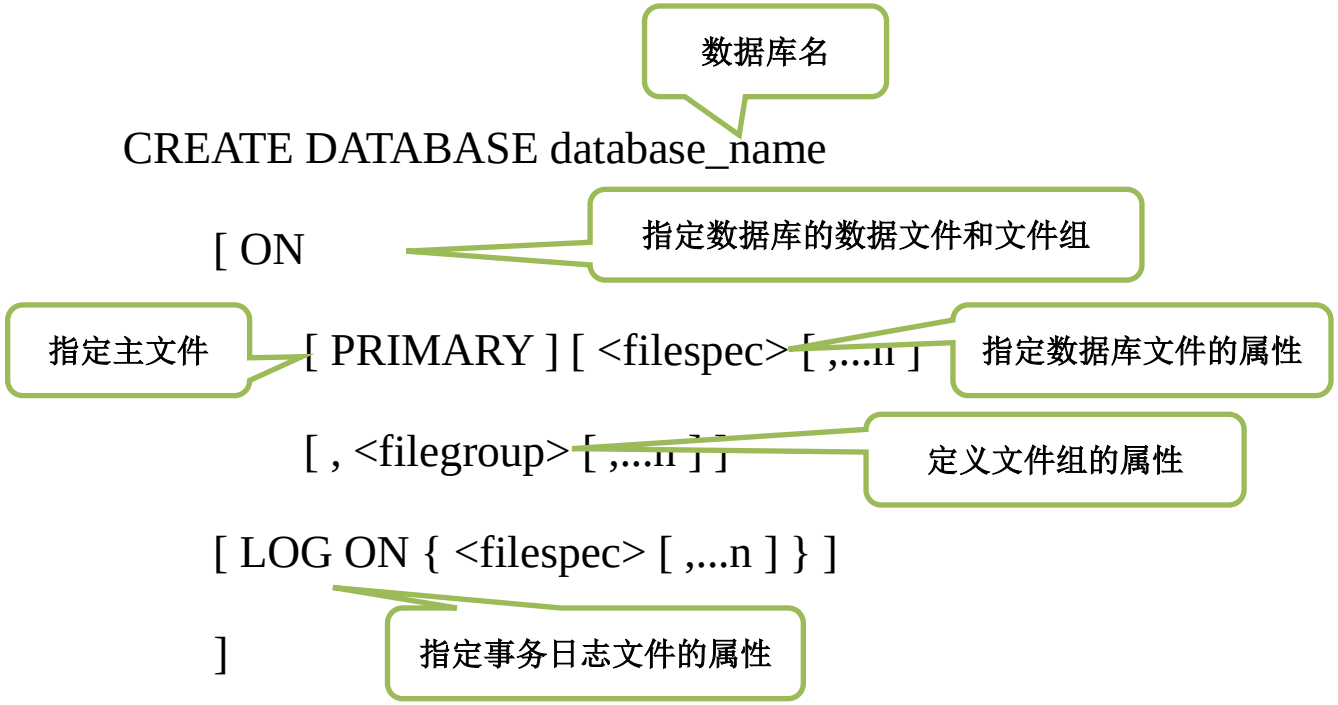
第三章：关系数据库标准语言SQL

创建和管理数据库、数据表

创建数据库语法格式


数据库名
CREATE DATABASE database_name
[ON
指定数据库的数据文件和文件组
指定主文件
[PRIMARY] [<filespec> [,...n]
指定数据库文件的属性
[, <filegroup> [,...n]]
定义文件组的属性
[LOG ON { <filespec> [,...n] }]
]
指定事务日志文件的属性

创建数据库语法格式

<filespec> ::=

{(NAME = logical_file_name ,

FILENAME = { 'os_file_name' | 'filestream_path' }

[, SIZE = size [KB | MB | GB | TB]]

[, MAXSIZE = { max_size [KB | MB | GB | TB] |

UNLIMITED }]

[, FILEGROWTH = growth_increment [KB | MB | GB | TB

| %]]

) [,...n]}

逻辑文件名

路径和文件名

文件大小不受限制，仅受磁盘可用空间限制

数据文件初始容量大小

指定文件的最大大小

文件每次的增量，有空间值和百分比两种格式

返回

创建数据库语法格式

<filegroup> ::=

{ FILEGROUP filegroup_name

[DEFAULT]

<filespec> [,...n]

