FUJITSU

shaping tomorrow with you

浅析PostgreSQL事务处理机制

南京富士通南大软件技术有限公司
陈华军
2013/12/06

目录

FUJITSU

- PostgreSQL简介
- 什么是事务
- 如何实现事务
- WAL
- MVCC
- Lock
- 各DB并发表现对比
- QA

PostgreSQL简介

FUJITSU

- 开源的对象关系数据库
- 起源于1977 年Michael Stonebraker 领导的加州伯克利分校的 ingres项目
- 支持大部分SQL标准(最新标准SQL:2011)
- 提供丰富的特性：复杂查询、MVCC、online backup， WAL 、PITR，Stream Replication，Audit ，Pool ，Parallel…
- 易于定制扩展（函数，操作符，数据类型，索引，过程语言）
- 可运行于大多数主流系统（Linux,UNIX,Windows）
- 丰富的编程接口:C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC
- 基于BSD-风格许可证 PostgreSQL License，可免费使用，修改，发布，开源，闭源

Michael Stonebraker

数据库发展史上几个重量级人物之一。
1992 年提出对象关系数据库模型，领导开发了Ingres, Postgres, Vertica, Streambase, Illustra, VoltDB, SciDB...

PostgreSQL的应用现状

FUJITSU

Database流行度排行

193 systems in ranking, September 2013

Rank	Last Month	DBMS	Database Model	Score	Changes
1.	1.	Oracle	Relational DBMS	1529.61	-14.83
2.	3.	Microsoft SQL Server	Relational DBMS	1313.78	+8.82
3.	2.	MySQL	Relational DBMS	1305.76	-19.07
4.	4.	PostgreSQL	Relational DBMS	182.23	+0.01
5.	5.	DB2	Relational DBMS	172.25	+9.31
6.	6.	MongoDB	Document store	152.19	-3.80
7.	7.	Microsoft Access	Relational DBMS	146.71	-4.18
8.	9.	SQLite	Relational DBMS	82.78	+3.33
9.	8.	Sybase	Relational DBMS	75.35	-6.24
10.	11.	Cassandra	Wide column store	51.69	+4.98

http://db-engines.com/en/ranking

日本市场

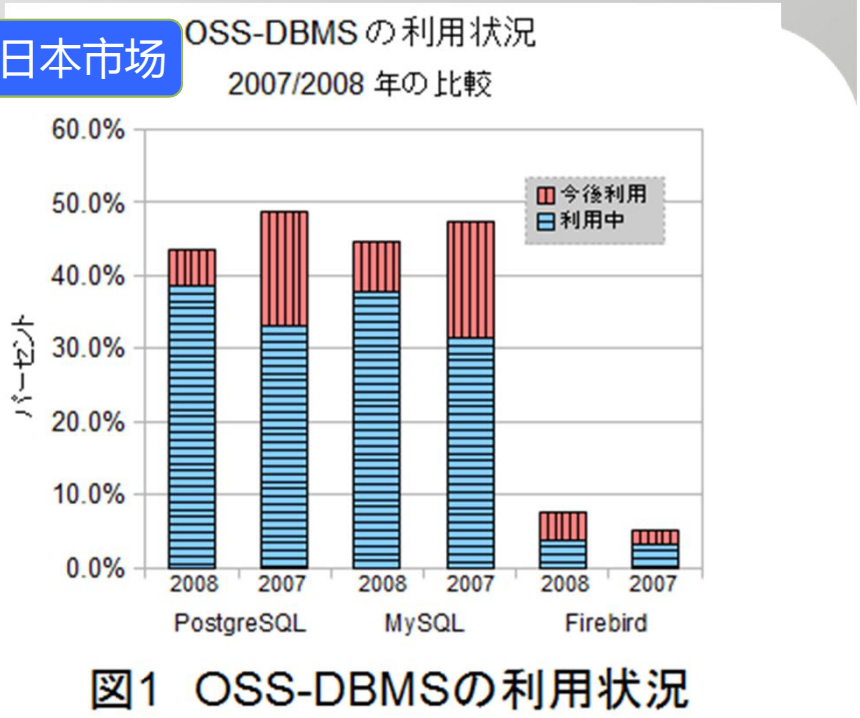


図1 OSS-DBMSの利用状況

http://lets.postgresql.jp/documents/tutorial/UserSurvey/Postgresql_Usage_Report_1/

- **知名用户**

  国际：Skype，instgram,NEC ，sony ，松下，NTT ...

  国内:去哪儿，华为 ...

  定制:Enterprise DB,Fujitsu,一些国产数据库...

- **概括对比**

  Oracle：最强大和最贵的RDB

  MySQL：使用最广泛的开源RDB

  PostgreSQL:最强大的开源RDB

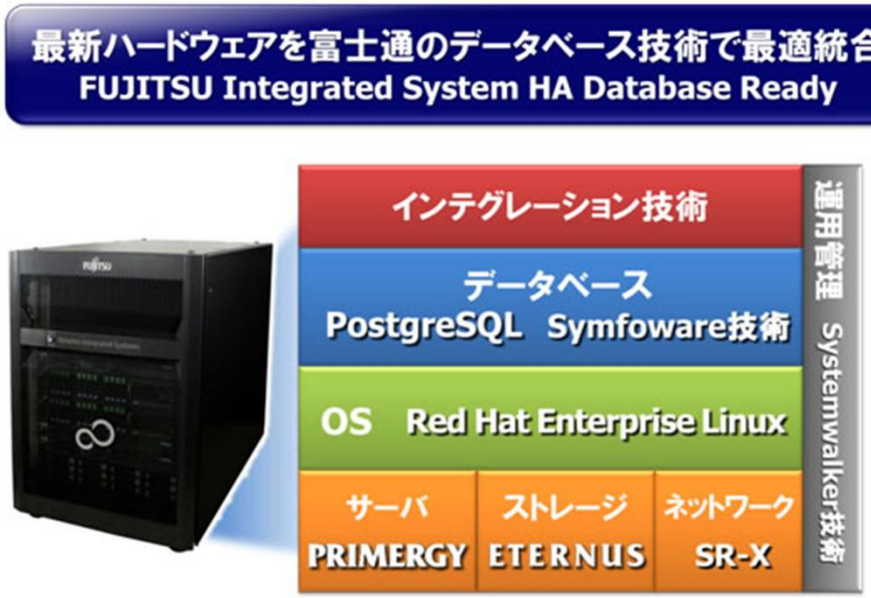
富士通的PostgreSQL关联产品

FUJITSU

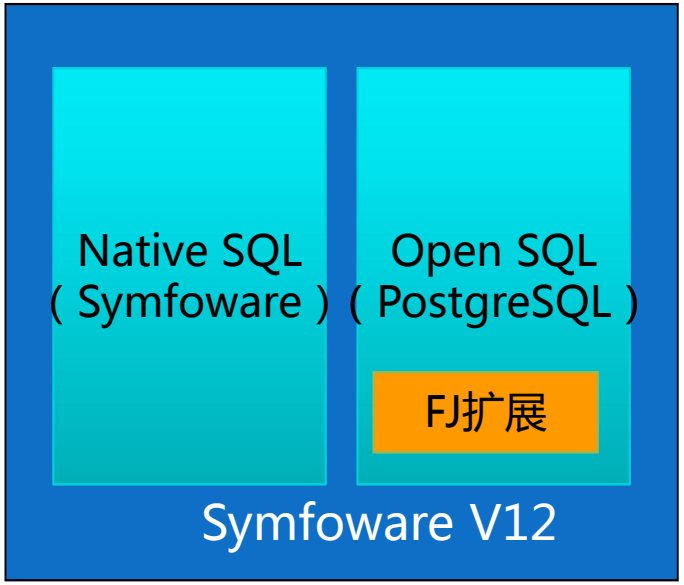
基于PG的早期产品

PowerGres Plus

HA Database Ready（2013/2发布）



Symfoware V12（2013/9发布）



目录

FUJITSU

- PostgreSQL简介
- 什么是事务
- 如何实现事务
- WAL
- MVCC
- Lock
- 各DB并发表现对比
- QA

什么是事务（1/4）

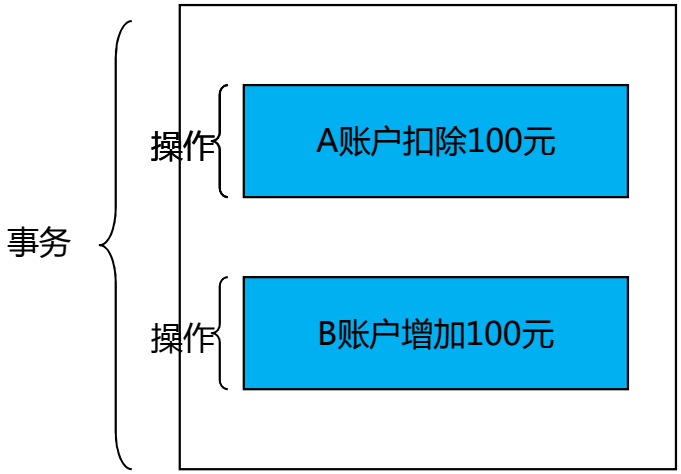
FUJITSU

概念

■事务是构成单一逻辑工作单元的操作集合，要么完整地执行，要么完全不执行。

举例:

从银行的A账户转100元到B账户

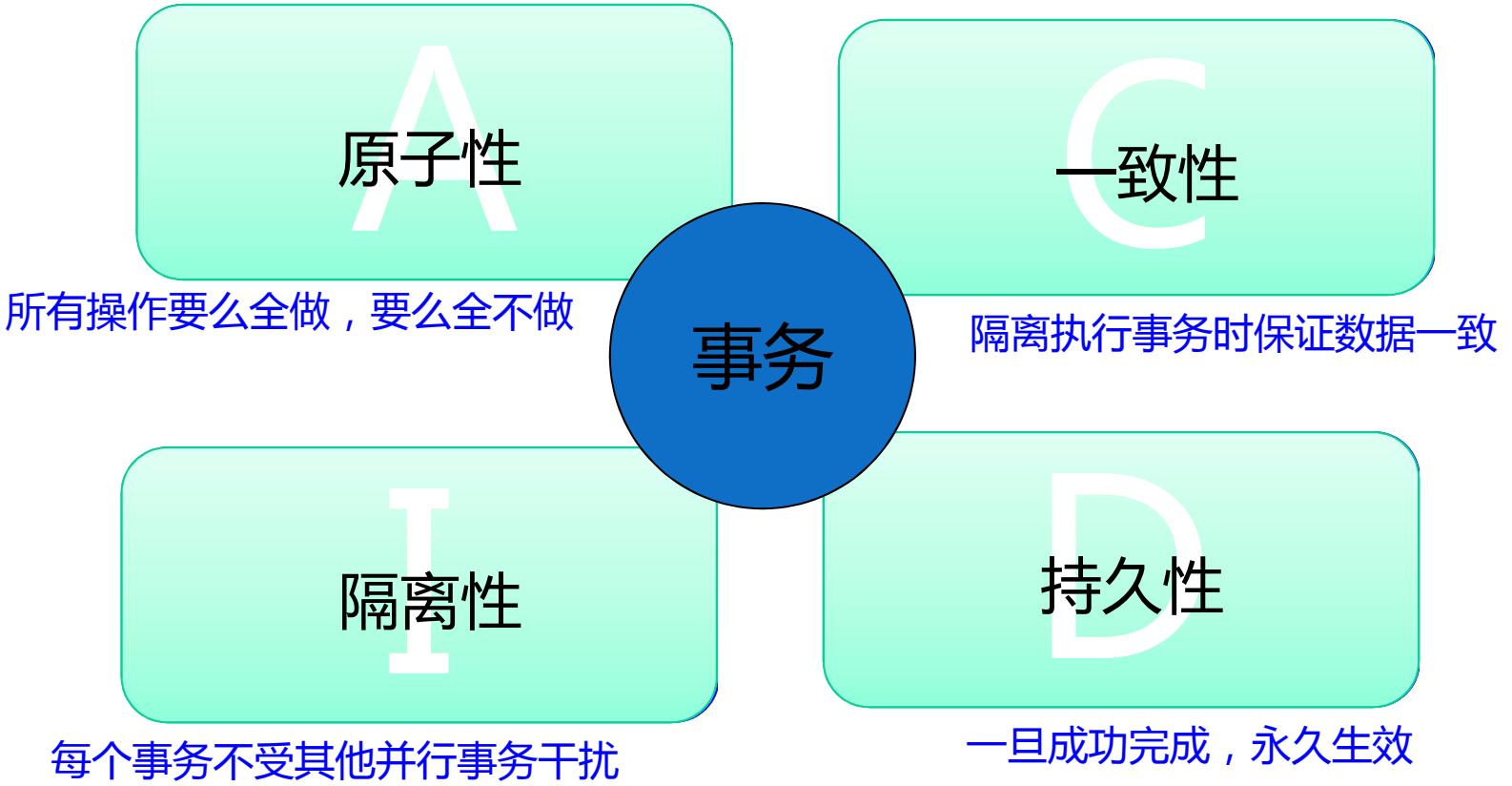


什么是事务（2/4）

FUJITSU

特征

■为保证数据完整性，DBMS必须维护事务的4个特性:原子性(Atomicity)，一致性(Consistency)，隔离性（Isolation）和持久性（Durability）。简称**ACID**。



什么是事务（3/4）

FUJITSU

ACID不是绝对的

隔离性

- 根据隔离的程度SQL规范定义了4个隔离级别

持久性

- 可选择的异步提交

数据完整性 VS 性能？

什么是事务（4/4）

FUJITSU

事务隔离级别

■SQL 标准用三个必须在并发的事务之间避免的现像定义了四个级别的事务隔离。

脏读

- 一个事务读取了另一个未提交事务写入的数据。

不可重复读

- 一个事务重新读取前面读取过的数据，发现该数据已经被另一个已提交事务修改。

幻读

- 一个事务重新执行一个查询，返回一套符合查询条件的行，发现这些行因为其它最近提交的事务而发生了改变。

SQL标准的事务隔离级别

隔离级别	脏读	不可重复读	幻读
读未提交	可能	可能	可能
读已提交	不可能	可能	可能
可重复读	不可能	不可能	可能
可串行化	不可能	不可能	不可能

PG中的事务隔离级别

隔离级别	脏读	不可重复读	幻读
读已提交	不可能	可能	可能
可重复读	不可能	不可能	不可能（*1）
可串行化	不可能	不可能	不可能

*1） PG中可重复读已满足SQL标准的可串行化

*)PG中读未提交被映射为读已提交

目录

FUJITSU

- PostgreSQL简介
- 什么是事务
- **如何实现事务**
- WAL
- MVCC
- Lock
- 各DB并发表现对比
- QA

如何实现事务（1/2）

FUJITSU

事务实现技术

原子性

- UNDO LOG

一致性

- 主键，非空等约束

隔离性

- 封锁
- MVCC

持久性

- UNDO LOG + REDO LOG

如何实现事务（2/2）

FUJITSU

几种常见数据库的ACID实现方法：

	原子性	一致性	隔离性	持久性
Symfoware(V11**以前**)	BI（UNDO LOG）	制约(主键，非空…)	封锁 PRECEDENCE(*1)	BI(UNDO LOG) AI(REDO LOG)
Oracle	UNDO LOG	同上	MVCC(*2)	UNDO LOG REDO LOG
PostgreSQL	MVCC	同上	MVCC(*3)	WAL(REDO LOG)
SQL Server	transaction log(ldf)	-	封锁 MVCC(*4)	transaction log(ldf)

*1）旧数据和新数据放在一起，最多只有1份旧数据
*2）旧数据放在单独的回滚段
*3）旧数据和新数据放在一起，可以有多份旧数据
*4）打开READ_COMMITTED_SNAPSHOT开关或使用SNAPSHOT隔离级别时启用MVCC

