第5章 数据表的管理和基本操作

CHAPTER

5.1 T-SQL的基本数据类型

5.1.1 数字类型

在T-SQL中，基本数据类型是数据库设计和操作的基础，它们用于存储和操作数据库中的各种数据。T-SQL中的数字类型如表5-1所示。

表5-1 数字类型

类 型	占用字节	范 围
INT(或INTEGER)	4	−2 147 483 648～2 147 483 647
SMALLINT	2	−32 768～32 767
TINYINT	1	0～255
BIGINT	8	$-2^{63}-1$(9 223 372 036 854 775 807)
DECIMAL(p,s) 或 NUMERIC(p,s)	可变	依赖于精度p和小数位数s，精度p最大为38，小数位数s最大为p
FLOAT	8	大约±1.79E+308(精度约为7位十进制数字)
REAL	4	大约±3.40E+38(精度较低，大约7位小数)

1. 整数类型

(1) INT(或INTEGER)：用于存储−2 147 483 648～2 147 483 647的所有正负整数。每个INT类型的数据占用4B的存储空间。

(2) SMALLINT：用于存储−32 768～32 767的所有正负整数。每个SMALLINT类型的数据占用2B的存储空间。

(3) TINYINT：用于存储0～255的所有正整数。每个TINYINT类型的数据占用1B的存储空间。

(4) BIGINT：用于存储-2^{63}～$2^{63}-1$的所有正负整数。每个BIGINT类型的数据占用8B的存储空间。

2. 精确数值类型

DECIMAL和NUMERIC：这两种数据类型用于存储精确的小数值，可以指定精度和小数位数。DECIMAL和NUMERIC数据类型在功能和存储

方式上几乎完全相同，但 DECIMAL 是更通用的术语。例如，DECIMAL(10,2)表示总共 10 位数字，其中 2 位是小数。

3. 浮点类型

(1) FLOAT：用于存储浮点数值，可以指定精度(1～53 位十进制数字)。FLOAT 数据类型可以写为 FLOAT[n]的形式，其中，n 是可选的精度参数。若未指定 n，则默认为 53 位精度。FLOAT 类型的数据占用 8B 的存储空间。

(2) REAL：是 FLOAT 类型的一种特例，用于存储精度较低的单精度浮点数，大约可以精确到 7 位小数。REAL 类型的数据占用 4B 的存储空间。

在选择数字类型时，应根据数据的实际范围和精度要求来选择合适的数据类型，以节省存储空间并提高效率。对于需要存储大量数据的场景，如财务计算、科学计算等，应优先考虑使用精确数值类型(如 DECIMAL 或 NUMERIC)，以避免浮点运算带来的精度损失。对于整数类型，应根据数据的实际大小来选择合适的数据类型，以避免不必要的存储空间浪费。T-SQL 中的数字类型包括整数类型(INT、SMALLINT、TINYINT、BIGINT)、精确数值类型(DECIMAL、NUMERIC)和浮点类型(FLOAT、REAL)，每种类型都有其特定的存储范围和精度要求，应根据实际需求进行选择。

5.1.2 字符类型

在 T-SQL 中，字符类型用于存储文本数据，如字符串、姓名、地址等。这些类型允许用户定义字段以存储可变长度的文本信息。T-SQL 中常用的几种字符类型如表 5-2 所示。

表 5-2 字符类型

类 型	描 述	最大长度
CHAR(n)	固定长度的字符数据	1～8000 个字符
VARCHAR（n | max)	可变长度的字符数据	1～8000 个字符(n)，或最大 $2^{31}-1$B
TEXT	用于存储大量的文本数据(已弃用，建议使用 VARCHAR(MAX))	最大 $2^{31}-1$B
NCHAR(n)	固定长度的 Unicode 字符数据	1～4000 个字符
NVARCHAR（n | max)	可变长度的 Unicode 字符数据	1～4000 个字符(n)，或最大 $2^{31}-1$B
NTEXT	用于存储大量的 Unicode 文本数据(已弃用，建议使用 NVARCHAR(MAX))	最大 $2^{30}-1$B

