为数据赋能

---腾讯TDSQL分布式金融级数据库前沿技术

李海翔 @那海蓝蓝

Query optimization technology

Informix

DTCC专家组

数据库架构

研发中心副总经理

那海蓝蓝

PostgreSQL

GreenPlum

首席数据库架构师

测试中心总监

MySQL

人大金仓

Distributed technology

DB Kernal

腾讯金融云

李海翔

腾讯金融云数据库技术专家

中国人民大学信息学院工程硕士企业导师

Oracle公司MySQL全球开发组
Optimizer Team

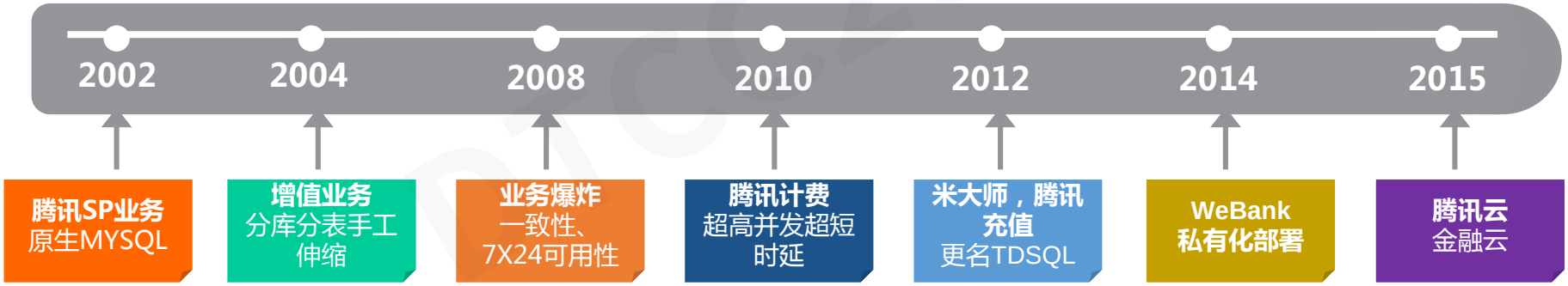
Transaction processing technology

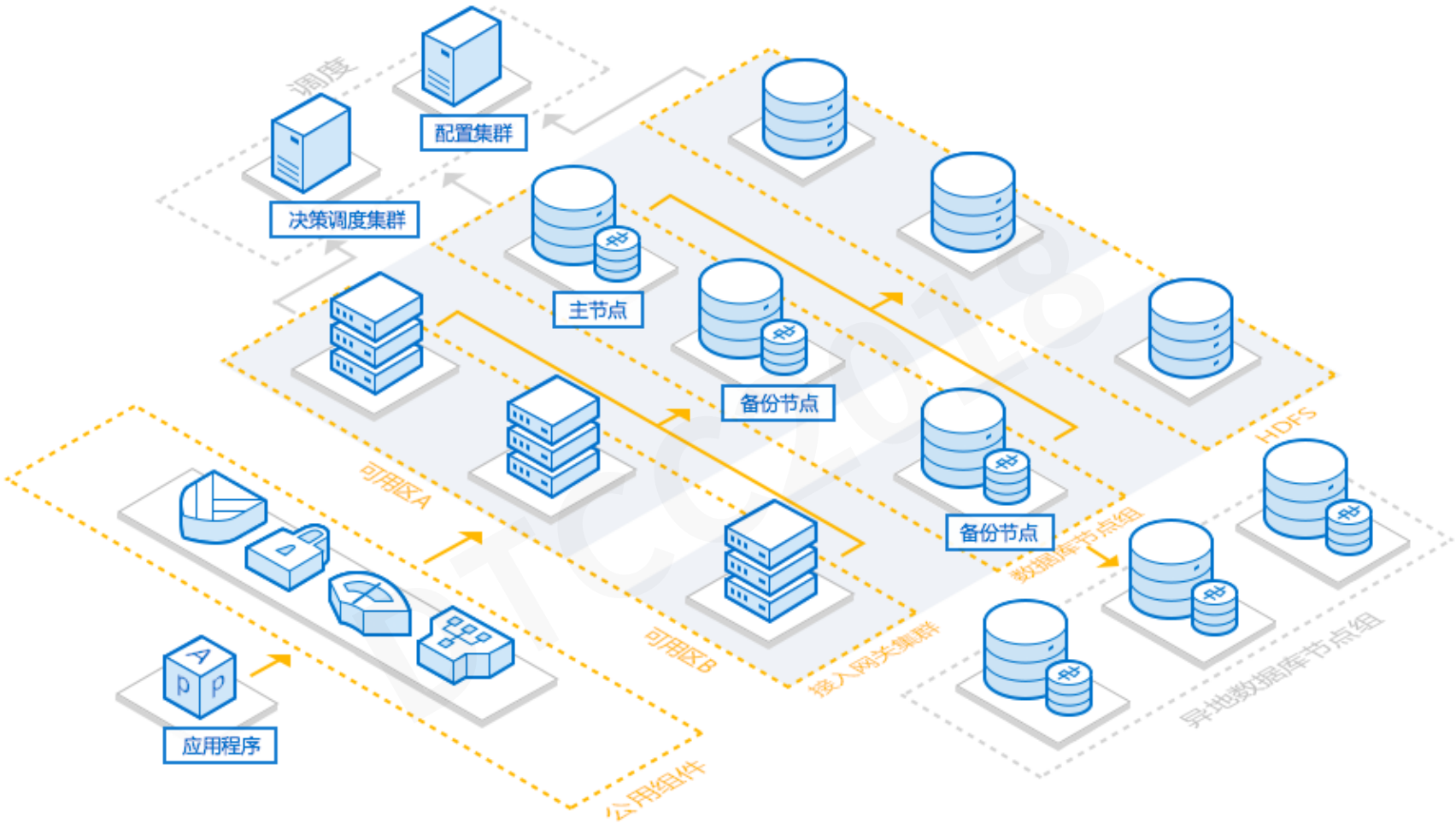
目录
CONTENTS

TDSQL分布式数据库

面向金融类业务，十年积累，亿级账户验证

腾讯公司内与计费、充值、转账、财务等核心系统90%以上都使用TDSQL！




调度
配置集群
决策调度集群
主节点
备份节点
备份节点
HDFS
可用区A
可用区B
接入网关集群
数据库节点组
异地数据库节点组
应用程序
公用组件

TDSQL，数据库的特点

跨机房部署
网络故障不影响业务

数据强同步
主备数据完全一致

三重保障
集群内保障3套节点，单点故障整体稳定

金融级安全
支持物理专享，支持数据库审计，支持加密等

可用性：99.999%

数据可靠性：99.99999%

目录
CONTENTS

What is the pain point of your business?
你的业务痛点是什么？

- **Application development is complex，应用开发复杂：**各种应用使用业务日志
  - 生产日志：不同应用创造了不同格式的日志
  - 存储日志：大量冗余信息（源库 + 日志）
  - 分析日志：读入、解析、分析

- **The logic of the data is fragmented，逻辑割裂：**业务按日分表，只能按日进行结算，不能灵活、方便的计算。如计算任意时间段内的数据，日表在物理上割裂了数据按时间的逻辑连续特性，需要指定若干个特定的日表才能进行计算。

- **No real time characteristics，实时特性丢失：**如上两个问题，隐含地，意味着进行计算的数据需要导入到一个新的分析系统进行计算，导出/导入数据的过程也带来了资源和时间的消耗、使得分析系统难以具备实时计算特性。

- **Complexity of data management，数据管理复杂：**另外，日志等信息，是历史数据，需要长期保存。对不同格式、海量的日志进行存储与管理，这成为一个巨大的挑战。

