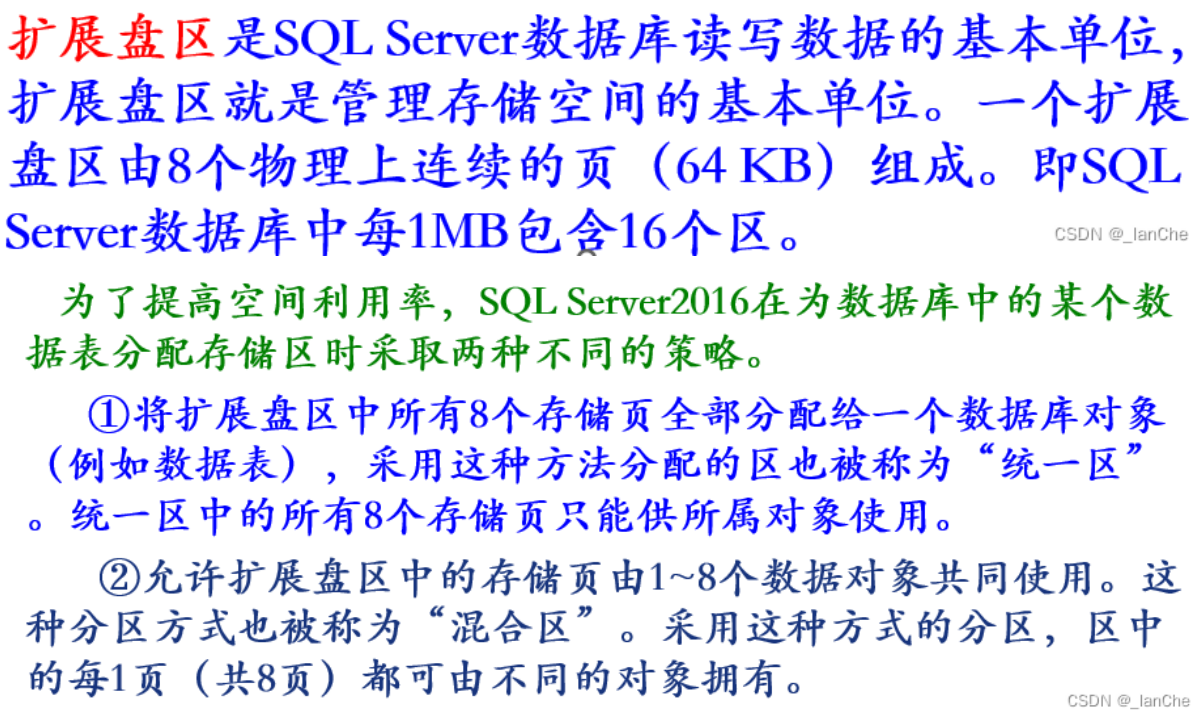
选择题

- SQL Server数据文件的存储结构
  - 数据页

    > SQL Server将8KB的数据划分为一页。即在SQL Server 数据库中的1MB数据中包含128页。包括数据页、索引页、文本/图像页等8种。每个页的开头为96字节的系统信息。数据区占有8060个字节，页尾的行偏移数组占有36个字节。

  - 区间



- 数据库文件的后缀
  - 主要数据文件：包含数据库的启动信息，并指向数据库中的其他文件。文件扩展名是 `.mdf`
  - 次要文件：将数据分散到多个磁盘上，文件扩展名是 `.ndf`
  - 事务日志文件：保存用于恢复数据库的日志信息，文件扩展名是 `.ldf`

- 管理数据库相关命令
  - 查看数据库状态信息

    > 在实际生产过程中的数据库总是处于一个特定的状态中，若要确认数据库的当前状态，代码如下

```
1. Select name,state,state_desc From sys.databases
```

```
![在这里插入图片描述](https://img-blog.csdnimg.cn/b834c691ff59475d9fde2e4b933bcc14.png)

