HISTEX (HISTory EXerciser)
An Overview

by

Dimitrios Liarokapis, Univ. of Ioannina, Greece

Elizabeth O'Neil, UMASS Boston, USA

Patrick O'Neil, UMASS Boston, USA

Proceedings of the 34th International Conference on Information Technologies (InfoTech-2020) IEEE Conference, Rec #49733, 17-18 September 2020 Bulgaria

A Database System

- A Database System maintains information about a world.
- It can be *queried* for providing a *view* of the world.
- It can be *updated* for reflecting *changes* in the world.

Transaction

- The interaction between a User and A DBMS is done by using *Transactions*.
- The following properties must be satisfied:
- **Atomicity -** Do all the work or not at all.
- **Consistency -** Leave the database consistent
- **Isolation -** Act as if you were alone
- **Durability -** The results are permanent

Concurrency

- ***Concurrency*** is the ability of a Database System to process interleaved transactions.
- Increases the utilization of a system. While a transaction is accessing a disk, another transaction can use the CPU.
- Better response. A small transaction need not wait until a long one finishes first.
- Parallelism is exploited (disks, cpus)

Are there any risks ?

- When transactions operate on different data there are virtually no risks.
- When they operate on common data their interleaving can produce an incorrect result.
- We consider the result of a transaction to be the values retrieved and the state the database is being left.

What is Correctness ?

- A ***concurrent*** execution of a set of transactions is considered correct if the results are the same with a ***serial*** execution of the same transactions.
- This criterion of correctness is called: SERIALIZABILITY

Modeling Transactions

- A transaction can be modeled as a sequence of *read* and *write* actions.
- It includes a final *commit* or *abort* action.
- The following can be the transaction that transfers $50 from account A to B.
- T : r(A,100) r(B,200) w(B,250) w(A,50) c

Transactional Histories

- We model interleaved executions of multiple transactions with histories.
- The following is a history representing the execution of two transactions T1 and T2.
- T1 transfers $50 from A to B.
- T2 computes the total of the two accounts.
- r1(A,100) r1(B,200) r2(A,100) w1(A,50), w1(B,250) c1 r2(B,250) c2

Non Serializable Histories

- r1(A,100) r1(B,200) r2(A,100) w1(A,50), w1(B,250) c1 r2(B,250) c2 *is not a serializable history.*
- If T1, T2 were executed serially then we would have one of the following histories:
- r1(A,100) r1(B,200) w1(B,250), w1(A,50) c1 r2(A,50) r2(B,250) c2
- r2(A,100) r2(B,200) c2 r1(A,100) r1(B,200) w1(B,250), w1(A,50) c1

Serializability by locking

- A database system can prevent non serializable executions by using locking.
- Locks can be shared or exclusive and of long or short duration.
- Shared locks are compatible with shared locks.
- Exclusive locks are not compatible with other locks.

2 Phase Locking

- PHASE I : ALL locks are acquired.
- PHASE II: Locks can be released.
- Usually phase II occurs at the end of a transaction.
- **Deadlocks** can occur under 2PL, and the system needs to detect them.

Example of 2 Phase Locking

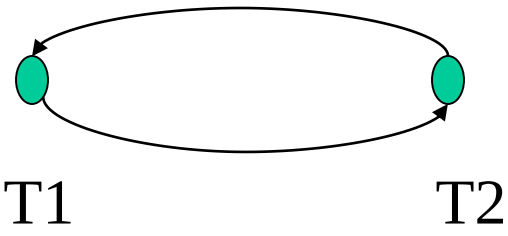
- Consider a database system receiving the following requests.
- r1(A) r1(B) r2(A) w1(A) r2(B) w1(B) c1 c2
- w1(A) and w1(B) will be blocked until T2 commits. The final execution will be:
- r1(A) r1(B) r2(A) r2(B) c2 w1(A) w1(B) c1
- The above will be equivalent to: T2,T1

Serialization Graph
(Conflict Serializability)

- The nodes of the graph represent the transactions in the History
- For every *conflicting* pair of operations there is an edge between the corresponding transactions. Two operations conflict if they operate on the same data and one of them is a *write*.
- If there is a *cycle* in the graph the history is not serializable.