}

文件组的名称

指定属于该文件组的文件，文件组中各文件的描述和数据文件描述相同

例1：要求创建一个学生-课程数据库（名称为student）。

```
Create database student
```

例2：要求在本地磁盘D的data文件夹中创建一个学生-课程数据库（名称为student1），只有一个数据文件和日志文件，文件名称分别为stu和stu_log，初始大小都为3MB，增长方式分别为10%和1MB，数据文件最大为500MB，日志文件大小不受限制。

```
CREATE   DATABASE  student1
ON
( NAME = stu,
FILENAME = 'D:\data\stu.mdf' ,
SIZE = 3MB ,
MAXSIZE = 500MB ,
FILEGROWTH = 10%)
 LOG ON
( NAME = stu_log, FILENAME = 'D:\data\stu_log.ldf' , SIZE = 3MB , MAXSIZE =
unlimited， FILEGROWTH = 1MB )
```

练习

1、创建一个名为TEST1的数据库，其初始大小为5MB，最大大小为50MB，允许数据库自动增长，增长方式是按10%比例增长。日志文件初始为2MB，最大可增长到5MB，按1MB增长。数据文件和日志文件的存放位置为SQL Server的数据库目录。

```
create database test1
  on
  (
      name='test1_data',
      filename='C:\Program Files\Microsoft SQL
Server\MSSQL10.SQL2008\MSSQL\DATA\test1.mdf',
      size=5mb,
      maxsize=50mb,
      filegrowth=10%
   )
   log on
   (
      name='test1_log',
      filename='C:\Program Files\Microsoft SQL
Server\MSSQL10.SQL2008\MSSQL\DATA\test1.ldf',
      size=2mb,
      maxsize=5mb,
      filegrowth=1mb
    );
```

练习

2、创建一个名为TEST2的数据库，它有两个数据文件，其中，主数据文件为20MB，最大大小不限，按10%增长。1个辅数据文件为20MB，最大大小不限，按10%增长，有1个日志文件，大小为50MB，最大大小为100MB，按10MB增长。

```
create database test2
  on
  primary
  (
      name='test2_data1',
      filename='d:\data\test2_data1.mdf',
      size=20mb,
      maxsize=unlimited,
      filegrowth=10%
   ),
   (
       name='test2_data2',
       filename='d:\data\test2_data1.ndf',
       size=20mb,
       maxsize=unlimited,
       filegrowth=10%
    )
    log on
    (
        name='test2_log1',
        filename='d:\data\test2_log1.ldf',
        size=50mb,
        maxsize=100mb,
        filegrowth=10mb
     );
```

练习

3、创建一个具有2个文件组的数据库TEST3。要求：（1）主文件组包括文件TEST3_dat1，文件初始大小为20MB，最大为60MB，按5MB增长；（2）有1个文件组名为TEST3Group1，包括文件TEST1_dat2，文件初始大小为10MB，最大为30MB，按10%增长。

```
create database test3
    on
    primary
    (
        name='test3_dat1',
        filename='d:\data\test3_dat1.mdf',
        size=20mb,
        maxsize=60mb,
        filegrowth=5mb
     ),
     filegroup test3group1
     (
         name='test3_dat2',
         filename='d:\data\test3_dat2.ndf',
         size=10mb,
         maxsize=30mb,
         filegrowth=10%
      );
```

修改数据库语法格式

```
Alter database database_name
{add file <filespec>[,…n] [to filegroup filegroupname]
                                                    在文件组中增加数据文件
|add log file <filespec>[,…n]                       增加日志文件
|remove file logical_file_name [with delete]        删除数据文件
|modify file <filespec>                             更改文件属性
|add filegroup filegroup_name                       增加文件组
|remove filegroup filegroup_name                    删除文件组
|modify filegroup filegroup_name                    更改文件组属性
}
```

练习

1、修改数据库TEST1现有数据文件TEST1_DATA的属性，将主数据文件的最大大小改为100MB，增长方式改为按每次5MB增长。

```
alter database test1
    modify file
    (
        name=test1_data,
        maxsize=100mb,
        filegrowth=5mb
         )
```

练习

2、先为数据库TEST1增加数据文件TEST1BAK。然后删除该数据文件。

```
alter database test1
   add file
   (
       name='test1bak',
       filename='d:\data\test1bak.ndf',
       size=10mb,
       maxsize=50mb,
       filegrowth=5%
   )
alter database test1
       remove file test1bak
```

