基于 openGauss 数据库查询优化实践教学案例研究

徐佳[1] 李学俊[2,3**] 汪粼波[2] 李春春[1] 丁转莲[1]

1 安徽大学互联网学院，合肥 230601

2 安徽大学计算机科学与技术学院，合肥 230601

3 合肥师范学院计算机与人工智能学院，合肥 230601

摘 要 针对数据库原理课程中查询优化实践薄弱、教学内容与实际应用脱节的问题，分析当前课程中案例抽象性强、工具环境局限等不足，阐述将国产数据库 openGauss 与智慧物流无人配送场景融合的教学设计思路，介绍涵盖索引优化、连接查询、嵌套查询等内容的实践教学案例，说明该方法有效提升了学生的查询性能分析与优化能力，具有良好的教学推广价值和实践效果。

关键字 数据库教学，查询优化，openGauss，实践教学案例

A Case Study on Practice-Oriented Teaching of Query Optimization Based on the openGauss Database *

Jia Xu[1], Xuejun Li[2,3], Linbo Wang[2], Chunchun Li[1], Zhuanlian Ding[1]

1 School of Internet, Anhui University, Hefei 230601, China;
2 School of Computer Science and Technology, Anhui University, Hefei 230601, China
3 School of Computer and Artificial Intelligence, Hefei Normal University, Hefei, 230601, China

Abstract—This paper addresses the insufficient emphasis on query optimization practice and disconnect between teaching content and real-world applications in database principles courses. It analyzes existing issues such as the high level of abstraction in teaching cases and the limitations of tool environments. To tackle these challenges, a teaching design is proposed that integrates the domestic openGauss database with a smart logistics unmanned delivery scenario. Practical teaching cases are developed, covering key topics including index optimization, join queries, and nested queries. The proposed approach effectively enhances students' abilities in query performance analysis and optimization. Experimental teaching results demonstrate that this method not only improves practical competence but also exhibits strong applicability and scalability in database education.

Keywords—database education, query optimization, openGauss, practice-oriented teaching case

1 引 言

数据库原理作为计算机类专业核心课程之一，是支撑后续软件工程、面向对象程序设计等课程的重要基础[1, 2]。该课程主要涵盖数据库的基本概念、关系模型、关系代数与关系演算、SQL 查询语言、数据库设计理论、事务管理及并发控制、数据恢复机制以及查询优化等内容[3]。尤其是在当今大数据、人工智能及云计算迅猛发展的背景下，数据库系统不仅是信息系统运行的核心，更是高效数据处理与管理的关键[4, 5]。因此，数据库原理课程的教学质量直接影响学生在数据处理与系统开发方面的综合能力[1, 2, 6]。

查询优化作为数据库系统性能提升的核心机制，其理论复杂性与系统实现性较强，是连接理论与实践的重要桥梁[7]。掌握查询优化原理不仅有助于学生深入理解数据库内部机制，更能提升其在实际系统开发中编写高性能 SQL 语句、优化数据库结构与配置能力。

目前，大多数高校在数据库原理课程的开设过程中，仍以讲授基本概念、SQL 语法和 E-R 建模为主，课程内容偏重理论而实践不足，尤其在查询优化模块的教学方面存在以下几个突出问题：首先，教学内容抽象难懂，查询优化涉及代价估算、执行计划生成、索

*基金资助：本文得到安徽省高等学校省级质量工程项目，双轨并进，产教融合”面向应用能力和系统能力培养的数据库实验教学体系研究(2024syyj011)；安徽省教学研究项目（重大），教育科技人才一体化背景下学科结构优化路径研究(2024jyxm0077)；安徽省新时代育人质量工程研究生教育教学改革研究项目，新工科背景下多元融合的机器学习研究生课程教学改革探索（2024jyjxggyjY083）项目支持。

**通讯作者：李学俊 xjli@ahu.edu.cn

引选择等多个系统内部机制，对初学者而言理解难度较大，往往流于公式化讲解，难以形成系统认知。其次，缺乏可操作的实验平台,传统实验环境多基于MySQL、SQLite 等轻量级数据库，虽便于部署，但在暴露执行计划、优化器决策逻辑、索引选择策略等方面支持有限，无法满足深入教学需求。再次，案例资源匮乏，实践教学薄弱,受限于教学时间与平台支持，教师往往难以组织有深度的查询优化实践项目，学生仅停留在简单 SQL 语句层面,缺乏真实场景的性能优化。

为此，本文提出以华为开源企业级数据库 openGauss 为支撑，设计并实施一套系统化的查询优化教学案例。通过引入 openGauss Server，结合其完善的执行计划展示与优化器调试机制，构建“理论—实践—反馈”闭环教学模式，从而提升学生对查询优化机制的理解与实战能力，推动数据库原理课程的高质量发展。

