DTCC

2017第八届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2017

数据库内核专场

2017年5月11日

时间	主持人	李海翔	腾讯金融云数据库技术专家
13:30-14:10	数据库的并发控制技术深度探索	李海翔	腾讯金融云数据库技术专家
14:10-14:50	Tencent MySQL内核优化解析	张青林	腾讯高级工程师
14:50-15:30	深入理解MySQL Group Replication	宋利兵	MySQL研发工程师
15:30-15:50	茶歇、展示		
15:50-16:30	数据库存储虚拟化及内核架构优化		浪潮数据库支持经理
16:30-17:10	MySQL复制演进：基于InnoDB的复制新框架	翟卫祥	阿里巴巴数据库研发工程师
17:10-17:50	宽表列存储在大数据分析中的应用与优化	卞昊穹	中国人民大学博士研究生

数据库的并发控制技术深度探索

李海翔 @那海蓝蓝
@腾讯 金融数据库 TDSQL

PostgreSQL, MySQL, Greenplum, Informix, etc

@那海蓝蓝 Blog: http://blog.163.com/li_hx/

《数据库查询优化器的艺术: 原理解析与SQL性能优化》

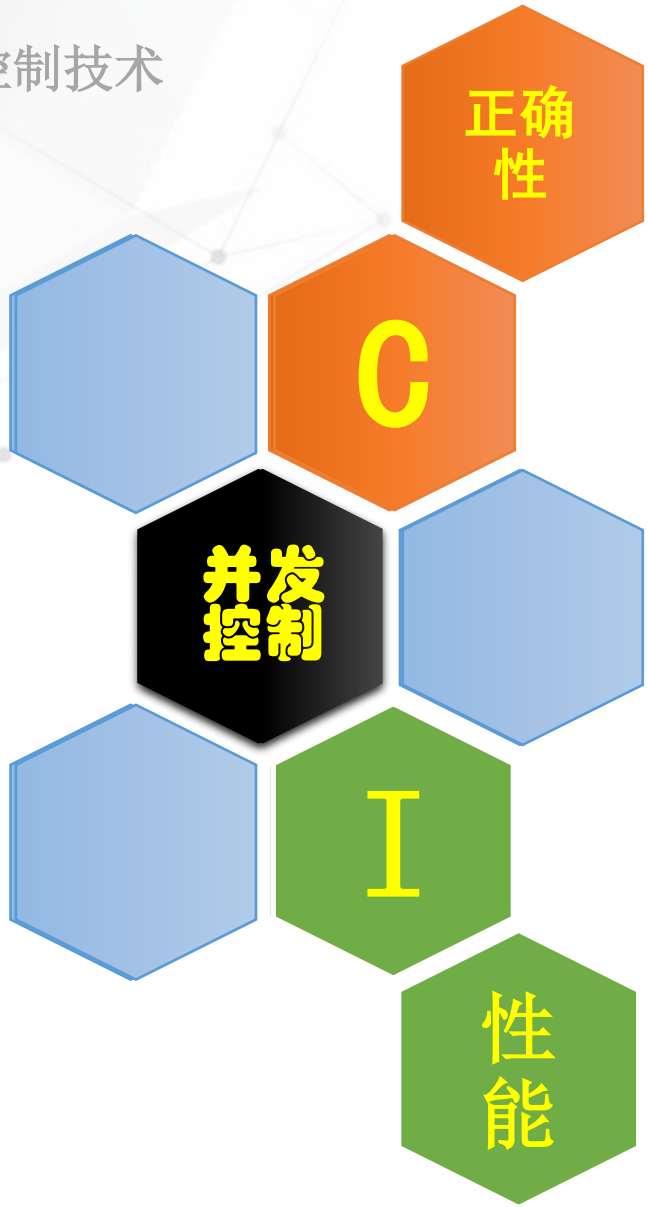
《数据库事务处理的艺术: 事务管理与并发控制》

数据库

数据库的事务处理技术

数据库的并发访问控制技术

今天分享什么?



数据库

数据库的事务处理技术

数据库的并发访问控制技术

今天分享什么?

PostgreSQL MySQL ORACLE 甲骨文 Informix SOFTWARE

今天分享什么?

- Q1 数据异常现象有哪些?
- Q2 为什么会产生数据异常现象?
- Q3 并发控制技术有哪些?
- Q4 为什么并发控制技术能解决数据异常现象?
- Q5 Oracle等主流数据库的并发控制使用了哪些技术?
- Q6 Pg和MySQL的并发控制技术有什么差别吗?

Q1 数据异常现象有哪些？--常见的读异常

	脏读		不可重复读		幻象	
	P1 ("Dirty read")		P2 ("Non-repeatable read")		P3 ("Phantom")	
时间	T1主事务	T2	T1主事务	T2	T1主事务	T2
t0		W(row)-Update	R(row)		R(rows)-WHERE<condiotn>	
t1	R(row)			W(row)-Update/Delete		W(rows)-Insert/Update =><conditon>
t2		Abort		Commit	R(rows)-WHERE<condiotn>	
t3			R(row)			

Q1 -1 除了读异常，还有其他异常现象吗？

Q1 数据异常现象有哪些？--常见的写异常

	脏写		丢失更新	
	Dirty write		Lost Update	
时间	T1	T2	T1	T2
t0	W(row)-Update		R(row)	
t1		W(row)-Update		W(row)-Update
t2		Commit	W(row)-Update	
t3	Abort		Commit	

