一次艰辛的字符集转换历程

——2017.3.25 ACMUG 深圳

六度人和 周晓

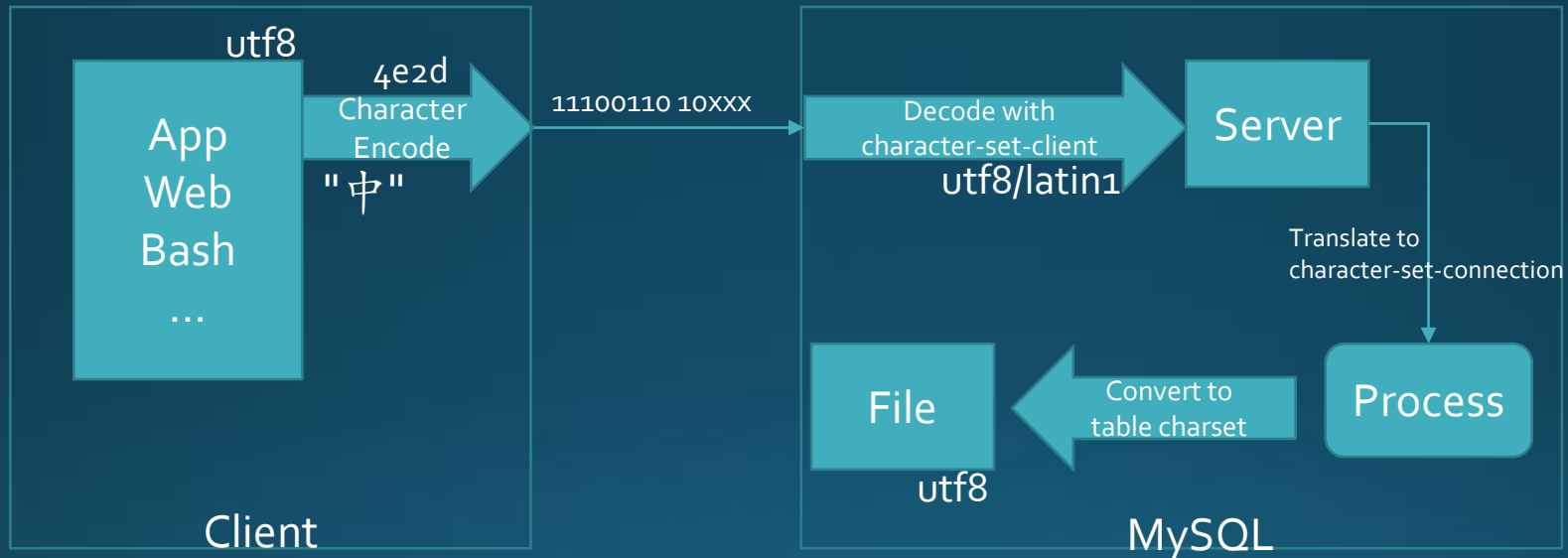
seanlook7@gmail.com

Agenda

- 历史的包袱和约束
- 我们的方案
- 踩过的坑
- 如何一致性校验
- 后面的故事

1. 历史的包袱

- 表utf8，连接latin1



Client	Unicode	中 (4e2d)
		0110 1100 0100 1001
	utf8	11100110 10110001 10001001
Decode	latin1	11100110 10110001 10001001
File	utf8	11000011 10100100 11000010 10111000 11000010 10101101
		c3 a4 c2 b8 c2 ad

