第5章 chapter 5

索引及视图

快速查询各种资源信息最为常用，索引类似于图书目录，提供了指向包含特定数据的指针。利用该索引，可以直接定位到所要查找的相应记录，从而提高数据库数据查询的速度，提升数据库系统性能。索引技术是数据库系统实现的重要部分。

视图如同常见的网页，是由其他关系上的查询所定义的一种特殊的关系。视图是一种虚表，数据库中只存放视图的定义。视图可以满足不同用户对数据的不同需求，并提供了一种数据安全保护机制。

教学目标

（1）理解索引的相关概念、作用及应用。

（2）了解索引的主要基本结构与原理。

（3）理解视图的相关概念和主要作用。

（4）掌握视图的定义、查询和更新等操作。

教学视频

5.1 教学微视频

5.1 索引基本知识

【案例 5-1】 在学生关系中查找姓名为“李凯”的学生，查询语句：

```
SELECT *
FROM 学生
WHERE 姓名='李凯'
```

该查询只涉及关系（表）中的少量记录。如果系统遍历关系中全部元组，逐行比较每个记录的姓名值是否匹配 WHERE 子句中的条件，这种查询方式显然是非常低效的。因为进行这个查询包含了大量的磁盘输入输出操作，造成数据库系统性能的瓶颈。如果在该关系的姓名属性列上建立索引，则不需要对整个表进行扫描，就能直接找到查询记录所在的磁盘位置，从而快速检索出需要的信息。

5.1.1 索引的概念、特点及类型

1. 索引的概念

为了实现快速查找关系中的记录，数据库系统设计了索引结构。每个索引结构与特

定的搜索码相关联。搜索码，是指用于在关系中建立索引的属性或属性集，如学号。索引文件中的索引项(Index Entry)或索引记录(Index Record)由一个搜索码值和指向具有该搜索码值的一条或多条记录的指针构成。指向记录的指针包括磁盘块号和标识磁盘块内记录的块内偏移量。类似于图书中的目录标注了各部分内容和所对应的页码，数据库中的索引也注明了关系中各行数据及其所对应的磁盘存储位置。查询数据时，数据库系统首先在索引中找到符合条件的索引值(搜索码值)，再通过该索引值所对应的指针定位到相应记录所在的磁盘块，取出该磁盘块，得到所要查找的元组信息。索引的概念涉及数据库中数据的物理存储顺序，因此属于数据库三级模式中的内模式范畴。

知识拓展
关系记录的物理存储

2. 索引的特点

数据库使用索引可以提高系统的性能，主要体现在5个方面。

(1) 极大地提高数据查询的速度，这也是其最主要优点。

(2) 通过创建唯一性索引，可以保证数据库中各行数据的唯一性。

(3) 建立在外键上的索引可以加速多表之间的连接，有益于实现数据的参照完整性。

(4) 查询涉及分组和排序时，也可显著减少分组和排序的时间。

(5) 通过使用索引可以在查询过程中使用优化隐藏器，提高系统的性能。

使用索引能够提高系统的性能，但是索引为查找带来的性能好处要付出代价。

(1) 物理存储空间中除了存放数据表之外，还需要一定的额外空间来存放索引。

(2) 对数据表进行插入、修改和删除操作时，相应的索引也需要动态维护更新，消耗系统资源。

3. 索引的类型

在SQL Server中，根据其索引记录的结构和存放位置可分为聚簇索引(Clustered Index，也称聚集索引)、非聚簇索引(Nonclustered Index，也称非聚集索引)和其他索引。聚簇索引也称为主索引，非聚簇索引也称为辅助索引。

聚簇是为了提高在某个属性(或属性组)上的查询速度，把这个或这些属性(称为聚簇码)上具有相同值的元组集合存放在连续的物理块中。如果数据文件中的记录按照聚簇码指定的排序顺序存储，那么该搜索码对应的索引称为聚簇索引。聚簇索引的搜索码常常是主键。可见，聚簇索引能够确定表中数据的物理存储顺序，即表中数据是按照聚簇码的顺序进行物理排序的。汉语字典的正文就是按照一个以文字拼音顺序为聚簇码建立的聚簇索引顺序组织的。在带有聚簇索引的表中添加数据后，数据的排列顺序与数据的输入先后顺序无关，而是由聚簇码的值所决定。聚簇索引强制表中插入记录时按聚簇码顺序存储。

