

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS2008 |
| 学 号： | U202015533 |
| 姓 名： | 徐瑞达 |
| 指导教师： | 左 琼 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2022年 11 月 20 日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |

|  |  |
| --- | --- |
| 总分 |  |

目 录

[1 课程任务概述 1](#_Toc120637276)

[2 任务实施过程与分析 2](#_Toc120637277)

[2.1 数据库、表与完整性约束的定义(Create) 2](#_Toc120637278)

[2.2 表结构与完整性约束的修改(ALTER) 2](#_Toc120637279)

[2.3 数据查询(Select) 3](#_Toc120637280)

[2.4 数据的插入、修改与删除(Insert,Update,Delete) 7](#_Toc120637281)

[2.5 视图 8](#_Toc120637282)

[2.6 存储过程与事务 8](#_Toc120637283)

[2.7 触发器 10](#_Toc120637284)

[2.8 用户自定义函数 10](#_Toc120637285)

[2.9 安全性控制 11](#_Toc120637286)

[2.10 并发控制与事务的隔离级别 11](#_Toc120637287)

[2.11 数据库应用开发(JAVA篇) 11](#_Toc120637288)

[2.12 备份+日志：介质故障与数据库恢复 12](#_Toc120637289)

[2.13 数据库设计与实现 13](#_Toc120637290)

[2.14 数据库的索引B+树实现 14](#_Toc120637291)

[3 课程总结 24](#_Toc120637292)

# 课程任务概述

“数据库系统原理实践”是配合“数据库系统原理”课程独立开设的实践课，注重理论与实践相结合。本课程以OpenGauss为例，系统性地设计了一系列的实训任务，内容涉及以下几个部分：

1. 数据库、表、索引、视图、约束、存储过程、函数、触发器、游标等数据对象的管理与编程
2. 数据查询，数据插入、删除与修改等数据处理相关任务
3. 数据库的安全性控制，完整性控制，恢复机制，并发控制机制等系统内核的实验
4. 数据库的设计与实现
5. 数据库应用系统的开发(JAVA篇)
6. 数据库内核实验(B+树)

课程依托头歌实践教学平台，采用Linux操作系统下的OpenGauss2.1实验环境。在数据库应用开发环节，使用JDK 1.8实验环境。

本实践课程旨在帮助学生熟悉数据库的编程操作、数据库系统的内核机制，掌握数据处理、数据库设计以及使用JAVA进行数据库应用开发。同时，通过B+树系列实验帮助学生了解数据库底层的索引结构。

# 任务实施过程与分析

## 数据库、表与完整性约束的定义(Create)

本小节主要涉及使用CREATE完成数据库、数据表的创建，各种约束（主码约束、外码约束、CHECK约束、DEFAULT约束、UNIQUE约束）的定义。

### 创建数据库

该关卡任务已完成，实施情况本报告略过。

### 创建表及表的主码约束

该关卡任务已完成，实施情况本报告略过。

### 创建外码约束(foreign key)

该关卡任务已完成，实施情况本报告略过。

### CHECK约束

该关卡任务已完成，实施情况本报告略过。

### DEFAULT约束

该关卡任务已完成，实施情况本报告略过。

### UNIQUE约束

该关卡任务已完成，实施情况本报告略过。

## 表结构与完整性约束的修改(ALTER)

本小节主要涉及在数据表创建后使用ALTER修改表名、增删改字段、增删改约束。

### 修改表名

该关卡任务已完成，实施情况本报告略过。

### 添加与删除字段

该关卡任务已完成，实施情况本报告略过。

### 修改字段

该关卡任务已完成，实施情况本报告略过。

### 添加、删除与修改约束

该关卡任务已完成，实施情况本报告略过。

## 数据查询(Select)

本小节使用SELECT完成数据查询，涉及单表查询、多表查询、嵌套查询、多条件查询、子查询、派生表、连接查询、分组统计、聚合函数（COUNT、MAX、SUM、RANK、DENSE\_RANK）的使用、各种函数（IF、CASE、日期函数）的使用、NULL值的判断、列的别名、各种关键字（LIKE、ALL、BETWEEN、DISTINCT）的用法、各种谓词（NOT EXISTS、NOT IN）的用法。

### 查询客户主要信息

该关卡任务已完成，实施情况本报告略过。

### 邮箱为null的客户

该关卡任务已完成，实施情况本报告略过。

### 既买了保险又买了基金的客户

该关卡任务已完成，实施情况本报告略过。

### 办理了储蓄卡的客户信息

该关卡任务已完成，实施情况本报告略过。

### 每份金额在30000～50000之间的理财产品

该关卡任务已完成，实施情况本报告略过。

### 商品收益的众数

该关卡任务已完成，实施情况本报告略过。

### 未购买任何理财产品的武汉居民

该关卡任务已完成，实施情况本报告略过。

### 持有两张信用卡的用户

该关卡任务已完成，实施情况本报告略过。

### 购买了货币型基金的客户信息

该关卡任务已完成，实施情况本报告略过。

### 投资总收益前三名的客户

该关卡任务已完成，实施情况本报告略过。

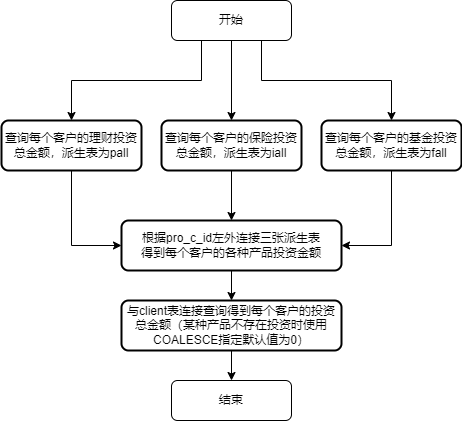
### 黄姓客户持卡数量

该关卡任务已完成，实施情况本报告略过。

### 客户理财、保险与基金投资总额

客户的资产信息都会被存储在property表中，而不同类型的投资产品需要通过查询对应的表计算投资金额。因此需要分别查询每个客户的理财、保险、基金的投资总金额。每种投资产品的查询结果分别作为派生表pall、iall、fall，然后使用左外连接将三张派生表根据pro\_c\_id连接为一张表，即可得到每个客户的各种产品投资金额。根据这张表和client连接查询即可得到欲得到的查询结果。然而，此时没有资产的客户将不会出现在查询结果中，不符合预期结果，需要使用COALESCE函数指定值为NULL时的默认值。

