第 3 章　创建和管理表

表是存储数据的容器，数据按照合理的存储方式存放在表中，有利于数据的查询、修改、删除和增加等操作。本章介绍表的创建和管理，包括表的构成、表的类型、Oracle 中的数据类型、如何创建表、如何修改表和列，以及表的完整性维护等。

本章学习要点：

- ❑ 了解表的构成
- ❑ 理解表的类型
- ❑ 掌握 Oracle 中的数据类型
- ❑ 掌握表的创建
- ❑ 理解虚拟列的功能和使用
- ❑ 了解不可见列的使用
- ❑ 掌握表和列的修改
- ❑ 掌握表和列的删除
- ❑ 理解表的完整性和约束
- ❑ 掌握常用的几种约束
- ❑ 掌握约束的禁止和激活

3.1 表和列

Oracle 中的表与生活中的表很像，有着表头和数据。不过数据库中的表，表头被定义为字段，是表的列；每一行存储一条记录。本节介绍表和列的基础知识。

3.1.1 表的构成

表是数据库最基本的逻辑结构，一切数据都存放在表中，一个 Oracle 数据库就是由若干个数据表组成。其他数据库对象都是为了用户很好地操作表中的数据。表是关系模型中反映实体与属性关系的二维表格，它由列和行组成，通过行与列的关系，表达出了实体与属性的关系，常见的数据库对象有 5 种，如表 3-1 所示。

表 3-1　常见的数据库对象

对象	描述
表	基本的数据存储集合，由行和列组成
视图	从表中抽出的逻辑上相关的数据集合
序列	提供有规律的数值
索引	提高查询效率
同义词	给对象起别名

在 Oracle 数据库中，表是最基本的数据存储结构，表是由行和列组合而成的表格。其中，行表示表中数据记录信息。

列又可以称作字段，每个字段都要设置其类型和长度，可以根据需要设置字段的约束属性。表的创建是 Oracle 数据库最基本，也是不可缺少的操作之一。在创建表的时候，可以为表指定存储空间，还可以对表的存储参数等属性进行设置。

在 Oracle 的数据库操作中，表的创建过程并不难。但是，作为一个合格的数据管理者或者开发者，在创建数据表之前首先必须要确定当前项目需要创建哪些表，表中要包含哪些列，以及这些列所要使用的数据类型等。这就是所谓的表的策略，是需要在创建表之前确定的。一般情况下，创建表所依据的策略主要有以下几个方面。

1．数据库设计理论

在设计表的时候，首先要根据系统需求和数据库分析提取所需要的表，以及每个表所包含的字段。然后根据数据库的特性，对表的结构进行分析设计。表的设计通常要遵循以下几点。

（1）表的类型，如堆表、临时表或者索引等。

（2）表中每个字段的数据类型，如 NUMBER、VARCHAR2 和 DATE 等。

（3）表中字段的数据类型长度大小。

（4）表中每个字段的完整性约束条件，如 PRIMARY KEY、UNIQUE 以及 NOT NULL 约束等。

2．数据表存储位置

在 Oracle 数据库中，需要将表放在表空间（TABLE SPACE）中进行管理，在定义表和表空间时，需要注意以下三点。

（1）设计数据表时，应该设计存放数据表的表空间，不要将表随意分散地创建到不同的表空间中去，这样对以后数据库的管理和维护将增加难度。

（2）如果将表创建在特定的表空间上，用户必须在表空间中具有相应的系统权限信息。

（3）在为表指定表空间的时候，最好不要将表指定在 Oracle 的系统表空间 SYSTEM 中，否则会影响数据库性能。

在创建表的时候，如果不指定特意的表空间，Oracle 会将表建立在用户的默认表空间中。

3．常用的 NOLOGGING 语句

在创建表空间的过程中，为了避免过多的重复记录而指定 NOLOGGING 子句，以节省重做日志文件的存储空间，提高数据库的性能，加快数据表的创建。一般来说，NOLOGGING 适合在创建大表的时候使用。

4．预计和规划表的大小

对索引、回退段和日志文件进行大小估计，从而预计所需磁盘的空间大小。通过这个预计，就可以对硬件和其他方面做出规划。

3.1.2 表的类型

Oracle 中的表有多种类型，不同类型的表有着不同的限制和数据处理方式。本节介绍 Oracle 中表的类型及其应用。

1．堆组织表

堆组织表是普通的标准数据库表，数据以堆的方式管理。堆其实就是一个很大的空间，会以一种随机的方式管理数据，数据会放在合适的地方。

当增加数据时，将使用在段中找到的第一个适合数据大小的空闲空间。当数据从表中删除时，留下的空间允许随后的 INSERT 和 UPDATE 重用。

2．索引组织表

索引组织表（IOT 表）存储在索引结构中，利用行进行物理排序。表中的数据按主键存储和排序，以排序顺序来存储数据。

如果只通过主键访问一个表，就可以考虑使用 IOT 表。父子关系表中，如果是一对多关系，经常根据父表查找子表，子表可以考虑使用 IOT 表。

3．索引聚簇表

聚簇是指一个或多个表的组。有相同聚簇值的行有着相邻的物理存储。Oracle 数据字典大量使用这种表，这样可以将表、字典信息存储在一起，提高访问效率。

如果数据只用于读，需要频繁地把一些表的信息连接在一起访问，可以考虑索引聚簇表。但聚簇会导致 dml、全表扫描的效率低下，而且索引聚簇表是不能分区的。

对于索引聚簇表，来自许多张表的数据可能被存储在同一个块上；包含相同聚簇码值的所有数据将物理上存储在一起。数据聚集在聚簇码值周围，聚簇码用 B 树索引构建。

4．散列聚簇表

散列聚簇表类似索引聚簇表，不使用 B 树索引定位数据，而使用内部函数或者自定义函数进行散列，然后使用这个散列值得到数据在磁盘上的位置。

散列聚簇把码散列到簇中，来到达数据所在的数据库块。在散列聚簇中，数据本身相当于索引。这适合用于经常通过码等式来读取的数据。

散列聚簇是一个高 CPU、低 IO 操作，如果经常按 hashkey 查找数据，可以考虑散列聚簇表。

5．有序散列聚簇表

有序散列聚簇表和索引聚簇表在概念上很相似，主要区别是散列函数代替了聚簇码

索引。有序散列聚簇表同时兼有索引聚簇表、散列聚簇表的一些特性。

有序散列聚簇表中的数据就是索引，却没有物理索引。Oracle 采用行码值，使用内部函数或用户函数对它进行散列运算，利用这些来指定数据应放在硬盘的哪个位置。

使用散列算法来定位数据的副作用是没有在表中增加传统的索引，因此不能区域扫描散列聚簇中的表。

6．嵌套表

嵌套表与传统的父子表模型很相似，但其里面的数据元素是一个无序集，所有数据类型必须相同，很少用嵌套表来存储实体数据，大多数在 PL/SQL 代码中使用。

7．临时表

临时表用来保存事务、会话中间结果集。临时表值对当前会话可见，可以创建基于会话的临时表，也可以创建基于事务的临时表。

创建基于会话的临时表格式如下：

```
CREATE GLOBAL TEMPORARY TABLE 表名(字段列表) ON COMMIT PRESERVE ROWS;
```

创建基于事务的临时表格式如下：

```
CREATE GLOBAL TEMPORARY TABLE 表名(字段列表) ON COMMIT DELETE ROWS;
```


如果应用中需要临时存储一个行集合供其他表处理，可以考虑临时表。

8．对象表

对象表用于实现对象关系模型，很少用来存储数据，可以在 PL/SQL 中用来得到对象关系组件。

9．外部表

外部表可以把一个操作系统文件当作一个只读的数据库表。

3.1.3 数据类型

Oracle 中的数据有多种数据类型，本节将 Oracle 数据类型分为字符类型、数字类型、日期类型和图片类型来介绍。这几种类型下又有多种 Oracle 数据类型来细化地区分数据，如字符类型下有两种类型：VARCHAR2 类型和 CLOB 类型。

1．字符类型

字符类型有定长（固定长度，长度不可变）和变长（长度可变，可设置长度最大值）两种。

VARCHAR2()数据类型是变长类型，括号里面可以定义字段的最大长度，可设置的最大值为 4000。VARCHAR2 字符查询速度相对慢。以该数据类型字段为查询条件，将

会根据数据中的一个字符一个字符进行比较。

CLOB 数据类型是长度最大的字符型数据类型，通常可用于存储新闻之类的大型对象。其最大长度为 4GB。

2．数字型

数字类型使用 NUMBER 关键字来表示，可以表示整数或小数。其范围是-10^{38}～10^{38}，可通过在 NUMBER 后面添加括号和数据的方式定义字段的具体范围。有以下几种表示方法。

（1）直接使用 NUMBER 表示整数或小数。

（2）使用 NUMBER(n)表示一个 n 位的整数，如 NUMBER(6)的数值范围是–999 999～999 999。

（3）使用 NUMBER(m,n)表示小数有 m 位有效数字，包含 n 位小数。如 NUMBER(6,3)表示一个小数有 6 位有效数，其中 3 位小数，数值范围是–999.999～999.999。

3．日期类型

日期类型表示时间，使用 DATE 关键字表示年月日时分秒。

4．图片

图片和视频文件通过二进制数据的方式存储在数据库中，使用 BLOB 二进制数据类型管理。BLOB 数据类型最大为 4GB，这样可以通过数据机制起到安全作用，限制用户访问量。

3.2 创建表

创建表是数据库管理的基础操作，包括表和列的定义，列数据类型的定义、虚拟列和不可见列的使用等。

3.2.1 表和列的命名规则

创建表和列需要用户为表和列命名，Oracle 数据库对表和列的命名有着一定的规则限制。好的命名能够有利于提高表和列的可读性，方便用户使用。本节介绍表和列的命名规则。

Oracle 数据库对表和列的命名要求如下。

（1）必须以字母开头。

（2）长度不能超过 30 字符。

（3）不能使用 Oracle 保留字。

（4）只能使用如下字符：A～Z,a～z,0～9,$,#等。

（5）不能和已经存在的其他对象重名。

另外，为了增加表和列的可读性，通常表以名词或名词短语命名，确定表名是采用复数还是单数形式。为了区分系统的对象，也可对表、视图、列等对象添加不同的前缀来区分，如对表添加 TB 前缀。