Example of a Serialization Graph

- r1(A,100) r1(B,200)
  r2(A,100) w1(A,50)
  w1(B,250) c1
  r2(B,250) c2
- A cycle indicates that a transaction takes place before and after of another transaction.



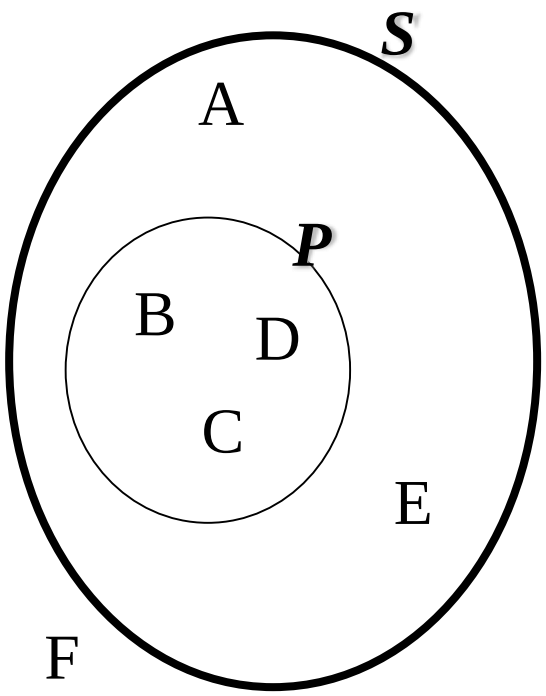
Serializability by multiple versions.

- A system can maintain multiple versions of a data item. Every write operation creates a new version. The system decides what version will be returned to a read operation.
- r1(A0,100) r1(B0,200) r2(A0,100) w1(A1,50), w1(B1,250) c1 r2(B0,200) c2

Predicates & Phantoms

- In real database systems there are operations that operate on a group of data items that satisfy a *predicate* (e.g. select sum(balance) from accounts where city=Boston)
- New types of conflicts are introduced: *predicate conflicts*
- Traditional data item locking is not protective enough leading to *phantoms* (e.g. insert into accounts (acount_id, balance,city) values (1001, $30, Boston)

