第5章 索引和视图

在 MySQL 数据库中，索引（Index）是影响数据性能的重要因素之一，设计高效、合理的索引可以显著提高数据信息的查询速度和应用程序的性能。

视图（View）是一个存储指定查询语句的虚拟表，视图中的数据来源于由定义视图所引用的表，并且能够实现动态引用，即表中的数据发生变化，视图中的数据随之变化。

本章将介绍索引和视图等数据库对象的基本概念与常用操作。

5.1 索引

索引是由数据库表中的一列或多列组合而成的一种特殊的数据库结构，利用索引可以快速查询数据库表中的特定记录信息。在 MySQL 中，所有的数据类型都可以被创建索引。

5.1.1 理解索引

MySQL 中的索引是为了加速对数据进行检索而创建的一种分散的、物理的数据结构。索引包含从表中生成的键，以及映射到指定数据行的存储位置指针。索引是依赖于表建立的，提供了数据库表中存储、管理数据的内部方法。表的存储由两部分组成，一部分是表的数据页面，另一部分是存放索引的索引页面。

数据库中索引的形式与图书的目录相似，键值就像目录中的标题，指针相当于页码。索引的功能就像图书目录能为读者提供快速查找图书页面内容一样，不必扫描整个数据表而找到想要的数据行。

由此可知，当 MySQL 数据库在执行一条查询语句的时候，默认的执行过程是根据搜索条件进行全表扫描，遇到匹配条件的就加入搜索结果集合。如果查询语句涉及多个表连接，包括了许多搜索条件（例如大小比较、like 匹配等），而且表数据量较大，在没有索引的情况下，MySQL 需要扫描的数据块数量会很大，查询速度自然会很慢。

索引一旦创建，将由数据库自动管理和维护。例如，向表中插入、更新和删除一条记录时，数据库会自动在索引中做出相应的修改。在编写 SQL 查询语句时，具有索引的表与不具有索引的表没有任何区别，索引只是提供一种快速访问指定记录的方法。

在实际过程中，当 MySQL 执行查询时，查询优化器会对可用的多种数据检索方法的成本进行估计，从中选择最有效的查询计划。

在数据库中使用索引的优点如下。

（1）加速数据检索：索引能够以一列或多列值为基础实现快速查找数据行。

（2）优化查询：查询优化器是依赖于索引起作用的，索引能够加速连接、排序和分组等操作。

（3）强制实施行的唯一性：通过给列创建唯一性索引，可以保证表中的数据不重复。

需要注意的是，索引并不是越多越好，用户要正确认识索引的重要性和设计原则，创建合适的索引。

5.1.2 索引的分类

按照分类标准的不同，MySQL 的索引有多种分类形式。

MySQL 的索引通常包括普通索引（index）、主键索引（primary key）、唯一性索引（unique）、全文索引（fulltext）和空间索引（spatial）等类型。

（1）普通索引（index）：索引的关键字是 index。普通索引是 MySQL 中的基本索引类型，允许在定义索引的列中插入重复值和空值。

（2）主键索引（primary key）：一种特殊的唯一性索引，不允许有空值。一般在建表的时候创建主键索引，也可以通过修改表的方法增加主键，但一个表只能有一个主键索引。

（3）唯一性索引（unique）：unique 索引中列的值必须唯一，允许有空值。如果是组合索引，则列值的组合必须唯一。在一个表上可以创建多个 unique 索引。

（4）全文索引（fulltext）：全文索引是指在定义索引的列上支持值的全文查找，允许在这些索引列中插入重复值和空值。在默认情况下，该索引对于中文作用不大。

（5）空间索引（spatial）：空间索引是对空间数据类型的字段建立的索引。对于初学者来说，这类索引很少会用到。

按照创建索引键值的列数分类，索引可以分为单列索引和复合索引。

按照存储方式分类，MySQL 的索引分为 B-Tree 索引和 Hash 索引。

目前许多主流数据库管理系统（例如 Oracle、SQL Server、MySQL 等），都是将 B-Tree 索引作为最主要的索引类型的，主要是因为 B-Tree 索引的存储结构在数据库的数据检索中有着非常优异的表现。

