CHAPTER 第5章
MySQL 索引与完整性约束

当查阅书中某些内容时，为了提高查阅速度，并不是从书的第一页开始顺序查找，而是首先查看书的目录索引，找到需要的内容在目录中所列的页码，然后根据这一页码直接找到需要的内容。

在 MySQL 中，为了更加快速访问表中的数据，引入索引；而为防止不符合规范的数据进入数据库，MySQL 系统自动按一定的完整性约束条件对用户输入的数据进行监测，以确保数据库中存储的数据正确、有效、相容。

5.1 MySQL 索引

5.1.1 索引及作用

1. 索引

索引是根据表中一列或若干列按照一定顺序建立的列值与记录行之间的对应关系表。在列上创建了索引之后，查找数据时可以直接根据该列上的索引找到对应行的位置，从而快速地找到数据。

例如，如果用户创建了 xs 表中学号列的索引，MySQL 将在索引中排序学号列，对于索引中的每一项，MySQL 在内部为它保存一个数据文件中实际记录所在位置的“指针”。因此，如果要查找学号为“081241”的学生信息，MySQL 能在学号列的索引中找到“081241”的值，然后直接转到数据文件中相应的行，准确地返回该行的数据。在这个过程中，MySQL 只需处理一行就可以返回结果。如果没有“学号”列的索引，MySQL 则要扫描数据文件中的所有记录。

2. MySQL 索引

在 MySQL 5.6 中，所有的 MySQL 列类型都能被索引，但要注意以下几点：

(1) MySQL 能在多个列上创建索引。索引可以由最多 15 个列组成。最大索引长度是 256 个字节。

(2) 对于 CHAR 和 VARCHAR 列，可以索引列的前缀。这样索引的速度更快并且比索引列的全部内容需要较少的磁盘空间。

(3) 一个表最多可有 16 个索引。

(4) 只有当表类型为 MyISAM、InnoDB 或 BDB 时，才可以包含 NULL、BLOB 或 TEXT 类型的列添加索引。

5.1.2　索引的分类

索引是存储在文件中的，所以索引也是要占用物理空间的，MySQL 将一个表的索引都保存在同一个索引文件中。如果更新表中的一个值或者向表中添加或删除一行，MySQL 会自动地更新索引，因此索引树总是和表的内容保持一致。

1. BTREE 索引

目前大部分 MySQL 索引都是以 B-树(BTREE)方式存储的，索引类型分成下列几个。

1) 普通索引(INDEX)

这是最基本的索引类型，它没有唯一性之类的限制。创建普通索引的关键字是 INDEX。

2) 唯一性索引(UNIQUE)

这种索引和前面的普通索引基本相同，但有一个区别：索引列的所有值都只能出现一次，即必须是唯一的。创建唯一性索引的关键字是 UNIQUE。

3) 主键(PRIMARY KEY)

主键是一种唯一性索引，它必须指定为 PRIMARY KEY。主键一般在创建表的时候指定，也可以通过修改表的方式加入主键。但是每个表只能有一个主键。

4) 全文索引(FULLTEXT)

MySQL 支持全文检索和全文索引。全文索引的索引类型为 FULLTEXT。全文索引只能在 VARCHAR 或 TEXT 类型的列上创建，并且只能在 MyISAM 表中创建。它可以通过 CREATE TABLE 命令创建，也可以通过 ALTER TABLE 或 CREATE INDEX 命令创建。对于大规模的数据集，通过 ALTER TABLE(或 CREATE INDEX)命令创建全文索引要比把记录插入带有全文索引的空表更快。

2. 哈希索引(HASH)

当表类型为 MEMORY 或 HEAP 时，除了 BTREE 索引外，MySQL 还支持哈希索引(HASH)。使用哈希索引，不需要建立树结构，但是所有的值都保存在一个列表中，这个列表指向相关页和行。当根据一个值获取一个特定的行时，哈希索引非常快。