Can the database solve your pain point?
数据库能解决痛点吗？

- The modern database system only stores the current value of the data
- 现代的数据库系统只保留有数据的当前值

- Historical data is discarded
- 历史数据被丢弃

- The value of the data is valuable
- 数据有价值

- Are you willing to throw away something valuable?
- 你愿意抛弃有价值的东西吗？

一切过往（数据的历史和状态）兼可追溯

Tencent solution
腾讯解决之道-FmSMC

- **Full-state temporal data model，全时态数据模型**。数据的诞生、修改、消亡的全过程管理。
  - **Full-state data，**全态数据
  - **Temporal data，**时态数据

- **Storage，全时态数据存储**。现有的数据库系统，只能保存数据的当前状态（当前态数据）。
  - 历史态数据怎么存储？
  - 海量历史态数据怎么存储？
  - 海量全时态数据怎么存储？

- **Management，全时态数据管理**。双时态数据的管理。

- **Computing，全时态数据计算**。
  - 历史态数据可见性判断
  - 如何利用索引高效读取历史态数据？
  - 海量全时态数据计算环境？

一切过往（数据的历史和状态）兼可追溯

Other solution
其他解决方案

- Therory，理论界
  - Full-state data，全态数据模型，没有
  - Temporal data，双时态数据模型，支持
  - 1993年出版的《Temporal Databases: Theory, Design, and Implementation》被称为“世界第1本关于时态数据库专著”。书中列出13种最具影响的时态数据库模型：TDB、HRDM、TempSQL、IXRM、TRM、HSQL、TQuel、TRC、TEER、TDM、OODAplex、Object History、Temporal Deductive Database。

- SQL：2011：对双时态模型进行了定义
  - 包含事务时间的表不删除数据，
  - 历史和当前数据同时存在于表中且都能被查询，但只有当前数据能被更新、删除。
  - 事务时间由两个时间域组成，事务开始时间表示“元组被新增的时间”，事务结束时间表示“元组被移除的时间”

