MySQL5.7新特性

# MySQL服务功能增强

## 计算列

|  |
| --- |
| 语法如下：在CREATE TABLE或者ALTER TABLE语句中支持增加计算列的方式  Col\_name data\_type [GENERATED ALWAYS] AS (表达式) [VIRTUAL | STORED] [UNIQUE [KEY]] [COMMENT comment] [[NOT] NULL] [ [PRIMARY] KEY]  说明 :  VIRTUAL不保存在磁盘，STORED持久化到磁盘  举例：c3 int generated always as (c1+c2) virtual, |

## json数据类型和函数

|  |
| --- |
| 1.新建数据表  *# 新建test1表*  **CREATE** **TABLE** test1 (  `id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT** **COMMENT** '主键',  `info` JSON **NULL**, **PRIMARY** **KEY** (`id`)  ) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=**utf8** **COMMENT**='测试5.7版本json格式';  2.插入JSON数据  **INSERT** **INTO** test1 (info) **VALUES**('{"name":"yuhaiwei","age":18,"hobby":"swim,reading,writting"}');  **INSERT** **INTO** test1 (info) **VALUES**('{"name":"tangwei","age":28,"hobby":" SHOW,song,movie"}');  3.json 函数详解  *#json\_array 数组json串*  **INSERT** **INTO** test1 (info) **VALUES** (json\_array('"testname', 99, 'basketball'));  *#json\_object 对象json串*  **INSERT** **INTO** test1 (info) **VALUES** (json\_object('name','"json\_object\_test\_name"', 'age',99, 'hobby','basketball'));  *#JSON\_MERGE(json\_doc, json\_doc[, json\_doc] ...) 合并json*  **SELECT** json\_merge('[{"name":"yhw"},{"name":"test"}]', '{"name":"test1"}');  *#json\_type 判断json类型，是数组还是对象*  **SELECT** json\_type(info) **FROM** test1;  *#JSON\_VALID(val) 判断是否合法json*  **SELECT** info,json\_valid(info) **FROM** test1;  *#json\_quote 用双引号括起来，并对json中的一些特殊字符转义*  **INSERT** **INTO** test1 (info) **VALUES**((('{"name":"noquote\_name","age":18,"hobby":"swim,reading,writting"}')));  *#JSON\_UNQUOTE(val) 和 json\_quote相反*  *#JSON\_EXTRACT(json\_doc, path[, path] ...) 查询出json的属性值 $-整个json，$[0]-json数组第一个元素，$.name-json的name属性值*  **SELECT** info,json\_extract(info, '$.name') **FROM** test1;  *# ->符号 ，作用等同于 json\_extract*  **select** info->'$.name' **from** test\_57.test1 **limit** 1;  *# JSON\_SET(json\_doc, path, val[, path, val] ...) 替换json串中的值, 有新属性值会添加*  **select** json\_set(info, '$.name', 'yuhaiweiset', '$.age', 11, '$.key1', 'value1') **from** test\_57.test1 **limit** 1;  *# json\_insert 添加值（不替换）,用法同json\_set*  *# json\_replace 替换值 （不添加）,用法同json\_set*  *# json\_remove 删除*  **select** json\_remove(info, '$.name', '$.age', '$.name') **from** test\_57.test1 **limit** 1;  *#JSON\_CONTAINS(json\_doc, val[, path]) json\_doc的path值 是否包含val*  **SELECT** JSON\_CONTAINS('{"a": 1, "b": 2, "c": {"d": 4}}', '{"d": 4}', '$.c'); *#结果 1*  *#JSON\_CONTAINS\_PATH(json\_doc, one\_or\_all, path[, path] ...) json\_doc是否包含路径path,第二个参数为one 只要包含一个就返回true*  **SET** @j = '{"a": 1, "b": 2, "c": {"d": 4}}';  **SELECT** JSON\_CONTAINS\_PATH(@j, 'one', '$.a', '$.e'); *#结果 1*  **SELECt** JSON\_CONTAINS\_PATH(@j, '  ALL', '$.a', '$.e'); *#结果 0*  *#json\_keys(json,[,path]) json的path路径的所有属性名*  **SELECT** JSON\_KEYS('{"a": 1, "b": {"c": 30}}', '$.b'); *#结果["c"]*  *#JSON\_SEARCH(json\_doc, one\_or\_all, search\_str[, escape\_char[, path] ...]) 返回search\_str在json\_doc的path路径下的全路径*  **SET** @j = '["abc", [{"k": "10"}, "def"], {"x":"abc"}, {"y":"bcd"}]';  **SELECT** JSON\_SEARCH(@j, '  ALL', '%b%', **NULL**, '$[2]'); *#结果 "$[2].x"*  *#JSON\_ARRAY\_APPEND(json\_doc, path, val[, path, val] ...) 在最后添加子节点*  **SELECT** JSON\_ARRAY\_APPEND('{"a": 1, "b": [2, 3], "c": 4}', '$.b[0]', '  Y'); *# 结果{"a": 1, "b": [[2, "y"], 3], "c": 4}*  *#JSON\_ARRAY\_INSERT(json\_doc, path, val[, path, val] ...) 在前面添加兄弟节点*  **SELECT** JSON\_ARRAY\_INSERT('{"a": 1, "b": [2, 3], "c": 4}', '$.b[0]', '  Y'); *# 结果{"a": 1, "b": ["y", 2, 3], "c": 4}*  *#JSON\_DEPTH(json\_doc) json深度*  **SELECT** JSON\_DEPTH('{}'), JSON\_DEPTH('[]'), JSON\_DEPTH('  TRUE'), JSON\_DEPTH('[10, 20]'), JSON\_DEPTH('[[], {}]'); *#结果：1 1 1 2 2*  *#JSON\_LENGTH(json\_doc[, path]) json的长度，不包括嵌套的子元素的长度*  **SELECT** JSON\_LENGTH('[1, 2, {"a": 3}]'); *#结果：3*  **SELECT** JSON\_LENGTH('{"a": 1, "b": {"c": 30}}', '$.b') *#结果 1* |