Q1 -2 除了读写异常，还有其他异常现象吗？

Q1 数据异常现象有哪些？--写偏序异常

	两个事务写偏序		三个事务写偏序		
时间	T1	T2	T1	T2	T3
t0	x ← SELECTCOUNT(*) FROM doctors WHERE on-call = true			x ← SELECT current_batch	
t1		x ← SELECTCOUNT(*) FROM doctors WHERE on-call = true			INCREMENT current_batch
t2	**IF x ≥ 2 THEN** UPDATE doctors SET **on-call = false** WHERE name = Alice				Commit
t3		**IF x ≥ 2 THEN** UPDATE doctors SET **on-call = false** WHERE name = Bob	x ← SELECT current_batch		
t4	Commit		SELECT SUM(amount) FROM **receipts** WHERE batch = x-1		
t5		Commit	COMMIT		
t6				INSERT INTO**receipts** VALUES(x, somedata)	
t7				COMMIT	

语义：
至少有一个医生在提供电话咨询服务

Q1 数据异常现象有哪些？--其他异常

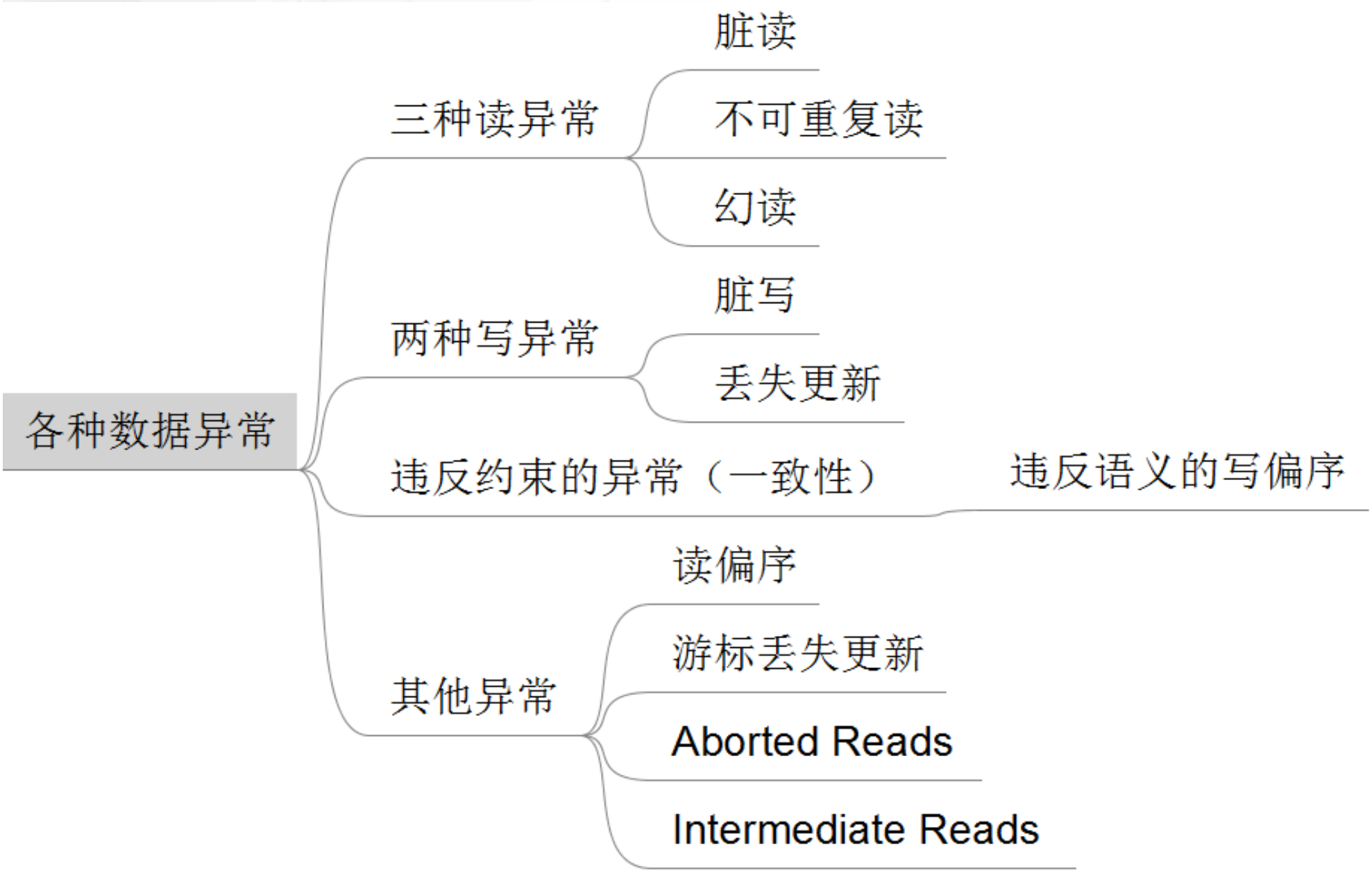
	读偏序	
	Read Skew	
时间	T1	T2
t0	R(x)	
t1		W(x)、W(y)
t2		Commit
t3	R(y)	

	游标丢失更新	
	Cursor Lost Update	
时间	T1	T2
t0	R(row)	
t1		W(row)-Update
t2	W(row)-Update	
t3	Commit	