3.2.2 使用设计器建表

使用设计器创建新表的步骤比较简单。首先打开 Oracle SQL Developer 并创建对数据库的连接，展开连接，找到表节点右击，选择【新建表】选项即可打开如图 3-1 所示的对话框。

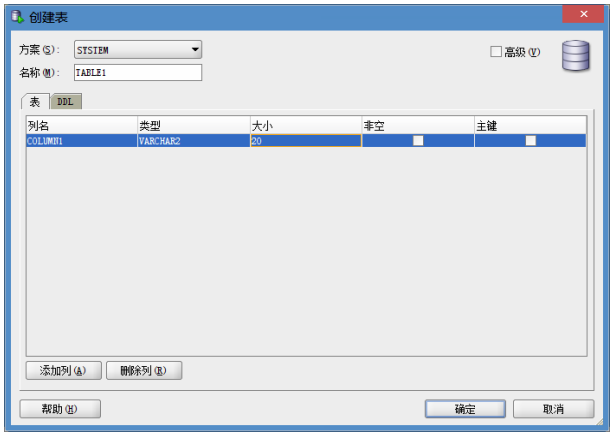

图 3-1 【创建表】对话框

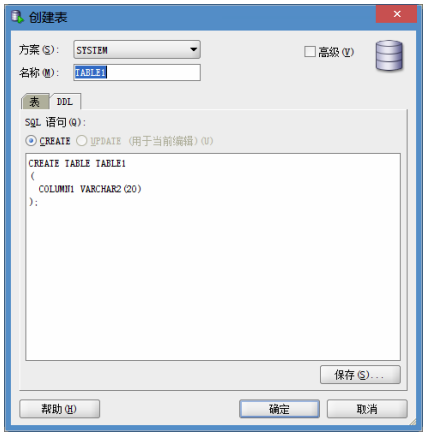

图 3-2 表的定义语句

如图 3-1 所示，可直接修改表的名称，单击右下方的【添加列】按钮可添加新的列，图 3-1 中有默认的一个列，可直接修改列的名称、类型、大小，是否为空和是否为主键。单击图 3-1 中的 DDL 标签可打开表的定义语句，如图 3-2 所示。

为空属性和主键属性是列的约束，将在 3.4 节详细介绍。图 3-1 可创建一个简单的表。若想详细定义列的属性，可选择右上角的【高级】复选框，打开表的高级设置对话框如图 3-3 所示。

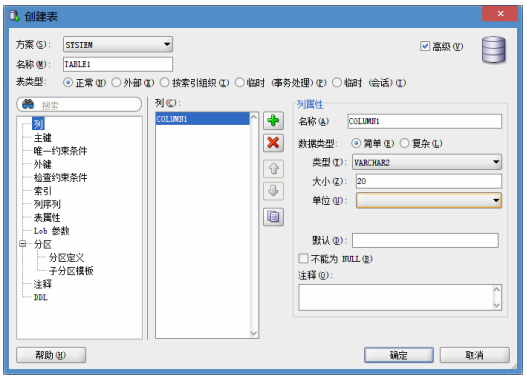

图 3-3 表的高级设置

如图 3-3 所示，在该对话框中可详细设置列的属性，包括主键设置、唯一约束条件设置、外键设置、检查约束条件设置、索引设置、列序列设置、表属性设置等。如选择【表属性】选项，可打开表的属性如图 3-4 所示。单击【存储选项】可打开【表存储选项】对话框，如图 3-5 所示。

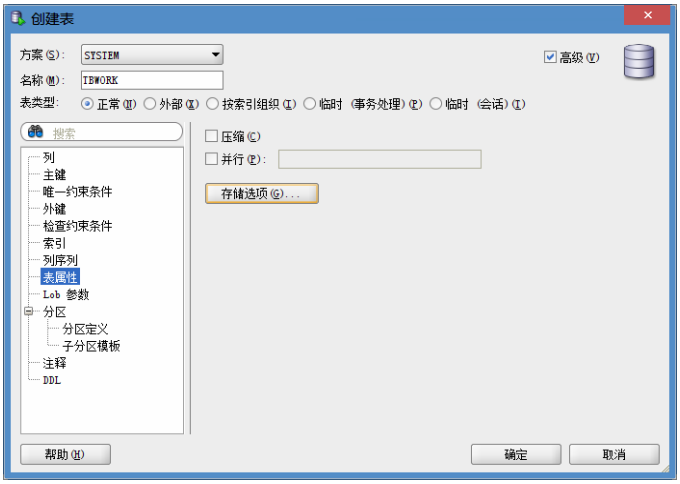

图 3-4 表的属性

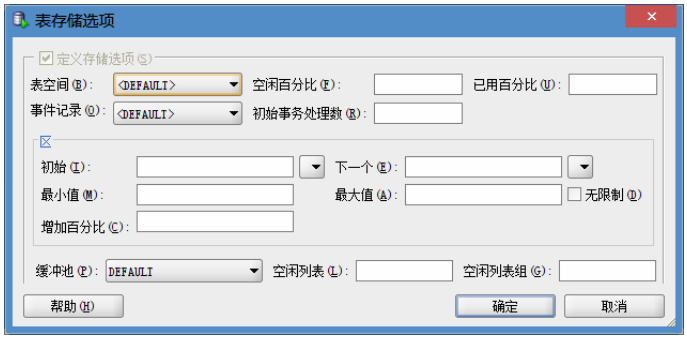

图 3-5 表存储选项

如图 3-5 所示，在该对话框下可设置表的存储选项，如表的初始、最大值、最小值等。设置完成后单击【确定】按钮回到图 3-4 对话框，再单击【确定】按钮回到图 3-3 所示的对话框，单击【确定】按钮即可实现表的创建。

新建表的时候除了可以在表节点下右击，选择【新建表】选项，还可以使用另外两种方式打开【创建表】对话框，如下所示。

（1）在 Oracle SQL Developer 下找到工具栏中的【文件】，单击并选择【新建】选项打开如图 3-6 所示的对话框，选择【表】并单击【确定】按钮打开【选择连接】对话框，如图 3-7 所示。选择表所在的连接，单击【确定】按钮即可打开【创建表】对话框。

（2）在 Oracle SQL Developer 下找到图标并单击，可打开如图 3-6 所示的对话框，使用上述（1）中的步骤可打开【创建表】对话框。

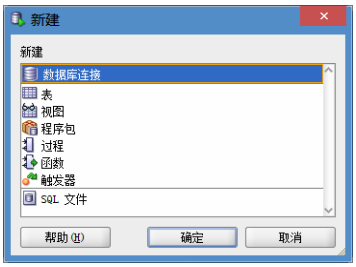

图 3-6 新建对象

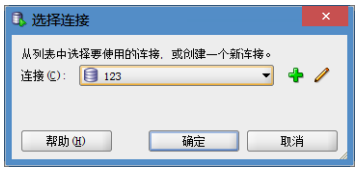

图 3-7 选择连接

注 意

如果用户需要在自己的模式下创建一个新表，必须具有 CREATE TABLE 权限；如果需要在其他用户模式中创建表，则必须具有 CREATE ANY TABLE 的系统权限。

3.2.3 使用 SQL 语句创建表

虽然使用设计器创建表步骤简单，但很多地方不允许使用设计器，此时只能使用 SQL 语句创建表。使用 SQL 语句创建表需要使用 CREATE 关键字，语法格式如下：

```
CREATE TABLE [schema.]table_name(
column_name data_type [DEFAULT expression] [constraint]
[,column_name data_type [DEFAULT expression] [constraint]]
[,column_name data_type [DEFAULT expression] [constraint]]
[,…]
)[TABLESPACE tablespace_name]|[STORAGE (INITIAL nk|M NEXT nk|M
PCTINCREASE n)]|[ CACHE];
```

中括号包括的内容为可选项。其中各个参数含义如下。

（1）schema：指定表所属的用户名或者所属的用户模式名称。

（2）table_name：所要创建的表的名称。

（3）column_name：表中包含的列的名称，列名在一个表中必须具有唯一性。

（4）data_type：列的数据类型。

（5）DEFAULT expression：列的默认值。

（6）constraint：为列添加的约束，表示该列的值必须满足的规则。

（7）TABLESPACE：指定将创建在哪个表空间。

（8）STORAGE：指定存储参数信息。

存储参数信息中各个参数的含义如下所示。

（1）INITIAL：用来指定表中的数据分配的第一个盘区的大小，以 KB 或者 MB 作为单位，默认值是 5 个 Oracle 数据块的大小。

（2）NEXT：用来指定表中的数据分配的第二个盘区的大小。该参数只有在字典管理的表空间中起作用，在本地化管理表空间中，该盘区大小将由 Oracle 自动决定。

（3）PCTINCREASE：用来指定表中的数据分配的第三个以及其后的盘区的大小，同样，在本地化管理表空间中，该参数不起作用。

使用 CACHE 关键字来对缓存块进行换入、换出调度操作，这样在查询已经查询过的数据时就不用再次查询数据库，加快了查询时间。

重做日志用来存储对表的一些操作记录信息。LOGGING 子句将对表的所有操作都记录到重做日志中。

重做日志文件的主要目的是，万一实例或者介质失败，重做日志文件就能派上用场，或者可以作为一种维护备用数据库的方法来完成故障恢复。如果数据库所在主机掉电，导致实例失败，Oracle 会使用在线重做日志将系统恢复到掉电前的那个时刻。如果包含数据文件的磁盘驱动器出现永久性故障，Oracle 会使用归档重做日志以及在线重做日志，将磁盘驱动器的备份恢复到适当的时间点。

在创建表的时候，如果使用 NOLOGGING 子句，则对该表的操作不会保存到日志文件中去。使用这种方式可以节省重做日志文件的存储空间。但是某些情况下将无法使用数据库的恢复操作，从而无法防止数据信息的丢失。