1. CHAR(n)

固定长度的字符数据，n 表示最大字符数(范围为 1～8000)。如果存储的字符串长度小于 n，则用空格填充到 n 个字符的长度。

占用字节：n 字符集大小(例如，在 UTF8 中，一个字符可能占用 1～4B，但 SQL Server 中的 CHAR 类型按固定字符集大小计算，通常是一字节对应一个字符)。

2. VARCHAR(n|max)

可变长度的字符数据，n 表示最大字符数(范围为 1～8000)，max 表示最大长度可达到 2^{311}B(大约 2GB)。与 CHAR 不同，VARCHAR 仅存储实际需要的字符数加上一个长度前

缀(通常为 1～2B),因此更节省空间。

占用字节：实际字符数(包括长度前缀)字符集大小。

3. TEXT

用于存储大量的文本数据(最大可达 2^{31}-1 B 或 2GB)。不过,在 SQL Server 的较新版本中,推荐使用 VARCHAR(MAX)代替 TEXT 类型,因为后者提供了更好的兼容性和功能。

占用字节：根据实际存储的文本量而变化。

4. NCHAR(n)

固定长度的 Unicode 字符数据,n 表示最大字符数(范围为 1～4000)。与 CHAR 类似,但使用 Unicode 字符集,因此可以存储全球多语言文本。

占用字节：2n 字节(每个 Unicode 字符占用 2B)。

5. NVARCHAR(n|max)

可变长度的 Unicode 字符数据,n 表示最大字符数(范围为 1～4000),max 表示最大长度可达到 2^{31}-1 B(大约 2GB)。与 VARCHAR 类似,但支持 Unicode 字符集。

占用字节：实际字符数(包括长度前缀 2B)。

6. NTEXT

用于存储大量的 Unicode 文本数据(最大可达 2^{30}-1 B 或 1GB)。同样,在 SQL Server 的较新版本中,推荐使用 NVARCHAR(MAX)代替 NTEXT 类型。

占用字节：根据实际存储的 Unicode 文本量而变化。

5.1.3　时间日期类型

在数据库系统中,时间日期类型用于存储和处理与时间相关的数据。T-SQL 提供了多种时间日期类型,以满足不同的应用需求。常用的时间日期类型如表 5-3 所示。

表 5-3　常用的时间日期类型

时间日期类型	占 用 空 间	示　　例
DATE	4B	2023-04-01
TIME	3～10B(根据精度)	14:30:45.1234567
DATETIME	8B	2023-04-01 14:30:45.123
DATETIME2	6～8B(根据精度)	2023-04-01 14:30:45.123456789
DATETIMEOFFSET	8～10B(根据精度)	2023-04-01 14:30:45.1234567 -07:00
SMALLDATETIME	4B	2023-04-01 14:30:00(注意：实际存储时分钟后的秒和毫秒会被截断)

1. DATE

仅存储日期值(年、月、日),不包含时间信息。

占用 4B 存储空间。

示例：2023-04-01

2. TIME

存储时间值(小时、分钟、秒),可选包含毫秒、微秒或纳秒。

占用空间根据精度而定,通常为 3～10B。

示例：14:30:45.1234567

3. DATETIME

存储日期和时间值，精确到秒（但秒的小数部分仅精确到三位）。

占用 8B 存储空间。

示例：2023-04-01 14:30:45.123

4. DATETIME2

DATETIME 的扩展，提供了更大的日期范围、更高的时间精度（最高可达纳秒级）。

占用空间根据精度而定，但最小为 6B。

示例：2023-04-01 14:30:45.123456789

5. DATETIMEOFFSET

存储日期、时间和时区偏移量，用于处理跨时区的日期和时间。

占用空间根据精度而定，但通常比 DATETIME2 大。

示例：2023-04-01 14:30:45.1234567 -07:00

6. SMALLDATETIME

存储日期和时间值，但时间精度较低（分钟级），且日期范围较小。

占用 4B 存储空间。

示例：2023-04-01 14:30:00

5.1.4 其他类型

在 T-SQL 中，除了数字类型（如 INT、FLOAT、DECIMAL 等）、字符类型（如 CHAR、VARCHAR、NCHAR、NVARCHAR 等）和时间日期类型（如 DATE、TIME、DATETIME、DATETIME2、DATETIMEOFFSET、SMALLDATETIME 等）之外，还存在一些其他重要的数据类型，它们用于满足特定场景下的数据存储需求。