MySQL 的 MEMORY 数据引擎还支持 Hash 索引。Hash 索引相对于 B-Tree 索引，检索效率要高不少。但 MySQL Hash 索引本身的特殊性也带来了很多限制和弊端，主要有以下内容。

- 仅能满足=、in 和<=>查询，不能使用范围查询。
- 无法被用来避免数据的排序操作。
- 不能利用部分索引键查询。
- 在任何时候都不能避免表扫描。
- 遇到大量 Hash 值相等的情况后，性能并不一定会比 B-Tree 索引高。

需要说明的是，在 MySQL 中运算符“ <=>”除了能够像常规的“=”运算符一样，对两个值进行比较以外，还能够用于比较 null 值。

- 运算符“ <=>”和“=”的相同点：像常规的“=”运算符一样，两个值进行比较，结果是 0（不相等）或 1（相等）。例如，'A'<=>'B'的值为 0，'a'<=>'a'的值为 1。
- 运算符“ <=>”和“=”的不同点：和“=”运算符不同的是，null 与常量进行比较运算，其值直接处理为 null。在使用“ <=>”运算符时，例如，'a' <=> null 的值为

0，null<=> null 的值为 1，相当于 'a' is null 和 null is null，而'a' is not null 相当于 not('a' <=> null)。

5.1.3 设置索引的原则

在数据表中创建索引，为了使索引的使用效率更高，必须考虑在哪些字段上创建索引和创建什么类型的索引。首先要了解以下常用的基本原则：

（1）一个表创建大量索引会影响 insert、update 和 delete 语句的性能，因此应避免对经常更新的表创建过多的索引，要限制索引的数目。

（2）若表的数据量较大，对表数据的更新较少而查询较多，可以通过创建多个索引来提高性能。在包含大量重复值的列上创建索引，查询的时间会较长。

（3）经常需要进行排序、分组和联合操作的字段一定要建立索引，即在用于 join、where 判断和 order by 排序的字段上创建索引。

（4）不要在数据库中某个含有大量重复的值的字段上建立索引，在这样的字段上建立索引有可能降低数据库的性能。对于不再使用或者很少使用的索引要及时删除。

（5）在主键上创建索引，在 InnoDB 中如果通过主键来访问数据，效率是非常高的。每个表只能创建一个主键索引。

（6）InnoDB 数据引擎的索引键最长支持 767 字节，MyISAM 数据引擎支持 1000 字节。


视频讲解

5.1.4 创建索引

创建索引通常有 3 种命令方式，即使用 create index 语句创建索引、通过修改表来创建索引和使用 alter table 语句创建索引。当然，利用 MySQL Workbench 等工具也可以用可视化方式创建索引。下面将详细讲解这些创建索引的方法。

1. 使用 create index 语句创建索引

如果基表已经创建完毕，就可以使用 create index 语句建立索引。

创建索引的基本形式如下：

```
create [unique|fulltext|spatial] index index_name
on table_name (index_col_name,…)
```

说明：

（1）create index：创建索引的关键字。

（2）unique|fulltext|spatial：创建索引的类型。unique 是唯一性索引，fulltext 是全文索引，spatial 是空间索引。

（3）index_name：索引名。索引名可以不写，若不写索引名，则默认索引名与列名相同。

（4）on table_name：创建索引对应的表。

（5）index_col_name：索引列名 index_col_name 的格式如下。

```
col_name [(length)] [asc | desc]
```

在创建索引时，可以使用 col_name[(length)]语法对前缀编制索引。前缀包括每列值的前 length 个字符。对于 char 和 varchar 列，只用一列的一部分就可以创建索引。blob 和 text 列也可以编制索引，但是必须给出前缀长度。因为多数名称的前 10 个字符通常不同，所以前缀索引不会比使用列的全名创建的索引的速度慢很多。

另外，使用列的一部分创建索引可以使索引文件大大减小，从而节省了大量的磁盘空间，有可能提高 insert 操作的速度。

【例 5.1】 为便于按电话进行查询，在 student 表的 phone 列上建立一个升序普通索引 phone_index。

代码和运行结果如下：

```
mysql> use teaching;
       Database changed
mysql> create index phone_index  on student(phone asc);
       Query OK, 0 rows affected (2.80 sec)
       Records: 0  Duplicates: 0  Warnings: 0
```