- DB2 V10：历史表存储历史数据，全面支持双时态数据模型

- SQL Server 2016：只支持事务时态数据模型，但有较多辅助功能

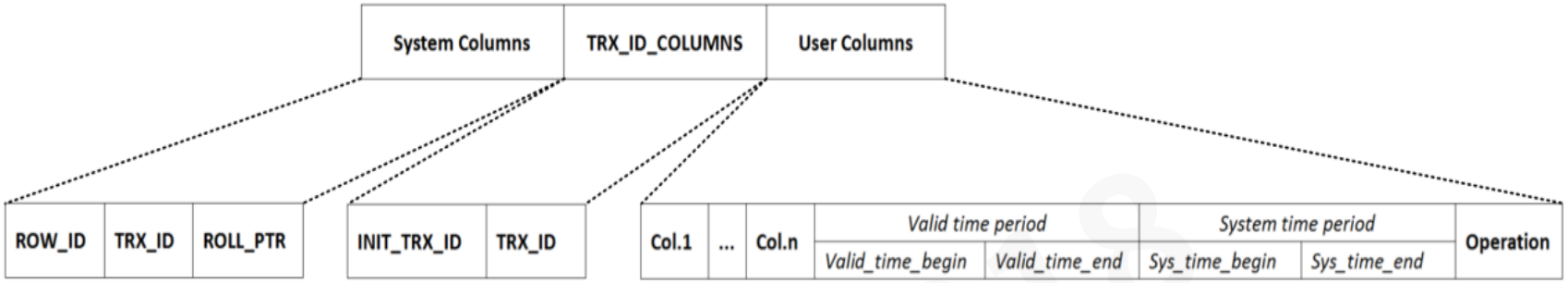
T-TDSQL New Feature

T-TDSQL特性表

T-TDSQL 功能对比（√支持；×不支持；通过配套工具支持）

比较项	TDSQL	T-TDSQL
事务型数据库的一切特性	√	√
分布式特性	√	√
有效时间类应用（履历管理、合同管理、图书管理、档案管理）	×	√
事务时间类应用（按 DML 操作追溯数据的变迁史）	×	√
数据**增量抽取**	×	√
数据**增量计算**（对账：**总账快对、细账精确**）	×	√
轨迹数据管理（类似轨迹数据库）	×	√
消失的数据查询（历史值和所发生操作）--**数据历史不丢失**	通过	√
海量数据存储（本地存储、网络存储）	通过	√
数据按**热度**存储	×	√
历史数据集群化存储、管理（**HTAP**）	×	√
安全：数据订正、历史追踪、联机闪回	通过	√
数据重演：在数据库引擎层提供分析应用负载	×	√
基于用户**数据的历史变迁**进行用户画像**多维度**分析	×	√

Full-state temporal data model
全时态数据模型



	PK	Name	Balance	Valid Time		Transaction Time		Operation
				Valid_start_time	Valid_end_time	Sys_start_time	Sys_end_time	
Init	(1, 2011-01-01, NOW)	Kim	100	2011-01-01	NOW	2011-01-01 00:00:00	UC	INSERT
Init	(2, 2011-01-01, NOW)	Peter	150	2011-01-01	NOW	2011-01-01 00:00:00	UC	INSERT
Init	(3, 2011-01-01, NOW)	John	70	2011-01-01	NOW	2011-01-01 00:00:00	UC	INSERT
op1	(4, 2010-02-01, NOW)	Jimmy	100	2010-02-01	NOW	2011-02-01 00:00:00	UC	INSERT
op2	(1, 2011-01-01, 2012-01-01)	Kim	100	2011-01-01	2012-01-01	2011-01-01 00:00:00	2012-01-01 00:00:00	UPDATE(DELETE)
op2	(1, 2012-01-01, NOW)	Kim	200	2012-01-01	NOW	2012-01-01 00:00:00	UC	UPDATE(INSERT)
op3	(1, 2012-01-01, 2013-01-01)	Kim	200	2012-01-01	2013-01-01	2012-01-01 00:00:00	2013-01-01 00:00:00	UPDATE(DELETE)
op3	(1, 2013-01-01, NOW)	Kim	300	2013-01-01	NOW	2013-01-01 00:00:00	UC	UPDATE(INSERT)
op4	(3, 2011-01-01, 2014-01-01)	John	70	2011-01-01	2014-01-01	2011-01-01 00:00:00	2014-01-01 00:00:00	DELETE

Storing Data--Format
储存数据--转储格式

PK	Name	Balance	Valid Time		Transaction Time	
			Valid_start_time	Valid_end_time	Sys_start_time	Sys_end_time
1	Kim	100	2011-01-01	NOW	2011-01-01 00:00:00	UC
2	Peter	150	2011-01-01	NOW	2011-01-01 00:00:00	UC
3	John	70	2011-01-01	NOW	2011-01-01 00:00:00	UC

- *Column Store数据格式*
- *支持Parquet、RCFile、ORCFile等列存格式*

Typical Cases
典型案例

增量计算：总账快对、细账精确

Old Method

User	
对账数据准备的时间点	账户余额
2018/5/5 0:00	100
2018/5/6 0:00	400

流水表相同

只能对总账

New Method

精确的细账

User

User_id	Name	Balance	Operation	Init_trx_id	Trx_id
1	Kim	100	UPDATE	T0	T1
1	Kim	200	UPDATE	T1	T2
3	John	70	DELETE	T3	T4
4	Jimmy	0	UPDATE	T5	T5
4	Jimmy	100	INSERT	T5	UC
1	Kim	400	INSERT	T2	UC

Water

Water_id	User_id	Trx_id	Change
1	1	T1	+100
2	1	T2	+100
3	3	T4	-70
Missing 4	4	T5	+100

ResultSet

User_id	Trx_id	Before	After	Change
1	T1	100	200	+100
1	T2	200	400	+100
3	T4	70	0	-70
4	T5	0	100	NULL

