DTCC2013

SQLAUTO——数据库SQL变更自动化实践

蔡歌@baidu

Baidu 百度

目录

当前数据库现状

- ✓ DBA 与 SQL 打交道的时间 > 70%
  - 优化
  - 变更
  - 故障处理
  - 备份恢复
  - 监控
  - 日志分析
  - ……

DBA的苦恼?

当前数据库现状

- ✓ 苦恼1：SQL越来越复杂
  - 超过1M的SQL(机器生成、业务统计需要)
  - 中文、法文、日文等多种语言
  - 业务端的偷懒：join/union/count/group by
  - 复杂的应用：事务、锁、触发器、授权
- ✓ 苦恼2：SQL变更越来越频繁
  - 人工处理费时费力
  - 数据库安全得不到保障

SQL变更到底有多少坑？

目录

一条SQL的地雷


```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
LIMIT 1
```

能否直接执行？

执行结果1

```
mysql> select * from t1 where c='db' limit 1;
+----+------+---+----+---------------------+
| id | a    | b | c  | d                   |
+----+------+---+----+---------------------+
|  1 | 网络     | 2 | db | 2013-04-12 00:00:00 |
+----+------+---+----+---------------------+
1 row in set (0.00 sec)

mysql> source utf8.sql;
Database changed
Query OK, 1 row affected, 2 warnings (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 1

mysql> select * from t1 where c='db' limit 1;
+----+----------+---+----+---------------------+
| id | a        | b | c  | d                   |
+----+----------+---+----+---------------------+
|  1 | 鏁版嵁搴擄     | 2 | db | 2013-04-12 00:00:00 |
+----+----------+---+----+---------------------+
```

✓扫雷第一步：处理字符集

常见字符集

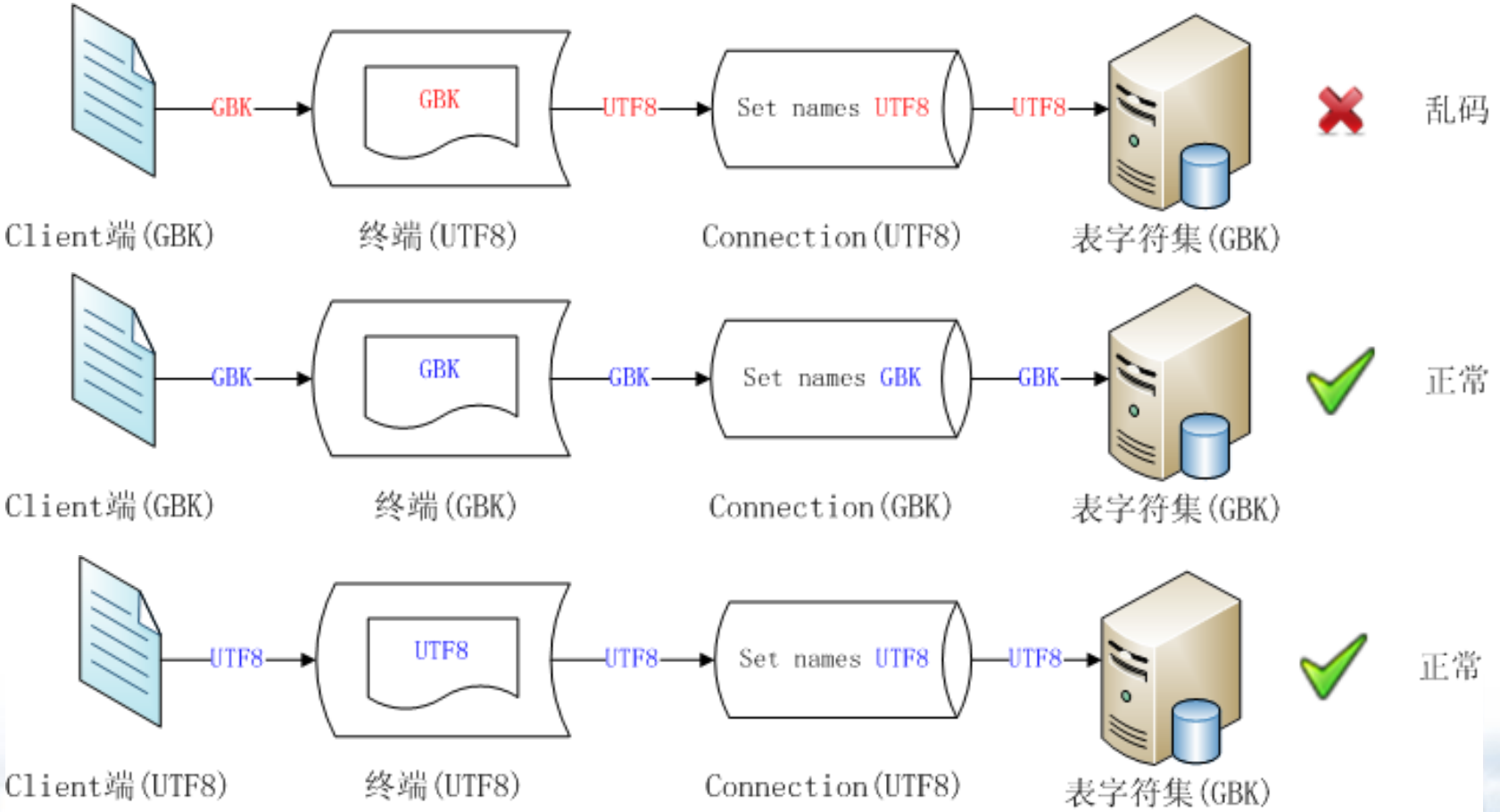
- ✓ Unicode
  - 2/4字节编码，容纳全世界所有语言字符的编码表
- ✓ UTF-8
  - 多字节编码，以字节为单位对Unicode进行再编码
- ✓ GBK
  - 双字节编码，汉字全集，与Unicode有转换表
- ✓ Latin1
  - 单字节编码，256个字符，支持任意字符集的无损转储

