数据库应用设计开发案例

—在线数据库实验平台 SQL-OJ

西北工业大学
计算机学院
李宁

编写日期：2022 年 9 月 1 日

目录

1. 选题背景

在数据库系统相关课程的教学中，让学生掌握并熟练运用 SQL 语句往往是该类课程的重要教学目标之一。在传统教学中，学生提交的 SQL 作业不仅增大了教师的批阅工作量，同时由于 SQL 语句写法灵活多样导致批阅难度较大，作业批阅结果可能也无法及时反馈给学生。因此，面向 SQL 语言开发一个高效的 OJ（Online Judge，简称 OJ）系统可以有效地提高数据库课程的教学效率和效果。

本章设计并实现一个在线数据库实验平台 MYSQL-OJ。教师可以在平台上方便地管理班级以及发布 SQL 练习题或者考试题。学生可以进行 SQL 练习或者考试，提交后平台自动批阅，学生能够及时得到结果反馈。此外，通过该平台的数据分析功能，教师还可以直观地掌握学生的答题情况，从而展开针对性的教学活动；学生也可以通过平台整理出自己的错题，便于强化练习弱点、总结复习。

2. 需求分析

2.1. 数据需求分析

数据需求指用户期望从数据库中获得信息的内容与性质。该应用系统中至少包括两类用户：教师和学生，其涉及的数据主要包括：

1. 用户信息：用户所属的学校、班级以及用户自身的姓名、学号等信息。

2. 题库与题目信息：系统内的 SQL 题目，每个题库可以由多个题目组成。

3. 试卷信息：教师抽取题目组成整套试卷，未发布之前学生不可见。

4. 练习或考试信息：发布试卷，包括试卷模式（练习或考试）、起止时间、发布对象班级等信息。

5. 答题信息：分为两种不同粒度的答题记录，包括每个学生的试卷（练习和考试）级别答题记录信息和每个学生每道题的答题历史记录。

根据以上数据分析，系统总体的数据流图如图 2-1，其中 P 是数据处理，D 是数据存储。

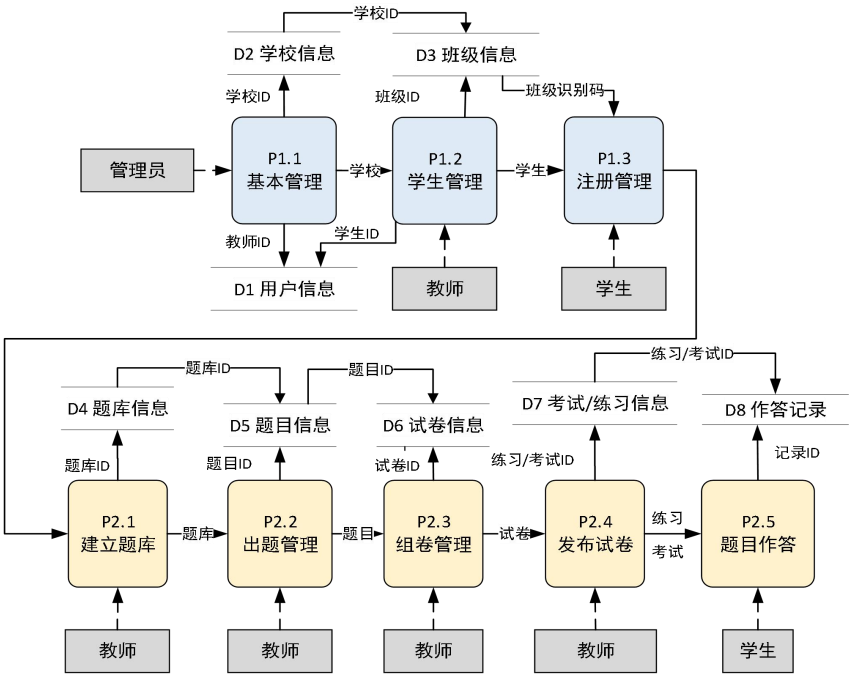

图 2-1 MYSQL-OJ 系统数据流图

下面列出了该数据流图中的部分主要数据字典（以服务 100 所学校，10 年为例）：

1. 主要数据处理

(1) P1.1：基本信息管理

处理过程名：基本管理

说　　明：管理员对学校信息、教师用户信息的基本增删改查处理。

输　　入：增删改查请求、学校基本信息、教师用户基本信息

输　　出：D1 用户信息中的教师信息和 D2 学校信息

(2) P1.2：学生信息管理

处理过程名：学生管理

说　　明：教师对班级、学生信息的基本增删改查处理，支持学生名单导入导出。

输　　入：增删改查请求、D2 学校信息

输　　出：D3 班级信息和 D1 用户信息中的学生信息

(3) P1.3：学生注册管理

处理过程名：注册管理

说　　明：学生使用教师告知的班级注册码进行注册，注册后会更新用户信息表。

输　　入：注册请求、D3 班级信息中的班级注册码和其他必要的注册基本信息

输　　出：D1 用户信息中学生的状态信息以及其他字段

(4) P2.1：题库信息管理

处理过程名：建立题库

说　　明：教师创建题库（库表结构以及初始数据准备）或者修改删除题库信息。

输　　入：题库的增删改查请求、D4 题库信息

输　　出：D4 题库信息

(5) P2.2：题目信息管理

处理过程名：出题管理

说　　明：教师对 SQL 题目的增删改查等操作，每个题目是基于某个题库创建。

输　　入：题目增删改查请求、D4 题库信息、以及 SQL 题目的题干、答案等

输　　出：D5 题目信息

2. 数据流（此处仅列出 2 个重点数据流作为示例，其余类似）

(1) 题目数据流

数据流名：题目

说　　明：教师通过选择题目完成组卷。

来源去向：P2.2 → P2.3

数据存储：D5 题目信息记录、D6 试卷信息记录

(2) 试卷数据流

数据流名：试卷

说　　明：教师将试卷发布为一次考试或者练习。

来源去向：P2.3 → P2.4

数据存储：D6 试卷信息记录、D7 考试信息或者练习信息

3. 主要数据存储

(1) D1：用户信息

数据存储名：用户信息

说　　明：记录每一个用户信息，码是用户 email。

数据描述：邮箱, 密码，角色，学校，姓名，学工号，学院，班级, 加入状态

数据　量：数据量约 500000 条（平均每校每年 500 人规模，100 校，10 年）

存取频度：平均每天 1000 次

(2) D2：学校信息

数据存储名：学校信息

说　　明：记录每一个学校信息，码是学校 ID。

数据描述：学校 ID，学校全称，校名英文，缩写，所在省市

数据　量：数据量约 100 条

存取频度：平均每天 50 次

(3) D3：班级信息

数据存储名：班级信息

说　　明：记录每一个班级信息，码是班级 ID。

数据描述：班级 ID，学校 ID，班级名称，负责教师，备注信息，学生名单

数据　量：数据量约 10000 条（假设每校每年 10 个班）

存取频度：平均每天 200 次

(4) D4：题库信息

数据存储名：题库信息

说　　明：记录每一个题库信息，码是题库 ID。

数据描述：题库 ID，题库名称，题库描述，创建 SQL 语句

数据　量：数据量约 100 条

存取频度：平均每天 50 次

(5) D5：题目信息

数据存储名：题目信息

说　　明：记录每一道题目信息，码是题目 ID。

数据描述：题目 ID，题目名，题库 ID，题目难度，题目描述，标准答案

数据　量：数据量约 10000 条（每个题库 100 道题）

存取频度：平均每天 1000 次

2.2. 功能需求分析

功能需求指用户期望利用该应用系统可以完成的操作，以下分为教师和学生角色分别说明该 MYSQL-OJ 在线实验平台的基本功能需求。

1．教师：

(1) 班级与班内成员管理：可对班级信息、班级成员信息进行增删改查等操作。可以按照指定格式导入学生成员名单，也可以导出系统中的学生名单。

(2) 题库管理：教师可以创建多个题库，题库可以私有或者公开。每个题库中可以添加若干道题目。

(3) 题目管理：教师可以自由增删改本人用户下的题目，可以查看其他教师公开的题目。每个题目需包括题干、答案等信息。

(4) 试卷管理：教师根据需要把题目自由组卷后以班级为单位给学生发布。发布试卷时，可以选择将其作为练习模式或者考试模式发布。

(5) 统计信息查看：教师可以查看一些重要统计数据，主要是基于学生的答题记录，如作答次数、正确性、分数等进行分析和计算得出相应的图表信息。

2．学生：

(1) 用户管理：学生可以进行注册，查看自己的信息，修改自己的密码等。本系统为学生用户设计了“加入状态”的信息。该信息表示此学生帐号注册选择的班级是否与此班级的内置学生名单相匹配，共分为“已加入”、“未加入”、“名单之外”3 种状态。该状态便于数据分析。

(2) 答题功能：学生可以进入教师发布的考试，逐题进行 SQL 答题。根据提交的 SQL

运行结果自动判题，给学生反馈。若系统设置答题时间已结束，则无法作答。

(3) 仪表盘：学生可以查看自己的答题统计数据，并且可以查阅自己的答题记录，便于错题整理和强化练习。

3．管理员：

(1) 学校班级管理：可对学校、班级等信息进行增删改查等管理。

(2) 教师用户管理：可对教师用户信息进行增删改查等管理。

2.3. 非功能需求分析

本系统的非功能需求包括：

1．安全性需求：严格按照系统设计的权限，不同用户角色仅能进行权限内操作。

2．性能需求：本系统至少支持 1000 人同时进行考试操作并在 3 秒内提交答案到系统。练习模式下，每道题提交后需在 5 秒内给出批改反馈。

3. 数据库设计

3.1. 概念结构设计

基于前面的数据需求和功能需求分析，本小节展示该 MYSQL-OJ 系统的概念设计 E-R 图。

1．实体与联系的语义描述

用户相关实体：学校、班级、用户（管理员、教师、学生）

考试相关实体：题库、题目、试卷、考试活动

根据现实世界与应用需求分析，实体之间语义描述如下：

(1) 一个学校拥有多个班级，一个班级只能属于某个学校；一个班级有多名学生，一个学生只能属于一个班级。

(2) 系统可以有多个管理员，任意一个管理员可以增删改所有学校、教师的基本信息；每位教师管理所负责班级的学生信息。

(3) 一位教师可以创建多个题库（题库指一个事先准备好的数据库，可包含多张表以及部分数据），每个题库只能被一位老师创建。基于一个题库可以出多道题目，每个题目只能属于某一个题库。每位教师可以出若干道题目。一套试卷可以由任意多道题目组成，相同题目可出现在多套试卷中，且相同题目在不同试卷中分值可以不同。每位教师可给所负责的一个或多个班级，将任意一套可访问的试卷发布为一次考练活动。学生可以作答自己可见的未过期考练活动。

2．实体属性图

为了使系统 E-R 图的描述更加简洁清晰，此处先将 SQL-OJ 系统中所有实体的属性用图 3- 1 所示的实体属性图描述，这样后续图 3- 2、图 3- 3 和图 3- 4 的 E-R 图中仅重点描述实体之间的联系以及联系的属性。

图 3- 1 SQL-OJ 系统的实体属性图

3．分 E-R 图

(1) 用户管理子 E-R 图

图 3- 2 为用户管理的子 E-R 图。经过分析可知，教师与学生的“管理 3”可通过教师与班级之间的“负责”和班级与学生之间的“包含”推理得到，故为冗余联系，可将其删除。