https://zh.wikipedia.org/wiki/Unicode字符平面映射

1. 历史的包袱

- 表utf8，连接latin1
  - 存储空间被放大。1汉字需要>=6字节
    - varchar单行最大65535bytes
  - 无法利用binlog回滚
  - 第三方软件抽取数据，大多默认utf8
  - 字符集不统一，也是规范问题
- 可以存emoji!

```
mysql> insert into t_charset(name) values('中'),('文'),('1🐟1');
Query OK, 3 rows affected, 2 warnings (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 2

mysql> show warnings;
+---------+------+-----------------------------------------------------------------------------+
| Level   | Code | Message                                                                     |
+---------+------+-----------------------------------------------------------------------------+
| Warning | 1300 | Invalid utf8 character string: 'F09F98'                                     |
| Warning | 1366 | Incorrect string value: '\xF0\x9F\x98\x821' for column 'name' at row 3      |
+---------+------+-----------------------------------------------------------------------------+
2 rows in set (0.01 sec)

mysql> set names latin1;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t_charset(name) values('中'),('文'),('1🐟1');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select c.*,length(c.name), char_length(c.name) from t_charset c;
+----+--------+----------------+---------------------+
| id | name   | length(c.name) | char_length(c.name) |
+----+--------+----------------+---------------------+
|  1 | ?      |              3 |                   1 |
|  2 | ?      |              3 |                   1 |
|  3 | 1      |              1 |                   1 |
|  4 | 中     |              6 |                   3 |
|  5 | 文     |              8 |                   3 |
|  6 | 1🐟1   |             11 |                   6 |
+----+--------+----------------+---------------------+
6 rows in set (0.00 sec)
```

utf8

目标：

utf8 - utf8 - utf8

client - connection - server

2.有哪些约束

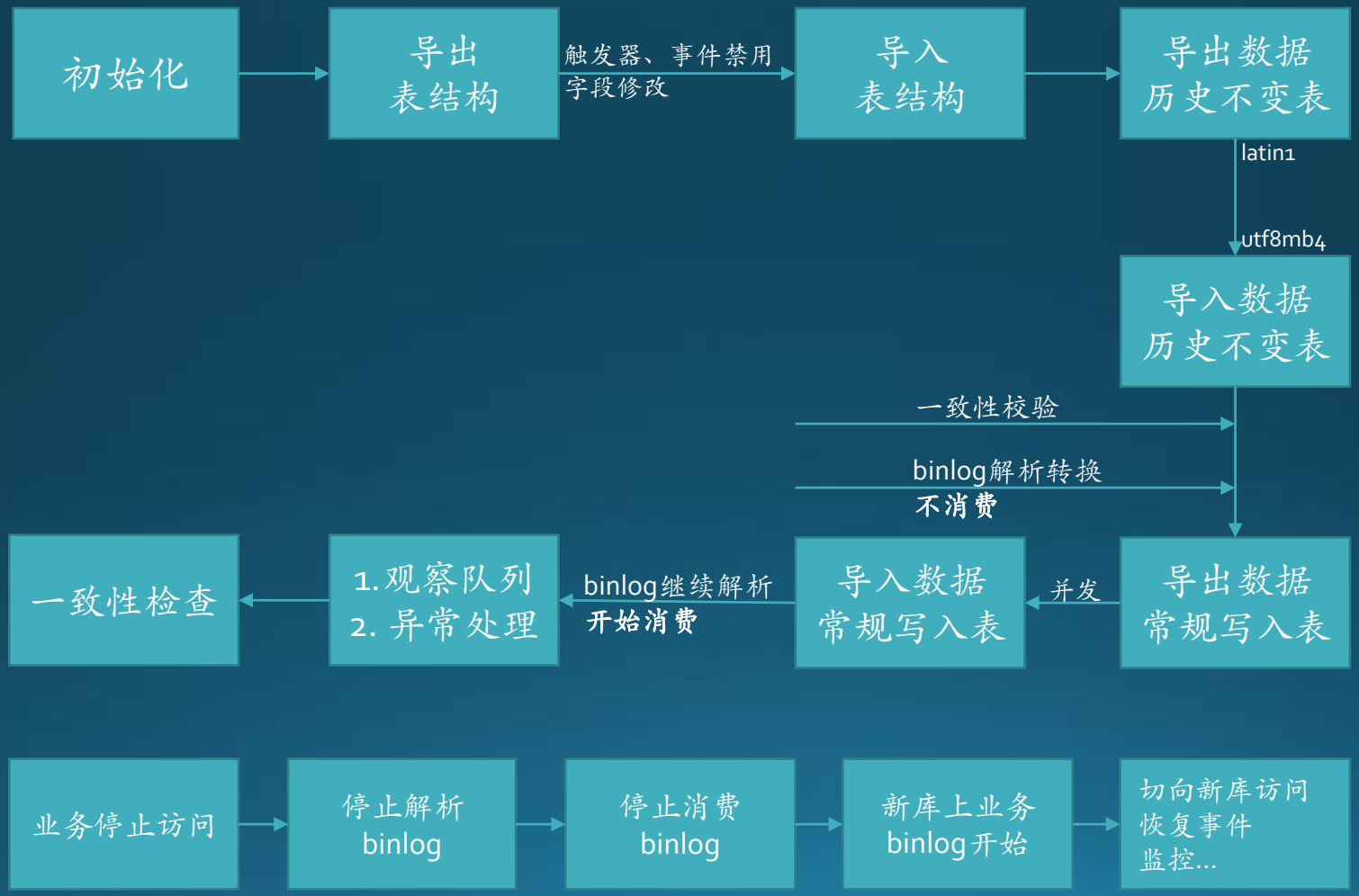
- 有java/php/c++客户端
- 无法按实例逐个进行
- 尽可能减少停服时间
- 必须保证数据一致性
- 部分表有emoji表情
- RDS带来的约束
  - 从库无binlog
  - select .. into outfile