1. 二进制数据类型

BINARY 和 VARBINARY：用于存储二进制数据，如图片、音频或视频文件的字节流。BINARY 是固定长度的，而 VARBINARY 是可变长度的，能够更有效地利用存储空间。

IMAGE（已弃用）：在旧版本的 SQL Server 中用于存储大量二进制数据，但在 SQL Server 2005 及更高版本中，推荐使用 VARBINARY（MAX）替代。

2. 特殊数据类型

UNIQUEIDENTIFIER：用于存储全局唯一标识符（GUID），这是一个 128 位的数字，常用于需要全局唯一性的场合，如数据库记录的 ID。

SQL_VARIANT：一种特殊的数据类型，可以存储除文本、ntext 和 image 之外任何类型的数据。由于其复杂性和性能问题，建议谨慎使用。

3. 货币数据类型

MONEY 和 SMALLMONEY：用于存储货币值，具有固定的精度和范围。MONEY 类型可以存储的金额范围比 SMALLMONEY 大。

4. 位数据类型

BIT：用于存储布尔值（真/假）或开关状态，占用 1B 的存储空间，但可以表示的值只有 0、1 或 NULL。

5.2　数据表管理

5.2.1　创建数据表

表是数据库中存储数据的基本单元，CREATE TABLE 语句则是用于定义并创建表结构的关键语句。CREATE TABLE 语句的基本格式如下。

```
CREATE TABLE table_name
(
    column1_name column1_datatype constraint1,
    column2_name column2_datatype constraint2,
    …
    columnN_name columnN_datatype constraintN,
    PRIMARY KEY(column_name),
    FOREIGN KEY(column_name) REFERENCES other_table(other_column),
    CHECK(condition)
);
```

下面是各个部分的说明。

table_name：要创建的表的名称。

column_name：表中的列名称，即属性名。

column_datatype：列的数据类型，如 INT、VARCHAR、DATE 等。

constraint：列的约束条件，如 NOT NULL、UNIQUE 等。

PRIMARY KEY：指定哪个列（或列的组合）是主键，用于唯一标识表中的每条记录。

FOREIGN KEY：定义外键，用于建立当前表和另一个表之间的关系。

REFERENCES other_table(other_column)：指定外键引用的另一个表和该表中的列。

CHECK(condition)：检查约束，用于确保列中的值满足特定条件。

例如，在学生管理系统中，需要创建一个存储学生信息的表，可以使用以下语句。

```
CREATE TABLE student
(
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    gender ENUM('男', '女') NOT NULL,
    age INT,
    department VARCHAR(50),
    major VARCHAR(50),
    class VARCHAR(50)
);
```

（1）CREATE TABLE student：这部分是 SQL 语句的开头，指示数据库系统创建一个新的表，表名为 student。

（2）id INT NOT NULL PRIMARY KEY：定义了一个名为 id 的列，表示学生的学号，数据类型为整数（INT）。NOT NULL 表示学号不允许为空，并且此列被指定为主键（PRIMARY KEY），意味着每个学生的 id 必须是唯一的，并且在表中不会有重复的 id 值。在数据库中，主键通常用于快速检索和索引记录。

（3）name VARCHAR(50) NOT NULL：定义了一个名为 name 的列，数据类型为最大长度为 50 个字符的可变字符（VARCHAR），不允许为空，用于存储学生的姓名。

（4）gender ENUM('男','女') NOT NULL：定义了一个名为 gender 的列，使用枚举类型(ENUM)，其值限制为'男'或'女'，不允许为空，用于存储学生的性别信息。