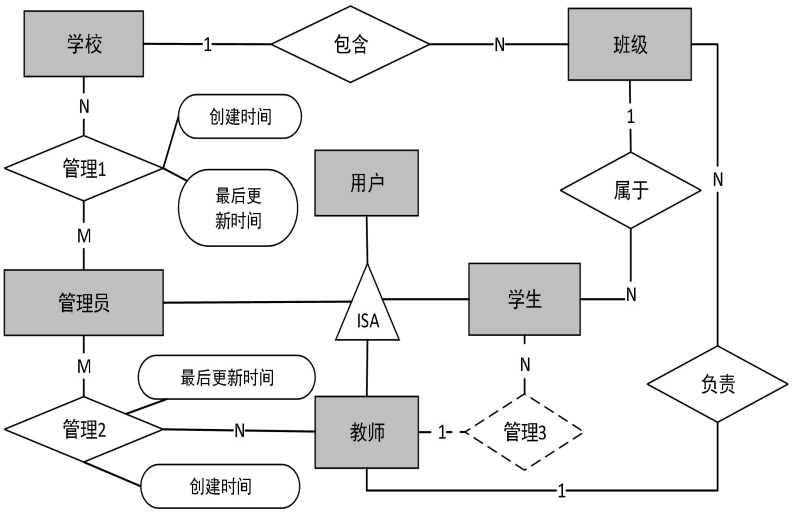

图 3- 2 SQL-OJ 系统中用户管理的子 E-R 图

(2) 考试管理子 E-R 图

图 3- 3 为考试管理的子 E-R 图，班级、教师、学生之间的完整联系可参考图 3- 2。

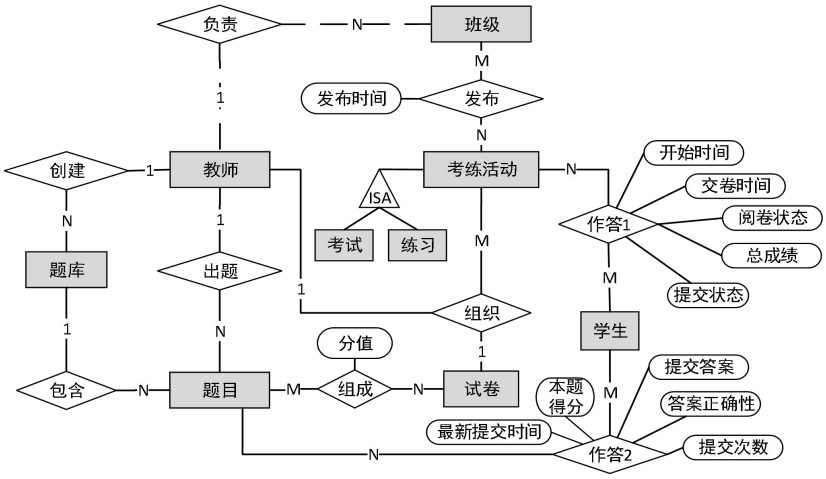

图 3- 3 SQL-OJ 系统中考试管理的分 E-R 图

4．总 E-R 图

根据前述分析，合并以上的两个分 E-R 图得到图 3- 4 所示的 SQL-OJ 系统整体 E-R 图。

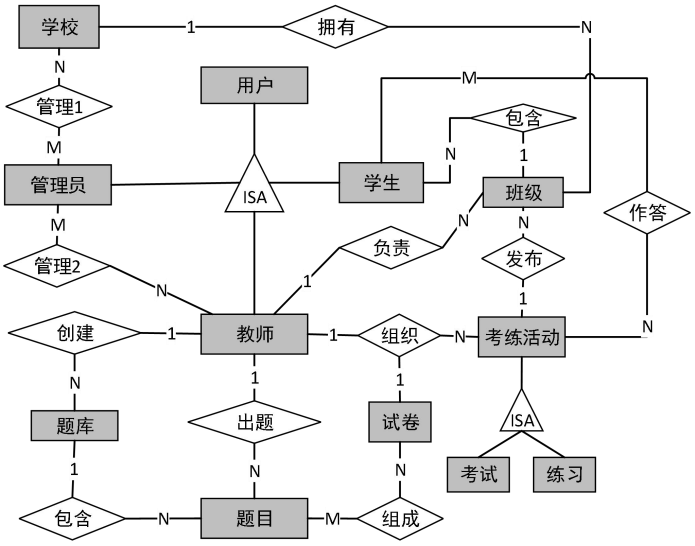

图 3- 4 SQL-OJ 系统总 E-R 图

3.2. 逻辑结构设计

为了方便描述，在本节后续的数据模型中采用表 3-1 符号表示特定含义。

表 3-1 数据模型描述符号

编号	符号	含义
1	FK	外键 (Foreign Key)
3	PRI	主键约束，也可以用下划线表示
4	UNI	唯一约束

1．E-R 图转换成逻辑模型

基于分析得到的 E-R 图，将其逐一转换成关系逻辑模型。

(1) 实体转换：表 3-2 展示了如何将每个实体单独转换成一个独立的关系。

表 3-2 实体转换表

编号	实体	实体转换说明与转换形成的关系
1	学校	user_school (学校 ID, 校名全称, 校名英文全称, 校名英文简称)
2	班级	user_classroom (班级 ID, 班级名, 班级描述, 班级识别码, 开放开始年度，开放结束年度，是否活跃状态)
3	用户	用户实体是一个父类，其子类包括管理员、教师、学生。可对父类创建一张表 User。用户中的权限等级用于区分三个子类。 user_user(用户 ID, 电子邮件, 用户名, 密码, 权限等级, 真实姓名, 学工号, 学院名称, 注册时间, 加入状态, 最后登录时间)。
4	管理员	子类管理员用户并无特有属性，该步骤可不创建单独关系。
5	教师	子类教师用户并无特有属性，该步骤可不创建单独关系。
6	学生	子类学生用户并无特有属性，该步骤可不创建单独关系。
7	题库	coding_questionset(题库 ID, 题库名, 题库描述, 数据库名, 创建 SQL)
8	题目	coding_question（题目 ID, 题目名, 难度, 题干描述, 参考答案）
9	试卷	coding_paper（试卷 ID, 试卷名, 试卷描述）
10	考练活动	该实体是一个父类，其子类包括考试和练习活动，此处期望考试和练习分别记录信息，故不创建父类关系，仅创建子类关系。
11	考试活动	coding_exam（考试 ID, 考试名称, 考试描述, 开始时间, 结束时间, 活跃状态, 是否在解析中公布答案）
12	练习活动	coding_exercise（练习 ID, 练习名称, 练习描述, 开始时间, 结束时间, 活跃状态）

(2) 联系转换：表 3-3 展示了如何按照联系类型分别转换各个联系。有的联系需要创建新关系，有的则需要在已有关系中增加属性（灰色背景属性表示在已有关系中的新增部分）。

表 3-3 联系转换表

编号	联系	联系转换说明与转换形成的关系
1	学校与班级： 包含 (1:N)	将两个 1:N 联系的 1 端，分别合并到 N 端的班级关系。 user_classroom：班级 ID, 班级名, 班级描述，班级识别码, 是否活跃状态, 所属学校 ID(FK), 负责教师 ID(FK)
2	教师与班级： 负责(1:N)	
3	班级与学生：包含 (1:N)	学生实体尚没有独立关系，但将该联系加入其父类用户表会导致非学生用户产生冗余信息，此处新建学生关系。 user_student：学生的用户 ID, 班级 ID(FK)
4	学校与管理员： 管理 1(M:N)	管理员可管理学校的基础信息，M:N 联系新建关系：

		user_admin_school：管理员 ID(FK), 学校 ID(FK), 创建时间, 最后更新时间
5	教师与管理员： 管理 2(M:N)	管理员管理教师的基础信息，M:N 联系新建关系： user_admin_teacher：管理员 ID(FK), 教师 ID(FK), 创建时间, 最后更新时间
6	教师与题库： 创建(1:N)	将 1:N 联系中 1 端合并到 N 端的题库关系 coding_questionset：题库 ID, 题库名称, 题库描述, 数据库名, 创建 SQL, 创建教师 ID(FK)
7	教师与题目： 出题 (1:N)	将两个 1:N 联系的 1 端教师 ID 和题库 ID，分别合并到 N 端的题目关系： coding_question：题目 ID, 题目名, 难度, 题干描述, 参考答案, 出题教师 ID(FK), 题库 ID(FK)
8	题库与题目： 包含 (1:N)	
9	试卷与题目： 组成(M:N)	M:N 联系新建关系 coding_paper_question: 试卷 ID(FK)， 题目 ID(FK)，分数
10	考试活动与教师、试卷： 组织考试 (N:1:1)	将三个实体之间多对 1 中的 1 端合并到 N 端的考试 coding_exam：考试 ID, 考试名, 考试描述, 开始时间, 结束时间, 活跃状态, 是否公布答案, 教师 ID(FK), 试卷 ID(FK)
11	练习活动与教师、试卷： 组织练习 (N:1:1)	将三个实体之间的多对 1 中的 1 端合并到 N 端的练习 coding_exercise：练习 ID, 练习名, 练习描述, 开始时间, 结束时间, 活跃状态, 教师 ID(FK), 试卷 ID(FK)
12	班级与考试活动: 发布考试(M:N)	M:N 联系新建关系 coding_exam_class：考试 ID(FK), 班级 ID(FK), 发布时间
13	班级与练习活动: 发布练习(M:N)	M:N 联系新建关系 coding_exercise_class：练习 ID(FK), 班级 ID(FK), 发布时间
14	学生与考试作答记录：作答 1(M:N)	M:N 联系新建关系 coding_exam_answer_rec：考试记录 ID,学生 ID(FK), 考试 ID(FK), 开始时间, 交卷时间, 提交状态, 阅卷状态, 总成绩
15	学生与练习作答记录：作答 1(M:N)	M:N 联系新建关系 coding_exer_answer_rec：练习记录 ID, 学生 ID(FK), 练习 ID(FK), 开始时间, 交卷时间, 提交状态, 阅卷状态, 总成绩
16	学生与考试题目作答记录： 作答 2(M:N)	M:N 联系新建关系（记录考试中每道题的答题情况） coding_exam_ques_answer_rec：考试题目记录 ID, 学生 ID(FK), 题目 ID(FK), 考试 ID(FK), 提交答案, 答案正确性, 最新提交时间, 提交次数, 本题得分
17	学生与练习题目作答记录： 作答 2(M:N)	M:N 联系新建关系（记录练习中每道题的答题情况） coding_exer_ques_answer_rec：练习题目记录 ID, 学生 ID(FK), 题目 ID(FK), 练习 ID(FK), 提交答案, 答案正确性, 最新提交时间, 提交次数, 本题得分