MySQL字符集

- ✓ Server端
  - 默认字符集：character_set_server
  - 作用：创建库表时不指定字符集时继承默认的
  - 动态修改：set global character_set_server=xxx
- ✓ Client端
  - 默认字符集：character_set_client
  - 作用：client和server交互时进行字符集转换
  - 动态修改：set names xxx

字符集转换

✓ client ➔ connection ➔ server 三者一致？

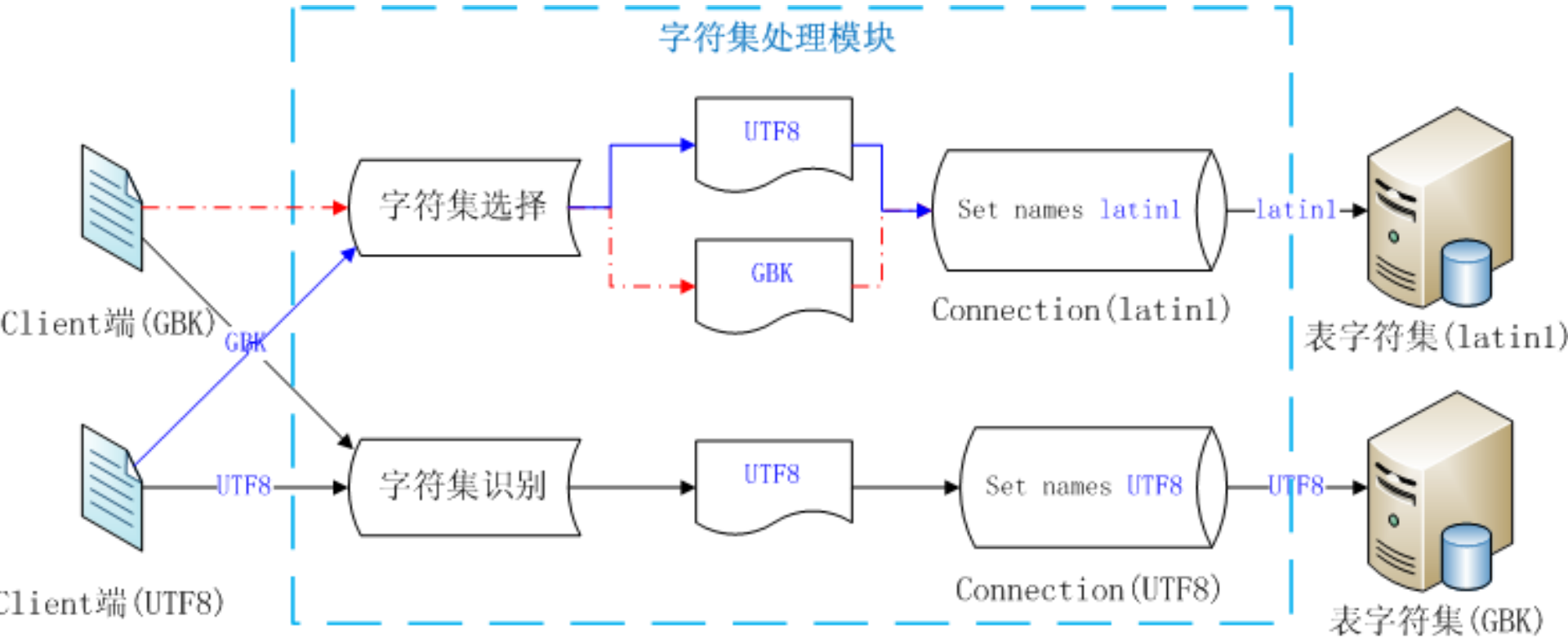


字符集转换关系

Client端	set names	数据库	结果
GBK	latin1		
latin1		正常	
UTF8	latin1		
latin1		正常	
GBK		GBK	
GBK		正常	
UTF8	UTF8		
GBK		正常	
GBK	GBK		
UTF8		正常	
UTF8		UTF8	
UTF8		正常	
UTF8		GBK	
GBK		部分正常	