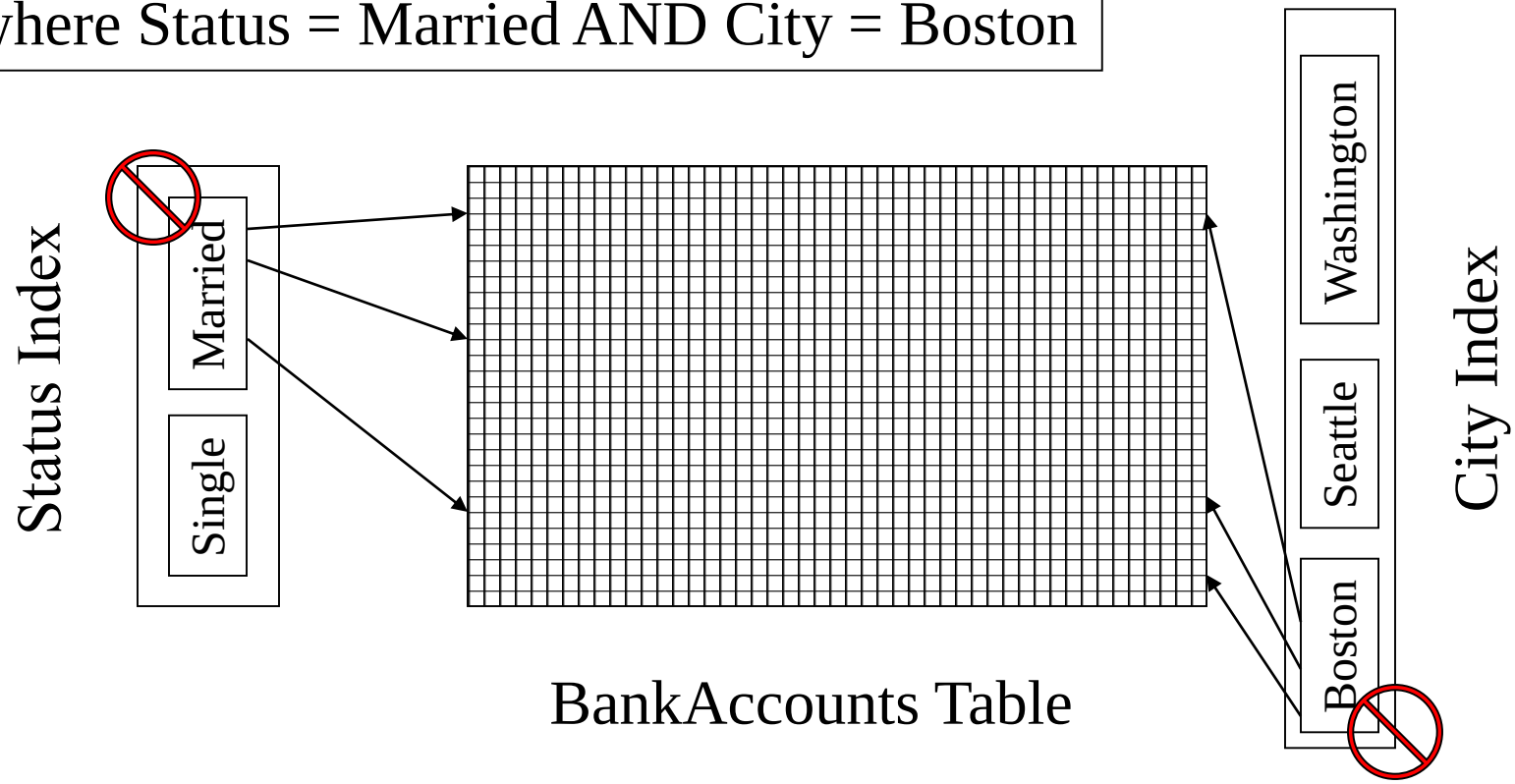
Predicate Conflicts



- **PR**(***P***): reads the data items {B,C,D} and conflicts with:
- **W**(A into ***P***): updates A so that satisfies P.
- **W**(D outof ***P***): updates D to not satisfy P.
- **I**(F in ***P***): inserts the item F that satisfies P.
- **D**(C in ***P***): deletes the item C that satisfies P.

Approximating Predicate Locks by Using Index Locking

Select * from BankAccounts
where Status = Married AND City = Boston



Isolation Levels

A feature provided by database management systems in order to increase performance when:

- Full correctness is not necessary or
- Correctness could be assured at the application level

Isolation Levels (ANSI)

Isolation Level	P1	P2	P3
READ UNCOMMITTED	Possible	Possible	Possible
READ COMMITTED	Not Possible	Possible	Possible
REPEATABLE READ	Not Possible	Not Possible	Possible
SERIALIZABLE	Not Possible	Not Possible	Not Possible

1. Dirty Read (P1). A transaction T1 modifies some row and another transaction T2 reads that row before T1 commits. The implications of this phenomenon is that if transaction T1 issues a ROLLBACK statement (abort) it will be as if transaction T2 read values that have never existed.
2. Non-Repeatable Read (P2). A transaction T1 reads some row. Another transaction T2 modifies that row and performs a commit. If T1 attempts to re-read that row it can observe the changes done by transaction T2.
3. Phantom (P3). A transaction T1 reads a set of rows that satisfy some condition. Another transaction T2 executes a statement that causes new rows to be added or removed from the search condition. If T1 repeats the read it will obtain a different set of rows.