## 并行逻辑备份工具mysqlpump

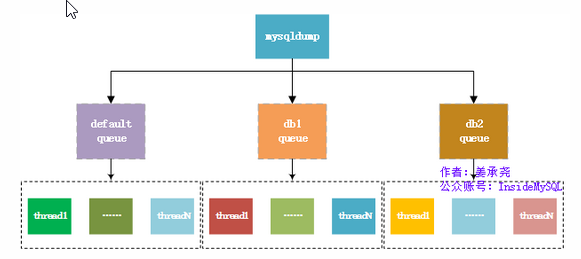
MySQL 5.7.11版本发布，其中 **最有意义** 的部分在于官方修复了之前mysqlpump工具一致性备份的问题，使得mysqlpump工具在生产环境中有了用武之地。而在之前MySQL 5.7的文档中明确写着，mysqlpump是不支持一致性的备份：

Before MySQL 5.7.11, use of the –single-transaction option is **mutually exclusive**with parallelism. To use --single-transaction, disable parallelism by setting --default-parallelism to 0 and not using any instances of --parallel-schemas

mysqlpump相对于之前的逻辑备份工具mysqldump来说，优势如下：

* 支持基于表的并行导出功能(参数--default-parallelism，默认为2，参数--parallel-schemas，控制并行导出的库)
* 导出的时候带有进度条（参数--watch-progress，默认开启）
* 支持直接压缩导出导入（参数--compress-output，支持ZLIB和LZ4）

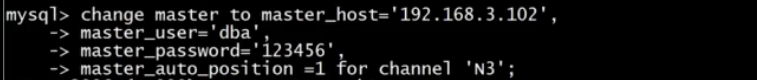
mysqlpump的并行导出功能的架构为：队列+线程，允许有多个队列，每个队列下有多个线程，而一个队列可以绑定1个或者多个数据库。但是，对于每张表的导出只能是单个线程的， 这和mydumper工具是不一样的 ，因为mydumper支持一张表多个线程以chunk的方式批量导出，这在主键是随机的情况下，导出速度还能有提升。mysqlpump的架构如下图所示：



|  |
| --- |
| #备份举例：  mysqlpump --single-transaction --default-parallelism=4 --compress-output=ZLIB --all-databases > all.sql |

# Replication相关增强

## 多源复制





## 多线程复制

|  |
| --- |
| #多线程配置  slave\_parallel\_type = LOGICAL\_CLOCK #默认为database,基于库。  slave\_parallel\_workers = 4 #线程个数 |

