MySQL学习笔记（Day010：Employees/临时表的创建/外键约束）

MySQL学习

一. Employees数据库安装

1. Employees数据库介绍

`Employees` 数据库是一个用于学习和测试的数据库，大约 `160MB` ， `4百万条记录`

2. Employees的安装

> 官方安装文档

2.1. 下载

根据官方文档的连接，我们可以找到下载该数据库的两种方式

- employees_db-full-1.0.6.tar.bz2
- github-test_db 使用 `git clone` 进行仓库拉取

2.2. 解压和拉取

```
[root@MyServer Downloads]> tar jxf employees_db-full-1.0.6.tar.bz2
# 老师讲解时说是解压有问题，但是我这里却可以解压
[root@MyServer Downloads]> cd employees_db
[root@MyServer employees_db]> ll
total 164492
-rw-r--r--. 1 501 games       752 Mar 30  2009 Changelog
-rw-r--r--. 1 501 games      6460 Oct  9  2008 employees_partitioned2.sql
-rw-r--r--. 1 501 games      7624 Feb  6  2009 employees_partitioned3.sql
-rw-r--r--. 1 501 games      5660 Feb  6  2009 employees_partitioned.sql
-rw-r--r--. 1 501 games      3861 Nov 28  2008 employees.sql # 主要文件
-rw-r--r--. 1 501 games       241 Jul 30  2008 load_departments.dump
-rw-r--r--. 1 501 games  13828291 Mar 30  2009 load_dept_emp.dump
-rw-r--r--. 1 501 games      1043 Jul 30  2008 load_dept_manager.dump
-rw-r--r--. 1 501 games  17422825 Jul 30  2008 load_employees.dump
-rw-r--r--. 1 501 games 115848997 Jul 30  2008 load_salaries.dump
-rw-r--r--. 1 501 games  21265449 Jul 30  2008 load_titles.dump
-rw-r--r--. 1 501 games      3889 Mar 30  2009 objects.sql
-rw-r--r--. 1 501 games      2211 Jul 30  2008 README
-rw-r--r--. 1 501 games      4455 Mar 30  2009 test_employees_md5.sql
-rw-r--r--. 1 501 games      4450 Mar 30  2009 test_employees_sha.sql


[root@MyServer github_employees]> git clone https://github.com/datacharmer/test_db.git
Initialized empty Git repository in /root/Downloads/github_employees/test_db/.git/
Cloning into 'test_db'...
remote: Counting objects: 94, done.
remote: Total 94 (delta 0), reused 0 (delta 0), pack-reused 94
Unpacking objects: 100% (94/94), done.
Checking connectivity... done.

[root@MyServer test_db]> ll
total 168340
-rw-r--r--. 1 root root      964 Dec  2 21:25 Changelog
-rw-r--r--. 1 root root     7948 Dec  2 21:25 employees_partitioned_5.1.sql
-rw-r--r--. 1 root root     6276 Dec  2 21:25 employees_partitioned.sql
-rw-r--r--. 1 root root     4193 Dec  2 21:25 employees.sql  # 主要文件
drwxrwxr-x. 2 root root     4096 Dec  2 21:25 images
-rw-r--r--. 1 root root      250 Dec  2 21:25 load_departments.dump
-rw-r--r--. 1 root root 14159880 Dec  2 21:25 load_dept_emp.dump
-rw-r--r--. 1 root root     1090 Dec  2 21:25 load_dept_manager.dump
-rw-r--r--. 1 root root 17722832 Dec  2 21:25 load_employees.dump
-rw-r--r--. 1 root root 39806034 Dec  2 21:25 load_salaries1.dump
-rw-r--r--. 1 root root 39805981 Dec  2 21:25 load_salaries2.dump
-rw-r--r--. 1 root root 39080916 Dec  2 21:25 load_salaries3.dump
-rw-r--r--. 1 root root 21708736 Dec  2 21:25 load_titles.dump
-rw-r--r--. 1 root root     4568 Dec  2 21:25 objects.sql
-rw-r--r--. 1 root root     4007 Dec  2 21:25 README.md
drwxrwxr-x. 2 root root     4096 Dec  2 21:25 sakila
-rw-r--r--. 1 root root      272 Dec  2 21:25 show_elapsed.sql
-rwxr-xr-x. 1 root root     1800 Dec  2 21:25 sql_test.sh
-rw-r--r--. 1 root root     4878 Dec  2 21:25 test_employees_md5.sql
-rw-r--r--. 1 root root     4882 Dec  2 21:25 test_employees_sha.sql
```