流程图如图2.1所示。



**图2.1 题目2.3.12流程图**

### 客户总资产

在任务2.3.12种，计算了客户的投资总额。在本任务中，求客户总资产时，还需要将投资收益、储蓄卡余额、信用卡透支余额计算在内，因此需要在任务2.3.12的基础上使用左外连接连接储蓄卡余额派生表ball和信用卡透支余额派生表b2all，同时使用COALESCE函数指定值为NULL时的默认值。

### 第N高问题

该关卡任务已完成，实施情况本报告略过。

### 基金收益两种方式排名

该关卡任务已完成，实施情况本报告略过。

### 拥有完全相同基金组合的客户

在该任务中，可以分别选择出任意两个用户持有的基金组合，使用union all语句判断两个结果集是否相同来判断是否持有相同基金组合。关键代码如下：

where not exists (select pro\_pif\_id from ((SELECT pro\_pif\_id FROM property WHERE pro\_type=3 and pro\_c\_id=c\_id1 group by pro\_pif\_id)union all(SELECT pro\_pif\_id FROM property WHERE pro\_type=3 and pro\_c\_id=c\_id2 group by pro\_pif\_id)) group by pro\_pif\_id having count(\*) < 2)

### 购买基金的高峰期

在该任务中，设三天的日期分别为t1、t2、t3，则应满足t3=t2+1且t2=t1+1或者跨越非交易日（需要用到时间函数）。由于选择符合要求的连续三个交易日的SQL语句过长，可以使用with as语句构建子查询，最终的子查询含有6个字段，分别表示符合要求的连续三个交易日的日期和金额。因此，最终可以分别选取每日的日期和金额，使用union进行合并。

### 至少有一张信用卡余额超过5000元的客户信用卡总余额

该关卡任务已完成，实施情况本报告略过。

### 以日历表格式显示每日基金购买总金额

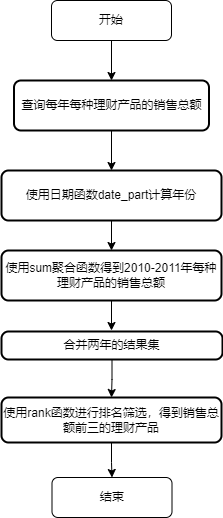
该任务主要考察case函数的用法。首先需要从property和fund表中计算出每天的基金购买总金额（同时计算周次和日期在周内的天数），然后使用case语句指定某一列的值。

以周一的计算为例，case day\_of\_week when 1 then amount else null end的含义指，当day\_of\_week为1时，值为amount，否则值为null。

### 查询销售总额前三的理财产品

首先查询每年每种理财产品的销售总额，查询时使用OpenGauss日期函数date\_part(‘year’,pro\_purchase\_time)处理得到年份，然后使用sum聚合函数得到2010年和2011年两年每种理财产品的销售总额，最后使用union合并结果集；然后对销售总额进行排名筛选，排名时根据pyear进行分组统计，根据sumamount进行降序排名；最后筛选出rk<=3的结果即为销售总额前三的理财产品。

流程图如图2.2所示。



**图2.2 题目2.3.20流程图**

### 投资积极且偏好理财类产品的客户

首先查询每个客户购买的理财产品种类数和基金种类数，然后使用左外连接连接结果集，最后统计pcount>=3且pcount>fount的用户即为投资积极且偏好理财类产品的客户集合。

### 查询购买了所有畅销理财产品的客户

对于每个用户，将畅销理财产品和该用户购买的理财产品列表进行左外连接，如果连接结果与畅销理财产品一致，说明该用户购买了所有畅销理财产品。判断两个结果集是否一致时，使用union all语句合并，然后使用count(\*)语句统计是否有记录只出现一次，如果有则说明两个结果集不一致。

### 查找相似的理财产品

首先查询拥有产品14最多的3个客户（使用dense\_rank排序），然后统计这3个客户持有的所有理财产品，最后统计这些理财产品的持有总人数并排名即可得到产品14的相似理财产品的相似度与排名。

### 查询任意两个客户的相同理财产品数

首先查询理财产品数至少2种的用户，作为子查询表base\_user\_list；然后查询base\_user\_list中用户的理财产品列表，作为子查询表base\_pif\_list；接着查询任意两个用户的相同理财产品，作为子查询表base\_common\_list；最后利用base\_common\_list表，查询出用户对的记录条数至少两条的用户对。

