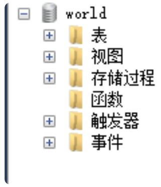
12.MySQL高级开发上（库层次下的操作对象-自定义函数，触发器，事件，视图）√



world
- 表
- 视图
- 存储过程
- 函数
- 触发器
- 事件

1.库层次下的自定义函数（user definition UDF）

1. 可以理解函数是一种特殊的存储过程，所以也可以支持循环，判断，游标功能

2.应用场景主要用于查询的需求

1.1 语法

```
CREATE FUNCTION `f1`(参数)
RETURNS int(11)              输出的值
DETERMINISTIC                固定关键字
BEGIN
函数体SQL;                  通常都是eelect
RETURN 变量;                  输出的变量
END
```

1.2 调用方法

```
select f1();
```

1.3 删除及查看

```
查看
select * from information_schema.ROUTINES\G
删除
drop FUNCTION f1;
```

1.4 函数案列

```bash
案列一：统计每个国家的总人口数
0.创建函数
DELIMITER $$
CREATE
    FUNCTION `world`.`f1`(cc VARCHAR(64))
    RETURNS INT
    DETERMINISTIC
    BEGIN
    DECLARE dd INT;
     SELECT SUM(population) FROM world.city WHERE CountryCode=cc INTO dd;
     RETURN dd;
    END$$

DELIMITER ;
1.调用函数
mysql> SELECT  world.f1('CHN');
+------------------+
| world.f1('CHN') |
+------------------+
|       175953614 |
+------------------+
```

2.库层次下的触发器（trigger）

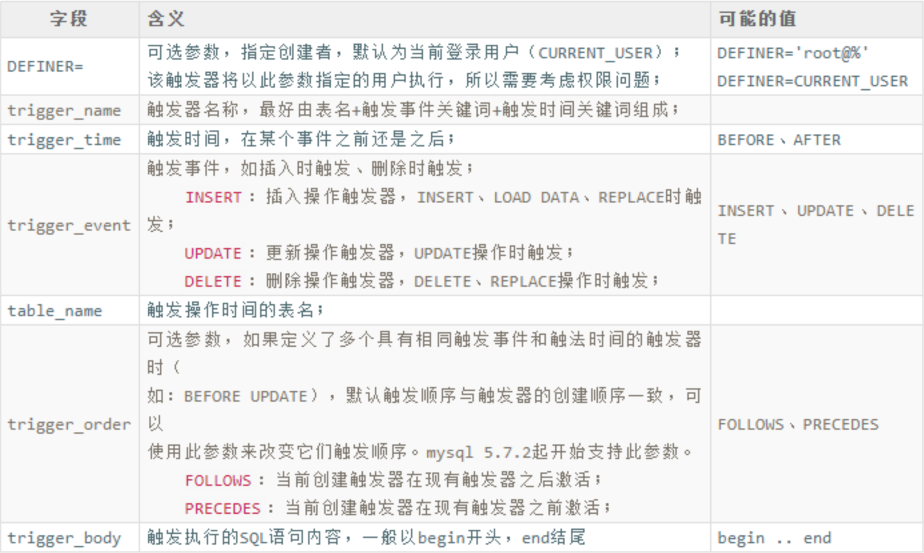
2.1 介绍

> 触发器是与表有关的数据库对象，在满足定义条件时触发，并执行触发器中定义的语句集合。
>
> 触发器的特性：
>
> 1、有begin end体，begin end;之间的语句可以写的简单或者复杂
>
> 2、什么条件会触发：I、D、U
>
> 3、什么时候触发：在增删改前或者后(before/after)
>
> 4、触发频率：针对每一行执行
>
> 5、触发器定义在表上，附着在表上。
>
> 也就是由事件来触发某个操作，事件包括INSERT语句，UPDATE语句和DELETE语句；可以协助应用在数据库端确保数据的完整性。

2.2 语法

```
CREATE [DEFINER = { 'user' | CURRENT_USER }]
TRIGGER trigger_name (触发器名称)
trigger_time: { BEFORE | AFTER }  trigger_event: { INSERT | UPDATE | DELETE }   触发事件在触发条件（增删改）之前还是之后
ON tbl_name
FOR EACH ROW                    附着在那张表，对于这张表的每一行
[trigger_odrer]
trigger_body                                 做的事情

```



字段	含义	可能的值

DEFINER=	可选参数，指定创建者，默认为当前登录用户（CURRENT_USER）；
	该触发器将以此参数指定的用户执行，所以需要考虑权限问题；
		DEFINER='root@%'
		DEFINER=CURRENT_USER

trigger_name	触发器名称，最好由表名+触发事件关键词+触发时间关键词组成；	
trigger_time	触发时间，在某个事件之前还是之后；	BEFORE、AFTER

trigger_event	触发事件，如插入时触发、删除时触发；
	INSERT：插入操作触发器，INSERT、LOAD DATA、REPLACE时触发；
	UPDATE：更新操作触发器，UPDATE操作时触发；
	DELETE：删除操作触发器，DELETE、REPLACE操作时触发；
		INSERT、UPDATE、DELETE

table_name	触发操作时间的表名；	
trigger_order	可选参数，如果定义了多个具有相同触发事件和触法时间的触发器时（如：BEFORE UPDATE），默认触发顺序与触发器的创建顺序一致，可以使用此参数来改变它们触发顺序。mysql 5.7.2起开始支持此参数。 FOLLOWS：当前创建触发器在现有触发器之后激活； PRECEDES：当前创建触发器在现有触发器之前激活；	FOLLOWS、PRECEDES
trigger_body	触发执行的SQL语句内容，一般以begin开头，end结尾	begin .. end