解压出来的主要文件大小是不一样的，且根据网页上发布和更新的时间上看，github中的时间比较新，所以这里使用github中源作为的安装文件

2.3. 安装

```
[root@MyServer test_db]> mysql -uroot -S /tmp/mysql.sock_57 -p < employees.sql
Enter password:
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
data_load_time_diff
00:01:51
```

2.4. 验证

```
[root@MyServer test_db]> time mysql -uroot -S /tmp/mysql.sock_57 -p -t < test_employees_sha.sql
Enter password:
+----------------------+
| INFO                 |
+----------------------+
| TESTING INSTALLATION |
+----------------------+
+--------------+------------------+------------------------------------------+
| table_name   | expected_records | expected_crc                             |
+--------------+------------------+------------------------------------------+
| employees    |           300024 | 4d4aa689914d8fd41db7e45c2168e7dcb9697359 |
| departments  |                9 | 4b315afa0e35ca6649df897b958345bcb3d2b764 |
| dept_manager |               24 | 9687a7d6f93ca8847388a42a6d8d93982a841c6c |
| dept_emp     |           331603 | d95ab9fe07df0865f592574b3b33b9c741d9fd1b |
| titles       |           443308 | d12d5f746b88f07e69b9e36675b6067abb01b60e |
| salaries     |          2844047 | b5a1785c27d75e33a4173aaa22ccf41ebd7d4a9f |
+--------------+------------------+------------------------------------------+
+--------------+------------------+------------------------------------------+
| table_name   | found_records    | found_crc                                |
+--------------+------------------+------------------------------------------+
| employees    |           300024 | 4d4aa689914d8fd41db7e45c2168e7dcb9697359 |
| departments  |                9 | 4b315afa0e35ca6649df897b958345bcb3d2b764 |
| dept_manager |               24 | 9687a7d6f93ca8847388a42a6d8d93982a841c6c |
| dept_emp     |           331603 | d95ab9fe07df0865f592574b3b33b9c741d9fd1b |
| titles       |           443308 | d12d5f746b88f07e69b9e36675b6067abb01b60e |
| salaries     |          2844047 | b5a1785c27d75e33a4173aaa22ccf41ebd7d4a9f |
+--------------+------------------+------------------------------------------+
+--------------+---------------+-----------+
| table_name   | records_match | crc_match |
+--------------+---------------+-----------+
| employees    | OK            | ok        |
| departments  | OK            | ok        |
| dept_manager | OK            | ok        |
| dept_emp     | OK            | ok        |
| titles       | OK            | ok        |
| salaries     | OK            | ok        |
+--------------+---------------+-----------+
+------------------+
| computation_time |
+------------------+
| 00:00:18         |
+------------------+
+---------+--------+
| summary | result |
+---------+--------+
| CRC     | OK     |
| count   | OK     |
+---------+--------+

real    0m19.406s
user    0m0.005s
sys     0m0.004s


[root@MyServer test_db]> time mysql -uroot -S /tmp/mysql.sock_57 -p -t < test_employees_md5.sql
Enter password:
+----------------------+
| INFO                 |
+----------------------+
| TESTING INSTALLATION |
+----------------------+
+--------------+------------------+----------------------------------+
| table_name   | expected_records | expected_crc                     |
+--------------+------------------+----------------------------------+
| employees    |           300024 | 4ec56ab5ba37218d187cf6ab09ce1aa1 |
| departments  |                9 | d1af5e170d2d1591d776d5638d71fc5f |
| dept_manager |               24 | 8720e2f0853ac9096b689c14664f847e |
| dept_emp     |           331603 | ccf6fe516f990bdaa49713fc478701b7 |
| titles       |           443308 | bfa016c472df68e70a03facafa1bc0a8 |
| salaries     |          2844047 | fd220654e95aea1b169624ffe3fca934 |
+--------------+------------------+----------------------------------+
+--------------+------------------+----------------------------------+
| table_name   | found_records    | found_crc                        |
+--------------+------------------+----------------------------------+
| employees    |           300024 | 4ec56ab5ba37218d187cf6ab09ce1aa1 |
| departments  |                9 | d1af5e170d2d1591d776d5638d71fc5f |
| dept_manager |               24 | 8720e2f0853ac9096b689c14664f847e |
| dept_emp     |           331603 | ccf6fe516f990bdaa49713fc478701b7 |
| titles       |           443308 | bfa016c472df68e70a03facafa1bc0a8 |
| salaries     |          2844047 | fd220654e95aea1b169624ffe3fca934 |
+--------------+------------------+----------------------------------+
+--------------+---------------+-----------+
| table_name   | records_match | crc_match |
+--------------+---------------+-----------+
| employees    | OK            | ok        |
| departments  | OK            | ok        |
| dept_manager | OK            | ok        |
| dept_emp     | OK            | ok        |
| titles       | OK            | ok        |
| salaries     | OK            | ok        |
+--------------+---------------+-----------+
+------------------+
| computation_time |
+------------------+
| 00:00:16         |
+------------------+
+---------+--------+
| summary | result |
+---------+--------+
| CRC     | OK     |
| count   | OK     |
+---------+--------+

real    0m18.452s
user    0m0.007s
sys     0m0.005s
```