流程图如图2.3所示。



**图2.3 题目2.3.24流程图**

### 查找相似的理财客户

## 数据的插入、修改与删除(Insert,Update,Delete)

本小节主要涉及数据的插入、批量插入、更新、删除以及连接更新。

### 插入多条完整的客户信息

该关卡任务已完成，实施情况本报告略过。

### 插入不完整的客户信息

该关卡任务已完成，实施情况本报告略过。

### 批量插入数据

该关卡任务已完成，实施情况本报告略过。

### 删除没有银行卡的客户信息

该关卡任务已完成，实施情况本报告略过。

### 冻结客户资产

该关卡任务已完成，实施情况本报告略过。

### 连接更新

该关卡任务已完成，实施情况本报告略过。

## 视图

本小节主要涉及视图的创建与查询。

### 创建所有保险资产的详细记录视图

该关卡任务已完成，实施情况本报告略过。

### 基于视图的查询

该关卡任务已完成，实施情况本报告略过。

## 存储过程与事务

本小节主要涉及存储过程的定义、创建与调用，变量的定义与赋值，流程控制语句的使用，游标的定义与使用，事务的定义与使用。

### 使用流程控制语句的存储过程

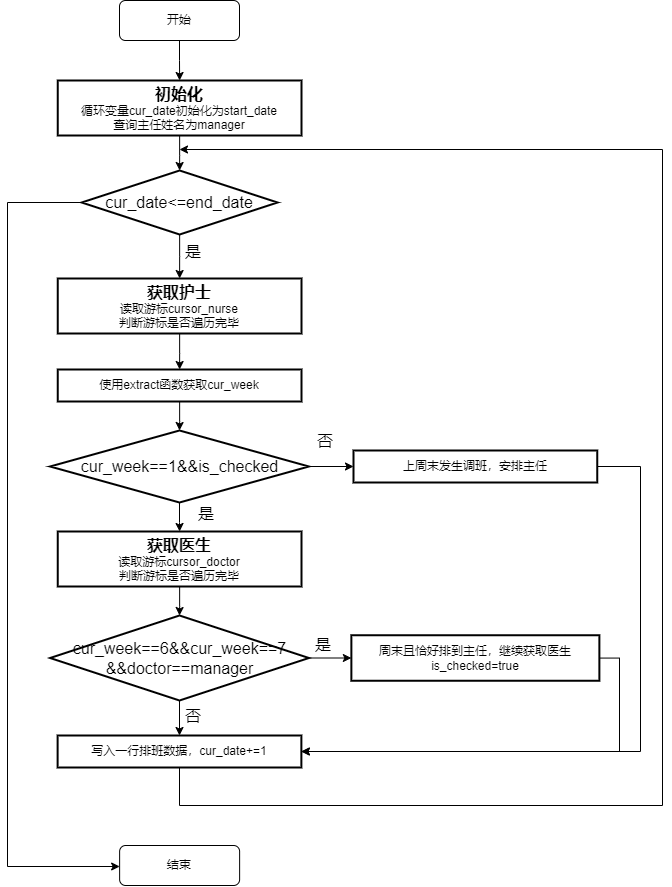
该任务要求创建存储过程，向表中插入斐波那契数列的前n项。首先定义初始值n1、n2、n3和循环变量i，在存储过程中，先根据n的值依次插入初始值，然后通过循环控制插入n>3之后的值，重点需要注意if语句和while loop语句的使用。

### 使用游标的存储过程

首先定义cur\_date和cur\_week标识当前日期和周次，定义manager、doctor、nurse1、nurse2依次标识主任、医生、护士1、护士2，定义is\_checked标识主任是否需要调班，定义游标cursor\_doctor遍历医生、游标cursor\_nurse遍历护士。

在存储过程中，首先将排班开始时间赋给cur\_date，查询科室主任的姓名赋给manager；然后定义while cur\_date<= end\_date loop循环，在循环中，使用时间函数extract计算cur\_week；获取两个护士的名字（获取后需要根据cursor\_nurse% NOTFOUND判断游标是否遍历完毕，如果遍历完毕则重新打开游标获取护士名字）；如果当前为周一且主任需要调班，则进行调班；如果当前为周六周日，则先获取医生，如果该医生恰好为manager，那么标识需要调班并再次获取医生；其余情况下均正常获取医生；最后将选取得到的doctor、nurse1、nurse2写入数据库，并自增cur\_date进行下一次循环。

流程图如图2.4所示。



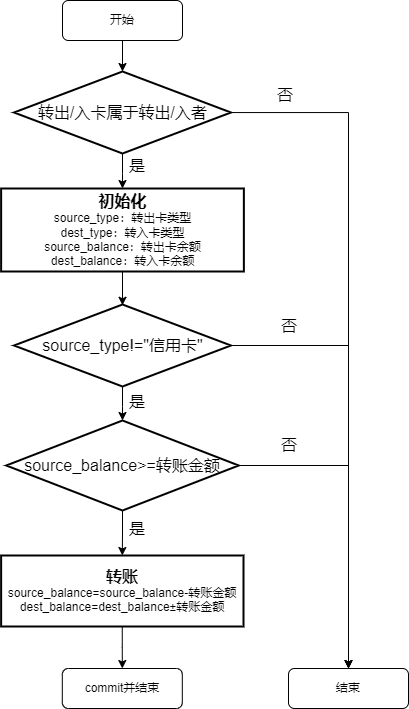
**图2.4 题目2.6.2流程图**

### 使用事务的存储过程

首先定义source\_type、dest\_type、source\_balance、dest\_balance分别标识转出卡类型、转入卡类型、转出卡余额、转入卡余额。

在存储过程中，首先查询转出卡和转入卡是否为转出者和转入者的银行卡，如果是，则依次取出转出卡类型、转入卡类型、转出卡余额、转入卡余额；如果转出卡是信用卡或转出卡余额少于转账金额，退出存储过程；然后更新转出卡余额，并根据转入卡类型更新转入卡余额，最后commit提交修改。

流程图如图2.5所示。



**图2.5 题目2.6.3流程图**

## 触发器

本小节主要涉及触发器的创建、触发器内的特殊变量（OLD、NEW）、字符串函数concat的使用。

### 为投资表property实现业务约束规则-根据投资类别分别引用不同表的主码

首先定义p\_type、pif\_id分别标识待插入记录的资产类型和对应的资产编号。

在触发器函数中，依次根据p\_type查询pif\_id是否在对应表中，如果不在，则抛出异常（使用concat函数连接字符串）；最后如果p\_type不合法，抛出pro\_type不合法的异常。

## 用户自定义函数

本小节涉及用户自定义函数的定义、在SELECT语句中使用自定义函数。

### 创建函数并在语句中使用它

该关卡任务已完成，实施情况本报告略过。

## 安全性控制

本小节主要涉及用户、角色的创建，权限的授予与撤销，通过角色授权。

### 用户和权限

该关卡任务已完成，实施情况本报告略过。

### 用户、角色与权限

该关卡任务已完成，实施情况本报告略过。

## 并发控制与事务的隔离级别

本小节主要涉及OpenGauss的事务隔离级别、并发操作可能产生的数据不一致情况（读脏、不可重复读、幻读）、加锁机制。

### 不可重复读

该关卡任务已完成，实施情况本报告略过。

### 幻读

该关卡任务已完成，实施情况本报告略过。

### 主动加锁保证可重复读

该关卡任务已完成，实施情况本报告略过。

### 可串行化

该关卡任务已完成，实施情况本报告略过。

## 数据库应用开发(JAVA篇)

本小节主要涉及JDBC的核心组件（DriverManager、Driver、Connection、Statement、PreparedStatement、ResultSet、SQLException）的使用、JDBC中SQL语句的书写与执行、JDBC中的事务处理。