2．关系模式的优化

如前所示 SQL-OJ 系统中所有的实体与联系经过初步转换和合并之后，共计 20 个关系。针对这些关系，需进行规范化理论分析与优化。本处选择其中较为复杂的关系如 Exam、Exam_class 以及 PaperQuestion 为例分析详细分析。其他关系均满足 BCNF，读者可自行分析。

① Exam (考试 ID, 考试名, 考试描述, 开始时间, 结束时间, 活跃状态, 是否公布答案, 教师 ID, 试卷 ID)：该关系的码是考试 ID，不存在非主属性和主属性（考试 ID）对码的部分函数依赖和传递函数依赖，因此满足 BCNF。

② Exam_class (考试 ID(FK), 班级 ID(FK), 发布时间)：假设允许把某个考试给某个班级发布多次，则码中需要加入发布时间。由此该关系为全码，无非主属性，满足 BCNF。

关于应用程序特定需求下的反范式化设计、分解或合并等也需仔细分析。本例中的示例分析如下：该系统中教师身份登录时，以班级为单位进行作答统计信息，在访问题目的作答记录表时，会频繁使用题目名称，因此在该表中设计如下的冗余题目名。

coding_exam_ques_answer_rec:考试题目记录 ID, 学生 ID, 题目 ID, 考试 ID, 提交答案, 答案正确性, 最新提交时间, 提交次数, 本题得分, 题目名。

3．外模式设计

本例的设计中，用户中的教师信息并没有使用独立关系存储，但实际中经常会用到所有教师的列表信息。根据该需求，创建一个教师信息视图：Teacher，仅包含教师 ID 和教师姓名即可。后续根据业务情况，可以根据需要增加视图。

3.3. 安全性与完整性设计

本系统中的安全性设计主要涉及权限管理、数据加密存储、SQL 注入防止等方面。关于权限管理，该系统中主要涉及了管理员、教师与学生三种角色，各种角色在系统中的权限有所不同，具体角色可以访问的不同功能参见表 5-1。数据加密主要考虑用户的密码信息使用 MD5 加密后存储。由于 Django 框架本身具有 SQL 防注入的功能，开发中不用特别考虑。

数据完整性设计根据系统的实际需要考虑实完整性约束和触发器。下面举例说明本系统中的完整性约束和触发器的设计。

1. 完整性约束设计示例：

(1) 实体完整性：User_Student 表的 student_id 为主键。

(2) 参照完整性:User_Student 表的 class_id 为外键（User_classroom 表的 class_id）。

(3) 用户自定义完整性：User_User 表的 priority 取值：0:学生, 1:教师, 2:管理员。

2. 触发器设计示例：本例中，为了避免统计功能中的实时统计给系统带来过大压力，设计了触发器 T1。该触发器在完成某个学生的判卷后更新 Coding_ExamAnswerRec 表的成绩字段时，会自动更新单独设计的统计信息表 Coding_Stat 中相关信息。

3.4. 物理结构设计

物理结构设计主要考虑数据存储与数据存取。本例中，在数据存储方面，为了提高系统的高可用性，在云数据库实例创建时，选择了主备架构，该架构可以定期备份主库的数据，防止数据丢失。在数据存取方面，重点考虑索引设计。

在数据存取方面，首先按照系统功能需求，列出系统中可能相对频繁执行的查询需求，然后逐一考虑索引设计。注意该步骤必须结合应用系统的功能设计深入思考，才能得到较好的设计。因此，推荐该步的索引设计在应用系统功能设计完成之后进行。示例设计如下：

1. 系统中每个人仅能看到自己或者自己关联班级的相关信息，学生 ID、教师 ID、班级 ID 经常出现在连接条件中。答题功能中，频繁使用考试 ID、试卷 ID、题目 ID 等属性，但因为这些字段都有主键索引，所以无需额外创建索引。

2. 学生仪表盘的统计功能中，需统计该学生所有练习中的总提交次数。该功能将对 coding_exer_ques_answer_rec 表的提交次数做求和计算，该表数据量较大，因此按照索引设计的启发式规则，对频繁使用聚集函数的字段设计复合索引（学生 ID，提交次数）。

3. 教师身份的统计功能中，需统计每次考试中每道题目的作答详细信息，该功能将对 coding_exam_ques_answer_rec 表进行题目 ID 相关的聚集查询，该表数据量较大，因此设计索引（题目 ID，学生 ID）。

其他更多关于索引的设计，请读者结合系统需求自行思考。

最终形成如下的关系设计如下。

1. User_School （学校表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
school_id	int	-	N	PK		学校 ID，自增
school_name	varchar	150	N	UNI		校名全称
school_name_en	varchar	150		UNI		校名英文全称
school_abbr	varchar	30		UNI		校名英文简称

2. User_Class （班级表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
class_id	int	-	N	PK		班级 ID，自增
class_name	varchar	150	N			班级名
class_desc	varchar	300				班级描述
start_year	datetime	-	N			开放开始年度
end_year	datetime	-	N			开放结束年度

join_code	varchar	150	N			班级识别码
active	varchar	30	N		是/否	是否活跃状态
school_id	int	-	N	FK		所属学校 ID
teacher_id	int	-		FK		负责教师 ID

3. User_User （用户表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
user_id	int	-	N	PK		用户 ID，自增
email	varchar	100	N	UNI		电子邮件
username	varchar	100	N			用户名
password	varchar	100	N		0	密码
priority	int	-	N			权限等级
full_name	varchar	150	N			真实姓名
internal_id	varchar	30		UNI		学工号
college_name	varchar	150			计算机学院	学院名称
register_time	datetime	-				注册时间
join_status	varchar	30			0	加入状态
login_time	datetime	-				最后登录时间
school_id	int	-		FK		所属学校 ID

注：priority 为枚举类型：0:学生, 1:教师, 2:管理员。

join_status 为枚举类型：0:名单外, 1:未加入, 2:已加入, 3:管理员或教师。

4. User_Student （学生表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
student_id	int	-	N	FK/UNI		学生的用户 ID
class_id	int	-	N	FK		班级 ID

5. User_Teacher （教师视图）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
teacher_id	int	-	N	用户表 user_id		教师 ID
username	varchar	100		用户表 username		教师名

6. Coding_Questionset （题库表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
ques_set_id	int	-	N	PK		题库 ID，自增
ques_set_name	varchar	150	N			题库名称
ques_set_desc	varchar	300				题库描述
db_name	varchar	100	N			数据库名
create_sql	varchar	1024	N			创建 SQL
initiator	int	-		FK		创建者 ID
share	bool	-	N		是	是否共享