2 案例概况

为提升本科生对数据库查询优化技术的理解与应用能力，本教学案例设计面向已掌握数据库基础理论与 SQL 编程能力的全体本科生，依托国产数据库 openGauss 构建实践平台，结合“智慧物流无人配送系统”应用场景，系统性开展数据库查询优化实验教学。案例聚焦查询性能提升的关键知识点，强调理论结合实践、系统知识融合能力的双重提升。

2．1 教学对象与目标

本教学案例面向已完成数据库原理相关理论学习，并具备基本 SQL 编程能力的本科学生群体，旨在通过一系列有针对性的实验任务，实现以下教学目标：

(1) 理解数据库查询与优化核心概念：包括连接查询、嵌套查询、索引、数据库模式设计等概念的作用及其实现原理；

(2) 掌握索引优化技术：能够合理设计与创建索引，提高查询效率；掌握索引覆盖、索引合并等关键优化策略；

(3) 提升系统性能优化能力：通过调整查询语句结构与数据库物理设计，提高整体系统响应速度与数据处理能力；

(4) 系统性掌握查询优化手段：熟练运用嵌套查询优化、连接查询优化及执行计划分析，识别并解决性能瓶颈；

(5) 具备数据库结构调整能力：掌握通过优化数据库关系模式以提高系统性能的方法。

2．2 教学方案特色与创新点

本教学案例在设计与实施过程中，具备以下三个显著特色与创新点：

(1) 方案的系统性：案例内容覆盖索引优化、模式设计优化、连接查询优化与嵌套查询优化等核心知识点，构建了系统化的查询优化知识结构，突出“结构—语句—计划”三个层面的协同优化理念。

(2) 场景的新颖性：教学案例以“智慧物流无人配送系统”为真实业务背景，引入车辆、订单、路线、调度等核心业务实体，提升学生问题建模与场景感知能力,实现从业务理解到查询优化策略应用能力转化。

(3) 工具的国产化：全面采用国产技术栈进行教学演示，包括部署于华为远程云平台的 openEuler 20.03 LTS 操作系统、openGauss 1.1.0 企业级数据库系统，以及 Data Studio 图形化客户端工具，强化学生对国产数据库生态的认知与适应能力。

3 案例设计

3．1 系统需求分析与功能结构设计

功能描述：随着电子商务的快速发展和人们对便利性的需求增加，无人配送系统成为了一种创新的解决方案[8, 9]。这种系统利用先进的技术，如无人机、自动驾驶车辆等，实现了从订单生成到物品配送的全自动化流程，极大地提高了配送效率和用户体验[9]。一种典型的无人配送系统所包含的大致功能如下：

(1) 订单管理：允许消费者通过系统下单购买商品，包括浏览商品、添加到购物车、提交订单等功能。同时，系统具有管理订单、查看订单历史记录功能。

(2) 消费者管理：管理消费者的个人信息，包括注册、登录、修改个人信息、查看订单状态等功能。

(3) 配送信息管理：管理配送任务的信息，包括配送路线规划、配送时间预估等功能。

(4) 物品信息管理：管理系统中所有物品的信息，包括物品名称、重量、尺寸等。

(5) 配送设备管理：管理系统中的配送设备，如无人机、无人车等。

3．2 数据库概念结构设计

根据系统需求分析与功能结构设计，一个无人配送系统包含订单表、消费者表、配送信息表、物品信息表和配送设备表。在配送过程中，多个配送任务可能需要多个配送设备协作完成，因此配送信息表与配送设备表之间是多对多的关系[10, 11]。每个订单在配送过程中可能有多个配送信息，例如订单确认、配送中、已送达等，因此订单表与配送信息表之间是一对多的关系。消费者通过系统下单购买商品，一个消费者可

以创建多个订单，因此消费者表与订单表是一对多的关系[12, 13]。此外，每个配送信息可能涉及多个物品，例如一个订单中有多个商品需要配送，所以配送信息表与物品信息表是一对多的关系。