### JDBC体系结构和简单的查询

JDBC体系进行数据库操作的基本流程如下：

* 使用Class.forName("org.postgresql.Driver")注册JDBC驱动程序
* 使用DriverManager.getConnection(URL, USER, PASS)打开数据库连接
* 使用connection.createStatement()构建SQL查询对象
* 使用executeQuery()或executeUpdate()执行SQL语句并返回结果集
* 使用resultSet.next()和resultSet.getXXX()方法从结果集中提取数据
* 使用close()释放connection、statement、resultset等数据库资源

以上步骤均需要在try catch语句中执行，以处理SQLException异常。

### 用户登录

该关卡任务已完成，实施情况本报告略过。

### 添加新客户

使用PreparedStatement对象，可以将SQL语句中变化的部分作为参数传入，然后使用executeQuery()或executeUpdate()执行查询或增删改操作。

### 银行卡销户

该关卡任务已完成，实施情况本报告略过。

### 客户修改密码

该关卡任务已完成，实施情况本报告略过。

### 事务与转账操作

JDBC中通过使用connection.setAutoCommit(false)使会话期内的语句不再自动提交，由此可以在try catch语句的catch子句中使用rollback方法撤销修改，实现事务回滚操作。

### 把稀疏表格转为键值对存储

在该任务中，insertSC函数可以实现将某个键值对插入到sc数据表中，因此只需遍历所有的键值对。

这里定义String数组cols存储所有科目字段名，在main函数中，查询数据表entrance\_exam的所有记录获得结果集resultSet，然后遍历结果集中的每一条记录，在该记录中查询每个科目的值，如果值不为空，则插入新的键值对到sc表中，从而实现从稀疏表格到键值对的转储。

## 备份+日志：介质故障与数据库恢复

本小节主要涉及OpenGauss的恢复机制、使用gs\_dump和gs\_restore完成数据库的备份与恢复。

### 备份与恢复

该关卡任务已完成，实施情况本报告略过。

## 数据库设计与实现

本小节主要涉及概念模型的设计、从概念模型到DBMS的实现、ER图的使用与对应的关系模式、建模工具PowerDesigner的使用。

### 从概念模型到OpenGauss实现

该任务中，根据ER图创建各数据表、主码约束、外码约束、索引、UNIQUE约束、NOT NULL约束即可。需要特殊处理的情况包括关键字冲突（如user、password、name）、UNIQUE INDEX、缺省值、字段类型。

### 从需求分析到逻辑模型

该关卡任务已完成，实施情况本报告略过。

### 建模工具的使用

该关卡任务已完成，实施情况本报告略过。

### 制约因素分析与设计

在完成“从需求分析到逻辑模型”关卡的过程中，考虑了许多制约因素。在业务流程的设计中，考虑了各种实际需求，如是否需要字段唯一以满足机票唯一等需求，根据实际需求设定字段类型等。在设计数据的约束时，考虑了各种外键约束、UNIQUE约束、NOTNULL约束。在设计用户操作权限时，考虑了用户可以为他人购票等实际情况，切实提升了数据库结构的功能性、稳定性。

### 工程师责任及其分析

工程师在设计实现工程时，应综合考虑项目的可行性、实施难度以及项目实施占用的资源，同时需要评估项目实施会对社会、健康、安全等造成的影响。工程师需要综合评估工程的安全指数，查找可能存在的漏洞，并为项目实施后可能产生的各种结果负责。

## 数据库的索引B+树实现

本小节主要涉及B+树的结构特征、结点父类BPlusTreePage的结构功能与设计、内部结点BPlusTreeInternalPage的结构功能与设计、叶子结点BPlusTreeLeafPage的结构功能与设计、B+树索引的插入过程、B+树迭代器的功能与实现、B+树索引的删除过程。

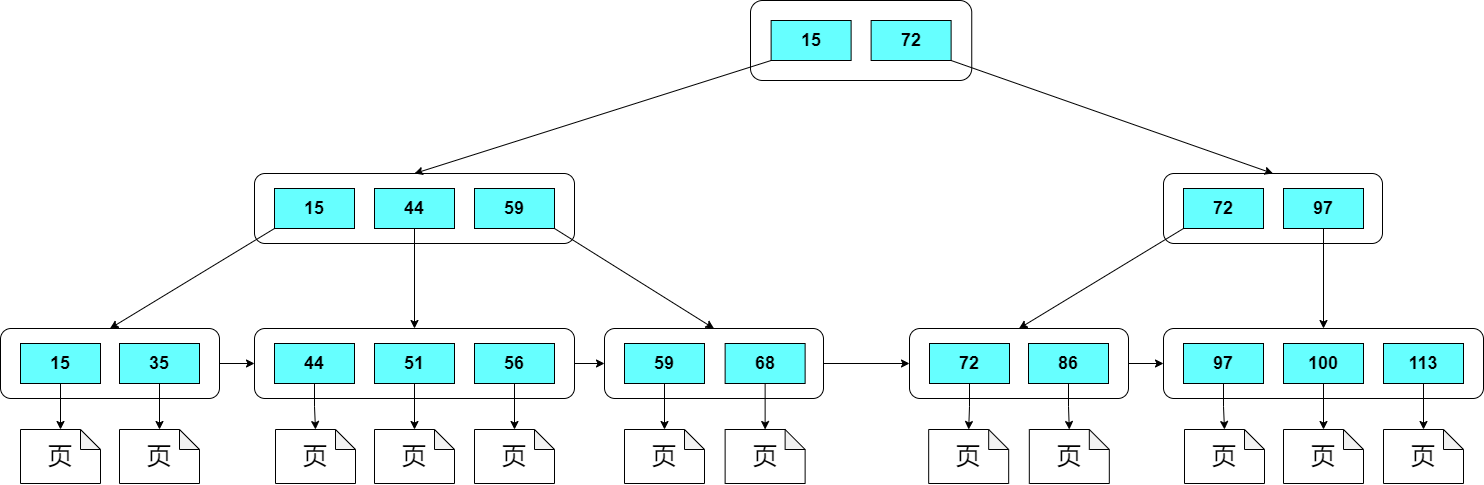
### BPlusTreePage的设计

1. 基本概述

与B树不同，B+树的根页和内部页不存储数据，只存储子页的索引，而叶子页才存储数据；每个叶子页增加指向相邻叶子页的指针，每个叶子页的最后一个数据指向下一个叶子页的第一个数据，形成一个有序链表。