1. The Serializable level should allow only serializable executions.
2. There are no visible uncommitted actions except for the RU level.
3. There are no update operations allowed at the RU level.

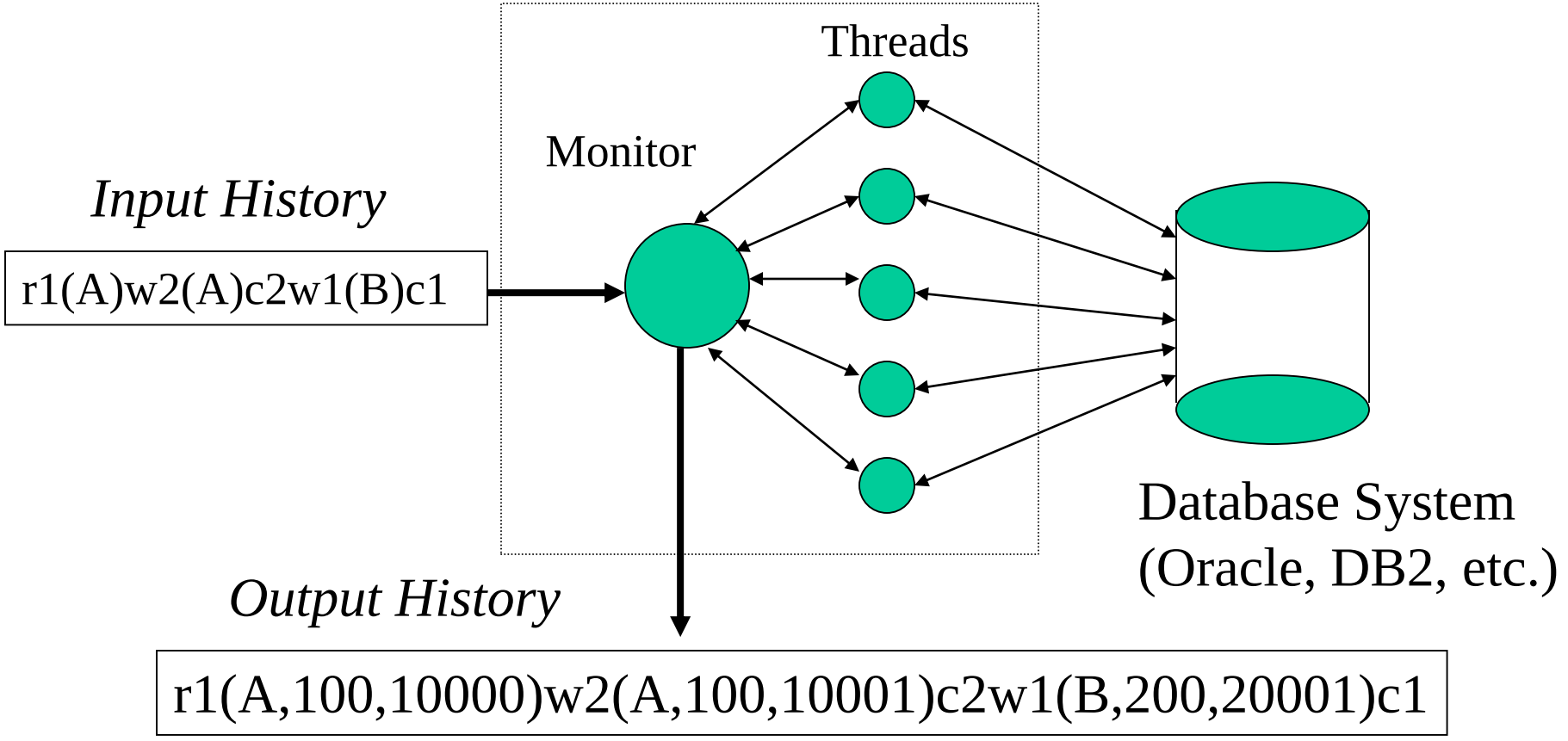
Definitions in “Generalized Isolation Levels”