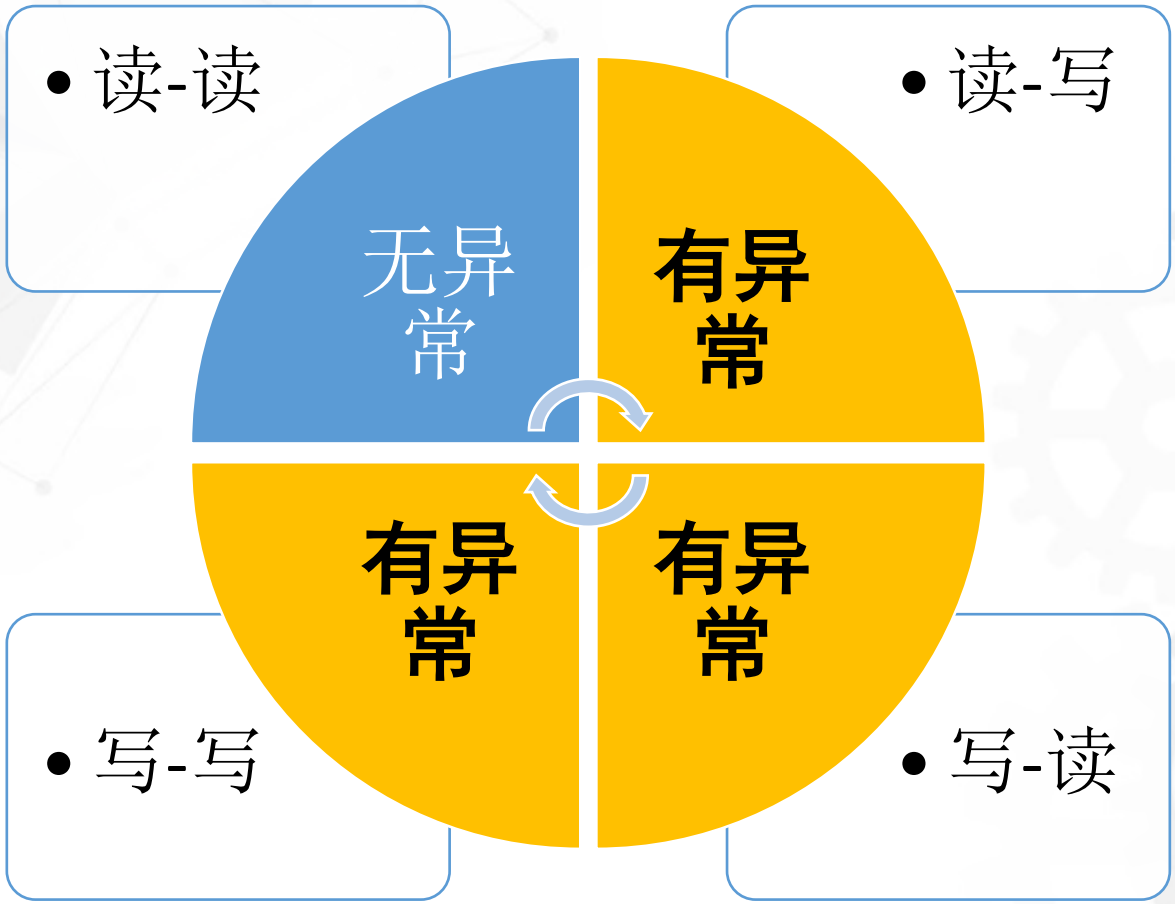
- *Aborted Reads*. A committed transaction T_2 reads some object (possibly via a predicate) modified by an aborted transaction T_1.
- *Intermediate Reads*. A committed transaction T_2 reads a version of an object x (possibly via a predicate) written by another transaction T_1 that was not T_1's final modification of x.

《High-Performance ACID via Modular Concurrency Control》

Q1 数据异常现象有哪些？--小结



Q2 为什么会产生数据异常现象？



并发操作可以被区分为四种：读-读、读-写、写-读、写-写

Q2 为什么会产生数据异常现象？

	脏读		不可重复读		幻象	
	P1 ("Dirty read")		P2 ("Non-repeatable read")		P3 ("Phantom")	
时间	T1 主事务	T2	T1 主事务	T2	T1 主事务	T2
t0		W(row)-Update	R(row)		R(rows)-WHERE<condition>	
t1	R(row)			W(row)-Update/Delete		W(rows)-Insert/Update =><conditon>
t2		Abort		Commit	R(rows)-WHERE<condition>	
t3			R(row)			