由于B+树上的每个节点可以看作对应一个磁盘页，因此这里将节点称为页，而将内部页和叶子页中的每个键值对称为结点（或以元素、键值对相称），以避免混淆。

本节使用以下B+树模型完成对关键函数的解释说明，如图2.6所示。



**图2.6 B+树模型**

1. B+树内部页

在B+树中每个内部页中，结点的关键字是**由小到大有序排列**的。每个键值对的值都是指向某个子页的指针。每个内部页的结点数量最多为max\_size，特别地，根页至少包含两个指向子树的结点指针。

内部页中存储m个键和m+1个指针值，因此设置第一个键为INVALID。本实验中，父页中的每个关键字都是对应子页中的最小关键字。

如图2.6所示，根页【15，72】的结点15是对应子页【15，44，59】的最小关键字，叶子页【44，51，56】中的结点均大于等于44且小于59。

1. B+树叶子页

在B+树中每个叶子页中，结点的关键字也是**由小到大有序排列**的。每个键值对的值都是指向磁盘上某个数据块的指针。与内部页相同，叶子页的结点数量也有限制。

1. BPlusTreePage类

BPlusTreePage是内部页BPlusTreeInternalPage和叶子页BPlusTreeLeafPage的父类，定义了B+树页的基本属性和函数。基本属性见表2.1，在该任务中依次实现该表中各个属性的Getter和Setter函数即可。

**表2.1 BPlusTrrePage类的属性**

| 属性 | 数据类型 | 说明 |
| --- | --- | --- |
| page\_type\_ | IndexPageType | 页类型，可选值为INVALID\_INDEX\_PAGE,LEAF\_PAGE,INTERNAL\_PAGE |
| size\_ | int | 页存储的键值对数量 |
| max\_size\_ | int | 页存储的键值对最大数量 |
| page\_id\_ | page\_id\_t | 页id |
| parent\_page\_id\_ | page\_id\_t | 父页的id，对于根节为INVALID\_PAGE\_ID |
| lsn\_ | lsn\_t | 日志序列号 |

### BPlusTreeInternalPage的设计

1. 基本概述

B+树内部页不存储数据，只存储索引信息。

涉及到的数据结构为#define MappingType std::pair<KeyType, ValueType>，对于内部页，ValueType为page\_id\_t，指向某个子页；对于叶子页，ValueType为RID，标识数据在磁盘中的位置。

1. BPlusTreeInternalPage类

BPlusTreeInternalPage类需要实现的函数见表2.2。

**表2.2 BPlusTreeInternalPage类的函数**

| 函数 | 说明 |
| --- | --- |
| Init | 初始化该内部页的id、parent\_id、max\_size等属性 |
| KeyAt | 根据索引index获取键key |
| SetKeyAt | 设置索引index处的键key |
| ValueIndex | 获取值value对应的索引index |
| ValueAt | 获取索引index对应的值value |
| Lookup | 根据键key查找值value，不存在时返回最接近的较大key对应的值value |
| PopulateNewRoot | 生成新的根元素，并设置根元素 |
| InsertNodeAfter | 在old\_value后添加新节点<new\_key,new\_value> |
| Remove | 移除索引index处的结点 |
| RemoveAndReturnOnlyChild | 当根页只有一个子页时，将根页删除，并将子页提升为根页 |
| MoveAllTo | 移动该页内所有结点到页recipient中 |
| MoveHalfTo | 移动该页内一半结点到页recipient中 |
| MoveFirstToEndOf | 移动该页的首元素到页recipient尾部 |
| MoveLastToFrontOf | 移动该页的尾元素到页recipient首部 |
| CopyNFrom | 从item处截取size个结点并入到该页的尾部 |
| CopyLastFrom | 将pair元素添加到该页尾部 |
| CopyFirstFrom | 将pair元素添加到该页首部 |

1. Lookup

Lookup函数用于查找键key对应的value（这里的value是page\_id，即查找键key对应的页）。由于B+树的每一页的结点是按照key递增的，因此可以采用二分法提高查找效率。如果没有找到key对应的结点，返回最接近的较大key对应的value。

关键点：由于本次实验中，父页的某个关键字是对应子页中的**最小关键字**，因此，当没有找到key对应的结点时，将返回最接近的较小key对应的value。

如图2.6所示，当在内部页【15，44，59】中查找关键字51时，将返回关键字44对应的指针，其指向的叶子页【44，51，56】恰好含有关键字51。

1. PopulateNewRoot和InsertNodeAfter

当根页分裂时，原来的根页变为内部页，PopulateNewRoot函数用于创建新的根页。当内部页分裂时，需要调用InsertNodeAfter在该页的父页中插入新分裂页的索引，在该函数中根据old\_value获取old\_index，在其后插入新元素即可。

1. MoveAllTo

当前页删除某个键值对后，结点数量小于min\_size，而且兄弟页的结点数量也很少时，需要进行内部页合并。在该函数中，通过循环逐个移动结点到recipient中，同时还要修改每个结点指向的子页的父页id为recipient的page\_id。

1. MoveHalfTo和CopyNFrom

页分裂时，通过待分裂页调用MoveHalfTo方法将一半结点移动到新页中实现。在该函数中，需要新页recipient调用CopyNFrom方法实现移动结点。

CopyNFrom函数的参数为待移动结点数组的指针MappingType \*items，待移动结点的数目size，在函数中，通过循环逐个移动结点，同时还要修改每个结点指向的子页的父页id。

