数据库记录

Illusionna

2023 年 4 月 25 日——2023 年 5 月 25 日

20:36, Thursday 25th May, 2023

目录

第一部分　数据库操作

1.1　创建数据库

```
CREATE DATABASE Illusionna
ON
PRIMARY
(
        NAME = 'Illusionna',    // 逻辑名
        FILENAME = 'A:\Illusion\Illusionna.mdf',
        SIZE = 12MB,
        MAXSIZE = UNLIMITED,
        FILEGROWTH = 2MB
)
// 创建主要数据文件

FILEGROUP myGroup
(
        NAME = 'Delusionna_1',
        FILENAME = 'A:\Illusion\Delusionna_1.ndf',    // 物理名
        SIZE = 7MB,
        MAXSIZE = UNLIMITED,
        FILEGROWTH = 20%
),
(
        NAME = 'Delusionna_2',
        FILENAME = 'A:\Illusion\Delusionna_2.ndf',
        SIZE = 4MB,
        MAXSIZE = UNLIMITED,
        FILEGROWTH = 5MB
)
// 创建次要数据文件以及归列到 myGroup 文件组

LOG ON
(
        NAME = 'Illusionna_log',
        FILENAME = 'A:\Illusion\Illusionna.ldf',
        SIZE = 10MB,
```

```
35          MAXSIZE = 100MB,
36          FILEGROWTH = 3%
37  )
38  GO
39  // 创建事务日志文件
```

1.2 修改数据库

事务日志文件一旦创建后最好不要修改.

1.2.1 MODIFY 修改

```
1   ALTER DATABASE Illusionna
2   MODIFY FILE
3   (
4           NAME = 'Illusionna',    // 修改的逻辑名
5           SIZE = 120MB,
6           MAXSIZE = UNLIMITED,
7           FILEGROWTH = 50%
8   )
9   GO
```

1.2.2 ADD 增加

```
1   ALTER DATABASE Illusionna
2   ADD FILE
3   (
4           NAME = 'Delusionna_3',
5           FILENAME = 'A:\Illusion\Delusionna_3.ndf',
6           SIZE = 3MB,
7           MAXSIZE = UNLIMITED,
8           FILEGROWTH = 5MB
9   )
10  GO
```

1.3 删除数据库

删除数据库前，必须确保数据库出于自由状态，不能被占用，否则删除过程会失败.

```
DROP DATABASE Illusionna
```

1.4 分离数据库

```
EXEC SP_DETACH_DB 'Illusionna'
```

1.5 附加数据库

1.5.1 创建数据库命令式

```
CREATE DATABASE Illusionna
ON
(
        FILENAME = 'A:\Illusion\Illusionna.mdf'
)
FOR ATTACH
GO
```

1.5.2 系统存储过程式

```
EXEC SP_ATTACH_DB
        @DBNAME = 'Illusionna',
        @FILENAME1 = 'A:\Illusion\Illusionna.mdf',
        @FILENAME2 = 'A:\Illusion\Delusionna.mdf'
```

1.6 收缩数据库

将 Illusionna 数据库收缩至原先剩余空间的 50%.

```
DBCC SHRINKDATABASE ('Illusionna', 50)
```

1.7 数据库备份

1.7.1 完整备份

占用内存大，但备份效果绝佳.

```
BACKUP DATABASE Illusionna
TO DISK = 'A:\Illusion\Illusionna_1.bak'
WITH FORMAT
```

1.7.2 差异备份

第一次备份必须是完整性备份，然后方可进行差异备份，差异备份占用内存小，但效果不好.

```
BACKUP DATABASE Illusionna
TO DISK = 'A:\Illusion\Illusionna_2.bak'
WITH DIFFERENTIAL
```

1.8 数据库还原

1.8.1 完整还原

```
RESTORE DATABASE Illusionna
FROM DISK = 'A:\Illusion\Illusionna_1.bak'
WITH REPLACE NORECOVERY
```

1.8.2 差异还原

```
RESTORE DATABASE Illusionna
FROM DISK = 'A:\Illusion\Illusionna_2.bak'
WITH RECOVERY
```