```sql
2. Select name,physical_name,type,type_desc,state, state_desc From sys.master_files
```

- 能够在服务器之间迁移数据库的操作是什么
  - **分离**用户数据库
  - **附加**数据库：附加数据库可以将已经分离的数据库重新附加到当前或其他SQL Server 2016的实例

    > 在SQL Server 2016中，除了系统数据库外，其他数据库都可以从服务器的管理中进行分离，以脱离服务器的管理，同时保持数据文件与日志文件的完整性和一致性。而分离出来的数据库可以附加到其他SQL Server服务器上，构成完整的数据库。分离和附加是系统开发过程中的重要操作。

- 局部变量标识符的定义

```
declare
DECLARE {@local_variable   data_type}[, …n]
```

- 了解系统函数的使用

```

```

- 在游标的执行过程中，@@FETCH_STATUS变量返回值
  - 其值有以下三种，分别表示三种不同含义：【返回类型 integer 】

返回值	描述
0	FETCH 语句成功。
-1	FETCH 语句失败或行数据超出游标数据结果集范围。
-2	提取的行数据不存在。
-9	游标未执行提取操作。

- 两种特殊的表：INSERTED表和DELETED表的相关内容
  - 插入表（INSERTED）里存放的是**更新后的数据**

    > 对于插入记录操作来说，插入表里存放的是要插入的数据；
    >
    > 对于更新记录操作来说，插入表里存放的是要更新的记录。
```

- 删除表（DELETED）里存放的是**更新前的数据**

  > 对于更新记录操作来说，删除表里存放的是更新前的记录（更新完后即被删除）；
  >
  > 对于删除记录操作来说，删除表里存入的是被删除的旧记录。

- 如何创建数据库用户名

```
-- 创建登录名和创建用户可以一起写，这里先创建了一个名为Alice的用户，登录密码是：henry626626
CREATE LOGIN Alice
    WITH PASSWORD = 'henry626626' /*, DEFAULT_DATABASE = STUDENT;   这里不指定的话，默认为master数据库*/
GO

-- 给刚刚创建的登录名Alice创建一个数据库用户Alice
CREATE USER Alice FOR LOGIN Alice;
GO

-- 创建没有登录名的用户。不能登录，但可以被授予权限
CREATE USER Mark WITHOUT LOGIN;
```

- 备份语句

```
USE master
GO
BACKUP DATABASE DB_TeachingMS
TO DISK='D:\TeachingMS_Bak\TeachingMS.bak'
GO
```

[快速入门：备份和还原数据库 - SQL Server | Microsoft Docs]

[BACKUP (Transact-SQL) - SQL Server | Microsoft Docs]

判断题

- SQL Server 数据库包含哪些操作系统文件
  - 数据库至少具有两个操作系统文件
    - 主数据文件：包含数据和数据库对象
    - 日志文件：包含恢复数据库中的所有事务所需的信息

- 文件组的作用
  - 文件组可以帮助数据库管理人员执行相应的数据布局，以及某些管理任务。

> 对于大型数据库，如果硬件设置上需要多个磁盘驱动器，就可以把特定的对象或文件分配到不同的磁盘上，将数据库文件组织成用户文件组。

- 通过创建用户文件组，可以将数据文件集合起来，以**便于管理、数据分配和放置。**

> 利用文件组，可以在特定的文件中定位特定的对象，从而将频繁查询和频繁修改的文件分离出来，以提高磁盘驱动器的效率，减少磁盘驱动器的争。

- 数据库快照可以由谁创建
  - **任何能创建数据库**的用户都可以创建数据库快照。

- 如何对多个变量赋值
  -

SELECT	SET	
同时对多个变量同时赋值时	支持	不支持
表达式返回多个值时	将返回的最后一个值赋给变量	出错
表达式未返回值时	变量保持原值	变量被赋null值

```
declare @name varchar(20),@num int =0;
select @name=StuName,@num=sum(TotalScore) from TB_Student a,TB_Grade b where
a.StuID=b.StuID group by a.StuID,a.StuName
```