1. MoveFirstToEndOf和MoveLastToFrontOf

当前页删除某个键值对后，结点数量小于min\_size，而且兄弟页的结点数量很满时，需要进行重分配。在该函数中，将当前页的首结点移动至recipient的尾部，需要调用memmove方法将数组整体向前移动一个单位，然后recipient调用CopyLastFrom方法将当前页的首结点移动至recipient中，最后父页需要修改指向该页的键值对中的value。MoveLastToFrontOf与之类似。

### BPlusTreeLeafPage的设计

1. 基本概述

叶子页与内部页大致相同，需要使用MappingType数组存储键值对，不同的是叶子页存储的Value值通常标识数据在磁盘中位置。同时，叶子页包含属性next\_page\_id\_指向相邻叶子页。

1. BPlusTreeLeafPage类

BPlusTreeLeafPage类需要实现的函数见表2.3。

**表2.3 BPlusTreeLeafPage类的函数**

| 函数 | 说明 |
| --- | --- |
| Init | 初始化该叶子页的id、parent\_id、max\_size等属性 |
| GetNextPageId | 获取相邻叶子页的id |
| SetNextPageId | 设置相邻叶子页 |
| KeyAt | 根据索引index获取键key |
| KeyIndex | 获取键key对应的索引index |
| GetItem | 根据索引index获取pair |
| Lookup | 根据键key查找值value，如果key不存在，返回false |
| Insert | 在保证有序的情况下插入新的结点 |
| RemoveAndDeleteRecord | 移除该页中键key对应的元素 |
| MoveAllTo | 移动该页内所有结点到页recipient中 |
| MoveHalfTo | 移动该页内一半结点到页recipient中 |
| MoveFirstToEndOf | 移动该页的首元素到页recipient尾部 |
| MoveLastToFrontOf | 移动该页的尾元素到页recipient首部 |
| CopyNFrom | 从item处截取size个结点并入到该页的尾部 |
| CopyLastFrom | 将pair元素添加到该页尾部 |
| CopyFirstFrom | 将pair元素添加到该页首部 |

1. KeyIndex和Lookup

KeyIndex函数用于查找键key对应的索引index。由于叶子页中的键值对按照键递增排序，因此可以使用二分法查找，如果键key不存在，则返回应插入位置的index。Lookup函数用于找到键key对应的值value，在该函数中，首先调用keyIndex查找键key对应的索引index，如果index处的键等于key，说明键key存在，此时返回true。

如图2.6所示，当在叶子页【44，51，56】中查找键48时，将返回索引0，表示键48应插入的位置。

1. Insert和RemoveAndDeleteRecord

Insert函数用于插入新的键值对。在该函数中首先调用KeyIndex查找应该插入key的位置index，将数组整体后移后，插入新的键值对。

RemoveAndDeleteRecord函数用于删除键key对应的键值对。在该函数中首先调用KeyIndex查找键key的位置index，如果查找到，则删除键值对并将该键值对后面的所有键值对向前移动一个单位。

1. 合并、分裂与重分配

叶子页的合并、分裂与重分配过程与内部页基本相同，具体函数如MoveAllTo、MoveHalfTo和CopyNFrom、MoveFirstToEndOf和CopyLastFrom、MoveLastToFrontOf和CopyFirstFrom等实现均基本一致。

### B+树索引：Insert

1. 基本概述

一颗完整的B+树从最初的空树，随着记录的插入，叶子页不断分裂增加并产生用于索引的内部页，逐渐生成完整的B+树。

实现插入功能后，还需实现对B+树的查找和迭代功能。

1. BPlusTree类

BPlusTree类需要实现的函数见表2.4。

**表2.4 BPlusTree类的函数（Insert相关）**

| 函数 | 说明 |
| --- | --- |
| IsEmpty | B+树是否为空 |
| Insert | 插入键值对 |
| GetValue | 获取键key对应的值value（这里的value指RID） |
| FindLeafPage | 根据键key获取其所在的叶子页，leftMost为true时返回最左叶子页 |
| StartNewTree | 当B+树为空时，初始化生成新B+树以插入记录 |
| InsertIntoLeaf | 在正确的叶子页中插入结点 |
| InsertIntoParent | 当生成新页后，在其父页中插入正确的索引 |
| Split | 分裂内部页或内部页 |
| UpdateRootPageId | 当B+树根页变化时，在header\_page中对根页ID进行更新 |
| Begin | 返回指向B+树特定记录的迭代器 |
| End | 返回IsEnd状态的B+树记录迭代器 |

1. IndexIterator类

IndexIterator类需要实现的函数见表2.5。

**表2.5 IndexIterator类的函数**

| 函数 | 说明 |
| --- | --- |
| IsEnd | 迭代器是否遍历完毕 |
| operator\* | 返回迭代器在当前页中指向的键值对 |
| operator++ | 迭代器移动至下一页后返回当前迭代器 |

1. FindLeafPage和GetValue

FindLeafPage函数用于查找键key对应键值对所在的叶子页。在该函数中，首先判断B+树是否为空，为空则返回空指针，否则使用for循环不断向下搜索。从根页开始搜索，如果leftMost为true，那么记录BPlusTreeInternalPage的最左结点的page\_id，否则调用BPlusTreeInternalPage类的Lookup函数查找键key对应的page\_id（在该函数中，如果未找到key对应的键值对，将返回最接近的较大key对应的page\_id），然后根据page\_id继续向下搜索，直至到达叶子页结束并返回该叶子页。

GetValue函数用于获取键key对应的value。在该函数中，首先调用FindLeafPage函数获取键key所在的叶子页，然后在调用该叶子页的Lookup函数查找键key对应的键值对，如果查找成功，则调用参数result的push\_back函数添加获取到的值value，否则返回false。

如图2.6所示，当要在B+树中查找键82时，首先查找根页【15，72】，指向内部页【72，97】，而后指向叶子页【72，86】，返回该叶子页的id即可。

1. Insert

Insert函数用于向B+树中插入新的键值对。在该函数中，如果B+树为空，则调用StartNewTree函数初始化新B+树；否则调用InsertIntoLeaf函数向叶子页中插入键值对。