2.3 触发条件类型及new old变量的使用

类型	NEW和OLD使用
INSERT	NEW变量，获取Insert后的数据。
update	NEW变量，获取update后的数据；OLD变量，获取update前的数据。
delete	OLD变量，获取删除前数据。

类型	New和old变量的使用
insert	只有new变量。是指获取insert后的数据
update	既有new变量也有old变量。NEW变量，获取update后的数据；OLD变量，获取update前的数据。
delete	只有old变量。是指获取删除前的数据

2.4 触发器的应用

```
案列一
在生成环境中对于核心业务会有对核心业务的操作行为记录生成的日志表。
我们通过日志表进行对核心业务的监控和审计，日志表可能会造成主从延时的问题，解决方式就是跳过对日志表的复制。
0.首先模拟环境，创建需要的表
创建t4表
CREATE TABLE t4(
id INT,
NAME VARCHAR(20)
)ENGINE=INNODB CHARSET=utf8mb4;
向t4表插入数据
INSERT INTO t4 VALUES(1,'a'),(2,'b'),(3,'c');
创建日志表
CREATE TABLE t4_log( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
act_user VARCHAR(64),
act_type VARCHAR(50),
act_time VARCHAR(50),
act_id VARCHAR(20),
act_comment VARCHAR(100));
1.创建一个对t4表的insert操作的触发器。
DELIMITER $$
CREATE TRIGGER tr_insert_t4
AFTER INSERT ON t4 FOR EACH ROW
BEGIN
INSERT INTO t4_log(act_user,act_type,act_time,act_id,act_comment)
VALUES(USER(),'insert',NOW(),new.id,
CONCAT('insert into t4 values(',new.id,',',new.name,');'));
END$$
DELIMITER ;
2.我们查看我们创建的两张表的内容
mysql> select * from t4;
+------+------+
| id   | name |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | c    |
+------+------+

mysql> select * from t4_log;
Empty set (0.00 sec)

3.我们向t4表插入数据，就会触发我们的insert触发器，并记录行为到日志表t4_log中。
插入t4数据
mysql> insert into t4 values(4,'d');
查看t4表
mysql> select * from t4;
+------+------+
| id   | name |
+------+------+
|    1 | a    |
|     2 | b     |
|     3 | c     |
|     4 | d     |
+-------+-------+
查看日志表t4_log,
mysql> select * from t4_log;
+----+----------------+----------+---------------------+--------+------------------------------+
| id | act_user       | act_type | act_time            | act_id | act_comment                  |
+----+----------------+----------+---------------------+--------+------------------------------+
|  1 | root@localhost | insert   | 2021-04-13 21:49:19 | 4      | insert into t4 values(4,d); |
+----+----------------+----------+---------------------+--------+------------------------------+

4.创建一个对t4表的update操作的触发器
DELIMITER $$
CREATE TRIGGER tr_update_t4
AFTER UPDATE ON t4 FOR EACH ROW
BEGIN
INSERT INTO t4_log(act_user,act_type,act_time,act_id,act_comment)
VALUES(USER(),'update',NOW(),new.id,
CONCAT("update t4 set id=",new.id,",name=",new.name,"where id=",old.id));
END$$
DELIMITER ;
5.创建一个对t4表的delete操作的触发器
DELIMITER $$
CREATE TRIGGER tr_delete_t4
BEFORE DELETE ON t4 FOR EACH ROW
BEGIN
INSERT INTO t4_log(act_user,act_type,act_time,act_id,act_comment)
VALUES(USER(),'delete',NOW(),old.id,
CONCAT("delete from t4 where id=",old.id));
END$$
DELIMITER ;
```

3.库层次下的事件

3.1 介绍

> 将数据库按自定义的时间周期触发某种操作，类似linux下的crontab(任务计划)

3.2 查看事件调度器

Bash Copy

```
1.查看库下的所有事件
mysql> show events;
2.要查看当前是否已开启事件调度器
mysql> SHOW VARIABLES LIKE 'event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | ON    |
+-----------------+-------+
```

3.3 开启事件调度器

Bash Copy

```
开启事件调度器
1.通过命令行
SET GLOBAL event_scheduler = ON;
SET @@global.event_scheduler = ON;
SET GLOBAL event_scheduler = 1;
SET @@global.event_scheduler = 1;
2.通过配置文件
my.cnf event_scheduler = 1 #或者ON
```