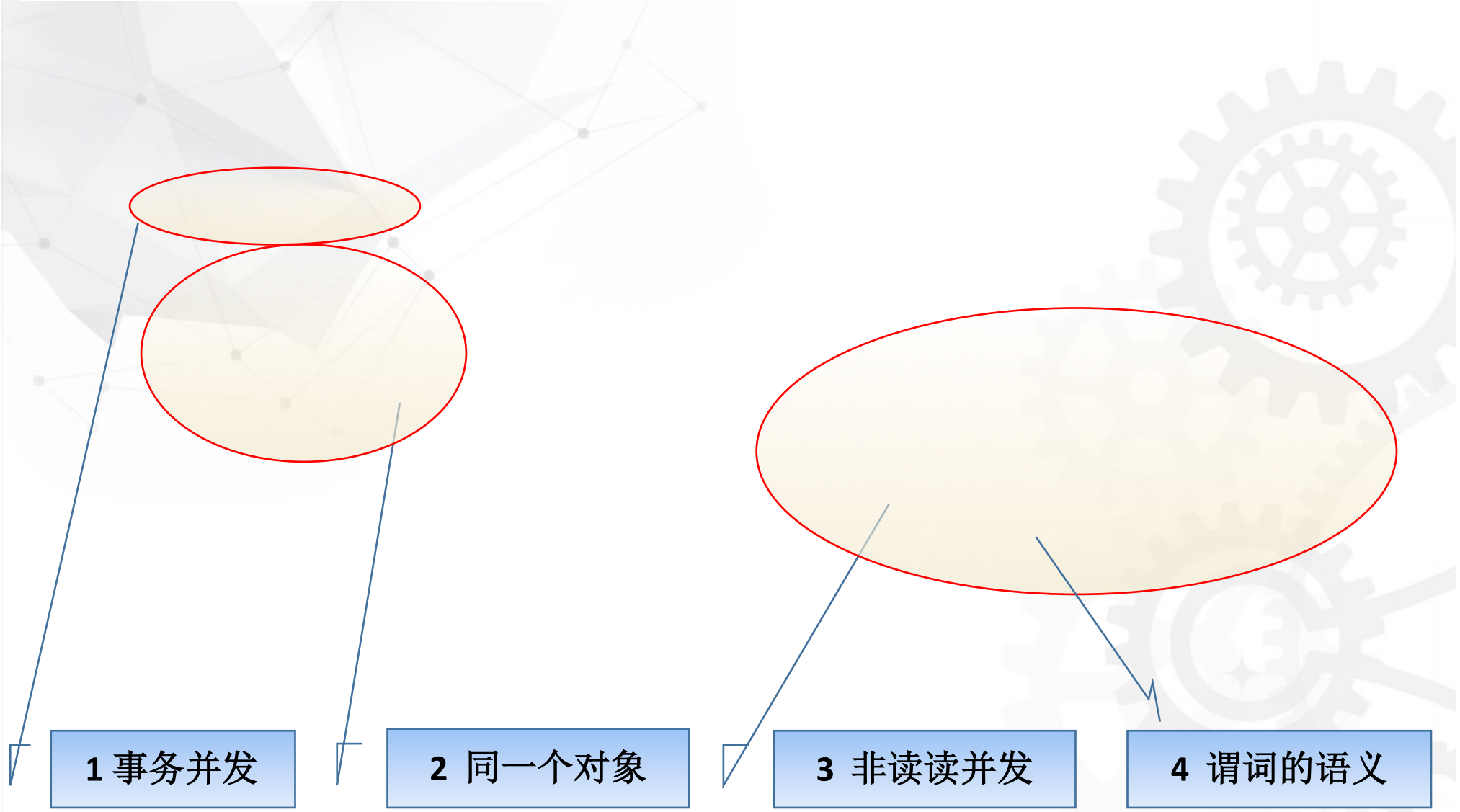
1 事务并发

2 同一个对象

3 非读读并发

4 谓词的语义

Q2 为什么会产生数据异常现象？



Q2 为什么会产生数据异常现象？

	两个事务写偏序		三个事务写偏序		
时间	T1	T2	T1	T2	T3
t0	x ← SELECTCOUNT(*) FROM doctors WHERE on-call = true			x ← SELECT current_batch	
t1		x ← SELECTCOUNT(*) FROM doctors WHERE on-call = true			
t2	IF x ≥ 2 THEN UPDATE doctors SET on-call = false WHERE name = Alice				Commit
t3		IF x ≥ 2 THEN UPDATE doctors SET on-call = false WHERE name = Bob	x ← SELECT current_batch		
t4	Commit		SELECT SUM(amount) FROM receipts WHERE batch = x-1		
t5		Commit	COMMIT		
t6					
t7				COMMIT	