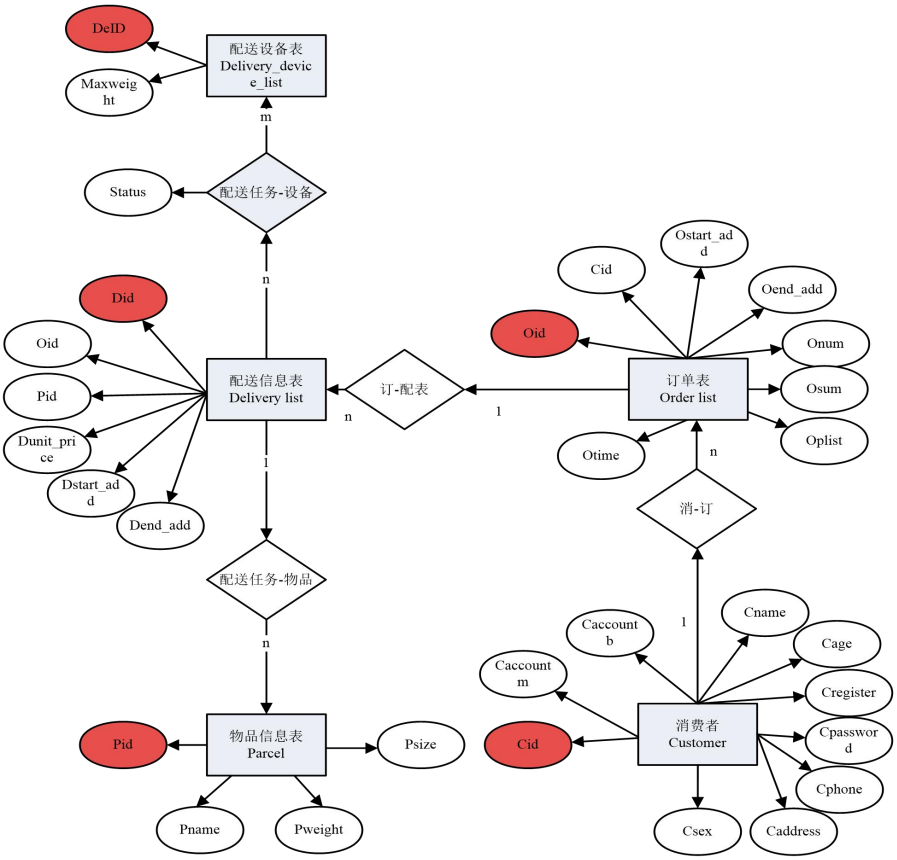


图 1 系统数据库 E-R 图

3．3 数据库逻辑结构设计

本系统基于 E-R 图建模结果，按照概念模型向关系模型的转换规则，完成了数据库逻辑结构设计。在逻辑结构设计中，遵循关系数据库规范化理论，通过分析各关系模式中的函数依赖，判断其满足的范式级别，并在必要时进行模式分解，以消除数据冗余与操作异常。以下为本系统的主要关系模式设计结果：

① 消费者（customer）

customer(cid, caccountm, cname, cage, cregister, cpassword, cphone, caddress, csex, caccountb)

函数依赖：cid → (caccountm, cname, cage, cregister, cpassword, cphone, caddress, csex, caccountb)

分析：无部分函数依赖和传递函数依赖，满足 BCNF。

② 订单（orderlist）

orderlist(oid, cid, ostart_add, oend_add, onum, osum, otime, pnamelist)

函数依赖：oid → (cid, ostart_add, oend_add, onum, osum, otime, pnamelist)

分析：满足 BCNF。pnamelist 为冗余字段，用于提高查询效率，冗余可控。

③ 包裹（parcel）

parcel(pid, pname, pweight, psize)

函数依赖：pid → (pname, pweight, psize)

分析：满足 BCNF。

④ 交通设备（vehicle）

vehicle(vid, vmax_load, vtype)

函数依赖：vid → (vmax_load, vtype)

分析：满足 BCNF。

⑤ 配送信息（deliverylist）

deliverylist(did, oid, pid, dunit_price, dstart_add, dend_add)

函数依赖：did → (oid, pid, dunit_price, dstart_add, dend_add)

分析：满足 BCNF。

⑥ 配送-设备关系（delivery_vehicle）

delivery_vehicle(did, vid, status)

函数依赖：(did, vid) → status

分析：主码为组合码(did, vid)，满足 BCNF。

综上分析，上述六个关系模式均已规范化至 BCNF，避免了非主属性对主码的部分依赖和传递依赖，实现了较好的信息分离和逻辑一致性。在后续的数据操作中，可以有效避免插入异常、删除异常、更新异常和冗余问题。

4 案例实验与分析

4.1 索引查询优化

(1) 设计思路

首先，通过执行 SQL 文件或者循环语句的方式向消费者表(customer)中插入 100 万条数据；接着，通过查询表中数据，比较当字段中加入索引和未加入索引时查询效率的变化，比较为不同字段加入索引时查询效率的变化，比较使用 LIKE 关键字在查询时通配符对索引的影响，比较覆盖索引对查询效率的影响，通过 EXPLAIN 关键字查看并理解组合索引的运行方式，最左前缀原则以及索引失效的情况。