5.2　MySQL 索引创建

1. 用 CREATE INDEX 语句创建

使用 CREATE INDEX 语句可以在一个已有表上创建索引，一个表可以创建多个索引。

语法格式：

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引名
    [索引类型]
    ON 表名 (索引列名 ...)
    [索引选项] ...
索引列名=:
    列名 [(长度)] [ASC|DESC]
```

说明：

- UNIQUE | FULLTEXT | SPATIAL——UNIQUE 表示创建的是唯一性索引；FULLTEXT 表示创建全文索引；SPATIAL 表示为空间索引，可以用来索引几何数据类型的列（本书不讨论这种索引）。
- 索引名——索引在一个表中名称必须是唯一的。
- 索引类型——MySQL 支持的索引类型有 BTREE 和 HASH。
- 索引列名——创建索引的列名后的长度表示该列前面创建索引字符个数。这可使索引文件大大减小，从而节省磁盘空间。

BLOB 或 TEXT 列必须用前缀索引。前缀最长为 255B，但对于 MyISAM 和 InnoDB 表，最长达 1000B。

另外，还可以规定索引按升序（ASC）或降序（DESC）排列，默认为 ASC。如果一条 SELECT 语句中的某列按降序排列，那么在该列上定义一个降序索引可加快处理速度。

但是，CREATE INDEX 语句并不能创建主键。

【例 5.1】 根据 xs 表的学号列上的前 5 个字符建立一个升序索引 xh_xs。

```
use xscj
create index xh_xs
    on xs(学号(5) asc);
```

也可以在一个索引的定义中包含多个列，中间用逗号隔开，但它们属于同一个表，这样的索引叫做复合索引。

【例 5.2】 在 xs_kc 表的学号列和课程号列上建立一个复合索引 xskc_in。

```
create index xskc_in
    on xs_kc(学号,课程号);
```

2. 用 ALTER TABLE 语句创建

前面章节介绍了如何使用 ALTER TABLE 语句修改表，其中也包括向表中添加索引。

语法格式：

```
ALTER [IGNORE] TABLE 表名
…
    |ADD {INDEX|KEY} [索引名]                                          /*添加索引*/
        [索引类型] (索引列名…) [索引选项] …
    |ADD [CONSTRAINT [symbol]] PRIMARY KEY                            /*添加主键*/
        [索引类型] (索引列名…) [索引选项] …
    |ADD [CONSTRAINT [symbol]]UNIQUE [INDEX|KEY] [索引名]
        [索引类型] (索引列名…) [索引选项] …                            /*添加唯一性索引*/
    |ADD FULLTEXT  [INDEX|KEY] [索引名](索引列名…)[索引选项] …        /*添加全文索引*/
    |ADD SPATIAL  [INDEX|KEY] [索引名](索引列名…)[索引选项] …         /*添加空间索引*/
    |ADD [CONSTRAINT [symbol]] FOREIGN KEY [索引名] (索引列名…) [参照性定义]
                                                                      /*添加外键*/
```

```
    |DISABLE KEYS
    |ENABLE KEYS
```

说明：

- 索引类型——语法格式为 USING {BTREE|HASH}。

当定义索引时默认索引名，则一个主键的索引叫做 PRIMARY，其他索引使用索引的第一个列名作索引名。如果存在多个索引的名字以某一个列的名字开头，就在列名后面放置一个顺序号码。

- CONSTRAINT [symbol]——为主键、UNIQUE 键、外键定义一个名字。这个将在 5.3 节中介绍。
- DISABLE KEYS | ENABLE KEYS：只在 MyISAM 表中有用，使用 ALTER TABLE...DISABLE KEYS 可以让 MySQL 在更新表时停止更新 MyISAM 表中的非唯一索引，然后使用 ALTER TABLE ... ENABLE KEYS 重新创建丢失的索引，这样可以极大地加快查询速度。

【例 5.3】 在 xs 表的姓名列上创建一个非唯一的索引。

```
alter table xs
    add index xs_xm using btree (姓名);
```

【例 5.4】 以 xs 表为例(假设表中主键未定)，创建这样的索引，以加速表的检索速度：

```
alter table xs
    add index mark(出生日期,性别);
```

这个例子创建了一个复合索引。

如果想要查看表中创建的索引的情况，可以使用 SHOW INDEX FROM 表名语句，例如：

```
show index from xs;
```

系统显示已创建的索引信息如图 5.1 所示。

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
xs	0	PRIMARY	1	学号	A	21	NULL	NULL		BTREE		
xs	1	xh_xs	1	学号	A	21	5	NULL		BTREE		
xs	1	xs_xm	1	姓名	A	21	NULL	NULL		BTREE		
xs	1	mark	1	出生日期	A	21	NULL	NULL		BTREE		
xs	1	mark	2	性别	A	21	NULL	NULL		BTREE		

图 5.1　例 5.4 执行结果

3. 在建立表时创建索引

在前两种情况下,索引都是在表建立之后创建的。索引也可以在创建表时一起创建。在创建表的 CREATE TABLE 语句中可以包含索引的定义。

语法格式:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] 表名
    [([列定义], ...|[索引定义])]
    [表选项] [select 语句];
```