Before	After	Change	对账结果
M1	M2	M2-M1	正确
M1	M2	NULL	流水缺失
M1	M2	(M2-M1)'	流水记录有误
NULL	NULL	M3	流水误增
M1	M2'	M2-M1	账户表更新有误
M1	NULL	M2-M1	账户表没有更新
NULL	M2	NULL	账户表误增元组

Typical Cases
典型案例

有效时间应用

1. 创建有效时间表：
表Employee，ID为员工编号，Dept为员工任职部门编号，Valid_Start和Valid_End为有效时间起、止，Valid_Period为有效时间区间。

```
CREATE TABLE Employee(ID INT, Dept INT, Valid_Start DATETIME, Valid_End DATETIME, PRIMARY KEY(ID, Valid_Start), PERIOD FOR Valid_Period (Valid_Start, Valid_End));
```

2. 有效时间表的插入：
新入职三位员工，ID分别为2018001,2018002,2018003，暂且认为该员工不会调离所属部门。

```
INSERT INTO Employee (ID, Dept, Valid_Start, Valid_End) VALUES (2018001, 1, '2018-1-1 00:00:00', '9999-12-31 23:59:59'), (2018002, 1, '2018-1-1 00:00:00', '9999-12-31 23:59:59'), (2018003, 1, '2018-1-1 00:00:00', '9999-12-31 23:59:59');
mysql> SELECT * FROM Employee;
+----------+------+---------------------+---------------------+
| ID       | Dept | Valid_Start         | Valid_End           |
+----------+------+---------------------+---------------------+
| 2018001 |    1 | 2018-01-01 00:00:00 | 9999-12-31 23:59:59 |
| 2018002 |    1 | 2018-01-01 00:00:00 | 9999-12-31 23:59:59 |
| 2018003 |    1 | 2018-01-01 00:00:00 | 9999-12-31 23:59:59 |
+---------+-------+---------------------+---------------------+
```

3. 有效时间表的更新,更新有效时间字段： **ID为2018002的员工将于2019-1-1 00:00:00调离部门1**

```
UPDATE Employee SET Valid_End = '2019-1-1 00:00:00' WHERE ID=2018002;
mysql> SELECT * FROM Employee;
+----------+------+---------------------+---------------------+
| ID       | Dept | Valid_Start         | Valid_End           |
+----------+------+---------------------+---------------------+
| 2018001 |    1 | 2018-01-01 00:00:00 | 9999-12-31 23:59:59 |
| 2018002 |    1 | 2018-01-01 00:00:00 | 2019-01-01 00:00:00 |
| 2018003 |    1 | 2018-01-01 00:00:00 | 9999-12-31 23:59:59 |
+---------+-------+---------------------+---------------------+
```

有效时间应用

5. 有效时间表的查询： 传统SELECT仍可行，T-TDSQL提供AS OF、FROM TO和BETWEEN AND三种时态查询方式：

a. **AS OF查询**

AS OF tv：在tv时刻，元组所表示的关系在现实世界正确

在2018-7-15 00:00:00时，各员工就职的部门

```
SELECT * FROM Employee WHERE PERIOD Valid_Period AS OF '2018-7-15 00:00:00';
mysql> SELECT * FROM Employee WHERE PERIOD Valid_Period AS OF '2018-7-15 00:00:00';
+----------+------+---------------------+---------------------+
| ID       | Dept | Valid_Start         | Valid_End           |
+----------+------+---------------------+---------------------+
| 2018001  |    2 | 2018-01-01 00:00:00 | 2019-01-01 00:00:00 |
| 2018003  |    2 | 2018-06-01 00:00:00 | 2019-01-01 00:00:00 |
+----------+------+---------------------+---------------------+
```

b. **FROM TO查询**

FROM tv1 TO tv2：在时间段[tv1, tv2)内，元组所表示的关系在现实世界正确，时间不及tv或超过tv2，该元组所表示的关系在现实世界不成立。

有效时间在2018-6-1 00:00:00至2019-6-1 00:00:00之间的员工信息

```
SELECT * FROM Employee WHERE PERIOD Valid_Period FROM '2018-6-1 00:00:00' TO '2019-6-1 00:00:00';
mysql> SELECT * FROM Employee WHERE PERIOD Valid_Period FROM '2018-6-1 00:00:00' TO '2019-6-1 00:00:00';
+----------+------+---------------------+---------------------+
| ID       | Dept | Valid_Start         | Valid_End           |
+----------+------+---------------------+---------------------+
| 2018003  |    2 | 2018-06-01 00:00:00 | 2019-01-01 00:00:00 |
+----------+------+---------------------+---------------------+
```