至此， `Employees` 测试数据库就安装完成了

二. 表(TABLE)

1. 表的介绍

- 表是关系数据库的核心
- 表 = 关系
- 表是记录的集合
- 二维表格模型易于人的理解
- MySQL默认存储引擎都是基于行(记录)存储
- 每行记录都是基于列进行组织的

2. 表是数据的集合

```
select * from table_name limit 1;
```

集合是无序的，上面的SQL语句的意思是 **从表(集合)中** 随机 **选出一条数据，结果是不确定的**, 不能简单的认为是取出第一条数据。

```
select * from table_name order by col_name limit 1;
```

只有通过 `order by` 排序之后取出的数据，才是确定的。

3. 创建表

3.1. 临时表

> 官方文档 表创建的语法

- **临时表的创建**

```
--
--  mysql 5.7.9
--
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.9-log |
+-----------+
1 row in set (0.00 sec)

mysql> use burn_test;
Database changed

mysql> create temporary table temp_a(a int);
Query OK, 0 rows affected (0.01 sec)

mysql> show  create table temp_a\G
*************************** 1. row ***************************
       Table: temp_a
Create Table: CREATE TEMPORARY TABLE `temp_a` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4   -- 使用的是innodb
1 row in set (0.00 sec)

--
-- mysql 5.6.27
--
mysql> select version();
+------------+
| version()  |
+------------+
| 5.6.27-log |
+------------+
1 row in set (0.00 sec)

mysql> create temporary table  temp_a_56(a int);
Query OK, 0 rows affected (0.06 sec)

mysql> show create table temp_a_56\G
*************************** 1. row ***************************
       Table: temp_a_56
Create Table: CREATE TEMPORARY TABLE `temp_a_56` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  -- 5.6 也是innodb
1 row in set (0.00 sec)

-----
--
-- 终端1 MySQL 5.7.9
--

mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.9-log |
+-----------+
1 row in set (0.00 sec)

mysql> show processlist\G
*************************** 1. row ***************************
     Id: 10  -- 当前ID 是 10
   User: root
   Host: localhost
     db: burn_test
Command: Query
   Time: 0
  State: starting
   Info: show processlist   -- 当前终端执行
*************************** 2. row ***************************
     Id: 12
   User: root
   Host: localhost
     db: NULL
Command: Sleep
   Time: 328
  State:
   Info: NULL
*************************** 3. row ***************************
     Id: 13
   User: root
   Host: localhost
     db: burn_test
Command: Sleep
   Time: 16
  State:
   Info: NULL
3 rows in set (0.00 sec)

mysql> insert into temp_a values(123);
Query OK, 1 row affected (0.00 sec)

mysql> select * from temp_a;
+------+
| a    |
+------+
|  123 |
+------+
1 row in set (0.00 sec)

--
-- 终端2 MySQL 5.7.9
--
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.9-log |
+-----------+
1 row in set (0.00 sec)

mysql> show processlist\G
*************************** 1. row ***************************
     Id: 10
   User: root
   Host: localhost
     db: burn_test
Command: Sleep
   Time: 75
  State:
   Info: NULL
*************************** 2. row ***************************
     Id: 12
   User: root
   Host: localhost
     db: NULL
Command: Sleep
   Time: 403
  State:
   Info: NULL
*************************** 3. row ***************************
     Id: 13   -- 当前 ID 是 13
   User: root
   Host: localhost
     db: burn_test
Command: Query
   Time: 0
  State: starting
   Info: show processlist  -- 当前终端执行
3 rows in set (0.00 sec)

mysql> use burn_test;
Database changed

mysql> show tables;
Empty set (0.00 sec)   -- 从其他终端登录的用户(session)无法看到temp_a这个临时表

----
--
-- 临时表 和 普通表 同名问题
--
mysql> create table test_1 (a int);  -- 创建一张普通的表叫做 test_1
Query OK, 0 rows affected (0.16 sec)

mysql> insert into test_1 values(23);
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_1 values(24);
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_1;
+------+
| a    |
+------+
|   23 |   -- 可以看到插入的数据
|   24 |
+------+
2 rows in set (0.00 sec)

mysql> create temporary table test_1 (a int);   -- 创建一种和test_1 同名的临时表
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test_1 values(1000);  -- 插入一个 不一样的值
Query OK, 1 row affected (0.00 sec)

mysql> select * from test_1;
+------+
| a    |
+------+
| 1000 |   -- 只能搜索到 临时表中的数据
+------+
1 row in set (0.00 sec)

mysql> create temporary table if not exists table_name  (a int);  -- 使用if not exists进行判断
```