3. 我们的方案-探索

- `ALTER TABLE tbl_name CONVERT TO CHARACTER SET utf8/latin1;`
- 数据传输服务(DTS)
- 数据库访问中间层

- 逻辑导出导入
  - dump用latin1，load用utf8
  - mysqldump全量 + canal同步增量

方案推演


初始化
导出
表结构
触发器、事件禁用
字段修改
导入
表结构
导出数据
历史不变表
latin1
utf8mb4
导入数据
历史不变表
一致性校验
binlog解析转换
不消费
一致性检查
1.观察队列
2. 异常处理
binlog继续解析
开始消费
导入数据
常规写入表
并发
导出数据
常规写入表
业务停止访问
停止解析
binlog
停止消费
binlog
新库上业务
binlog开始
切向新库访问
恢复事件
监控...

如何让导入导出更快

- 切割dump文件效率达不到(tbdba-restore-mysqldump.pl)
- 并发导出导入 mypumpkin
  - https://github.com/seanlook/mypumpkin

- 功能
  - 不改变mysqldump使用习惯，多进程加快load速度
  - 兼容 -B/--tables/--ignore-table 选项
  - 表级别并发，生成 dbname.tblname.sql 文件
  - 不能保证不同表之间数据的一致性

- 速度对比
  - ./mypumpkin [mysqldump|mysql] -hxx ... --dump-dir=xxx --threads=4
  - 导入10G速度，60min -> 20min (ssd)
  - 瓶颈在目标库cpu，其次是带宽和硬盘

3.1 为什么是mysqldump

- 逻辑导出导入
  - why not mydumper or mysqlpump?

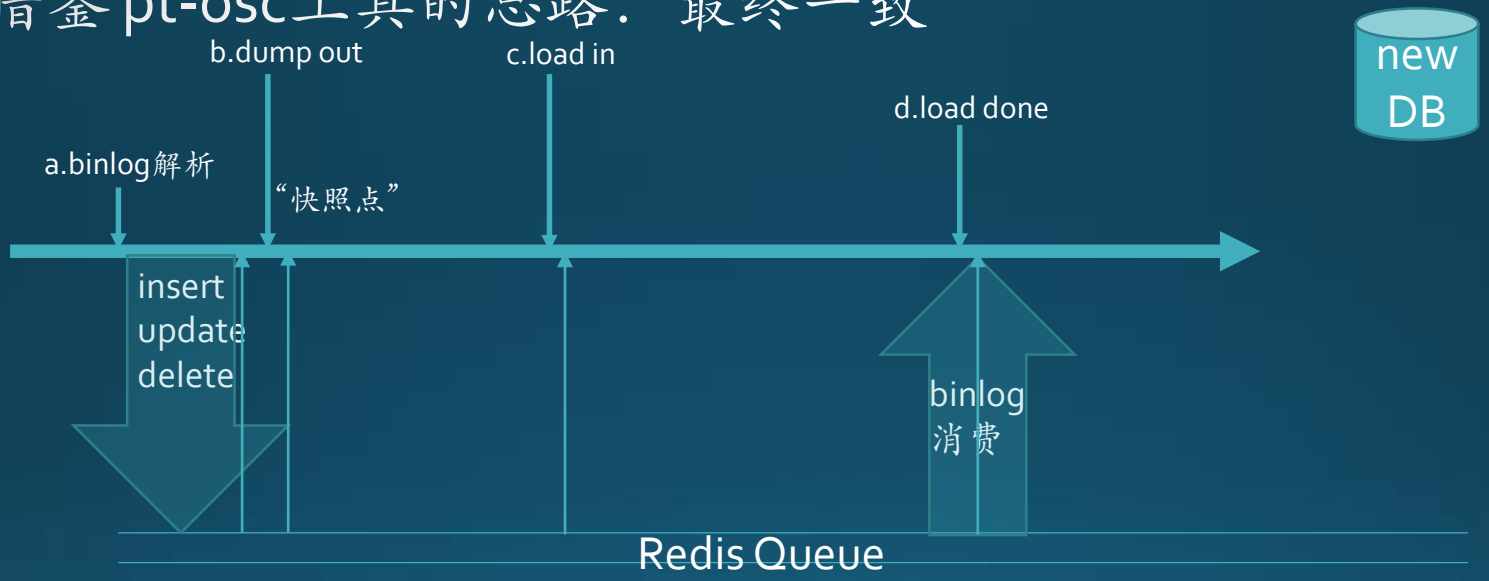
  - mysqldump
    - --single-transaction -tn ->不锁表
    - --default-character-set=latin1 ->必须以latin1导出
    - --no-set-names ->导出文件里不能有set names latin1
    - --skip-opt
    - --set-gtid-purged=OFF -eq ->阿里云无gtid权限
    - --skip-triggers ->忽略触发器
    - --max-allowed-packet=134217728 --net-buffer-length=819200 ->调整速度

