国家“十一五”规划教材

数据库原理与应用教程(第4版)

机械工业出版社
China Machine Press

第3章 SQL语言基础及数据定义功能

- 3.1 基本概念
- 3.2 SQL的数据类型
- 3.3 数据定义功能
- 3.4 数据完整性

3.1 基本概念

- 3.1.1 SQL语言的发展
- 3.1.2 SQL语言的特点
- 3.1.3 SQL语言功能概述

3.1.1 SQL语言的发展

- 1986年10月由美国ANSI公布最早的SQL标准。
- 1989年4月，ISO提出了具备完整性特征的SQL，称为SQL-89。
- 1992年11月，ISO又公布了新的SQL标准，称为SQL-92（以上均为关系形式）。
- 1999年颁布SQL-99，是SQL92的扩展。

3.1.2 SQL语言的特点

- 1. 一体化
- 2. 高度非过程化
- 3. 简洁
- 4. 使用方式多样

3.1.3 SQL语言功能概述

四部分：数据定义功能、数据控制功能、数据查询功能和数据操纵功能。

SQL功能	命令动词
数据查询	SELECT
数据定义	CREATE、DROP、ALTER
数据操纵	INSERT、UPDATE、DELETE
数据控制	GRANT、REVOKE

3.2 SQL Server提供的数据类型

数据类型类别名	数据类型类别名
精确数字类型	Unicode字符串类型
近似数字类型	二进制字符串类型
日期和时间类型	其他数据类型
字符串类型	

准确数值类型

精确数值类型	说明	存储空间
bigint	存储从-2^{63} (−9,223,372,036,854,775,808)到$2^{63}-1$ (9,223,372,036,854,775,807) 范围的整数	8字节
int	存储从-2^{31} (−2,147,483,648) 到 $2^{31}-1$ (2,147,483,647) 范围的整数。	4字节
smallint	存储从-2^{15} (−32,768) 到 $2^{15}-1$ (32,767) 范围的整数	2字节
tinyint	存储从0 到 255 之间的整数。	1字节
bit	存储1或0。如果一个表中有<=8个的bit列，则这些列公用一个字节存储	1字节
numeric(p,s) 或 decimal(p,s)	定点精度和小数位数。使用最大精度时，有效值从$-10^{38}+1$ 到$10^{38}-1$。其中，s为精度，指定小数点左边和右边可以存储的十进制数字的最大个数。精度必须是从1到最大精度之间的值。最大精度为38。s为小数位数，指定小数点右边可以存储的十进制数字的最大个数，0 <= s <= p。s的默认值为0	最多17字节

近似数值数类型

近似数值类型	说明	存储空间
float[(n)]	存储从-1.79E + 308至-2.23E –308、0以及2.23E–308至1.79E + 308范围的浮点数。n有两个值，如果指定的n在1~24之间，则使用24，占用4字节空间；如果指定的n在25~53之间，则使用53，占用8字节空间。若省略(n)，则默认为53	4字节或8字节
real	存储从–3.40E + 38到3.40E + 38范围的浮点型数	4字节

字符串类型

- 非unicode字符串类型
- unicode字符串类型
- 二进制字符串类型

非unicode字符串类型

数据类型	说明	存储空间
char[(n)]	固定长度，非Unicode 字符串数据。n 用于定义字符串长度，取值范围为1 到 8000。 char 的 ISO 同义词为 character	n字节
varchar[(n|max)]	可变长度，非Unicode 字符串数据。n 用于定义字符串长度，取值范围为1 到 8,000。 max 指示最大存储大小是 2^{31}-1 个字节（2GB）。 varchar 的 ISO 同义词为char varying 或 character varying	n+2字节
text	可存储$2^{31}-1$ （2,147,483,647） 个长度可变的非Unicode 字符数据	

unicode字符串类型

数据类型	说明	存储空间
nchar[(n)]	固定长度的Unicode 字符串数据。n 用于定义字符串长度，取值范围为 1 到 4000。 nchar 的ISO同义词为 national char 和national character	2*n 字节
nvarchar[(n|max)]	可变长度的Unicode 字符串数据。n 用于定义字符串长度，取值范围为 为1 到 4000。max 指示最大存储大小是 2^{31}-1 个字节 (2GB)。nvarchar 的 ISO 同义词为 national char varying 和 national character varying	2*n + 2 个字节
ntext	长度可变的Unicode字符串数据，字符串最大长度为 2^{30}-1 (1,073,741,823) 个字节。ntext 的ISO 同义词为 national text	所输入字符串长度的两倍（以字节为单位）