4 结果违反约束的语义

语义：
至少有一个医生在提供电话咨询服务

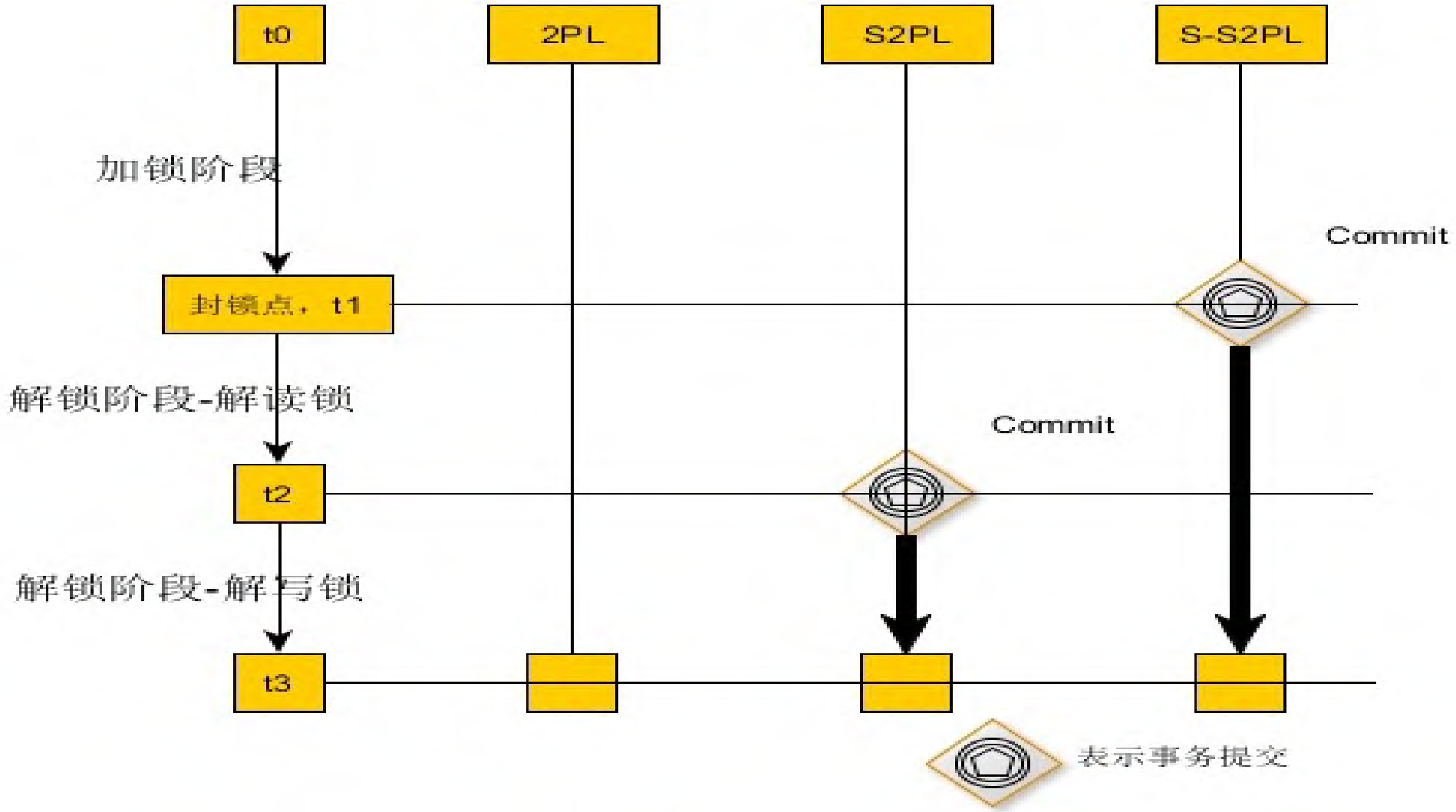
1 事务并发

2 不同的对象

3 非读读并发

Q3 并发控制技术有哪些？

Q3 并发控制技术有哪些？--强严格两阶段锁



Q3 并发控制技术有哪些？--强严格两阶段锁

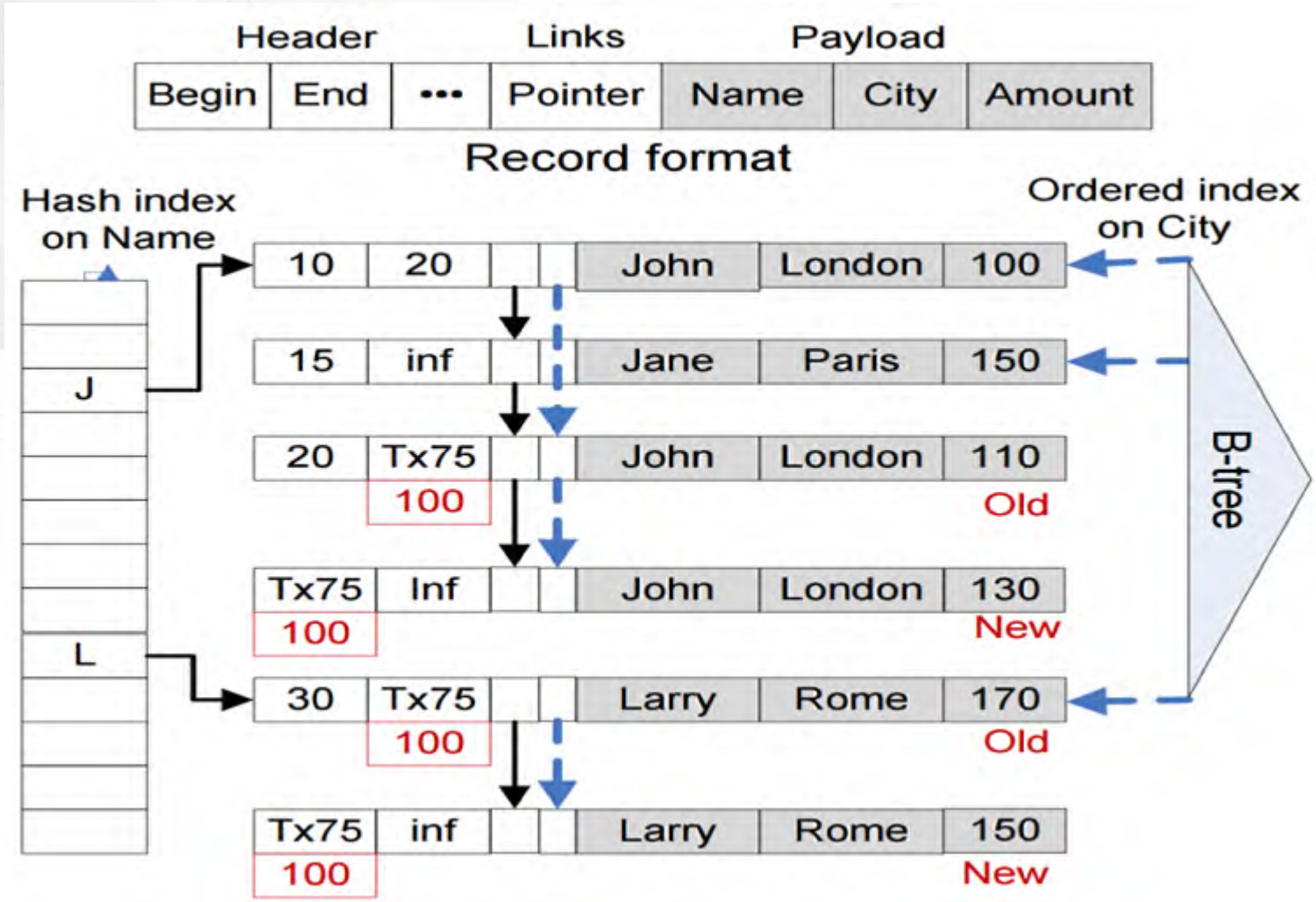
Q3 -1 哪个数据库使用了两阶段锁技术？

Q3 -2 使用两阶段锁技术一定能解决并发访问造成的数据不一致？

SS2PL在读操作加锁 ➔ **序列化隔离级别** ➔ **数据一致**

- 并发控制技术
  - 确保正确性
- I：隔离性
  - 其他的隔离级别，提高并发度
- C：一致性

Q3 并发控制技术有哪些？--多版本，MVCC



图源自论文《Hekaton: SQL Server's Memory - optimized OLTP Engine》

Q3 并发控制技术有哪些？--多版本，MVCC

《Concurrency Control in Distributed Database Systems》，1981年

比较项	多版本时间戳排序机制	多版本两阶段封锁协议
可串行化的方式	通过开始事务的时间戳值来排序事务的提交顺序以及本协议确定的规则来确保可串行化	通过两阶段封锁机制和严格性确定事务的提交顺序来确保可串行化
读请求不失败不等待	是	是
读-写冲突的解决方式	写操作被撤消	写操作能够执行，生成一个新的版本
写-读冲突的解决方式	读不被阻塞	读不被阻塞
写-写冲突的解决方式	取决于事务的时间戳，可能被回滚，也可能被允许执行	排它锁保证后发生的写操作被阻塞
产生潜在的磁盘访问	读取数据项要更新 R-timestamp(Xi)字段，一读一更新共两个 IO	--
是否区分只读事务和更新事务	不区分	区分
适用范围	大多数事务为只读或很少有并发事务更新相同数据项	并发冲突高的情况（延迟一些操作避免了事务回滚带来的更大开销）