    - --hex-blob ->针对二进制字段入varbinary
    - --insert-ignore ->只针对特殊情况

  - mysql --default-character-set=utf8mb4 --max-allowed-packet=134217728 --net-buffer-length=819200

3.2 dbsync数据同步服务

- 借鉴pt-osc工具的思路：最终一致

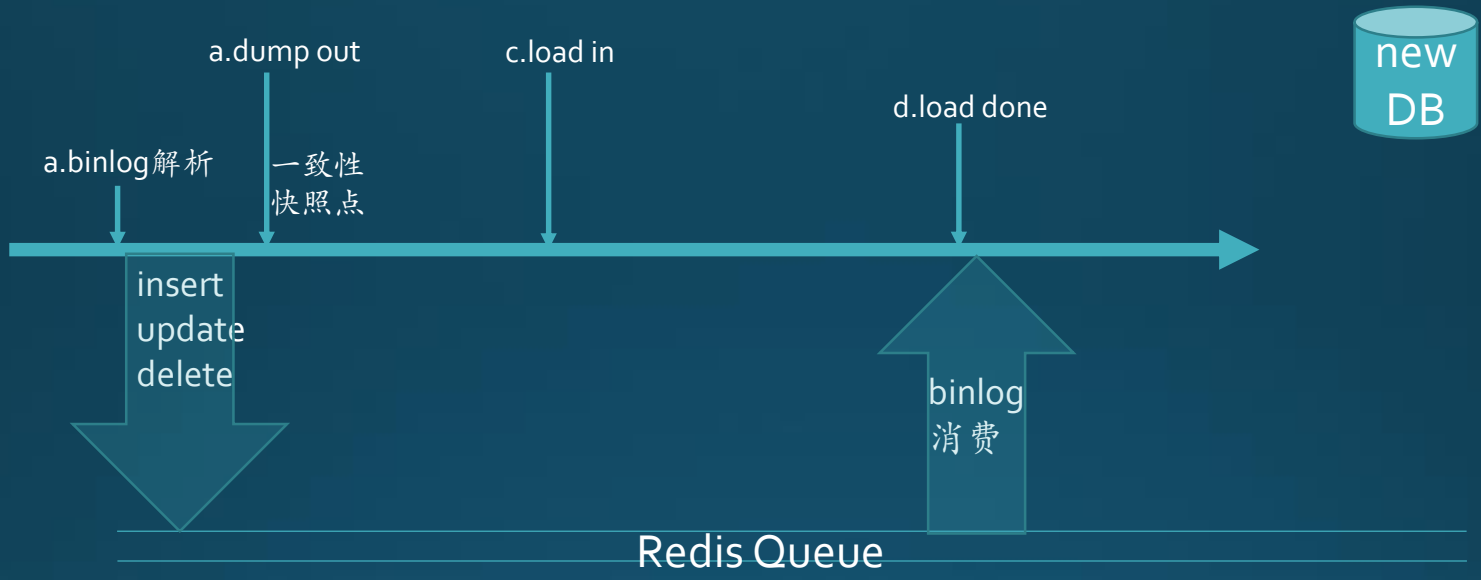


能否在binlog解析开始之后，立马进行消费？
a到b之间的dml如何处理？
同步异常时，怎样才不需要从头开始？

pt-osc：

- insert -> replace into
- update -> replace into
- delete -> delete ignore
- copy table -> insert ignore

