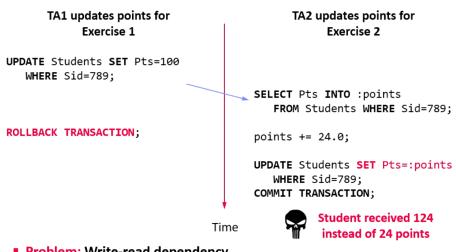
Transaction Anomalies

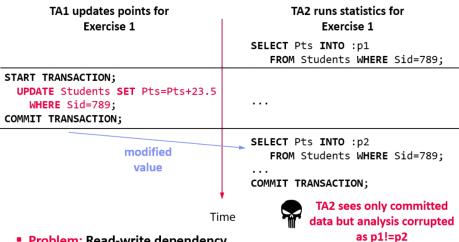
Anomalies - Lost Update

TA1 updates points for TA2 updates points for Exercise 1 Exercise 2 **SELECT** Pts **INTO** :points FROM Students WHERE Sid=789; **SELECT** Pts **INTO** :points FROM Students WHERE Sid=789; points += 23.5; points += 24.0; UPDATE Students SET Pts=:points WHERE Sid=789; UPDATE Students SET Pts=:points COMMIT TRANSACTION; WHERE Sid=789; COMMIT TRANSACTION; **Student received 24** Time instead of 47.5 points (lost update 23.5) Problem: Write-write dependency Solution: Exclusive lock on write

Anomalies - Dirty Read



- Problem: Write-read dependency
- Solution: Read only committed changes; otherwise, cascading abort



- Problem: Read-write dependency
- Solution: TA works on consistent snapshot of touched records

Anomalies - Phantom

TA1 inserts missing student	TA2 runs statistics for Exercise 1
	SELECT Avg(Pts) INTO :p1 FROM Students WHERE Sid<1000;
START TRANSACTION; INSERT INTO Students VALUES (999,, 0); COMMIT TRANSACTION;	
added row (harder to track because new database object)	SELECT Avg(Pts) INTO :p2 FROM Students WHERE Sid<1000; COMMIT TRANSACTION;
Til	TA2 sees only committed data but analysis corrupted as p1!=p2

- Similar to non-repeatable read but at set level
- (snapshot of accessed data objects not sufficient)

Transaction Isolation Level

- guarantees certain anomalies cannot happen
- sacrifices performance
 - Tradeoff: isolation (and related guarantees) vs performance
 - READ UNCOMMITTED (lost update, dirty read, unrepeatable read, phantom R)
 - READ COMMITTED (lost-update, dirty read, unrepeatable read, phantom R)
 - REPEATABLE READ (lost update, dirty read, unrepeatable read, phantom R)
- SERIALIZABLE (lost update, dirty read, unrepeatable read, phantom R)

SQL Standard Isolation Levels

Isolation Level	Lost Update	Dirty Read (P1)	Unrepeatable Read (P2)	Phantom Read (P3)
READ UNCOMMITTED	No*	Yes	Yes	Yes
READ COMMITTED	No*	No	Yes	Yes
REPEATABLE READ	No*	No	No	Yes
[SERIALIZABLE]	No*	No	No	No

Serializable w/ highest guarantees (pseudo-serial execution)

- isolation level can be set by user or system
 - User: set default/transaction isolation level (mixed TX workloads possible)
 - System: dedicated concurrency control strategies + scheduler
- maximum and default isolation level may vary by DBMS

Database	Default	Maximum		
Actian Ingres 10.0/10S [1]	S	S		
Aerospike [2]	RC	RC		
Akiban Persistit [3]	SI	SI		
Clustrix CLX 4100 [4]	RR	RR		
Greenplum 4.1 [8]	RC	S		
IBM DB2 10 for z/OS [5]	CS	S		
IBM Informix 11.50 [9]	Depends	S		
MySQL 5.6 [12]	RR	S		
MemSQL 1b [10]	RC	RC		
MS SQL Server 2012 [11]	RC	S		
NuoDB [13]	CR	CR		
Oracle 11g [14]	RC	SI		
Oracle Berkeley DB [7]	S	S		
Oracle Berkeley DB JE [6]	RR	S		
Postgres 9.2.2 [15]	RC	S		
SAP HANA [16]	RC	SI		
ScaleDB 1.02 [17]	RC	RC		
VoltDB [18]	S	S		
RC: read committed, RR: repeatable read, SI: snapshot isola-				

RC: read committed, RR: repeatable read, SI: snapshot isolation, S: serializability, CS: cursor stability, CR: consistent read

^{*} Lost update potentially w/ different semantics in standard

Critique of SQL Isolation Level

 Criticism: SQL standard isolation levels are ambiguous (strict/broad interpretations) O'Neil, Patrick E. O'Neil: A of ANSI SQL Isolation SIGMO

- Additional anomalies: dirty write, cursor lost update, fuzzy read, read skew, write skew
- Additional isolation levels: cursor stability and snapshot isolation

Snapshot Isolation (< Serializable)

- Type of optimistic concurrency control via multi-version concurrency control
- TXs reads data from a snapshot of committed data when TX started
- TXs never blocked on reads, other TXs data invisible
- TX T1 only commits if no other TX wrote the same data items in the time interval of T1

Current Status?

[nttp://apmsmusings.biogspot.com/zu]
introduction-to-transaction-isolation

 "SQL standard that fails to accurately define database isolation levels and database vendors that attach liberal and non-standard semantics"