# 实验九 查询优化

## 一、实验目的

1、理解查询优化的基本思想

2、掌握书写高效查询的方法

## 二、预备知识

SQL Server 查询优化器通过分析查询和决定最有效的执行计划来自动操作查询优化。优化器通过分析查询决定查询中的哪个子句可以优化，然后为可以优化的子句选择有用的索引。最后，优化器比较所有可能的执行计划，选择最有效的一个计划来执行。

## 三、实验示例

可以在执行一个查询时，使用 WHERE 子句限制**必须**处理的行的数目。除非绝对需要，否则，应避免必须读取并处理表的所有行的非限制查询。例如，以下的限制查询：

```
select  Sno  from  Student
where  Sdept = '计算机'
```

比以下的非限制查询有效：

```
select  Sno  from  Student
```

要避免给客户返回一个**很大**的数据集。限制结果集的大小可以提高查询性能，这可以减少网络 I/O 的次数和提高并发相关性能。

因为 WHERE 子句是优化器的**主要焦点**，所以要使用可以利用有用索引的查询。对于某个给定的列，一般使用以下索引来优化其性能。

- 在该列上的单列索引；
- 以该列为索引的第一列的多列索引。

例如，数据库 Stu_Cou 中的表 Student 的列 Sno 上创建了一个索引，则以下查询将可以使用这个索引： 这样速度就会快很多，但是要注意，建立索引增加了维护代价。

```
select  Sname, Sbirth, Sdept  from  Student
where  Sno = '95001'
```

而以下查询则不能使用这个索引：

```
select  Sname , Sbirth , Sdept  from  Student
                                where  Sname = '李勇'
```

## 四、思考

考虑以下 3 种 SQL 操作，查看查询分析器给出的查询计划。数据库基本表应该有足够大的数据量，才能体现优化的效果。可以使用存储过程为数据库基本表插入数据。

查询优化主要有以下方法：

（1）建立索引；

（2）重写 SQL 语句；

（3）其他优化方法（调整参数，建立视图或临时表等）。

# 实验十 事务处理

## 一、实验目的

1、理解事务

2、掌握如何应用 SQL 语言创建事务

## 二、预备知识

1、定义：用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。

2、定义事务的语句（操作指更新操作）

BEGIN TRANSACTION：开始事务

COMMIT：事务正常结束的标志。

ROLLBACK：表示回滚，即在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部取消，滚回到事务开始时的状态。

3、事务的特性

- 原子性

事务中包括的诸操作要么全做，要么全不做。

- 一致性

数据库只包含成功事务提交的结果时，就说数据库处于一致性状态。

如果数据库系统运行中发生故障，有些事务尚未完成就被迫中断，这些未完成的事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态。

- 隔离性

一个事务的执行不能被其他事务干扰，并发执行的各个事务之间不能互相干扰。

- 持续性

一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。

## 三、实验示例

注：set **xact_abort** 用于切换当命令产生运行时的错误时是否放弃事务处理。错误可能是系统产生的，也可能是用户产生的。本质上相当于在每条语句后检测@@ERROR，如果检测到错误就回滚事务处理。请观察下面例子。

【例 10-1】创建一个事务，在 xact_abort 为 off 的情况下，修改 S_1 表中学号为‘95001’的学生的已修学分为 27，然后删除表 s_1 中学号为‘95001’的学生的记录。事务结束后，请查看 表 S_1 中学号为‘95001’学生的已修学分是否已修改为 27？

```
use Stu_Cou
set xact_abort off
go
begin tran
    update S_1
    set Spre = 27   where   Sno='95001'
    delete   from S_1 where Sno='95001'
commit tran
```

【例 10-2】创建一个事务，在 XACT_ABORT 为 on 的状况下，修改 s_1 表中学号为‘95001’的学生的已修学分为 20，然后删除表 s_1 中学号为‘95001’学生的记录。事务结束后，请查看 表 s_1 中学号为‘95001’的学生的已修学分是否已修改为 20？

```
use student
set xact_abort on
go
begin tran
    update S_1
    set Spre = 20    where    Sno='95001'

    delete    from S_1 where Sno='95001'
commit tran
```

【例 10-3】创建一个事务，修改 S_1 表中学号为‘95001’学生的已修学分为 20，然后删除表 S_1 中学号为‘95001’学生的记录。如果错误发生，则回滚事务，否则提交事务。事务结束后请查看 表 S_1 中学号为‘95001’学生的已修学分是否已修改为 20？

```
use Stu_Cou
begin tran
    update S_1
    set Spre = 20    where    Sno='95001'

    delete    from S_1 where Sno='95001'
    if @@error<>0
        rollback tran
    else
        commit tran
```

自己测试下面代码，理解：XACT_ABORT

SET XACT_ABORT ON 分为两种：

1、总体作为一个事务，整体提交或整体回滚，格式为：

```
SET XACT_ABORT ON
BEGIN TRAN
    --要执行的语句
COMMIT TRAN
GO
```

2、每个语句作为一个事务，事务在错误行终止，错误行回滚，错误行之前的不回滚，格式为：

```
SET XACT_ABORT ON
BEGIN
```

```
    --要执行的语句
END
GO
```
测试
复制代码
```
--创建测试表
use MyDB
CREATE TABLE student
(
        stuid int NOT NULL PRIMARY KEY,
        stuname varchar(50)
)
CREATE TABLE score
(
        stuid int NOT NULL REFERENCES student(stuid),
        score int
)
GO

--插入测试数据
INSERT INTO student VALUES (101,'zhangsan')
INSERT INTO student VALUES (102,'wangwu')
INSERT INTO student VALUES (103,'lishi')
INSERT INTO student VALUES (104,'maliu')
GO

----------测试事务提交----------
use MyDB
--只回滚错误行，语句还继续执行
SET XACT_ABORT OFF
BEGIN TRAN
    INSERT INTO score   VALUES (101,90)
    INSERT INTO score VALUES (102,78)
    INSERT INTO score VALUES (107,76) /* Foreign Key Error */
    INSERT INTO score VALUES (103,81)
    INSERT INTO score VALUES (104,65)
COMMIT TRAN
GO
/*
stuid       score
----------- -----------
101         90
102         78
103         81
```

```
104                 65

(4 row(s) affected)
*/

use MyDB
--事务终止并全部回滚
SET XACT_ABORT ON
BEGIN TRAN
        INSERT INTO score    VALUES (101,90)
        INSERT INTO score VALUES (102,78)
        INSERT INTO score VALUES (107,76) /* Foreign Key Error */
        INSERT INTO score VALUES (103,81)
        INSERT INTO score VALUES (104,65)
COMMIT TRAN
GO
/*
stuid               score
----------- -----------
(0 row(s) affected)
*/

use MyDB
--事务在错误行终止，错误行回滚，错误行之前的不回滚
SET XACT_ABORT ON
BEGIN
INSERT INTO score    VALUES (101,90)
        INSERT INTO score VALUES (102,78)
        INSERT INTO score VALUES (107,76) /* Foreign Key Error */
        INSERT INTO score VALUES (103,81)
        INSERT INTO score VALUES (104,65)
END
GO
/*
stuid               score
----------- -----------
101                 90
102                 78
(2 row(s) affected)
*/
```