3.4 事件调度器应用

> 0.模拟环境创建一张表
>
> create table ev1
>
> (
>
> ev_name varchar(20) not null,
>
> ev_started timestamp not null);
>
> # 案例1（立即启动事件）
>
> create event event_now
>
> on schedule
>
> at now()
>
> do insert into ev1 values('ev_test', now());
>
> # 案例2（每分钟启动事件）
>
> create event ev2
>
> on schedule
>
> every 1 minute
>
> do insert into ev1 values('ev_test1', now());

```
# 案例3（每秒钟启动事件）
CREATE event ev3
ON SCHEDULE
EVERY 1 SECOND
DO INSERT INTO ev3 VALUES(1);

# 案例4（每秒钟调用存储过程）
CREATE DEFINER=`root`@`localhost` EVENT `eventUpdateStatus`
ON SCHEDULE EVERY 1 SECOND
STARTS '2017-11-21 00:12:44'
ON COMPLETION PRESERVE
ENABLE
DO call updateStatus()

# 过程式创建events
DELIMITER $$
//事件的名称
CREATE EVENT `test`
//60秒循环一次
ON SCHEDULE EVERY 60 MINUTE_SECOND
// 开始时间,结束时间
STARTS '2017-11-01 00:00:00.000000' ENDS '2017-11-30 00:00:00.000000'
//过期后禁用事件而不删除
ON COMPLETION PRESERVE ENABLE
DO
BEGIN
//执行的内容
insert into ev1 values('event_now', now());
insert into ev1 values('event_now1', now());
ENDR $$
DELIMITE ;
```

4.库层次下的视图

4.1 视图的介绍

> 视图的作用是保存select语句的执行方法，不保存数据。
> 分为自定义视图和系统视图
> 自定义视图：当select语句经常使用且比较的复杂，那么就可以把select的执行方法保存下来作为视图，方便调用。
> 系统视图：　　数据库系统开发好的视图

4.2 自定义视图的应用

```
0.创建视图，相当于一张虚拟表，不存储数据，但是查询和表的方式一样。
create view v_city as
select a.name as aname ,b.name as bname ,a.population ,b.surfacearea from city as a join country as b where
a.population<100;
1.调用创建的视图
select * from v_test;
也可以对视图在做一些条件查询
2.查询创建视图的语句
mysql> show create view world.v_city;
+--------+---------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------+----------------------+------------
----------+
| View   | Create View

                                                                         | character_set_client | collation_c
onnection |
+--------+---------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------+----------------------+------------
----------+
| v_city | CREATE ALGORITHM=UNDEFINED DEFINER=`oldguo`@`10.0.0.%` SQL SECURITY DEFINER VIEW `v_city` AS selec
t `a`.`Name` AS `aname`,`b`.`Name` AS `bname`,`a`.`Population` AS `population`,`b`.`SurfaceArea` AS `surfacea
rea` from (`city` `a` join `country` `b`) where (`a`.`Population` < 100) | utf8                 | utf8_genera
l_ci      |
+--------+---------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------+----------------------+------------
----------+
1 row in set (0.00 sec)
```