【例 5.2】 在 course 表的 cname 列上建立一个唯一性索引 cname_index。

代码和运行结果如下：

```
mysql> create  unique  index  cname_index on course(cname);
       Query OK, 0 rows affected (0.72 sec)
       Records: 0  Duplicates: 0  Warnings: 0
```

【例 5.3】 在 score 表的 studentno 和 courseno 列上建立一个复合索引 sc_index。

代码和运行结果如下：

```
mysql> create  index  sc_index  on  score(studentno,courseno);
       Query OK, 0 rows affected (0.73 sec)
       Records: 0  Duplicates: 0  Warnings: 0
```

2. 在创建表时创建索引

在创建表时可以直接创建索引，这种方式最简单、方便。

【例 5.4】 在 teacher1 表的 tname 字段上建立一个唯一性索引 tname_index，一个前缀索引 dep_index。

代码和运行结果如下：

```
mysql> use mytest;
       Database changed
mysql> create table if not exists teacher1 (
    -> teacherno  char(6) not null comment '教师号',
    ->  tname  char(8) not null comment '教师名',
    ->  major  char(10) not null comment '专业',
    ->  prof char(10) not null comment '职称',
    ->  department char(16) not null comment '部门',
    ->  primary key (teacherno),
    ->  unique  index  tname_index(tname),
```

```
    -> index dep_index(department(5))
    -> );
       Query OK, 0 rows affected (1.29 sec)
```

3. 通过 alter table 语句创建索引

【例 5.5】在 teacher1 表上建立 teacherno 主键索引（假定未创建主键索引），建立 tname 和 prof 的复合索引。

代码和运行结果如下：

```
mysql> alter table teacher1
    -> add primary key(teacherno),
    -> add index mark(tname, prof);
       Query OK, 0 rows affected (0.71 sec)
       Records: 0  Duplicates: 0  Warnings: 0
```

如果主键索引已经创建，则会出现如下信息：

```
ERROR 1068 (42000): Multiple primary key defined
```

说明：

（1）只有表的所有者才能给表创建索引。索引的名称必须符合 MySQL 的命名规则，且必须在表中是唯一的。

（2）主键索引必定是唯一的，唯一性索引不一定是主键。在一张表上只能有一个主键，但可以有一个或者多个唯一性索引。

（3）当给表创建 unique 约束时，MySQL 会自动创建唯一性索引。在创建唯一性索引时，应保证创建索引的列不包括重复的数据，并且没有两个或两个以上的空值（null）。因为在创建索引时将两个空值也视为重复的数据，如果有这种数据，必须先将其删除，否则索引不能被成功创建。

（4）要查看表中已经创建的索引的情况，可以使用 show index from table_name 语句实现。

5.1.5 删除索引

删除不再需要的索引，可以用 drop 语句，也可以用 alter table 语句。利用 drop index 语句删除索引的语法格式如下：

```
drop index index_name on table_name ;
```

例如，删除 teacher1 表的 mark 索引：

```
mysql> drop index mark on teacher1;
```

利用 alter table 语句删除索引的语法格式如下：

```
alter [ignore] table table_name
| drop primary key
| drop index index_name
```

```
| drop foreign key fk_symbol
```

用户也可以利用 alter table 语句删除前面表中创建的索引。例如：

```
mysql> alter table  teacher1  drop  index  mark;
```

说明：

（1）使用 drop index 子句可以删除各种类型的索引，包括唯一性索引。

（2）如果要删除主键索引，直接使用 drop primary key 子句进行删除，不需要提供索引名称，因为一个表中只有一个主键。

5.1.6 利用 MySQL Workbench 工具创建和管理索引

1. 利用 MySQL Workbench 创建索引

利用 MySQL Workbench 创建索引的步骤如下：

（1）启动 MySQL Workbench 工具，在导航区 Navigator 下的 SCHEMAS 区域选择当前数据库 teaching。

（2）在 teaching 数据库中选择 Tables，再展开 Tables 选项，右击 student 表，在弹出的快捷菜单中选择 Alter Table 命令，如图 5-1 所示。

