**2024-2025学年度第一学期**

**华南师范大学**

**计算机学院**

**《大型数据库技术及应用课程设计》报告**

|  |  |
| --- | --- |
| **题目：** | 基于PL/SQL的学生成绩管理系统 |
| **班级：** | 计算机科学与技术2班 计算机科学与技术3班 |
| **学号：** | 20212821020 20212821028 |
| **姓名：** | 林泽勋 林江荣 |
| **指导老师：** | 贺超波 |

目录

[1 背景介绍 1](#_Toc185000338)

[2 数据库设计 1](#_Toc185000339)

[2.1 概念设计 1](#_Toc185000340)

[2.2 逻辑设计 1](#_Toc185000341)

[2.3 物理设计 3](#_Toc185000342)

[3 关键技术实现 3](#_Toc185000343)

[4 测试 3](#_Toc185000344)

[5 总结 3](#_Toc185000345)

[参考文献 4](#_Toc185000346)

# 1 背景介绍

教育质量的高低，决定着一个国家的富强程度和发展水平，尤其是在高等教育领域，培养国家的脊梁和精英是高校的重要使命。随着科学技术的迅猛发展，计算机科学技术的进步不仅深刻改变了人们的生活方式，也在社会各领域发挥着越来越不可替代的作用。在高校教育领域，因特网技术的高速发展已经改变了传统的教学管理模式，并为教学模式提出了新的变革。特别是在高校学生成绩管理方面，计算机应用技术的使用已成为一个显著的趋势，其展现出的优势是传统手工管理无法比拟的[1]。

在这样的背景下，学生成绩管理系统（Student Grade Management System，简称SGMS）的应用，为高校教师提供了一个强大的工具，使他们能够更加高效地管理并分析所教授各个班级的成绩情况。通过这样的系统，教师可以轻松录入、查询、统计和分析学生成绩，从而更准确地把握学生的学习情况，及时调整教学策略，提高教学质量。

# 2 数据库设计

## 2.1 概念设计

概念设计的核心目的是为了建立概念数据模型。概念数据模型（Conceptual Data Model），简称概念模型，是面向数据库用户的现实世界的模型，主要用来描述世界的概念化结构，它使得我们在设计数据库的初始阶段，摆脱计算机系统及DBMS的具体技术问题，集中精力分析数据以及数据之间的联系等。

概念数据模型通常采用实体-关系图（E-R图）作为其标准描述方式。下图展示了本系统的E-R图，详细描绘了系统中的实体及其相互关系。

E-R图

## 2.2 逻辑设计

继概念设计阶段成功建立概念数据模型之后，即可进入逻辑设计的阶段。逻辑设计的主要任务是将概念模型转化为逻辑数据模型，这一模型将更加贴近于特定数据库管理系统的实现。这一阶段将详细定义数据结构、数据类型、属性、键以及实体之间的关系，同时考虑数据的完整性、一致性和安全性。

为了实现这一目标，我们选择采用关系数据模型作为逻辑数据模型的表现形式，依据E-R图建立本系统的关系数据模型，具体描述如下：

1. 学生表（stu\_tb）
   1. 记录ID（id）：表主键，整型，自动增长
   2. 学生姓名（name）：存储学生姓名，最大长度为100个字符
   3. 学号（campus\_id）：存储学生学号，最大长度为20个字符
   4. 专业（major）：存储学生所属专业名称，最大长度为100个字符
   5. 年级（grade）：存储学生的年级信息，最大长度为4个字符
2. 课程表（course\_tb）
   1. 课程ID（id）：表主键，整型，自动增长
   2. 课程名（name）：存储课程名称，最大长度为100个字符，不允许为空
   3. 学分（credit）：存储课程的学分数，类型为数字，保留一位小数
   4. 平时成绩占比（daily\_ratio）：存储平时成绩在总成绩中的占比，类型为数字，保留两位小数，取值范围为0到1
3. 班级表（class\_tb）
   1. 班级ID（id）：表主键，整型，自动增长
   2. 班级名（name）：存储班级名称，最大长度为100个字符，不允许为空
   3. 班级容量（capacity）：存储班级的最大容纳学生数，必须为正整数
4. 学生-班级表（stu\_class\_tb）
   1. 记录ID（id）：表主键，整型，自动增长
   2. 学生记录ID（stu\_id）：外键，存储关联的学生ID，不允许为空
   3. 班级ID（class\_id）：外键，存储关联的班级ID，不允许为空
5. 课程-班级表（cou\_class\_tb）
   1. 记录ID（id）：表主键，整型，自动增长
   2. 班级ID（class\_id）：外键，存储关联的班级ID，不允许为空
   3. 课程ID（course\_id）：外键，存储关联的课程ID，不允许为空
6. 学生-课程-成绩表（stu\_cou\_score\_tb）
   1. 记录ID（id）：表主键，整型，自动增长
   2. 学生记录ID（stu\_id）：外键，存储关联的学生ID，不允许为空
   3. 课程ID（course\_id）：外键存储关联的课程ID，不允许为空
   4. 平时成绩（daily\_score）：存储学生的平时成绩，取值范围为0到100
   5. 考试成绩（exam\_score）：存储学生的考试成绩，取值范围为0到100