第二部分 表操作

2.1 创建表

```
USE Illusionna
```

```
GO
CREATE TABLE Student
(
        studentID NCHAR(9) NOT NULL PRIMARY KEY,
        studentName NVARCHAR(5) NOT NULL,
        studentSex NCHAR(2) NULL,
        studentAge TINYINT NULL DEFAULT 0
)
// PRIMARY KEY 是主键约束
GO
```

2.2 修改表结构

2.2.1 ADD 增加

```
USE Illusionna
GO
ALTER TABLE Student
ADD studentRemark NTEXT
// NTEXT 是Unicode 编码的备注型
GO
```

2.2.2 ALTER COLUMN 修正

```
USE Illusionna
GO
ALTER TABLE Student
ALTER COLUMN studentAge INT
GO
```

2.2.3 DROP 删除

```
USE Illusionna
GO
ALTER TABLE Student
DROP COLUMN studentRemark, studentSex
```

```
GO
```

2.3 删除表

```
USE Illusionna
GO
DROP TABLE Student
GO
```

2.4 修改表内容

2.4.1 INSERT 插入

如果属性列不允许为空（NOT NULL），则插入时必须传实参，或者默认值约束，否则报错.

```
USE Illusionna
GO
INSERT Student(studentID, studentName)
VALUES('2021150XX', '幻')
INSERT Student(studentID, studentName)
VALUES('2021150YY', '象')
INSERT Student
VALUES('2021150XY', '幻象', '男', 19, '他是一位虚无主义者.')
GO
```

2.4.2 UPDATE 更新

```
USE Illusionna
GO
UPDATE Student
SET studentName = '约罗'
WHERE studentID = '202115024'
GO
// WHERE 是条件，SET 是更新内容

UPDATE Course
```

```
SET courseCredit = courseCredit + 1.5
WHERE courseName = '数学建模'
GO
```

2.4.3 DELETE 删除

```
USE Illusionna
GO
DELETE FROM Student
WHERE studentID = '202115012'
// 删除表中的一行数据
GO

DELETE FROM Course
// 删除 Course 表所有数据，但没有删除 Course 表
GO
```

第三部分　约束操作

3.1 实体完整性

3.1.1 主键约束

```
USE Illusionna
GO
ALTER TABLE Student
ADD PRIMARY KEY (studentID)
// 单主键

ALTER TABLE StudentMark
ADD PRIMARY KEY (studentID, courseID)
// 复合主键

ALTER TABLE Course
ADD CONSTRAINT PK_Course PRIMARY KEY (courseID)
// PK_Course 是主键名称
GO
```

3.1.2 唯一性约束

```
USE Illusionna
GO
ALTER TABLE Course
ADD UNIQUE (courseName)

ALTER TABLE Institute
ADD CONSTRAINT U_Institute UNIQUE (instituteName)
// U_Institute 是唯一性约束名称
GO
```

3.2 域完整性

3.2.1 空值约束

```
USE Illusionna
GO
ALTER TABLE Institute
ALTER COLUMN instituteID NCHAR(12) NOT NULL

ALTER TABLE Student
ALTER COLUMN studentRemark NTEXT NULL
GO
```

3.2.2 默认值约束

```
USE Illusionna
GO
ALTER TABLE Student
ADD DEFAULT 12 FOR studentAge

ALTER TABLE Student
ADD CONSTRAINT D_Student DEFAULT 18 FOR studentAge
// D_Student 是默认值约束名称
GO
```

3.2.3 检查约束

```
1 USE Illusionna
2 GO
3 ALTER TABLE Student
4 ADD CHECK (studentBirthday < GETDATE())
5
6 ALTER TABLE Mark
7 ADD CONSTRAINT C_Mark CHECK (markScore >=0 AND
      markScore <= 100)
8 // C_Mark 是检测约束名称
9 GO
```

3.3 引用完整性

```
1 USE Illusionna
2 GO
3 ALTER TABLE StudentMark
4 ADD FOREIGN KEY (courseID) REFERENCES Course(courseID)
5 // StudentMark 表的courseID 引用Course 表的 courseID
6
7 ALTER TABLE StudentMark
8 ADD FOREIGN KEY (studentID) REFERENCES Student(studentID)
9 // StudentMark 表的studentID 引用Student 表的 studentID
10
11 ALTER TABLE Student
12 ADD CONSTRAINT FK_Student FOREIGN KEY (studentInstituteID)
      REFERENCES Institute(instituteID)
13 // FK_Student 是外键约束名称
14 GO
```

3.4 索引

索引分类：

- **索引键值唯一性：**唯一索引、不唯一索引.
- **索引列个数：**单列索引、组合索引或符合索引.