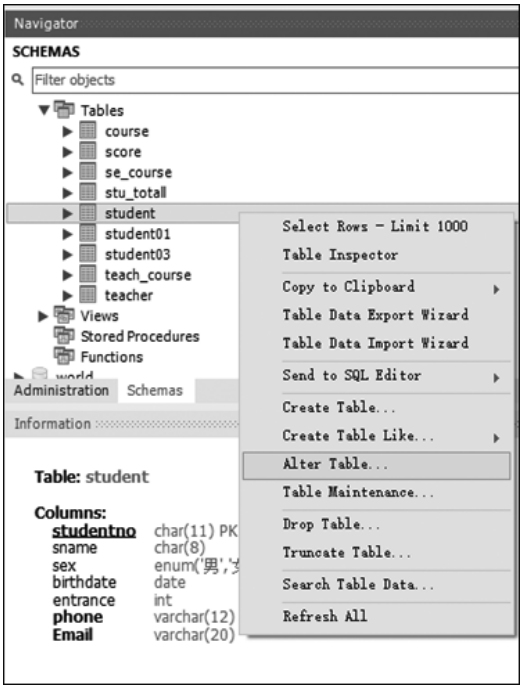

图 5-1　执行修改表命令

（3）进入修改表 student 的界面，如图 5-2 所示。选择 Indexes 选项卡，在如图 5-3 所示的界面中可以观察到如下信息。

① 数据库名和表名：指出创建索引的数据库 teaching 和表 student 的名称。

② 表 student 的默认字符集及排序规则和数据引擎 InnoDB 的索引名称。

③ 索引名 Index Name，可以查看到前面创建的主键索引 PRIMARY 和唯一性索引 phone_index。其后依次是索引类型 Type、索引引用字段 Index Columns、索引参数 Index Options 和索引注解 Index Comment 等。