1：临时表是 `SESSION` 级别的，当前用户logout或者其他用户登录上来，是无法看到这张表的
2：当临时表和普通表同名时，当前用户只能看到同名的临时表
3：创建表时带上 `if not exists` 进行表的存在性检查；同时建议在临时表的表名前面加上统一的 `prefix`

- **临时表的作用**
  - 临时表主要的作用是给当前登录的用户存储临时数据或者临时结果的。
  - 不要和SQL优化器在排序过程中内部帮你创建的临时表相混淆。

- **临时表的存储引擎**

```
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.9-log |
+-----------+
1 row in set (0.00 sec)

mysql> show variables like "default%tmp%";
+-----------------------------+--------+
| Variable_name               | Value  |
+-----------------------------+--------+
| default_tmp_storage_engine | InnoDB |  -- 5.7.9 的临时表默认存储引擎就是 InnoDB
+-----------------------------+--------+
1 row in set (0.00 sec)


mysql> select version();
+------------+
| version()  |
+------------+
| 5.6.27-log |
+------------+
1 row in set (0.00 sec)

mysql> show variables like "default%tmp%";
+-----------------------------+--------+
| Variable_name               | Value  |
+-----------------------------+--------+
| default_tmp_storage_engine | InnoDB |  -- 5.7.26 的临时表默认存储引擎也是 InnoDB
+-----------------------------+--------+
1 row in set (0.00 sec)

-- 5.6 之前用的是MyISAM
```