练习

3、为数据库TEST1添加文件组FGROUP，并为此文件组添加两个大小均为10MB的数据文件。

```
alter database test1
    add filegroup fGroup
go
alter database test1
    add file
    (
        name='test1_data4',
        filename='C:\Program Files\Microsoft SQL
Server\MSSQL10.SQL2008\MSSQL\DATA\test1_data4.ndf',
        size=10mb
     ),
     (
        name='test1_data5',
        filename='C:\Program Files\Microsoft SQL
Server\MSSQL10.SQL2008\MSSQL\DATA\test1_data5.ndf',
        size=10mb
      )
      to filegroup fgroup
```

练习

4、将上题中添加的文件组FGROUP删除。（被删除的文件组中的数据文件必须先删除，且不能删除主文件组）

```
alter database test1
    remove file test1_data4
alter database test1
    remove file test1_data5
alter database test1
    remove filegroup fgroup
```

练习

5、为数据库TEST1添加一个日志文件。

```
alter database test1
    add log file
    (
        name='test1_log2',
        filename='d:\data\test1_log2.ldf',
        size=5mb,
        maxsize=10mb,
        filegrowth=1mb
     )
```

6、将上题中添加的日志文件删除。（不能删除主日志文件）

```
alter database test1
    remove file test1_log2
```

删除数据库语法格式

DROP DATABASE 〈，数据库名组〉

例1：将数据库student删除，可使用下述语句。

```
DROP DATABASE student
GO
```

注：使用DROP DATABASE语句不会出现确认信息，要小心使用，并且不能删除系统数据库，否则将导致服务器无法使用。

创建及管理数据表

- 建立数据库最重要的一步就是**创建其中的数据表**，即决定数据库包括哪些表，每个表中包含哪些字段，每个字段的数据类型等。
- 创建表的实质就是**定义表结构及约束**等属性，在创建数据表之前，先要设计表，即确定**表的名字**、所包含的**各字段名**、**字段的数据类型**、**长度**、**是否可为空值**等等，这些属性构成表结构。

创建及管理数据表

主要内容：

- 常见数据类型
- 界面方式下创建及编辑数据表
- 命令方式下创建及编辑数据表

常见数据类型

数据类型	符 号 标 识
整数型	bigint , int , smallint , tinyint
精确数值型	decimal , numeric
浮点型	float , real
货币型	money , smallmoney
位型	bit
字符型	char , varchar
Unicode 字符型	nchar , nvarchar
文本型	text , ntext
二进制型	binary , varbinary
日期时间型	datetime , smalldatetime
时间戳型	timestamp
图像型	image
其他	cursor ,sql_variant , table , uniqueidentifier

整数型

表 3.6　4 类整数的精度、长度和取值范围

整数类型	精度	长度(字节数)	数值范围
bigint(大整数)	19	8	$-2^{63} \sim 2^{63}-1$ 即 -9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807
int(整数)	10	4	$-2^{31} \sim 2^{31}-1$ 即 -2,147,483,648 ～ 2,147,483,647
smallint(短整数)	5	2	$-2^{15} \sim 2^{15}-1$ 即 -32768 ～ 32767
tinyint(微短整数)	3	1	0～255

精确数型

- 精确数值型数据由整数部分 和小数部分构成，其所有的数字都是有效位，能够以完整的精度存储十进制数。
- 精确数值型包括decimal和numeric两类，在SQL Server2008中，这两种数据类型在功能上完全等价。
- 数据的存储长度随精度变化而变化，最少为5个字节，最多为17个字节。
- 声明精确数值型数据的格式是 numeric | decimal（ p[,s] ），其中p为精度，s为小数位数，s的默认值为0。声明精确数值型数据时，其小数位数必须小于精度。

精度	长度(字节数)
1~9	5
10~19	9
20~28	13
29~38	17

浮点型

- 浮点型也称近似数值型，这种类型不能提供精确表示数据的精度，使用这种类型来存储某些数值时，可能会损失一些精度，所以它可用于处理取值范围非常大且对精确度要求不太高的数值量，如一些统计量。
- 有两种浮点型：float[(n)]和real。两者通常使用科学计数法表示数据。