我们的方案

✓ 字符集处理模块



一条SQL的地雷


```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
LIMIT 1
```

✓ 字符集处理完毕

继续执行？

执行结果2


```
mysql> show warnings;
+-------+------+-----------------------------------------------------+
| Level | Code | Message                                             |
+-------+------+-----------------------------------------------------+
| Note  | 1592 | Statement may not be safe to log in statement format. |
+-------+------+-----------------------------------------------------+
```

- ✓ 原因：不带order by的limit会导致不安全复制

- ✓ 扫雷第二步：SQL规则检查

如何识别SQL规则报错？

- ✓方法一：肉眼识别？
  - 全角与半角
  - 多个空格，少个回车
- ✓方法二：自己造轮子
  - 逐个分支覆盖
  - 字段名是中文？

```
drop table testl;
drop table testl;
```

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr, ...
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name']
    [FROM table_references
    [WHERE where_definition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_definition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC] , ...]
```

我们的方案

- ✓ SQL规则检查器
  - 基于MySQL解析器修改： 支持全部SQL语法
  - 横向扩展：自定义规则
    - delete/update必须带where
    - limit必须带Order by
    - 不允许临时表
    - ……

一条SQL的地雷

```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
ORDER BY d DESC
LIMIT 1
```

- ✓ 字符集处理完毕
- ✓ 规则检查通过

继续执行？

执行结果3


```
mysql> update t1 set a='数据库' where `c
`='database';
ERROR 1054 (42S22): Unknown column 'c' in
'where clause'
```

✓ 原因：字段不存在

执行结果3

```
session1:
begin;
update t1 set a='数据库' where `c`='database';

session2:  update t1 set a='数据库' where id=6901797;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