(2) 实验内容与效果分析

实验 1：请查询 caccountm 为 1179615 的消费者信息，比较加入索引和未加入索引时查询效率的区别。

执行如下 SQL 语句：

```
SELECT * FROM customer WHERE Caccountm='179615';
EXPLAIN SELECT * FROM customer WHERE Caccountm='179615';
CREATE INDEX ON customer(caccountm);
SELECT * FROM customer WHERE caccountm='179615';
EXPLAIN SELECT * FROM customer WHERE caccountm='179615';
DROP INDEX customer_caccountm_idx ;
```

执行第 1 行 SQL 语句可以获得结果，如图 1 所示，执行时间为 524ms。

[2024-04-11 10:21:12.491 CST] : [INFO] 操作影响的记录行数 : 1
[2024-04-11 10:21:12.491 CST] : [INFO] 执行时间 : 524 ms
[2024-04-11 10:21:12.491 CST] : [INFO] 执行成功...

	cid	caccount1	cname	cage	Cregister	cpassword	Cphone
1	270778	179615	孙瑞	41	2015-10-03 07:37:54	2M2tXhnqNV1ZVVbIQ^MI	15967328951

图 2 第 1 行 SQL 语句查询结果

执行第 2 行 SQL 语句可以得出，如图 2 所示，该查询是一个顺序扫描全表的过程：

	QUERY PLAN
1	Seq Scan on customer (cost=0.00..33493.00 rows=1 width
2	Filter: ((caccount1)::text = '179615'::text)

图 3 第 2 行 SQL 语句查询结果

在加入索引后，执行第 3-4 行可以获得结果，如图 4 所示，操作时间为 149ms。

[2024-04-10 14:38:05.441 CST] : [INFO] 操作影响的记录行数 : 1
[2024-04-10 14:38:05.441 CST] : [INFO] 执行时间 : 149 ms
[2024-04-10 14:38:05.441 CST] : [INFO] 执行成功...

图 4 第 3-4 行 SQL 语句查询结果

执行第 5 行查看查询执行计划，如图 5 所示。

	QUERY PLAN
1	[Bypass]
2	Index Scan using customer_caccount1_idx on customer (cost=0.00..8.38 rows=1 width=135)
3	Index Cond: ((caccount1)::text = '179615'::text)

图 5 第 5 行 SQL 语句查询结果

原因分析：由此可以得出，索引由于使用了 B+树的底层结构，减少了访问磁盘的 IO 次数，能够显著加快查询速度。

实验 2：请查询 csex 为女的所有消费者信息，并为性别字段加入索引，比较索引加入前后对查询效率的影响。

执行如下 SQL 语句：

```
SELECT * FROM customer WHERE csex='女';
CREATE INDEX idx_sex ON customer(csex);
EXPLAIN SELECT * FROM customer WHERE csex='女';
DROP INDEX idx_sex;
```

执行 SQL 语句第 1 行后可以得到结果，如图 6 所示，查询的执行时间为 490ms。

[2024-04-10 14:54:34.588 CST] : [INFO] 操作影响的记录行数 : 1,000
[2024-04-10 14:54:34.588 CST] : [INFO] 执行时间 : 490 ms
[2024-04-10 14:54:34.588 CST] : [INFO] 执行成功...

	cid	caccount1	cname	cage	Cregister	cpassword	C
1	80012	100765	陶丽	63	2019-01-21 05:45:49	xp3+kikK+Y1fQoJ7K@!V	15172
2	870006	100767	万丹丹	74	2012-04-26 07:39:31	_Pq9x06dq+qlQqnlx%p_	18519
3	190010	100769	高峰	47	2008-05-13 15:04:20	3x7V#HhdVdXE9Ika_QUs	15382
4	880006	100770	陈秀芳	19	2001-03-10 05:12:31	*A@+(b&q%Bi2D!N^#854	13622
5	890006	100772	张秀荣	63	2019-12-17 20:23:09	@9d7eYEUpfg%Dyxb+1r4	18054
6	420008	100773	苗瑞	75	2013-12-28 22:25:20	e8^BT8Y6zT43TWpwr$m8	15565
7	620007	100774	蔡婷	24	2008-07-09 09:15:05	BDYJUlXS*6vf7ewn+^ql	14500
8	900006	100775	张晶	26	2023-06-20 06:24:00	(+Vs62)K42mx8qdKVUOH	14528
9	280009	100776	张洋	75	2001-01-29 04:55:25	gSN#$o3)dPguehq0s5ls	18192
10	10019	100777	吴峰	57	2021-04-16 13:36:47	k%SE7iaUNz9f#d@Ebn5L	18817
11	630007	100782	尤慧	14	2000-05-28 03:56:57	lydiSwB4I1FpsRov_IRo	13566
12	430008	100784	[illegible]	63	2008-02-04 23:53:09	[illegible]	1572

图 6 第 1 行 SQL 语句查询结果

执行 SQL 语句第 2-3 行后可以得到结果，如图 7 所示，查询时间为 358ms。

[2024-04-10 14:57:23.801 CST] : [INFO] 操作影响的记录行数 : 1,000
[2024-04-10 14:57:23.801 CST] : [INFO] 执行时间 : 358 ms
[2024-04-10 14:57:23.801 CST] : [INFO] 执行成功...