by A.Atul, B.Liskov, P.Oneil

Isolation Level	GO	G1	G2-item	G2
READ UNCOMMITTED	NA	NA	NA	NA
READ COMMITTED	Not Possible	Not Possible	Possible	Possible
REPEATABLE READ	Not Possible	Not Possible	Not Possible	Possible
SERIALIZABLE	Not Possible	Not Possible	Not Possible	Not Possible

- G0: The Serialization Graph can contain a cycle consisting only of w-w edges
- G1: Aborted or Intermediate Reads or the SG contains a cycle consisting only of w-w, w-r, w-pr edges
- G2-item: The SG contains a cycle with one or more r-w edges.
- G2: The SG contains a cycle with one or more r-w or pr-w edges.

HISTEX


Threads
Monitor
Input History
r1(A)w2(A)c2w1(B)c1
Database System
(Oracle, DB2, etc.)
Output History
r1(A,100,10000)w2(A,100,10001)c2w1(B,200,20001)c1

Histex Data Model

Data Item Map

A	100
B	200
C	300

Value Map

X1	1000
X2	0
Y	300

Predicate Map

P	k2=0
Q	k3 > 1
…	

Operation Examples:

R(A,X1)

W(B,898), W(C,X1)

PRED(W,"K50=49")

PR(P)

PR(P;10, Z1)

Database

REC KEY	REC VAL	C2	C3	C4	C5	C6	C50	C100	K2	K3	K4	K5	K6	K50	K100
100	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
200	2000	1	1	1	1	1	1	1	1	1	1	1	1	1	1
300	3000	0	2	2	2	2	2	2	0	2	2	2	2	2	2
400	4000	1	0	3	3	3	3	3	1	0	3	3	3	3	3
…															

Histex Operations

Text Book Notation	Functionality	Examples of Implemented Notation
A_i	**Abort**	1,a,,
I_i(A[;col1[;col2 ...]] [,num[;num ...]])	**Insert Row**	1,I,A;c2,4
C_i	**Commit**	1,c,,
D_i(A)	**Delete Row**	1,D,A,
EXECSQLI$_i$(statement)	**Execute a SQL statement (immediate mode)**	1,execsqli,"update T set recval = recval + 1 where %P",
EXECSQLS$_i$(statement)	**Execute a SQL statement (open cursor)**	1,execsqls,"select sum(recval) from T group by k1, k2 having %P",
IL$_i$(UR|CS|RS|RR)	**Isolation Level (DB2)**	1,il,UR,
IL$_i$(RU|RC|RR|SR)	**Isolation Level (Informix)**	1,il,RC,
IL$_i$(RC|SR)	**Isolation Level (Oracle)**	1,il,SR,
MAP(A,ID)	**Map a row**	0,map,A,100
PRED(P,predicate)	**Predicate Declaration**	0,pred,P,*" k2=1 and k3<2"*
PR$_i$(P;col;i[;A] [,X])	**Predicate Read**	1,pr,P;recval;1;A,A0
R$_i$(A[;column][,X])	**Read a row**	1,r,A,A0
RW$_i$(A[;column][,exp]	**Read & update a row**	1,rw,A;k2,k2+k3
W$_i$(A[;column][,{X|num}])	**Update a row**	1,w,A,1001

Adequacy of testing pairs of conflicting operations

Theorem: If a database system prevents the concurrent operations in the following table then it implements the isolation levels correctly. By correctly we mean that at a given isolation level there will be no phenomenon occurring that is proscribed by that level.