在默认情况下，大多数数据系统，包括SQL Server自动在主键上创建聚簇索引。因此，可以看到关系中的元组通常是按照主键的值排序的。

注意：一个表中只能包含一个聚簇索引，表中行的物理顺序与索引顺序一致。在查询频率较高的属性列上建立聚簇索引，可避免每次查询该列时都进行排序，提高查询效率。但是聚簇索引不适用于建立在数据更新比较频繁的列上，因为这将导致数据的多次重新排序和索引的多次更新，这种频繁的索引维护会增加数据库系统的额外负担。

非聚簇索引中,数据与索引的存储位置可以完全独立,数据存储在一个地方,索引存储在另一个地方,索引中带有指向数据存储位置的逻辑指针。索引顺序与数据的物理排列顺序无关,类似汉语字典中按照偏旁部首的查找方式。因此,非聚簇索引中的索引项是按照索引值顺序存储的,但是表中的数据则可能按照其他顺序存储(由聚簇索引决定)。可以在一个表上建立多个非聚簇索引。

特别理解 聚簇索引与非聚簇索引的区别

如果一个表中包含一个非聚簇索引但没有聚簇索引,新插入的数据将会被插入到最末一个数据页中,然后非聚簇索引会被更新。如果该表中还包含聚簇索引,则先根据聚簇索引确定新数据的位置,然后再更新聚簇索引和非聚簇索引。

其他类型索引中最常见的是唯一索引(Unique Index),其索引列中不包含重复的值,即唯一索引中每一个索引值都对应表中唯一的数据记录。在建立数据表并声明主键或唯一约束时,通常情况下 SQL Server 会自动创建与之对应的唯一索引。当然每个表中可以包含多个唯一索引,只要建立唯一索引的列中没有重复的值即可。

聚簇索引和非聚簇索引也都可以是唯一的。因此,只要索引列中的数据是唯一的,就可以在表上创建一个唯一的聚簇索引和多个唯一的非聚簇索引。

SQL Server 中还提供了视图索引、列存储索引、XML 索引等其他索引,其相关说明见表 5-1,各种索引的具体用法可参阅相关文档。

表 5-1 SQL Server 2022 的索引类型及其简单说明

索引类型	简单说明
聚簇索引	创建索引时,索引键值的逻辑顺序决定表中对应行的物理顺序。聚簇索引的底层(或称为叶)包含该表的实际数据行,因此要求数据库具有额外的可用工作空间来容纳数据的排序结果和原始表或现有聚簇索引数据的临时副本。一个表或者视图只允许同时有一个聚簇索引
非聚簇索引	创建一个指定表的逻辑排序的索引。对于非聚簇索引,数据行的物理排序独立于索引排序。一般来说,先创建聚簇索引,后创建非聚簇索引
唯一索引	唯一索引保证在索引列中的全部数据是唯一的,不能包含重复数据。如果存在唯一索引,数据库引擎会在每次插入操作添加数据时检查重复值。可生成重复键值的插入操作将被回滚,同时数据库引擎显示错误消息
分区索引	为了改善大型表的可管理性和性能,常会对其进行分区。分区表在逻辑上是一个表,而物理上是多个表,对应的可以为已分区表建立分区索引。但是有时也可以在未分区的表中使用分区索引,为表创建一个使用分区方案的聚簇索引后,一个普通表就变成了分区表
筛选索引	筛选索引是一种经过优化的非聚簇索引,适用于从表中选择少数行的查询。筛选索引使用筛选谓词对表中的部分数据进行索引。与全文索引相比,设计良好的筛选索引可以提高查询性能,减少索引维护开销,降低索引存储开销
全文索引	全文索引主要包含分词器、词干分析器和同义词分析器三种分析器。生成全文索引就是把表中的文本数据进行分词和提取词干,并转换同义词,过滤掉分词中的停用词,最后把处理之后的数据存储到全文索引中。全文索引中存储分词及其位置等信息,由 SQL Server 全文引擎生成和维护。使用全文索引可以大大提高从长字符串数据中搜索复杂的词的性能