4.3 查询视图

系统视图主要位于在information_schema和sys两个系统库下，还有我们创建的自定义视图。

Bash Copy

```
查询系统中所有视图对象
mysql> select table_schema,table_name,table_type from information_schema.tables where table_type like '%V
+--------------------+----------------------------------------------+-------------+
| TABLE_SCHEMA       | TABLE_NAME                                   | TABLE_TYPE  |
+--------------------+----------------------------------------------+-------------+
| information_schema | CHARACTER_SETS                               | SYSTEM VIEW |
| information_schema | CHECK_CONSTRAINTS                            | SYSTEM VIEW |
| information_schema | COLLATIONS                                   | SYSTEM VIEW |
| information_schema | COLLATION_CHARACTER_SET_APPLICABILITY        | SYSTEM VIEW |
| information_schema | COLUMNS                                      | SYSTEM VIEW |
| information_schema | COLUMN_STATISTICS                            | SYSTEM VIEW |
| information_schema | EVENTS                                       | SYSTEM VIEW |
| information_schema | FILES                                        | SYSTEM VIEW |
| information_schema | INNODB_DATAFILES                             | SYSTEM VIEW |
| information_schema | INNODB_FOREIGN                               | SYSTEM VIEW |
| information_schema | INNODB_FOREIGN_COLS                          | SYSTEM VIEW |
| information_schema | INNODB_FIELDS                                | SYSTEM VIEW |
| information_schema | INNODB_TABLESPACES_BRIEF                     | SYSTEM VIEW |
| information_schema | KEY_COLUMN_USAGE                             | SYSTEM VIEW |
| information_schema | KEYWORDS                                     | SYSTEM VIEW |
| information_schema | PARAMETERS                                   | SYSTEM VIEW |
| information_schema | PARTITIONS                                   | SYSTEM VIEW |
| information_schema | REFERENTIAL_CONSTRAINTS                      | SYSTEM VIEW |
| information_schema | RESOURCE_GROUPS                              | SYSTEM VIEW |
| information_schema | ROUTINES                                     | SYSTEM VIEW |
| information_schema | SCHEMATA                                     | SYSTEM VIEW |
| information_schema | ST_SPATIAL_REFERENCE_SYSTEMS                 | SYSTEM VIEW |
| information_schema | ST_UNITS_OF_MEASURE                          | SYSTEM VIEW |
| information_schema | ST_GEOMETRY_COLUMNS                          | SYSTEM VIEW |
| information_schema | STATISTICS                                   | SYSTEM VIEW |
| information_schema | TABLE_CONSTRAINTS                            | SYSTEM VIEW |
| information_schema | TABLES                                       | SYSTEM VIEW |
| information_schema | TRIGGERS                                     | SYSTEM VIEW |
| information_schema | VIEW_ROUTINE_USAGE                           | SYSTEM VIEW |
| information_schema | VIEW_TABLE_USAGE                             | SYSTEM VIEW |
| information_schema | VIEWS                                        | SYSTEM VIEW |
| information_schema | COLUMN_PRIVILEGES                            | SYSTEM VIEW |
| information_schema | ENGINES                                      | SYSTEM VIEW |
| information_schema | OPTIMIZER_TRACE                              | SYSTEM VIEW |
| information_schema | PLUGINS                                      | SYSTEM VIEW |
| information_schema | PROCESSLIST                                  | SYSTEM VIEW |
| information_schema | PROFILING                                    | SYSTEM VIEW |
| information_schema | SCHEMA_PRIVILEGES                            | SYSTEM VIEW |
| information_schema | TABLESPACES                                  | SYSTEM VIEW |
| information_schema | TABLE_PRIVILEGES                             | SYSTEM VIEW |
| information_schema | USER_PRIVILEGES                              | SYSTEM VIEW |
| sys                | version                                      | VIEW        |
| sys                | innodb_buffer_stats_by_schema                | VIEW        |
| sys                | x$innodb_buffer_stats_by_schema              | VIEW        |
| sys                | innodb_buffer_stats_by_table                 | VIEW        |
| sys                | x$innodb_buffer_stats_by_table                | VIEW        |
| sys                | schema_object_overview                        | VIEW        |
| sys                | schema_auto_increment_columns                 | VIEW        |
| sys                | x$schema_flattened_keys                       | VIEW        |
| sys                | schema_redundant_indexes                      | VIEW        |
| sys                | ps_check_lost_instrumentation                 | VIEW        |
| sys                | latest_file_io                                | VIEW        |
| sys                | x$latest_file_io                              | VIEW        |
| sys                | io_by_thread_by_latency                       | VIEW        |
| sys                | x$io_by_thread_by_latency                     | VIEW        |
| sys                | io_global_by_file_by_bytes                    | VIEW        |
| sys                | x$io_global_by_file_by_bytes                  | VIEW        |
| sys                | io_global_by_file_by_latency                  | VIEW        |
| sys                | x$io_global_by_file_by_latency                | VIEW        |
| sys                | io_global_by_wait_by_bytes                    | VIEW        |
| sys                | x$io_global_by_wait_by_bytes                  | VIEW        |
| sys                | io_global_by_wait_by_latency                  | VIEW        |
| sys                | x$io_global_by_wait_by_latency                | VIEW        |
| sys                | innodb_lock_waits                             | VIEW        |
| sys                | x$innodb_lock_waits                           | VIEW        |
| sys                | memory_by_user_by_current_bytes               | VIEW        |
| sys                | x$memory_by_user_by_current_bytes             | VIEW        |
| sys                | memory_by_host_by_current_bytes               | VIEW        |
| sys                | x$memory_by_host_by_current_bytes             | VIEW        |
| sys                | memory_by_thread_by_current_bytes             | VIEW        |
| sys                | x$memory_by_thread_by_current_bytes           | VIEW        |
| sys                | memory_global_by_current_bytes                | VIEW        |
| sys                | x$memory_global_by_current_bytes              | VIEW        |
| sys                | memory_global_total                           | VIEW        |
| sys                | x$memory_global_total                         | VIEW        |
| sys                | schema_index_statistics                       | VIEW        |
| sys                | x$schema_index_statistics                     | VIEW        |
| sys                | x$ps_schema_table_statistics_io               | VIEW        |
| sys                | schema_table_statistics                       | VIEW        |
| sys                | x$schema_table_statistics                     | VIEW        |
| sys                | schema_table_statistics_with_buffer           | VIEW        |
| sys                | x$schema_table_statistics_with_buffer         | VIEW        |
| sys                | schema_tables_with_full_table_scans           | VIEW        |