7. Coding_Question （题目表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
ques_id	int	-	N	PK		题目 ID
ques_name	varchar	150	N			题目名称
ques_difficulty	varchar	20	N		0	难度
ques_desc	varchar	500	N			题干描述
ques_ans	varchar	1024	N			参考答案
initiator	int	-	N	FK		出题者用户 ID
ques_set_id	int	-	N	FK		所属题库 ID

注：ques_difficulty 为枚举类型：-1:未知, 0:简单, 1:中等, 2:困难。

8. Coding_Paper （试卷表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
paper_id	int	-	N	PK		试卷 ID

paper_name	varchar	150	N			试卷名
paper_desc	varchar	300				试卷描述
initiator	int	-	N	FK		创建者用户 ID
share	bool	-	N		是	是否共享

9. Coding_Paper_Question （试卷与题目表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
paper_id	int	-	N	PK		试卷 ID
question_id	int	-	N	PK		题目 ID
score	int	-				分数

10. Coding_Exam （考试活动表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
exam_id	int	-	N	PK		考试 ID
exam_name	varchar	100				名称
exam_desc	varchar	300				描述
start_time	datetime	-	N			开始时间
end_time	datetime	-	N			结束时间
active	bool	-	N		否	发布状态
show_answer	bool	-	N		否	是否公布答案
teacher_id	int	-	N	FK		教师 ID
paper_id	int	-	N	FK		试卷 ID

11. Coding_Exam_Class （班级与考试表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
exam_id	int	-	N	PK		考试 ID
class_id	int	-	N	PK		班级 ID
publish_time	datetime	-	N	PK		发布时间

12. Coding_ExamAnswerRec （考试作答记录表）

字段名	数据类型	长度	允许空	Key/索引	默认值	含义
id	int		N	PK		记录 ID，自增
student_id	int	-	N	FK		学生用户 ID
exam_id	int	-	N	FK		考试 ID
start_time	datetime	-				开始时间
end_time	datetime	-				交卷时间
status	bool		N		False	提交状态
mark_status	bool		N		False	阅卷状态
score	int	-				总成绩

4. 数据库实施

4.1. 创建数据库

根据最终完成的关系模式设计，将每个关系写成建表语句，执行建库 oj 和建表操作。注意在本实例中，实际是通过在 Django 框架中对每张表构建 Model 时自动创建的表，最终创建完成的表如下（表名以 auth_和 django_开头的部分是 Django 框架创建）。

```
> MariaDB [oj]> show tables;
+----------------------------+
| Tables_in_oj               |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| coding_exam                |
| coding_exam_answer_rec     |
| coding_exam_class          |
| coding_exam_ques_answer_rec |
| coding_exer_answer_rec     |
| coding_exer_ques_answer_rec |
| coding_exercise            |
| coding_exercise_class      |
| coding_paper               |
| coding_paper_question      |
| coding_question            |
| coding_questionset         |
| django_admin_log           |
| django_content_type        |
```

```
| django_migrations          |
| django_session             |
| user_admin_school          |
| user_admin_teacher         |
| user_classroom             |
| user_school                |
| user_student               |
| user_user                  |
+----------------------------+
```

4.2. 加载数据

待数据库表创建完成之后，首先手动生成少量测试数据，包括用户管理部分和考试管理各个表中的必要数据，用于测试系统的基础功能。

其次，借助自编脚本程序等生成大量数据，特别是题库、题目、学生信息，数据至少支撑模拟随机 1000 名学生并发答题的场景，重点确认系统的稳定性、性能等。模拟数据量规划参考：100 所学校，每所学校 5 个班级，每个班 100 名学生，共计学生 50000 名，题库 50 个，其中使用最频繁的 5 个题库共计包含题目 1000 道（平均每个题库 200 道），10 名教师发布至少 10 套考试和 10 套练习，1000 名学生同时随机答题并提交试卷。

5. 数据库应用程序设计

5.1. 系统功能模块图

本系统采用 Python 语言的 Django 框架实现。遵循软件设计中“高内聚，低耦合”的设计思想，本 MYSQL-OJ 在线实验平台分为 2 个子系统（即 2 个 Django App）：用户管理和考试管理。整体的系统模块图如图 5-1 所示，各个模块的详细功能说明如表 5-1 所示。

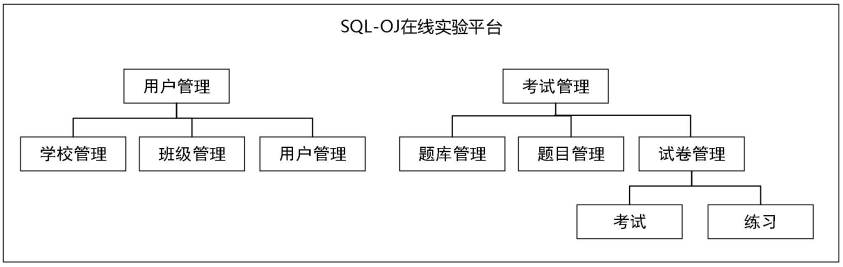

图 5-1 MYSQL-OJ 系统模块图

1. 用户基本管理模块（User App）：用于学校信息、班级信息的管理，以及学生用户、教师用户、管理员用户的注册、登录等功能。

2. 考试管理模块（Coding App）：该模块包括本系统的核心功能，如题库、题目、试卷、考试以及答题记录等信息进行管理和分析。

表 5-1 详细功能模块列表

编号	应用	子模块	功能简介	对应页面	权限
1	用户管理	注册	用户注册	auth-register.html	所有人
2		登录	用户登录	auth-login.html	所有人
3		重置密码	忘记密码时重置	auth-resetpw.html	所有人
4		用户资料	详细个人资料	user-info.html	所有人
5		班级管理	创建或查看班级	class-manage.html	教师
6		班级详情	修改班级详细信息	class-details.html	教师
7	考试管理	仪表盘	重要统计数据	index.html	所有人
8		考试管理	发起考试或练习	exams-manage.html	教师
9		题目管理	添加题目	questions-manage.html	教师
10		答题列表	查看考试	exams-manage.html	所有人
11		答题详情	学生答题页面	coding-editor.html	所有人
12		统计信息	展示全部统计数据	teacher-statistics.html	教师

由于篇幅有限，本章将选择部分核心功能模块（考练作答、统计信息）详细展开介绍。其他模块的分析、设计与实现与之类似，读者可自行练习。

5.2. 考练列表

当以学生身份登录时，图 5- 2 的答题列表页面显示当前已发布并分配给该学生所在班级的考试或练习。考试或者练习的状态存在三种情况：进行中（蓝紫色）、已完成（绿色）、已结束（红色）共 3 种状态，其中显示为“已结束”的考试或练习将无法进入作答页面。