- **索引组织方式：**聚集索引、非聚集索引.

```
USE Illusionna
GO
CREATE UNIQUE CLUSTERED INDEX nameIndex
ON StudentCopy (studentName)
```

第四部分　数据查询

4.1　投影列

4.1.1　仅查询

```
USE Illusionna
GO
SELECT studentID, studentName
FROM Student

SELECT *
FROM Institute

SELECT studentID AS '学号', studentName AS '姓名'
FROM Student

SELECT '学号' = studentID, '姓名' = studentName
FROM Student
GO
```

4.1.2　查询 & 修改

```
USE Illusionna
GO
SELECT courseID, courseCredit + 0.5
FROM Course
GO
```

4.1.3 CASE 表达式替换

```
USE Illusionna
GO
SELECT studentID, '排名' =
CASE studentGrade
        WHEN studentGrade >= 80 THEN '优秀'
        WHEN studentGrade >= 60 THEN '合格'
        ELSE '不及格'
END
FROM StudentMark
// CASE 后面接studentGrade 筛选后赋给'排名'

SELECT studentID, studentRank =
CASE
        WHEN studentGrade >= 80 THEN '优秀'
        WHEN studentGrade >= 60 THEN '合格'
        ELSE '不及格'
END
FROM StudentMark
// 筛选后赋给 studentRank

SELECT studentID,
CASE
        WHEN studentGrade >= 80 THEN '优秀'
        WHEN studentGrade >= 60 THEN '合格'
        ELSE '不及格'
END
AS studentRank
FROM StudentMark
// SELECT 一个CASE–END 体，筛选后AS 一个名称studentRank
GO
```

4.2 投影行

4.2.1 DISTINCT 唯一化

```
USE Illusionna
```

```
GO
SELECT DISTINCT instituteID
FROM Student
// 输出 Student 表中的学院号，并且不重复
GO
```

4.2.2 TOP 顶行

```
USE Illusionna
GO
SELECT TOP 5 *
FROM StudentMark
// 返回 StudentMark 表的前五行的所有信息

SELECT TOP 20 PERCENT *
FROM StudentMark
// 返回 StudentMark 表的前20% 的所有信息
GO
```

4.2.3 任意行

```
USE Illusionna
GO
DECLARE @n INT
SET @n = 3
SELECT TOP (@n) studentName
FROM Student
GO
```

4.2.4 WHERE 条件选择

```
USE Illusionna
GO
SELECT *
FROM StudentMark
WHERE studentGrade >= 90
```

```
// 选择学生成绩大于 90 分以上的同学信息

SELECT studentName AS '姓名'
FROM Student
WHERE DATEDIFF (Year, Birthday, GETDATE()) < 20
            AND
            studentSex = '女'
// 选择年龄小于 20 岁的女生的姓名
GO
```

4.2.5 限制取值

```
USE Illusionna
GO
SELECT authorName, authorProvince
FROM Author
WHERE authorProvince IN ('合肥','成都')
// IN 表示在集合中

SELECT authorName, authorProvince
FROM Author
WHERE authorProvince = '合肥'
            OR
            authorProvince = '成都'
// 用 OR 和= 号限制取值
GO
```

4.2.6 限制范围

```
USE Illusionna
GO
SELECT *
FROM Student
WHERE YEAR(studentBirthday) BETWEEN 2000 AND 2020
// BETWEEN ... AND ... 表示介于之间
GO
```