图 7 第 2-3 行 SQL 语句查询结果

原因分析：从结果中可以看出加索引与不加索引对查询效率的提升并不大，这是因为性别字段只有两

个选择，选择性很低。如果一个字段的选择性很低，即该字段的不同值的数量与表中总行数相差无几，那么这样的字段不适合建立索引，因为索引的优势在于快速定位到少量数据。

4．2 连接查询优化

连接查询是数据库中常用的一种查询技术，用于联合多个表中的数据，以获取更丰富的查询结果。连接查询通过在多个表之间建立关联关系，将它们的记录进行匹配和组合，从而实现数据的关联和合并。优化连接查询需要综合考虑查询的复杂性、数据的分布情况和数据库管理系统的特性。

(1) 设计思路

首先，通过 SQL 文件或者循环语句的方式向物品表(parcel)和配送信息表(deliverylist)中插入 10 万条数据；接着，通过查询表中数据，比较连接查询条件顺序的变化，比较子查询结构的变化对查询效率的影响。

(2) 实验内容与效果分析

实验 1：请查询“订单号(oid)为 28005 的消费者性别”，比较连接查询条件的顺序改变对查询效率的影响

```
//连接顺序中customer放orderlist前面
SELECT csex
FROM customer,orderlist
WHERE customer.cid = orderlist.cid AND orderlist.oid=28005;
//连接顺序中orderlist放customer前面
SELECT csex
FROM orderlist,customer
WHERE customer.cid = orderlist.cid AND orderlist.oid=28005;
```

执行第 1-3 行，可以得到如下结果，如图 8 所示，执行时间为 683ms。

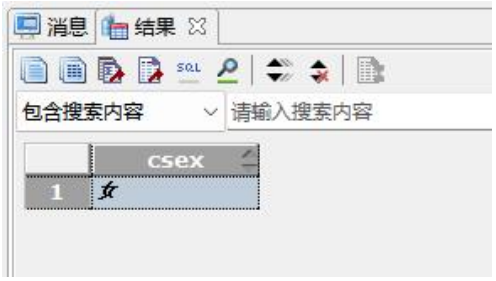


图 8 第 1-3 行 SQL 语句查询结果

执行第 4-6 行，可以得到如下结果，如图 9 所示，执行时间为 929ms。

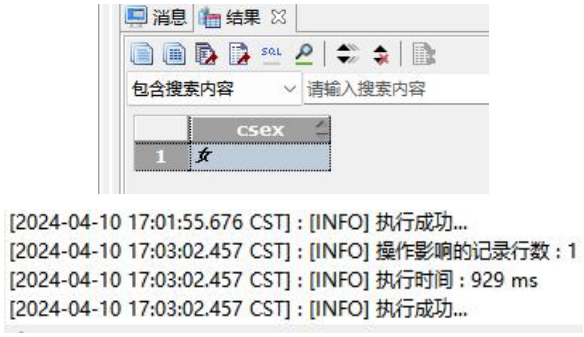


图 9 第 4-6 行 SQL 语句查询结果

原因分析：如果查询涉及多个表的连接，连接的顺序会影响查询的性能。通常情况下，将尺寸较小的表放在连接的前面，可以减少连接操作时的扫描范围，提高查询效率。对于多表连接，先连接数据量小的表可以提高性能。

实验 2：请查询“所有订单对应的消费者名称和订单总金额”，比较子查询结构对查询效率的影响。

```
//使用连接查询：
SELECT c.cname, SUM(o.osum)
FROM customer c
JOIN orderlist o ON c.cid = o.cid
GROUP BY c.cname;
//使用子查询优化连接查询：
SELECT c.cname, total_amount
FROM customer c
JOIN (
    SELECT cid, SUM(osum) AS total_amount
    FROM orderlist
    GROUP BY cid
    ) o ON c.cid = o.cid;
```