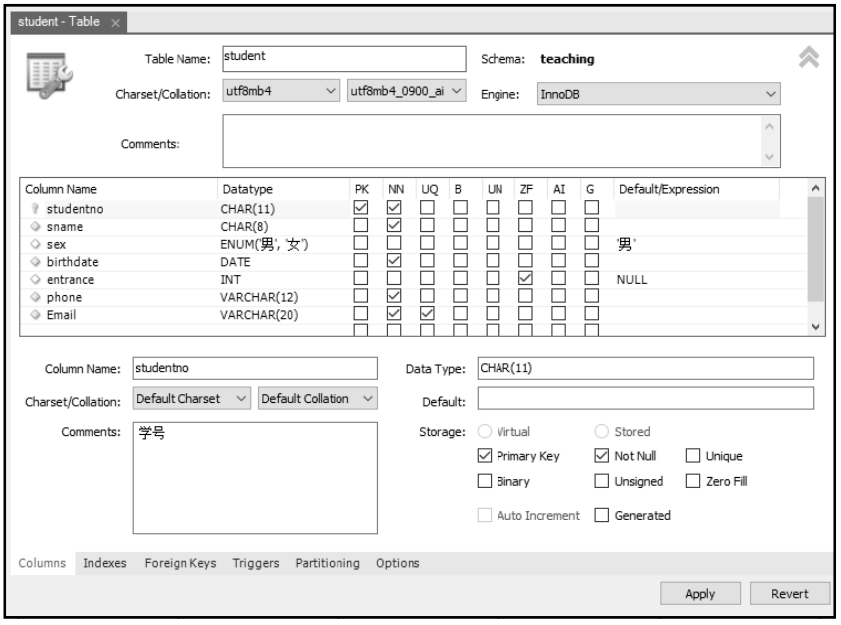

图 5-2 修改表界面

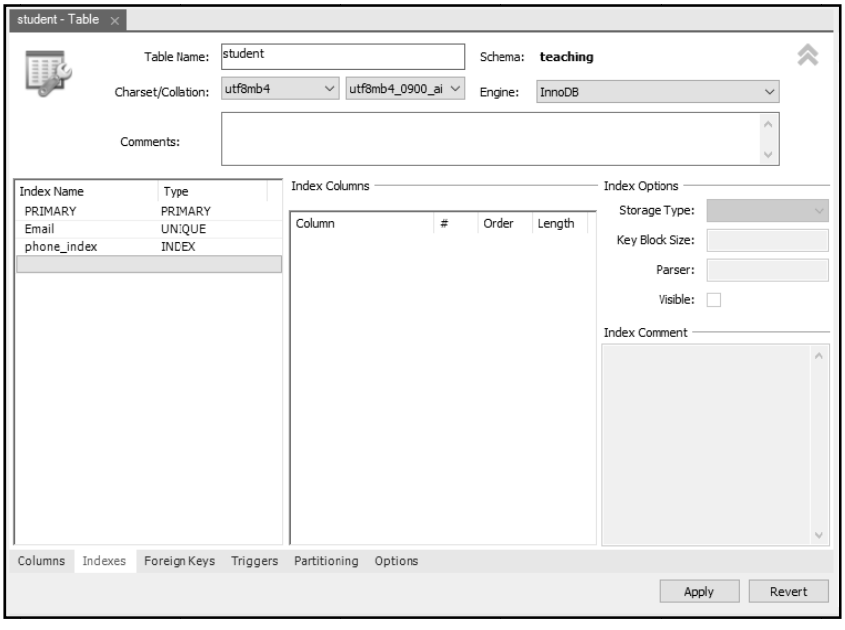

图 5-3 创建索引对话框

（4）在 Index Name 的文本框中输入索引名称 un_phone，右侧的 Index Columns 中会自动显示 student 表中的所有列名，选择 phone 列，存储类型选择 BTREE，选择索引类型

UNIQUE，按降序排列（DESC），表示创建唯一性索引，其他参数采用默认值，如图 5-4 所示。

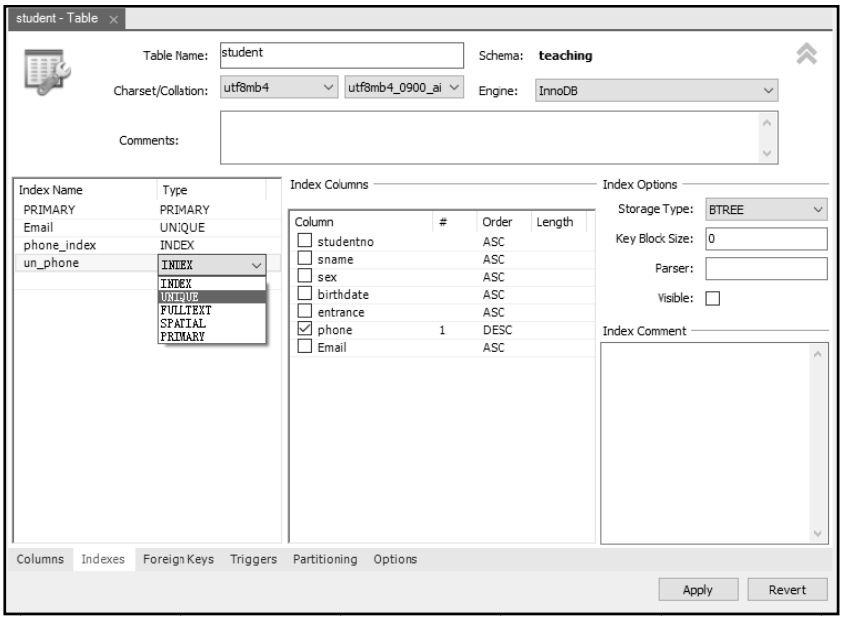

图 5-4　设置索引参数

（5）设置完成后，单击 Apply 按钮，出现如图 5-5 所示的应用脚本对话框。再单击 Apply 按钮，进入完成对话框，单击 Finish 按钮，即可完成数据库 teaching 中 student 表上的唯一性索引 un_phone 的创建。

