

**内存数据库实验方案设计**

|  |
| --- |
| Boying——博影票务系统 |

指导教师*:* **袁时金**

In-Memory Database

Tongji University

School of Software Engineering

目录

[实验一 查询优化实验 1](#_Toc54807762)

[1.1 实验目的 1](#_Toc54807763)

[1.2 实验方案 1](#_Toc54807764)

[1.2.1 订单查询 1](#_Toc54807765)

[1.2.2 演出场次查询 2](#_Toc54807766)

[1.2.3 演出票档查询 3](#_Toc54807767)

[1.2.4 演出条件查询 4](#_Toc54807768)

[实验二 索引优化实验 5](#_Toc54807769)

[2.1 实验目的 5](#_Toc54807770)

[2.2 实验方案 5](#_Toc54807771)

[2.2.1 针对日期建立索引 5](#_Toc54807772)

[2.2.2 针对类型建立索引 5](#_Toc54807773)

[2.2.3 针对日期和类型建立复合索引 6](#_Toc54807774)

[2.2.4 以上实验结果比较 6](#_Toc54807775)

[2.3 实验结果推测 6](#_Toc54807776)

[实验三 冷热数据对比实验 8](#_Toc54807777)

[3.1 实验目的 8](#_Toc54807778)

[3.2 实验方案 8](#_Toc54807779)

[实验四 批量写入实验 9](#_Toc54807780)

[4.1 实验目的 9](#_Toc54807781)

[4.2 实验方案 9](#_Toc54807782)

[4.2.1 单表写入实验 9](#_Toc54807783)

[4.2.2 多表写入实验 9](#_Toc54807784)

[实验五 数据报表实验 11](#_Toc54807785)

[5.1 实验目的 11](#_Toc54807786)

[5.2 实验方案 11](#_Toc54807787)

[5.2.1 用户年终观演总结报表 11](#_Toc54807788)

[5.2.2 售票平台的财务统计报表 11](#_Toc54807789)

[5.2.3 城市演出业务统计报表 12](#_Toc54807790)

# 查询优化实验

针对现实业务逻辑中存在的各种场景，找到对应的查询语句，并进行SQL语句级别的优化实验，找到最佳方案。

## 实验目的

在博影购票系统中，存在着大量的查询业务。这是整个系统的核心功能，如果不进行详细的优化设计，将直接影响系统的使用体验。SQL查询语句的不同写法会直接影响查询效率。因此，我们针对几个主要查询业务，进行SQL语句级别的优化实验。

## 实验方案

### 订单查询

数据表: USER, ORDER, TICKET, SHOW, SESSION, CLASS, FREQUENT, ADDRESS

优化策略: from子句用于完成对多个表的连接操作，SQL语法扫描器从右到左的顺序处理FROM子句中的表名，记录条数最少的表作为基础表应该置于from连接的最右侧。同时，对于 3 个以上的表连接查询，应将被其他引用的交叉表作为基础表。

SQL语句:

1. 优化前:

**SELECT User**.username, **Order**.order\_id, **Order**.status, **Order**.time,

Show.name,**Session**.start\_time, **Session**.end\_time, Class.name,

Frequent.name,Address.province, Address.city, Address.region,

Address.street

**FROM** Ticket, **Order**, Class, **Session**, Show, **User**, Address, Frequent

**WHERE Order**.user\_id = **User**.user\_id

**AND Order**.session\_id = **Session**.session\_id

**AND Session.show\_id = Show.show\_id**

**AND Order.order\_id = Ticket.order\_id**

**AND Ticket.class\_id = Class.class\_id**

**AND Order.frequent\_id = Frequent.frequent\_id**

**AND Order**.address\_id = Address.address\_id;

1. 优化后:

**SELECT User**.username, **Order**.order\_id, **Order**.status, **Order**.time,

Show.name,**Session**.start\_time, **Session**.end\_time, Class.name,

Frequent.name,Address.province, Address.city, Address.region,

Address.street

**FROM Order**, Ticket, Show, **Session**, Class, Address, Frequent, **User**

**WHERE Order**.user\_id = **User**.user\_id

**AND Order**.session\_id = **Session**.session\_id

**AND Session.show\_id = Show.show\_id**

**AND Order.order\_id = Ticket.order\_id**

**AND Ticket.class\_id = Class.class\_id**

**AND Order.frequent\_id = Frequent.frequent\_id**

**AND Order**.address\_id = Address.address\_id;

实验结果:

|  |  |  |
| --- | --- | --- |
| 数据库 | 优化前 | 优化后 |
| Oracle  TimesTen |  |  |

Tab. 1.1: 订单查询优化实验记录表格

### 演出场次查询

数据表: CATEGORY, SHOW, SESSION

优化策略: 演出场次查询涉及多项where条件匹配，where条件末尾的过滤条件最先被检索。条件过滤的顺序会影响查询效率，展开实验，根据实际情况确定where条件过滤顺序。

SQL语句:

1. 优化前:

**SELECT** Category.name, Show.show\_id, Show.name, Show.city, Show.address,  
**Session**.session\_id, **Session**.name, **Session**.start\_time, **Session**.end\_time

**FROM** Show, **Session**, Category  
**WHERE** Category.category\_id = Show.category\_id  
 **AND Session**.show\_id = Show.show\_id  
 **AND Session**.session\_id = 1234567;

1. 优化后:

**SELECT** Category.name, Show.show\_id, Show.name, Show.city, Show.address,  
**Session**.session\_id, **Session**.name, **Session**.start\_time, **Session**.end\_time

**FROM** Show, **Session**, Category  
**WHERE Session**.session\_id = 1234567  
 **AND Session**.show\_id = Show.show\_id  
 **AND** Category.category\_id = Show.category\_id;

实验结果:

|  |  |  |
| --- | --- | --- |
| 数据库 | 优化前 | 优化后 |
| Oracle  TimesTen |  |  |

Tab. 1.2: 演出场次查询优化实验记录表格

### 演出票档查询

数据表: CLASS, TICKET

优化策略: 演出票档查询需要进行条件检索。DECODE 函数相当于一条件语句(IF), 它将输入数值与函数中的参数列表相比较，根据输入值返回一个对应值。区别于SQL的其它函数，DECODE 函数还能识别和操作空值。使用 DECODE 函数可以避免重复扫描相同记录或重复连接相同的表

SQL语句:

1. 优化前:

**SELECT Class**.name   
**FROM Class  
WHERE Class**.Session\_id = 1234567;

1. 优化后:

**SELECT** *DECODE*("Session\_id", 1234567, **name**, **null**)  
**FROM Class**;

实验结果:

|  |  |  |
| --- | --- | --- |
| 数据库 | 优化前 | 优化后 |
| Oracle  TimesTen |  |  |

Tab. 1.3: 演出票档查询优化实验记录表格

### 演出条件查询

数据表: SHOW, CATEGORY

优化策略: 演出按分类查询涉及结果集合，in 和 exists 都可以用来写查询。如果子查询得出的结果集记录较少，主查询中的表较大且又有索引时应该用 in, 反之如果外层的主查询记录较少，子查询中的表大，又有索引时使用 exists。

SQL语句:

1. 优化前:

**SELECT** show\_id, **name**, details  
**FROM Show  
WHERE** *EXISTS* (  
 **SELECT** category\_id  
 **FROM Category  
 WHERE name** = "演唱会"   
 **AND Show**.category\_id = **Category**.category\_id  
);

1. 优化后:

**SELECT** show\_id, **name**, details  
**FROM Show  
WHERE** category\_id **IN** (  
 **SELECT** category\_id  
 **FROM Category  
 WHERE name** = "演唱会"  
);

实验结果:

|  |  |  |
| --- | --- | --- |
| 数据库 | 优化前 | 优化后 |
| Oracle  TimesTen |  |  |

Tab. 1.4: 演出条件查询优化实验记录表格

# 索引优化实验

索引可以大大提高数据库在某方面的性能，但不可避免地会在另一方面降低性能，而且索引种类很多，不同索引适用的场景也有所不同。

## 实验目的

通过针对不同字段及其组合，在Oracle和TimesTen数据库中分别建立不同类型的索引，并进行查询和插入两种操作检验性能的变化，最终找到适用于不同数据库的最合理的索引方案，达到提高系统性能的目的。