3.2 dbsync数据同步服务



既无主键也无唯一索引

insert → replace into
insert into ... select ... from dual where not exists(select * from where [all cols])

update → replace into / insert ... on duplicate key update ...
update ... where [all cols]

delete → delete from ... where [all cols]

4.有哪些坑

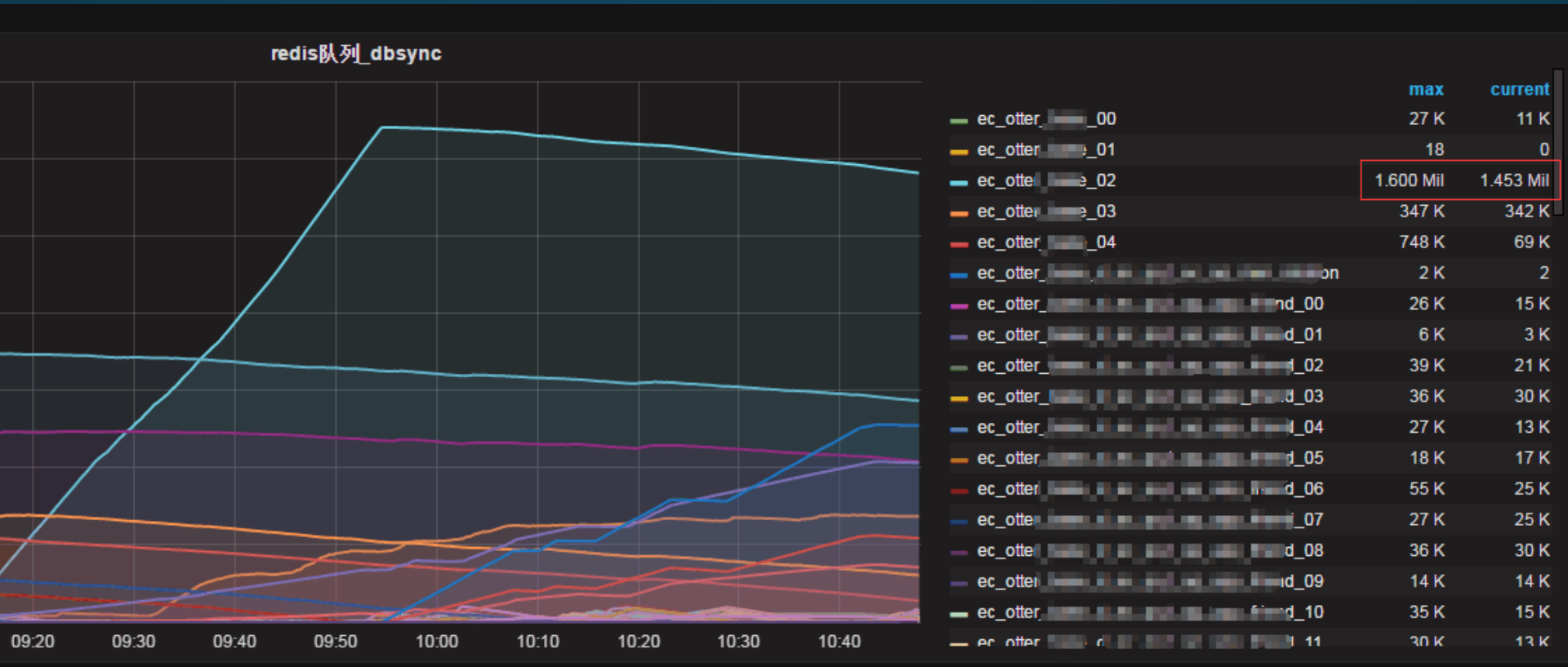
- 缺少字段同步异常
- 主从同步程序延迟严重
- redis队列溢出
- varbinary，enum类型
- 某些字段值在新库被截断，出现不一致
- 转utf8mb4后原联合主键超长768，报错
- emoji 表情乱码

