18 外键是一个非常特殊的存在

更新时间：2020-04-13 09:23:48

> 衡量一个人的真正品格，是看他在知道没人看见的时候干些什么。——孟德斯鸠

关于外键应用的印象，可能大多还停留在书本中，因为你只要一提到外键，就会有人告诉你：企业级开发中禁用外键，别问为什么，我们都是这样做的。之后，这种思想逐渐沉淀下来，对于外键，也越来越“疏远”。但是，外键真的那么不好吗 ？如果不好，为什么 MySQL 在版本更新中不把这项复杂的功能去掉呢 ？那么，这一节里，我们将会详细的探讨这些问题，揭开外键神秘的外纱。

1. 聊一聊外键

一项技术或者功能的出现，一定会有它的原因、它的需求。这里，我们将要去讨论下 MySQL 为什么会创造外键，它有怎样的作用。之后，再去从理论层面讲解外键的概念，让你掌握外键的思想。最后，谈一谈外键的优缺点，你也就能够发现，在哪些情况下确实不适合使用外键。

1.1 外键的前世今生

谈到外键的前世今生，我们先去考虑一个业务场景：学生一定会属于某一个班级，那么，在创建学生表（存储学生的信息）时，是不是要把班级信息一块放进去呢 ？你一定知道，学生和班级信息放在一张表中肯定是不合适的，这会造成大量的数据冗余，难以维护。正确的解决方案是创建学生表和班级表，并让这两张表之间建立某种联系。而实现这种联系的技术方案就是外键，这也就是外键诞生的需求。

虽然外键功能非常强大，MySQL 也对它做了很多优化，但是，外键的使用频率并不高。甚至在很多企业中，都是直接禁用外键。所以，你也知道了，外键的今生过的不好。

1.2 外键的概念

简单的说，外键就是表中存在一个字段指向另一个表的主键，那么，这个字段就被称之为外键。外键有很多属性，它们也都非常重要，总结如下：

- 两个关联的表中，主键所在的表被称为母表，外键所在的表被称为子表
- 外键可以为空值（NULL），如果不为空，则外键值必须等于另一个表的主键值
- 一张表可以有多个外键
- 如果子表中有相应的外键值，母表不可以随意删除或更新这个字段

外键对应的是参照完整性，用于约束表与表之间的关系。可以说，外键是表之间的映射关系，这个关系可以帮助我们处理表之间的紧密性和存在性。所以，你需要知道，外键的核心思想是约束。

1.3 外键的优缺点

外键存在的目的就是让 MySQL 去管理表与表之间的关系，否则，就需要我们用代码去管理。但是，有利就一定会有弊，下面，我们一起来看看外键的优势与劣势。

外键的优势

- 由数据库来保证数据的一致性、完整性，使程序的逻辑更加简单
- 对于存在外键关系的表，可以使用客户端生成可读性更好的 ER 图

外键的劣势

- 外键会额外的占据存储空间
- 外键会使数据库对数据的管理更加复杂，在操作时会降低性能
- 母表数据出错或者丢失，正确的数据迁移和数据恢复几乎成为不可能的事

所以，根据这里提到的外键存在的优势与劣势，可以得出结论：外键保证了数据的完整性，但是会降低性能且对故障恢复难度太大。最后，关于应不应该使用外键，我给出几点建议：

- 对于性能要求不高，但是安全性要求高的系统，使用外键；反之，不使用
- 对于表数据量特别大的系统，例如上千万行，不使用
- 对于偏小的业务系统，也没有很大的流量，建议使用外键

2. 外键的操作

通过以上对外键的解读、分析，你应该理解了外键，知道了外键的好与不好。接下来，我们去看一看在 MySQL 中应该怎样操作外键（这并不简单，你需要好好的思考和理解）。

2.1 增加外键

应用、修改或删除外键之前，一定得先有外键。但是，在讲解怎么给表增加外键之前，我们得先要知道外键自身的一些约束条件：

- 母表必须已经建好在数据库中

- 外键关联的一定是母表的主键
- 外键列的个数必须和母表主键列个数相同（考虑到一张表的主键是多个列的组合）
- 外键列的数据类型必须和母表主键列的数据类型一致

给数据表增加外键有两种情况，一种是在创建表时指定，另一种是给已经存在的表添加。在创建表时指定外键使用 `FOREIGN KEY` 关键字，语法如下：

```
[CONSTRAINT <外键名>] FOREIGN KEY 字段名 [，字段名2，…]
REFERENCES <母表名> 主键列1 [，主键列2，…]
```