- ✓原因：session1锁表导致session2锁等待超时
- ✓扫雷第三步：评估SQL风险

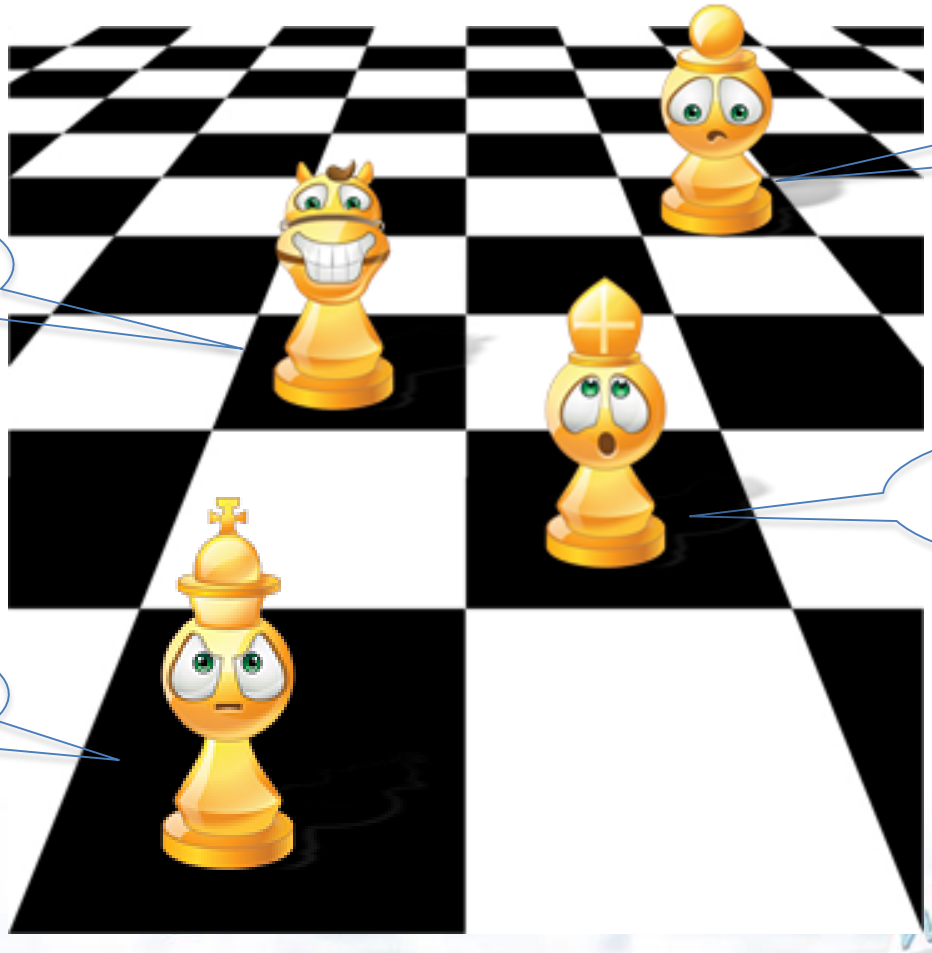
传统的执行SQL方式

- ✓ 摸着石头过河：走一步算一步

转变思路？

DTCC2013

✓ 磨刀不误砍柴工：提前探雷



order by+limit存在风险点

影响500万行数据

updated_time字段不存在

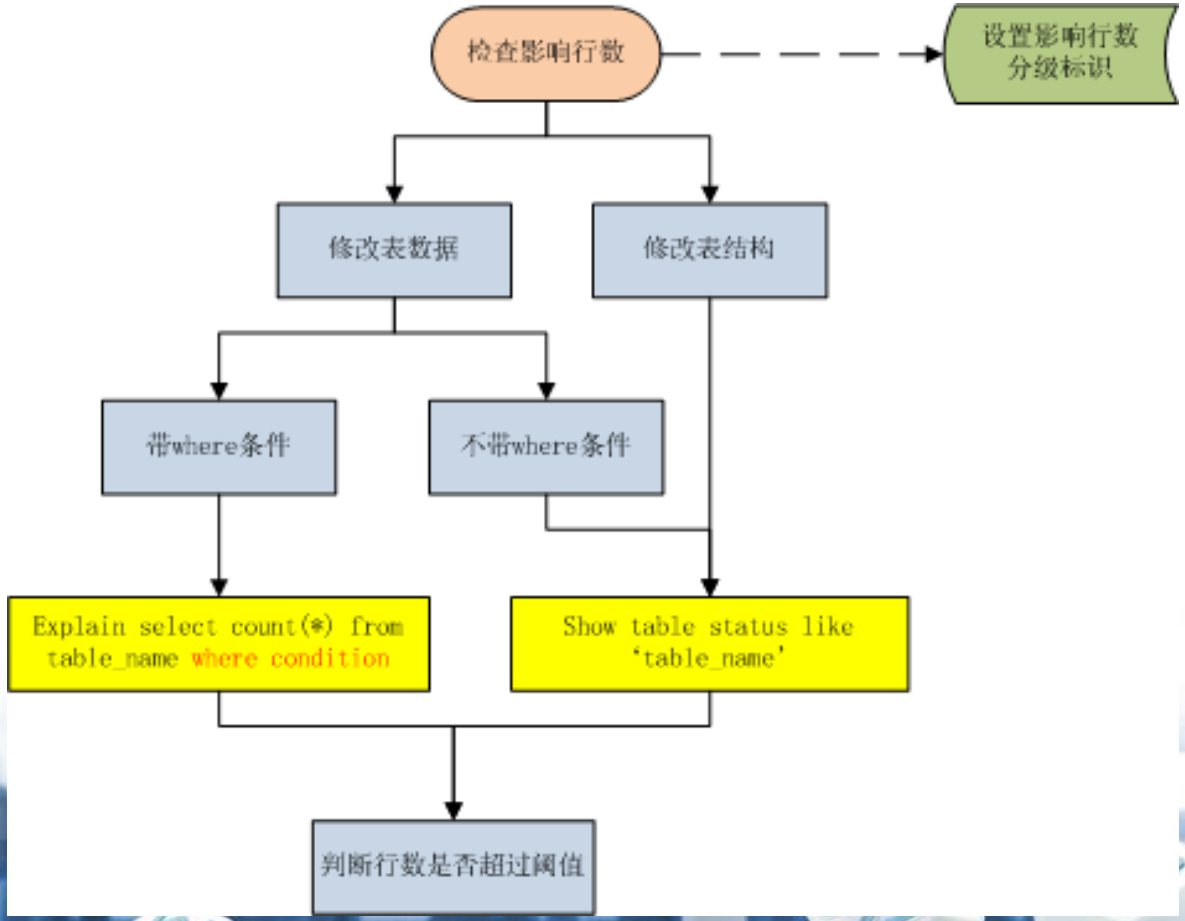
where条件未使用索引

SQL有哪些风险？

- ✓ 功能：
  - 索引：是否使用主键/索引？
  - 字段、库表信息：是否冲突？存在？拼写问题？
- ✓ 性能：
  - 影响范围：影响行数？锁表？事务？同步延迟？
  - 语法注意：
    - 不带where？带limit不带order by？
    - 临时表？Select …into？子查询？
- ✓ 集群：
  - 主从状态：同步状态，主库负载
  - mysql版本：主从是否一致

我们的方案

- ✓ SQL评估策略——划分类别，建立规则库
  - 功能：SQL类型过滤、会话状态、语法分支
  - 性能：库表存在、使用索引、影响行数、执行耗时、备份耗时



一条SQL的地雷


```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
ORDER BY d DESC
LIMIT 1
```