目录

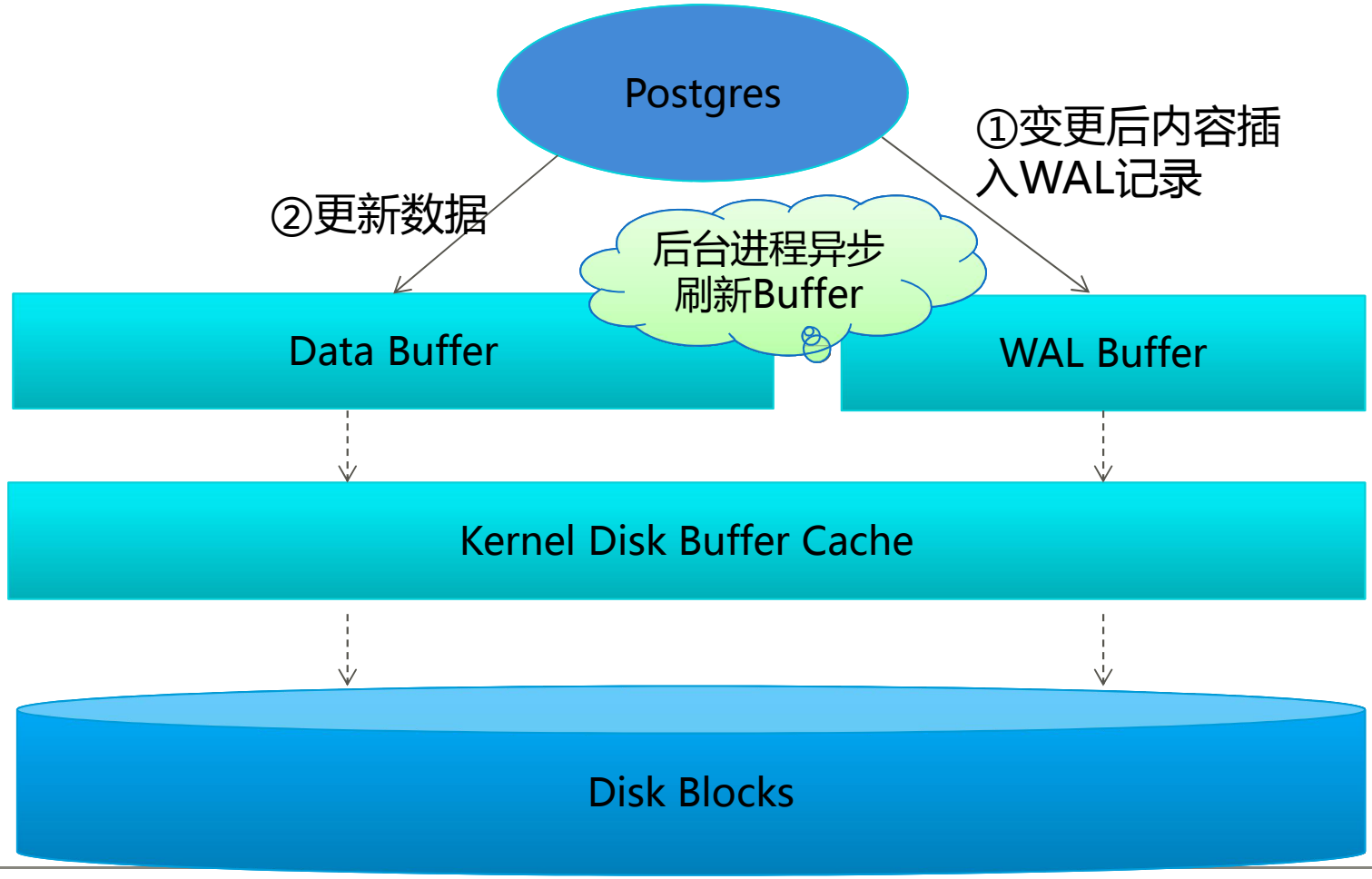
FUJITSU

- PostgreSQL简介
- 什么是事务
- 如何实现事务
- **WAL**
- MVCC
- MVCC VS Lock
- 各DB并发表现对比
- QA

WAL（Write-Ahead Log）

FUJITSU

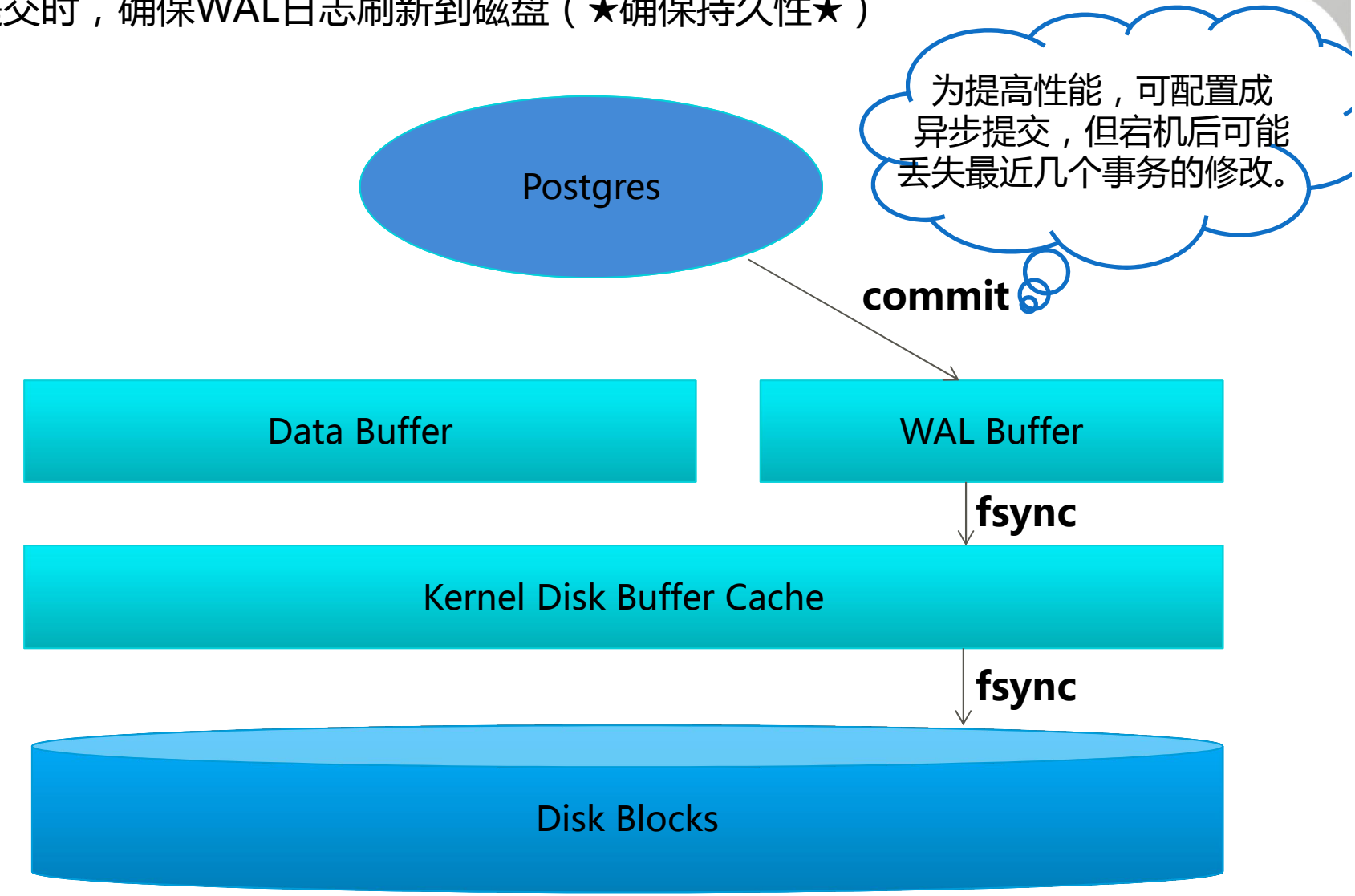
- 更新数据页前先将更新内容记入日志
- 异步刷新数据Buffer的脏页和WAL Buffer到磁盘
- Buffer管理器确保绝不会先于对应的WAL记录刷新脏数据到磁盘