- **临时表存储位置**

```
#
# MySQL 5.7
#
mysql> system ls -l /tmp  # 使用system 可以解析执行linux shell命令
total 20
drwxr-xr-x. 4 mysql mysql 4096 Dec  2 10:06 mysql_data
srwxrwxrwx. 1 mysql mysql    0 Dec  2 21:20 mysql.sock_56
srwxrwxrwx. 1 mysql mysql    0 Dec  2 20:51 mysql.sock_57
-rw-------. 1 mysql mysql    5 Dec  2 20:51 mysql.sock_57.lock
-rw-r-----. 1 mysql mysql 8554 Dec  2 22:04 #sqlf18_a_0.frm  -- temp_1 的表结构

mysql> system ls -l /data/mysql_data/5.7/ | grep ib
-rw-r-----. 1 mysql mysql       879 Dec  2 20:47 ib_buffer_pool
-rw-r-----. 1 mysql mysql  12582912 Dec  2 22:21 ibdata1
-rw-r-----. 1 mysql mysql 134217728 Dec  2 22:20 ib_logfile0
-rw-r-----. 1 mysql mysql 134217728 Dec  2 21:33 ib_logfile1
-rw-r-----. 1 mysql mysql  12582912 Dec  2 22:33 ibtmp1  -- 这个是我们的表结构对应的数据

mysql> show variables like "innodb_temp%";
+----------------------------+------------------------+
| Variable_name              | Value                  |
+----------------------------+------------------------+
| innodb_temp_data_file_path | ibtmp1:12M:autoextend |
+----------------------------+------------------------+
1 row in set (0.00 sec)

#-----

#
# MySQL 5.6
#
mysql> system ls -l /tmp
total 68
drwxr-xr-x. 4 mysql mysql  4096 Dec  2 10:06 mysql_data
srwxrwxrwx. 1 mysql mysql     0 Dec  2 21:20 mysql.sock_56
srwxrwxrwx. 1 mysql mysql     0 Dec  2 20:51 mysql.sock_57
-rw-------. 1 mysql mysql     5 Dec  2 20:51 mysql.sock_57.lock
-rw-rw----. 1 mysql mysql  8554 Dec  2 22:38 #sql13f3_7_0.frm   -- 表结构
-rw-rw----. 1 mysql mysql 49152 Dec  2 22:38 #sql13f3_7_0.ibd   -- 表数据

# 5.6.27 中没有 innodb_temp_data_file_path 变量
mysql> show variables like "innodb_temp%";
Empty set (0.00 sec)

mysql> show variables like "%innodb%temp%";
Empty set (0.00 sec)
```

> MySQL5.7.9 把临时 表结构 放在 tmpdir ，而数据 表数据 放在 datadir
> MySQL5.6.27 把临时 表结构 和 表数据 都放在 tmpdir

4. 查看表结构

```
mysql> show create table test_1\G   -- 表结构
*************************** 1. row ***************************
       Table: test_1
Create Table: CREATE TABLE `test_1` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
1 row in set (0.00 sec)

mysql> desc test_1\G  -- 表的描述,描述二维表信息
*************************** 1. row ***************************
  Field: a
   Type: int(11)
   Null: YES
    Key:
Default: NULL
  Extra:
1 row in set (0.00 sec)

mysql> show table status like "test_1"\G    -- 看表结构的元数据信息
*************************** 1. row ***************************
           Name: test_1
         Engine: InnoDB
        Version: 10
     Row_format: Dynamic
           Rows: 2
 Avg_row_length: 4096
    Data_length: 8192
Max_data_length: 0
   Index_length: 0
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2015-12-02 22:20:19
    Update_time: 2015-12-02 22:20:44
     Check_time: NULL
      Collation: utf8mb4_general_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

5. ALTER TABLE

> ALTER TABLE语法官方文档

```
mysql> alter table test_1 add column b char(10);  -- 添加列 b
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from test_1;
+------+------+
| a    | b    |
+------+------+
|   23 | NULL |
|   24 | NULL |
+------+------+
2 rows in set (0.00 sec)

mysql> alter table test_1 drop column b;  -- 删除列 b
Query OK, 0 rows affected (0.27 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from test_1;
+------+
| a    |
+------+
|   23 |
|   24 |
+------+
2 rows in set (0.00 sec)
```

> 注意，当表记录很大的时候， alter table 会很耗时，影响性能

- **ONLINE DDL**
  5.6以后对在线DDL操作进行了优化，以提高性能。官方文档

三. 外键约束

1. 外键的介绍

> 官方文档

```
--
-- 摘自 MySQL官方文档
--

CREATE TABLE product (          -- 商品表
    category INT NOT NULL,      -- 商品种类
    id INT NOT NULL,            -- 商品id
    price DECIMAL,
    PRIMARY KEY(category, id)  -- 主键是 (category, id)
)   ENGINE=INNODB;

CREATE TABLE customer (    -- 客户表
    id INT NOT NULL,       -- 客户id
    PRIMARY KEY (id)       -- 主键是 id
)   ENGINE=INNODB;

CREATE TABLE product_order (          -- 订单表
    no INT NOT NULL AUTO_INCREMENT,  -- number，自增长
    product_category INT NOT NULL,   -- 商品种类
    product_id INT NOT NULL,         -- 商品id
    customer_id INT NOT NULL,        -- 客户id