4.2.7 LIKE 模糊模式匹配

```
USE Illusionna
GO
SELECT *
FROM Student
WHERE studentName LIKE '特__'
// 两个 _ 线表示两个占位符，查询“特某某”

SELECT *
FROM Student
WHERE studentName LIKE '马_'
// 一个 _ 线表示一个占位符，查询“马某”

SELECT *
FROM Student
WHERE studentName LIKE '术%'
            AND
            LEN(studentName) = 4
// % 号表示无限占位符，并且姓名长度为四，查询姓“术”

SELECT *
FROM Student
WHERE studentName LIKE '[^Z]%'
// [^Z] 表示不是 Z 开头，% 无限占位，查询首字母非 Z 的姓名
GO
```

4.2.8 空值查询

```
USE Illusionna
GO
SELECT *
FROM Student
WHERE studentName IS NULL
GO
```

= 号单值精确匹配，IN 多值精确匹配，LIKE 多值模糊匹配.

4.3 连接

4.3.1 传统连接方式

```
1  USE Illusionna
2  GO
3  SELECT studentName,instituteName
4  FROM Student,Institute
5  WHERE Student.instituteID = Institute.instituteID
6              AND
7              studentSex = '女'
8  GO
9  // FROM 后面接上所有用到的表，WHERE 后面接上查询条件
10
11
12 // 查询数统学院学生选修课程信息
13 USE Illusionna
14 GO
15 SELECT studentName,instituteName,courseName,StudentMark.
       markScore
16 FROM Student,Institute,Course,StudentMark
17 WHERE Student.studentID = StudentMark.studentID
18             AND
19             Course.courseID = StudentMark.courseID
20             AND
21             Institute .instituteName = '数统学院'
22 GO
23 // StudentMark.markScore 是指明某张表中的某个属性，防止属性重名
```

4.3.2 SQL 连接方式

```
1  // 先将表全部连接再增加条件
2  USE Illusionna
3  GO
4  SELECT studentName,courseName,markScore
5  // JOIN 从右往左连接
6  FROM Student JOIN StudentMark JOIN Course
7  ON Course.courseID = StudentMark.courseID
```

```
ON Student.studentID = StudentMark.studentID
// 注意条件的逻辑顺序
GO

// 两两连接再写条件
USE Illusionna
GO
SELECT studentName,courseName,markScore
FROM Student JOIN StudentMark
ON Student.studentID = StudentMark.studentID
JOIN Course
ON StudentMark.courseID = Course.courseID
// 依然要注意连接和条件的逻辑顺序
GO
```

4.3.3 LEFT JOIN 左连接

```
// 查询学生选修课程，如果有课程没学生选修，也输出课程
USE Illusionna
GO
SELECT studentID,courseName
FROM Course LEFT JOIN StudentMark
ON Course.courseID = StudentMark.courseID
GO
```

4.3.4 RIGHT JOIN 右连接

```
// 查询学生选修课程，如果有学生没课程选修，也输出学生
USE Illusionna
GO
SELECT studentName,StudentMark.courseID
FROM StudentMark RIGHT JOIN Student
ON StudentMark.studentID = Student.studentID
GO
```

LEFT JOIN 表示左侧有右侧没有，RIGHT JOIN 表示右侧有左侧没有.

4.3.5 自连接

```
// 自连接必须给表分别起别名加以区分
USE Illusionna
GO
SELECT aliasA.courseID,aliasA.markScore
FROM StudentMark aliasA JOIN StudentMark aliasB
ON aliasA.studentID = aliasB.studentID
GO

USE Illusionna
GO
SELECT aliasA.courseID,aliasA.markScore
FROM StudentMark aliasA,StudentMark aliasB
WHERE aliasA.studentID = aliasB.studentID
GO
```

4.4 聚合函数

4.4.1 SUM( ) 合计

```
USE Illusionna
GO
SELECT SUM(markScore) AS '总分'
FROM StudentMark
WHERE StudentMark.courseID = '120'
GO
```

4.4.2 AVG( ) 平均

```
USE Illusionna
GO
SELECT AVG(markScore) AS '均分'
FROM StudentMark
WHERE StudentMark.courseID = '120'
GO
```

4.4.3 MAX( ) 最高 & MIN( ) 最低

```
USE Illusionna
GO
SELECT MAX(markScore) AS '最高分',MIN(markScore) AS '最低分'
FROM StudentMark
WHERE StudentMark.courseID = '120'
GO
```