WAL Buffer的同步刷新

FUJITSU

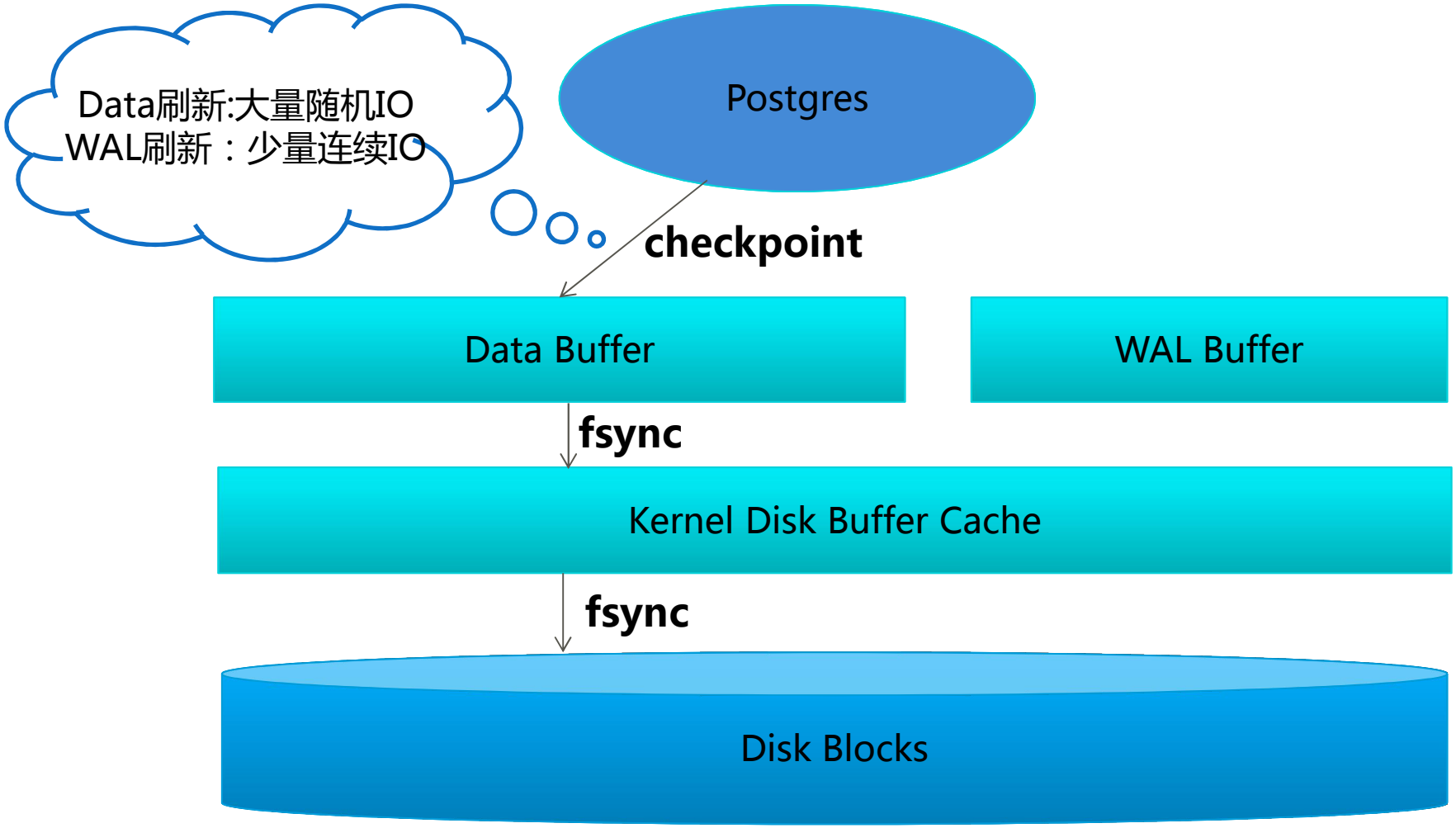
- 事务提交时，确保WAL日志刷新到磁盘（★确保持久性★）



Data Buffer的同步刷新

FUJITSU

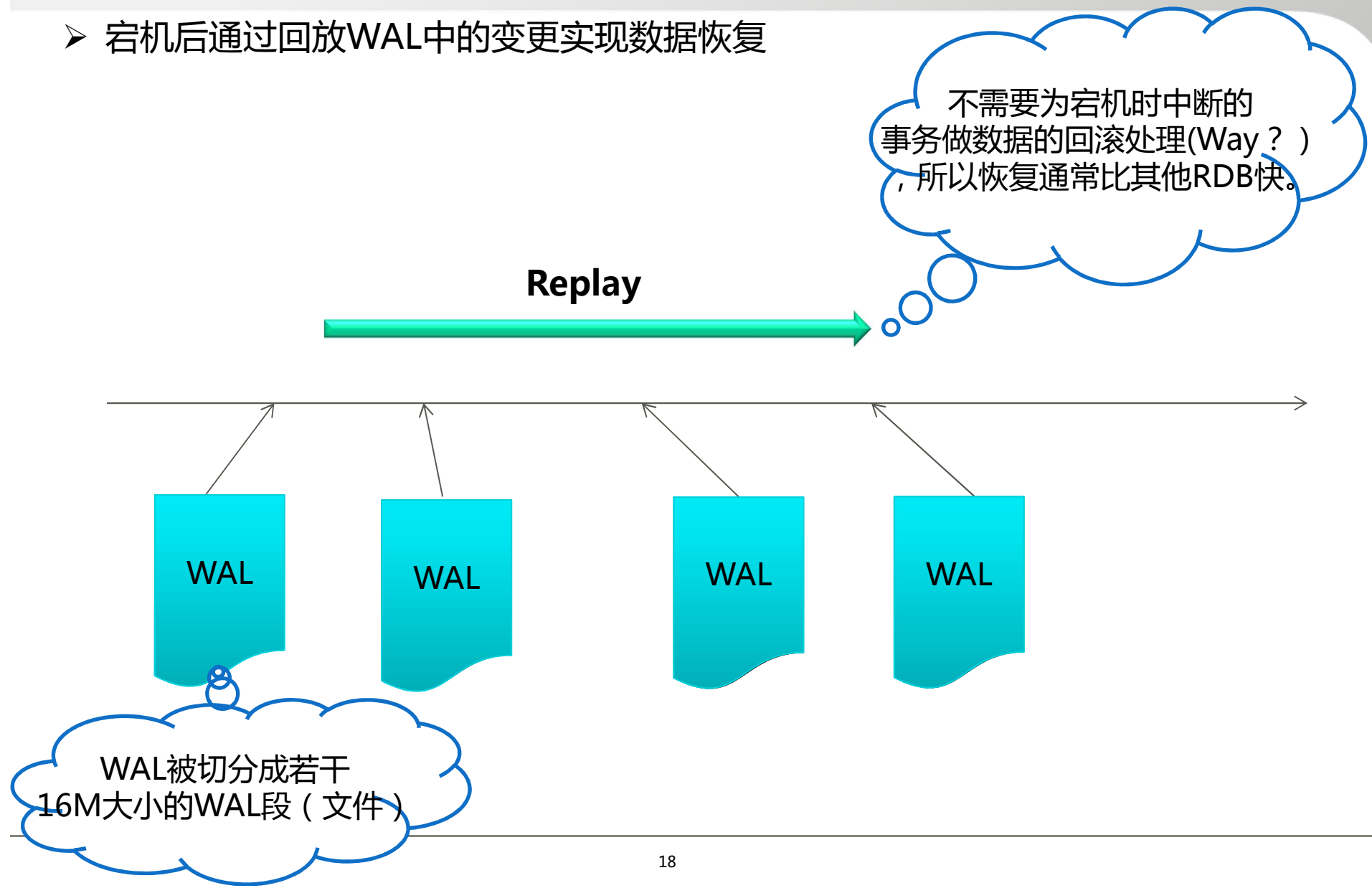
- Checkpoint发生时，确保Data Buffer的所有脏页刷新到磁盘



宕机恢复

FUJITSU

- 宕机后通过回放WAL中的变更实现数据恢复

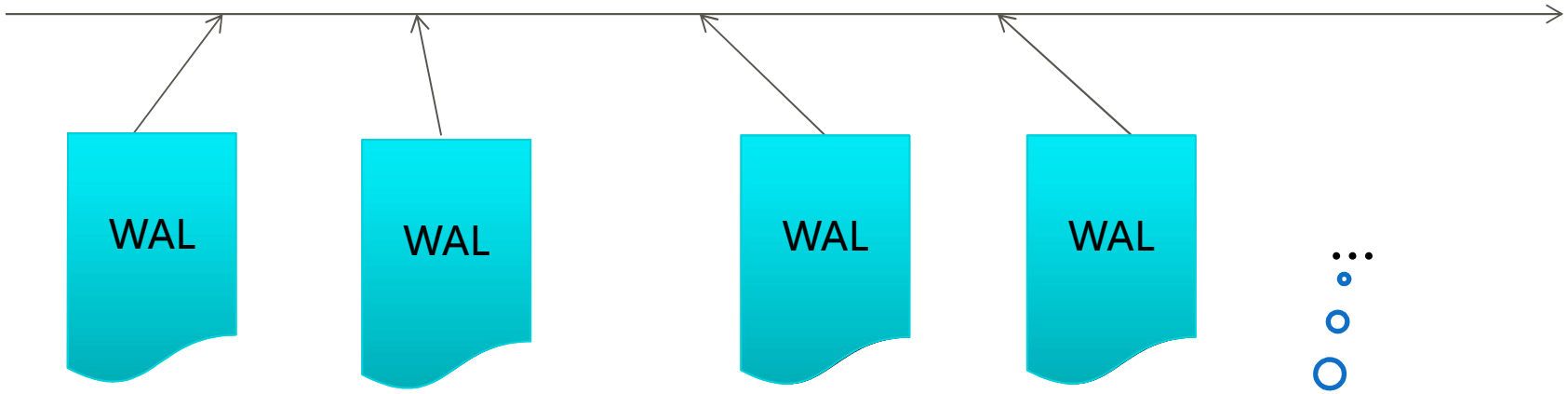


WAL的回收

FUJITSU

从哪里开始回放WAL?

Replay

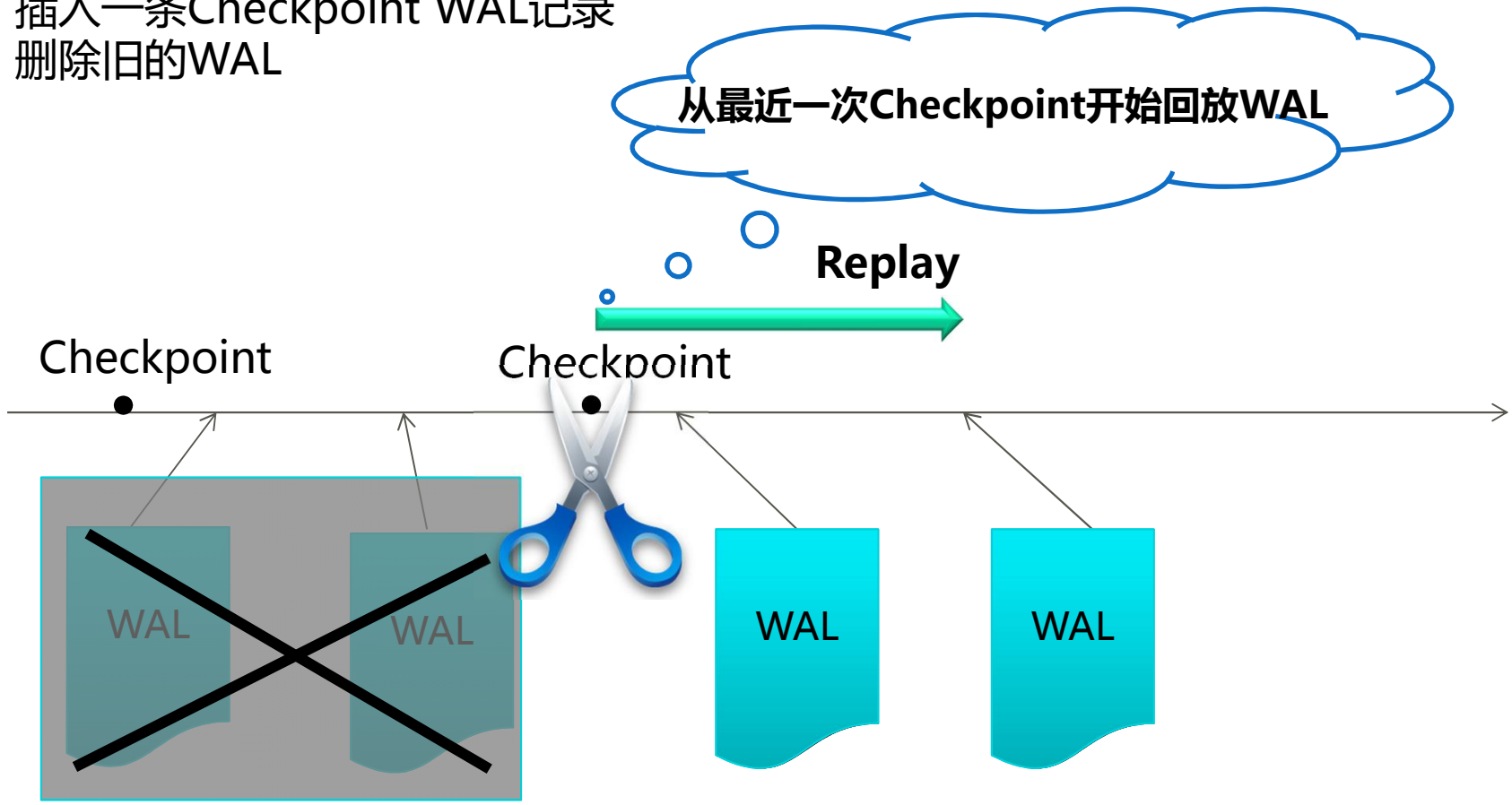


WAL文件会无限增长？

Checkpoint

FUJITSU

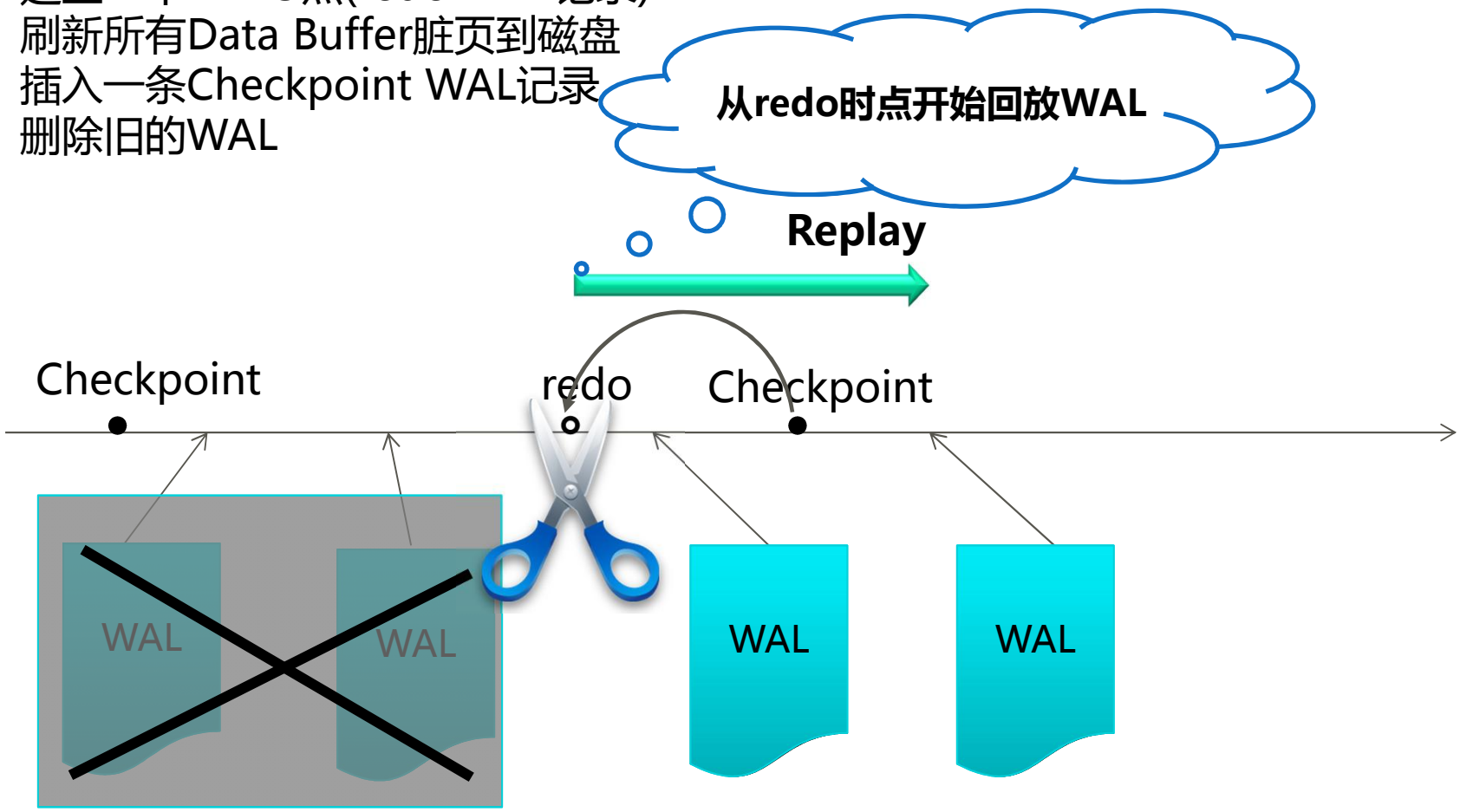
- 刷新所有Data Buffer脏页到磁盘
- 插入一条Checkpoint WAL记录
- 删除旧的WAL