其中：“外键名”是定义的外键的名称，与索引名是类似的，同时，MySQL 不允许一个表中存在相同的外键名；“字段名”标识子表添加外键约束的数据列；“母表名”即与子表存在外键关联的表；“主键列”标识母表中的主键列。

另外，在建立外键时，可以指定 `ON UPDATE <action>` 和 `ON DELETE <action>` 子句来标识发生 UPDATE 和 DELETE 操作时，子表和母表的数据应该如何处理。MySQL 支持四种约束条件：

- RESTRICT：MySQL 的默认约束条件，禁用母表中的更新和删除操作
- NO ACTION：与 RESTRICT 含义相同
- CASCADE：更新或删除母表记录时，自动更新或删除子表中对应的记录，即级联
- SET NULL：更新或删除母表记录时，将子表中对应记录外键设置为 NULL，前提是子表外键列是允许 NULL 值的

好的，既然已经知道了增加外键的语法和约束条件。我们就以学生表和班级表的例子来构造表之间的外键约束，建表 SQL 语句如下：

```
-- 班级表作为母表，所以，需要先创建出来
CREATE TABLE class(
 id int(11) AUTO_INCREMENT COMMENT '主键 id',
 info varchar(256) NOT NULL COMMENT '班级信息',
 PRIMARY KEY(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 学生表建表语句
CREATE TABLE student(
 id int(11) AUTO_INCREMENT COMMENT '主键 id',
 name varchar(64) NOT NULL COMMENT '姓名',
 c_id int(11) NOT NULL COMMENT '班级 id',
 PRIMARY KEY(`id`),
 CONSTRAINT `s_class_id` FOREIGN KEY(`c_id`) REFERENCES `class`(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

执行以上建表语句，就在 student 表中添加了名称为“s_class_id”的外键约束，外键列的名称是 c_id，依赖于表 class 的主键（注意使用默认的约束条件）。那么，假如在创建 student 表时没有指定外键约束，又该怎么增加外键呢？也就是第二种情况了。我们也来看一看它的语法：

```
ALTER TABLE <数据表名> ADD CONSTRAINT <外键名>
FOREIGN KEY(<列名>) REFERENCES <主表名> (<列名>);
```

两种增加外键的语法几乎是一样的，所以，我这里不做重复说明，直接给出一个在建表之后增加外键的示例 SQL 语句：

```
ALTER TABLE `student` ADD CONSTRAINT `s_class_id` FOREIGN KEY(`c_id`) REFERENCES `class`(`id`);
```

2.2 删除外键

外键还有一个很特殊的性质，你在创建外键之后，就不能再去修改了。这也很好理解，毕竟外键维护两张表数据之间的完整性，修改外键会增加数据库的大量验证工作。所以，当你确实想要修改外键时，只能先删除，再增加（不过这个成本依然很高，谨慎使用）。删除外键的语法如下：

```
ALTER TABLE 表名 DROP FOREIGN KEY 外键名;
```

我们可以尝试将之前 student 表中的外键约束删除，执行如下 SQL 语句：

```
mysql> ALTER TABLE `student` DROP FOREIGN KEY `s_class_id`;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

需要注意，删除外键使用的是外键的名称，而不是外键列字段的名称。此时的你可能会产生疑问，如果我在增加外键的时候没有指定外键名（外键名是可选的），我应该怎么办呢？继续看下面的内容吧。

2.3 查看外键的引用关系

接着上面的话题继续说，我怎样知道表中的外键名称是什么呢？最简单的方式当然是查看下表的定义语句，可以看到关于表的最完整信息。如下所示：

```
-- SHOW CREATE TABLE 可以查看表的定义语句
mysql> SHOW CREATE TABLE student\G
*************************** 1. row ***************************
       Table: student
Create Table: CREATE TABLE `student` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '主键 id',
  `name` varchar(64) NOT NULL COMMENT '姓名',
  `c_id` int(11) NOT NULL COMMENT '班级 id',
  PRIMARY KEY (`id`),
  KEY `s_class_id` (`c_id`),
  CONSTRAINT `s_class_id` FOREIGN KEY (`c_id`) REFERENCES `class` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

另外，你可以验证下在增加外键时不指定外键的名称，MySQL 是否会帮你填充一个名称。我们通过这种方式可以查看一个表的外键关联信息，但是，如果我想反向查找这种关系呢？这种需求也很常见，当我们需要清理表的数据时，就需要知道当前有哪些子表与之关联。

这些系统元数据当然存储在系统库中（你应该能够想到这一点），如下所示：