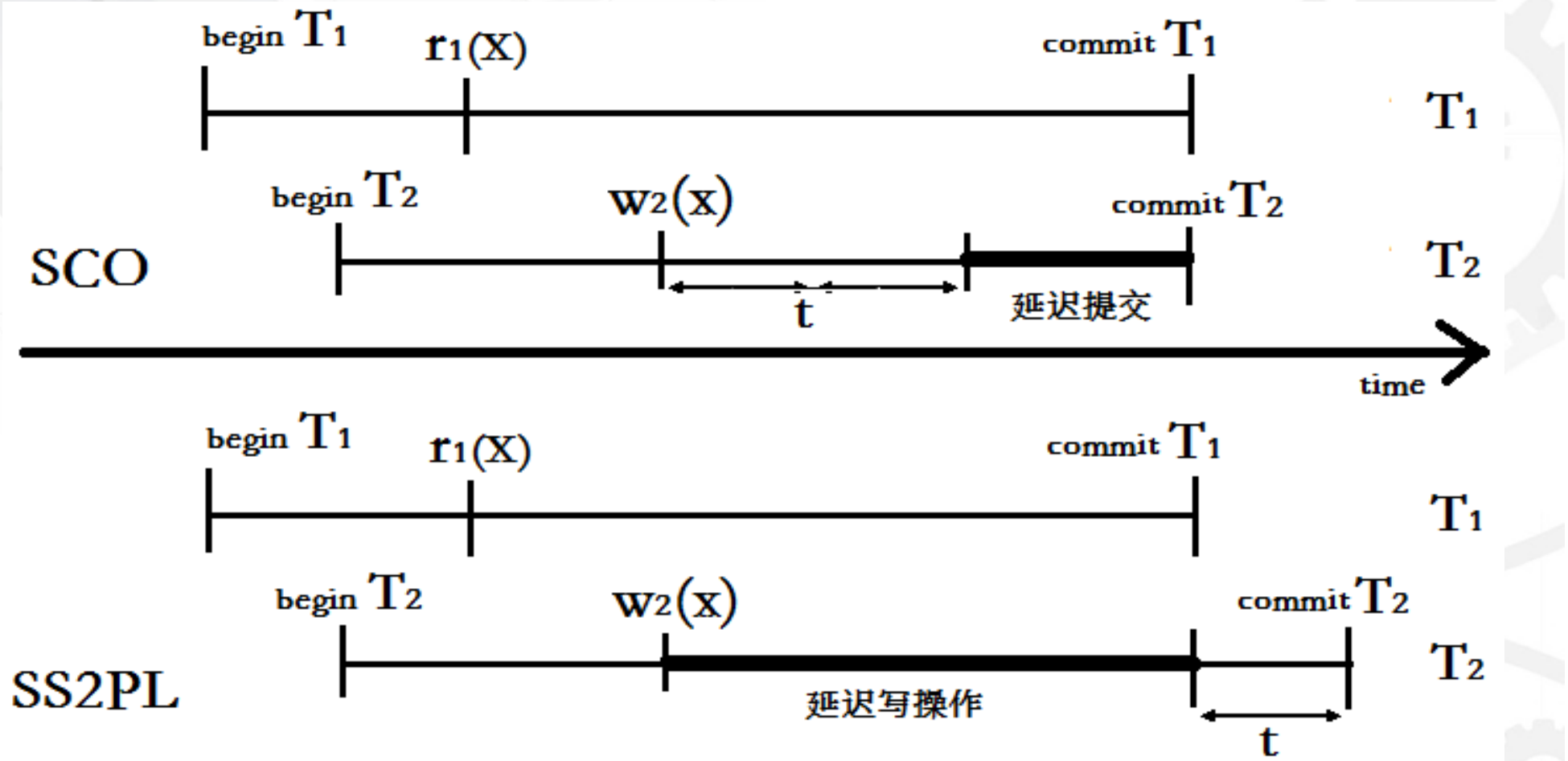
Q3 并发控制技术有哪些？--多版本，MVCC

比较项	多版本时间戳排序机制	多版本两阶段封锁协议
可串行化的方式	通过开始事务的时间戳值来排序事务的提交顺序以及本协议确定的规则来确保可串行化	通过两阶段封锁机制和严格性确定事务的提交顺序来确保可串行化
读请求不失败不等待	是	是
读-写冲突的解决方式	写操作被撤消	写操作能够执行，生成一个新的版本
写-读冲突的解决方式	读不被阻塞	读不被阻塞
写-写冲突的解决方式	取决于事务的时间戳，可能被回滚，也可能被允许执行	排它锁保证后发生的写操作被阻塞

Q3 -3 多版本 + 快照算作上述哪一种技术？

Q3 并发控制技术有哪些？--严格提交顺序

Commitment ordering



Q4 为什么并发控制技术能解决异常现象？ --封锁协议

规则

	脏读		不可重复读		幻象	
	P1 ("Dirty read")		P2 ("Non-repeatable read")		P3 ("Phantom")	
时间	T1主事务	T2	T1主事务	T2	T1主事务	T2
t0		W(row)-Update	R(row)		R(rows)-WHERE<conditon>	
t1	R(row)			W(row)-Update/Delete		W(rows)-Insert/Update =><conditon>
t2		Abort		Commit	R(rows)-WHERE<conditon>	
t3			R(row)			
封锁技术	并发事务T2施加写锁成功后，事务T1的读操作被阻塞，所以封锁技术中不存在脏读异常		并发事务T1施加读锁成功后，事务T2的写操作被阻塞，所以封锁技术中不存在不可重复读异常		并发事务T1和T2操作的不是同一个对象，事务T1可以在数据库中存在的数据项上加锁，但事务T2可能插入新的数据，这些数据是之前不存在的因而无法用锁排斥，所以幻象异常需要新的解决方式	

Q4 为什么并发控制技术能解决异常现象？ --封锁协议

锁的相容性矩阵表（**不同的事务间**新锁的申请）

		Granted Mode，已经授予的锁						
		N	IS	IX	S	SIX	U	X
Requested Mode 正申请的锁	IS	Y	Y	Y	Y	Y		
	IX	Y	Y	Y				
	S	Y	Y		Y			
	SIX	Y	Y					
	U	Y			Y			
	X	Y	Y					

封锁技术规则

锁的合并图（**同一个事务内**锁的升级）

		Lock Held，持有的锁					
		IS	IX	S	SIX	U	X
Lock Requested 正申请的锁	IS	IS	IX	S	SIX	U	X
	IX	IX	IX	SIX	SIX	X	X
	S	S	SIX	S	SIX	U	X
	SIX	SIX	SIX	SIX	SIX	X	X
	U	U	X	U	X	U	X
	X	X	X	X	X	X	X