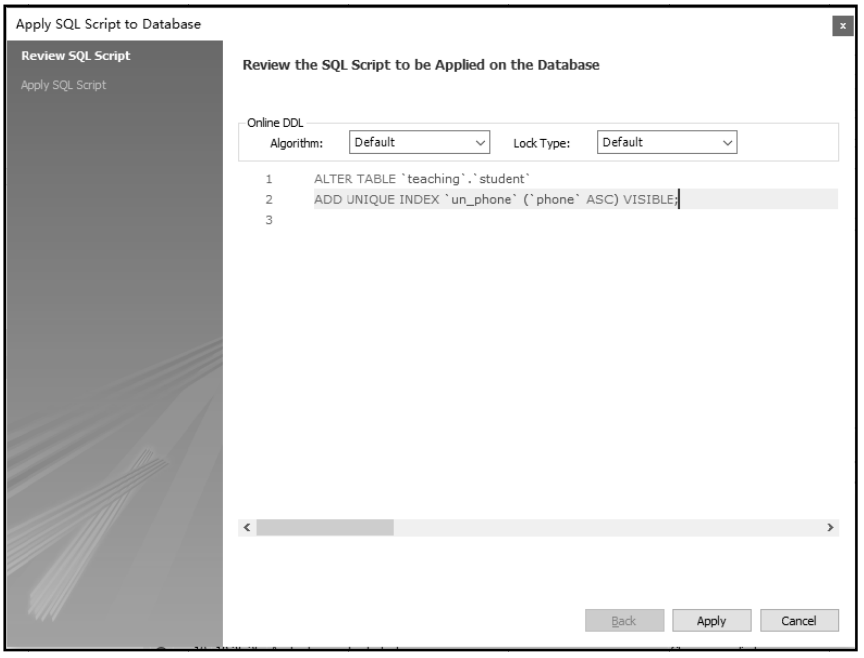

图 5-5　创建索引脚本

在 MySQL Workbench 中创建主键索引和普通索引的操作步骤基本相同。

2. 利用 MySQL Workbench 管理索引

利用 MySQL Workbench 管理索引的方法如下：

（1）利用 MySQL Workbench 可以修改索引的名字、类型以及索引引用字段和索引参数等。例如，修改 student 表中的 un_phone 索引为普通索引 un_phone_Email，将索引类型改为 INDEX，引用字段为 phone 和 Email，且为升序排列，如图 5-6 所示。单击 Apply 按钮，出现如图 5-7 所示的应用脚本对话框。再单击 Apply 按钮，进入完成对话框，如图 5-8 所