## 2.3 物理设计

在本系统的物理设计阶段，将采用Oracle数据库和PL/SQL进行实现，以下是详细的物理设计步骤。

1. 确定存储结构和存取方法：本系统将在Oracle数据库中使用表空间来管理数据存储，为不同类型的数据选择合适的表空间，如用户数据、临时数据等；
2. 存储空间优化：利用Oracle的自动扩展表空间功能，以适应数据量的增长；除此之外，通过调整PCTFREE和PCTUSED参数，以优化行的插入和更新操作，以减少行链接、行迁移等情况；
3. 索引建立：针对查询频繁且数据量大的表，我们将实施索引策略以加速查询处理。例如，对于学生表（stu\_tb），可在学号（campus\_id）等关键字段上创建索引，以支持快速的数据检索和连接操作。这将显著提高查询响应时间，尤其是在执行联表查询和数据聚合时；

# 3 关键技术实现

数据库端编程、应用程序开发（如有）、数据库优化

## 3.1 数据库端技术实现

|  |
| --- |
| -- 创建表空间  CREATE TABLESPACE SGMS  DATAFILE 'D:\ora\oradata\ORCL2\SGMS.dbf' SIZE 100M  AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;  -- 创建用户  CREATE USER c##sgms IDENTIFIED BY sgms;  -- 将表空间设置为用户的默认表空间  ALTER USER c##sgms DEFAULT TABLESPACE SGMS;  -- 授予相关权限  GRANT UNLIMITED TABLESPACE TO c##sgms;  GRANT SELECT ANY TABLE TO c##sgms;  GRANT INSERT ANY TABLE TO c##sgms;  GRANT UPDATE ANY TABLE TO c##sgms;  GRANT DELETE ANY TABLE TO c##sgms;  GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE VIEW, CREATE PROCEDURE, CREATE TYPE, CREATE TRIGGER TO c##sgms;  -- 使得更改生效  ALTER USER c##sgms ACCOUNT LOCK;  ALTER USER c##sgms ACCOUNT UNLOCK; |

为了确保添加到班级的学生数量不会超过该班级的容量限制，我们设置了一个在向stu\_class\_tb表插入或更新数据前进行检查的触发器。触发器在每次插入或更新操作之前被激活，对每一行影响的数据执行以下操作：

1. 首先，触发器会计算出当前班级（即新插入或更新的班级）在stu\_class\_tb表中已有的学生数量，并将这个值存储在变量current\_count中。
2. 然后，触发器会从class\_tb表中获取对应班级的容量限制，并将这个值存储在变量class\_capacity中。
3. 如果current\_count + 1（即插入或更新后的学生数量）超过了class\_capacity（班级的容量限制），触发器会通过RAISE\_APPLICATION\_ERROR抛出一个自定义异常，错误信息为“班级学生数量超过容量限制”。这个异常会阻止当前的插入或更新操作，确保班级的学生数量不会超过设定的容量。

|  |
| --- |
| -- 创建触发器，在向stu\_class\_tb表插入或更新数据前进行检查  CREATE OR REPLACE TRIGGER check\_student\_count  BEFORE INSERT OR UPDATE ON stu\_class\_tb  FOR EACH ROW  DECLARE  current\_count NUMBER;  class\_capacity NUMBER;  BEGIN  -- 获取当前班级已有的学生数量  SELECT COUNT(\*) INTO current\_count  FROM stu\_class\_tb  WHERE class\_id = :NEW.class\_id;  -- 获取当前班级的容量限制  SELECT capacity INTO class\_capacity  FROM class\_tb  WHERE id = :NEW.class\_id;  -- 如果插入或更新后学生数量超过容量限制，则抛出异常  IF current\_count + 1 > class\_capacity THEN  RAISE\_APPLICATION\_ERROR(-20001, '班级学生数量超过容量限制');  END IF;  END;  / |

## 系统开发技术实现

本系统采用Vue3作为前端框架，结合Express框架，致力于构建一个现代化、高效、可靠的学生成绩管理系统。这种技术组合使得SGMS在提供强大的数据存储和快速处理能力的同时，也能够实现灵活的用户交互，满足高校教师在成绩管理方面的多样化需求。

### 3.2.1 前端部分

Vue3的响应式和组件化特性为用户提供了一个直观且流畅的操作界面，极大地方便教师日常的成绩管理工作。

### 3.2.2 后端部分

Express框架作为后端技术选型，以其轻量级和高性能的特点，为系统提供了强大的数据处理能力和稳定的服务支持，确保了后台运行的高效率和数据处理的安全性。

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境，它能让 JavaScript 在服务器端运行。结合 Express 框架，可快速搭建起功能强大的 Web 应用程序。而使用 oracledb 库，则能够与 Oracle 数据库进行高效交互，实现数据的存储、查询和更新等操作。将这三者结合起来，就可以开发出一个学生成绩管理系统，能够方便地对学生的成绩信息进行管理，包括录入、查询、统计分析以及成绩的更新等功能，为教学管理工作提供高效的信息化支持。

# 4 测试

对关键技术的实现效果进行验证

# 5 总结

不少于500字

# 参考文献

不少于5条

1. 陈亮.基于Web的高校学生成绩管理系统的设计和实现[D].湖北工业大学,2017.
2. 张孝斌,王超.基于PL/SQL的批量处理应用的性能优化策略[J].计算机系统应用,2016,25(05):1-7.