## 实验方案

### 针对日期建立索引

日期是用户进行演出场次挑选时的重要依据，用户可以指定演出的开始日期范围，之后系

统返回符合条件的演出结果。针对这点，我们计划对SESSION表的演出开始时间字段建立索引，比较性能。现设计实验记录表格如下:

|  |  |  |
| --- | --- | --- |
| 索引类型 | 查询操作 | 插入操作 |
| 无索引 |  |  |
| B 树索引 |  |  |
| 位图索引 |  |  |
| 哈希索引 |  |  |

Tab. 2.1: 针对日期建立索引的实验记录表格

注: TimesTen中使用的索引与Oracle中使用的索引有所不同，我们将针对TimesTen设计不同的实验表格。

### 针对类型建立索引

演出类型同样是用户筛选演出的重要依据。由于目前数据库中演出类型数量较少，我们预估通过类型名查询类型id将只消耗极少的时间，而通过类型id查询演出将成为查询的瓶颈所在。所以我们计划针对SHOW表的演出类型字段建立索引，比较性能。现设计实验记录表格如下:

|  |  |  |
| --- | --- | --- |
| 索引类型 | 查询操作 | 插入操作 |
| 无索引 |  |  |
| B 树索引 |  |  |
| 位图索引 |  |  |
| 哈希索引 |  |  |

Tab. 2.2: 针对类型建立索引的实验记录表格

注: TimesTen中使用的索引与Oracle中使用的索引有所不同，我们将针对TimesTen设计不同的实验表格。

### 针对日期和类型建立复合索引

由于每次用户筛选演出时，查询条件都是日期和演出类型，是固定不变的，这就符合了复合索引的使用条件，所以我们计划建立针对 SHOW 表中最早场次日期、演出类型字段的复合索引，进行查询和插入操作。

|  |  |  |
| --- | --- | --- |
| 索引类型 | 查询操作 | 插入操作 |
| 无索引 |  |  |

Tab. 2.3: 针对日期和类型建立索引的实验记录表格

注: TimesTen中没有复合索引类型，所以这个实验仅针对Oracle数据库。

### 以上实验结果比较

比较针对不同字段使用不同索引的查询结果，找到效率最高的索引方案。

## 实验结果推测

预计复合索引的总体效率和查询操作效率高于单字段索引，但是插入操作效率有所不如，这是因为复合索引针对设计多字段的查询有更好的优化，但是由于涉及字段较多，较为复杂，所以在插入数据之后重建索引时会消耗更多时间；对于单字段索引，情况较为复杂，因为 SHOW 表中包含大量数据，而不论是日期还是类型键值数目都与数据量有着巨大的数量级差距，故难以推测实验结果。

# 冷热数据对比实验

越是近期的演出越是会被频繁查询，而已经结束的演出可能只是偶尔被查询。通过冷热数据对比实验，可以选择最佳的冷热数据比，评估优化前与优化后的查询性能对比。

## 实验目的

对于不同日期的演出来说，其数据访问频次的差距是很明显的。而演出场次、票务信息数据庞大，内存无法承载。热数据因为访问频次需求大，效率要求高，所以就近计算和部署；冷数据访问频次低，效率要求慢，综上对冷热数据可以分开存储，从而达到提升系统整体性能的目的。

## 实验方案

1. 首先将近期的演出场次、票务信息数据放在TimesTen中，将已经结束的演出数据放入Oracle中，可以预想到这样会对系统有性能提升，但冷热数据分配的比例点要通过实验测试。