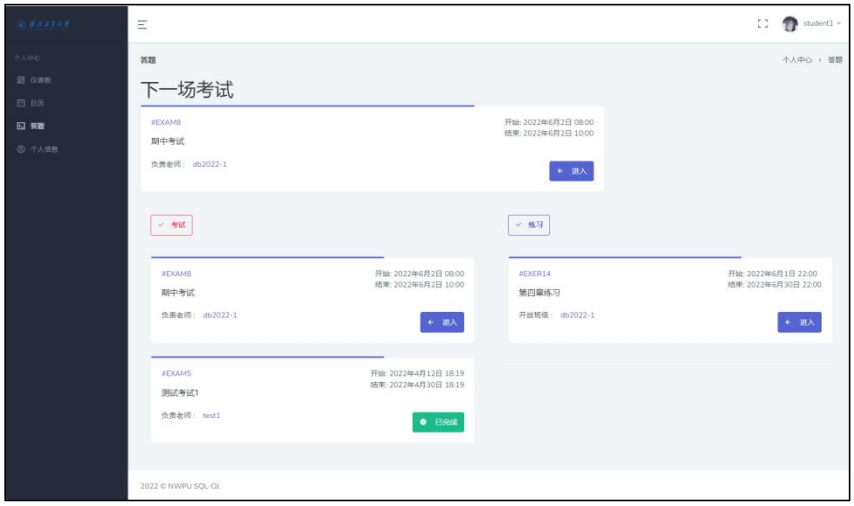

图 5- 2 SQL-OJ 学生答题列表

5.3. 考练作答

学生选择某考试或者练习进入后，逐个题目进行作答。每道题完成后，点击【提交运行】，则进入下一道题。作答时，学生也可以直接跳过某道题，直接点击【下一题】或【上一题】，最后一道题完成后，在该画面点击【交卷】，不论是考试还是练习交卷后都不能继续作答。当多人同时答题的情况下，系统会使用任务队列进行判卷处理。

图 5- 3 SQL-OJ 学生考练作答

5.4. 统计信息

学生身份登录时，在首页如图 5- 4 所示的“仪表盘”页面，列出了该学生的考试练习汇总信息以及作答历史数据，如提交次数、作答题目难度分布、考试成绩分析等。以教师身份登录时，图 5- 5 的班级数据中展示了该教师所负责班级的各种统计数据，具体包括题目数量、提交次数、平均完成率、平均正确率等。点击每条考试或者练习记录的详情，显示如图 5- 6 的每道题目的答题状况，便于教师在教学中挑选重点题目进行讲练。