Q4 为什么并发控制技术能解决异常现象？

对 并发 进行 控制的技术

↓

对 并发 进行 限制的技术

无并发：串行化 → 确保一致性

有并发：可串行化 → 序列化隔离级别

} 一致性 正确

↓

} 隔离性 ← 性能

牺牲一致性：可重复读隔离级别
读已提交隔离级别
读未提交隔离级别

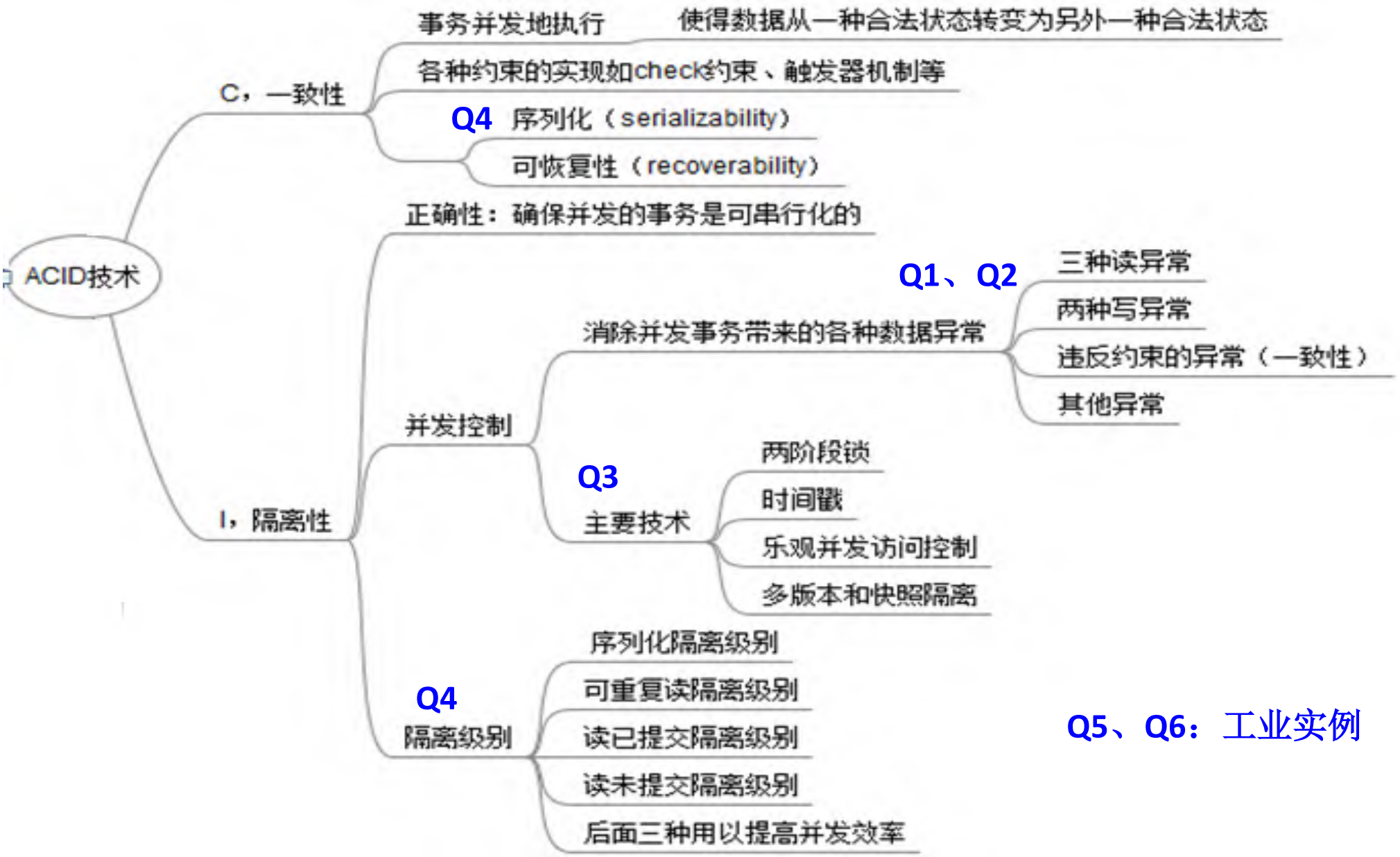
Q5 主流数据库的并发控制技术？

	封锁技术	MVCC
Informix	SS2PL	-
Oracle	SS2PL	采用 Snapshot isolation，假的serializable
PostgreSQL	SS2PL 显式+元数据（DDL）	主要技术 +快照 DML 可重复性读、读已提交、**序列化(SSI)**
MySQL	SS2PL **序列化(读加锁)**	辅助 +快照 可重复读、读已提交

Q6 PostgreSQL V MySQL?

	PostgreSQL	MySQL
系统锁	Spinlock/latch(Lwlock)	my_mutex_lock
事务锁	lock	Lock
元数据锁 DDL并发、DDL+DML并发	ReguarLock锁	MDL_lock
记录锁	在MVCC下，记录在元组头	记录锁表
序列化隔离级别	SSI（Serializable Snapshot Isolation），基于MVCC	加读锁
其他隔离级别 RR、RC	MVCC用快照实现	依靠MVCC用快照实现

今天分享了什么？收获了什么？



Q5、Q6：工业实例

希望有机会分享……

- Q7 为什么SQL标准只规定了三种读异常？
- Q8 如何识别某个DB的并发访问控制技术是什么？
- Q9 为什么封锁技术可以和MVCC技术混合使用？
- Q10 为什么主流数据库都采用了封锁的SS2PL？
- Q11 元数据和用户数据的并发访问控制技术有差别吗？
- Q12 怎么深入理解隔离级别？

2017 DTCC 数据库内核技术专场 @那海蓝蓝

PostgreSQL, MySQL, Greenplum, Informix, etc

@那海蓝蓝 Blog: http://blog.163.com/li_hx/ Database_xx@126.com

2014 《数据库查询优化器的艺术: 原理解析与SQL性能优化》

2017 《数据库事务处理的艺术: 事务管理与并发控制》

2017年 机械工业出版社出版

本次分享的Ppt位于:
http://pan.baidu.com/s/1jI6MBg6

今天分享了什么？收获了什么？

《A Critique of ANSI SQL Isolation Levels》

Table 4. Isolation Types Characterized by Possible Anomalies Allowed.

Isolation level	P0 Dirty Write	P1 Dirty Read	P4C Cursor Lost Update	P4 Lost Update	P2 Fuzzy Read	P3 Phantom	A5A Read Skew	A5B Write Skew
READ UNCOMMITTED == Degree 1	Not Possible	Possible	Possible	Possible	Possible	Possible	Possible	Possible
READ COMMITTED == Degree 2	Not Possible	Not Possible	Possible	Possible	Possible	Possible	Possible	Possible
Cursor Stability	Not Possible	Not Possible	Not Possible	Sometimes Possible	Sometimes Possible	Possible	Possible	Sometimes Possible
REPEATABLE READ	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Possible	Not Possible	Not Possible
Snapshot	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Sometimes Possible	Not Possible	Possible
ANSI SQL SERIALIZABLE == Degree 3 == Repeatable Read Date, IBM, Tandem, ...	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible

TDSQL

@腾讯 金融云

THANKS

SequeMedia 盛拓传媒 IT168.com ITPUB ChinaUnix