2. 基于给定的查询次数概率分布，调配冷热数据在Oracle和TimesTen中的比例进行对比试验。

# 批量写入实验

大规模数据批量写入效率，是考验数据库性能的重要指标。通过批量写入的压力测试实验，评估数据库处理单表写入和多表写入的能力。

## 实验目的

对于演出购票系统，主要有两类插入操作。一类是初始演出数据的导入操作，属于单表写入操作；另一类是订单创建操作，涉及多表的插入更新操作。对两类写入业务分别进行实验，测试其性能。

## 实验方案

### 单表写入实验

1. 业务情境:

系统初始需要导入大批量演出票数据，包括演出票类型编号、演出种类、演出名称、座位容量等信息，需要对演出票类表执行插入操作

1. 数据表: CLASS
2. 实验结果:

|  |  |  |  |
| --- | --- | --- | --- |
| 线程数 | 1000条 | 100000条 | 1000000条 |
| 单线程 |  |  |  |
| 多线程 |  |  |  |

Tab. 4.1: 单表写入实验记录表格

### 多表写入实验

1. 业务情境:

用户订购演出票的时候会生成新的订单，新创建订单操作会涉及订单表的写入以及演出票表的级联更新，属于多表写入。

1. 数据表: ORDER、TICKET
2. 实验结果:

|  |  |  |  |
| --- | --- | --- | --- |
| 线程数 | 1000条 | 100000条 | 1000000条 |
| 单线程 |  |  |  |
| 多线程 |  |  |  |

Tab. 4.2: 多表写入实验记录表格

# 数据报表实验

数据报表的生成涉及多表联合查询，并要配合使用聚合函数等，可以很好地反映查询性能。

## 实验目的

作为一个完善的管理系统，还应当具备数据分析与报表生成功能。报表包括用户年终观演总结报表、售票平台的财务统计报表以及城市的演出业务报表。分别用My- batis 持久化框架与 PL/SQL存储过程执行报表分析功能，对其性能进行对比分析。

## 实验方案

### 用户年终观演总结报表

1. 业务情境: 对用户过去一年观看演出次数、去过的城市、最钟情的演出类别以及消费情况等进行分析，生成统计报表。
2. 实验结果:

|  |  |
| --- | --- |
| 类型 | 时间 |
| PL/SQL |  |
| Mybatis |  |

Tab. 5.1: 用户年终观影总结报表记录表格

### 售票平台的财务统计报表

1. 业务情境: 对售票平台过去一年的售票种类、售票量、收入等进行分析，生成统计报表。
2. 实验结果:

|  |  |
| --- | --- |
| 类型 | 时间 |
| PL/SQL |  |
| Mybatis |  |

Tab. 5.2: 售票平台的财务统计报表记录表格

### 城市演出业务统计报表

1. 业务情境: 对演出持续时间、演出数量和演出所在城市的关联关系进行分析，生成统计报表。
2. 实验结果:

|  |  |
| --- | --- |
| 类型 | 时间 |
| PL/SQL |  |
| Mybatis |  |

Tab. 5.3: 城市演出业务报表记录表格

**附录A. 图表索引**

[Tab. 1.1: 订单查询优化实验记录表格 2](#_Toc54809055)

[Tab. 1.2: 演出场次查询优化实验记录表格 3](#_Toc54809056)

[Tab. 1.3: 演出票档查询优化实验记录表格 3](#_Toc54809057)

[Tab. 1.4: 演出条件查询优化实验记录表格 4](#_Toc54809058)

[Tab. 2.1: 针对日期建立索引的实验记录表格 5](#_Toc54809059)

[Tab. 2.2: 针对类型建立索引的实验记录表格 6](#_Toc54809060)

[Tab. 2.3: 针对日期和类型建立索引的实验记录表格 6](#_Toc54809061)

[Tab. 4.1: 单表写入实验记录表格 9](#_Toc54809062)

[Tab. 4.2: 多表写入实验记录表格 10](#_Toc54809063)

[Tab. 5.1: 用户年终观影总结报表记录表格 11](#_Toc54809064)

[Tab. 5.2: 售票平台的财务统计报表记录表格 11](#_Toc54809065)

[Tab. 5.3: 城市演出业务报表记录表格 12](#_Toc54809066)