```
mysql> SELECT
    ->   TABLE_NAME,
    ->   COLUMN_NAME,
    ->   CONSTRAINT_NAME,
    ->   REFERENCED_TABLE_NAME,
    ->   REFERENCED_COLUMN_NAME
    -> FROM
    ->   INFORMATION_SCHEMA.KEY_COLUMN_USAGE
    -> WHERE
    ->   CONSTRAINT_SCHEMA = 'imooc_mysql'
    ->   AND REFERENCED_TABLE_NAME = 'class';
+------------+-------------+-----------------+-----------------------+------------------------+
| TABLE_NAME | COLUMN_NAME | CONSTRAINT_NAME | REFERENCED_TABLE_NAME | REFERENCED_COLUMN_NAME |
+------------+-------------+-----------------+-----------------------+------------------------+
| student    | c_id        | s_class_id      | class                 | id                     |
+------------+-------------+-----------------+-----------------------+------------------------+
1 row in set (0.06 sec)
```

可以看到，通过查询 INFORMATION_SCHEMA 系统库中的 KEY_COLUMN_USAGE 表（属性列的含义比较简单，这里不过多说明）就得到了子表的相关信息。如果要做清理数据的工作，可以先删除掉子表中的外键约束，再去处理母表。

3. 外键的应用

外键约束很有用，它不仅仅能够维护多个表之间数据的完整性，而且还会让我们的编码工作简单许多（因为数据正确性不再需要使用代码去维护）。另外，外键还能够约束多种关系，接下来，我将以广告系统的应用来举例说明怎样使用外键。

3.1 一对一关系的应用

当一张表的数据列太多，数据量庞大的时候，最好的办法就是拆表，把一张表拆分成两张或多张表。但是，表记录之间需要有关联关系，要保证数据的正确、完整性。我这里以广告系统中的广告创意（物料）举例，首先，来看一看原表的定义：

```
-- 创意表建表 SQL 语句
CREATE TABLE `creative` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL DEFAULT '' COMMENT '创意名称',
  `type` varchar(20) NOT NULL DEFAULT 'image' COMMENT '类型',
  `width` int(11) NOT NULL DEFAULT '0' COMMENT '宽',
  `height` int(11) NOT NULL DEFAULT '0' COMMENT '高',
  `size` int(11) NOT NULL DEFAULT '0' COMMENT '大小，单位字节',
  `audit_status` varchar(10) NOT NULL DEFAULT 'pass' COMMENT '审核状态',
  `status` varchar(10) NOT NULL DEFAULT 'normal' COMMENT '创意状态',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '所属用户ID',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  `create_by` int(11) NOT NULL DEFAULT '0' COMMENT '创建者user_id',
  `update_by` int(11) NOT NULL DEFAULT '0' COMMENT '更新者user_id',
  `url` varchar(1024) NOT NULL DEFAULT '' COMMENT '物料地址',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=129 DEFAULT CHARSET=utf8 COMMENT='创意信息';
```

考虑到 creative 表数据列过多，我们可以把它拆分成两张表：creative（创意表）、creative_detail（创意详情表）。建表语句如下所示：

```
-- 创意详情表
CREATE TABLE `creative_detail` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` varchar(20) NOT NULL DEFAULT 'image' COMMENT '类型',
  `width` int(11) NOT NULL DEFAULT '0' COMMENT '宽',
  `height` int(11) NOT NULL DEFAULT '0' COMMENT '高',
  `size` int(11) NOT NULL DEFAULT '0' COMMENT '大小，单位字节',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '所属用户ID',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  `create_by` int(11) NOT NULL DEFAULT '0' COMMENT '创建者user_id',
  `update_by` int(11) NOT NULL DEFAULT '0' COMMENT '更新者user_id',
  `url` varchar(1024) NOT NULL DEFAULT '' COMMENT '物料地址',
  `preview_url` varchar(1024) NOT NULL DEFAULT '' COMMENT '预览地址',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=129 DEFAULT CHARSET=utf8 COMMENT='创意详情';