二进制字符串类型

数据类型	说明	存储空间
binary[(n)]	固定长度为n 字节的二进制数据，n 的取值从1 到 8000	n 字节
varbinary[(n|max)]	可变长度二进制数据。n 的取值从1 到 8000，max指示最大存储大小为 2^{31}-1 字节。varbinary 的 ANSI SQL 同义词为 binary varying	所输入数据的实际长度 + 2 个字节
image	长度可变的二进制数据，二进制数据最大长度为2^{31}-1 (2,147,483,647) 个字节	0 到 2^{31}-1 个字节

日 期时间 类型

日期时间类型	说明	存储空间
date	定义一个日期，范围为 到 。字符长度10位，默认格式为：YYYY-MM-DD。YYYY表示4位年份数字，范围从0001到9999；MM表示2位月份数字，范围从01到12；DD表示2位日的数字，范围从01 到 31（最大值取决于具体月份）	3字节
time[(n)]	定义一天中的某个时间，该时间基于24小时制。默认格式为：hh:mm:ss[.nnnnnnn]，范围为00:00:00.0000000到23:59:59.9999999。精确到100纳秒。 n为秒的小数位数，取值范围是0到7的整数。默认秒的小数位数是7(100ns)	3~5字节
datetime	定义一个采用24小时制并带有秒的小数部分的日期和时间，范围为到-，时间范围是00:00:00到23:59:59.997。默认格式为：YYYY-MM-DD hh:mm:ss.nnn，n为数字，表示秒的小数部分（精确到0.00333 秒）	8字节

日期时间类型(续)

日期时间类型	说明	存储空间
smalldatetime	定义一个采用24小时制并且秒始终为零(:00)的日期和时间，范围为到。默认格式为：YYYY-MM-DD hh:mm:00。精确到1分钟	4字节
datetime2	定义一个结合了**24**小时制时间的日期。可将该类型看成是**datetime**类型的扩展，其数据范围更大，默认的小数精度更高，并具有可选的用户定义的精度。默认格式是：**YYYY-MM-DD hh:mm:ss[.nnnnnnn]**，**n**为数字，表示秒的小数位数（最多精确到**100** 纳秒），默认精度是**7**位小数。该类型的字符串长度最少**19**位（**YYYY-MM-DD hh:mm:ss**），最多**27**位**(YYYY-MM-DD hh:mm:ss.0000000)**	6~8字节
datetimeoffset	定义一个与采用 24 小时制并与可识别时区的一日内时间相组合的日期，该数据类型使用户存储的日期和时间（24小时制）是时区一致的。语法格式为：datetimeoffset [(n)]，n为秒的精度，最大为7。默认格式为：YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh1:mm1]，其中hh1的取值范围为-14 到 +14，mm1的取值范围为00 到 59。该类型的日期范围为到，时间范围为00:00:00 到23:59:59.9999999。时区偏移量范围为-14:00 到+14:00。该类型的字符串长度为：最少26位(YYYY-MM-DD hh:mm:ss {+|-}hh:mm)，最多34位 (YYYY-MM-DD hh:mm:ss.nnnnnnn {+|-}hh:mm)	8~10字节

3.3 数据定义功能

- 3.3.1 基本表的定义与删除
- 3.3.2 修改表结构

3.3.1 基本表的定义与删除

- 1．定义基本表

使用SQL语言中的CREATE TABLE语句实现，其一般格式为：

CREATE TABLE <表名>（
<列名> <数据类型> [列级完整性约束定义]
{, <列名> <数据类型>
[列级完整性约束定义] …}
[, 表级完整性约束定义] ）

在列级完整性约束定义处可以定义的约束

- NOT NULL：限制列取值非空。
- DEFAULT：给定列的默认值。
- UNIQUE：限制列取值不重。
- CHECK：限制列的取值范围。
- PRIMARY KEY：指定本列为主码。
- FOREIGN KEY：定义本列为引用其他表的外码。

(foreign:外国的)使用形式为：

［FOREIGN KEY(<外码列名>)］
REFERENCES <外表名>(<外表列名>)