c. **BETWEEN AND查询**

BETWEEN tv1 AND tv2：在时间段[tv1, tv2)内，元组所表示的关系在现实世界正确。

在2018-6-1 00:00:00至2019-6-1 00:00:00时间段内，员工任职过的部门

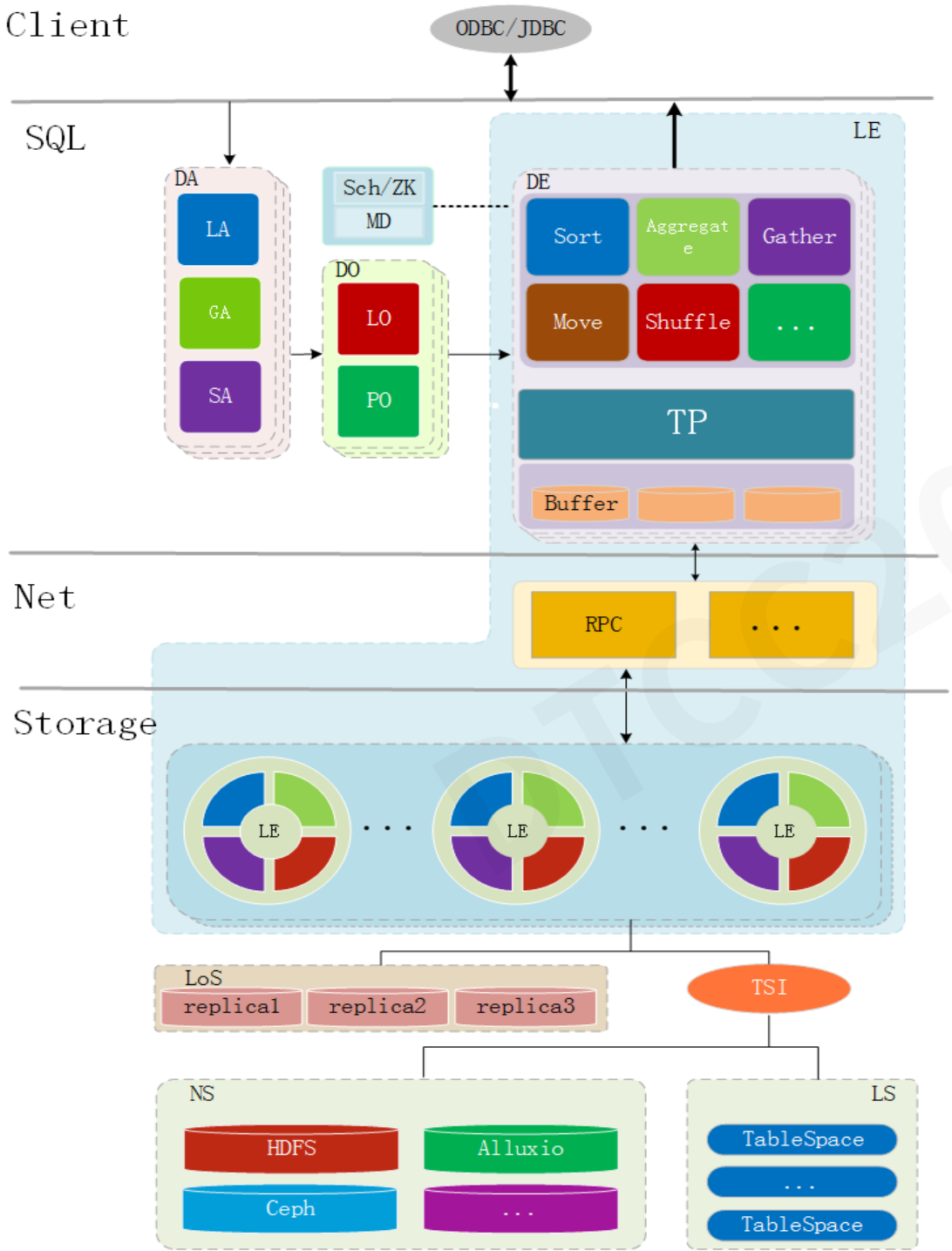
```
SELECT * FROM Employee WHERE PERIOD Valid_Period BETWEEN '2018-6-1 00:00:00' AND '2019-6-1 00:00:00';
mysql> SELECT * FROM Employee WHERE PERIOD Valid_Period BETWEEN '2018-6-1 00:00:00' AND '2019-6-1 00:00:00';
+----------+------+---------------------+---------------------+
| ID       | Dept | Valid_Start         | Valid_End           |
+----------+------+---------------------+---------------------+
| 2018001  |    2 | 2018-01-01 00:00:00 | 2019-01-01 00:00:00 |
| 2018003  |    1 | 2018-01-01 00:00:00 | 2018-06-01 00:00:00 |
| 2018003  |    2 | 2018-06-01 00:00:00 | 2019-01-01 00:00:00 |
| 2018003  |    1 | 2019-01-01 00:00:00 | 9999-12-31 23:59:59 |
+----------+------+---------------------+---------------------+
```

数据安全：数据订正、联机闪回、历史追踪

- T-TDSQL提供联机的数据闪回，可以查询过去某个时间段的数据库状态。
- 而读取数据库的过去某个时间点的数据状态（历史态被储存而不是被清理），依据的是三种快照读操作。这是闪回实现的原理。
- 基于此原理，实现了多种类型的联机闪回功能，包括：闪回查询，闪回删除，闪回归档。