缺少字段同步异常

- 无主键无唯一索引表同步异常
  - 原因：
    - 从binlog解析出的字段数，比从新库获取的表结构字段数要多
    - RDS为优化性能在最后一列了隐藏id
    - ### @9=87337958 /* LONGINT meta=0 nullable=0 is_null=0 */
  - 解决：拼凑sql时以新库表结构为准
- 迁移期间有人做DDL
  - 提前做好。转换期间（包括测试）不允许表结构变更

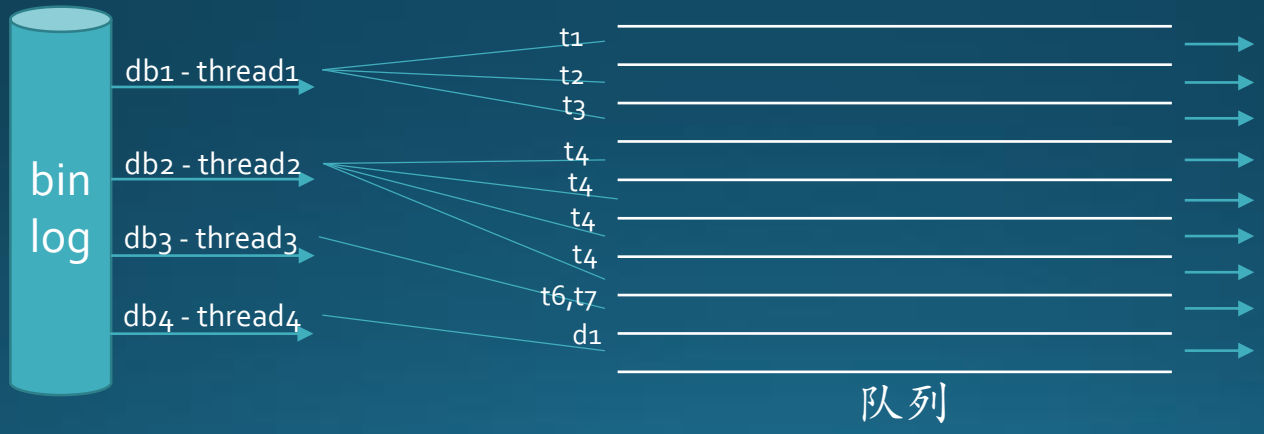
“主从同步”延迟严重

- 原因：
  - 部分表高频写入，队列消费不完
  - binlog落后好几个，切换前追赶不上



“主从同步”延迟严重

- 解决：
  - 高频表单独的队列
  - 一个表分多个队列，按主键散列
  - 解析binlog推送至Redis过程，改用批量提交效率会更高 (LPUSH list1 x y, BRPOP)

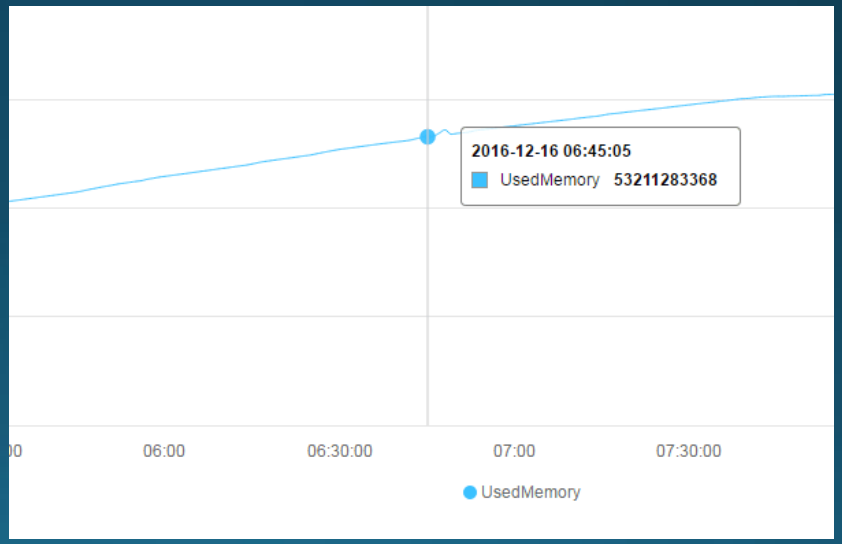


不能入简单多个队列，可能死锁，以及更新顺序

redis队列溢出

- 原因：
  - key逐出策略volatile-lru，expire Integer.MAX_VALUE
  - binlog预估过小

- 解决：
  - 不能设置过期，设为0
  - 宁愿报错，不要吞掉
  - ssdb-成本考虑



其它坑

- varbinary ， enum类型
  - canal将varbinary类型误解析成varchar
    - 解决：需要对TableMetaCache进行初始化
  - 枚举类型值存在非法的''
    - 解决：jdbc默认设置sql_mode=STRICT_TRANS_TABLES，改为''