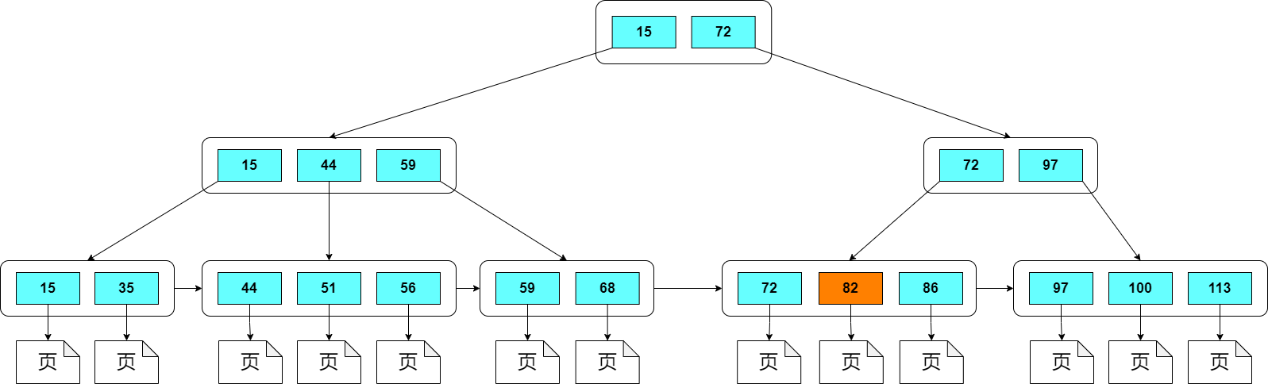
1. StartNewTree

StartNewTree函数用于当B+树为空时，使用待插入键值对初始化新B+树。初始化后，根页为叶子页。在该函数中，首先调用BufferPoolManager的NewPage函数初始化根页，然后调用根页的Init函数初始化页id，调用Insert函数插入新键值对，调用UpdateRootPageId函数更新header\_page中的根页记录，最后以脏页形式unpin根页。

1. InsertIntoLeaf

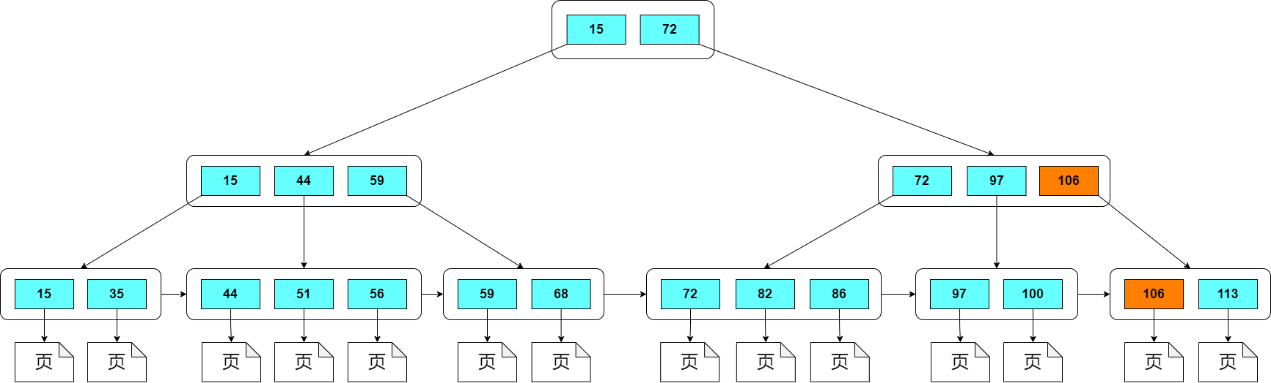
InsertIntoLeaf函数用于当B+树不为空时，在正确的叶子页中插入新键值对。在该函数中，首先调用FindLeafPage函数和查找到子页的Lookup函数判断键值对是否已存在，如果已存在返回false，否则调用Insert函数插入新键值对。插入后，如果子页的Size大于子页的MaxSize，则需要调用Split函数分裂子页，并调用InsertIntoParent函数在父页中插入新分裂子页的索引。

如图2.6所示，当要在B+树中插入键82时，首先调用FindLeafPage查找得到应该在叶子页【72，86】中插入键82，调用该叶子页的Insert函数插入键82即可。完成插入后得到的新B+树如图2.7所示。



**图2.7 插入键82后的B+树**

当继续在B+树中插入键106时，首先调用FindLeafPage查找得到应该在叶子页【97，100，113】中插入键106，调用该叶子页的Insert函数插入键106。插入后，该叶子页【97，100，106，113】的结点数量达到最大值，需要进行分裂操作。调用Split函数分裂后得到旧叶子页【97，100】和新叶子页【106，113】，然后调用InsertIntoParent函数插入对新叶子页【106，113】的索引。完成插入后得到的新B+树如图2.8所示。