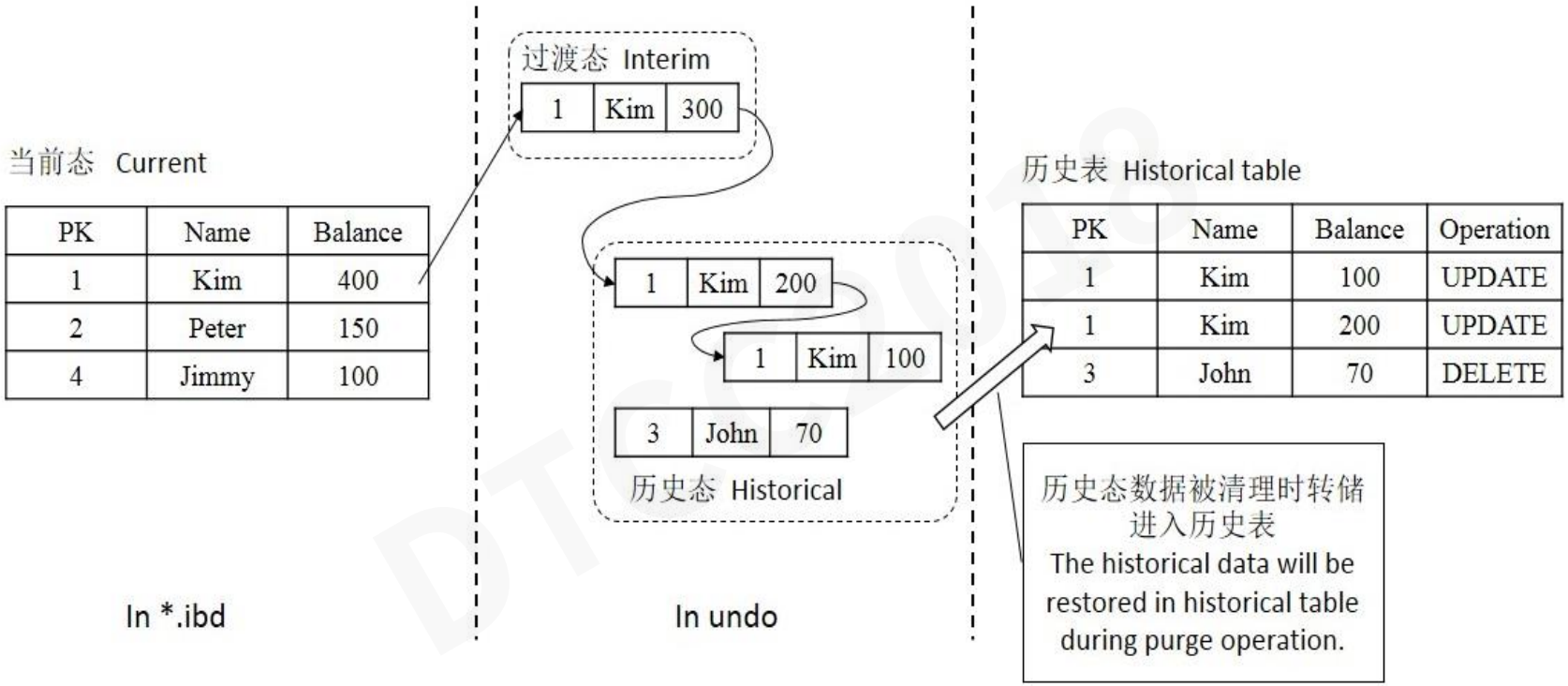
1. 闪回查询：可以查询过去某个时间段的数据库状态，可将某个表回退到过去某个时间点。
2. 闪回删除：闪回删除可以将一个已经被Drop的表还原。相应的索引也会被还原。
3. 闪回归档：闪回数据归档可使表具有回退到过去任何时间点。

目录
CONTENTS



Sch: Scheduler 调度器/ZK: ZooKeeper
MD: MetaData 元数据
DA: Distributed Analyzer 分布式分析器
LA: Lexical Analysis 词法分析器
GA: Grammatical Analysis 语法分析器
SA: Semantic Analysis 语义分析器
DO: Distributed Optimizer 分布式优化器
LO: Logic Optimizer 逻辑优化器
PO: Physical Optimizer 物理优化器
LC: Logical Executor: 逻辑执行器
DE: Distributed Executor 分布式执行器
TP: Transaction Processing 事务处理
LE: Local Executor 本地执行器
LS: Local Storage 本地存储
LoS: Logic Storage 逻辑存储
NS: Net Storage 网络存储
TSI: Tencent Storage Interface 腾讯存储接口

Storing temporal Data—When
储存时态数据--转储时机



Principle
原理

Algorithm 1 Snapshot-based incremental data extraction algorithm

Input: R: Source relation; s_start: Snapshot represents the start point; s_stop: Snapshot represents the stop point;
Output: $I(R)$: Incremental data set

```
1:  initial I(R) = ∅;
2:  repeat
3:      Get current positioned record r_i in relation R;
4:      opT = isVisible(r_i, s_start, s_stop, 1)
5:      if opT != 0 then                          ▷ this version is visible
6:          put r_i,opT into I(R)
7:      end if
8:      while r_i.prev()! = NULL do
9:          r_i = r_i.prev();
10:         opT = isVisible(r_i, s_start, s_stop, 0)
11:         if opT != 0 then                      ▷ this version is visible
12:             put r_i,opT into I(R)
13:         end if
14:     end while
15: until Relation Scan Finished
```