- Error 1071: Specified key was too long; max key length is 767 bytes
  - 有个类似url索引字段原长度定义200，转utf8mb4后，200*4 > 767
    - 解决：1.非唯一索引时，改为前缀索引
      2. 需作为唯一索引时，该表使用utf8

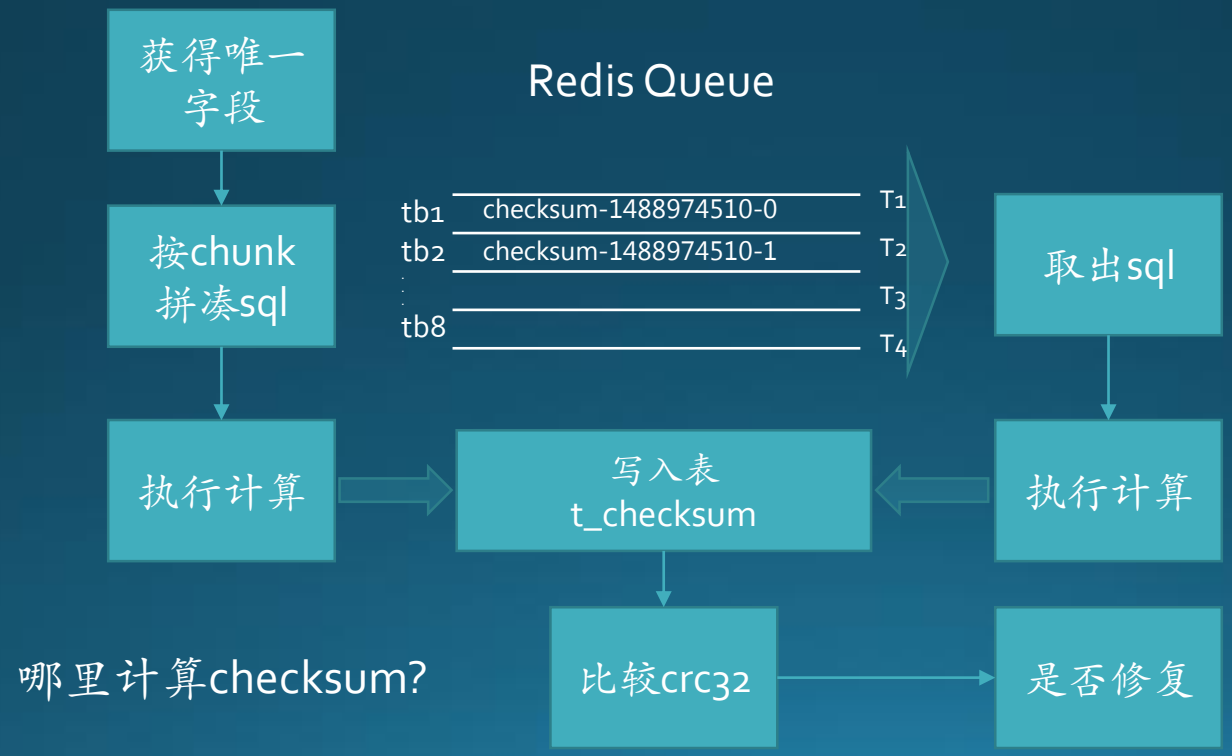
规范！

其它坑

- emoji 表情乱码
  - 包含表情的字符串插入失败 java.sql.SQLException: Incorrect string value: '\xF0\x9F\x8C\xB4\xE9\xA2...' for column 'f_nick' at row 1
    - 主要是mysql-connector 对utf8mb4编码支持有特殊要求：character-set-server=utf8mb4
    - 小心 --skip-character-set-client-handshake