- ✓ 字符集处理完毕
- ✓ 规则检查通过
- ✓ SQL风险评估完毕

继续执行？

执行结果4

主库

id	book_name	book_type	book_class
6901797	高性能MySQL	数据库技术	database

从库

id	book_name	book_type	book_class
6901797	高性能MySQL	网络技术	database

✓原因：主从数据不一致，需要回滚

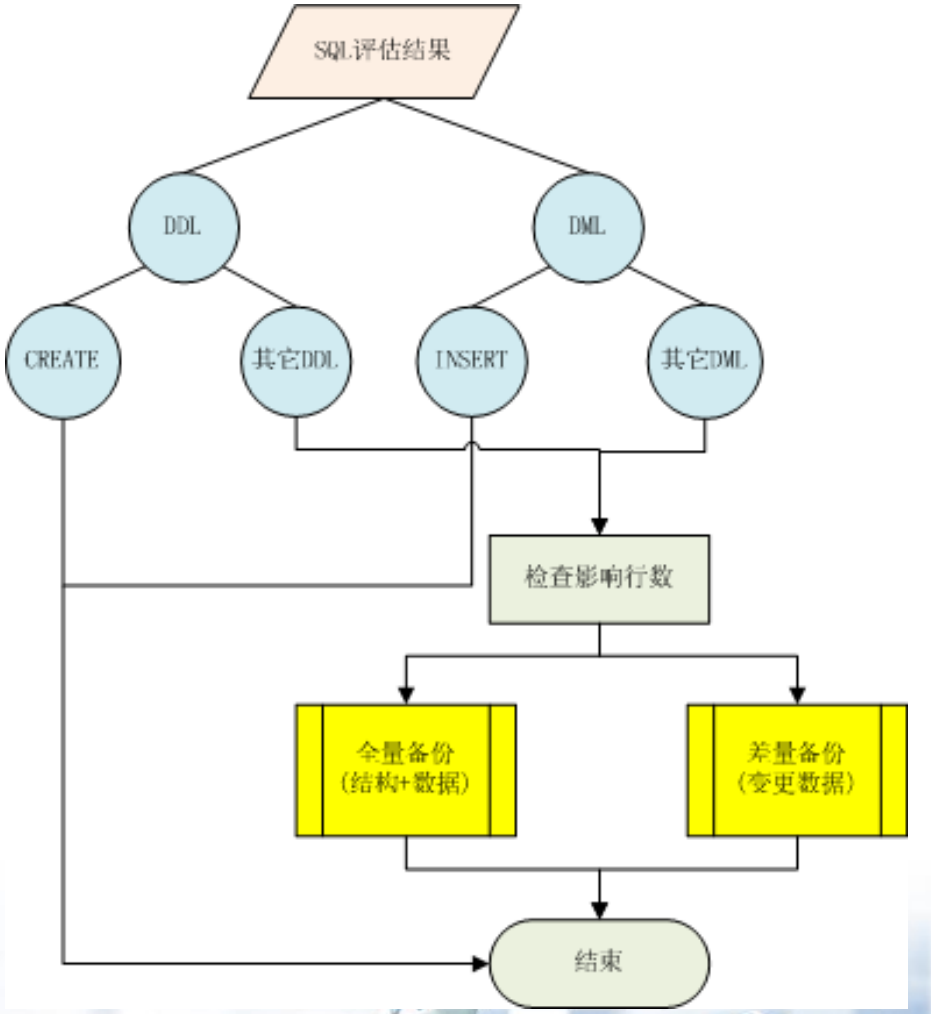
✓扫雷第四步：执行前的数据备份

SQL级别备份

- ✓ 如何备份？
  - 数据类
    - 全表：dump？停同步？拷文件？建软链？
    - 变更数据：select…where？select…into？create+insert？
  - 结构类
    - 结构定义：存储过程、触发器、视图

我们的方案

- ✓ SQL备份模型——按需备份
  - 修改schema的DDL：全量备份
  - 只修改数据的DML：差量备份
  - 修改结构的：备份结构定义



一条SQL的地雷


```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
ORDER BY d DESC
LIMIT 1
```