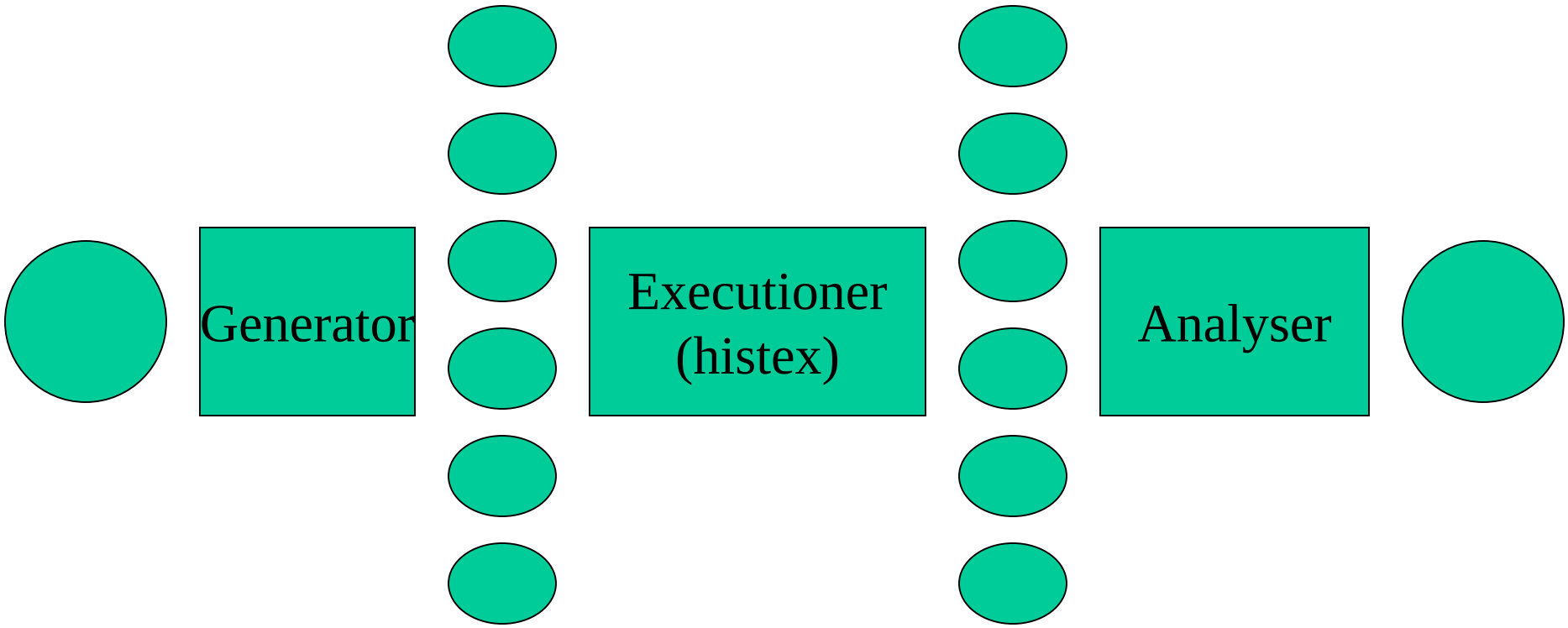
Isolation Levels
(Restrictive Definitions)

Isolation Level	Proscribed pairs of concurrent operations
Read Uncommitted	Transactions are READ ONLY. There should not be concurrent conflicting operations.
Read Committed	$W_1(A)W_2(A)$ $W_1(A)R_2(A)$ $W_1(A \text{ changes } P)\ PR_2(A)$
Repeatable Read	All above and $R_1(A)\ W_2(A)$
Serializable	All above and $PR_1(A)\ W_2(A \text{ changes } P)$