（5）age INT：定义了一个名为 age 的列，数据类型为整数(INT)，用于存储学生的年龄。

（6）department VARCHAR(50)：定义了一个名为 department 的列，数据类型为最大长度为 50 个字符的可变字符(VARCHAR)，用于存储学生所属的院系名称。

（7）major VARCHAR(50)：定义了一个名为 major 的列，同样使用 VARCHAR(50)类型，用于存储学生所学的专业名称。

（8）class VARCHAR(50)：定义了一个名为 class 的列，也是 VARCHAR(50)类型，用于存储学生的班级信息。

根据上述表结构的创建，通过 CREATE TABLE 语句创建的学生表模式如表 5-4 所示。

表 5-4 学生表模式

序号	列 名 称	数 据 类 型	约 束 条 件	描述或用途
1	id	INT	NOT NULL，PRIMARY KEY	学生的学号，不允许为空，唯一标识
2	name	VARCHAR(50)	NOT NULL	学生的姓名，不允许为空
3	gender	ENUM('男','女')	NOT NULL	学生的性别，限制为'男'或'女'，不允许为空
4	age	INT	—	学生的年龄
5	department	VARCHAR(50)	—	学生所属的院系名称
6	major	VARCHAR(50)	—	学生所学的专业名称
7	class	VARCHAR(50)	—	学生的班级信息

表格中的“—”表示该列没有特定的约束条件。“PRIMARY KEY”表示该列是表的主键，用于唯一标识表中的每条记录。“NOT NULL”表示该列不允许为空值。“ENUM('男','女')”是一个特定的数据类型，用于限制列的值只能是指定的枚举值之一。

5.2.2 数据表的约束

1. 约束的概念

约束(Constraint)是定义在数据表上的一种强制规则，用于限制表中数据的类型、格式或值。当为某个表定义约束后，对该表进行的所有 SQL 操作都必须满足这些约束条件，否则操作将失败。

2. 约束的类型

（1）非空约束(NOT NULL)。

强制字段在插入或更新记录时不能为空值。

关键字：NOT NULL。

示例：在创建学生信息表时，“学号”和“姓名”字段通常会被设置为非空约束。

(2) 唯一性约束(UNIQUE)。

保证表中的每一行在该字段上的值都是唯一的。允许字段值为 NULL,但最多只能有一个 NULL 值。

关键字:UNIQUE。

示例:在教师信息表中,教师的“工号”通常会被设置为唯一性约束。

(3) 主键约束(PRIMARY KEY)。

用于唯一标识表中的每一行记录。主键的值不能重复,且不能为空。一个表只能有一个主键,但主键可以由一个或多个字段组成(称为复合主键)。

关键字:PRIMARY KEY。

示例:在学生信息表中,“学号”通常会被设置为主键。

(4) 外键约束(FOREIGN KEY)。

用于维护两个表之间的数据一致性。一个表的外键字段必须参照另一个表的主键字段。

关键字:FOREIGN KEY。

示例:在学生信息表中,学生所在的“院系”字段可能会作为外键,参照院系信息表的主键字段。

(5) 默认约束(DEFAULT)。

为字段指定一个默认值,当插入记录时没有为该字段提供值时,将自动使用默认值。

关键字:DEFAULT。

示例:在插入记录时,如果没有填写某个字段,可以将其默认设置为默认的值。

3. 约束的添加与删除

添加约束:可以在创建表时通过 CREATE TABLE 语句直接添加约束,也可以在表创建后通过 ALTER TABLE 语句添加约束。

删除约束:对于已经添加的约束,如果需要删除,可以使用 ALTER TABLE 语句配合 DROP CONSTRAINT 来实现。

4. 约束的重要性

1) 确保数据准确性

数据表约束通过设定一系列规则,对数据的输入和修改进行严格的限制,从而确保数据在存储和传输过程中的准确性。例如,通过设定字段的数据类型和长度限制,可以防止无效或不合理的数据被录入数据库中。通过设定唯一性约束、主键约束等,可以保证数据的唯一性和标识性,避免数据重复或混淆。