- 简单CASE语句和搜索CASE语句
  - 简单 Case 语句将某个表达式与一组简单表达式进行比较，以返回特定的值。

```
Simple Case Statement
CASE [input_expression]
WHEN when_expression THEN when_true_result_expression
[...n]
[ELSE else_result_expression]
END

CASE sex
WHEN '1' THEN '男'
WHEN '2' THEN '女'
ELSE '其他' END
```

  - 搜索 Case 语句计算一组布尔表达式，以返回特定的值。

```
Search Case Statement
CASE
WHEN Boolean_expression THEN when_true_result_expression
[...n]
[ELSE else_result_expression]
END

CASE
WHEN sex = '1' THEN '男'
WHEN sex = '2' THEN '女'
ELSE '其他' END
```

- 各种函数返回的值

  根据函数返回值形式的不同将用户定义函数分为3种类型。

  - 标量值自定义函数

    > 标量值函数返回一个确定类型的标量值，其函数值类型为系统数据类型（除text、ntext、image、cursor、timestamp、table类型外）。函数体语句定义在BEGIN...END语句内。

  - 内嵌表值自定义函数

    > 内嵌表值函数返回的函数值为一个表。内嵌表值函数的函数体不使用BEGIN...END语句，其返回的表是RETURN子句中的SELECT命令查询的结果集，其功能相当于一个提供参数化的视图。

  - 多语句表值自定义函数

    > 多语句表值函数可以看作标量函数和内嵌表值函数的结合体。其函数的返回值也是一个表，但函数体也用BEGIN...END语句定义，返回值的表中的数据由函数体中的语句插入。因此，多语句表值函数可以进行多次查询，弥补了内嵌表值自定义函数的不足。

- 输出参数
  - 自定义函数不支持输出参数
  - 存储过程参数中，添加 `OUTPUT` 标识符

- 登录名与数据库用户的关系
  - 登录名是用来登录SQL Server服务器的登录账户，而数据库用户是登录SQL Server服务器后用来访问具体某个数据库的用户账户
  - 一般一个登录名总是与一个或多个数据库用户相关联，这样才能访问对应的数据库
  - 要访问特定的数据库还必须具有对应的数据库用户名，而用户名在特定的数据库内创建时，必须**关联一个登录名**。

- 权限管理相关语句
  - 授权

```
GRANT <权限> [, <权限>]
[ON <对象类型> <对象名>]
TO <用户> [, <用户>]
[WITH GRANT OPTION][AS 用户]
```