执行第 1-4 行，可以得到如下结果，如图 10 所示，运行时间为 1s。

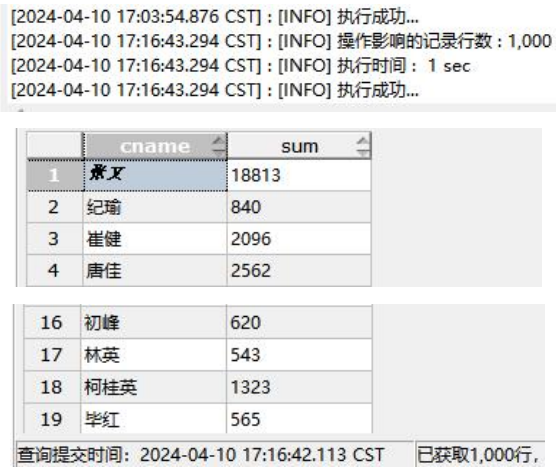
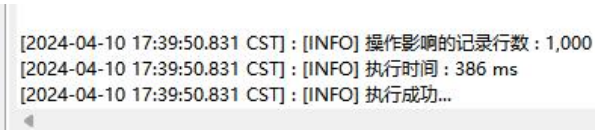


图 10 第 1-4 行 SQL 语句查询结果

执行第 5-11 行，可以得到如下结果，如图 11 所示，执行时间为 386ms。



	cname	total_amount
1	李春梅	753
2	秦辉	790
3	连勇	696
4	王静	2602
5	赵玉英	969
15	梁玉兰	939
16	周玉珍	974
17	谢秀兰	110
18	边玲	1816

查询提交时间：2024-04-10 17:39:50.430 CS

图 11 第 5-11 行 SQL 语句查询结果

原因分析：第一个查询将使用连接条件将 customer 表和 orderlist 表进行关联，并根据消费者名称进行分组求和。在优化后的第二个查询中，将连接字段上的索引创建完善。同时，将计算每个消费者的订单总金额的逻辑放入子查询中，并将其命名为 o。然后，将 customer 表和子查询结果进行连接，获取消费者姓名和订单总金额作为结果。在连接查询中，需要将两个表的数据进行传输和匹配。而通过使用子查询优化的方式，内部的子查询将先执行，并将结果作为临时表可以提高查询的执行效率和性能，只计算必要的数据，减少不必要的开销。

4．3 嵌套查询优化

嵌套查询是指在一个外层查询中包含有另一个内层查询。其中外层查询称为主查询，内层查询称为子查询。SQL 允许多层嵌套，由内而外地进行分析，子查询的结果作为主查询的查询条件。

(1) 设计思路

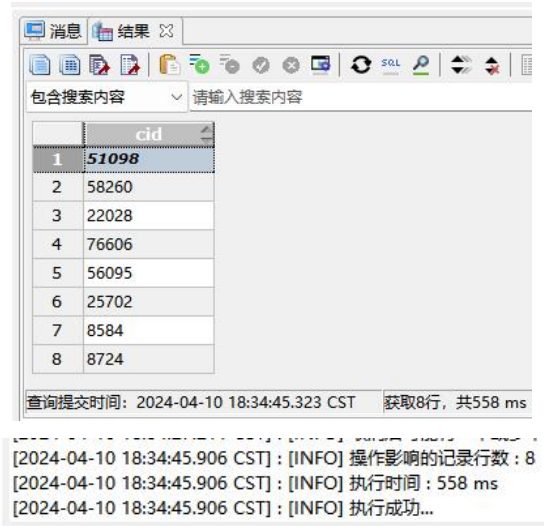
首先，通过 SQL 文件或者循环语句的方式分别向消费者表(customer)、订单表(orderlist)和订单配送表(deliverylist)中插入 10 万条数据；接着，通过查询表中数据，比较当字段中有嵌套查询和无嵌套查询时查询效率的变化，比较为嵌套查询条件的顺序的变化，比较子查询的结构对查询效率的影响，比较使用 EXISTS 或 NOT EXISTS 对查询效率的影响。

(2) 实验内容与效果分析

实验 1：请查询“求购买了货物 ID=3072 的消费者 ID”，比较使用连接查询查询和嵌套查询时查询效率的变化。

```
//使用嵌套查询SQL语句：
SELECT cid
FROM orderlist
WHERE oid IN (
                SELECT oid
                FROM deliverylist WHERE pid = 3072
              );
//使用连接查询SQL语句：
SELECT cid
FROM orderlist,deliverylist
WHERE orderlist.oid = deliverylist.oid AND deliverylist.pid=3072;
```

执行第 1-6 行 SQL 语句，得到结果，如图 12 所示，执行时间为 558ms：



消息 结果

包含搜索内容 请输入搜索内容

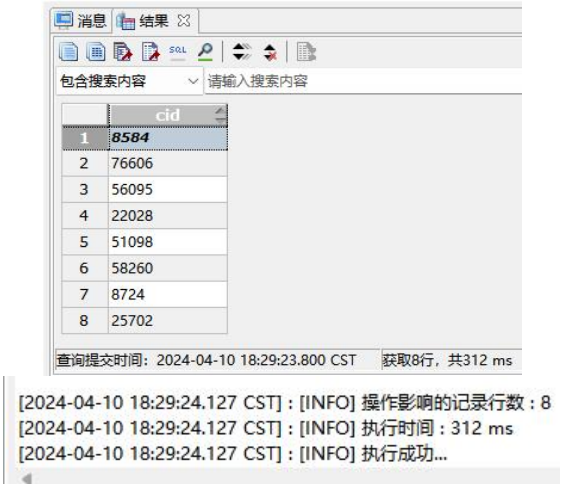
	cid
1	51098
2	58260
3	22028
4	76606
5	56095
6	25702
7	8584
8	8724