- ✓ 字符集处理完毕
- ✓ 规则检查通过
- ✓ SQL风险评估完毕
- ✓ 按需备份完成

继续执行？

执行结果5

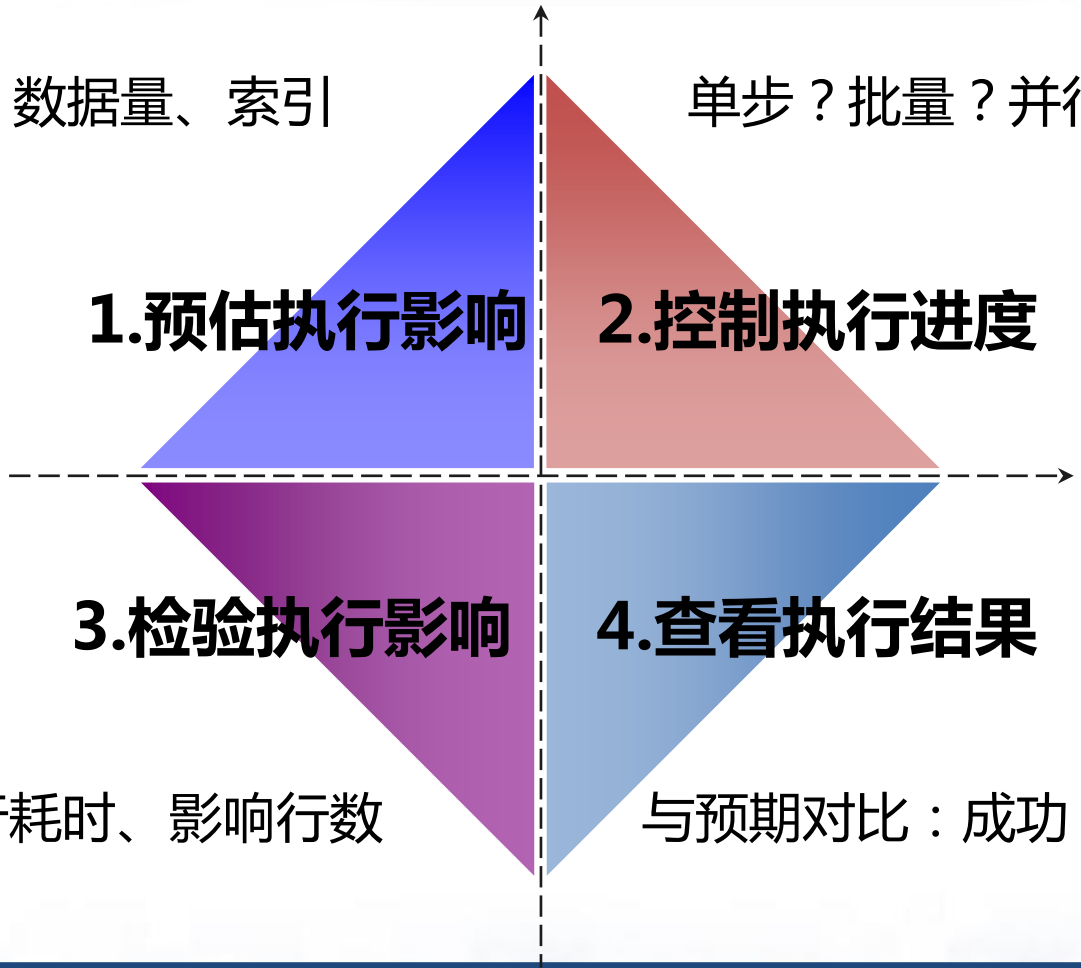
CDbException

CDbConnection failed to open the DB connection: SQLSTATE[00000] [1040] Too many connections

✓原因：锁表加重主库负载，引起连接数打满

✓扫雷第五步：选择执行方案

执行SQL = ‘mysql<’ ?



根据实际情况选择不同的执行方案

我们的方案

- ✓ SQL执行方案
  - 区分SQL类型、使用索引情况、数据量

数据量	小数据量		大数据量	
索引情况	使用索引	未使用索引	使用索引	未使用索引
DML执行方案	直接执行			先OSC建索引，再执行
DDL执行方案	直接执行			直接OSC

  - OSC：Online Schema Change

一条SQL的地雷


```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
ORDER BY d DESC
LIMIT 1
```

- ✓ 字符集处理完毕
- ✓ 规则检查通过
- ✓ SQL风险评估完毕
- ✓ 按需备份完成
- ✓ 执行完毕

结束？

执行结果6

```
mysql> select * from t1 where c='db' order by d desc limit 1;
+----+--------+---+----+---------------------+
| id | a      | b | c  | d                   |
+----+--------+---+----+---------------------+
|  1 | 网络   | 2 | db | 2013-04-12 00:00:00 |
+----+--------+---+----+---------------------+
1 row in set (0.00 sec)

mysql> UPDATE t1 SET a='数据库' and b=1 WHERE c='db' ORDER BY d DESC LIMIT 1;
Query OK, 1 row affected, 2 warnings (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from t1 where c='db' order by d desc limit 1;
+----+---+---+----+---------------------+
| id | a | b | c  | d                   |
+----+---+---+----+---------------------+
|  1 | 0 | 2 | db | 2013-04-12 00:00:00 |
+----+---+---+----+---------------------+
```

✓原因：执行成功，但业务数据错误

✓扫雷第六步：确认执行结果

我们的方案

✓ SQL检查点

- DDL：对比变更前后表结构
- 小数据量DML：对比变更前后数据
- 低误差：同一会话内获取

```
select col1 from tbl_name where condition
……

update tbl_name set col1=xx where condition
```