- 回收权限

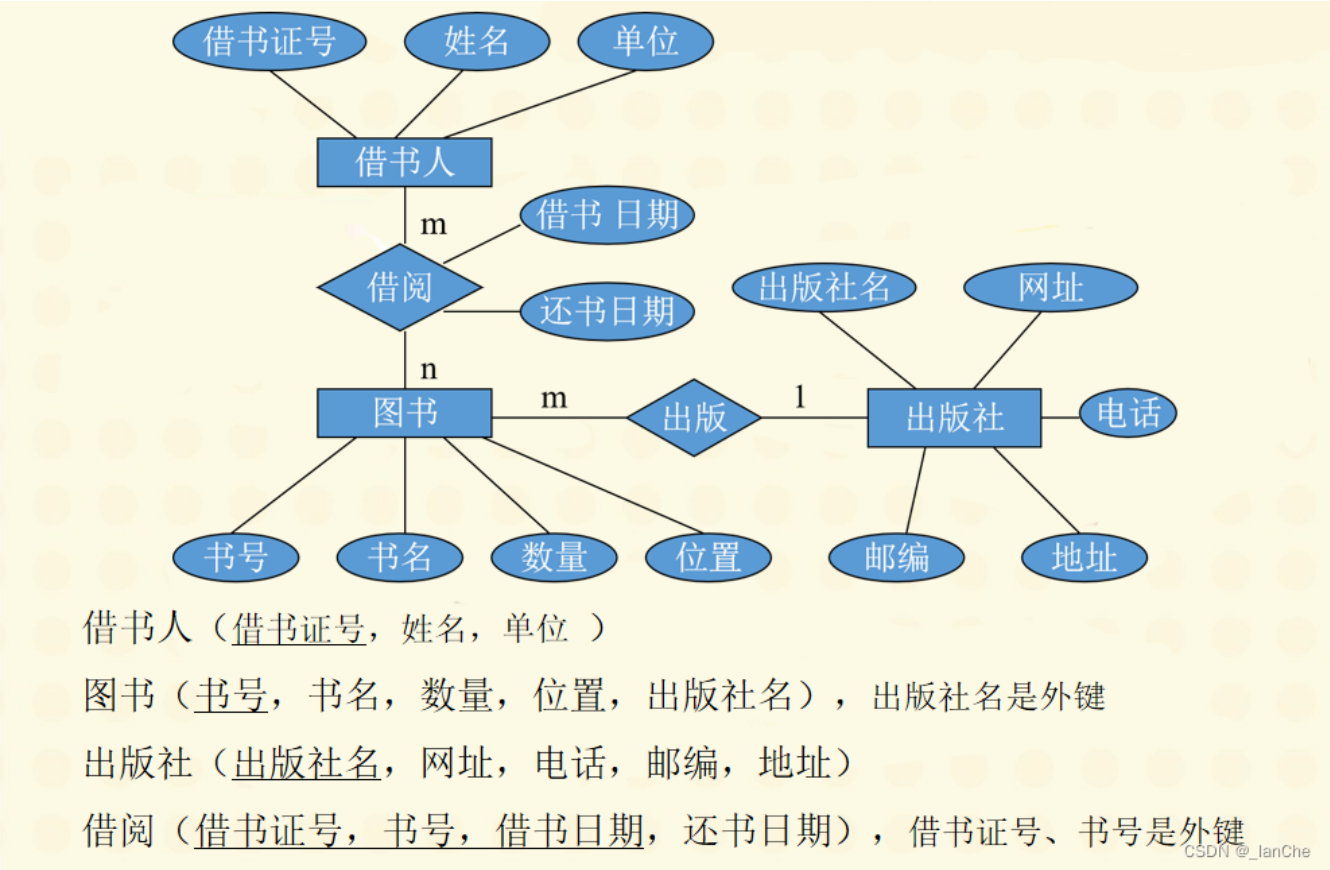
```
REVOKE [GRANT OPTION FOR] <权限> [, <权限>]
[ON <对象类型> <对象名>]
FROM <用户> [, <用户>]
[CASCADE] [AS 用户]
```

- 拒绝

```
DENY <权限> [, <权限>]
[ON <对象类型> <对象名>]
TO <用户> [, <用户>]
```

- 临时备份与永久备份的差异
  - 临时备份：直接指定路径进行备份
  - 永久备份：通过逻辑备份设备进行备份

设计题



- 多对多需拆成新的表

简答题

- 改表
  - 增加字段

```
ALTER TABLE <表名> Add <字段名> <字段类型>

增加字段Address，数据类型为varchar(60)
ALTER TABLE TB_Teacher ADD Address varchar(60)
```

  - 更改字段类型

```
ALTER TABLE <表名> ALTER COLUMN <字段> <字段类型>
alter table 表名 modify 表中字段名 新类型;

将字段“Sex”的字段类型改为char(2)
ALTER TABLE TB_Teacher ALTER COLUMN Sex char(2)
```

  - 修改表中字段名

```
alter table 表名 change 旧字段名 新字段名 新字段名类型;
```

- 删除表中字段

```
alter table 表名 drop 字段名;
```

- 修改表名

```
rename table 旧表名 to 新表名;
```

- 向表中插入字段

```
alter table 表名 add 新字段 新字段类型 after 字段名;  将新字段插入到指定字段之后
```

- 创建唯一性约束

```
ALTER TABLE <表名> ADD CONSTRAINT <约束名> UNIQUE (<字段名>)

创建TeacherName为唯一性约束
ALTER TABLE TB_Teacher ADD CONSTRAINT TN UNIQUE (TeacherName)
```