2) 维护数据一致性

数据表约束能够有效地维护数据之间的一致性关系。在数据库中,数据往往不是孤立存在的,而是相互关联、相互影响的。通过设定外键约束、检查约束等,可以确保数据表之间的关联关系正确无误,防止数据不一致的情况发生。例如,通过外键约束,可以确保一个表中的记录在另一个表中存在相应的关联记录,从而维护表间数据的一致性。

3) 提高数据可靠性

数据表约束还能够提高数据的可靠性。通过设定非空约束、默认值约束等,可以确保数据在录入时不会遗漏重要信息,或者自动填充默认值以避免数据缺失。这些措施能够有效

地提高数据的完整性和可用性，为数据分析和决策支持提供可靠的基础。

4）简化数据管理

数据表约束的设定还能够简化数据管理的过程。通过约束，可以自动地执行数据校验和错误处理，减少人工干预和错误发生的可能性。约束还能够为数据库管理员提供清晰的规则说明，便于他们进行数据管理和维护。

5）适应业务需求变化

随着业务需求的变化，数据表的结构和约束也需要进行相应的调整和优化。通过设定灵活的数据表约束，可以轻松地适应这些变化，确保数据库系统能够持续地为业务提供高效、可靠的数据支持。

5.2.3 修改数据表

在数据库管理中，随着业务需求的变化，对数据表进行修改是一项常见的任务。修改数据表涉及对表结构的调整，包括添加、删除或修改列（字段），以及调整表之间的关系等。

1. 添加列（字段）

在数据表的使用过程中，可能会发现需要存储新的信息，这时就需要向表中添加新的列。在 T-SQL 中，可以使用 ALTER TABLE 语句配合 ADD 子句来添加新列。格式如下。

```
ALTER TABLE 表名
ADD 列名 数据类型 [约束条件];
```

表名是要修改的数据表的名称，列名是新添加的列的名称，数据类型指定了该列的数据类型，约束条件是可选的，用于对新列的数据进行限制。

在学生表中，若要使用 T-SQL 的 ALTER TABLE 语句配合 ADD 子句来添加新列，可以根据具体的需求来构造相应的 SQL 语句。以下是一些如何在学生表（student）中添加不同类型列的举例。

1）添加入学日期列

假设需要记录学生的入学日期，可以在学生表中添加一个名为 entrance_date 的列，数据类型为 DATE。

```
ALTER TABLE student
ADD entrance_date DATE;
```

2）添加联系电话列

假设需要记录学生的联系电话，可以在学生表中添加一个名为 phone_number 的列，数据类型为 VARCHAR(20)，并可以添加一些约束条件，如不允许为空。

```
ALTER TABLE student
ADD phone_number VARCHAR(20) NOT NULL;
```

3）添加邮箱地址列

假设需要记录学生的邮箱地址，并且希望邮箱地址是唯一的（即每个学生只能有一个唯一的邮箱地址），可以在学生表中添加一个名为 email 的列，数据类型为 VARCHAR(100)，并添加唯一性约束。

```
ALTER TABLE student
ADD email VARCHAR(100) UNIQUE;
```

4）添加备注信息列

假设需要为学生添加一些备注信息，如特别说明或者兴趣爱好等，可以在学生表中添加一个名为 remarks 的列，数据类型为 TEXT。

```
ALTER TABLE student
ADD remarks TEXT;
```

5）注意事项

数据类型选择：在添加新列时，要根据实际需求选择合适的数据类型。例如，日期类型选择 DATE 或 DATETIME，字符串类型选择 VARCHAR 并指定合适的长度，等等。

约束条件：根据业务需求，可以为新列添加约束条件，如 NOT NULL（不允许为空）、UNIQUE（唯一性约束）、DEFAULT（默认值）等。

表名与列名：确保表名和列名在数据库中唯一，且符合命名规范。

操作权限：执行 ALTER TABLE 语句需要相应的数据库操作权限。

通过上述例子，可以看到如何使用 T-SQL 语句在学生表中添加新列，以满足业务需求的变化。

2. 删除列（字段）

如果某个列不再需要存储信息，或者为了优化表结构，可以将其从表中删除。在 T-SQL 中，使用 ALTER TABLE 语句配合 DROP COLUMN 子句可以删除指定的列。格式如下。

```
ALTER TABLE 表名
DROP COLUMN 列名;
```