变更前

id	name
12	hanxin
15	zhaoyun

变更后

id	name
12	machao
15	guanyu

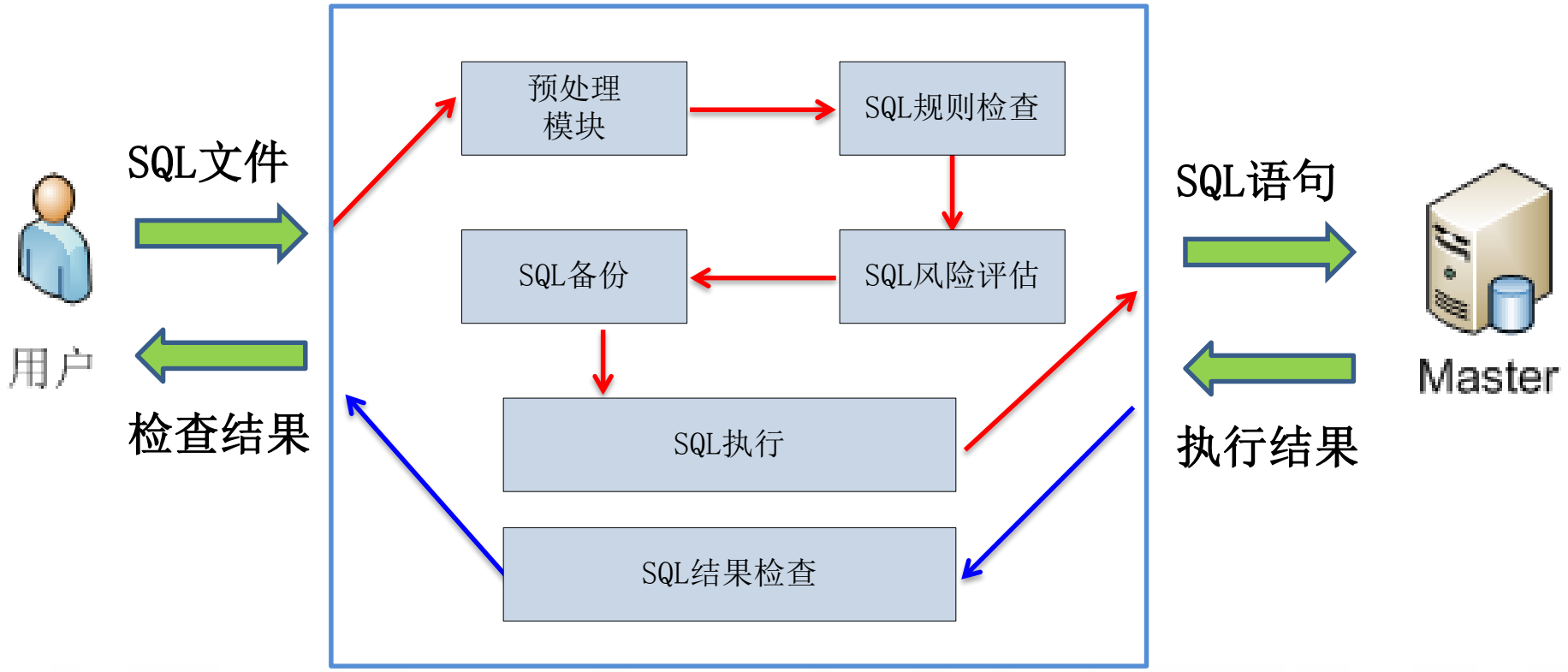
一条SQL的地雷


```
UPDATE t1
SET a='数据库' and b=1
WHERE c='db'
ORDER BY d DESC
LIMIT 1
```