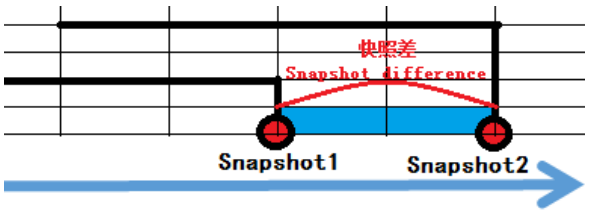
历史数据可见性算法

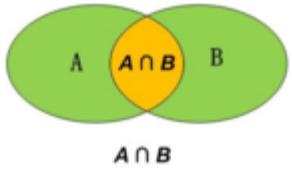
Algorithm 2 Record visibility judgment algorithm

Input: r_i: A given version of a record; s_start: Snapshot represents the start point; s_stop: Snapshot represents the stop point; is_first: 1-latest version, 0-previous version;
Output: opT: operation tag, 0-invisible, 1-insert, 2-update, 3-delete;

```
1:  initial opT = 0;
2:  if s_start.createTime < r_i.commit_time < s_stop.createTime then
3:      if r_i.isDelete == false then
4:          return opT=1;
5:      else
6:          if is_first then
7:              return opT=3;
8:          else
9:              return opT=1;
10:         end if
11:     end if
12: else
13:     return opT=0;
14: end if
```

快照差增量抽取算法





数学上，两个集合的对称差是只属于其中一个集合，而不属于另一个集合的元素组成的集合。

Principle
原理

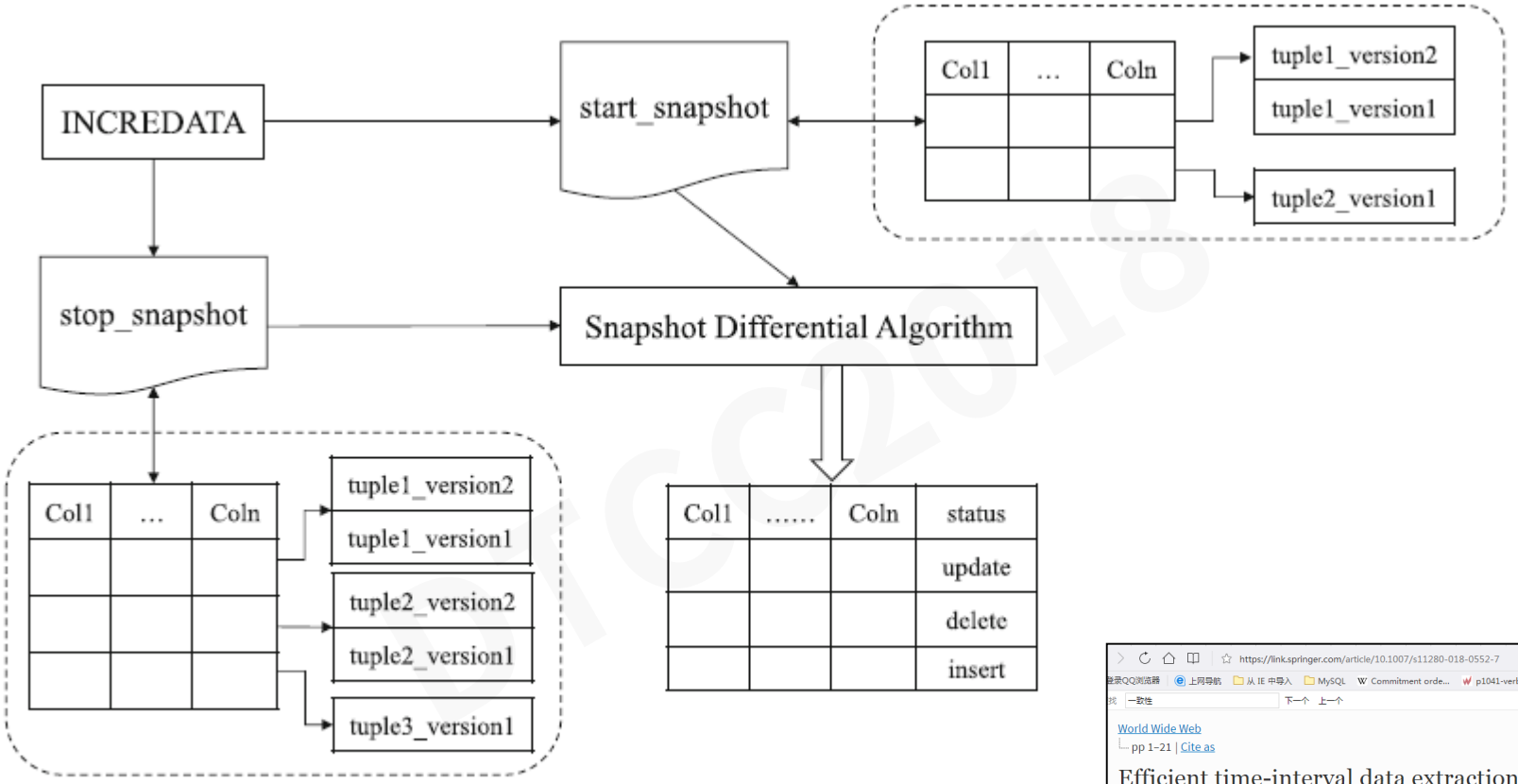
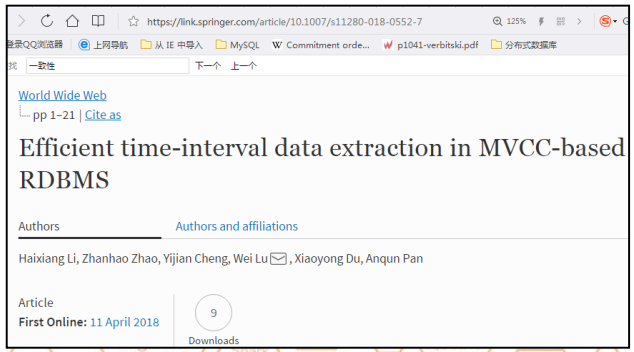