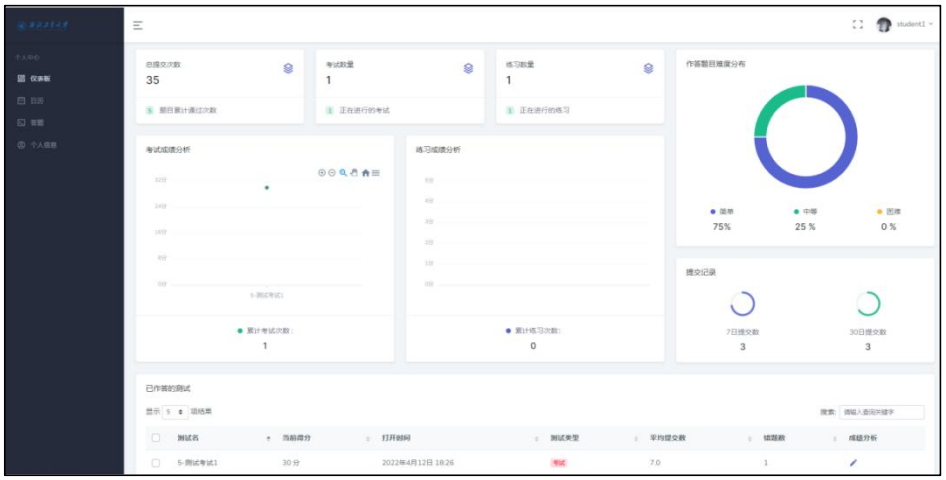

图 5- 4 SQL-OJ 统计信息——仪表盘（学生）

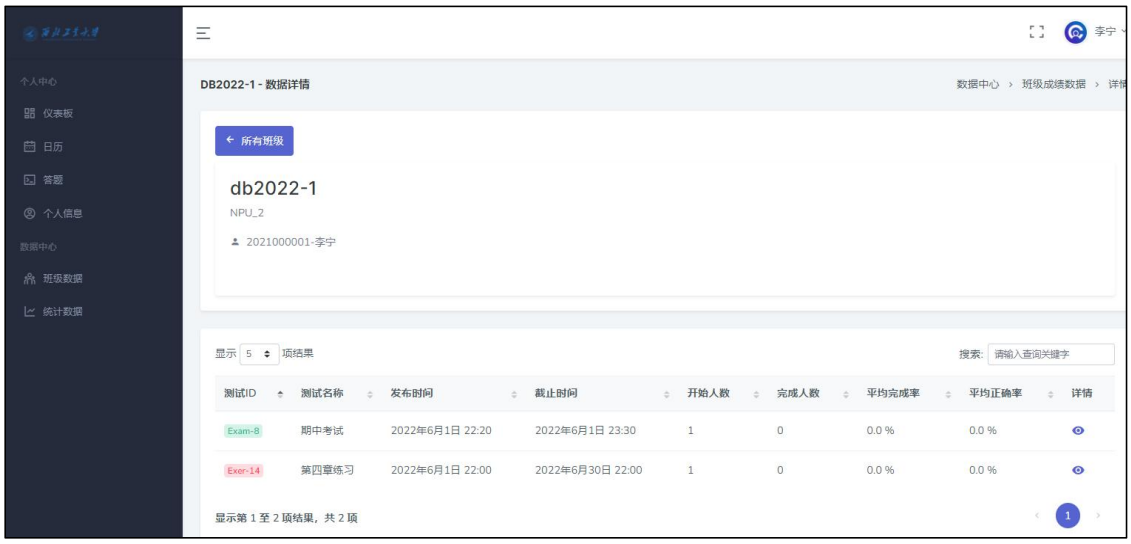

图 5- 5 SQL-OJ 统计班级数据——考练详情（教师）

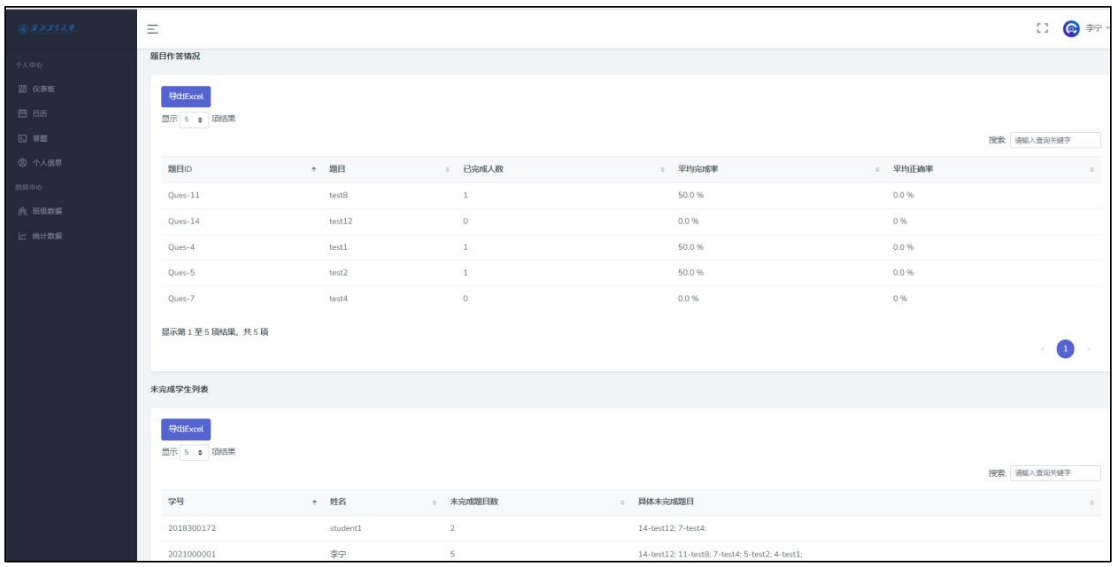

图 5- 6 SQL-OJ 统计班级数据——题目详情（教师）

6. 数据库应用程序开发

6.1. 数据库连接

在应用程序中需要采用合适的方式进行数据库连接，连接技术的选择可能会影响应用程序性能等。选择时通常需要考虑数据库连接技术（JDBC, ODBC, pymysql 等），以及连接池技术（如 Java 的 C3P0, HikariCP, Druid 等，Python 的 django_mysqlpool.backends.mysqlpool）等方面。连接池技术可以有效节约系统资源，提升系统性能。本例中若使用 Django 的 mysqlpool 时，配置如下。

```
DATABASES = {
    'default': {
```

```
        'ENGINE': 'django_mysqlpool.backends.mysqlpool',
        'NAME': 'SQLOJ',
        'HOST': '127.0.0.1',
        'PORT': 3306,
        'USER': 'user1',
        'PASSWORD': '123456'
    }
}
```

若使用普通的非连接池，则把上述'ENGINE'修改成：django.db.backends.mysql。这种配置下，需要在 project 同名目录下面的__init__.py 文件中，添加如下代码，Django 后台使用的 pymysql 进行数据库连接。

```
import pymysql

pymysql.install_as_MySQLdb()
```

6.2. 考练列表管理

Django 中的各个功能都包括 model, view, template 三个主要部分，template 层负责 web 页面显示，model 层定义了相关的数据模型（与数据库中的表对应），view 层进行业务处理。考练列表管理的 template 层页面设计参考图 5- 2，model 层的考试表主要实现如下。

```
class Exam(models.Model):
    exam_id = models.AutoField(verbose_name=_('考试ID'), primary_key=True)
    exam_name = models.CharField(verbose_name=_('考试名称'), max_length=100, default=_('未命名'))
    desc = models.TextField(verbose_name=_('考试描述'), null=True, blank=True)
    start_time = models.DateTimeField(verbose_name=_('开始时间'), default=None)
    end_time = models.DateTimeField(verbose_name=_('结束时间'), default=None)
    active = models.BooleanField(verbose_name=_('发布状态'), default=False)
    show_answer = models.BooleanField(verbose_name=_('在解析中公布答案'),default=False)
    paper = models.ForeignKey(verbose_name=_('试卷'), to=Paper, on_delete=models.CASCADE)
    teacher = models.ForeignKey(verbose_name=_('组织者'), to='user.Teacher', on_delete=models.SET_NULL, null=True)

    def __str__(self):
        return str(self.exam_id) + str('-') + str(self.exam_name)

    @property
    def is_over(self):
        return timezone.now() > self.end_time

    @property
    def finish_info(self):
        query_result = ExamAnswerRec.objects.filter(exam=self, status=True)
        have_finished = query_result.count()
        all_students = Student.objects.filter(classroom__in=self.classroom.all()).count()
        unfinished = all_students - have_finished
        total_score = self.paper.total_score()
        excellent = query_result.filter(score__gte=total_score * 0.85).count()
        good = query_result.filter(score__gte=total_score * 0.70).count() - excellent
        fair = query_result.filter(score__gte=total_score * 0.6).count() - excellent - good
        fail = query_result.filter(score__lt=total_score * 0.6).count()
        average_score = query_result.aggregate(average_score=Avg('score'))
        return have_finished,unfinished,excellent,good,fair,fail,all_students,average_score['average_score']
```

图 6- 1 考练信息列表的 model 层考试表的实现

Django 的各种操作都是基于 ORM 对象实现，以下是显示考练列表信息的主要处理逻辑，

在 view 层实现（图 6- 2）。此外，template 层的部分实现示例如图 6- 3。

1. 获得所有符合查询要求（登录学生所在班级的考练 ID，且状态为进行中的考试或练习）的考练 ID 列表 1，并按照发布时间降序排列。
2. 查询已完成（登录学生所在班级，且提交状态为 TRUE 的考试）的考练 ID 列表 2。
3. 获得进行中的考练 ID 列表 =（考练 ID 列表 1）EXCEPT（已完成考练 ID 列表 2）。