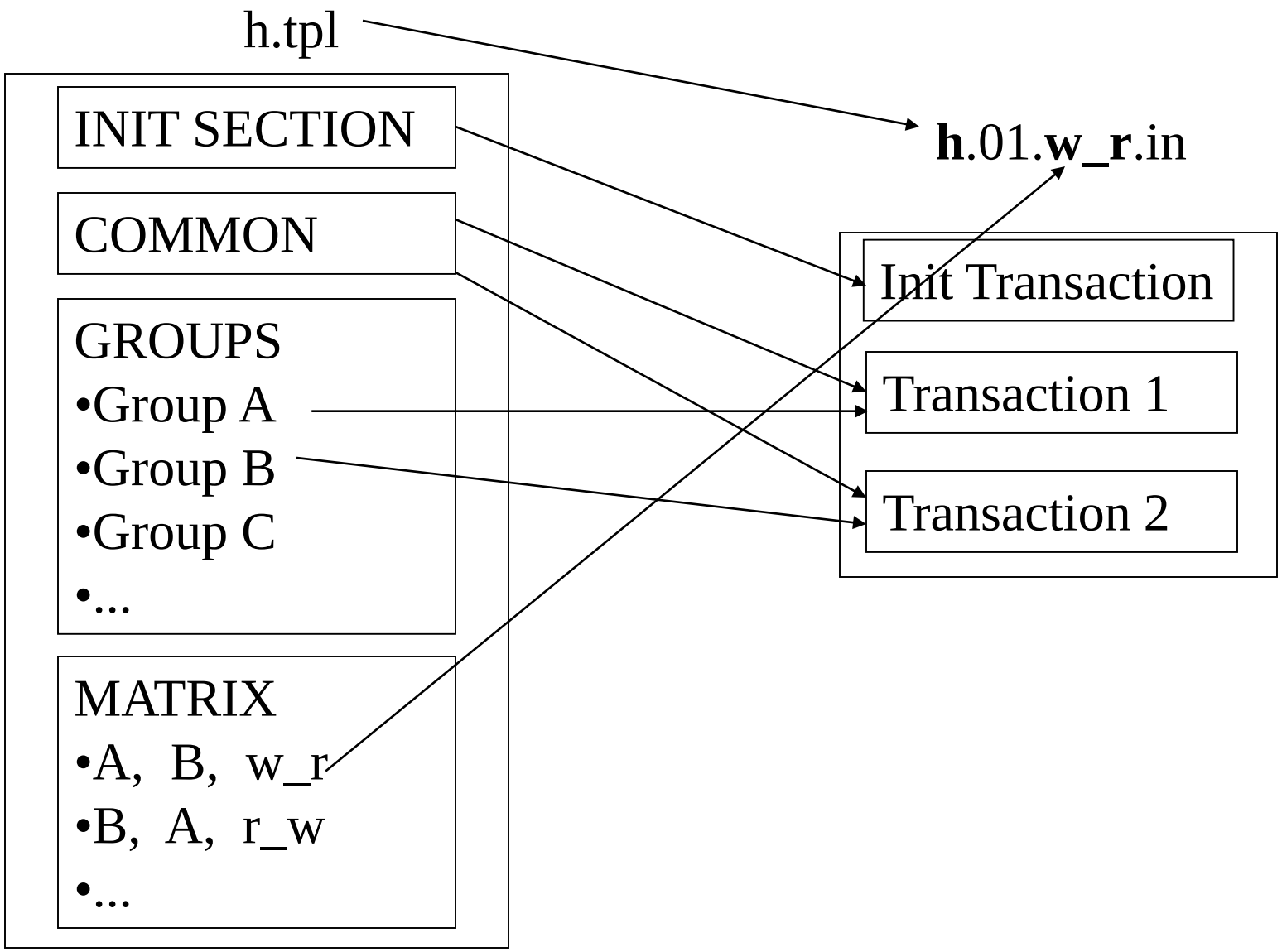
Theorem Proof
(READ COMMITTED)