Figure 4 The overall system architecture


https://link.springer.com/article/10.1007/s11280-018-0552-7

World Wide Web
pp 1–21 | Cite as

Efficient time-interval data extraction in MVCC-based RDBMS

Authors | Authors and affiliations

Haixiang Li, Zhanhao Zhao, Yijian Cheng, Wei Lu, Xiaoyong Du, Anqun Pan

Article
First Online: 11 April 2018

9
Downloads

T-TDSQL

TDSQL Full-state Temporal Database System

Tencent

Please Follow Us

Historical data are valuable

Business is a sword, Technology is only a shield

T-TDSQL
TDSQL Full-state Temporal Database System
Tencent

Please Follow Us

Historical data are valuable
Business is a sword, Technology is only a shield

为数据赋能

Empower For Data

腾讯TDSQL分布式金融级数据库前沿技术

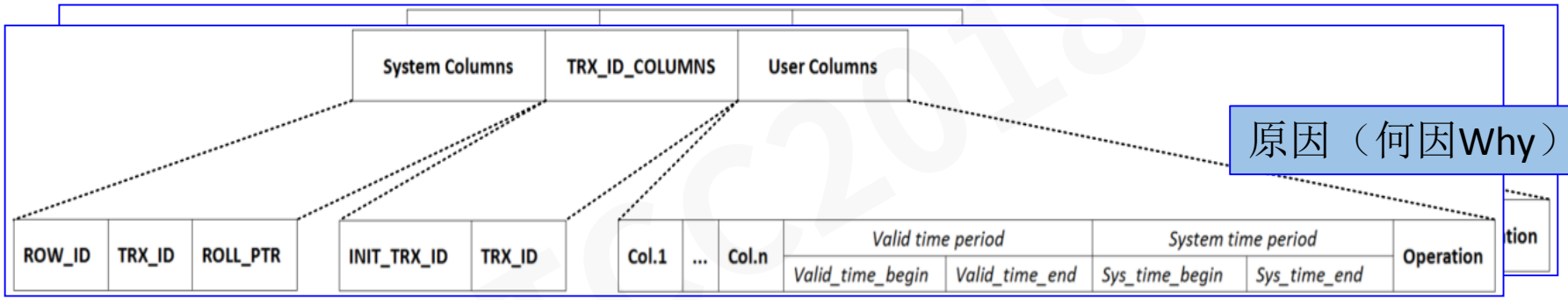
Empower For Data
为数据赋能

5W—
原因（何因Why）、对象（何事What）、地点（何地Where）、时间（何时When）、人员（何人Who）、方法（何法How）

地点（何地Where）--表对象

人员（何人Who）

对象（何事What）



原因（何因Why）

LineAge

时间（何时When）

数据智能

Acknowledgments

This work was supported by
The Tencent Company and
The Renmin University of China.

THANKS

TDSQL@Tencent

距离金融业务最近的数据库

The Nearest Database To The Financial Business

讲师申请
联系电话（微信号）：18612470168

关注“ITPUB”更多
技术干货等你来拿~

与百度外卖、京东、魅族等先后合作系列分享活动

微学堂

让学习更简单

微学堂是以ChinaUnix、ITPUB所组建的微信群为载体，定期邀请嘉宾对热点话题、技术难题、新产品发布等进行移动端的在线直播活动。

截至目前，累计举办活动期数60+，参与人次40000+。

ITPUB学院

ITPUB学院是盛拓传媒IT168企业事业部（ITPUB）旗下
企业级在线学习咨询平台
历经18年技术社区平台发展
汇聚5000万技术用户
紧随企业一线IT技术需求
打造全方式技术培训与技术咨询服务
提供包括企业应用方案培训咨询（包括企业内训）
个人实战技能培训（包括认证培训）
在内的全方位IT技术培训咨询服务

ITPUB学院讲师均来自于企业
一些工程师、架构师、技术经理和CTO
大会演讲专家1800+
社区版主和博客专家500+

培训特色

无限次免费播放
随时随地在线观看
碎片化时间集中学习
聚焦知识点详细解读
讲师在线答疑
强大的技术人脉圈

八大课程体系

基础架构设计与建设
大数据平台
应用架构设计与开发
系统运维与数据库
传统企业数字化转型
人工智能
区块链
移动开发与SEO

联系我们

联系人：黄老师
电　话：010-59127187
邮　箱：edu@itpub.net
网　址：edu.itpub.net
培训微信号：18500940168