类型	精度	长度(字节数)	数值范围
real	7	4	-3.40E+38 ～ 3.40E+38
float[(n)] (当 n 在 1~24 之间时)	7	4	-3.40E+38 ～ 3.40E+38
float[(n)] (当 n 在 25~53 之间时)	15	8	-1.79E+308 ～ 1.79E+308

货币型

类型	精度	小数位数	长度(字节数)	数值范围
money	19	4	8	-2^{63} ～ $2^{63}-1$，即： -922,337,203,685,477.5808 ～ 922,337,203,685,477.5807
smallmoney	10	4	4	-2^{31} ～ $2^{31}-1$，即： -214,748.3648 ～214,748.3647

- 当向表中插入money或smallmoney类型的值时，必须在数据前面加上货币表示符号（$），并且数据中间不能有逗号（，）；若货币值为负数，则需要在符号$的后面加上负号（-）。例如 $15000.32　$-20000.9088　$680

字符型

- **char[(n)]**
- 定长字符数据类型，其中n定义字符型数据的长度，n在1到8000之间，缺省为1。当表中的列定义为char(n)类型时，若实际要存储的串长度不足n时，则在串的尾部添加空格以达到长度n，所以char(n)的长度为n。
- **varchar[(n)]**
- 变长字符数据类型，其中n的规定与定长字符型char中n完全相同，但这里n表示的是字符串可达到的最大长度。varchar(n)的长度为输入的字符串的实际字符个数，而不一定是n。
- 当列中的字符数据值长度接近一致时，如姓名，可使用char；而当列中的数据值长度显著不同时，使用varchar较为恰当，可以节省存储空间。

文本型

- 用于存储大量的字符数据，如较长的备注、日志信息等。
- text：数据的存储长度为实际字符数字节；
- ntext：表示Unicode字符，数据的存储长度是实际字符数的两倍。

二进制型

- 二进制数据类型表示的是位数据流，包括binary（固定长度）和varbinary（可变长度）。
- binary[(n)]：固定长度的n字节二进制数据。数据的存储长度为n+4字节。
- varbinary[(n)]：n字节变长二进制数据。数据的存储长度为实际输入数据长度+4字节。

日期时间型

- 用于存储日期和时间信息。
- date：表示从公元元年1月1日至9999年12月31日的日期，只存储日期数据，不存储时间数据，存储长度为3字节。
- time：只存储时间数据。表示格式为“hh:mm:ss[.nnnnnnn]”，存储大小为5字节。
- datetime：存储日期和时间信息，存储大小随着微秒数的位数（精度）而改变，精度小于3时为6字节，精度为4和5时为7字节，所有其他精度则需要8字节。
- datetimeoffset：存储日期和时间信息，具有偏移量。

创建数据表语法格式

CREATE TABLE [database_name . [schema_name] . | schema_name .] table_name

（

{ < column_definition > /*列的定义*/

| column_name AS computed_column_expression [PERSIRTED [NOT NULL]]

/*定义计算列*/

}

[<table_constraint>][,…n] /*指定表的约束*/

）

[ON { partition_scheme_name (partition_column_name) | filegroup | “default” }]

/*指定分区方案和存储表的文件组*/

[{TEXTIMAGE_ON { filegroup | “default” }] /*指定存储 text、ntext、image 等类型数据的文件组*/

数据库名

新表所属架构名

表名

列名

计算表达式

分区列名

指定存储表的文件组

分区方案名

没有指定或指定为default，包含这些类型的列将与表存储在同一文件组中

列的定义格式

< column_definition > ::=

column_name data_type /*指定列名、类型*/

[FILESTREAM] /*指定 FILESTREAM 属性*/

[COLLATE collation_name] /*指定排序规则*/

[NU /*定是否为空*/

[

[CONSTRAINT constraint_name]

DEFAULT constant_expression] /*指定默认值*/

排序规则名称

表示PRIMARY KEY、UNIQUE、FOREIGN KEY 或 CHECK约束定义的开始

- 以学生管理管理系统数据库为例，创建学生表（XSB）、课程表（KCB）和成绩表（CJB）。

列名	数据类型	长度	是否为空	默认值	说明
Stu_ID	char	6	否	无	学号，主键
Sname	char	8	否	无	姓名
Ssex	bit	默认值	是	1	性别
Sdate	date	默认值	是	无	出生日期
Major	char	12	是	无	专业
Tcredit	int	默认值	是	0	总学分
Remark	varchar	500	是	无	备注