4. 根据考练 ID 列表 2 的获得已完成的考练详细信息列表，并按发布时间降序排列。
5. 获得下一场考试详细信息 = 进行中考练详细信息列表中的第一条记录。
6. 给 template 页面返回需要的 content 信息。

```python
def coding(request):
    '''Render coding template'''
    conditions = {
        'classroom' : request.user.student.classroom,
        'active' : True
    }
    exams_list = models.Exam.objects.order_by('publish_time').filter(**conditions)
    exer_list = models.Exercise.objects.order_by('publish_time').filter(**conditions)
    have_finished = models.ExamAnswerRec.objects.filter(student=request.user.student,status = True)
    have_finished_exam_id = []
    for element in have_finished:
        have_finished_exam_id.append(element.exam.exam_id)
    unfinished = exams_list.exclude(exam_id__in=have_finished_exam_id)
    have_finished = models.Exam.objects.order_by('publish_time').filter(exam_id__in=have_finished_exam_id)
    next_exam = unfinished.first()
    content = {
        'exams_list': unfinished,
        'finished' : have_finished,
        'exer_list': exer_list,
        'next_exam': next_exam,
    }
    return render(request, 'coding/coding.html', context=content)
```

图 6- 2 考练信息列表 view 层主要处理

```
<div id="todo-task" class="task-list">
    {% for exam in exams_list %}
        <div class="card task-box">
            <div class="progress progress-sm animated-progess" style="height: 3px;">
                <div class="progress-bar" role="progressbar" style="width: 72%" aria-valuenow="72" aria-valuemin="0" aria-valuemax=
                "100"></div>
            </div>
            <div class="card-body">
                <div class="float-right ml-2"> <div> 开始: {{ exam.start_time }} <br> 结束: {{ exam.end_time }} </div> </div>
                <div class="mb-3"> <a href="javascript: void(0);"class="">#EXAM{{ exam.exam_id }}</a> </div>
                <div>
                    <h5 class="font-size-16"><a href="javascript: void(0);" class="text-dark">{{ exam.exam_name }}</a></h5>
                    <p class="mb-4">{{ exam.desc }}</p>
                </div>
                <div class="d-inline-flex team mb-0">
                    <div class="mr-3 align-self-center"> 负责老师 : </div>
                    <div class="team-member">
                        {% for class in exam.classroom.all %}
                            <a href="javascript: void(0);" class="team-member d-inline-block" data-toggle="tooltip" data-placement="top"
                                title="{{class.teacher.user.full_name}}"> {{class}} </a>
                        {% endfor %}
                    </div>
                </div>
            {% if exam.is_over %}
                <div class="float-right">
                    <a href="{% url 'coding:coding' %}" class="btn btn-danger mt-1 waves-effect waves-light">
                        <i class="mdi mdi-alert-octagon mr-3"></i>已结束 </a> </div>
            {% else %}
                <div class="float-right">
                    <a href="{% url 'coding:coding-editor' 'exam' exam.exam_id exam.first_ques %}" class="btn btn-primary mt-1
                    waves-effect waves-light"> <i class="mdi mdi-arrow-left mr-3"></i>进入 </a> </div>
            {% endif %}
            </div>
        </div>
        <!-- end task card -->
    {% endfor %}
```