-- 创意表
CREATE TABLE `creative` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL DEFAULT '' COMMENT '创意名称',
  `audit_status` varchar(10) NOT NULL DEFAULT 'pass' COMMENT '审核状态',
  `status` varchar(10) NOT NULL DEFAULT 'normal' COMMENT '创意状态',
  `c_detail_id` int(11) NOT NULL COMMENT '与创意详情表关联',
  PRIMARY KEY (`id`),
  UNIQUE KEY `c_detail_id` (`c_detail_id`),
  CONSTRAINT `creative_ibfk_1` FOREIGN KEY (`c_detail_id`) REFERENCES `creative_detail` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=129 DEFAULT CHARSET=utf8 COMMENT='创意信息';
```

可以看到，除了外键约束之外（包括更新、删除的约束），我这里还使用了 `UNIQUE KEY` 去限制 `c_detail_id` 列值唯一，这就整体保证了 creative 与 creative_detail 一对一的关系。

3.2 一对多关系的应用

一对多关系是外键约束最基本的应用，我们之前所讲解的学生与班级的关系就是一对多：一个学生属于一个班级，而一个班级可以有很多学生。在广告系统中，这种一对多的关系是非常多的（其实在任何业务系统中都很多），例如：一个创意属于一个用户，但是一个用户可以有很多创意。

创意表直接使用刚刚所介绍的 creative（未做拆分的创意表）即可，我们还需要去创建一个用户表，建表语句如下所示：

```
-- 用户表
CREATE TABLE `user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_name` varchar(128) NOT NULL DEFAULT '' COMMENT '用户名',
  `user_status` int(10) NOT NULL DEFAULT '0' COMMENT '用户状态',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  `create_by` bigint(20) NOT NULL DEFAULT '0' COMMENT '创建者 user_id',
  `update_by` bigint(20) NOT NULL DEFAULT '0' COMMENT '更新者 user_id',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8 COMMENT='用户信息表';
```

创建用户表之后，我们给 creative 表增加外键约束即可，SQL 语句如下：

```
ALTER TABLE `creative` ADD CONSTRAINT `user_user_id` FOREIGN KEY(`user_id`) REFERENCES `user`(`id`) ON DELETE CASCADE ON UPDA
TE CASCADE;
```

3.3 多对多关系的应用

对于多对多的场景，我还是以创意表去解释说明。在广告系统中，推广单元代表一次广告的投放过程，所以，必然会关联创意，且可以关联多个创意（轮播广告）。但同时，创意同样可以应用在多个推广单元中去，这也很好理解。那么，创意与推广单元就构成了多对多的关系，即一个创意可以关联多个推广单元，而一个推广单元也可以关联多个创意。

对于多对多关系，我们必须要创建第三张表，用来专门记录两张表之间的联系。不管第三张表，我们先来把推广单元表创建出来（创意表仍然沿用之前未拆分的表）：

```
-- 推广单元表
CREATE TABLE `ad_unit` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `unit_name` varchar(48) NOT NULL COMMENT '推广单元名称：',
  `unit_state` varchar(24) NOT NULL DEFAULT 'ready' COMMENT '推广单元状态',
  `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '实际消费',
  `create_type` tinyint(4) NOT NULL DEFAULT '0' COMMENT '推广单元创建类型',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='推广单元信息';
```

执行以上建表语句，我们就有了 creative 和 ad_unit 表，且它们之间目前还没有任何联系。好的，重点来了，我们需要去创建第三张表实现关联了。建表语句如下：

```
CREATE TABLE `creative_2_ad_unit` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '主键 id',
  `creative_id` int(11) NOT NULL,
  `ad_unit_id` bigint(20) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `creative_2_ad_unit_ibfk_1` FOREIGN KEY (`creative_id`) REFERENCES `creative` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `creative_2_ad_unit_ibfk_2` FOREIGN KEY (`ad_unit_id`) REFERENCES `ad_unit` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='创意与推广单元多对多关联表';
```

可以看到，creative_2_ad_unit 表中包含了两个外键约束，分别与 creative 和 ad_unit 表的主键相关联。以此，实现了两张表的多对多关系。同时，这也是一张表中存在多个外键约束的经典应用。

4. 总结

外键很特殊，它虽然非常好用、非常有价值，但是却被隐藏的很深。那么，到底要不要在业务系统中使用外键呢？这其实是个很泛的问题，更多的是看你对外键的理解。我在这一节中讲解的三个外键使用案例你需要去认真思考，搞清楚它们分别解决了什么问题。以便将来，在遇到类似的问题时（相信我，这概率很大），能够做到从容不迫、得心应手。

5. 问题

你在工作中使用过外键吗？是怎么使用的呢？

对于外键的 UPDATE、DELETE 约束，试一试它们会对母表和子表产生怎样的影响？

根据你的理解，谈一谈你觉得外键适用的场景？

6. 参考资料

《高性能 MySQL（第三版）》

《高性能 MySQL（第三版）》

MySQL 官方文档：FOREIGN KEY Constraints

MySQL 官方文档：FOREIGN KEY Constraint Differences

MySQL 官方文档：Using Foreign Keys

MySQL 官方文档：Foreign Key Optimization

}

17 分区表是什么，又该怎么使用呢？

19 听过存储过程，但是你会用吗？