在创建表的时候，如果没有使用 LOGGING 或者 NOLOGGING 子句的时候，则 Oracle 会默认使用 LOGGING 子句。

根据表中要添加的数据长度大小，在创建表的时候选择合适的数据类型以及数据类型精度，能够避免使用最大精度，减少了 Oracle 数据库占用的不必要的资源空间。

【范例 1】

创建有一个主键的图书信息表 TBBOOK，有数字类型表示标号的 B_ID 字段、长度可变的字符串类型表示书名的 B_TITLE 字段、长度可变的字符串类型表示图书类型的 B_TYPE 字段和长度可变的字符串类型表示图书出版社的 B_PUBLISHER 字段，其中将 B_ID 字段设置为主键，代码如下。

```
CREATE TABLE TBBOOK
(
  B_ID NUMBER NOT NULL ,
  B_TITLE VARCHAR2(20) ,
  B_TYPE VARCHAR2(20) ,
  B_PUBLISHER VARCHAR2(20) ,
  CONSTRAINT TBBOOK_PK PRIMARY KEY(B_ID)
  ENABLE
);
```

3.2.4 虚拟列

在 Oracle 11g 中，Oracle 以不可见索引和虚拟列的形式引入了一些增强特性；Oracle 12c 继承前者并发扬光大，引入了不可见列的思想，来隐藏重要的数据列。这里先介绍虚拟列的创建和使用，在 3.2.5 节介绍不可见列的知识。

在老的 Oracle 版本中，需要使用表达式或者一些计算公式来计算列的值时，通常会创建数据库视图；如果需要在这个视图上使用索引，通常会创建基于函数的索引。Oracle

11g 允许用户直接在表上使用虚拟列来存储表达式。

虚拟列的值是不存储在磁盘上的，它们是在查询时根据定义的表达式临时计算的。用户不能往虚拟列中插入数据；也不能隐式地添加数据到虚拟列。虚拟列的数据并没有存储在数据文件中，而是 Oracle 通过列数据的生成放到了数据字典中。

创建表的时候可以创建虚拟列，其完整定义包括列名、数据类型、GENERATED ALWAYS 关键字、AS(列表达式)和 VIRTUAL 关键字，如下所示。

```
列名 数据类型 GENERATED ALWAYS AS (列表达式) VIRTUAL
```

其中，GENERATED ALWAYS 和 VIRTUAL 为可选关键字，主要用于描述虚拟列的特性。如果忽略列的数据类型，那么 Oracle 会根据 AS 后面的表达式最终结果的数据类型来确定虚拟列的数据类型。

虚拟列的数值是通过真实列中的数据计算而来的，虚拟列的位置可以放在它参考的列的前面，也可以包括多个实际列的值，但是不能引用其他的虚拟列。一个表中不能只有虚拟列。

虚拟列可以使用 Oracle 自带的函数，也可以使用用户定义的函数，不过对于用户定义的函数要求必须声明函数的确定性。Oracle 虽然在创建表的时候会检查函数的确定性，在表建立之后，却可以将函数替换为非确定性函数。

建立虚拟列可以有效地减少数据的存储，简化查询语句中对列进行的处理，而且还可以利用虚拟列进行分区。不过虚拟列还会带来其他问题，如下所示。

（1）包含虚拟列的表在 INSERT INTO 语句中不能省略字段列表。

（2）由于虚拟列的值是由其他列的值计算得出的，且 Oracle 并不存储虚拟列的值，因此无论是 INSERT 还是 UPDATE 都不能对虚拟列进行修改。

（3）如果程序选择使用了一些工具来自动生成表的 INSERT、UPDATE 语句，那么遇到包含虚拟列的表时就会报错。

（4）无法使用 CREATE TABLE AS SELECT 创建一个包含虚拟列的表。但可以在创建表之后通过 ALTER TABLE 添加虚拟列。

（5）当虚拟列的值被实体化，那么虚拟列表达式发生变化会造成实体化结果与虚拟列不一致。因为虚拟列的结果是在查询的时候确定的，如果修改了虚拟列的表达式，下次执行查询时，虚拟列的值就会发生变化。

（6）一旦对虚拟列建立了索引，或者对包含虚拟列的表建立了物化视图，那么虚拟列的数值就被实际地存储下来，当虚拟列的表达式发生修改后，会导致索引或物化视图中已有的数据与目前虚拟列结果不一致。

【范例 2】

创建职工信息表，有职工编号 W_ID 字段，职工姓名 W_NAME 字段，职工出生年份 W_BIRTHYEAR 字段和入职年份 W_INTIME 字段，另外有虚拟列 W_INAGE 表示职工入职时的年龄，由职工出生年份和入职年份计算得出。创建职工信息表代码如下。

```
CREATE TABLE TBWORK
(
  W_ID NUMBER,
```

```
  W_NAME VARCHAR2(20),
  W_BIRTHYEAR NUMBER,
  W_INTIME NUMBER,
  W_INAGE NUMBER GENERATED ALWAYS AS (W_INTIME-W_BIRTHYEAR) VIRTUAL
);
```

可向表中添加一条记录，查看虚拟列的效果。添加一条数据，包含职工编号、姓名、出生年份和入职年份，代码如下。

```
INSERT INTO TBWORK(W_ID,W_NAME,W_BIRTHYEAR,W_INTIME) VALUES(1,'何明
',1984,2007);
COMMIT;
```

上述代码执行后查看表中的数据，代码如下。

```
SELECT * FROM TBWORK;
```

上述代码的执行效果如下所示。从下面的执行效果可以看出，虚拟列在查询时被系统默认添加了数据。

```
W_ID   W_NAME     W_BIRTHYEAR   W_INTIME    W_INAGE
----   ---------  -------------  -----      --------
1      何明         1984            2007        23
```


3.2.5 不可见列

在之前的版本中，为了隐藏重要的数据列，用户往往会创建一个视图来隐藏所需信息或应用某些安全条件。在 Oracle 12c 中，用户可以在表中创建不可见列。当一个列定义为不可见时，这一列就不会出现在通用查询中，除非在 SQL 语句或条件中有显式地提及这一列，或是在表定义中有 DESCRIBED。

虚拟列和分区列同样也可以定义为不可见类型。但临时表、外部表和集群表并不支持不可见列。对于不可见列添加数据，必须在 INSERT 语句中显式提及不可见列名，将不可见列插入到数据库中。

列的可见和不可见属性可通过 INVISIBLE（不可见）和 VISIBLE（可见）两个关键字来设置，如在创建表的时候创建不可见列，代码如下。

```
列名 数据类型 INVISIBLE
```

也可使用 ALTER TABLE 语句修改数据列的可见和不可见属性，代码如下。

```
ALTER TABLE 表名 MODIFY (列名 INVISIBLE|VISIBLE);
```

【范例 3】

创建学生成绩表，有学生学号、姓名、语文成绩、数学成绩和总成绩字段，为保护学生隐私，将学生姓名设置为不可见列，在查询成绩时仅显示学生学号和成绩信息。另外，将总成绩字段设置为虚拟列，值为语文成绩和数学成绩的和，代码如下。

```
CREATE TABLE TBREPORT
```

```
(
  R_ID NUMBER NOT NULL ,
  R_NAME VARCHAR2(20) INVISIBLE ,
  R_CHINESE NUMBER ,
  R_MATHS NUMBER ,
  R_TOTAL NUMBER GENERATED ALWAYS AS (R_CHINESE + R_MATHS) VIRTUAL
);
```

向表中添加数据，包含学生编号、姓名、语文成绩和数学成绩，代码如下。

```
INSERT INTO TBREPORT (R_ID,R_NAME,R_ CHINESE, R_MATHS) VALUES(1,'梁红
',88,72);
COMMIT;
```

上述代码执行后查询表中的数据，代码如下。

```
SELECT * FROM TBREPORT;
```

上述代码查询表中的所有数据，其执行效果如下所示。

```
R_ID  R_CHINESE  R_MATHS    R_TOTAL
----  ---------  -------    -------
1     88         72          160
```


由上述执行效果可以看出，默认的查询没有提供学生的姓名信息。使用查询语句查询表中指定列的数据，包括学生姓名、语文成绩、数学成绩和总成绩信息，代码如下：

```
SELECT R_NAME,R_CHINESE,R_MATHS,R_TOTAL FROM TBREPORT;
```

上述代码的执行效果如下所示。

```
R_NAME              R_CHINESE    R_MATHS    R_TOTAL
--------            ----------- --------- ----------
梁红                 88           72          160
```

由上述执行效果可以看出，不可见列只有在指定查询时才能够看到。

3.3 修改表和列

表和列在创建之后是可以修改的，如修改表的名称、修改列的名称、修改列的数据类型、向表中添加新的列、删除表中指定的列等。本节详细介绍表和列的修改。

3.3.1 修改表

修改表的前提是这个表已经存在，可使用 ALTER TABLE 语句修改表。表的修改包括修改表的名称和存储选项，可以使用设计器修改，也可以使用 SQL 语句修改。

1．使用 SQL 语句修改表的名称

修改表的名称可以使用 ALTER TABLE…RENAME TO 关键字，语法格式如下：

```
ALTER TABLE table_name RENAME TO new_table_name;
```

各个参数含义如下。

（1）table_name：要修改的表名。

（2）new_table_name：表示修改之后的表名称。

【范例 4】

将 TBBOOK 表的名称修改为 TBBOOKS，代码如下。

```
ALTER TABLE TBBOOK RENAME TO TBBOOKS;
```

2．使用设计器修改表的名称

使用设计器不只可以修改表的名称，还可修改列的属性。找到需要修改的表的节点，右击并选择【编辑】选项打开【编辑表】对话框，如图 3-8 所示。

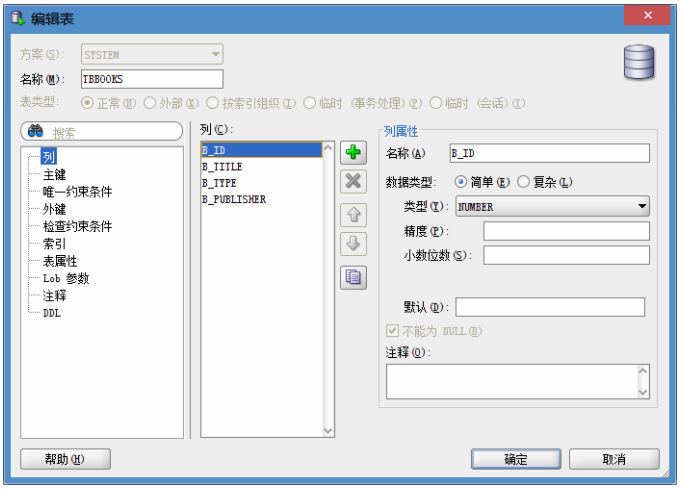

图 3-8 编辑表

如图 3-8 所示，对话框中包括表的名称、列的名称、数据类型和约束等内容，可直接进行修改。在修改完成后单击【确定】按钮即可实现表的修改。

3.3.2 添加列

表在一开始的设计并不是完美的，在后期使用时可能存在添加和删除列的情况。列可以使用设计器添加，也可以使用 SQL 语句添加。

1．使用设计器添加列

使用设计器修改列，在如图 3-8 所示的对话框中，单击按钮即可在【列】列表中添加新的列，如图 3-9 所示。可直接在界面中设置新建列的属性，单击【确定】按钮保存设置，实现列的添加。