续表

索引类型	简单说明
空间索引	空间索引是一种扩展索引，允许对数据库中的空间数据类型（如 geometry 或 geography）列编制索引
XML	可以对 xml 数据类型列创建 XML 索引。它们对列中 XML 实例的所有标记、值和路径进行索引，从而提高查询性能
计算列上的索引	从一个或多个其他列的值或者某些确定的输入值派生的列上建立的索引
带有包含列的索引	可以将非键列（称为包含列）添加到非聚簇索引的叶级别，从而通过涵盖查询来提高查询性能。也就是说，查询中引用的所有列都作为键列或非键列包含在索引中。这样，查询优化器可以通过索引扫描找到所需要的全部信息，而无须访问表或聚簇索引数据
列存储索引	在常规索引中，表中每一行的数据都会存储在一起，每列数据在一个索引中是跨所有页保留的。而在列存储索引中，将数据按列来存储并压缩，每一列数据存放在一起。这种将数据按列压缩存储的方式减少了查询对磁盘 I/O 开销和 CPU 开销，最终达到提升查询效率、降低响应时间的目的

* 5.1.2 索引的结构与原理

顺序索引是数据库系统最早采用的一种索引模式。顺序索引按照顺序存储搜索码的值，并且将每个搜索码与包含该搜索码的记录关联起来。这种索引模式存在的主要缺陷是随着文件的增大，索引查找速度下降。

B 树或 B^+ 树索引结构是数据库系统广泛使用的一种索引结构。采用平衡树（Balanced Tree）结构，其中树根到树叶的每条路径的长度都相同。B 树的顶端称为根结点，最底端的结点称为叶层结点或叶结点，中间层的结点称为非叶层结点。其结构如图 5-1 所示，索引树包含索引页（Index Page）和数据页（Data Page）。其中，索引页用来存放索引项和指向下一层的指针，数据页用来存放数据，索引 B 树中的每一个索引页称为一个索引结点。当在建有索引的表上执行查询操作时，系统会先在其索引页（Index Page）进行查找。首先从 B 树的根结点出发，对结点内的已排好序的索引关键字（Key）序

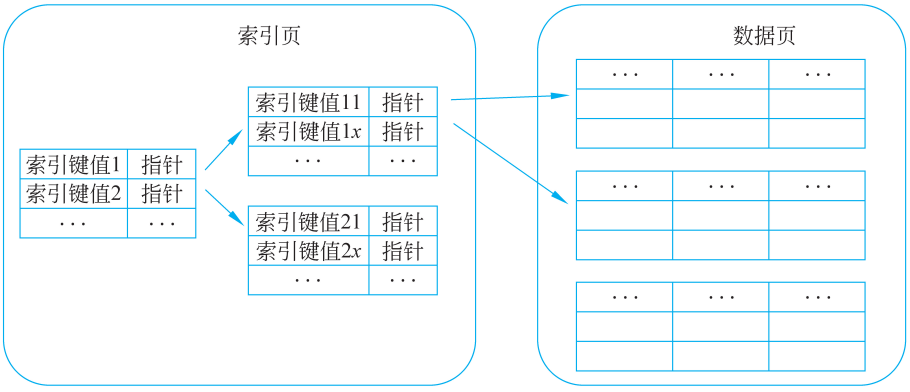

图 5-1 B 树索引结构

列进行二分查找，如果命中，则进入查询关键字所属范围的子结点，重复查找，直到到达B树索引的叶子结点，最后从数据页读取到相应的数据。因此，B树的查询性能等价于二分查找，效率较高。

此外，多数情况下索引记录中仅包含索引关键字和较小的指针链接，比完整的数据行所占空间要小很多，因此索引页相对数据页要密集很多，一个索引页可以存储数量更多的索引记录。这意味着在索引中执行查询操作在I/O上占很大的优势，这也是索引本质上的优势所在。

1. 聚簇索引的结构

聚簇索引的非叶层结点中，只包含下一结点的第一个键值及指向下一结点的指针。指针的格式：文件编号＋数据页编号。聚簇索引的叶子结点就是表中的数据行，并且数据按照聚簇索引项的值进行物理排序存储。如图5-2所示为在姓名字段上建立的聚簇索引，当需要根据姓名字段查找记录时，数据库系统会先找到该索引的根结点，然后根据