几点说明

- NOT NULL和DEFAULT只能是列级完整性约束；
- 其他约束均可在表级完整性约束处定义。
- 注意以下几点：
  - 第一，如果CHECK约束是定义多列之间的取值约束，则只能在表级完整性约束处定义；
  - 第二，如果表的主码由多个列组成，则也只能在表级完整性约束处定义，并将主码列用括号括起来，即：PRIMARY KEY（列1｛[，列2] ..｝）；
  - 第三，如果在表级完整性约束处定义外码，则“FOREIGN KEY (<外码列名>)”部分不能省。

约束定义

- ① 列取值非空约束

  <列名> <类型> NOT NULL

  例：sname char(10) NOT NULL

约束定义（续）

- **② 表主码约束**

- 在定义列时定义主码（仅用于单列主码）

  列定义 PRIMARY KEY

  例： SNO char(7) PRIMARY KEY

- 在定义完列时定义主码（用于单列或多列主码）

  PRIMARY KEY （<列名序列>）

  例： PRIMARY KEY(SNO)

  PRIMARY KEY(SNO , CNO)

约束定义（续）

③外码引用约束

- 指明本表外码列引用的表及表中的主码列。

[FOREIGN KEY (<本表列名>)]
REFERENCES <外表名>(<外表主码列名>)

例：

FOREIGN KEY (sno)
　　REFERENCES 学生表(sno)

创建学生表

```
CREATE TABLE Student (
   Sno      char ( 7 )  PRIMARY KEY,
   Sname char ( 10 ) NOT NULL,
   Ssex     char (2),
   Sage     tinyint ,
   Sdept    char (20)
)
```

创建课程表

```
CREATE TABLE Course (
   Cno          char(10)   NOT NULL,
   Cname        char(20)   NOT NULL,
   Ccredit      tinyint ,
   Semester tinyint,
   PRIMARY KEY(Cno)
)
```

创建SC表

```
CREATE TABLE SC (
  Sno      char(7)   NOT NULL,
  Cno      char(10)  NOT NULL,
  Grade    tinyint,
  XKLB     char(4) ,
  PRIMARY KEY ( Sno, Cno ),
  FOREIGN KEY ( Sno )
      REFERENCES  Student ( Sno ),
  FOREIGN KEY ( Cno )
      REFERENCES  Course ( Cno ) )
```

指明本表外码列引用的表及表中的主码列。

删除表

- 当确信不再需要某个表时，可以将其删除
- 删除表时会将与表有关的所有对象一起删掉，包括表中的数据。
- 删除表的语句格式为：

DROP TABLE <表名> { [, <表名>] …}

- 例：删除test表的语句为：

  DROP TABLE test

3.3.2 修改表结构

- 在定义完表之后，如果需求有变化，比如添加列、删除列或修改列定义，可以使用ALTER TABLE语句实现。
- ALTER TABLE语句可以对表添加列、删除列、修改列的定义、定义主码、外码，也可以添加和删除约束。

修改表结构语法

```
ALTER TABLE <表名>
   [ ALTER COLUMN <列名> <新数据类型>]
  | [ ADD [COLUMN] <列名> <数据类型>
  | [ DROP COLUMN <列名> ]
  | [ADD PRIMARY KEY (列名 [, … n ] ) ]
  | [ADD FOREIGN KEY (列名)
     REFERNECES 表名 (列名) ]
```

示例

- 例2．为SC表添加“修课类别”列，此列的定义为：XKLB　char(4)

```
ALTER TABLE SC
    ADD XKLB char(4) NULL
```

示例

- 例3．将新添加的XKLB的类型改为char(6)。

```
ALTER TABLE SC
    ALTER COLUMN XKLB char(6)
```

示例

- 例3.删除Course表的Period列。

```
ALTER TABLE Course
    DROP COLUMN Period
```

3.4 数据完整性

- 3.4.1 完整性约束条件的作用对象
- 3.4.2 实现数据完整性

完整性约束条件的作用对象

- 完整性检查是围绕完整性约束条件进行的，因此，完整性约束条件是完整性控制机制的核心。完整性约束条件的作用对象可以是表、元组和列。
  - 列级约束
  - 元组约束
  - 关系约束

列级约束

- 列级约束主要是对列的类型、取值范围、精度等的约束，具体包括：
- 对数据类型的约束：包括数据类型、长度、精度等。
- 对数据格式的约束：如规定学号的前两位表示学生的入学年份，第三位表示系的编号，第四位表示专业编号，第五位代表班的编号等等。
- 对取值范围的约束：如学生的成绩取值范围为0～100。
- 对空值的约束。