查询提交时间：2024-04-10 18:34:45.323 CST 获取8行，共558 ms

[2024-04-10 18:34:45.906 CST] : [INFO] 操作影响的记录行数 : 8
[2024-04-10 18:34:45.906 CST] : [INFO] 执行时间 : 558 ms
[2024-04-10 18:34:45.906 CST] : [INFO] 执行成功...

图 12 第 1-6 行 SQL 语句查询结果

执行第 7-9 行 SQL 语句，得到结果，如图 13 所示，执行时间为 312ms。



消息 结果

包含搜索内容 请输入搜索内容

	cid
1	8584
2	76606
3	56095
4	22028
5	51098
6	58260
7	8724
8	25702

查询提交时间：2024-04-10 18:29:23.800 CST 获取8行，共312 ms

[2024-04-10 18:29:24.127 CST] : [INFO] 操作影响的记录行数 : 8
[2024-04-10 18:29:24.127 CST] : [INFO] 执行时间 : 312 ms
[2024-04-10 18:29:24.127 CST] : [INFO] 执行成功...

图 13 第 7-9 行 SQL 语句查询结果

原因分析：对于 orderlist 表中的每个元组来说，都要检索一次 deliverylist 表，效率非常低下。可以转换成连接操作，因为子查询会多次遍历所有的数据，而连接查询只会遍历一次。数据量大的情况下，优选使用连接查询。需要注意的是，连接查询和嵌套查询得到的查询结果表格顺序不一定相同。查询结果的顺序取决于数据库引擎的具体实现。如果需要保证查询结果的顺序一致，可以使用 ORDER BY 子句指定特定的排序规则来确保结果的顺序一致性。

实验 2：请查询“购买了货物名称为薯片并且运输单价小于 2 的订单 ID”，比较为嵌套查询条件的顺序的变化。

```
//条件顺序为先筛选特定商品再比较运输单价
SELECT oid
```

```
FROM deliverylist
WHERE pid IN (
                    SELECT pid
                    FROM parcel
                    WHERE pname = '薯片'
                  )
AND dunit_price <2;
//条件顺序为先比较运输单价再筛选特定商品
SELECT oid
FROM deliverylist
WHERE dunit_price < 2
AND pid IN (
                SELECT pid
                FROM parcel
                WHERE pname = '薯片'
              );
```

执行第 1-8 行 SQL 语句，得到结果，如图 14 所示，运行时间为 473ms。

图 14 第 1-8 行 SQL 语句查询结果

执行第 9-16 行 SQL 语句，得到结果，如图 15 所示，执行时间为 243ms。

图 15 第 9-16 行 SQL 语句查询结果

原因分析：在第一个查询中，先进行了子查询来获取特定商品的 ID，然后将该结果作为 IN 条件进行筛选。接着，在外部查询中再根据订单金额进行筛选。这样的顺序可能会导致先筛选出较多的商品，然后再进行金额比较，效率相对较低。而在第二个查询中，先进行了外部查询，通过订单金额的比较筛选出满足条件的订单，然后再进行子查询来获取特定商品的 ID。这样的顺序可以先筛选出较少的订单，再进行商品的匹配，可能会提高查询效率。因此，嵌套查询条件的顺序的变化可以导致不同的查询效率。选择合适的条件顺序，根据实际情况和数据的分布情况，可以提高查询性能。

5 案例应用

5．1 案例使用方法与课时安排

本案例整体用于《数据库原理》课程的实践教学模块，要求学生在参与实践前已具备数据库建模与 SQL 基本能力。案例共设计四个实验任务，建议安排在 3 课时内完成，具体内容与知识要点如下：