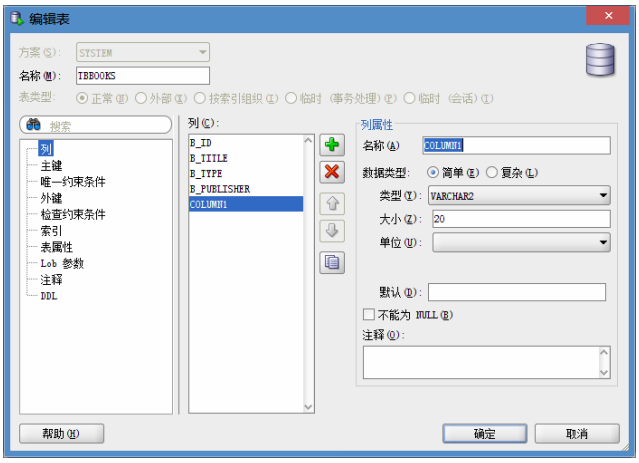

图 3-9 添加列

2. 使用 SQL 语句添加列

添加列使用 ALTER TABLE…ADD 语句，语法如下：

```
ALTER TABLE table_name ADD list_name date_type;
```


各个含义如下。

（1）table_name：指定要修改的表名。

（2）list_name：指定要添加的列名。

（3）date_type：列的数据类型以及大小。

【范例 5】

向 TBBOOKS 表中添加 B_WRITER 列表示图书作者，代码如下。

```
ALTER TABLE TBBOOKS ADD B_WRITER VARCHAR2(20);
```

上述代码执行后，查询表的结构，代码如下。

```
DESC TBBOOKS;
```

上述代码的执行结果如下所示。

```
名称                                  是否为空?      类型
---                                   --------      ---
B_ID                                   NOT NULL     NUMBER
B_TITLE                                             VARCHAR2(20)
B_TYPE                                              VARCHAR2(20)
B_PUBLISHER                                          VARCHAR2(20)
B_WRITER                                            VARCHAR2(20)
```

3.3.3 修改列

修改列包括对列名称的修改、数据类型的修改、数据精度的修改和默认值的修改等，

可以使用设计器修改，也可以使用 SQL 语句修改。

使用设计器修改列的属性，可以参考修改表的方法，在如图 3-8 所示的对话框进行修改，保存修改的方法与表的修改一样。

使用 SQL 语句修改表的属性，不同的 SQL 语法格式进行不同的修改。以下分别介绍列名称的修改、数据类型和精度的修改、默认值的修改格式。

1．修改列名

修改列名称是在更新列操作中经常使用的，修改表中已经存在的列名称语法如下：

```
ALTER TABLE table_name RENAME COLUMN oldcolumn_name TO newcolumn_name;
```

各个参数含义如下。

（1）table_name：表示被修改的列所属的表名称。

（2）oldcolumn_name：表示要修改的列的名称。

（3）newcolumn_name：表示修改之后的列的名称。

【范例 6】

修改 TBBOOK 表中 B_PUBLISHER 列的名称为 B_PUB，代码如下。

```
ALTER TABLE TBBOOKS RENAME COLUMN B_PUBLISHER TO B_PUB;
```


2．修改列数据类型以及数据精度

在修改数据类型的时候，要注意如果表中存在数据，那么修改的数据的长度是不可逆的，也就是说，只能比修改前的长度大，而不能比修改之前的长度小。如果该表中没有数据，则可以将数据的长度由大值修改为小值。语法格式如下：

```
ALTER TABLE table_name MODIFY column_name new_datatype;
```

其中，new_datatype 表示修改之后的数据类型。

3．修改列的默认值

列的默认值就是当对列对象不赋值时所使用的字母或者符号。列的默认值在没有设置的情况下为 NULL。修改默认值的语法如下：

```
ALTER TABLE table_name MODIFY(column_name DEFAULT default_value);
```

其中，各个参数含义如下。

（1）table_name：表示被修改数据列所属的表名称。

（2）column_name：表示要修改的列名。

（3）default_value：表示修改之后的列的默认值。

3.3.4 删除列

删除列可使用设计器，也可以使用 SQL 语句。使用设计器删除列与添加列的步骤一

样，在如图 3-9 所示的对话框中，选中需要删除的列（主键除外），接着单击按钮即可删除指定的列，接着单击【确定】按钮保存表的修改。

使用 SQL 语句删除列要使用 ALTER…DROP 语句，语法如下：

```
ALTER TABLE table_name DROP COLUMN list_name;
```

各个参数的含义如下。

（1）table_name：指定要修改的表名。

（2）list_name：指定要删除的列名。

【范例 7】

删除 TBBOOKS 表中的 B_WRITER 列，代码如下。

```
ALTER TABLE TBBOOKS DROP COLUMN B_WRITER;
```

3.3.5 删除数据表

数据表内存有大量的数据，这些数据长期占用着存储空间。然而生活中，很多表是有一定的期限的，在不再需要时将会成为系统的累赘，因此需要对表进行删除操作。

表的删除包括表的定义和表中的数据，使用 DROP 语句格式如下：


```
DROP TABLE table_name [CASCADE CONSTRAINTS] [PURGE];
```

其中，各个参数含义如下。

（1）table_name：表示要删除的表的名称，是不可缺少的参数。

（2）CASCADE CONSTRAINTS：是可选择参数。表示删除表的同时也删除该表的视图、索引、约束和触发器等。

（3）PURGE：可选择参数，表示表删除成功后释放占用的资源。

若表的定义仍然需要使用，而表中的数据已经过期，那么可以使用 TRUNCATE 关键字清空表中数据，语法如下：

```
TRUNCATE TABLE table_name;
```

TRUNCATE 语句删除表中所有的数据，释放表的存储空间，而且 TRUNCATE 语句不能回滚。

3.4 数据完整性

数据有着多种数据类型用来满足不同的需求。实际应用中的数据也有着一定取值范围，如年龄不小于 0、性别只能是男或女等。表之间的联系使相连接的字段要保持一致和完整。

维护数据完整性归根到底就是要确保数据的准确性和一致性，表内的数据不相矛盾，表之间的数据不相矛盾，关联性不被破坏。本节介绍如何通过约束来维护数据完整性。

3.4.1 约束简介

约束是作用在列的，对列数据进行限制的一种机制，不同的约束可监管列数据的不同方面。如非空约束限制向表中添加数据时，列的值不能为空；唯一约束限制列中的数据不能重复等。

根据约束的作用域，可以将约束分为以下两类。

（1）表级别的约束：定义在一个表中，可以用于表中的多个列。

（2）列级别的约束：对表中的一列进行约束，只能够应用于一个列。

根据约束的用途，可以将约束分为以下 5 类。

（1）PRIMARY KEY：主键约束。

（2）FOREIGN KEY：外键约束。

（3）UNIQUE：唯一性约束。

（4）NOT NULL：非空约束。

（5）CHECK：检查约束。

下面对这些常用约束以及其他类型进行总结说明，如表 3-2 所示。

表 3-2 约束的类型及其使用说明

约束	约束类型	说明
NOT NULL	C	指定一列不允许存储空值。这实际就是一种强制的 CHECK 约束
PRIMAPY KEY	P	指定表的主键。主键由一列或多列组成，唯一标识表中的一行
UNIQUE	U	指定一列或一组只能存储唯一的值
CHECK	C	指定一列或一组列的值必须满足某种条件
FOREIGN KEY	R	指定表的外键，外键引用另外一个表中的一列，在自引用的情况中，则引用本表中的一列

在 Oracle 系统中定义约束时，使用 CONSTRAINT 关键字为约束命名。如果用户没有为约束指定名称，Oracle 将自动为约束建立默认名称。

约束可以在创建表的时候设置，也可在现有表中添加约束，还可以在有约束的表中修改或删除约束。后面几节将详细介绍各类约束的使用。

3.4.2 主键约束

主键约束又称作 PRIMARY KEY 约束，是表中最重要的约束，一个表可以没有其他约束，但一定要有主键；这也是主键列不能够被直接删除的原因。

主键是主关键字，用来限制列的数据唯一且不为空，即这一字段的数据没有重复数据值并且不能有空值。每个表只能有一个主键，一般用来作标识。

表中列的数据大多会有重复，例如描述会员信息的表，会员的用户名、密码、注册时间和会员等级等字段值都会有重复，能确定身份的身份证在大多数网站上也不方便使用。那么如何确定是某一个会员而不和其他会员搞混，这就用到了主键。一个不重复并且不能有空值的列，就可以确定具体是哪一个会员。主键约束具有以下三个特点。