- ✓ 字符集处理完毕
- ✓ 规则检查通过
- ✓ SQL风险评估完毕
- ✓ 按需备份完成
- ✓ 执行完毕
- ✓ 确认执行结果

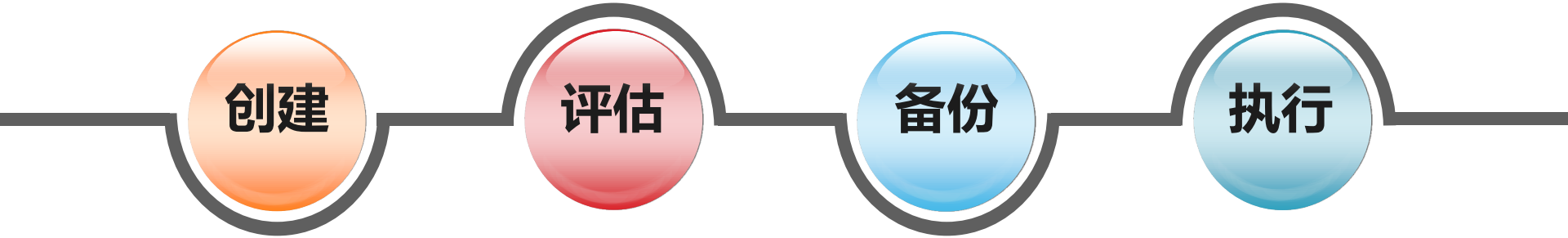
结束！

回顾：一条SQL的执行过程



目录

SQLAUTO理念



✓产品定位

- 字符集透明化
- SQL语法全面覆盖
- 在线风险评估
- 按需备份
- 一键式执行
- 可视化检查

✓关键要求

- 高效率
- 低风险

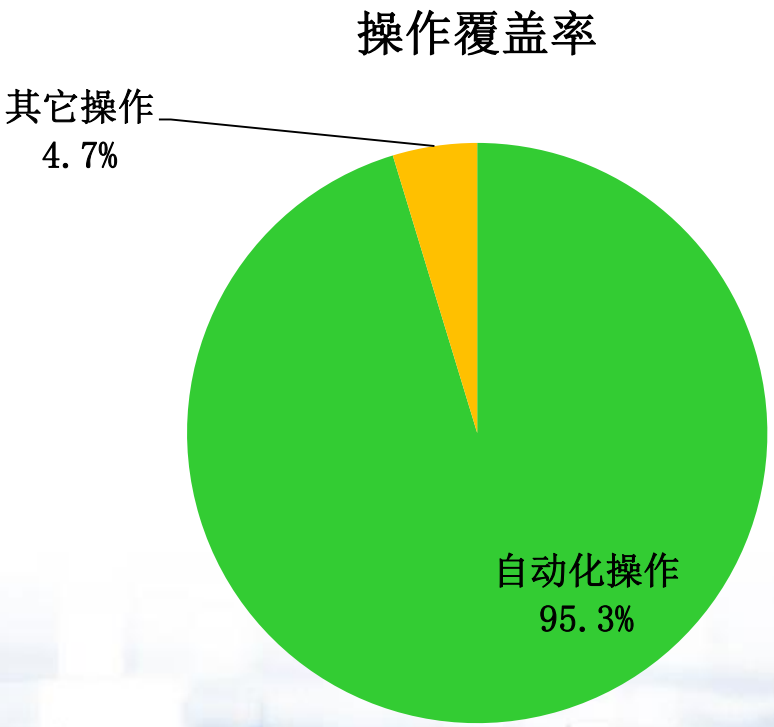
目录

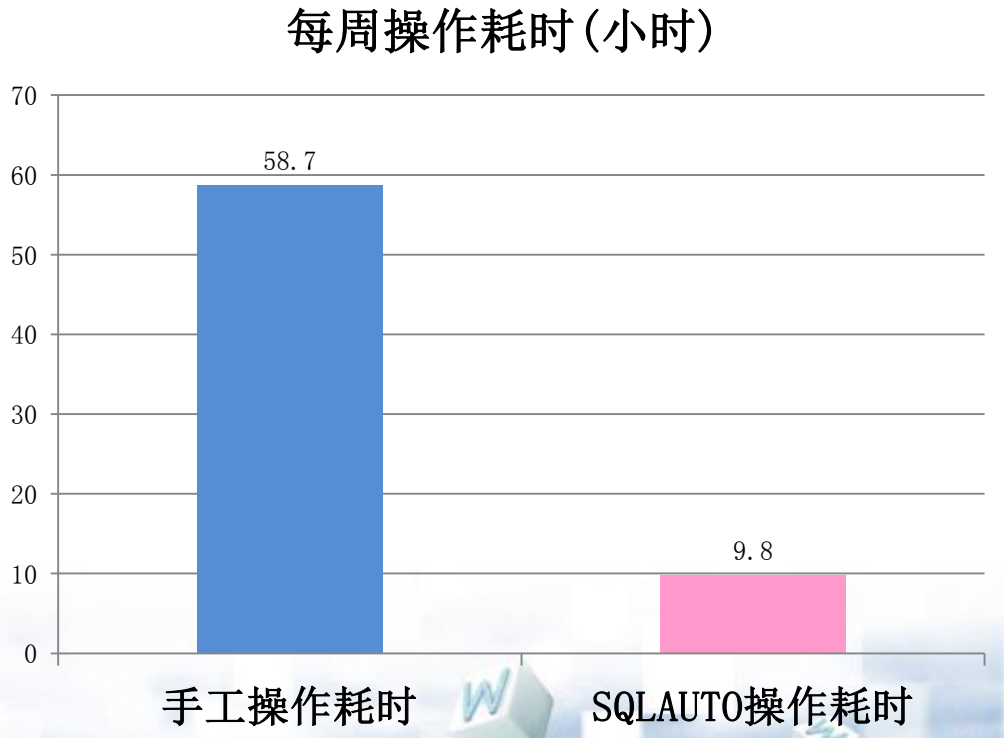
解决的问题

- ✓ 抽丝剥茧：打破复杂SQL的魔咒
- ✓ 提升效率：自动化代替手工
- ✓ 安全卫士：再也不用担心新人的误操作了

SQLAUTO效果说明

- ✓ 操作覆盖率：95%
- ✓ 效率提升：5倍
- ✓ 安全性提升：零误操作





目录

未来的发展

SQL准入　SQL优化　SQL监控

申请资源、服务准入

服务运维

服务下线

- SQLAUTO：覆盖整个数据库生命周期

谢谢！