索引定义:

```
[CONSTRAINT [symbol]]PRIMARY KEY [索引类型] (索引列名 ...)              /* 主键 */
|{INDEX|KEY} [索引名] [索引类型](索引列名 ...)                          /* 索引 */
|[CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [索引名] [索引类型] (索引列名 ...)
                                                                        /* 唯一性索引 */
|[FULLTEXT|SPATIAL] [INDEX|KEY] [索引名] (索引列名 ...)                 /* 全文索引 */
|[CONSTRAINT [symbol]] FOREIGN KEY [索引名] (索引列名 ...)[参照性定义] /* 外键 */
```

说明:KEY 通常是 INDEX 的同义词。在定义列选项的时候,也可以将某列定义为 PRIMARY KEY,但是当主键是由多个列组成的多列索引时,定义列时无法定义此主键,必须在语句最后加上一个 PRIMARY KEY(列名,...)子句。

【例 5.5】 在 mytest 数据库中创建成绩(cj)表,学号和课程号的联合主键,并在成绩列上创建索引。

```
use mytest
create table xs_kc
(
    学号        char(6) not null,
    课程号      char(3) not null,
    成绩        tinyint(1),
    学分        tinyint(1),
    primary key(学号,课程号),
    index cj(成绩)
);
```

说明:使用“SHOW INDEX FROM 表名”命令查看执行结果。

4. 删除索引

当一个索引不再需要的时候,可以用 DROP INDEX 语句或 ALTER TABLE 语句删除它。

1) 使用 DROP INDEX 删除

语法格式:

```
DROP INDEX 索引名 ON 表名
```

2）使用 ALTER TABLE 删除

语法格式：

```
ALTER [IGNORE] TABLE 表名
…
|DROP PRIMARY KEY                          /* 删除主键 */
|DROP {INDEX|KEY} 索引名                   /* 删除索引 */
|DROP FOREIGN KEY fk_symbol                /* 删除外键 */
```

其中，DROP {INDEX|KEY}子句可以删除各种类型的索引。使用 DROP PRIMARY KEY 子句时不需要提供索引名称，因为一个表中只有一个主键。

【例 5.6】 删除 xs 表上的 mark 索引。

```
alter table xs
    drop index mark;
```

读者可使用 SHOW INDEX FROM 表名语句查看执行结果。

如果从表中删除了列，索引可能会受影响。如果所删除的列为索引的组成部分，则该列也会从索引中删除。如果组成索引的所有列都被删除，则整个索引都将被删除。

5.3　MySQL 数据完整性约束

保持数据库的数据完整性是 DBMS 最为重要的功能之一。数据完整性包括数据的一致性和正确性。完整性约束（或简称“约束”）是数据库的内容必须随时遵守的规则。

完整性约束的声明对于一个表的可能值做出了限制，可以通过 CREATE TABLE 或 ALTER TABLE 语句定义几个完整性约束。例如，对于每一列，可以声明 NOT NULL，这意味着不允许为空值，即列必须填充，在前面章节已经讨论了这种约束。除此之外，MySQL 还有其他各种不同类型的完整性约束，下面将系统介绍。

5.3.1　主键约束

主键就是表中的一列或多个列的一组，其值能唯一地标志表中的每一行。通过定义 PRIMARY KEY 约束来创建主键，而且 PRIMARY KEY 约束中的列不能取空值。由于 PRIMARY KEY 约束能确保数据的唯一，所以经常用来定义标志列。当为表定义 PRIMARY KEY 约束时，MySQL 为主键列创建唯一性索引，实现数据的唯一性，在查询中使用主键时，该索引可用来对数据进行快速访问。如果 PRIMARY KEY 约束是由多列组合定义的，则某一列的值可以重复，但 PRIMARY KEY 约束定义中所有列的组合值必须唯一。

可以用两种方式定义主键，作为列或表的完整性约束。作为列的完整性约束时，只需在列定义的时候加上关键字 PRIMARY KEY，这一点在 3.2.1 节中已做过介绍。作为表的完整性约束时，需要在语句最后加上一条 PRIMARY KEY(col_name,...)语句。