元组约束

- 元组的约束是元组中各个字段之间的联系的约束，
- 如：
  - 开始日期小于结束日期，
  - 职工的最低工资不能低于规定的最低保障金。

关系约束

- 指若干元组之间、关系之间的联系的约束。
- 比如:
  - 学号的取值不能重复也不能取空值，
  - 学生修课表中的学号的取值受学生表中的学号取值的约束

实现数据完整性

- 声**明完整性**
  - 在表定义时声明
  - 使用约束、缺省值(DEFAULT)等
  - 由SQL Server自动加以保证
- 过**程完整性**
  - 在客户端或服务器端用编程语言或工具实现
  - 在Server端用触发器（trigger）来实现

实现约束

- 1. PRIMARY KEY 约束
- 2. UNIQUE 约束
- 3. FOREIGN KEY 约束
- 4. DEFAULT 约束
- 5. CHECK 约束

PRIMARY KEY 约束

- 保证实体完整性
- 每个表有且只有一个PRIMARY KEY 约束
- 格式：

ALTER TABLE 表名

ADD [CONSTAINT 约束名]

PRIMARY KEY （列名 [, …n] ）

示例

- 例: 对雇员表和工作表分别添加主码约束。

```
ALTER TABLE 雇员表
    ADD  CONSTRAINT  PK_EMP
    PRIMARY KEY (雇员编号)

ALTER TABLE 工作表
    ADD  CONSTRAINT  PK_JOB
    PRIMARY KEY (工作编号)
```

UNIQUE 约束

- 确保在非主键列中不输入重复值。
- 应用在客观具有唯一性质的列上，如身份证号、社会保险号等。
- 格式：

```
ALTER TABLE 表名
  ADD [ CONSTRAINT 约束名 ]
  UNIQUE ( <列名> [, ...n] )
```

示例

- 例. 为雇员表的“ 电话号码” 列添加UNI
  约束。

```
ALTER TABLE 雇员表
    ADD  CONSTRAINT  UK_SID
        UNIQUE (电话号码)
```

FOREIGN KEY约束

- 用于建立和加强两个表数据之间的连接的一列或多列
- 格式：

```
ALTER TABLE 表名
    ADD [ CONSTRAINT 约束名 ]
    FOREIGN KEY (<列名>)
      REFERENCES 引用表名 (<列名>)
```

示例

- 例. 为雇员表的工作编号添加外码引用约束，此列引用工作表的工作编号列。

```
ALTER TABLE 雇员
    ADD CONSTRAINT FK_job_id
    FOREIGN KEY (工作编号)
    REFERENCES 工作表 (工作编号)
```

DEFAULT约束

- 当向表中插入数据时，如果没有为定义了 DEFAULT 的列提供值，则是隐式要求为此列使用默认值。
- 一个**Default**只能约束一列。
- 格式：

ALTER TABLE 表名

ADD [CONSTRAINT 约束名]

DEFAULT 默认值 FOR 列名

示例

- 例. 定义雇员表的工资的默认值为1000。

```
ALTER TABLE 雇员
    ADD CONSTRAINT   DF_SALARY
    DEFAULT   1000   FOR 工资
```

CHECK约束

- 通过限制输入到列中的值来强制域的完整性。
- 可定义同表多列之间的约束关系
- 格式：

ALTER TABLE 表名

ADD [CONSTRAINT 约束名]

CHECK （逻辑表达式）

示例1

- 例1. 在雇员表中，添加限制雇员的工资必须大于等于500的约束。

```
ALTER TABLE 雇员
   ADD CONSTRAINT  CHK_Salary
CHECK （ 工资 >= 500 ）
```

示例2

- 例2. 添加限制工资表的最低工资小于等于最高工资的约束。

```
ALTER TABLE 工作
  ADD CONSTRAINT  CHK_Job_Salary
        CHECK （ 最低工资 <= 最高工资 ）
```

综合起来

```
CREATE TABLE 工作(
    工作编号   char(8) PRIMARY KEY,
    最低工资 int ,
    最高工资 int,
    CHECK ( 最低工资 <= 最高工资 ) )
CREATE TABLE 雇员 (
  雇员编号 char(7) PRIMARY KEY,
  雇员名    char(10),
  工作编号 char(8) REFERENCES 工作 ( 工作编号 ),
  工资   int DEFAULT 1000 CHECK ( 工资 >= 500 ),
  电话号码 char(8) not null UNIQUE )
```