图 6- 3 考练信息列表 template 层部分处理

6.3. 考练作答详情

考试或者练习的答题功能同样需要 template, view, model 共同协作完成。该功能的 template（coding-editor.html）设计界面参考 5.3 小节，model 类同图 6- 2。

在考练作答处理的 view 层，除了常规基于 ORM 对象的数据创建、更新、删除等，如何实现 SQL 题目正确性的判定是该阶段最重要的问题。本例中使用数据库表复制验证的方式实现（sql_check.py）。整个过程如图 6- 4 所示：首先，创建临时数据库。其次，在临时数据库中复制与已有表结构相同的表。最后，在临时数据库中分别执行标准答案 SQL 和学生提交 SQL 语句，并对两种执行结果进行比对，完成后将临时数据库删除。

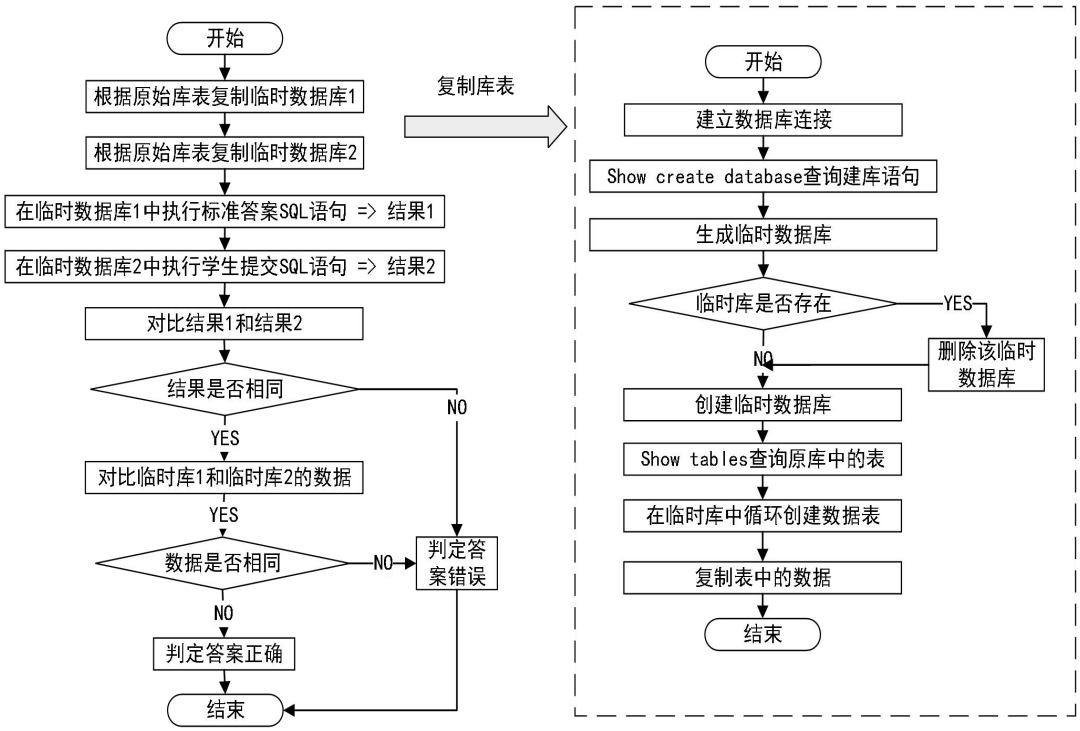

图 6- 4　SQL 正确性判断流程图

6.4. 统计信息

统计信息可向教师或者学生直观地展示一些重要的统计数据，该部分主要是利用了学生考试练习和具体每道题目的作答记录，如作答次数、正确性、分数等数据，在此之上加以分析和运算的结果绘制的图表，包括未完成学生列表、有错题学生列表、考试或者练习完成率等。各种统计数据计算都在 coding\view.py 或者 model.py 中实现，显示页面的各种图表都在 templates\coding\ teacher-analysis.html 或者 analysis.html 中实现（如图 6- 5 所示）。

```
{% for exam in exam_objects %}
<div class="col-lg-6">
    <div class="card">
        <div class="card-body">
            <h4 class="card-title mb-4">考试{{exam.exam_id}} - {{exam.exam_name}} 完成情况
            <h6 class="text-muted text-truncate text-center"> {{exam.start_time}} - {{exam.end_time}} </h6>
            <a href="{% url 'coding:teacher-analysis' 'exam' exam.exam_id %}" class="mr-3 text-primary"
               data-toggle="tooltip" data-placement="top" title="" data-original-title="查看解析" style="float: right;">
                <i class="mdi mdi-eye font-size-18"></i> 详细数据
            </a>
            </h4>
            <div class="row text-center">
                <div class="col-4">
                    <h5 class="mb-0">{{exam.finish_info.0}}</h5>
                    <p class="text-muted text-truncate">已完成</p>
                </div>
                <div class="col-4">
                    <h5 class="mb-0">{{exam.finish_info.1}}</h5>
                    <p class="text-muted text-truncate">未完成</p>
                </div>
                <div class="col-4">
                    <h5 class="mb-0">{{exam.finish_info.6}}</h5>
                    <p class="text-muted text-truncate">总人数</p>
                </div>
            </div>
            <canvas id="pie-exam-{{exam.exam_id}}" height="260"></canvas>
        </div>
    </div>
</div>
{% endfor %}
```

图 6- 5 统计信息实现的 templates 层示例代码

7. 数据库应用系统运行环境

该系统的运行环境为支持 centos, ubuntu 等系统, mysql8.0 以上版本。

该系统开发完成后，按照软件工程中的测试方法，分别实施单元测试、功能测试、性能测试等，对于测试中发现的问题修改后，再次进行回归测试，待问题收敛后可以投入使用。

8. 案例总结

本案例描述一个 Web 数据库在线实验平台的设计与开发过程，该系统使用了 Python 的 Django 框架，数据库使用了 MySQL8.0 版本。该案例整体涉及实体 12 个（其中含有 5 个子类实体），20 张表。案例介绍中重点突出了数据库的设计与开发，包括需求分析、数据库设计、数据库实施等各个阶段，在数据库设计中详细描述的概念结构、逻辑结构和物理结构设计的全过程。案例讲解详尽，但由于涉及实体较多，略显复杂，可以进一步简化。