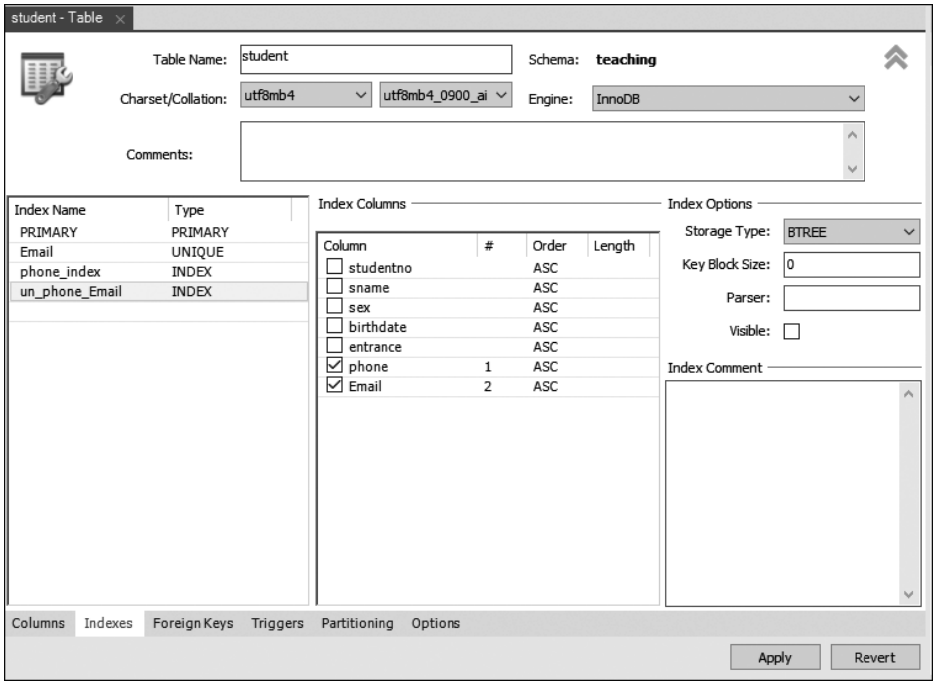

图 5-6　修改索引参数

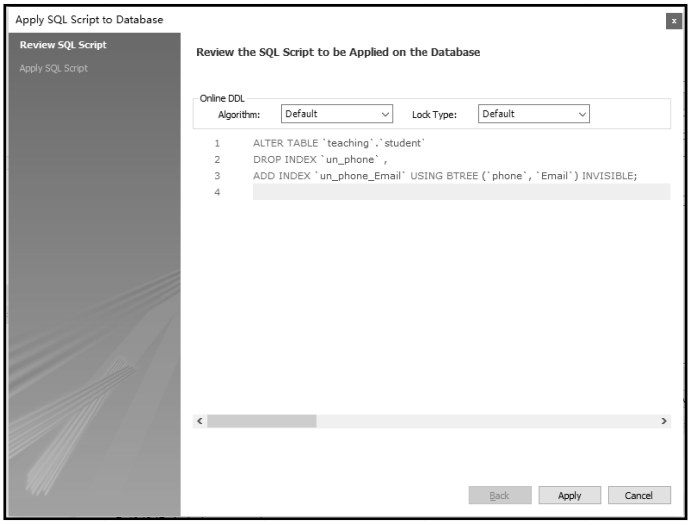

图 5-7　修改索引脚本

示，单击 Show Logs（Hide Logs）按钮，可以查看（隐藏）日志消息。单击 Finish 按钮，即可完成数据库 teaching 中 student 表上的索引 un_phone 的修改。

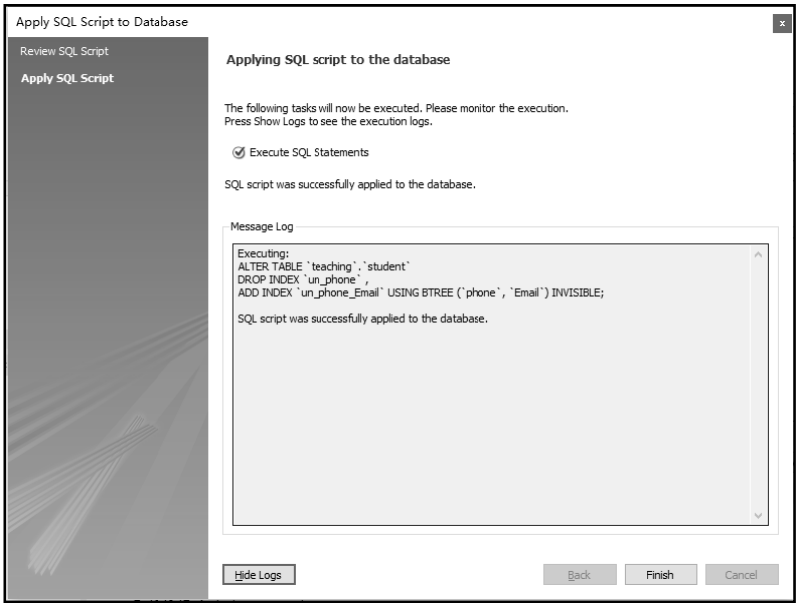

图 5-8　完成索引信息修改

（2）利用 MySQL Workbench 删除索引，例如删除普通索引 un_phone_Email。在索引界面中右击索引 un_phone_Email，执行 Delete Selected 命令，索引 un_phone_Email 即从列表中消失。单击 Apply 按钮，出现删除索引的应用脚本对话框。再单击 Apply 按钮，进入完成对话框。单击 Finish 按钮，即可删除索引 un_phone_Email。

5.2　视图的创建和管理

视图是从一个或者多个表及其他视图中通过 select 语句导出的虚拟表，在数据库中只存放了视图的定义，并没有存放视图中的数据。在浏览视图时所对应的行和列数据来自定义视图查询所引用的表，并且在引用视图时动态生成。通过视图可以实现对基表数据的查询与修改。

视图为数据库用户提供了很多便利，主要包括以下几方面。

（1）简化数据查询和处理：视图可以为用户集中多个表中的数据，简化用户对数据的查询和处理。

（2）屏蔽数据库的复杂性：数据库表的更改不影响用户对数据库的使用，用户也不必了解复杂数据库中的表结构。例如，那些定义了若干张表连接的视图，将表与表之间的连接操作对用户隐蔽起来。

（3）安全性：如果要使用户只能查询或修改用户有权限访问的数据，也可以只授予用户访问视图的权限，而不授予访问表的权限，这样就提高了数据库的安全性。