## 在线变更复制方式

|  |
| --- |
| #1在主从上设置ENFORCE\_TGID\_CONSISTENCY，保证所有服务器没有警告发生  SET @@BLOBAL. ENFORCE\_TGID\_CONSISTENCY=WARN;  #2在主从上设置ENFORCE\_TGID\_CONSISTENCY=ON  SET @@BLOBAL. ENFORCE\_TGID\_CONSISTENCY=ON;  #主从上设置GTID\_MODE值（OFF,OFF\_PERMISSIVE,ON\_PERMISSIVE,ON这四种状态，需要按顺序设置）  SET @@GLOBAL.GTID\_MODE=OFF\_PERMISSIVE;  SET @@GLOBAL.GTID\_MODE=ON\_PERMISSIVE;  #3在从上查看状态变量值，空值或者0值表示成功，保证基于日志的事务已经在从上执行完成。  Show status like 'ongoing\_anonymous\_transaction\_count ';  #在主从上设置gtid-mode=ON;并加入到my.cnf  SET @@GLOBAL.GTID\_MODE=ON;  #4重启从服务器  STOP SLAVE [FOR CHANNEL 'channel '];  CHANGE MASTER TO MASTER\_AUTO\_POSITION=1;  START SLAVE [FOR CHANNEL 'channel ']; |

# InnoDB引擎增强

## 缓冲池大小在线变更

|  |
| --- |
| 在MySQL5.7之前：要变更innodb\_buffer\_pool大小必须更改my.cnf文件后重启数据库服务器：MySQL5.7之后：innodb\_buffer\_pool\_size参数变为动态参数，可以在线调整innodb缓冲池的大小。  SET GLOBAL innodb\_buffer\_pool\_size=10G; |

## Innodb\_buffer\_pool导入导出功能

|  |
| --- |
| innodb\_buffer\_pool\_chunk\_size：134217728 # innodb缓冲最小单位  innodb\_buffer\_pool\_dump\_at\_shutdown：ON # 关机是转储缓冲池到磁盘  innodb\_buffer\_pool\_dump\_now；OFF #立即导出缓冲池内容到磁盘  innodb\_buffer\_pool\_dump\_pct：25 #转储总缓冲池的百分比  innodb\_buffer\_pool\_filename：ib\_buffer\_pool #缓冲池加载或者转储的文件名  innodb\_buffer\_pool\_instances：4 #缓冲池分成几块  innodb\_buffer\_pool\_load\_abort：OFF #中断当前正从磁盘加载的所有缓冲池  innodb\_buffer\_pool\_load\_at\_startup：ON #启动时候是否从磁盘加载缓冲池  innodb\_buffer\_pool\_load\_now：OFF #立即从磁盘加载到缓冲池  innodb\_buffer\_pool\_size：6442450944 #缓冲池总大小 |

## 支持innodb表建立表空间

|  |
| --- |
| MySQL5.7之前：具有系统表空间和可以为每个表建立一个独立的表空间。但是两者不能同时出现。  MySQL5.7之后：支持CREATE TABLESPACE语法为一个表或多个表建立公用的表空间。可以节约内存，但是不能够回收空间。  语法：  CREATE TABLESPACE tablespace\_name  ADD DATAFILE ‘file\_name.ibd’  [FILE\_BLOCK\_SIZE = n]  [ENGINE=xxx]  举例：  CREATE TABLESPACE ts1  ADD DATA 'ts1.ibd '  ENGINE=InnoDB  使用表空间：create table t2(…) tablespace `ts1` |

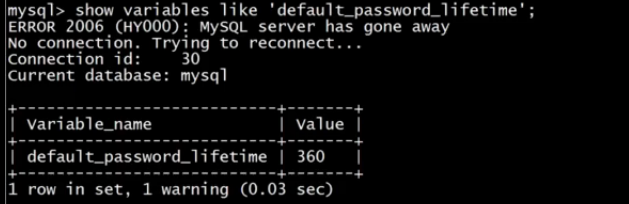
# 安全及管理方面增强

## 不支持old\_password认证

用户表mysql.user的plugin字段不允许为空，默认值mysql\_native\_password，而不是mysql\_old\_password，不再支持旧密码格式； 增加密码过期机制，过期后需要修改密码，否则可能会被禁用，或者进入沙箱模式；

使用mysql\_install\_db初始化时，默认会自动生成随机密码，并且不创建除root@localhost 外的其他账号，也不创建test库； 在5.7中，推荐使用mysqld --initialize对数据库进行初始化，在初始化时如果加上--initial-insecure，则会创建空密码的 root@localhost 账号，否则会创建带密码的 root@localhost 账号，密码直接写在 log-error 日志文件中

## 增加账号默认过期时间



## 增加了sys管理库