- *Aborted Reads* [w1(A)…r2(A)…a1c2] and *Intermediate Reads* [w1(A)…r2(A)…w1(A)...c2] could not occur because a concurrent pair of operations w1(A) r2(A) could not occur.
- *No cycle with w-w, w-r, w-pr edges could exist in the SG.* If a cycle (T1,T2…Tn) existed then T2 should commit after T1 and consequently Tn should commit after T1. But because of the edge Tn->T1 in the graph Tn should commit before T1

Testing System



Generator


h.tpl
INIT SECTION
COMMON
GROUPS
•Group A
•Group B
•Group C
•...
MATRIX
•A, B, w_r
•B, A, r_w
•...
h.01.w_r.in
Init Transaction
Transaction 1
Transaction 2

Mixed Isolation Levels
(Inverted Definitions)

History Class	Isolation Levels permitting execution
w_w	None
w_r	None
w_pr	None
r_w	RC_RC, RC_RR, RC_SR
pr_w	RC_RC, RR_RR, RC_RR, RR_RC, RC_SR, RR_SR

Tested Commercial RDBMS

- IBM DB2 (5.0, 6.1, 11.1.4)
- INFORMIX UDB (9.0, 14.1.0)
- ORACLE (8.1.6, 11) (*ad hoc examination*)

Erroneous Output
(Retrieved from a Commercial RDBMS)

Output file	prkey index	prkey noindex	noprkey index	noprkey noindex
h.14.w_pr.db2.CS_CS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.14.w_pr.db2.CS_RS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.14.w_pr.db2.RR_CS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.14.w_pr.db2.RR_RS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.14.w_pr.db2.RS_CS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.14.w_pr.db2.RS_RS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.16.w_pr.db2.CS_CS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.16.w_pr.db2.CS_RS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.16.w_pr.db2.RR_CS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.16.w_pr.db2.RR_RS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.16.w_pr.db2.RS_CS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.16.w_pr.db2.RS_RS	EXECUTED*	TIMEOUT	EXECUTED*	TIMEOUT
h.17.w_pr.db2.CS_CS	EXECUTED*	EXECUTED*	EXECUTED*	EXECUTED*
h.17.w_pr.db2.CS_RS	EXECUTED*	EXECUTED*	EXECUTED*	EXECUTED*
h.17.w_pr.db2.RR_CS	EXECUTED*	EXECUTED*	EXECUTED*	EXECUTED*
h.17.w_pr.db2.RR_RS	EXECUTED*	EXECUTED*	EXECUTED*	EXECUTED*
h.17.w_pr.db2.RS_CS	EXECUTED*	EXECUTED*	EXECUTED*	EXECUTED*
h.17.w_pr.db2.RS_RS	EXECUTED*	EXECUTED*	EXECUTED*	EXECUTED*
h.23.w_pr.db2.CS_CS	TIMEOUT	TIMEOUT	TIMEOUT	EXECUTED*
h.23.w_pr.db2.CS_RS	TIMEOUT	TIMEOUT	TIMEOUT	EXECUTED*
h.23.w_pr.db2.RS_CS	TIMEOUT	TIMEOUT	TIMEOUT	EXECUTED*
h.23.w_pr.db2.RS_RS	TIMEOUT	TIMEOUT	TIMEOUT	EXECUTED*

Testing the Read Uncommitted Isolation Level

$R_1(A)R_1(B)C_1W_2(A)\ R_3(A, A0)\ W_3(B, A0)\ C_3\ A_2\ R_4(A)\ R_4(B)\ C_4$

(map, A, 100)
(map, B, 200)
(1, r, A [=100], [=10000])
(1, r, B [=200], [=20000])
(1, c)
(2, w, A [=100], [=1730691225])
(3, execsqls, select recval from T where reckey = 100 FOR READ ONLY, A0 [=1730691225])
(3, w, B [=200], A0 [=1730691225])
(3, c)
(2, a)
(4, r, A [=100], [=10000])
(4, r, B [=200], [=1730691225])
(4, c)

Thank you!

dili@uoi.gr

Additional info is available at

www.cs.umb.edu/~dimitris/thesis