| sys                | x$schema_tables_with_full_table_scans         | VIEW        |
| sys                | schema_unused_indexes                         | VIEW        |
| sys                | schema_table_lock_waits                       | VIEW        |
| sys                | x$schema_table_lock_waits                     | VIEW        |
| sys                | statement_analysis                            | VIEW        |
| sys                | x$statement_analysis                          | VIEW        |
| sys                | statements_with_errors_or_warnings            | VIEW        |
| sys                | x$statements_with_errors_or_warnings          | VIEW        |
| sys                | statements_with_full_table_scans              | VIEW        |
| sys                | x$statements_with_full_table_scans            | VIEW        |
| sys                | x$ps_digest_avg_latency_distribution          | VIEW        |
| sys                | x$ps_digest_95th_percentile_by_avg_us         | VIEW        |
| sys                | statements_with_runtimes_in_95th_percentile   | VIEW        |
| sys                | x$statements_with_runtimes_in_95th_percentile | VIEW        |
| sys                | statements_with_sorting                       | VIEW        |
| sys                | x$statements_with_sorting                     | VIEW        |
| sys                | statements_with_temp_tables                   | VIEW        |
| sys                | x$statements_with_temp_tables                 | VIEW        |
| sys                | user_summary_by_file_io_type                  | VIEW        |
| sys                | x$user_summary_by_file_io_type                | VIEW        |
| sys                | user_summary_by_file_io                       | VIEW        |
| sys                | x$user_summary_by_file_io                     | VIEW        |
| sys                | user_summary_by_statement_type                | VIEW        |
| sys                | x$user_summary_by_statement_type              | VIEW        |
| sys                | user_summary_by_statement_latency             | VIEW        |
| sys                | x$user_summary_by_statement_latency           | VIEW        |
| sys                | user_summary_by_stages                        | VIEW        |
| sys                | x$user_summary_by_stages                      | VIEW        |
| sys                | user_summary                                  | VIEW        |
| sys                | x$user_summary                                | VIEW        |
| sys                | host_summary_by_file_io_type                  | VIEW        |
| sys                | x$host_summary_by_file_io_type                | VIEW        |
| sys                | host_summary_by_file_io                       | VIEW        |
| sys                | x$host_summary_by_file_io                     | VIEW        |
| sys                | host_summary_by_statement_type                | VIEW        |
| sys                | x$host_summary_by_statement_type              | VIEW        |
| sys                | host_summary_by_statement_latency             | VIEW        |
| sys                | x$host_summary_by_statement_latency           | VIEW        |
| sys                | host_summary_by_stages                        | VIEW        |
| sys                | x$host_summary_by_stages                      | VIEW        |
| sys                | host_summary                                  | VIEW        |
| sys                | x$host_summary                                | VIEW        |
| sys                | wait_classes_global_by_avg_latency            | VIEW        |
| sys                | x$wait_classes_global_by_avg_latency          | VIEW        |
| sys                | wait_classes_global_by_latency                | VIEW        |
| sys                | x$wait_classes_global_by_latency              | VIEW        |
| sys                | waits_by_user_by_latency                      | VIEW        |
| sys                | x$waits_by_user_by_latency                    | VIEW        |
| sys                | waits_by_host_by_latency                      | VIEW        |
| sys                | x$waits_by_host_by_latency                    | VIEW        |
| sys                | waits_global_by_latency                       | VIEW        |
| sys                | x$waits_global_by_latency                     | VIEW        |
| sys                | metrics                                       | VIEW        |
| sys                | processlist                                   | VIEW        |
| sys                | x$processlist                                 | VIEW        |
| sys                | session                                       | VIEW        |
| sys                | x$session                                     | VIEW        |
| sys                | session_ssl_status                            | VIEW        |
| information_schema | ENABLED_ROLES                                 | SYSTEM VIEW |
| information_schema | APPLICABLE_ROLES                              | SYSTEM VIEW |
| information_schema | ADMINISTRABLE_ROLE_AUTHORIZATIONS             | SYSTEM VIEW |
| information_schema | ROLE_COLUMN_GRANTS                            | SYSTEM VIEW |
| information_schema | ROLE_ROUTINE_GRANTS                           | SYSTEM VIEW |
| information_schema | ROLE_TABLE_GRANTS                             | SYSTEM VIEW |
| information_schema | INNODB_SESSION_TEMP_TABLESPACES               | SYSTEM VIEW |
| information_schema | INNODB_VIRTUAL                                | SYSTEM VIEW |
| information_schema | INNODB_BUFFER_POOL_STATS                      | SYSTEM VIEW |
| information_schema | INNODB_BUFFER_PAGE                            | SYSTEM VIEW |
| information_schema | INNODB_CMPMEM_RESET                             | SYSTEM VIEW |
| information_schema | INNODB_CMPMEM                                   | SYSTEM VIEW |
| information_schema | INNODB_TRX                                      | SYSTEM VIEW |
| information_schema | INNODB_CMP_PER_INDEX_RESET                      | SYSTEM VIEW |
| information_schema | INNODB_CMP_RESET                                | SYSTEM VIEW |
| information_schema | INNODB_FT_DEFAULT_STOPWORD                      | SYSTEM VIEW |
| information_schema | INNODB_CMP                                      | SYSTEM VIEW |
| information_schema | INNODB_TABLES                                   | SYSTEM VIEW |
| information_schema | INNODB_FT_BEING_DELETED                         | SYSTEM VIEW |
| information_schema | INNODB_METRICS                                  | SYSTEM VIEW |
| information_schema | INNODB_TEMP_TABLE_INFO                          | SYSTEM VIEW |
| information_schema | INNODB_FT_DELETED                               | SYSTEM VIEW |
| information_schema | INNODB_CACHED_INDEXES                           | SYSTEM VIEW |
| information_schema | INNODB_COLUMNS                                  | SYSTEM VIEW |
| information_schema | INNODB_FT_INDEX_TABLE                           | SYSTEM VIEW |
| information_schema | INNODB_TABLESTATS                               | SYSTEM VIEW |
| information_schema | INNODB_BUFFER_PAGE_LRU                          | SYSTEM VIEW |
| information_schema | INNODB_CMP_PER_INDEX                            | SYSTEM VIEW |
| information_schema | INNODB_FT_CONFIG                                | SYSTEM VIEW |
| information_schema | INNODB_FT_INDEX_CACHE                           | SYSTEM VIEW |
```