- 设置默认值

```
ALTER TABLE <表名> ADD CONSTRAINT <约束名> DEFAULT <默认值> For <字段名>

添加TitleID的默认值为“T3”
ALTER TABLE TB_Teacher ADD CONSTRAINT DF DEFAULT 'T3' FOR TitleID
```

- 增删改查

```
insert into 表名(字段名1,…,字段N) values(value1,…,valueN);
select * from 表名; 其中星号*表示表中所有字段
update 表名 set 要修改的字段名=value;
delete from 表名 where 条件;
```

- 查询
  - 单表查询

```
查询学生姓名包含“丽”这个字的学生的所有信息，并按学号排序 %任意字符 _单个字符 DESC降序排列
select * from TB_Student where StuName like '%丽%' order by StuID

查询1985年后出生的男生信息
select * from TB_Student where year(Birthday)>= '1985' and Sex = 'M'

其他关键字的使用 count() max() min() avg() between..and..
group by having..

查询成绩表中总评平均成绩高于75分的班级编号，列出班级编号及平均分，按总评平均成绩降序排列。
select ClassID round(avg(TotalScore),1) as 平均分 from TB_Grade group by ClassID
having avg(TotalScore)>75 order by avg(TotalScore) desc
```

- 多表查询

```
根据课程班和任课教师对学生平均分进行分类汇总，并显示平均成绩大于75分的信息，要求显示标题为：课程班号、教师姓名、平均分。
select b.CourseClassID 课程班号, a.TeacherName 教师姓名,avg(b.TotalScore) 平均分from TB_Teacher a,TB_Grade b,TB_CourseClass c where b.CourseClassID = c.CourseClassID and c.TeacherID = a.TeacherID group by b.CourseClassID,a.TeacherID,a.TeacherName having avg(b.TotalScore) > 75;

左外连接(左边的表不加限制) 返回包括左表中的所有记录和右表中连接字段相等的记录,如果右表没有则为null
右外连接(右边的表不加限制)

统计每个学生的选课数（包括没有选课的学生），列出学号、姓名、选课门数。
select a.StuID,a.StuName,count(CourseClassID) 选课门数 from TB_Student a left join TB_SelectCourse b on a.StuID = b.StuID group by a.StuID,a.StuName;

找出没有选修任何课程的学生的学号、姓名及所在系名
select StuID,StuName,DeptName from TB_Student a,TB_Dept b
where a.DeptID = b.DeptID and StuID not in
(select distinct StuID from TB_SelectCourse)
```

- 权限管理
  - 授权

```
GRANT <权限> [, <权限>]
[ON <对象类型> <对象名>]
TO <用户> [, <用户>]
[WITH GRANT OPTION][AS 用户]

GRANT SELECT ON TB_Student TO Student
GRANT Update ON TB_CourseClass TO Teachers
```

  - 回收权限

```
REVOKE [GRANT OPTION FOR] <权限> [, <权限>]
[ON <对象类型> <对象名>]
FROM <用户> [, <用户>]
[CASCADE] [AS 用户]

REVOKE SELECT ON TB_Student TO Student
REVOKE Update ON TB_CourseClass TO Teachers
```

  - 拒绝

```
DENY <权限> [, <权限>]
[ON <对象类型> <对象名>]
TO <用户> [, <用户>]

DENY SELECT ON TB_Student TO Student
DENY Update ON TB_CourseClass TO Teachers
```

- 备份
  - 创建备份设备

```
EXEC sp_addumpdevice 'DISK', 'TS_Bak_Device', 'D:\TS_Bak_Device\TeachingMS.bak'
```