# 附录A Oracle建表（视图及索引）语句

|  |
| --- |
| -- 创建学生表  DROP TABLE stu\_tb;  CREATE TABLE stu\_tb (  id NUMBER GENERATED BY DEFAULT AS IDENTITY START WITH 1 INCREMENT BY 1,  name VARCHAR2(100),  campus\_id VARCHAR2(20),  major VARCHAR2(100),  grade VARCHAR2(4),  CONSTRAINT pk\_stu PRIMARY KEY (id)  );  -- 创建课程表  DROP TABLE course\_tb;  CREATE TABLE course\_tb (  id NUMBER GENERATED BY DEFAULT AS IDENTITY START WITH 1 INCREMENT BY 1,  name VARCHAR2(100) NOT NULL,  credit NUMBER(3,1),  daily\_ratio NUMBER(3,2) CHECK (daily\_ratio >= 0 AND daily\_ratio <= 1),  CONSTRAINT pk\_course PRIMARY KEY (id)  );  -- 课程名称索引  CREATE INDEX idx\_course\_name  ON course\_tb (name);  -- 创建班级表  DROP TABLE class\_tb;  CREATE TABLE class\_tb (  id NUMBER GENERATED BY DEFAULT AS IDENTITY START WITH 1 INCREMENT BY 1,  name VARCHAR2(100) NOT NULL,  capacity NUMBER CHECK (capacity > 0),  CONSTRAINT pk\_class PRIMARY KEY (id)  );  -- 班级名称索引  CREATE INDEX idx\_class\_name  ON class\_tb (name);  -- 创建学生-班级关联表  DROP TABLE stu\_class\_tb;  CREATE TABLE stu\_class\_tb (  id NUMBER GENERATED BY DEFAULT AS IDENTITY START WITH 1 INCREMENT BY 1,  stu\_id NUMBER NOT NULL,  class\_id NUMBER NOT NULL,  CONSTRAINT pk\_stu\_class PRIMARY KEY (id),  CONSTRAINT fk\_stu\_id FOREIGN KEY (stu\_id) REFERENCES stu\_tb(id) ON DELETE CASCADE,  CONSTRAINT fk\_class\_id FOREIGN KEY (class\_id) REFERENCES class\_tb(id) ON DELETE CASCADE  );  -- 创建课程-班级关联表  DROP TABLE cou\_class\_tb;  CREATE TABLE cou\_class\_tb (  id NUMBER GENERATED BY DEFAULT AS IDENTITY START WITH 1 INCREMENT BY 1,  class\_id NUMBER NOT NULL,  course\_id NUMBER NOT NULL,  CONSTRAINT pk\_cou\_class PRIMARY KEY (id),  CONSTRAINT fk\_cou\_class\_class\_id FOREIGN KEY (class\_id) REFERENCES class\_tb(id) ON DELETE CASCADE,  CONSTRAINT fk\_cou\_class\_course\_id FOREIGN KEY (course\_id) REFERENCES course\_tb(id) ON DELETE CASCADE  );  -- 创建学生-课程-成绩表  DROP TABLE stu\_cou\_score\_tb;  CREATE TABLE stu\_cou\_score\_tb (  id NUMBER GENERATED BY DEFAULT AS IDENTITY START WITH 1 INCREMENT BY 1,  stu\_id NUMBER NOT NULL,  course\_id NUMBER NOT NULL,  daily\_score NUMBER CHECK (daily\_score >= 0 AND daily\_score <= 100),  exam\_score NUMBER CHECK (exam\_score >= 0 AND exam\_score <= 100),  CONSTRAINT pk\_stu\_cou\_score PRIMARY KEY (id),  CONSTRAINT fk\_stu\_cou\_score\_stu\_id FOREIGN KEY (stu\_id) REFERENCES stu\_tb(id) ON DELETE CASCADE,  CONSTRAINT fk\_stu\_cou\_score\_course\_id FOREIGN KEY (course\_id) REFERENCES course\_tb(id) ON DELETE CASCADE  );  -- 创建学生-课程-成绩查询视图  CREATE OR REPLACE VIEW stu\_cou\_score\_view AS  SELECT  scs.stu\_id,  scs.course\_id,  scs.daily\_score,  c.daily\_ratio,  scs.exam\_score,  1 - c.daily\_ratio AS exam\_ratio, -- 计算考试成绩占比  scs.daily\_score \* c.daily\_ratio + scs.exam\_score \* (1 - c.daily\_ratio) AS final\_score -- 按照上述规则计算最终成绩  FROM stu\_cou\_score\_tb scs  JOIN course\_tb c ON scs.course\_id = c.id; |