4.4 information_schema（i_s）系统库下视图的基本应用

4.4.1 作用

> 0.元数据是什么?
> 除了真实数据行，索引之外的数据信息。例如：数据字典信息，属性信息等。
> 1.i_s视图产生的原因?
> 1.1 元数据是mysql中比较核心的数据，不允许查询和修改。
> 1.2 元数据构造及其复杂。
> 2.i_s视图的作用：
> mysql 提供了用来查询系统“元数据”的视图。封装了元数据查询的方法。
> 我们可以通过I_S和show更加方便的查询元数据信息。

4.4.2 常用视图的应用

> **tables：提供数据库中所有表相关元数据**
> **TRIGGERS： 提供数据库中所有触发器相关元数据**
> **VIEWS：提供数据库中所有视图相关元数据**
> **ROUTINES： 提供数据库中所有存储过程相关元数据**
> **COLUMNS： 提供数据库中所有表中列相关元数据**
> **EVENTS： 提供数据库中所有事件相关元数据**
> **processlist： 提供数据库连接方面的系统状态。**

4.4.3 i_s下的tables应用

4.4.3.1 结构介绍

```bash
mysql> desc information_schema.tables;数据库中所有表相关元数据
TABLE_CATALOG
TABLE_SCHEMA     表所在的库
TABLE_NAME       表名
TABLE_TYPE       表的类型（base table 基础表）
ENGINE           存储引擎类型
VERSION          版本
ROW_FORMAT       行格式
TABLE_ROWS       数据行（从统计信息中获取，粗略的不准确）
AVG_ROW_LENGTH   每行的平均行长度（字节）
DATA_LENGTH      数据长度
MAX_DATA_LENGTH  最大数据长度
INDEX_LENGTH     索引长度
DATA_FREE        碎片量
AUTO_INCREMENT   自增长
CREATE_TIME      创建时间
UPDATE_TIME      更新时间
CHECK_TIME       检查时间
TABLE_COLLATION
CHECKSUM
CREATE_OPTIONS   创建选项
TABLE_COMMENT    表的注释
```

4.4.3.2 应用实列

Bash Copy

```
实列一：统计当前数据库实列业务相关的库和表的信息（  排序系统库 mysql,sys,information_schema,performance_schema)
显示格式：库名 表名列表 表个数
mysql> select table_schema,group_concat(table_name),count(*) from information_schema.tables
where table_schema not in ('sys','mysql','information_schema','performance_schema') group by table_schema;
+--------------+----------------------------------------------------------------------------------------+-----
| TABLE_SCHEMA | group_concat(table_name)                                                               | cour
+--------------+----------------------------------------------------------------------------------------+-----
| oldguo       | b,stu,stu1,stu2,stu3,stu5,stu6,a                                                       |
| world        | v_city,tt,teacher,t4_log,t4,t2,t1,student,sc,ev1,course,countrylanguage,country,city,a |
+--------------+----------------------------------------------------------------------------------------+-----

实列二： 统计当前实例每个数据库的数据总量（排除掉mysql sys information_schema performance_schema)
mysql> select table_schema,sum(table_rows * avg_row_length + index_length)/1024/1024 as total_mb
from information_schema.tables where table_schema not in ('sys','mysql','information_schema','performance_sche
+--------------+------------+
| TABLE_SCHEMA | total_mb   |
+--------------+------------+
| oldguo       | 0.07812119 |
| world        | 0.87085152 |
+--------------+------------+

实列三：统计当前数据库实例非innodb的表（排除掉mysql sys information_schema performance_schema)
mysql> select table_schema,table_name ,engine from information_schema.tables
where table_schema not in ('sys','mysql','information_schema','performance_schema') and engine <> 'INNODB';
修改表存储引擎为innodb
alter table 库名.表名 engine=innodb;

实列四：查询有碎片的表信息
mysql> select table_schema,table_name ,data_free from information_schema.tables
where table_schema not in ('sys','mysql','information_schema','performance_schema') and data_free >0;

实列五：拼接sql语句
需求一：查询当前系统中所有非INNODB的表。
mysql> select table_schema,table_name ,engine from information_schema.tables
where table_schema not in ('sys','mysql','information_schema','performance_schema') and engine <> 'INNODB';
需求二：将这些非INNODB的表替换为INNODB
mysql> select concat("alter table ",table_schema,".",table_name," engine=innodb;") from information_schema.tab
where table_schema not in ('sys','mysql','information_schema','performance_schema') and engine <> 'INNODB' int
source /tmp/alter.sql
```