  - 完全备份

```
BACKUP DATABASE DB_TeachingMS TO TS_Bak_Device
```

  - 差异备份

```
BACKUP DATABASE DB_TeachingMS TO TS_Bak_Device WITH DIFFERENTIAL
```

  - 日志备份

```
BACKUP LOG DB_TeachingMS TO TS_Bak_Device
```

编程题

- 存储过程

```
CREATE PROC SP_GradeProc @CourseClassID CHAR(10)
AS
------定义课程考试比例系数变量并获取相应的值-------
DECLARE @CPart REAL,@MPart REAL,@LPart REAL
SELECT @CPart=CommonPart,@MPart=MiddlePart,@LPart=LastPart
FROM TB_CourseClass
WHERE CourseClassID=@CourseClassID
------定义用来存放平时、期中、期末、总评成绩变量-------
DECLARE @CScore REAL,@MScore REAL,@LScore REAL,@TotalScore REAL
------------------声明游标------------------
DECLARE CUR_GradeProc CURSOR FOR
SELECT CommonScore,MiddleScore,LastScore FROM TB_Grade
WHERE CourseClassID=@CourseClassID ORDER BY StuID
------------------打开游标------------------
OPEN CUR_GradeProc
-----------循环提取游标中成绩并处理------------
FETCH NEXT FROM CUR_GradeProc INTO @CScore,@MScore,@LScore
```

```
WHILE @@FETCH_STATUS = 0
BEGIN
    SET @TotalScore = ROUND((@CScore*@CPart+@MScore*@MPart+@LScore*@LPart)/100,0)
UPDATE TB_Grade SET TotalScore=@TotalScore
WHERE CURRENT OF CUR_GradeProc
FETCH NEXT FROM CUR_GradeProc INTO @CScore,@MScore,@LScore
END
----------------关闭释放游标----------------
CLOSE CUR_GradeProc
DEALLOCATE CUR_GradeProc
```

- 函数

```
编一函数，要求输入学生姓名，返回该学生的选课门数
create function CourseCount(@sname char(8))
   returns int
as
begin
   Return(select count(*) from TB_SelectCourse TBSC join TB_Student TBS on
TBSC.StuID=TBS.StuID
       where StuName=@sname)
End

执行用户自定义函数
[database_name.]owner_name.function_name ([argument_expr] [, …] )
SELECT dbo.DatetoQuarter ('2021-3-2')
```

- AFTER触发器
  - 实现选课人数自减功能

```
USE DB_TeachingMS
GO
CREATE TRIGGER TR_SelectCourse_addNum
ON TB_SelectCourse AFTER DELETE
AS
UPDATE TB_CourseClass SET SelectedNumber = SelectedNumber-1
WHERE CourseClassID = (SELECT CourseClassID FROM DELETED)
```

  - 设计一触发器，要求在TB_Student中加入学生时，自动将对应班级表中的ClassNumber加1

```
CREATE TRIGGER TR_AddStudent
ON TB_Student AFTER INSERT
AS
UPDATE TB_Class SET ClassNumber = ClassNumber+1
WHERE ClassID = (SELECT ClassID FROM INSERTED)
```

- 当向TB_Grade表插入记录后，如果成绩非空则在该学生的TotalGrade中自动加上该门课程的得分。

```
CREATE TRIGGER TR_AddStuScore
ON TB_Grade AFTER INSERT
AS
   UPDATE TB_Student SET TotalGrade = TotalGrade +INSERTED.TotalScore
   FROM TB_Student TS,INSERTED
   WHERE TS.StuID = INSERTED.StuID
```

- 在TB_Grade中插入成绩时，如果成绩大于等于60分则在该学生的TotalCredit中自动加上相应课程的学分

```
create Trigger SumCredit on TB_Grade after insert
as
   declare @credit real,@grade real
   set @grade=(select TotalScore from inserted)
   if (@grade>=60)
   begin
      set @credit=(select CourseGrade from TB_Course where CourseID=(select
CourseID from INSERTED))
      update TB_Student set TotalCredit=TotalCredit+@credit where StuID=(select
StuID from INSERTED)
   end
```