【例 5.7】 创建表 xs1，将姓名定义为主键。

```
create table xs1
(
    学号     varchar(6) null,
    姓名     varchar(8) not null primary key ,
    出生日期 datetime
);
```

说明：例中主键定义于空指定之后，空指定也可以在主键之后指定。

当表中的主键为复合主键时，只能定义为表的完整性约束。

【例 5.8】 创建 course 表来记录每门课程的学生学号、姓名、课程号、学分和毕业日期。其中学号、课程号和毕业日期构成复合主键。

```
create table course
(
    学号         varchar(6)    not null,
    姓名         varchar(8)    not null,
    毕业日期     date          not null,
    课程号       varchar(3) ,
    学分         tinyint ,
    primary key (学号, 课程号, 毕业日期)
);
```

原则上，任何列或者列的组合都可以充当一个主键。但是主键列必须遵守下列一些规则：

(1) 每个表只能定义一个主键。

(2) MySQL 可以创建一个没有主键的表。但是，从安全角度应该为每个基础表指定一个主键。主要原因在于：若没有主键，则可能在一个表中存储两个相同的行，这两个行由于不能彼此区分，在查询中，它们满足同样的条件，在更新时也总是一起更新，这样可能会导致数据库崩溃。

(3) 表中的两个不同的行在主键上不能具有相同的值，即所谓的“唯一性规则”。

(4) 如果从一个复合主键中删除一列后，剩下的列构成的主键仍然满足唯一性原则，那么，这个复合主键是不正确的，这条规则称为“最小化规则”(Minimality Rule)。也就是说，复合主键不应包含任何不必要的列。

MySQL 自动地为主键创建一个索引。通常，这个索引名为 PRIMARY。然而，可以重新给这个索引取名。

【例 5.9】 创建的 course 表，同时创建主键索引，索引命名为 index_course。

```
create table course
(
    学号         varchar(6)    not null,
```

```
    姓名          varchar(8)     not null,
    毕业日期      date           not null,
    课程号        varchar(3),
    学分          tinyint ,
    primary  key    index_course(学号, 课程号, 毕业日期)
);
```

5.3.2　替代键约束

在关系模型中，替代键像主键一样，是表的一列或一组列，它们的值在任何时候都是唯一的。替代键是没有被选作主键的候选键。定义替代键的关键字是 UNIQUE。

【例 5.10】 在表 xs1 中将姓名列定义为一个替代键。

```
create table xs1
(
    学号          varchar(6) null,
    姓名          varchar(8) not null unique,
    出生日期      datetime null,
    primary key(学号)
);
```

说明：关键字 unique 表示“姓名”是一个替代键，其列值必须是唯一的。

替代键还可以定义为表的完整性约束，故前面的语句也可定义如下：

```
create table xs1
(
    学号          varchar(6) null,
    姓名          varchar(8) not null,
    出生日期      datetime null,
    primary key(学号),
    unique(姓名)
);
```

在 MySQL 中，替代键和主键的区别主要有以下 3 点：

(1) 一个数据表只能创建一个主键。但一个表可以有若干个 UNIQUE 键，并且它们甚至可以重合，例如，在 C_1 和 C_2 列上定义了一个替代键，并且在 C_2 和 C_3 上定义了另一个替代键，这两个替代键在 C_2 列上重合了，而 MySQL 允许这样。

(2) 主键字段的值不允许为 NULL，而 UNIQUE 字段的值可取 NULL，但是必须使用 NULL 或 NOT NULL 声明。

(3) 一般创建 PRIMARY KEY 约束时，系统会自动产生 PRIMARY KEY 索引。创建 UNIQUE 约束时，系统自动产生 UNIQUE 索引。

通过 PRIMERY KEY 约束和 UNIQUE 约束可实现表的所谓实体完整性约束。定义为 PRIMERY KEY 和 UNIQUE KEY 的列上都不允许出现重复的值。

5.3.3 参照完整性约束

在本书所例举的 xscj 数据库中，有很多规则是和表之间的关系有关的。例如，存储在 xs_kc 表中的所有学号必须同时存在于 xs 表的学号列中。xs_kc 表中的所有课程号也必须出现在 kc 表的课程号列中。这种类型的关系就是“参照完整性约束”。参照完整性约束是一种特殊的完整性约束，表现为一个外键。所以 xs_kc 表中的学号列和课程号列都可以定义为一个外键。可以在创建表或修改表时定义一个外键声明。