4.4.4 i_s下的 TRIGGERS（触发器），VIEWS(视图)，ROUTINES（存储过程）、EVENTS（事件）的应用

需求一：迁移备份前需要确认是否有特殊对象

4.4.4.1 TRIGGERS（触发器）

TRIGGERS的列信息

> TRIGGER_CATALOG: 触发器目录
> TRIGGER_SCHEMA: world 触发器在那个库下
> TRIGGER_NAME: tr_update_t4 触发器的名字
> EVENT_MANIPULATION: 触发器操作
>
> EVENT_OBJECT_CATALOG: 触发器项目目录
> EVENT_OBJECT_SCHEMA: 触发器项在那个目库
> EVENT_OBJECT_TABLE: 触发器对于那个表作用
> ACTION_ORDER: 1
> ACTION_CONDITION:
> ACTION_STATEMENT: 创建触发器的语句
> ACTION_ORIENTATION:
> ACTION_TIMING: AFTER 触发器事件
> ACTION_REFERENCE_OLD_TABLE:
> ACTION_REFERENCE_NEW_TABLE:
> ACTION_REFERENCE_OLD_ROW:
> ACTION_REFERENCE_NEW_ROW:
> CREATED: 创建时间
> SQL_MODE:
> DEFINER: oldguo@10.0.0.% 创建用户

> CHARACTER_SET_CLIENT: 客户端字符集

> COLLATION_CONNECTION:

> DATABASE_COLLATION

查询业务库下的触发器

```bash
mysql> select TRIGGER_SCHEMA,TRIGGER_NAME,EVENT_OBJECT_SCHEMA from information_schema.triggers
where TRIGGER_SCHEMA not in ('sys','mysql','information_schema','performance_schema');
+----------------+--------------+---------------------+
| TRIGGER_SCHEMA | TRIGGER_NAME | EVENT_OBJECT_SCHEMA |
+----------------+--------------+---------------------+
| world          | tr_insert_t4 | world               |
| world          | tr_update_t4 | world               |
| world          | tr_delete_t4 | world               |
+----------------+--------------+---------------------+
```

4.4.4.2 VIEWS(视图)

VIEWS(视图)列信息

> TABLE_CATALOG 视图目录

> TABLE_SCHEMA 视图在那个库下

TABLE_NAME	视图名字
VIEW_DEFINITION	
CHECK_OPTION	
IS_UPDATABLE	
DEFINER	视图创建者
SECURITY_TYPE	
CHARACTER_SET_CLIENT	
COLLATION_CONNECTION	

查询业务库下的视图

```bash
mysql> select TABLE_SCHEMA,TABLE_NAME from  information_schema.views where TABLE_SCHEMA not in ('sys','mysql',
+--------------+------------+
| TABLE_SCHEMA | TABLE_NAME |
+--------------+------------+
| world        | v_city     |
+--------------+------------+
```

4.4.4.3 ROUTINES（存储过程）

ROUTINES（存储过程）列信息

SPECIFIC_NAME: p_case2	存储过程具体信息
ROUTINE_CATALOG: def	存储过程目录
ROUTINE_SCHEMA: world	存储过程在那个库下
ROUTINE_NAME: p_case2	存储过程名字
ROUTINE_TYPE: PROCEDURE	
DATA_TYPE:	
CHARACTER_MAXIMUM_LENGTH: NULL	
CHARACTER_OCTET_LENGTH: NULL	
NUMERIC_PRECISION: NULL	
NUMERIC_SCALE: NULL	
DATETIME_PRECISION: NULL	
CHARACTER_SET_NAME: NULL	

```
    COLLATION_NAME: NULL
    DTD_IDENTIFIER: NULL
     ROUTINE_BODY: SQL
 ROUTINE_DEFINITION: BEGIN
DECLARE result VARCHAR(20);
DECLARE COUNT INT DEFAULT 0;
SELECT COUNT(*)  FROM world.t1 WHERE t1.username=u AND t1.pass=p INTO COUNT;
CASE WHEN COUNT>0 THEN SET result='success!';
ELSE
SET result='error!';
END CASE;
SELECT result;
END
     EXTERNAL_NAME: NULL
  EXTERNAL_LANGUAGE: SQL
   PARAMETER_STYLE: SQL
   IS_DETERMINISTIC: NO
   SQL_DATA_ACCESS: CONTAINS SQL
        SQL_PATH: NULL
     SECURITY_TYPE: DEFINER
        CREATED: 2021–04–12 16:24:34
     LAST_ALTERED: 2021–04–12 16:24:34
        SQL_MODE:
   ROUTINE_COMMENT:
        DEFINER: oldguo@10.0.0.%
CHARACTER_SET_CLIENT: utf8
COLLATION_CONNECTION: utf8_general_ci
 DATABASE_COLLATION: utf8mb4_0900_ai_ci
```