（1）在一个表中，只能定义一个 PRIMARY KEY 约束。

（2）定义为 PRIMARY KEY 的列或者列组合中，不能包含任何重复值，并且不能包含 NULL 值。

（3）Oracle 数据库会自动为具有 PRIMARY KEY 约束的表建立一个唯一索引，以及一个 NOT NULL 约束。

在定义 PRIMARY KEY 约束时，可以在列级别和表级别上分别进行定义，如下所示。

（1）如果主键约束是由一列组成，那么该主键约束被称为列级别上的约束。

（2）如果主键约束定义在两个或两个以上的列上，则该主键约束被称为表级别约束。

PRIMARY KEY 约束既可以在类级别上定义，也可以在表级别定义，但是不允许在两个级别上都进行定义。

主键约束可以在设计器中进行设置，也可以使用 SQL 语句进行设置。可以在创建表的时候设置，也可以在现有表中添加主键。

1．在设计器中设置主键约束

在设计器中设置主键可以使用创建表的方法，在如图 3-3 所示的高级设置对话框中选择【主键】，如图 3-10 所示。

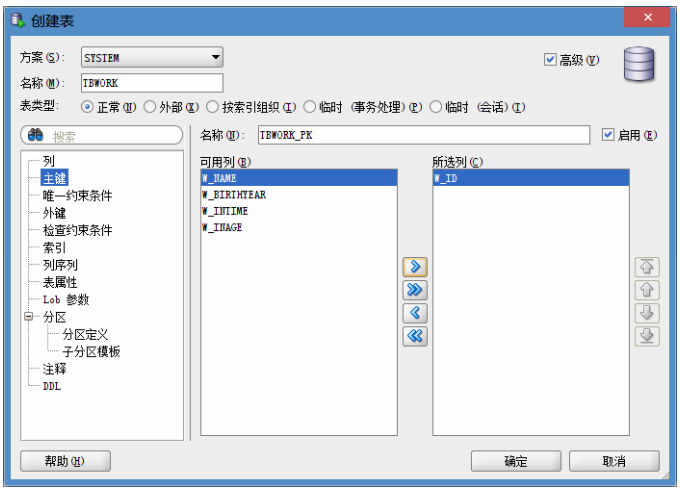

图 3-10 设置主键

如图 3-10 所示，主键需要在已经添加的列上面进行设置，在选择了【主键】节点后，对话框中列举了已经添加的列。选择需要设置主键的列，单击按钮可将选中的列移到右侧；单击按钮可将左侧所有列移到右侧；单击按钮可将右侧被选择的列移回左侧；单击按钮可将右侧所有列移回左侧。

右侧的列是被选择要设置为主键的列，在 Oracle 中支持主键组的使用，即将多个字段作为一个主键来使用。这一组字段中的每个字段，作为主键的构成缺一不可。对主键

的操作即对这一组字段的操作。

设置完成后单击【确定】按钮即可，保存设置的操作与创建表的操作一样。

也可在设计器中为没有主键的列设置主键，其操作与修改表的操作一样，在如图 3-8 所示的对话框中选择【主键】打开主键的设置，操作步骤与图 3-10 的操作步骤一样。

2．使用 SQL 语句设置主键约束

创建表时设置主键使用 CONSTRAINT…PRIMAPY KEY 语句，可以为主键约束定义一个名称，格式如下：

```
CONSTRAINT 约束名 PRIMARY KEY (主键字段)
```

本章范例 1 所创建的表中，就有 B_ID 列的主键约束。为已经创建的表添加主键约束时，需要使用 ADD CONSTRAINT 语句，格式如下：

```
ALTER TABLE 表名 ADD CONSTRAINT 约束名 PRIMARY KEY (主键字段)
```

如果表中已经存在主键约束，则向该表中再添加主键约束时，系统将出现错误。若要修改现有的主键约束，则需要先删除表中原有主键，再添加新的主键。删除主键使用 ALTER TABLE…DROP 语句，格式如下：

```
ALTER TABLE 表名 DROP CONSTRAINT 约束名;
```


3.4.3 唯一约束

唯一约束又称作 UNIQUE 约束，用来限制列中的数据不能重复，如通过邮箱注册微博时，会要求邮箱不能重复。Oracle 中的唯一约束是用来保证表中的某一列，或者是表中的某几列组合起来不重复的约束。Oracle 唯一约束具有以下 4 个特点。

（1）如果为列定义 UNIQUE 约束，那么该列中不能包括重复的值。

（2）在同一个表中，可以为某一列定义 UNIQUE 约束，也可以为多个列定义 UNIQUE 约束。

（3）Oracle 将会自动为 UNIQUE 约束的列建立一个唯一索引。

（4）可以在同一个列上建立 NOT NULL 约束和 UNIQUE 约束。

唯一约束可以在设计器中进行设置，也可以使用 SQL 语句进行设置。可以在创建表的时候设置，也可以在现有表中添加主键。

1．在设计器中设置唯一约束

在设计器中设置主键可以使用创建表的方法，在如图 3-3 所示的高级设置对话框中选择【唯一约束条件】，如图 3-11 所示。

如图 3-11 所示，唯一约束列也是需要在已经添加的列上面进行设置，在选择了【唯一约束条件】节点后，对话框中列举了已经添加的列。选择需要设置唯一约束的列，单击按钮可将选中的列移到右侧；单击按钮可将左侧所有列移到右侧；单击按钮可将右侧被选择的列移回左侧；单击按钮可将右侧所有列移回左侧。

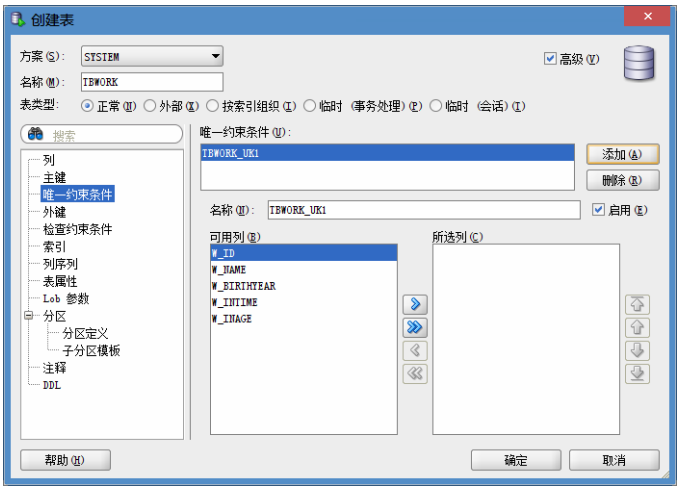

图 3-11 设置唯一约束

需要注意的是，唯一约束与之前的约束不同，一个表中可设置任意多个唯一约束，一个唯一约束可包括一个或多个列。

若一个唯一约束只作用于一个列，那么该列不能有重复的数据；如果一个唯一约束作用于多个列，那么这几个列的值组合起来不能重复。

图 3-11 中，对右边所选列设置唯一约束，一次只能设置一个唯一约束。可以设置单列或多列。若需要为表中的多个列分别设置唯一约束，需要单击右上角的【添加】按钮添加多个约束，并分别进行设置。设置完成后单击【确定】按钮即可，保存设置的操作与创建表的操作一样。

也可在设计器中为没有唯一约束的列设置唯一约束、修改唯一约束或取消唯一约束，其操作与修改表的操作一样。在如图 3-8 所示的对话框中选择【唯一约束条件】打开唯一约束的设置，选中已经存在的约束进行列的选取，或单击【删除】按钮删除约束。

2．使用 SQL 语句设置唯一约束

使用 SQL 语句可以在创建表的时候创建唯一约束，可以对现有表添加唯一约束，可以修改唯一约束或删除唯一约束。

在创建表时，可以为相对应的列使用 CONSTRAINT UNIQUE 语句添加指定 UNIQUE 约束，如下所示：

```
字段名 字段类型 CONSTRAINT 约束名 UNIQUE
```

【范例 8】

创建职工信息表 WORKERS，有编号、姓名、性别和年龄字段，设置姓名字段的唯一约束，代码如下。

```
CREATE TABLE WORKERS
(
  W_ID NUMBER NOT NULL ,
  W_NAME VARCHAR2(20) CONSTRAINT NAME_PK UNIQUE ,
```

```
  W_SEX VARCHAR2(20),
  W_AGE NUMBER
);
```

如果为一个列添加了 UNIQUE 约束，却并没有添加 NOT NULL 约束，那么该列的数据可以包含多个 NULL 值。也就是说多个 NULL 值不算重复值。

在现有的表中添加 UNIQUE 约束，使用 ADD UNIQUE 语句，语法如下：

```
ALTER TABLE 表名 ADD UNIQUE(列名)
```

可以使用 CONSTRAINT 语句在添加约束的同时为唯一约束命名，语法如下：

```
ALTER TABLE 表名 ADD CONSTRAINT 约束名 UNIQUE(列名)
```

【范例 9】

使用范例 8 中创建的职工信息表 WORKERS，为职工编号 W_ID 列添加唯一约束，代码如下。

```
ALTER TABLE WORKERS ADD UNIQUE(W_ID);
```

如果需要将表中指定列的 UNIQUE 约束删除，可以使用 ALTER TABLE…DROP UNIQUE 语句，语法如下：

```
ALTER TABLE 表名 DROP UNIQUE(列名);
```

上述代码只能够删除单列的唯一约束。若要删除有着多个列的唯一约束，需要使用如下格式语句：

```
ALTER TABLE 表名 DROP CONSTRAINT 约束名;
```

【范例 10】

分别删除 WORKERS 表中 W_ID 列的约束和名为 NAME_PK 的约束，代码如下。

```
ALTER TABLE WORKERS DROP UNIQUE(W_ID);
ALTER TABLE WORKERS DROP CONSTRAINT NAME_PK;
```

3.4.4 非空约束

表中一些列在实际生活中是不能够没有数据的，如用户注册时是不能够没有用户名的。非空约束用于限制列中的数据不能是空的，但可以是 0 或空字符串。非空约束又称作 NOT NULL 约束，具有以下 4 个特点。

（1）NOT NULL 约束只能在类级别上定义。

（2）在一个表中可以定义多个 NOT NULL 约束。

（3）在类定义 NOT NULL 约束后，该列中不能包含 NULL 值。

（4）如果表中数据已经存在空值 NULL，添加 NOT NULL 约束就会失败。

非空约束与其他约束不同，一个字段只能有允许为空和不允许为空这两种情况。允许为空可使用 NULL 设置；不允许为空使用 NOT NULL 设置。因此对非空约束的修改相当于对字段在 NULL 和 NOT NULL 之间进行切换，而不需要对字段进行约束的添加和删除。

非空约束可以在设计器中进行设置，也可以使用 SQL 语句进行设置。可以在创建表的时候设置，也可以在现有表中修改。

1．使用设计器设置 NOT NULL 约束

在 3.2.2 节介绍了表的创建，在图 3-1 中有字段的设置，可以看非空复选框。选中即可设置字段的 NOT NULL 约束，取消则恢复字段为 NULL。若要在设计器中修改字段的非空约束，找到需要修改的表的节点，右击并选择【编辑】选项打开【编辑表】对话框，如图 3-12 所示。