列名	数据类型	长度	是否为空	默认值	说明
C_ID	char	3	否	无	课程号，主键
Cname	char	16	否	无	课程名
Term	tinyint	1	是	1	开课学期
Chour	tinyint	1	是	0	学时
Credit	tinyint	1	否	0	学分

列名	数据类型	长度	是否为空	默认值	说明
Stu_ID	char	6	否	无	学号，主键
C_ID	char	3	否	无	课程号，主键
Grade	int	默认值	是	0	成绩

```
create table PXSCJ..XSB
(
Stu_ID char(6) not null primary key,
Sname char(8) not null,
Ssex bit null ,
Sdate date null,
Major char(12) null,
Tcredit int null,
Remark varchar(500) null
)
```

```
create table PXSCJ..KCB
(
C_ID char(3) not null primary key,
Cname char(16) not null,
Term tinyint null default 1,
Chour tinyint null,
Credit tinyint not null
)
```

```
create table PXSCJ..CJB
(
Stu_ID char(6) not null,
C_ID char(3) not null,
Grade int null
primary key(Stu_ID,C_ID)
)
```

修改表结构语法格式

```
ALTER TABLE [ database_name . [ schema_name ] . | schema_name . ] table_name
{
    ALTER COLUMN column_name                                          /*修改已有列的属性*/
    {
        type_name [ ( { precision [ , scale ]
        [ COLLATE collation_name ]
        [ NULL | NOT NULL ]
    | {ADD | DROP } { ROWGUIDCOL | PERSISTED | NOT FOR REPLICATION | SPARSE }
    }
  | ADD                                                               /*添加列*/
    {
        <column_definition>
      | <computed_column_definition>
      | <table_constraint>
    } [ ,...n ]
  | DROP
    {
        [ CONSTRAINT ] constraint_name                                /*删除约束*/
      [ WITH ( <drop_clustered_constraint_option> [ ,...n ] ) ]
      | COLUMN column_name                                            /*删除列*/
    } [ ,...n ]
```

修改成数值类型时，指定数值的精度和小数位数。

排序规则

列的定义

计算列的定义

表的完整性约束

1、在XSB中增加1个新列“奖学金等级”；

```
alter table PXSCJ..XSB
add 奖学金等级 tinyint null;
```

2、在上题的基础上，删除“奖学金等级”列；

```
alter table PXSCJ..XSB
drop column 奖学金等级;
```

3、修改XSB中已有列的属性：将名为“姓名”的列长度由原来的8改为10,；将名为“出生时间”的列的数据类型由原来的date改为samlldatetime。

```
alter table PXSCJ..XSB
alter column Sname char(10);
alter table PXSCJ..XSB
alter column Sdate smalldatetime;
```

删除数据表语法格式

DROP TABLE [database_name . [schema_name] . | schema_name .]

table_name [,...n] [;]

插入表数据语法格式

```
INSERT    [ TOP ( expression ) [ PERCENT ] ]
      [ INTO ]
      {    table_name                                              /*表名*/
           |view_name                                              /*视图名*/
           |rowset_function_limited
                [ WITH ( <Table_Hint_Limited> [ ...n ] ) ]         /*指定表提示*/
      }
{
      [ ( column_list ) ]                                         /*列列表*/
      [ <OUTPUT Clause> ]                                         /*OUTPUT 子句*/
      { VALUES ( { DEFAULT | NULL | expression } [ ,...n ] ) [ ,...n ]   /*指定列值的 value 子句*/
```

新插入数据的各列的名称

执行插入操作时返回插入的行

练习

- 1、向XSB中插入如下一行数据：081101，王林，1,1990-02-10，计算机，50，null。

```
insert into XSB
values('081101','王林',1,'1990-02-10','计算机',50,NULL);
```

- 2、一次向XSB表中插入两行数据：091101，王海，1,1991-05-10，软件工程，50，null；091102，李娜，0,1991-04-12，软件工程，52，null。

```
insert into XSB
values('091101','王海',1,'1991-05-10','软件工程',50,NULL),
('091102','李娜',0,'1991-04-12','软件工程',52,NULL);
```