查询业务库下的存储过程

```
mysql> select ROUTINE_SCHEMA,ROUTINE_NAME from  information_schema.ROUTINES where  ROUTINE_SCHEMA not in ('sys
+----------------+--------------+
| ROUTINE_SCHEMA | ROUTINE_NAME |
+----------------+--------------+
| world          | p_null       |
| world          | p_in         |
| world          | p_in1        |
| world          | p_in2        |
| world          | p_out        |
| world          | p_in_out     |
| world          | p_inout      |
| world          | p_inout1     |
| world          | p_1          |
| world          | p_if         |
| world          | p_case       |
| world          | p_case1      |
| world          | p_case2      |
| world          | p_2          |
| world          | p_while      |
| world          | p_repeat     |
| world          | p_loop       |
| world          | p_loop1      |
| world          | p_c          |
| world          | p_c2         |
| world          | f1           |
+----------------+--------------+
```

4.4.4.4 EVENTS（事件）

EVENTS（事件）列信息

EVENT_CATALOG	事件目录
EVENT_SCHEMA	事件在那个库下
EVENT_NAME	事件名字
TIME_ZONE	
EVENT_BODY	
EVENT_DEFINITION	
EVENT_TYPE	
EXECUTE_AT	
INTERVAL_VALUE	
INTERVAL_FIELD	

> SQL_MODE

> STARTS

> ENDS

> STATUS

> ON_COMPLETION

> CREATED

> LAST_ALTERED

> LAST_EXECUTED

> EVENT_COMMENT

> ORIGINATOR

> CHARACTER_SET_CLIENT | varchar(64)

> COLLATION_CONNECTION | varchar(64)

> DATABASE_COLLATION

查询业务库下的事件

```
mysql> select EVENT_SCHEMA,EVENT_NAME from  information_schema.EVENTS where  EVENT_SCHEMA not in ('sys','mysql
```

4.4.5 i_s下的 COLUMNS应用

提供数据库中所有表中列相关元数据（数据字典信息）

4.4.5.1 COLUMNS列信息

> TABLE_CATALOG

> TABLE_SCHEMA 库名

> TABLE_NAME 表名

> COLUMN_NAME 列名

> ORDINAL_POSITION

> COLUMN_DEFAULT

> IS_NULLABLE 是否非空

> DATA_TYPE 数据类型

CHARACTER_MAXIMUM_LENGTH	
CHARACTER_OCTET_LENGTH	
NUMERIC_PRECISION	
NUMERIC_SCALE	
DATETIME_PRECISION	
CHARACTER_SET_NAME	
COLLATION_NAME	
COLUMN_TYPE	列类型
COLUMN_KEY	列上是否有索引，索引类型
EXTRA	
PRIVILEGES	权限
COLUMN_COMMENT	列注释
GENERATION_EXPRESSION	
SRS_ID	

4.4.5.2 COLUMNS应用

```bash
以每个库下的每个表为单位，更加好看的显示列信息
mysql> select * from  world.tt;
+------+------+
| id   | num  |
+------+------+
|    1 |  110 |
|    2 |  119 |
|    3 |  120 |
+------+------+

mysql> select concat(TABLE_SCHEMA,'.',TABLE_NAME),group_concat(concat("c_name: ",COLUMN_NAME," ","null: ",IS_N
from  information_schema.COLUMNS where TABLE_SCHEMA   not in ('sys','mysql','information_schema','performance_
group by TABLE_SCHEMA,TABLE_NAME;
|world.tt                            | c_name: id null: YES date_typeint key: ,c_name: num null: YES date_type

```

4.4.6 i_s下的 processlist应用

```bash
mysql> show processlist;
+----+-----------------+-----------+------+---------+------+------------------------+------------------+
| Id | User            | Host      | db   | Command | Time | State                  | Info             |
+----+-----------------+-----------+------+---------+------+------------------------+------------------+
|  5 | event_scheduler | localhost | NULL | Daemon  | 1264 | Waiting on empty queue | NULL             |
|  8 | root            | localhost | NULL | Query   |    0 | starting               | show processlist |
+----+-----------------+-----------+------+---------+------+------------------------+------------------+
```

4.4.6.1 processlist视图的列信息

ID	id号
USER	用户
HOST	主机
DB	数据库
COMMAND	状态
TIME	连接事件
STATE	状态
INFO	

4.4.6.2 processlist视图应用

```bash
需求： 维护性操作需要停业务。需要将所有外部连接进行释放。(或者使用pt-kill)
select concat("kill ",id,";") from PROCESSLIST where host not in ('localhost','127.0.0.1','db01')
```

%E8%87%AA%E5%AE%9A%E4%B9%89%E5%87%BD%E6%95%B0%EF%BC%8C%E8%A7%A6%E5%8F%91%E5%99%A8%EF%BC%8C%E4%BA%8B%E4%BB%B6%EF%BC%8C%E8%A7%86%E5%9B%BE%E