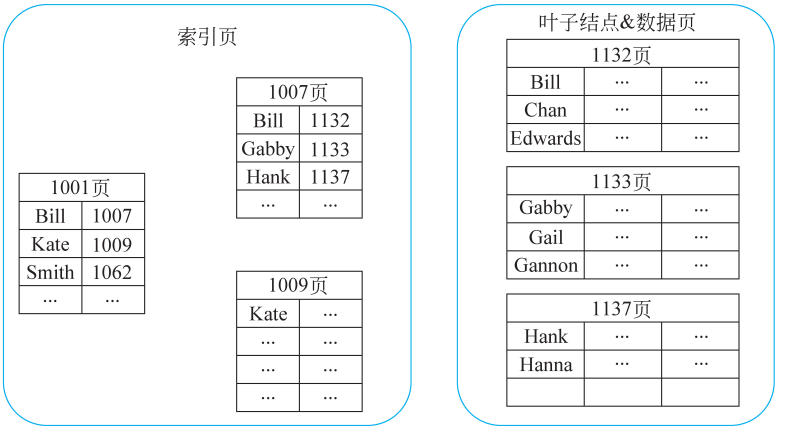

图5-2 聚簇索引结构

指针查找下一个，直到找出需要的某个记录。例如，现在要查询Gail，由于它介于Bill和Kate之间，因此系统先找到Bill对应的索引页1007；在该页中Gail介于Gabby和Hank之间，因此找到1133(也是数据结点)，并最终在此页中找到了Gail对应的记录行。

知识拓展
聚簇索引B树的建立

2. 非聚簇索引的结构

非聚簇索引中的叶子结点并非数据结点，叶子结点存储的是键值＋指针，指针中包括页指针和指针偏移量，用于定位到具体的数据行。在除叶结点外的其他索引结点，存储的也是类似的内容，只不过它是指向下一级的索引页的。图5-3为在姓名字段上建立的非聚簇索引。假设要查询Gill，系统从根结点出发，考虑到Gill位于Bill和Kate之间，则先定位到索引页1007；下一步判断Gill位于Gabby和Hank之间，再定位到索引页1133，找到Gill。此时，索引页1133为索引的叶子结点，Gill指针中指示数据页为1421，偏移量为2。根据指针内容，在数据页1421第2行中找到Gill对应的记录行。

特别理解
基于索引的数据查找过程

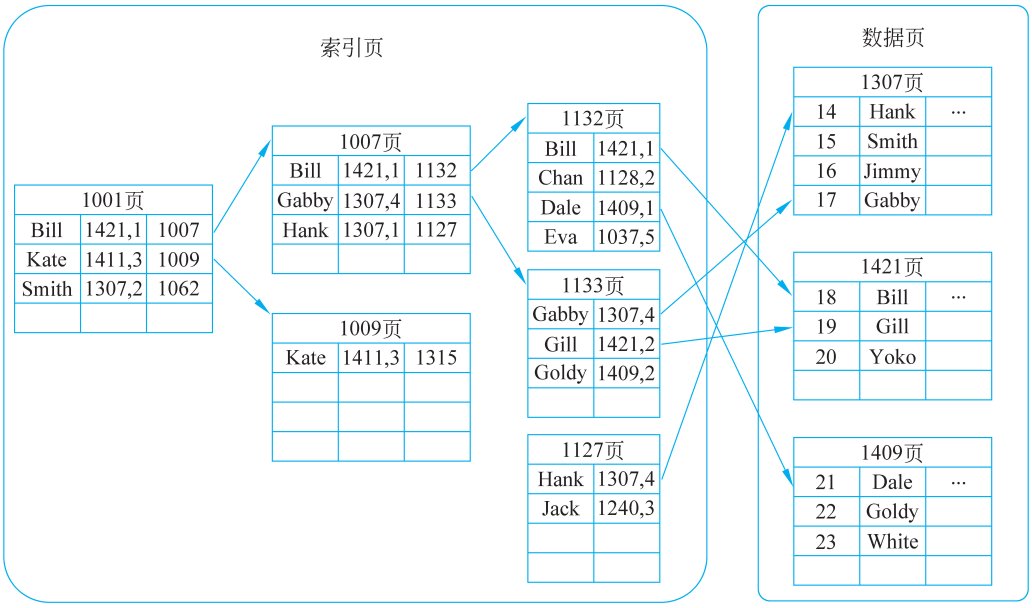

图 5-3 非聚簇索引结构

* 5.1.3 创建索引的策略

索引能够提高数据库的查询速度，但是这种时间效率上的提高都要付出代价。索引作为一个数据库对象，存储在数据库中。因此，存放索引需要占用一定的数据库存储空间。此外，当基本表进行插入、修改、删除操作时，需要维护索引，使其与新的数据保持一致，这会一定程度上增加数据库的负担。所以要根据实际应用需要，有选择地建立索引，原则就是使得建立索引带来的性能提高大于系统在存储空间和处理时间方面所付出的代价。