删除记录语法格式

```
DELETE    [ TOP ( expression ) [ PERCENT ] ]
[ FROM ]
{     table_name                                                    /*从表中删除数据*/
      |view_name                                                   /*从视图删除数据*/
      |rowset_function_limited
      [ WITH ( <table_hint_limited> [ ...n ] ) ]                   /*指定表提示*/
 }
      [ <OUTPUT Clause> ]                                          /*OUTPUT 子句*/
      [ FROM <table_source> [ ,...n ] ]                            /*从 table_source 删除数据*/
      [ WHERE { <search_condition>                                 /*指定条件*/
                | { [ CURRENT OF    { { [ GLOBAL ] cursor_name }    | cursor_variable_name}]}
                }                                                  /*有关游标的说明*/
      ]
```

执行插入操作时返回插入的行

练习

- 1、将pxscj数据库的xsb表中总学分大于52的行删除。

```
delete from XSB
where Tcredit>25;
```

- 2、将pxscj数据库的xsb表中备注为空的行删除。

```
delete from XSB
where Remark IS NULL;
```

删除指定表中所有的行

TRUNCATE TABLE tb_name

修改表数据的语法格式

```
UPDATE    [ TOP ( expression ) [ PERCENT ] ]
    { table_name                                                          /*修改表数据*/
    |view_name                                                            /*修改视图数据*/
    | rowset_function_limited
       [ WITH ( <Table_Hint_Limited> [ ...n ] ) ]                         /*指定目标表允许的一个或多个表提示*/
     }
SET                                                                       /*赋予新值*/
          { column_name = { expression | DEFAULT | NULL }                 /*为列重新指定值*/
             | column_name { .WRITE ( expression , @Offset , @Length ) }  /*指定修改列的一部分*/
             | @variable = expression                                     /*指定变量的新值*/
             | @variable = column = expression                            /*指定列和变量的新值*/
             | column_name { += | -= | *= | /= | %= | &= | ^= | |= } expression
             | @variable { += | -= | *= | /= | %= | &= | ^= | |= } expression
             | @variable = column { += | -= | *= | /= | %= | &= | ^= | |= } expression
          } [ ,...n ]

     [ <OUTPUT Clause> ]                                                  /*返回更新后的数据*/
     [ FROM{ <table_source> } [ ,...n ] ]                                 /*指定为更新操作提供表*/
     [ WHERE { <search_condition>                                         /*指定条件*/
               | { [ CURRENT OF                                           /*有关游标的说明*/
                     { { [ GLOBAL ] cursor_name } | cursor_variable_name } ]}
               }
     ]
     [ OPTION ( <query_hint> [ ,...n ] ) ]                                /*使用优化程序*/
[ ; ]
```

column_name 值中的起点。

列中某个部分的长度

复合赋值运算符

- 将xsb表中所有学生的总学分都增加10。将姓名为“罗琳琳”的同学的专业改为“软件工程”，备注改为“提前修完学分”，学号改为081261。

```
USE PXSCJ
GO
UPDATE XSB SET Tcredit=Tcredit+10;
UPDATE XSB SET Major='软件工程',Remark='提前修完学分
',Stu_ID='081261'
WHERE Sname='罗琳琳';
```

- **试一试**：将xsb表中姓名为“张林”的学生的出生日期由“1985-2-15”改为“1985-2-20”。（修改列的一部分）

几个问题

- 修改表结构时，关于ALTER COLUMN、ADD、DROP COLUMN，能否同时修改多列？能否同时增加多列？能否同时删除多列？

```
ALTER TABLE PXSCJ..XSB
ADD 说明 tinyint null,
ADD 等级 tinyint null;
```

```
ALTER TABLE PXSCJ..XSB
DROP COLUMN 说明,
DROP COLUMN 等级;
```

```
ALTER TABLE PXSCJ..XSB
ALTER COLUMN Sname char(10),
ALTER COLUMN Sdate smalldatetime;
```

```
ALTER TABLE PXSCJ..XSB
ADD 说明 tinyint null,
DROP COLUMN Ssex,
ALTER COLUMN Sname char(10);
```

几个问题

- 插入记录时，“VALUSE”中的数值个数是否一定要和“COLUMN_LIST列列表”中指定的列名个数一致？插入空值能否省略？

```
INSERT INTO XSB(Stu_ID,Sname,Ssex)
VALUES('081170','王二');
```

```
INSERT INTO XSB
VALUES ('081177','王二',1,'1990-02-10','计算机','','补考');
```

```
INSERT INTO XSB
VALUES ('081177','王二',1,'1990-02-10','计算机',NULL,'补考');
```