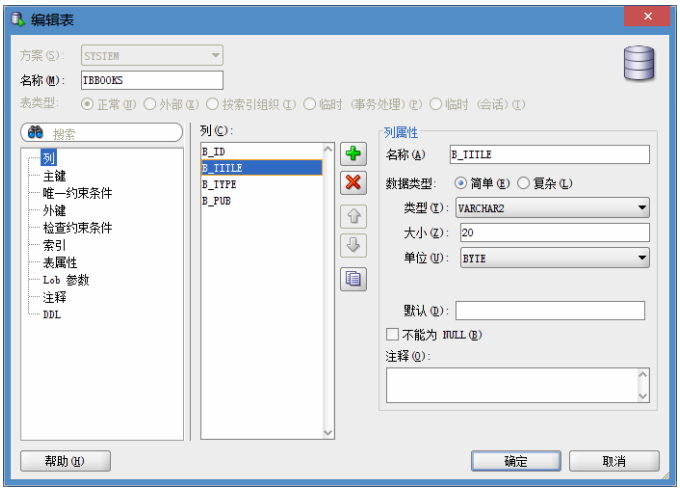

图 3-12　非空设置

如图 3-12 所示，除了不可见列，其他列都可以通过对话框右侧【不能为 NULL】复选框进行非空约束的修改。修改之后使用修改表的步骤进行保存。

2．使用 SQL 语句设置 NOT NULL 约束

使用 SQL 语句可在创建表的时候设置列为 NULL 或 NOT NULL，直接在列的数据类型后面添加 NULL 或 NOT NULL 即可。

【范例 11】

创建 WORKERS 表有编号、姓名、性别和年龄字段，设置其编号和姓名字段不能为空，性别和年龄字段可以为空，代码如下。

```
CREATE TABLE WORKERS
(
  W_ID NUMBER NOT NULL ,
  W_NAME VARCHAR2(20) NOT NULL ,
```

```
  W_SEX VARCHAR2(20) NULL,
  W_AGE NUMBER
);
```

上述代码中，性别字段使用 NULL 关键字来设置，而年龄字段没有使用 NULL 或 NOT NULL，但这两个字段的约束效果一样，再添加数据时可以为空。

若要修改现有表中字段的 NOT NULL 属性，可以修改 NULL 为 NOT NULL，或修改 NOT NULL 为 NULL，使用 ALTER TABLE…MODIFY 语句格式如下：

```
ALTER TABLE 表名 MODIFY 列名 NOT NULL|[NULL];
```

上述代码可修改 NULL 为 NOT NULL，或修改 NOT NULL 为 NULL。但是为列添加 NOT NULL 约束时，Oracle 将检查表中的所有数据行，以保证所有行对应的该列都不能存在 NULL 值。若列中存在 NULL 值则无法设置列的 NOT NULL 属性，因为修改列为 NOT NULL 将与表中已有的 NULL 数据冲突。

【范例 12】

修改 WORKERS 表中 W_NAME 字段为 NOT NULL，代码如下。

```
ALTER TABLE WORKERS MODIFY W_NAME NOT NULL;
```

3.4.5 外键约束

外键约束又称作 FOREIGN KEY 约束，它的作用是将不同表的字段关联起来，这些字段在修改数据、删除时有着关联。

外键除了关联表，还将在数据操作时维护数据完整性。以学生选课表为例，学生选课表有学生编号、所选科目等数据，而没有记录学生的详细信息，学生的详细信息在学生信息表中。那么，选课表学生编号字段中的值必须在学生表的学生编号字段中有记录；而且学生表在删除学生信息时，需要确保选课表中没有该学生的记录。

在使用 FOREIGN KEY 约束时，被引用的列应该具有主键约束，或者具有唯一性约束。要使用 FOREIGN KEY 约束，就应该具有以下 4 个条件。

（1）如果为某列定义 FOREIGN KEY 约束，则该列的取值只能为相关表中引用列的值或者 NULL 值。

（2）可以为一个字段定义 FOREIGN KEY 约束，也可以为多个字段的组合定义 FOREIGN KEY 约束。因此，FOREIGN KEY 约束既可以在列级别定义，也可以在表级别定义。

（3）定义了 FOREIGN KEY 约束的外键列，与被引用的主键列可以存在于同一个表中，这种情况称为“自引用”。

（4）对于同一个字段，可以同时定义 FOREIGN KEY 约束和 NOT NULL 约束。

外键的设置涉及两个表，设置方法相对于其他约束较为麻烦。外键约束可以在设计器中进行设置，也可以使用 SQL 语句进行设置。可以在创建表的时候设置，也可以在现有表中设置。

1．使用设计器设置外键约束

Oracle 外键的设置要求对应的表中要有主键或唯一约束，而且该表的外键列与对应表的主键或唯一约束设置关联，因此在创建时需要确保对应的表中有主键或唯一约束。

如果表中定义了外键约束，那么该表就被称为“子表”；如果表中包含引用键，那么该表称为“父表”。

在创建表的时候设计外键，需要确保相关联的表已经存在，并且有主键或唯一约束。外键约束需要在已经添加的列上面进行设置，打开【创建表】对话框并选择【外键】节点，如图 3-13 所示。首先需要单击【添加】按钮添加新的外键，接着选择引用表并选择引用约束条件。约束条件的选择直接影响了父表关联的列。此时右下方的【引用列】一栏中将出现引用表的列处于不可编辑状态，而【本地列】有下拉框可选择新建表中需要设置外键的列。

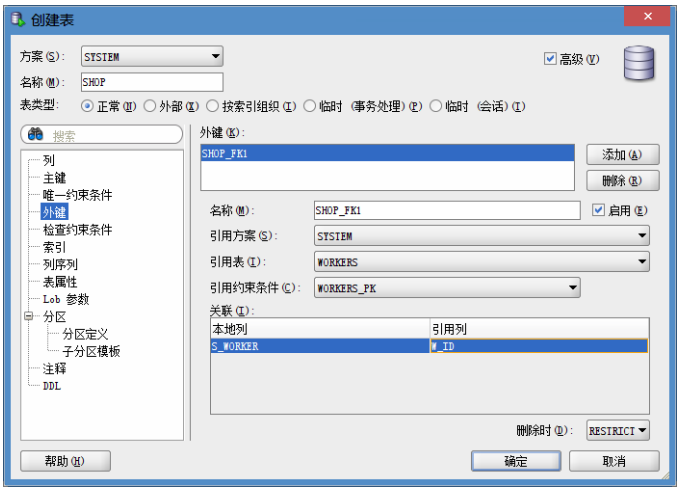

图 3-13　设置外键

如图 3-13 所示，右下角的【删除时】下拉框用于选择当前表中外键列数据修改时将执行的操作，有以下三个选项。

（1）RESTRICT 表示拒绝删除或者更新父表。

（2）CASCADE 表示父表中被引用列的数据被删除时，子表中对应的数据也将被删除。

（3）SET NULL 表示当父表被引用的列的数据被删除时，子表中对应的数据被设置为 NULL。要使这个选项起作用，子表中对应的列必须要支持 NULL 值。

外键设置完成之后，可根据创建表的步骤执行表的创建。

若所引用的表中没有主键或唯一约束，那么【引用约束条件】将处于空白状态，同时下方的【本地列】和【引用列】处于不可编辑状态，如图 3-14 所示。

对于现有表，若修改表的外键约束，可打开【编辑表】对话框如图 3-8 所示，选择【外键】节点并进行外键的设置，设置方法与图 3-13 的设置方法一样。接着保存表的修

改即可。

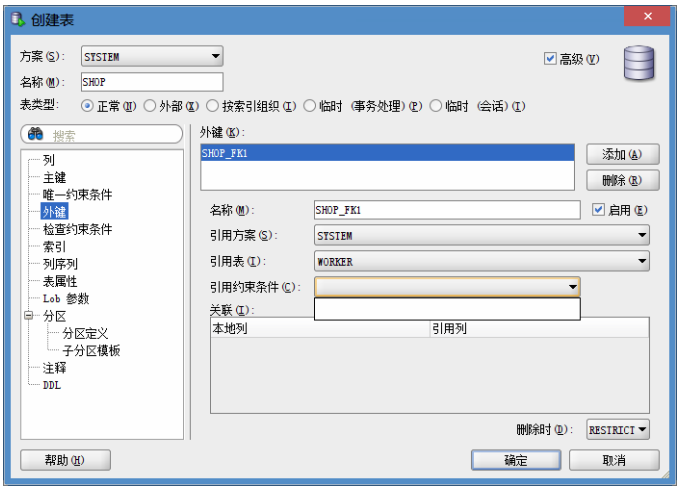

图 3-14 外键限制

删除外键约束，可直接选中外键约束的名称，单击【删除】按钮，保存表的修改即可。

2. 使用 SQL 语句设置外键约束

使用 SQL 语句设置外键包括创建表的时候设置外键；在现有表中添加、修改或删除外键等。

创建表的时候设置外键，只需在列的定义中添加 REFERENCES 关键字并指出相关联的表和列，语法如下：

```
列名 数据类型 REFERENCES 父表名(关联列)
```

【范例 13】

创建水果信息表 FRUIT，有编号、名称、性味类型、价格和负责人编号字段，其中负责人编号字段关联 WORKERS 表中的职工编号 W_ID 列，代码如下。

```
CREATE TABLE FRUIT
(
  F_ID NUMBER NOT NULL ,
  F_NAME VARCHAR2(20) NOT NULL ,
  F_TYPE VARCHAR2(20) NULL,
  F_PRICE NUMBER ,
  F_WORKER REFERENCES WORKERS(W_ID)
);
```

注意

外键列和被引用列的列名可以不同，但是数据类型必须完全相同。

创建好的表，可以使用 ALTER CONSTRAINT FOREIGN KEY REFERENCES 子句添加外键，格式如下：

```
ALTER TABLE 子表名 ADD CONSTRAINT 约束名 FOREIGN KEY (子表的外键列)
REFERENCES 父表名(关联列)
```

【范例 14】

修改 TBBOOKS 表，添加 NUMBER 类型字段 B_WRITER 并设置唯一约束，为其外键关联 WORKERS 表中的职工编号 W_ID 列。省略 B_WRITER 字段的添加，设置外键代码如下：

```
ALTER TABLE TBBOOKS ADD CONSTRAINT WRITER_PK FOREIGN KEY (B_WRITER)
REFERENCES WORKERS(W_ID);
```

对于一些不需要使用的外键约束，删除的时候需要使用 ALTER TABLE…DROP CONSTRAINT 语句，如下所示：

```
ALTER TABLE 表名 DROP CONSTRAINT 约束名;
```