一般可以考虑在以下属性列上建立索引。

（1）在查询条件中常用的列上建立索引，可以加快查询速度。

（2）在连接条件中常出现的列上（通常是一些外键）建立索引，可以加快连接的速度。

（3）在经常使用排序的列上建立索引，利用索引的排序，可加快查询的速度。

（4）在经常需要搜索连续范围值的列上建立聚簇索引，找到第一个匹配行后，满足要求的后续行在物理上是连续且有序的。因此，只要将数据值直接与查找的终止值进行比较即可连续提取后续行。

如果有些列上建立索引没有明显提高查询速度，反而增加系统负担，那么这些列上不适合建立索引。

（1）在查询中很少用到的列上不应该建立索引。

（2）在只有很少数据值的列上不应建立索引。如在学生表中查询所有男生的信息，结果集中的行占了所有行的很大比例，则在性别列上建立索引并不能明显提高查询速度。

（3）在修改性能远远大于检索性能的列上不应该建立索引，因为增加索引时会降低

增加、删除和更新行的速度,即降低修改性能。

讨论思考

(1) 案例 5-1 中在学号属性列上建立索引后如何提高了查询效率?

(2) 以查询学生选课信息为例,在外码列上建立索引后如何加快学生表和选课表的连接速度?

5.2 常用的索引操作

教学视频
5.2 教学微视频

尽管索引的创建并非 SQL 标准,但是大部分商用数据库管理系统都提供了索引机制,用于在关系中创建索引。SQL Server 中同样提供了索引的创建、更新和删除操作。一般,创建和删除索引由数据库管理员或者表的属主(Owner)完成。RDBMS 会在执行查询操作时自动选择合适的索引作为存取路径,这个过程对于用户是透明的(索引结构称为存取路径,因为它们提供了定位和存取数据的一条路径)。

5.2.1 索引的创建及使用

1. 创建索引

T-SQL 中,建立索引使用 CREATE INDEX 语句,其语法格式:

```
CREATE [UNIQUE][CLUSTERED|NONCLUSTERED] INDEX <索引名> ON <表名>(<列>[<次序>][,<列名>[<次序>]]…)
```

其中的参数说明如下。

(1) UNIQUE 表明要创建的是唯一索引。

注意:只有当数据本身具有唯一特征时,建立唯一索引才有意义。如果必须要实施唯一性来确保数据的完整性,则应在列上建立唯一约束,而不要创建唯一索引。当在表上创建唯一约束时,系统会自动在该列建立唯一索引。

(2) CLUSTERED 表示要创建的是聚簇索引,若无显式声明,系统默认创建的是非聚簇索引。

(3) <表名>是要创建索引的基本表的名称。

(4) 索引可以建立在一个列上,也可以建立在多个列上,各<列名>之间用逗号分隔。建立在多个列上的索引称为复合索引。

(5) <次序>指定索引值的排序方式,包括 ASC(升序)和 DESC(降序),默认为 ASC。

【案例 5-2】 在学生表的学号上建立唯一的聚簇索引,按照升序排列。

```
CREATE UNIQUE CLUSTERED INDEX SNO_Index ON 学生(学号)
```

【案例 5-3】 在教师表中按照职称建立非聚簇索引，按照职称降序排列。

```
CREATE INDEX Tzc_Index ON 教师(职称 DESC)
```

【案例 5-4】 在教师任课表按课程代码升序和工号升序建立唯一索引。

```
CREATE UNIQUE INDEX TC_Index ON 教师任课(课程代码,工号)
```

注意：复合索引中，系统按照索引列出现的先后顺序对索引项排序。例如案例 5-4 中，先按照课程代码的值升序排列，课程代码相同时则按照工号的值升序排列。

特别理解
建立在组合列上的唯一索引限制

* 2. 索引的查看与使用

1）查看索引

查看指定表的索引信息，可通过以下语句执行存储过程 SP_HELPINDEX：

```
EXEC SP_HELPINDEX <表名>
```

该语句的执行结果可返回指定表上所有索引的名称、类型和建立索引的列。具体要查看某个索引的统计信息（如索引名称、统计密度信息和统计直方图信息等），可通过 SSMS 在“对象资源管理器”窗口中展开指定表中的统计信息结点，右击某个索引，从弹出的快捷菜单中选择“属性”命令，从弹出的“统计信息属性”窗口中可看到该索引的统计信息。当然也可以执行以下语句：

```
DBCC SHOW_STATISTICS(表名, 索引名)
```