(1) 索引查询优化（1 课时）

理解索引的类型、结构与代价模型；

掌握创建、删除与修改索引的方法；

能够分析不同索引策略对查询性能的影响，并在执行计划中识别索引使用情况。

(2) 连接查询优化（1 课时）

掌握内连接、外连接、自连接等不同类型的连接操作；

理解连接顺序、连接算法（如嵌套循环连接、哈希连接等）对性能的影响；

能够根据执行计划对连接查询进行重写与优化。

(3) 嵌套查询优化（1 课时）

掌握相关与非相关子查询的差异；

理解嵌套查询与 JOIN 重写之间的性能差异；

能够运用优化技巧提升复杂查询的执行效率。

5．2 实施与推广

数据库实践平台是数据库实践教学的重要基础。教学团队基于学习通平台构建了实践课程网站和智慧教学资源，并构建了涵盖从数据库实践应用能力到系统能力培养的教学资源库。依托头歌平台，团队搭建了支持学生在线完成数据库设计、开发与测试任务的实践平台。同时与华为合作推动国产化实践教学云平台的建设，相关课程获批教育部产学协作协同育人项目，为实践平台的综合化发展提供了重要支撑。此外，教学团队基于现有的教学实践，出版了 2 部数据库实验教材，为实践平台建设提供了重要的实践资源保障。这些努力不仅提升了实践教学的效率，也为学生提供了更加灵活和多样化的学习环境。

该实践案例自设计以来，已在近三年的数据库原理实验教学中持续使用，获得良好教学反馈，获教育部数据库课程虚拟教研室一等奖（全国年度唯一），系统部署于华为远程云服务器，运行环境为 openEuler 20.03 LTS 操作系统和 openGauss 1.1.0 数据库系统，学生通过 Data Studio 客户端与数据库交互，实现实验操作的便捷性与可视化。同时，案例资源已面向公众开源发布，便于其他高校教学共享与持续改进[1]。

6 结 语

本文围绕数据库原理课程中查询优化教学的关键问题，构建了一个基于 openGauss 平台的实践教学案例，系统覆盖了索引优化、模式设计、连接与嵌套查询等核心知识点。通过融合智慧物流无人配送的应用场景与国产软硬件平台，案例不仅增强了教学的现实关联性，也体现了国产化技术的教学适配性与可操作性。教学实践表明，该案例显著提升了学生在查询性能分析与优化方面的综合能力，取得了良好的教学成效。

未来的工作中，实验室将进一步拓展案例的覆盖范围，例如引入分布式查询优化、执行计划可视化分析等高级内容，同时结合学生反馈持续优化教学任务设计，推动数据库教学从知识讲授走向能力培养，服务于新工科背景下高质量人才的培养目标。

参 考 文 献

[1] 施珮, 张永宏, 王泉, 等. 面向新工科的数据库原理课程教学改革探讨——以车联网应用实践教学为例 [J]. 高教学刊, 2025, 11(11): 138-141.

[2] 刘斌, 彭煜玮. 提升数据库教学与实践质量的校企合作路径探索 [J]. 计算机教育, 2025, 364(04): 186-190.

[3] 郑力宁, 李嘉哲, 李镇宇. 应用型本科院校《数据库原理及应用》课程教学改革探索与实践[C]. 2025 年高等教育发展论坛. 中国河南郑州, 2025: 40-41.

[4] 乔少杰, 李洲, 韩楠, 等. 人工智能赋能关系型数据库优化技术：现状与展望 [J]. 计算机学报, 2025, 48(7): 1639-1669.

[5] 曹金鑫, 鞠小林, 陈翔. 协同过滤推荐算法在数据库原理及应用课程教学中的应用 [J]. 计算机教育, 2025, 364(04): 197-201.

[6] 张申勇, 蔡培茂, 谭忠兵, 等. 基于费曼技巧构建知识输出型教学——以数据库原理与应用课程为例 [J]. 计算机教育, 2025, 361(01): 179-183.

[7] 刘喜平, 舒晴, 何佳壕, 等. 基于自然语言的数据库查询生成研究综述 [J]. 软件学报, 2022, 33(11): 4107-4136.

[8] 于彦鹏, 余墨多, 汤奇荣, 等. 面向城市应急物资配送的多无人机协同路径规划算法 [J]. 控制与决策, 2025, 40(04): 1098-1106.

[9] 王飞, 杨清平. 面向多无人机物流配送的双层任务规划方法 [J]. 北京航空航天大学学报, 2026, 52(1): 1-14.

[10] 朱玉全, 周李威, 陈耿. 数据库理论教学中关联规则与函数依赖之间联系的探讨 [J]. 计算机应用研究, 2014, 31(07): 2085-2087.

[11] 郭文越, 曲海成, 崔丽群, 等. 基于 BOPPPS 教学模型的 Oracle 数据库课程思政教学探索 [J]. 计算机教育, 2025, 366(06): 88-94.

[12] 于明远, 杨良怀, 雷艳静, 等. 新质生产力赋能“数据库原理及应用”课程的教学探索 [J]. 中国信息技术教育, 2025, (10): 87-92.

[13] 罗丹, 崔晓晖, 陈志泊. AI 赋能“数据库原理与应用”课程思政教学改革与实践 [J]. 计算机技术与教育学报, 2026, 03(14): 73-77.