3.4.6 检查约束

检查约束又称作 CHECK 约束，它的作用就是查询用户向该列插入的数据是否满足了约束中指定的条件，如果满足则将数据插入到数据库内，否则就返回异常。CHECK 约束具有以下几个特点。

（1）在 CHECK 约束的表达式中，必须引用表中的一个或者多个列，表达式的运算结果是一个布尔值，且每列可以添加多个 CHECK 约束。

（2）对于同一列，可以同时定义 CHECK 约束和 NOT NULL 约束。

（3）CHECK 约束既可以定义在列级别中，也可以定义在表级别中。

（4）约束条件必须返回布尔值，这样插入数据时 Oracle 将会自动检查数据是否满足条件。

检查约束可以在设计器中进行设置，也可以使用 SQL 语句进行设置。可以在创建表的时候设置，也可以在现有表中设置。

1．使用设计器设置检查约束

在设计器中设置检查约束可以使用创建表的方法，在如图 3-3 所示的高级设置对话框中选择【检查约束条件】，如图 3-15 所示。

如图 3-15 所示，首先单击右侧的【添加】按钮添加新的检查约束，接着编辑约束的名称和检查条件。检查约束在设置时不需要针对具体的列，但若对列数据进行限制，需要在条件中有表示。如图 3-15 创建的约束限制了 S_AGE 的值要在 0～20 之间，虽然没有显式地设置列，但在条件中表示了出来。

检查约束设置完成之后，可根据创建表的步骤执行表的创建。

对于现有表，若修改表的检查约束，可打开【编辑表】对话框，选择【检查约束条

件】节点并进行检查约束的设置，设置方法与图 3-15 的设置方法一样。接着保存表的修改即可。

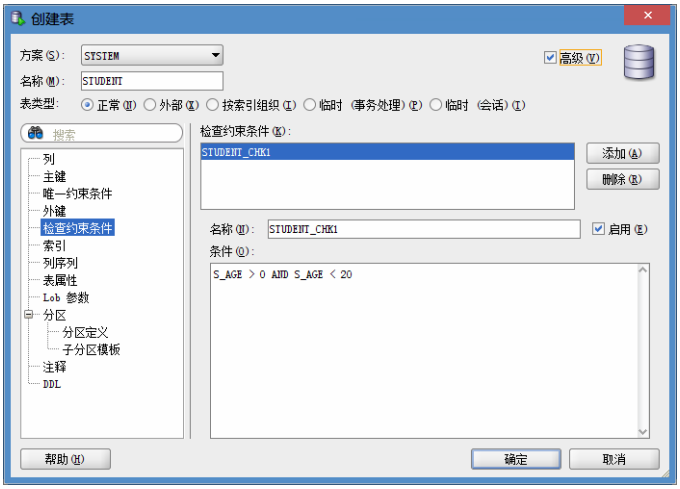

图 3-15 设置检查约束

删除检查约束，可直接选中检查约束的名称，单击【删除】按钮，保存表的修改即可。

2. 使用 SQL 语句设置检查约束

使用 SQL 语句设置检查约束包括创建表的时候设置检查约束；在现有表中添加、修改或删除检查约束等。

在创建表的时候设置检查约束，需要在检查约束所作用的列后面使用 CONSTRAINT…CHECK 语句创建，语法如下：

```
列名 数据类型 CONSTRAINT 约束名 CHECK(约束条件)
```

【范例 15】

创建 STUDENT 表，有学生编号、姓名、性别和年龄字段，其中年龄字段添加检查约束限制字段的值在 0～20 之间，代码如下。

```
CREATE TABLE STUDENT
(
  S_ID NUMBER NOT NULL ,
  S_NAME VARCHAR2(20) ,
  S_SEX VARCHAR2(20) ,
  S_AGE NUMBER CONSTRAINT STUDENT_CHK1 CHECK(S_AGE > 0 AND S_AGE < 20)
);
```

为已经创建好的表中已经存在的列添加 CHECK 约束，需要使用 ALTER TABLE…ADD CHECK 语句，格式如下：

```
ALTER TABLE 表名 ADD CONSTRAINT 约束名 CHECK(约束条件);
```

【范例 16】

为 STUDENT 表的性别字段添加检查约束，要求 S_SEX 字段的值为“男性”或“女性”，代码如下。

```
ALTER TABLE STUDENT ADD CONSTRAINT SEX_CHECK CHECK(S_SEX='男性' OR S_SEX='女性');
```

如果要删除已经创建好的表中已经存在的列 CHECK 约束，需要使用 ALTER TABLE…DROP CONSTRAINT 语句，语法如下：

```
ALTER TABLE 表名 DROP CONSTRAINT 约束名;
```

3.4.7 禁止和激活约束

在 Oracle 数据库中根据对表的操作与约束规则之间的关系，将约束分为 DISABLE 和 ENABLE 两种，也就是说可以通过这两种约束状态来控制约束是禁用还是激活。

当约束状态处于激活状态时，如果对表的操作与约束规则相冲突，则操作就会被取消。在默认的情况下，新添加的约束，默认状态是激活的。只有在手动配置的情况下约束才能被禁止。

（1）禁止约束：DISABLE 关键字用来设置约束的状态为禁止状态。也就是说约束状态禁止的时候，即使对表的操作与约束规则相冲突，操作也会被执行。

（2）激活约束：ENABLE 关键字用来设置约束的状态为激活状态。也就是说约束状态激活的时候，如果对表的操作与约束规则相冲突，操作就会被取消。

禁止和激活约束可以在设计器中进行设置，也可以使用 SQL 语句进行设置。可以在创建表的时候设置，也可以在现有表中设置。

在设计器中设置约束的禁止和激活状态，可以在创建表的时候设置，也可在现有表中设置。如设置主键约束的状态，可在如图 3-10 所示的对话框中选择【启用】复选框激活约束，或取消【启用】复选框禁止约束。

其他约束的状态设置与主键约束一样，在约束的设置对话框中选择【启用】复选框激活约束，或取消【启用】复选框禁止约束。在现有的表中修改约束状态，也是对【启用】复选框的设置，这里不再介绍。

使用 SQL 设置约束的状态，分为在创建表的时候设置和在现有表中修改。

在创建表的时候设置时，需要在约束语句的后面添加 DISABLE 关键字禁止约束（默认是激活状态，不需要显式地定义）。

【范例 17】

创建 TEACHER 表，有编号、姓名、性别和年龄字段，其中年龄字段添加检查约束限制字段的值在 20～60 之间，禁用该约束，代码如下。

```
CREATE TABLE TEACHER
(
  T_ID NUMBER NOT NULL ,
  T_NAME VARCHAR2(20) ,
```

```
  T_SEX VARCHAR2(20) ,
  T_AGE NUMBER CONSTRAINT TAGE_CHK1 CHECK(T_AGE > 20 AND T_AGE < 60) DISABLE
);
```

在现有的表中修改约束的状态分为：将约束状态修改为禁止状态；将约束状态修改为激活状态。

将约束状态修改为激活状态有两种方法，如下所示。

（1）使用 ALTER TABLE…ENABLE 语句语法如下：

```
ALTER TABLE 表名 ENABLE CONSTRAINT 约束名;
```

（2）使用 ALTER TABLE…MODIFY…ENABLE 语句语法如下：

```
ALTER TABLE 表名 MODIFY ENABLE CONSTRAINT 约束名;
```

将约束状态修改为禁止状态使用 DISABLE 关键字，语法如下：

```
ALTER TABLE 表名 DISABLE CONSTRAINT 约束名;
```

约束的状态可以通过一些 Oracle 数据库提供的数据字典视图和动态性能视图来查询，例如，使用 USER_CONSTRAINTS 和 USER_CONS_COLUMNS 等来查询。

通过这些视图可以查询表和列中的约束信息，包括约束的所有者、约束名、约束类型、所属的表和约束状态等。如数据字典视图 USER_CONSTRAINTS 中常用的字段及其含义如表 3-3 所示。

表 3-3 USER_CONSTRAINTS 视图常用字段及其说明

字段名	类型	说明
owner	VARCHAR2(30)	约束的所有者
constraint_name	VARCHAR2(30)	约束名
constraint_type	VARCHAR2(1)	约束类型（P、R、C、U、V、O）
table_name	VARCHAR2(30)	约束所属的表
status	VARCHAR2(8)	约束状态（ENABLE、DISABLE)
deferrable	VARCHAR2(14)	约束是否延迟（DEFERRABLE、NOTDEFERRABLE)
deferred	VARCHAR2(9)	约束是立即执行还是延迟执行（IMMEDIATE、DEFERRED)

表 3-3 中，约束类型的含义如下所示。

（1）C 代表 CHECK 或 NOT NULL 约束。

（2）P 代表 PRIMARY KEY 约束。

（3）R 代表 FOREIGN KEY 约束。

（4）U 代表 UNIQUE 约束。

（5）V 代表 CHECK OPTION 约束。

（6）O 代表 READONLY 只读约束。

【范例 18】

利用范例 17 中的 TEACHER 表，添加编号字段的唯一约束，查询表中约束的信息，修改其检查约束为激活状态，再次检查表中的约束信息，步骤如下。

（1）添加 T_ID 字段的唯一约束代码如下。

```
ALTER TABLE TEACHER ADD CONSTRAINT TID_PK UNIQUE(T_ID);
```

（2）首先查询 TEACHER 表中的约束信息，代码如下。

```
SELECT CONSTRAINT_NAME ,CONSTRAINT_TYPE ,STATUS FROM USER_CONSTRAINTS
WHERE TABLE_NAME='TEACHER';
```

上述代码的执行效果如下所示。

```
CONSTRAINT_NAME  C    STATUS
---------------  -    ------
TAGE_CHK1        C    DISABLED
TID_PK           U    ENABLED
SYS_C009878      C    ENABLED
```

（3）修改检查约束为激活状态，代码如下。

```
ALTER TABLE TEACHER ENABLE CONSTRAINT TAGE_CHK1;
```

（4）再次查询 TEACHER 表中的约束信息，参考步骤（2）中的代码，其效果如下所示。


```
CONSTRAINT_NAME    C  STATUS
---------------    -  ------
TID_PK             U  ENABLED
SYS_C009878        C  ENABLED
TAGE_CHK1          C  ENABLED
```