该命令也可以用于查看指定表中某个索引的统计信息。

2）聚簇索引与非聚簇索引的比较

聚簇索引可以决定数据的物理存储顺序，而非聚簇索引则和数据的物理顺序无关。

【案例 5-5】 在学生表的学号属性上建立非聚簇索引和建立聚簇索引后分别插入新的元组(BX23230,测试,男,软件工程,null,null)，观察学生表中新元组所在的位置，更好地理解聚簇索引。

如图 5-4 所示，学生表中原有 3 条记录，在学生表的学号属性上建立非聚簇索引后，插入新元组(BX23230,测试,男,软件工程,null,null)，新元组在学生表中最后一行。

当在学生表的学号属性上建立的是聚簇索引时，表中原有元组和插入的新元组都自动按照学号的大小顺序重新排序，结果如图 5-5 所示。

注意：在数据表频繁更新的列上，不适于创建聚簇索引。因为这可能导致数据的重新排序和移动。

5.2.2 索引的更新与删除

* 1. 更新索引

随着数据库中数据的不断插入、修改和删除，索引对系统的优化性能将会出现降低。

图 5-4　在非聚簇索引的表中插入数据

图 5-5　在聚簇索引的表中插入数据

此时数据库管理员应该定期对索引进行分析和更新。如图 5-1 所示，索引中每一个叶子结点为一页，每页不可分割，而每页能存储的数据行数是有限制的，当新插入行或者更新行使得叶子结点无法容纳时，结点就需要进行页拆分(Page Split)，拆分过程中就会产生碎片。可能会出现有的页上只有一条记录，有的页上有几条记录，即一个页上保存的数据量远远达不到饱和。碎片如果过多很容易造成空间的浪费，并且增加数据查询的额外开销。极端情况下，例如，系统为了获取 10 行记录，可能需要加载 10 个页，而不是一个页。

知识拓展

对有索引的表操作时会影响操作性能

如果想检查索引因分页产生的碎片程度，可以使用 SQL Server 提供的 DBCC SHOWCONTIG 命令。在 SQL Server 的查询分析器中输入以下命令：

```
USE database_name
DECLARE @table_id int
SET @table_id=object_id ('表名')
DBCC SHOWCONTIG (@table_id)
```

命令执行后返回的参数中，经常查看 Scan Density [Best Count：Actual Count]（扫描密度［最佳计数：实际计数］），这是扩展盘区的最佳计数和实际计数的比率，该值应尽可能靠近 100%，低了则说明有碎片。若需要维护索引，处理方式有两种：①利用 DBCC INDEXDEFRAG 整理索引碎片，即 DBCC INDEXDEFRAG('数据库名','表名','索引名')；②利用 DBCC DBREINDEX 重建索引。前者是对表或者视图上的索引和非聚簇索引进行碎片整理，但是在重组织数据方面没有聚簇索引的重建更加有效。常用的重建索引语法格式：

```
DBCC DBREINDEX('表名', 索引名, 填充因子)
```

其中，第一个参数可以是表名，也可以是表的 ID。第二个参数可通过索引名指定某个索引，也可省去索引名，用('')代替，表示重建所有索引。填充因子是指索引页的数据填充程度，通常为 90%。如果是 100%，表示每一个索引页全部填满，此时查询效率最高，但是插入新索引值时，就得移动后面的所有页效率低。

2. 删除索引

索引建立后，由系统使用和维护。建立索引是为了提高查询速度，但如果数据增、删、改频繁，系统将花费很多时间维护索引，从而降低了查询效率。因此，可以删除一些不必要的索引。SQL Server 中提供了索引的删除功能。删除索引使用 DROP INDEX 语句，其语法格式：

```
DROP INDEX <索引名>ON <表名或视图名>
```

或者

```
DROP INDEX <表名或视图名>.<索引名>
```

说明： DROP INDEX 语句不能删除通过 PRINARY KEY 约束和 UNIQUE KEY 约束创建的索引。

【案例 5-6】 删除教师表中教师职称列上的 Tzc_Index 索引。

```
DROP INDEX Tzc_Index ON 教师
```

删除索引时，系统会同时从数据字典中删除该索引的相关描述。

讨论思考

（1）在数据表的唯一索引列上能插入空值吗？

（2）理解并体会聚簇索引与非聚簇索引有哪些区别。

5.3 视图基本知识

在数据库的三级模式结构中，模式是数据库中全体数据逻辑结构和特征的描述。当不同的用户或应用需要使用基本表中的不同数据，甚至一些经过聚集函数或表达式计算的值时，可以为其建立外模式。外模式中的数据来自于模式，是模

教学视频
5.3 教学微视频