在线Checkpoint

FUJITSU

- 建立一个REDO点(redo WAL记录)
- 刷新所有Data Buffer脏页到磁盘
- 插入一条Checkpoint WAL记录
- 删除旧的WAL

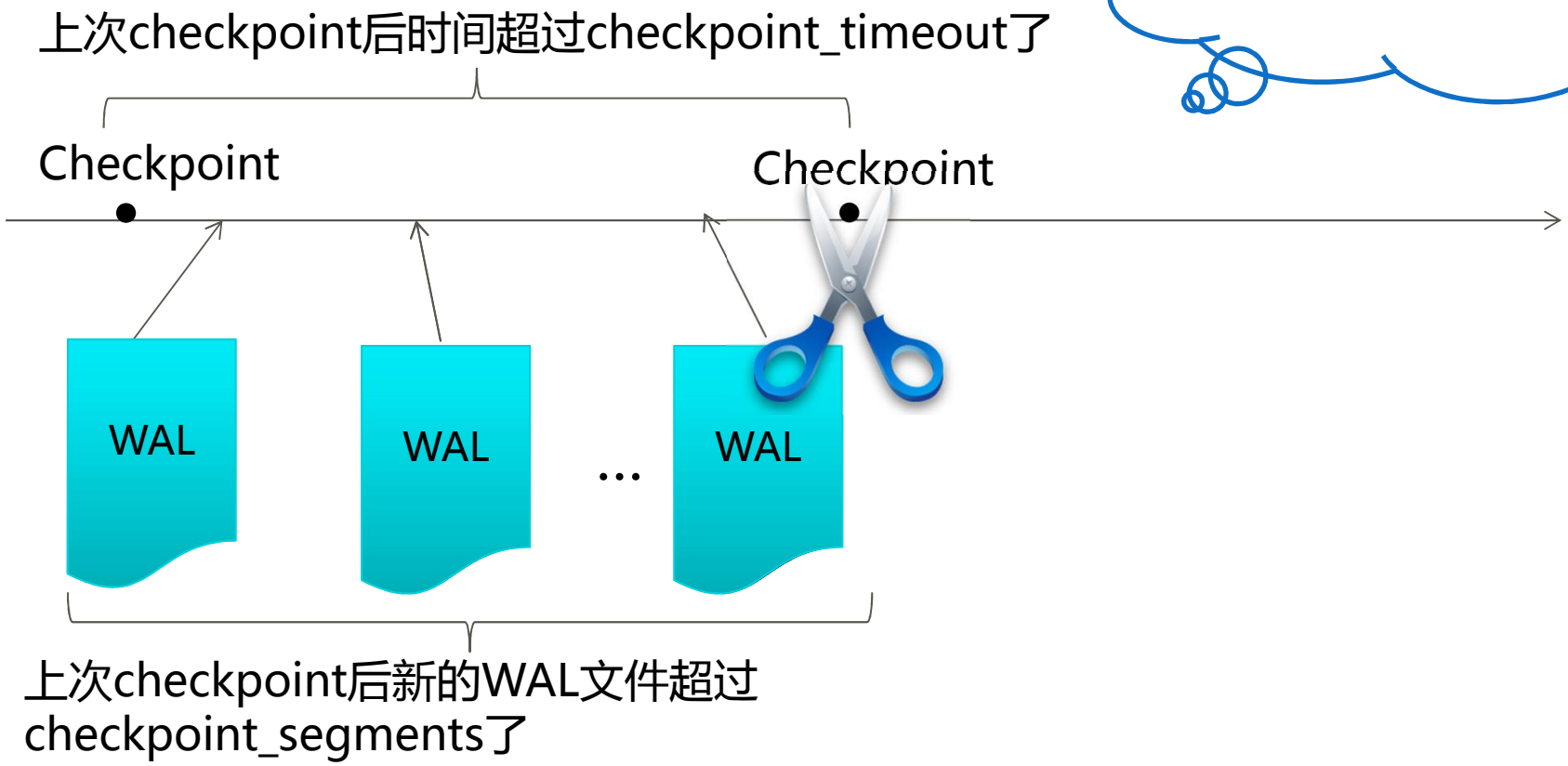


何时生成Checkpoint

FUJITSU

- 后台自动执行(超过checkpoint_segments或checkpoint_timeout时)
- 手动执行checkpoint语句

Checkpoint如果太频繁,影响
太稀疏，WAL占用空间大，恢复



目录

FUJITSU

- PostgreSQL简介
- 什么是事务
- 如何实现事务
- WAL
- **MVCC**
- Lock
- 各DB并发表现对比
- QA

MVCC（Multiversion Concurrency Control）

FUJITSU

基本原理

每个事务看到的都只是一个特定时点的数据快照,避免其它并发更新事务的影响。

特点

- 读写互不阻塞
- 更新相同数据的事务互相阻塞

优点:

- 提高并发度
- 减少响应时间

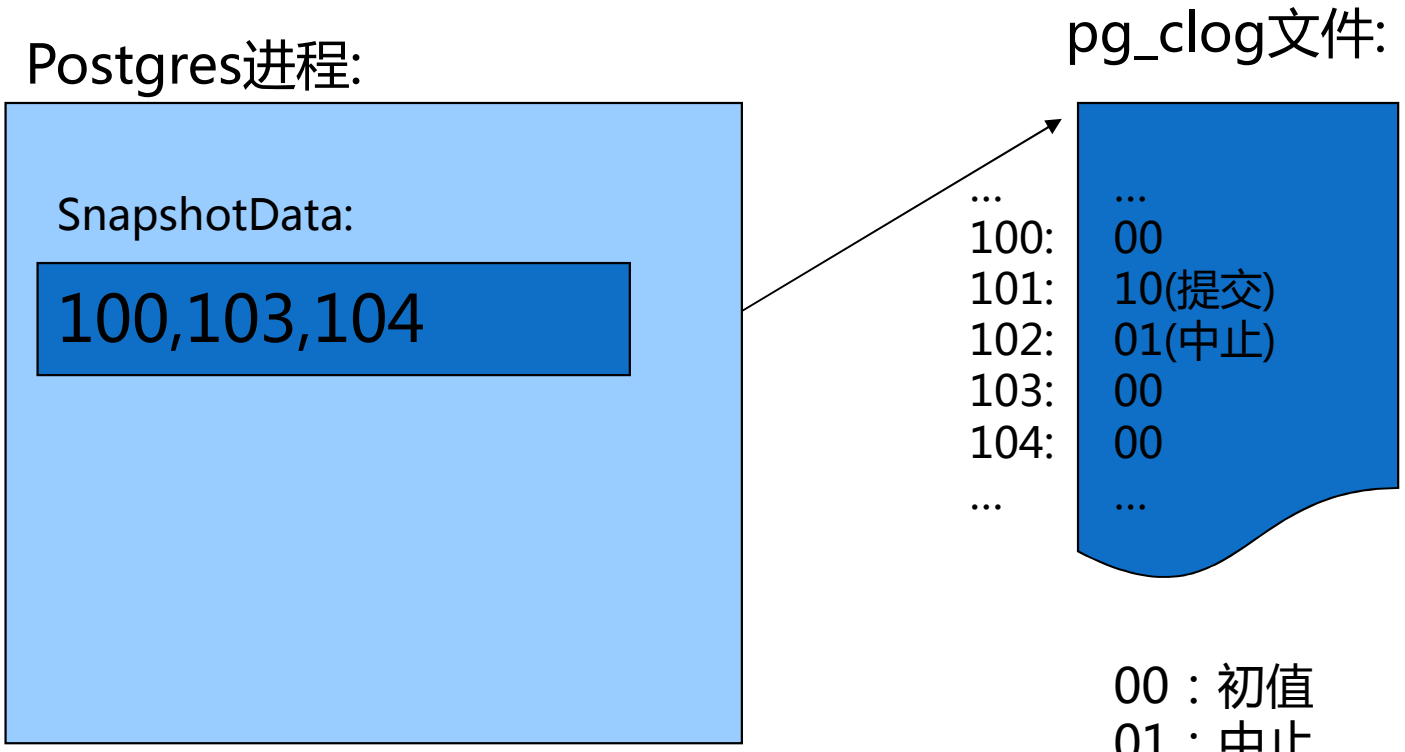
缺点:

- 存储的额外负担(可能保存数据的多个旧版本)
- 容易由并发冲突导致事务回滚

事务ID

FUJITSU

- 每个事务执行第一个SQL时系统分配唯一的事务ID（ 又称为Xid ）
- 事务ID按逻辑时间戳分配，通过事务ID值可知道事务执行的先后顺序
- 系统记录所有活动事务一览(又叫: SnapshotData)
- 系统记录所有事务状态（记录到 pg_clog文件）



00：初值
01：中止
10：提交

Tuple(行版本)

FUJITSU

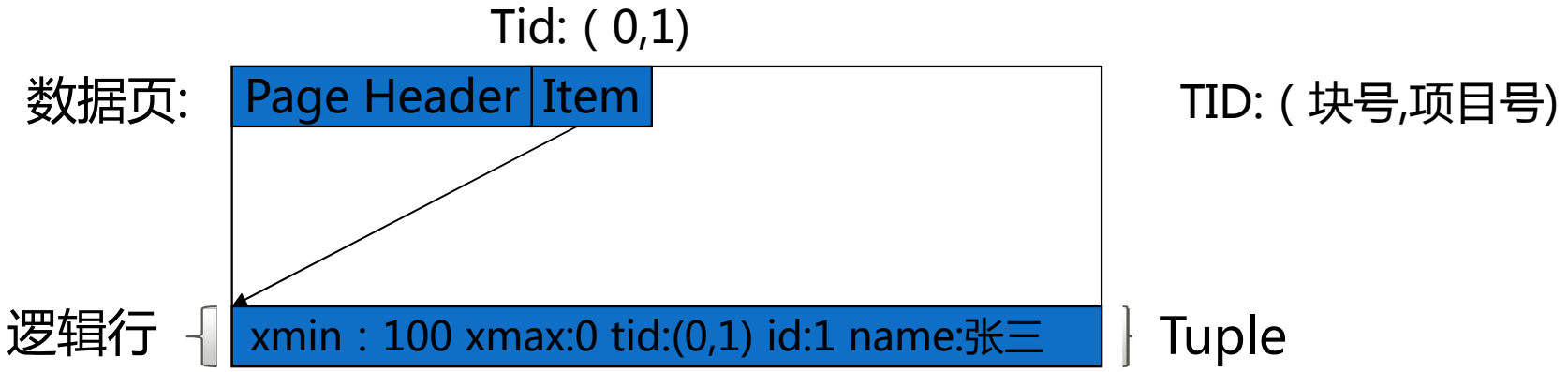
- Tuple代表一个数据行的一个特定版本
- 每个Tuple中包含以下头信息
  Xmin: 创建该元组的事务ID
  Xmax: 更改或删除该元组的事务ID
  Tid: 到相同逻辑行的最新版本的链接

Tuple的例子

FUJITSU

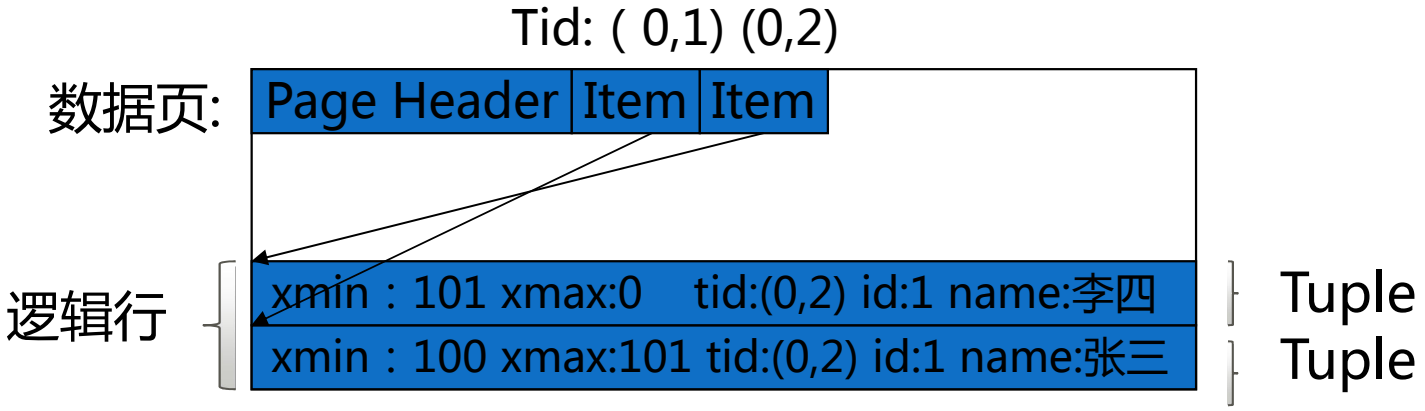
假设事务ID为100

Insert into TB1(id,name) values (1, '张三')



假设事务ID为101

Update TB1 Set name ='李四' where ID = 1



事务可以看到哪些Tuple?

FUJITSU

Tuple对事务T1可见的条件

T1的隔离级别为可重复读或可串行化时:

- 创建该元组的事务（xmin）在**T1开始**前已提交
- 更新（如果有）该元组的事务(xmax)在**T1开始**时不处于已提交状态

T1的隔离级别为读已提交时:

- 创建该元组的事务（xmin）在**T1的查询执行**前已提交
- 更新（如果有）该元组的事务(xmax)在**T1的查询执行**时不处于已提交状态

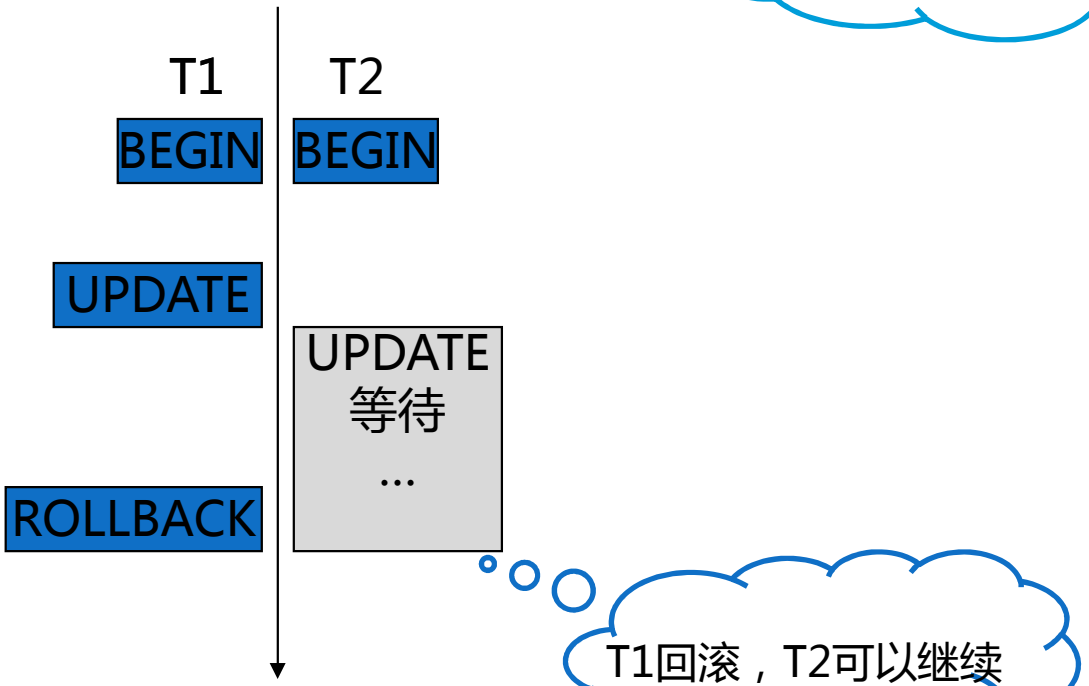
某一时刻t，事务已提交的判断条件

1）pg_clog中该事物状态为已提交。且
2）该事务ID小于t时刻的事务计数。且
3）t时刻采取的SnapshotData中不包含该事务ID

如果2个事务同时更新或删除同一逻辑行？

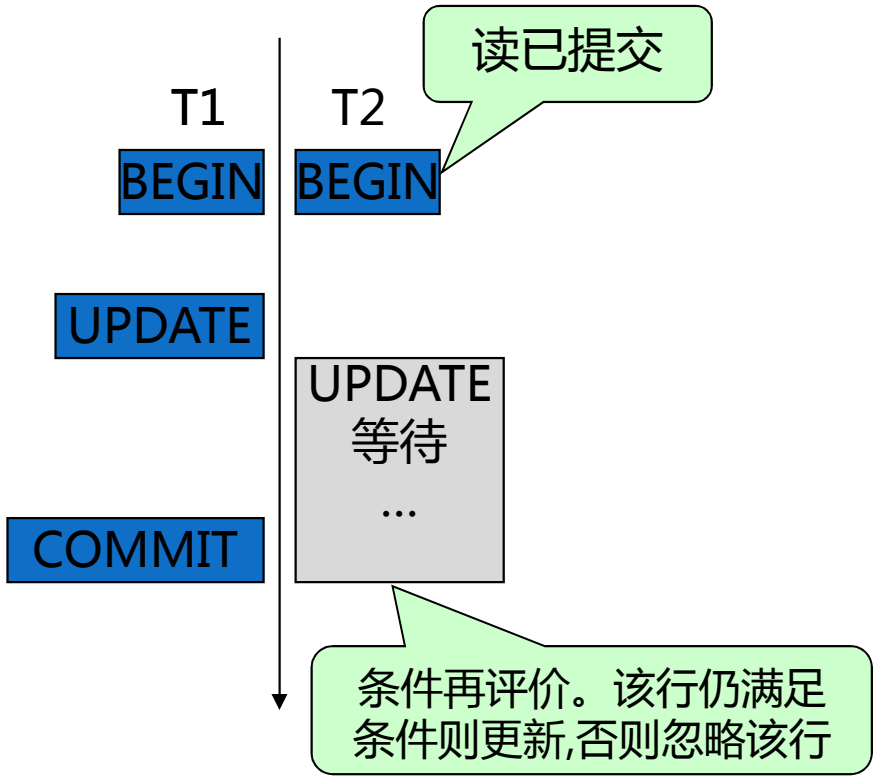
FUJITSU

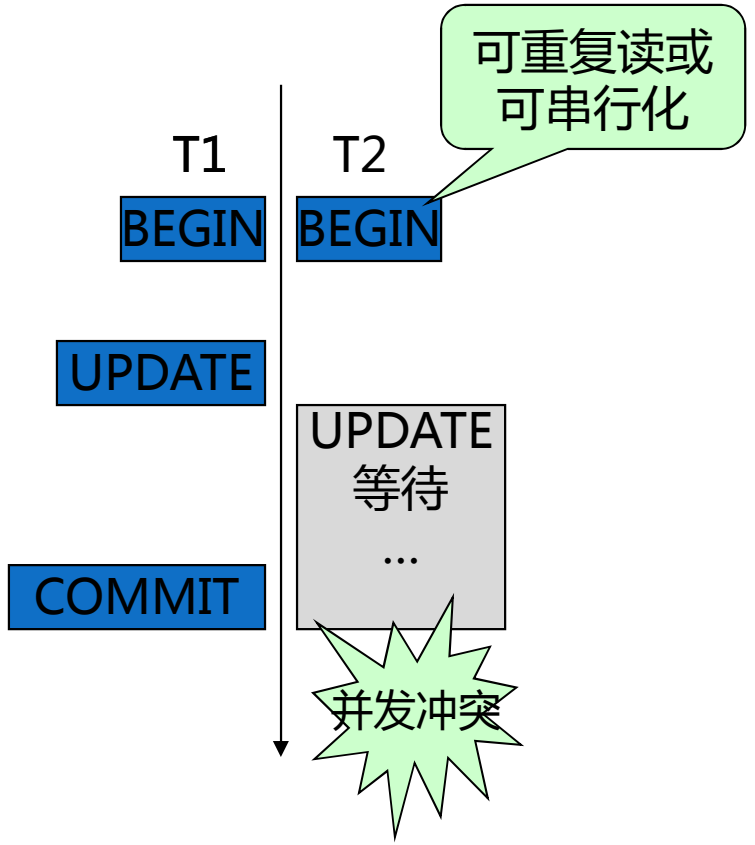
对不同行的并发更新
不会导致事务阻塞



T1回滚，T2可以继续

等待的事务T1提交的场景

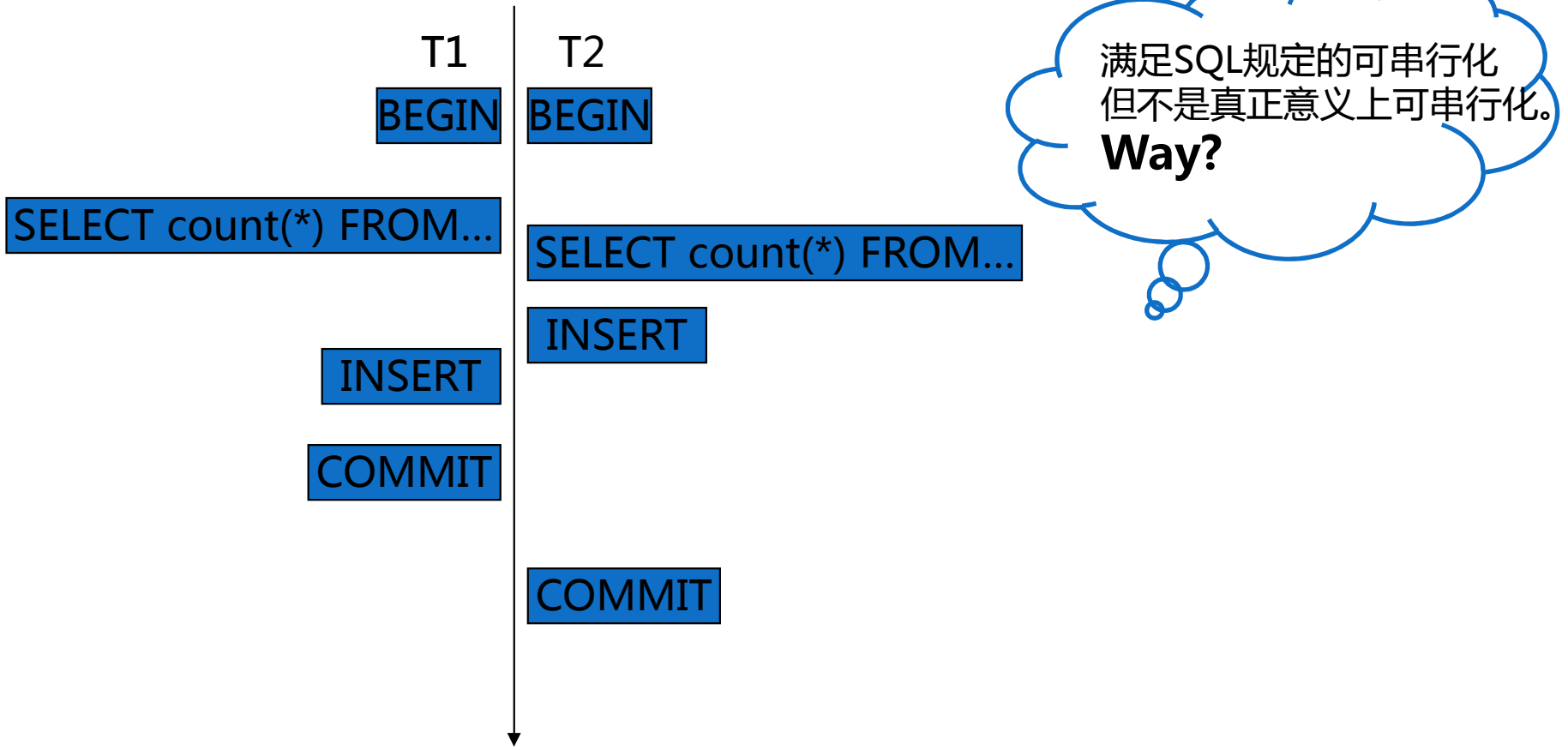




ERROR: could not serialize access due to concurrent update

满足SQL规约的可串行化=真正的可串行化？

FUJITSU

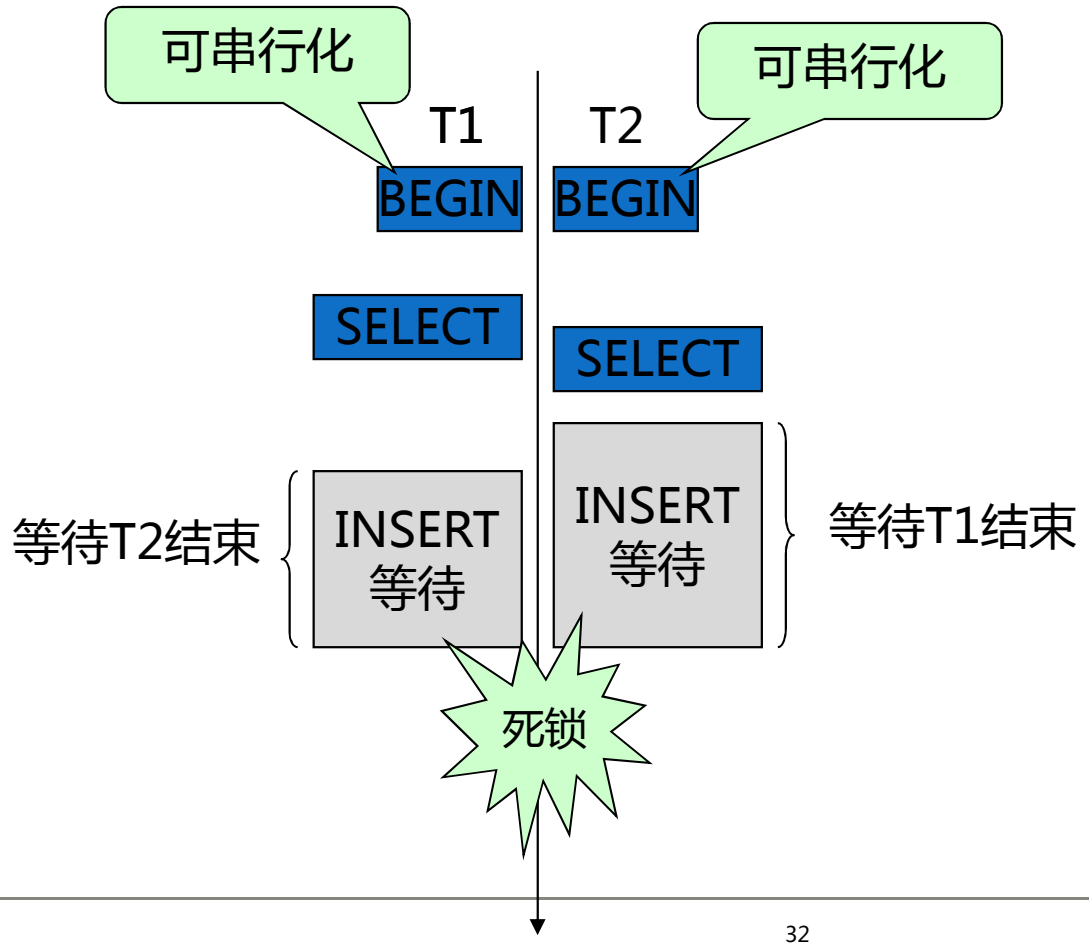


传统的串行化实现方法

FUJITSU

Strict Two-Phase Locking(S2PL):