定义外键的语法格式已经在介绍索引时给出了，这里列出“参照性定义”。

语法格式：

参照性定义：

```
REFERENCES 表名 [(索引列名 ...)]
    [ON DELETE {RESTRICT|CASCADE|SET NULL|NO ACTION}]
    [ON UPDATE {RESTRICT|CASCADE|SET NULL|NO ACTION}]
```

索引列名：

```
列名 [(长度)] [ASC|DESC]
```

说明：

(1) 外键被定义为表的完整性约束，参照性定义中包含了外键所参照的表和列，还可以声明参照动作。这里表名叫做被参照表。而外键所在的表叫做参照表。

其中，列名是外键可以引用一个或多个列，外键中的所有列值在引用的列中必须全部存在。外键可以只引用主键和替代键。外键不能引用被参照表中随机的一组列，它必须是被参照表的列的一个组合，且其中的值都保证是唯一的。

(2) ON DELETE|ON UPDATE：可以为每个外键定义参照动作。

参照动作包含两部分：

在第一部分中，指定这个参照动作应用哪一条语句。这里有两条相关的语句，即 UPDATE 和 DELETE 语句；

在第二部分中，指定采取哪个动作。可能采取的动作是 RESTRICT、CASCADE、SET NULL、NO ACTION 和 SET DEFAULT。

接下来说明这些不同动作的含义。

- RESTRICT：当要删除或更新父表中被参照列上在外键中出现的值时，拒绝对父表的删除或更新操作。
- CASCADE：从父表删除或更新行时自动删除或更新子表中匹配的行。
- SET NULL：当从父表删除或更新行时，设置子表中与之对应的外键列为 NULL。如果外键列没有指定 NOT NULL 限定词，这就是合法的。
- NO ACTION：NO ACTION 意味着不采取动作，就是如果有一个相关的外键值在被参考的表里，删除或更新父表中主要键值的企图不被允许，和 RESTRICT 一样。
- SET DEFAULT：作用和 SET NULL 一样，只不过 SET DEFAULT 是指定子表中的外键列为默认值。

如果没有指定动作，两个参照动作就会默认使用 RESTRICT。

外键目前只可以用在那些使用 InnoDB 存储引擎创建的表中，对于其他类型的表，MySQL 服务器能够解析 CREATE TABLE 语句中的 FOREIGN KEY 语法，但不能使用或保存它。

【例 5.11】 创建 xs1 表，所有的 xs 表中学生学号都必须出现在 xs1 表中，假设已经使用学号列作为主键创建了 xs 表。

```
create table xs1
(
    学号        varchar(6) null,
    姓名        varchar(8) not null,
    出生日期    datetime null,
    primary key (姓名),
    foreign key (学号)
        references xs (学号)
            on delete restrict
            on update restrict
);
```

说明：在这条语句中，定义一个外键的实际作用是，在这条语句执行后，确保 MySQL 插入到外键中的每一个非空值都已经在被参照表中作为主键出现过。

这意味着，对于 xs1 表中的每一个学号，都执行一次检查，看这个号码是否已经出现在 xs 表的学号列(主键)中。如果情况不是这样，用户或应用程序会接收到一条出错消息，并且更新被拒绝。这也适用于使用 UPDATE 语句更新 xs1 表中的学号列。即 MySQL 确保了 xs1 表中的学号列的内容总是 xs 表中学号列的内容的一个子集。也就是说，下面的 SELECT 语句不会返回任何行：

```
select *
    from xs1
    where 学号 not in
            (select 学号
                from xs
            );
```

当指定一个外键的时候，以下的规则适用：

(1) 被参照表必须已经用一条 CREATE TABLE 语句创建了，或者必须是当前正在创建的表。在后一种情况下，参照表是同一个表。

(2) 必须为被参照表定义主键。

(3) 必须在被参照表的表名后面指定列名(或列名的组合)。这个列(或列组合)必须是这个表的主键或替代键。

(4) 尽管主键是不能够包含空值的，但允许在外键中出现一个空值。这意味着，只要外键的每个非空值出现在指定的主键中，这个外键的内容就是正确的。

(5) 外键中的列的数目必须和被参照表的主键中的列的数目相同。

(6) 外键中的列的数据类型必须和被参照表的主键中的列的数据类型对应相等。