假设想要在学生表（student）中删除一些不再需要的列，如 department 或 class 列，可以使用 ALTER TABLE 语句配合 DROP COLUMN 子句来完成这个操作。例如：

1）删除 department 列

```
ALTER TABLE student
DROP COLUMN department;
```

执行上述语句后，student 表将不再包含 department 列。

2）删除 class 列

```
ALTER TABLE student
DROP COLUMN class;
```

删除这两列后，student 表的结构将变为如表 5-5 所示。

表 5-5　学生表模式

序号	列　名　称	数 据 类 型	约 束 条 件	描述或用途
1	id	INT	NOT NULL，PRIMARY KEY	学生的学号，不允许为空，唯一标识
2	name	VARCHAR(50)	NOT NULL	学生的姓名，不允许为空
3	gender	ENUM('男','女')	NOT NULL	学生的性别，限制为'男'或'女'，不允许为空
4	age	INT	—	学生的年龄
5	major	VARCHAR(50)	—	学生所学的专业名称

删除列是一个不可逆的操作，一旦删除，该列中的所有数据也将丢失，因此在执行此操作之前，请确保已经备份了相关数据或确认不再需要这些数据。

3. 修改列（字段）

有时可能需要修改现有列的数据类型或约束条件，如将某个列的长度增加、修改默认值或添加新的约束等。在 T-SQL 中，可以使用 ALTER TABLE 语句配合 ALTER COLUMN 子句来实现这一操作。格式如下。

```
ALTER TABLE 表名
ALTER COLUMN 列名 新的数据类型 [新的约束条件];
```

在 T-SQL 中，若想修改现有列的数据类型或约束条件，通常会用到 ALTER TABLE 语句，并搭配 ALTER COLUMN 子句。以下是一些修改学生表（student）中列的示例。

1）修改列的数据类型

假设想要将 name 列的数据类型从 VARCHAR(50)更改为 VARCHAR(100)，以适应更长的名字，可以使用以下语句。

```
ALTER TABLE student
ALTER COLUMN name VARCHAR(100) NOT NULL;
```

2）修改列的约束条件

如果希望 age 列不能为空，并为其设定一个默认值，如 18 岁，可以使用以下语句。

```
ALTER TABLE student
ALTER COLUMN age INT NOT NULL DEFAULT 18;
```

如果 age 列中已存在 NULL 值，而又试图将其设置为 NOT NULL，此操作将会失败。除非先为这些 NULL 值提供一个有效的默认值，或者显式地更新它们为非 NULL 值。

3）同时修改数据类型和约束条件

如果想同时更改 major 列的数据类型和添加一个新的约束条件（例如，不允许为空），可以使用以下语句。

```
ALTER TABLE student
ALTER COLUMN major VARCHAR(100) NOT NULL;
```

在这个例子中，major 列的数据类型从 VARCHAR(50)变为 VARCHAR(100)，并且被添加了一个 NOT NULL 约束。

4）注意事项

数据类型兼容性：在更改数据类型时，必须确保新类型与旧类型兼容，或者原列中的数据能够安全地转换到新类型中。

约束条件的影响：在添加新的约束条件时，必须确保原列中的数据满足这些条件，否则操作将会失败。

备份数据：在进行任何结构性更改之前，始终建议备份数据，以防万一操作失败导致数据丢失。

操作权限：执行 ALTER TABLE 语句需要相应的数据库操作权限。

通过上述示例，可以看到如何使用 T-SQL 语句来修改学生表中的列，以适应业务需求的变化。不是所有的数据类型更改都是允许的，如将文本类型直接更改为数字类型可能会导致数据丢失或转换错误。因此，在进行此类操作时，应先确保数据可以安全地进行转换，