**图2.8 插入键106后的B+树**

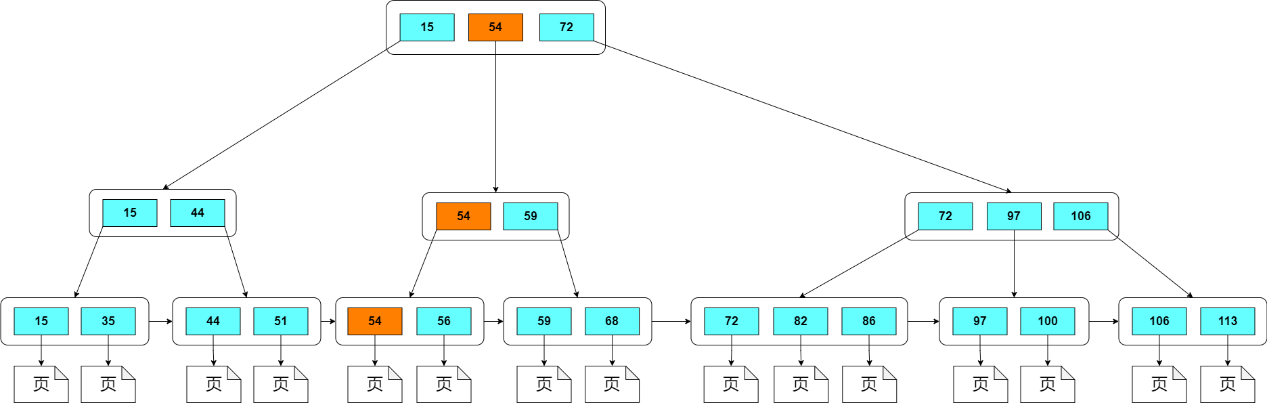
1. Split

Split函数用于分裂页并返回新生成的页。在该函数中，首先调用BufferPoolManager生成新页，然后初始化新页，并移动node的一半结点到新页。

1. InsertIntoParent

InsertIntoParent函数用于向新页的父页中插入正确的索引。在该函数中，首先判断分裂过的旧页是否为根页，如果是，则应该构造新的根页，并调用PopulateNewRoot添加旧页和新分裂页的索引；如果不是根页，则调用父页的InsertNodeAfter方法插入对新分裂页的索引，插入之后，需要判断父页是否超过max\_size，如果超过则需要再次调用Split和InsertIntoParent进行分裂操作。

当继续在B+树中插入键54时，首先调用FindLeafPage查找得到应该在叶子页【44，51，56】中插入键54，调用该叶子页的Insert函数插入键54。插入后，该叶子页【44，51，54，56】的结点数量达到最大值，需要进行分裂操作。调用Split函数分裂后得到旧叶子页【44，51】和新叶子页【54，56】，然后调用InsertIntoParent函数插入对新叶子页【54，56】的索引。完成插入后内部页【15，44，59】变为【15，44，54，59】，结点数量达到最大值，继续进行分裂操作，完成插入后得到的新B+树如图2.9所示。



**图2.9 插入键54后的B+树**

### B+树索引：Remove

1. 基本概述

一颗完整的B+树由内部页和叶子页组成，随着删除记录，页不断合并减少。

1. BPlusTree类

BPlusTree类需要实现的函数见表2.6。

**表2.6 BPlusTree类的函数（Remove相关）**

| 函数 | 说明 |
| --- | --- |
| IsEmpty | B+树是否为空 |
| Remove | 删除键key对应的记录 |
| CoalesceOrRedistribute | 对结点个数小于min\_size的页进行处理 |
| FindSibling | 寻找node页的兄弟页sibling |
| Coalesce | 将node中的结点全部合并到neighbor\_node中 |
| Redistribute | 将node中的元素移动到neighbor\_node中 |
| AdjustRoot | 对元素个数小于Min\_size的根页进行更新 |
| UpdateRootPageId | 当B+树根页变化时，在header\_page中对根页ID进行更新 |

1. FindSibling

FindSibling函数用于寻找某页的兄弟页。在函数中，首先查找该页在父页中的index，如果该页为父页的首个子页，则返回其后继页，否则返回其前驱页。

1. Remove

Remove函数用于删除键key对应的记录。在函数中，首先查找键key所在的叶子页，然后调用该叶子页的RemoveAndDeleteRecord函数删除键key对应的键值对，删除后判断该叶子页的size是否小于min\_size，如果小于则调用函数CoalesceOrRedistribute处理。

1. CoalesceOrRedistribute

CoalesceOrRedistribute函数用于对结点数量小于min\_size的页进行处理。如果该页为根页，则调用AdjustRoot更新；否则，查找该页的兄弟页，如果当前页和兄弟页的总结点数量小于max\_size，则进行合并，此时如果查找到的兄弟页为该页的后继页，则将兄弟页合并到当前页，如果查找到的兄弟页为该页的前驱页，则将该页合并到兄弟页。如果无法进行合并，调用Redistribute函数进行重分配。

1. Coalesce

Coalesce函数用于将node中的元素全部合并到neighbor\_node中。在该函数中，调用待合并页的MoveAllTo方法，将其全部结点移动至幸存页neighbor\_node中。在父页中删除待合并页的索引，并调用BufferPoolManager的DeletePage方法删除页。删除后，如果父页的size小于min\_size或者父页仅剩余一个有效结点时，需要继续调用CoalesceOrRedistribute函数进行处理。

1. Redistribute

Redistribute函数用于将将node中的元素移动到neighbor\_node中，根据index值确定移动首元素还是尾元素。

1. AdjustRoot

当根页为叶子页时，直接删除该页并UpdateRootPageId；如果根页仅剩余一个节点，即只剩余一个叶子页，直接使用叶子页替代原根页。

# 课程总结

本次课程实践的总体任务是通过关卡任务提升数据库编程能力，掌握数据库索引内核，掌握使用JAVA进行数据库开发。在本实践中，从数据库的创建入手，我顺利完成了表的完整性约束的创建和修改、数据查询、数据插入、数据修改、数据删除、视图、触发器、存储过程与事务、用户自定义函数、安全性控制、并发控制与事务的隔离级别、数据库的备份与日志、数据库的设计与实现、B+树索引等实践提供的所有关卡。

在前几部分内容中，我重点掌握了数据库的创建、数据表的增删改查等操作，这部分属于数据库的基本操作，关卡难易结合，很好地考察了我的数据库基本编程能力。

接下来，我主要掌握了数据库中的其他组件——视图、触发器、事务、存储过程、自定义函数、安全控制、并发控制、备份与恢复，极大地提高了我对数据库相关知识的掌握程度。

在实践的最后阶段，我将精力放在了B+树上，学习掌握了B+树的结构、原理、增删查操作。

在完成实践的过程中，我经历过长时间无法通关数据查询的部分关卡、B+树索引关卡难度较高等挫折，最终在不懈努力下独立完成了这些关卡，受益匪浅，极大地提高了我的数据库编程能力，并且很好地掌握了B+树这一数据结构。

我们是OpenGauss数据库实验的试点班级，存在许多任务要求为MySQL版与实际OpenGauss语法出入较大的关卡 ，希望老师能够修订测试各关关卡，保证各关关卡任务要求正确严谨。

最后，衷心感谢课程组老师们为OpenGauss数据库实践做出的努力！