通过查询数据字典 USER_CONS_COLUMNS，可以了解定义约束的列。下面是 USER_CONS_COLUMNS 视图中部分列的说明，如表 3-4 所示。

表 3-4 USER_CONS_COLUMNS 视图常用字段及其说明

字段名	类型	说明
owner	VARCHAR2(30)	约束的所有者
constraint_name	VARCHAR2(30)	约束名
table_name	VARCHAR2(30)	约束所属的表
column_name	VARCHAR2(4000)	约束所定义的列

【范例 19】

查询 TEACHER 表中的约束定义在哪个列上，代码如下。

```
SELECT CONSTRAINT_NAME,COLUMN_NAME FROM USER_CONS_COLUMNS WHERE
TABLE_NAME='TEACHER';
```

上述代码的执行效果如下所示。

```
CONSTRAINT_NAME   COLUMN_NAME
---------------   -----------
TID_PK              T_ID
```

```
TAGE_CHK1           T_AGE
SYS_C009878           T_ID
```

激活和禁用两种约束状态是对表进行更新和插入操作时是否验证操作符合约束规则。在 Oracle 中，除了激活和禁用两种约束状态，还有另外两种约束状态，用来决定是否对表中已经存在的数据进行约束规则检查。

通常约束的验证状态有两种，一种是验证约束状态，如果约束处于验证状态，则在定义或者激活约束时，Oracle 将对表中所有已经存在的记录进行验证，检验是否满足约束限制；另外一种是非验证约束，如果约束处于非验证状态，则在定义或者激活约束时，Oracle 将对表中已经存在的记录不执行验证操作。

将禁止、激活、验证和非验证状态相互结合，则可以将约束分为 4 种状态，如表 3-5 所示。

表 3-5 约束的状态

状态	说明
激活验证状态（ENABLE VALIDATE）	激活验证状态是默认状态，这种状态下，Oracle 数据库不仅对以后添加和更新数据进行约束检查，也会对表中已经存在的数据进行检查，从而保证表中的所有记录都满足约束限制
激活非验证状态（ENABLE NOVALIDATE）	这种状态下，Oracle 数据库只对以后添加和更新的数据进行约束检查，而不检查表中已经存在的数据
禁止验证状态（DISABLE VALIDATE）	这种状态下，Oracle 数据库对表中已经存在的记录执行约束检查，但是不允许对表执行添加和更新操作，因为这些操作无法得到约束检查
禁止非验证状态（DISABLE NOVALIDATE）	这种状态下，无论是表中已经存在的记录，还是以后添加和更新的数据，Oracle 都不进行约束检查

在非验证状态下激活约束比在验证状态下激活约束节省时间。所以，在某些情况下，可以选择使用激活非验证状态。例如，当需要从外部数据源引入大量数据时。

3.4.8 延迟约束

由表 3-3 可以看出，约束有着延迟属性。在 Oracle 程序中，如果使用了延迟约束，那么当执行增加和修改等操作时，Oracle 将不会像以前一样立即做出回应和处理，而是在规定条件下才会被执行。这样用户可以自定义何时验证约束，例如将约束检查放在失误结束后进行。

默认情况下，新添加的 Oracle 约束延迟操作时没有开启的，也就是说在执行 INSERT 和 UPDATE 操作语句时，Oracle 程序将会马上做出对应的处理和操作，如果语句违反了约束，则相应的操作无效。要想对约束进行延迟，那么就使用关键字 DEFERRABLE 创建延迟约束。延迟约束还有以下两种初始状态，

（1）INITIALLY DEFERRED：约束的初始状态是延迟检查。

（2）INITIALLY IMMEDIATE：约束的初始状态是立即检查。

修改约束的延迟状态使用 ALTER TABLE…MODIFY CONSTRAINT 语句，语法如下：

```
ALTER TABLE 表名 MODIFY CONSTRAINT 约束名 INITIALLY DEFERRED| INITIALLY
IMMEDIATE;
```

如果约束的延迟已经存在，则可以使用 SET CONSTRAINTS ALL 语句将所有约束切换为延迟状态，表现如下。

（1）如果设置为 SET CONSTRAINTS ALL DEFERRED，则延迟检查。

（2）如果设置为 SET CONSTRAINTS ALL IMMEDIATE，则立即检查。

延迟约束是在事务被提交时强制执行的约束。添加约束时可以通过 DEFERRED 子句来指定约束为延迟约束。约束一旦创建以后，就不能修改为 DEFERRED 延迟约束。

Oracle 中是不能修改任何非延迟性约束的延迟状态的。

3.5 实验指导——家电信息管理

结合本章内容，根据家电信息创建相关的表、字段和约束，具体要求如下。

1）创建家电信息表，有商品编号、名称、类型、品牌、价格和能效等级字段，其中商品编号、类型和品牌字段为 NUMBER 类型，商品编号为主键。

2）创建类型表，有类型编号和类型名称字段。

3）创建品牌表，有品牌编号和品牌名称字段。

4）为家电信息表添加外键设置，使其类型字段关联类型表的类型编号；品牌字段关联品牌表中的品牌编号。

5）为家电信息表的能效等级字段添加检查约束，使字段值在 1～5 之间，包含 1 和 5。

6）为家电信息表添加唯一约束，使商品名称和品牌字段的组合不能重复。

7）检查约束信息和约束所作用的列。

（1）创建家电信息表，有商品编号、名称、类型、品牌、价格和能效等级字段，其中商品编号、类型和品牌字段为 NUMBER 类型，商品编号为主键，代码如下。

```
CREATE TABLE APPLIANCES
(
  A_ID NUMBER NOT NULL ,
  A_TITLE VARCHAR2(20) ,
  A_TYPE NUMBER,
  A_BRAND NUMBER,
  A_PRICE NUMBER,
  A_GRADE NUMBER,
  CONSTRAINT AID_PK PRIMARY KEY(A_ID)
  ENABLE
);
```

（2）创建类型表，有类型编号和类型名称字段，代码如下。

```
CREATE TABLE A_TYPE
(
  T_ID NUMBER NOT NULL ,
  T_TITLE VARCHAR2(20) ,
  CONSTRAINT ATID_PK PRIMARY KEY(T_ID)
  ENABLE
);
```

（3）创建品牌表，有品牌编号和品牌名称字段，代码如下。

```
CREATE TABLE A_BRAND
(
  B_ID NUMBER NOT NULL ,
  B_TITLE VARCHAR2(20) ,
  CONSTRAINT BID_PK PRIMARY KEY(B_ID)
  ENABLE
);
```

（4）为家电信息表添加外键设置，使其类型字段关联类型表的类型编号；品牌字段关联品牌表中的品牌编号，代码如下。

```
ALTER TABLE APPLIANCES ADD CONSTRAINT AT_PK FOREIGN KEY (A_TYPE) REFERENCES
A_TYPE(T_ID);
```


（5）为家电信息表的能效等级字段添加检查约束，使字段值在 1～5 之间，包含 1 和 5，代码如下。

```
ALTER TABLE APPLIANCES ADD CONSTRAINT AB_PK FOREIGN KEY (A_BRAND)
REFERENCES A_BRAND(B_ID);
```

（6）为家电信息表添加唯一约束，使商品名称和品牌字段的组合不能重复，代码如下。

```
ALTER TABLE APPLIANCES ADD CONSTRAINT UNIQUE_PK UNIQUE(A_TITLE,A_BRAND);
```

（7）检查约束信息，代码如下。

```
SELECT CONSTRAINT_NAME ,CONSTRAINT_TYPE ,STATUS FROM USER_CONSTRAINTS
WHERE TABLE_NAME='APPLIANCES';
```

上述代码的执行效果如下所示。

```
CONSTRAINT_NAME  C  STATUS
---------------  -  ------
AID_PK           P  ENABLED
UNIQUE_PK        U  ENABLED
AT_PK            R  ENABLED
AB_PK            R  ENABLED
SYS_C009886      C  ENABLED
```

（8）检查约束所作用的列，代码如下。

```
SELECT CONSTRAINT_NAME,COLUMN_NAME FROM USER_CONS_COLUMNS WHERE
TABLE_NAME='APPLIANCES';
```

上述代码的执行效果如下所示。

```
CONSTRAINT_NAME   COLUMN_NAME
---------------   -----------
UNIQUE_PK          A_BRAND
UNIQUE_PK          A_TITLE
AB_PK              A_BRAND
AT_PK              A_TYPE
AID_PK             A_ID
SYS_C009886         A_ID
已选择 6 行。
```

思考与练习

一、填空题

1．Oracle 数据类型有字符类型、_____、日期类型和图片类型。

2．_____又称作 IOT 表，存储在索引结构中。

3．_____是普通的标准数据库表，数据以堆的方式管理。

4．Oracle 允许用户在表上使用_____列来存储表达式。

5．临时表、_____和集群表不支持不可见列。

二、选择题

1．下列关于虚拟列说法错误的是_____。

A．虚拟列可以使用 Oracle 自带的函数，也可以使用用户定义的函数

B．Oracle 含有虚拟列的表中不存储虚拟列的值

C．虚拟列的值存储在数据文件中

D．虚拟列在创建时可以省略数据类型

2．下列关于不可见列说法正确的是_____。

A．不可见列是存储在表中，但是不能够查询出来的

B．虚拟列和分区列同样也可以定义为不可见类型

C．不可见列使用 VISIBLE 定义

D．不可见列与虚拟列不同，可以使用 INSERT 语句隐式插入数据

3．重命名列使用_____。

A．NEWNAME

B．NAME

C．RENAME

D．RENNAME

4．UNIQUE 表示_____。

A．唯一约束

B．特别约束

C．检查约束

D．附加约束

5．如果一个列定义了一个 PRIMARY KEY 约束，那么该列_____。

A．不能为空，可以重复

B．可以为空，不能重复

C．可以为空也可以重复

D．不能为空也不可以重复检查约束

6．一个表中，外键约束所关联的列要满足以下要求_____。

A．必须是主键约束

B．必须有唯一约束

C．既要有主键约束也要有唯一约束

D．可以是唯一约束或主键约束

三、简答题

1．简述主键约束的作用。

2．总结外键设置的注意事项。

3．简述如何设置外键。

4．概括约束的 4 种状态以及使用的限制。

5．总结约束的种类以及应用。