1() values(),() | |
| delete from t1 where id=10 | 删 |
| update stu set mid=x where age=y and name= ″″; | 改 |
| select * from t1 where id | 查 |
| select name from t1 | |
| select * from t1 | |
| **服务器相关：** | |
| Service mysqld start | 启动 MySQL 服务器 |
| Mysql -u ‘’/root -p | 连接服务器 |
| \h | 显示帮助内容 |
| \c | 清除命令行 |

## 二：数据库五大范式范式

1. 数据库范式

设计数据库时为了使数据库为合理的关系型数据库，需要遵从不同的规范要求，这些不同的规范要求被称为范式，满足规范祭祀说属于对应范式，范式越高数据库冗杂越小。

2. 第一范式（1NF）

原子型，每一列的字段不可分割。每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者重复的属性。消除非主键属性对键的部分函数依赖。

3. 第二范式（2NF）

主键，第二范式在满足第一范式的前提下建立起来的，即每个表中都要有一个主键，这个主键关键字段与其它非主键字段紧密相连，可以说依赖这个主键。非主属性非部分依赖于主关键字。消除非主属性对键的传递函数依赖。

4.第三范式（3NF）

外键，要求非主键列互不依赖，一个数据库表中不包含已在其他表中已包含的非主关键字信息，如果需要只需把另一张表的关键字作为外键保存在需要的表中。消除主属性对键的部分和传递函数依赖。

5.第四范式（BCNF）

禁止主键列和非主键列一对多关系不受约束。消除非平凡且非依赖的多值依赖。

6.第五范式（4NF）

将表分割成尽可能小的块，为了排除在表中所有的冗余。

## 三：JDBC 的执行步骤

1.JDBC 概念

JAVA 数据库连接，是 JAVA 语言中用来规范客户端程序如何来访问数据库的运用接口，提供诸如查询和更新数据库中数据的方法。

2.执行步骤

| | |
|---|---|
| **加载驱动** | Class.forName(“com.mysql.jdbc.Driver”)不会产生依赖 |
| | System.setProperty(“jdbc.drivers”,“driver1:driver2”);注册不太方便 |
| | DriverManager.registerDriver(com.mysql.jdbc.Driver);产生依赖 |
| **建立连接** | Connection conn =DriverManager.getConnection(url,user,password); |
| | URL:子协议:子名称//主机名:port/数据库名？属性名=属性值& |
| | User:登录数据库的 username;password:登录数据库的密码，为空就填“” |
| **创建运行对象** | Statement st = connection.createStatement(); |
| | 子类 PreparedStatement 能够对 SQL 语句进行预编译防止 SQL 注入 |
| | 孙子类 CallableStatement 主要用于运行 SQL 存储过程 |
| **运行 SQL 语句** | ResultSet rs =st.executeQuery(sql);用于运行实现查询功能的语句，返回结果集 |
| | int flag = st.executeUpdate(sql);用于运行实现增删改查功能的语句，返回整型 |
| **处理运行结果** | ResultSet 通过游标操作，游标是一个可控制的、能够指向随意一条记录的指针。 |
| **释放资源** | Close();数据库资源不关闭，其占用的内存不会被释放，徒耗资源，影响系统 |

## 四：什么是事务

1. 事务概念

事务是访问数据库的一个操作序列，数据库运用系统通过事务集来完成对数据库的存取，使得数据库从一种状态转换为另一种状态。

2. 事务特点

必须事务必须服从 ISO/IEC 所制定的 ACID 原则。即原子性、一致性、隔离性和持久性。

## 五：事务的四个特性

1. 原子性

不可分割性，事务要么全部被执行，要么就全部不被执行。如果事务的所有子事务全部提交成功，则所有的数据库操作被提交，数据库状态发生转换；如果有子事务失败，则其他子事务的数据库操作被回滚，即数据库回到事务执行前的状态，不会发生状态转换。

2. 一致性

事务的执行使得数据库从一种正确状态转换成另一种正确状态。

3. 隔离性

在事务正确提交之前，不允许把该事务对数据的任何改变提供给任何其他事务，即在事务正确提交之前，它可能的结果不应显示给任何其他事务。

4. 持久性

事务正确提交后，其结果将永久保存在数据库中，即使在事务提交后有了其他故障，事务的处理结果也会得到保存。

## 六：事务的隔离级别

1. 并发事务存在的问题

**脏读**：指事务 A 读到了事务 B 还没有提交的数据；

**幻读**：在一个事务操作里面发现了未被操作的数据；

**不可重复读**：一个事务里面读取两次某个数据，但结果数据不一样；

2. 隔离的基本概率

隔离已解决并发存在的问题，事务隔离级别越高，在并发下会产生的问题就越少。因为通过消耗性能解决问题，所以设立事务隔离级别，以便不同的项目根据各自情况选择合适的事务隔离级别。

3. 隔离等级

| | |
|---|---|
| DEFAULT | 默认隔离级别 |
| READ_UNCOMMITTED | 无法解决脏读、不可重复读、幻读中的任何一种 |
| READ_COMMITED | 防止脏读，但是无法限制不可重复读和幻读 |
| REPEATABLE_READ | 解决了脏读、不可重复读的问题，但是幻读的问题还是无法解决 |
| SERLALIZABLE | 解决了脏读、不可重复读和幻读问题 |

## 七：索引的类别和每种索引的创建方式

1. 索引类别

单列索引：主键索引、唯索引、普通索引；

组合索引：包含两个或两个以上的索引

2. 索引的创建方式

| | |
|---|---|
| 主键索引 | 不允许有空值 |
| 唯索引 | CREATE UNIQUE INDEX account_UNIQUE_Index ON `award`(`account`); |
| 普通索引 | CREATE INDEX account_Index ON `award`(`account`); |
| | ALTER TABLE award ADD INDEX account_Index(`account`) |
| 组合索引 | CREATE INDEX nickname_account_createdTime_Index ON `award`(`nickname`, `account`, `created_time`); |

## 八：聚簇索引和非聚簇索引的区别

1. 聚簇索引

表数据按照索引的顺序来存储的，索引项的顺序与表中记录的物理顺序一致。其叶子结点即存储了真实的数据行，不再有另外单独的数据页。 在一张表上最多只能创建一个聚集索引，因为真实数据的物理顺序只能有一种。

2. 非聚簇索引

表数据存储顺序与索引顺序无关。其叶结点包含索引字段值及指向数据页数据行的逻辑指针，其行数量与数据表行数据量一致。

3. 其它

聚簇索引为稀疏索引，数据页上一级的索引页存储的是页指针，而不是行指针；

非聚簇索引则是密集索引，在数据页的上一级索引页它为每一个数据行存储一条索引记录。

## 九： B 和 B+树的区别

1. B 树

每个节点最多包含 m 个孩子，m 称为 b 树的阶，m 的大小取决于磁盘页的大小。关键字集合分布子啊整棵树中；任何一个关键字出现且只出现在一个结点中；搜索有可能在非叶子结点结束；其搜索性能等价于在关键字全集内做一次二分查找；

2. B+树

有 n 棵子树的非叶子结点中含有 n 个关键字（b 树是 n-1 个），关键字不保存数据，只用来索引，所有数据都保存在叶子节点（b 树是每个关键字都保存数据）；

叶子结点中包含了全部关键字的信息，及指向含这些关键字记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接。所有的非叶子结点可以看成是索引部分，结点中仅含其子树中的最大（或最小）关键字；

通常在 b+树上有两个头指针，一个指向根结点，一个指向关键字最小的叶子结点；

同一个数字会在不同节点中重复出现，根节点的最大元素就是 b+树的最大元素；