4.4.4 COUNT( ) 统计

```
USE Illusionna
GO
SELECT COUNT(*) AS '全部行数',COUNT(studentAge) AS '年龄个数'
FROM Student
// 如果有的学生年龄未知，是空值，那么不统计
GO
```

4.5 排序

4.5.1 单类排序

```
USE Illusionna
GO
SELECT studentName,studentID
FROM Student
WHERE studentProvince = '四川'
ORDER BY studentID DESC
// 选择四川省学生姓名合学号，DESC 按照学号降序，默认升序
GO
```

4.5.2 复合排序

```
USE Illusionna
GO
SELECT StudentMark.courseID,StudentMark.markScore
FROM StudentMark
```

```
5  ORDER BY StudentMark.courseID,StudentMark.markScore DESC
6  // 先升序排列课程号，再把每个课程号中成绩降序排列
7  GO
```

4.6 分组

4.6.1 单分组

```
1  // 统计每门课程平均分，并按平均分降序输出
2  USE Illusionna
3  GO
4  SELECT courseName,AVG(markScore) AS '均分'
5  FROM StudentMark,Course
6  WHERE Course.courseID = StudentMark.courseID
7  GROUP BY courseName
8  // GROUP BY courseName 分为一组
9  ORDER BY AVG(markScore) DESC
10 // 均分降序排列
11 GO
```

4.6.2 多分组

```
1  // 统计每个学院男女生人数
2  USE Illusionna
3  GO
4  SELECT instituteName,studentSex,COUNT(*) AS '人数'
5  FROM Student,Institute
6  WHERE Student.instituteID = Institute.instituteID
7  GROUP BY instituteName,studentSex
8  // 先把性别分组，再给每个性别中的学院分组
9  GO
```

GROUP BY 和 JOIN 条件从右往左执行，ORDER BY 从左往右执行；
没有聚合函数的 GROUP BY 失效，退化成 DISTINCT 效果.

4.6.3 CUBE 全集统计

```
// 统计每门课程选修人数，并且输出选修课总人数
USE Illusionna
GO
SELECT courseName,COUNT(*) AS '人数'
FROM Course,StudentMark
WHERE StudentMark.courseID = Course.courseID
GROUP BY courseName WITH CUBE
// 正常输出，并且最后一行输出统计全部人数
GO
```

4.6.4 ROLLUP 子集统计

```
// 统计每个学院男女生人数
USE Illusionna
GO
SELECT instituteName,studentSex,COUNT(*) AS '人数'
FROM Student,Institute
WHERE Student.instituteID = Institute.instituteID
GROUP BY instituteName,studentSex WITH ROLLUP
// 正常输出，并且每个学院最后一行输出统计该学院人数
GO
```

4.6.5 HAVING 筛选分组

```
// 统计各省份女大学生人数
USE Illusionna
GO
SELECT studentProvince,studentSex,COUNT(*) AS '人数'
FROM Student
GROUP BY studentProvince,studentSex
HAVING studentSex = '女'
// 对分组结果筛选，保留性别为女的组
GO


```

```
◄ ★ ✳ ♠ ⊛ ♠ ✳ ★ ►
// 等价效果
USE Illusionna
GO
SELECT studentProvince,studentSex,COUNT(*) AS '人数'
FROM Student
WHERE studentSex = '女'
GROUP BY studentProvince,studentSex
GO
◄ ★ ✳ ♠ ⊛ ♠ ✳ ★ ►


// 查询均分在 90 以上的课程名称、学生和均分
USE Illusionna
GO
SELECT courseName,studentName,AVG(markScore) AS '均分'
FROM Course,Student,StudentMark
WHERE Student.studentID = StudentMark.studentID
        AND
        StudentMark.courseID = Course.courseID
GROUP BY courseName
// 查询结果按照 courseName 分组
HAVING AVG(markScore) >= 90
// 保留均分在 90 以上的分组结果
GO
```

4.7 子查询

4.7.1 无关子查询

4.7.2 相关子查询

4.8 集合操作

4.8.1 并

4.8.2 交

4.8.3 差

4.9 查询结果储存