    PRIMARY KEY(no),                        -- 主键是  no
    INDEX (product_category, product_id),   -- 对 (product_category, product_id) 做索引
    INDEX (customer_id),                    -- 对 customer_id 做索引

    FOREIGN KEY (product_category, product_id)  -- 两个外键约束
    REFERENCES product(category, id)    -- 字段 product_category 引用自 product表的category
                                        -- 字段 product_id 引用自 product表的id
      ON UPDATE CASCADE ON DELETE RESTRICT,     -- 级联跟新  和  严格模式删除

    FOREIGN KEY (customer_id)
      REFERENCES customer(id)
)   ENGINE=INNODB;
```

2. 外键操作

```
--
-- 表结构 摘自  MySQL 官方文档
--

mysql> create table parent (
    ->     id int not null,
    ->     primary key (id)
    -> ) engine=innodb;
Query OK, 0 rows affected (0.14 sec)

mysql> create table child (
    ->     id int,
    ->     parent_id INT,
    ->     index par_ind (parent_id),
    ->     foreign key (parent_id)
    ->         references parent(id)
    ->         on delete cascade on update cascade  -- 比官网例子增加 update cascade
    -> ) engine=innodb;
Query ok, 0 rows affected (0.15 sec)

mysql> insert into child values(1,1);  -- 我们插入一条数据，id=1，parent_id=1
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`burn_test`.`child`, CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`id`) ON DELETE CASCADE)
-- 直接报错了，因为此时parent表中没有任何记录

mysql> insert into parent values(1);   -- 现在parent中插入记录
Query OK, 1 row affected (0.03 sec)

mysql> insert into child values(1,1);  -- 然后在child中插入记录，且parent_id是在parent中存在的
Query OK, 1 row affected (0.02 sec)

mysql> insert into child values(1,2);  -- 插入parent_id=2的记录，报错。因为此时parent_id=2的记录不存在
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`burn_test`.`child`, CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`id`) ON DELETE CASCADE)

mysql> select * from  child;
+------+-----------+
| id   | parent_id |
+------+-----------+
|    1 |         1 |  -- parent_id = 1
+------+-----------+
1 row in set (0.00 sec)

mysql> select * from  parent;
+----+
| id |
+----+
|  1 |  -- 根据表结构的定义（Foreign_key），这个值就是 child表中的id
+----+
1 row in set (0.00 sec)

mysql> update parent set id=100 where id=1;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from parent;
+-----+
| id  |
+-----+
| 100 |  -- 已经设置成了100
+-----+
1 row in set (0.00 sec)

mysql> select * from child;
+------+-----------+
| id   | parent_id |
+------+-----------+
|    1 |       100 |  -- 自动变化，这是on update cascade的作用，联级更新，parent更新，child也跟着更新
+------+-----------+
1 row in set (0.00 sec)

mysql> delete from parent where id=100;  -- 删除这条记录
Query OK, 1 row affected (0.03 sec)

mysql> select * from parent;  -- id=100的记录已经被删除了
Empty set (0.00 sec)

mysql> select * from child;  -- id=1，parent_id=100的记录跟着被删除了。on delete cascade的作用
Empty set (0.00 sec)

mysql> alter table child drop foreign key child_ibfk_1;  -- 删除 之前的外键
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table child add foreign key(parent_id)
    -> references parent(id) on update cascade on delete restrict;  -- 使用严格模式
Query OK, 0 rows affected (0.27 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> insert into parent values(50);
Query OK, 1 row affected (0.03 sec)

mysql> insert into child values(3,50);
Query OK, 1 row affected (0.03 sec)

mysql> insert into child values(3,51);  -- 和之前一样会提示错误
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`burn_test`.`child`, CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`id`) ON UPDATE CASCADE)

mysql> delete from parent where id=50;  -- 删除失败了，因为是restrict模式
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`burn_test`.`child`, CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`id`) ON UPDATE CASCADE)

-- 注意，delete 后面说明都不写表示 no action == restrict
```

> 外键约束，可以让数据进行一致性更新，但是会有一定的 性能损耗 ，线上业务使用不多。通常上述级联更新和删除都是由应用层业务逻辑进行判断并实现。