5.一致性保证

- 非主从环境下数据一致性校验px-table-checksum
  - https://github.com/seanlook/px-table-checksum
  - 借鉴 pt-table-checksum思路



5.一致性保证

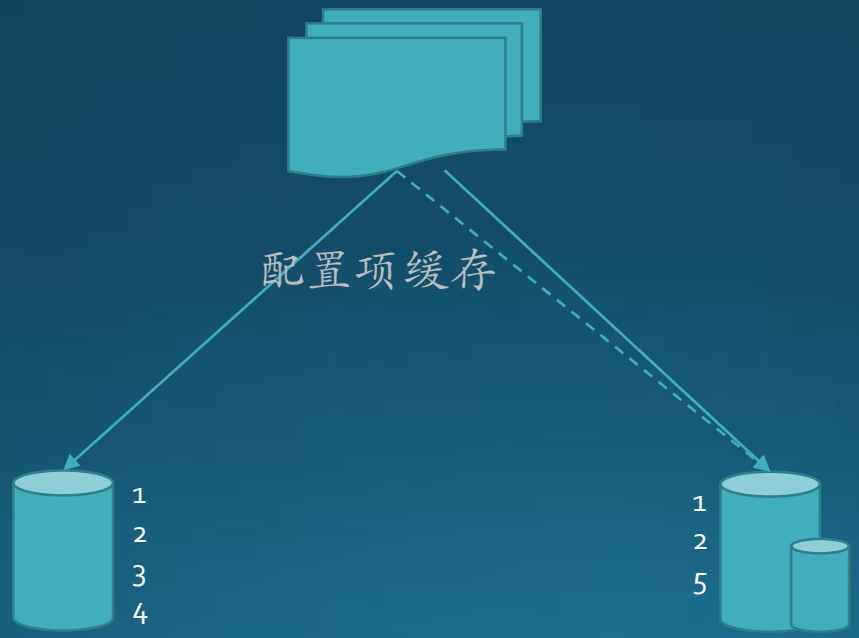
- px-table-checksum
  - 可配置化
    - DB_CHECKSUM, TABLES_CHECK
      REDIS_INFO, REDIS_QUEUE_C
      CHUNK_SIZE,
      DO_COMPARE
  - 检测并不完全准确
    - 可多次运行
- 配合自动化测试

```
CREATE TABLE `t_checksum` (
  `f_dbid` varchar(80) NOT NULL,
  `f_table_name` varchar(50) NOT NULL,
  `f_chunk_no` int(11) NOT NULL,
  `f_create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `f_schema_name` varchar(30) DEFAULT NULL,
  `f_min_id` varchar(50) NOT NULL,
  `f_max_id` varchar(50) NOT NULL,
  `f_chunk_crc32` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`f_dbid`,`f_table_name`,`f_chunk_no`,`f_create_time`),
  KEY `idx_tbname_maxid` (`f_table_name`,`f_max_id`),
  KEY `idx_chunkno` (`f_chunk_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

f_provice	f_city
美国	新泽西州月桂
黑龙江省哈尔	联通
广东省江门市	联通
黑龙江省哈尔	联通
北京市	北京数讯达通
河北省石家庄	联通

6.后面的故事

- 人为失误，数据写入了两个实例
  - 解决：分析ER图，主键数字高位 +1



6.后面的故事

- 一个查询出现严重的性能问题
  - select * from t1 left join t2 on t1.id=t2.id
    where t1.col1>a group by t1.id order by t1.id desc limit 10
  - 执行计划徘徊在PRIMARY 与 filesort 之间
  - 解决：
    - t1,t2记录一一对应，group by id无意义
    - group by id desc
- AUTO-IN
  - 历史表
  - 不使用

Message | Explain

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	a	range	PRIMARY,web_cor	web_corp	12	(Nu	196	Using index condition; Using where; Using temporary; Using filesort
1	SIMPLE	d	eq_ref	PRIMARY	PRIMARY	8	d_e	1	Using where

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	a	index	PRIMARY,web_cor	PRIMARY	8	(Null	30	Using where
1	SIMPLE	d	eq_ref	PRIMARY	PRIMARY	8	d_ec	1	Using where

70ms

10min

成果&心得

- MySQL存储空间节省30%
- 稳定的binlog订阅服务

- 创造工具，而不甘做工具使用者
- 困难就是机会

- 规范！

谢谢观赏！