- 更新操作将阻塞直到已读取过同一数据的并发事务结束
- 更新和读取操作都将阻塞直到已更新过同一数据的并发事务结束



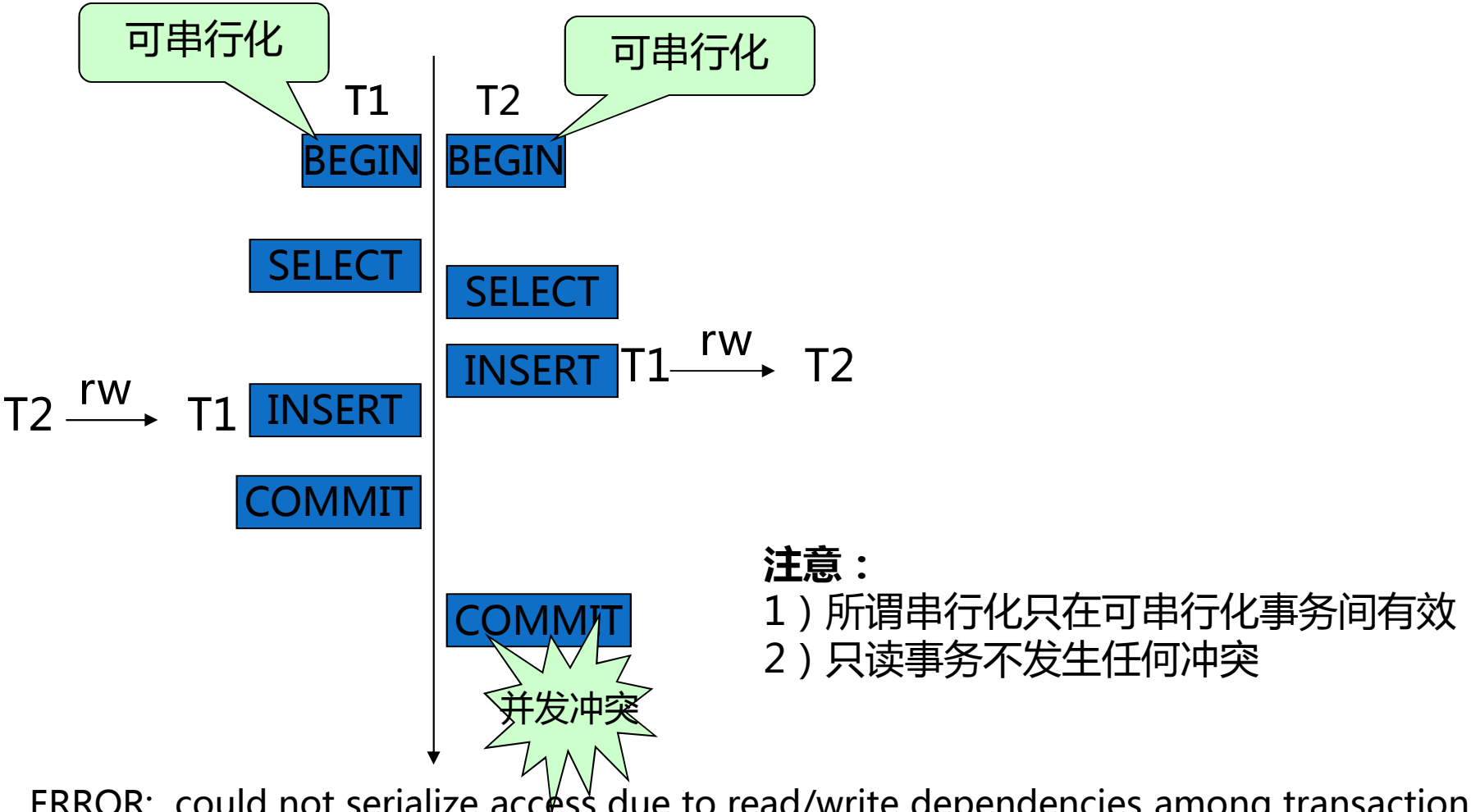
问题:
容易发生阻塞和死锁

PostgreSQL的串行化实现方法

FUJITSU

Serializable Snapshot Isolation (SSI)

- 跟踪事务对数据的读写操作形成RW依赖
- 提交顺序和RW依赖关系不一致时回滚事务



注意：

1）所谓串行化只在可串行化事务间有效

2）只读事务不发生任何冲突

ERROR: could not serialize access due to read/write dependencies among transactions

目录

FUJITSU

- PostgreSQL简介
- 什么是事务
- 如何实现事务
- WAL
- MVCC
- **Lock**
- 各DB并发表现对比
- QA

表级锁（1/2）

FUJITSU

- 如果不想应用重做事务，或进行系统维护，可以**手动**加表级锁

LOCK [TABLE] [ONLY] *name* [*] [, ...] [IN *lockmode* MODE] [NOWAIT]

Requested Lock Mode	Current Lock Mode							
	ACCESS SHARE	ROW SHARE	ROW EXCLUSIVE	SHARE UPDATE EXCLUSIVE	SHARE	SHARE ROW EXCLUSIVE	EXCLUSIVE	ACCESS EXCLUSIVE
ACCESS SHARE								X
ROW SHARE							X	X
ROW EXCLUSIVE					X	X	X	X
SHARE UPDATE EXCLUSIVE				X	X	X	X	X
SHARE			X	X		X	X	X
SHARE ROW EXCLUSIVE			X	X	X	X	X	X
EXCLUSIVE		X	X	X	X	X	X	X
ACCESS EXCLUSIVE	X	X	X	X	X	X	X	X

表级锁（2/2）

FUJITSU

- 有些Command会**自动**获取表级锁

Requested Lock Mode	**自动获取表级锁的Command**
ACCESS SHARE	SELECT
ROW SHARE	SELECT FOR UPDATE and SELECT FOR SHARE
ROW EXCLUSIVE	UPDATE, DELETE, and INSERT
SHARE UPDATE EXCLUSIVE	VACUUM (without FULL), ANALYZE, CREATE INDEX CONCURRENTLY, and some forms of ALTER TABLE
SHARE	CREATE INDEX
SHARE ROW EXCLUSIVE	
EXCLUSIVE	
ACCESS EXCLUSIVE	ALTER TABLE, DROP TABLE, TRUNCATE, REINDEX, CLUSTER, and VACUUM FULL

行级锁

FUJITSU

SELECT ... FOR UPDATE

- 互相冲突

SELECT ... FOR SHARE

- 互相不冲突，但阻塞对锁定行的更新或删除

锁的危害

FUJITSU

并发性能下降

- 锁对象的粒度
- 锁阻塞的操作
- 锁持有的时间

死锁

- 事务的互相等待形成死锁
- DBMS提供死锁检测机制(pg默认1秒超时)，通过回滚其中一个（或多个）事务打破死锁
- 应用设计时应考虑尽量避免死锁

目录

FUJITSU

- PostgreSQL简介
- 什么是事务
- 如何实现事务
- WAL
- MVCC
- Lock
- **各DB并发表现对比**
- QA

各DB并发表现对比

FUJITSU

■比较对象
PostgreSQL 9.2.4
Oracle 11g
SQL Server 2008（ * 1 ）
Symfoware 10.0.0（ * 2 ）

* 1 ）READ_COMMITTED_SNAPSHOT=O F F
* 2 ）未使用PRECEDENCE

■表定义
create table tb1(id int primary key,name varchar(30)

■初始数据
insert into tb1 values(1,'a');
insert into tb1 values(2,'a');

各DB并发表现对比

FUJITSU

2个并发事务更新同一行

	读未提交	读已提交	可重复读	可串行化
PostgreSQL	-	等待(*)	等待 (**)	等待(**)
Oracle	-	等待(*)	-	等待(**)
SQL Server	等待(*)	等待(*)	等待(*)	等待(*)
Symfoware	等待(*)	等待(*)	等待(*)	等待(*)

等待(*):如果先行的SQL提交，则基于更新后的数据，否则基于原来的数据
等待(**)：如果先行的SQL提交，则报错

测试方法

事务1：

```
update tb1 set name =    'b' where id = 1
…
commit
```

事务2：

```
update tb1 set name = 'c' where id = 1
```

各DB并发表现对比

FUJITSU

一个事务先查询另一个事务更新(或：一个事务先更新另一个事务查询)

	读未提交	读已提交	可重复读	可串行化
PostgreSQL	-	不影响	不影响	不影响
Oracle	-	不影响	-	不影响
SQL Server	不影响/脏读	不影响/等待(*)	等待/等待(*)	等待/等待(*)
Symfoware	不影响/脏读	不影响/等待(*)	等待/等待(*)	等待/等待(*)

等待(*):如果先行的SQL提交，则基于更新后的数据，否则基于原来的数据

并发性能不好

测试方法

事务1：

```
select * from tb1
…
commit
```

事务2：

```
update tb1 set name = 'c' where id = 1
```

事务1：

```
update tb1 set name = 'c' where id = 1
…
commit
```

事务2：

```
select * from tb1
```

各DB并发表现对比

FUJITSU

2个并发事务分别先执行全表查询后插入一条新行

	读未提交	读已提交	可重复读	可串行化
PostgreSQL	-	不影响	不影响	报错（*）
Oracle	-	不影响	-	正常终了（*）
SQL Server	不影响	不影响	不影响	死锁(*)
Symfoware	不影响	不影响	不影响	死锁(*)

报错(*):后提交的事务在提交时报错
正常终了（*）：这个模式违反真正的可串行化，不应该正常终了
死锁(*):事务1插入时，等待事务2结束，事务2插入时发生死锁报错

Oracle不支持真正的可串行化

测试方法

事务1：

```
select * from tb1

insert into tb1 values(5,'c')


commit
```

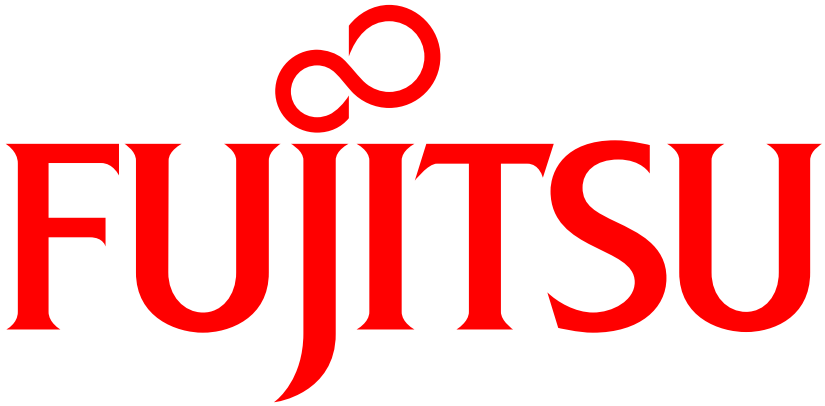
事务2：

```
select * from tb1

insert into tb1 values(6,'c')


commit
```

FUJITSU

Q&A

Thanks



shaping tomorrow with you