IMAT3104
Database Management and Progamming

Relational database transactions, concurrency and recovery

Objectives

- Explain the concept of a relational database transaction.
- Explain the problems of transaction concurrency
- Examine the recovery of lost transactions

Introduction

- In database design we consider the support of transactions on the database.
- RDBMS facilities include:
  - Concurrency control.
  - Recovery.
  - Security.
  - Integrity.
- The transaction is central to understanding how a relational database is kept in a complete & consistent state.

Transactions

- Fundamental purpose of a database system is to carry out transactions.
- A transaction is a logical unit of work.
- A transaction consists of the execution of an application-specified sequence of operations (e.g SELECT, INSERT, UPDATE, DELETE).

Transaction Example

- One transaction with two operations:

```
UPDATE bank_account
SET Balance = Balance - 100
WHERE AccountNo = 348973;

UPDATE bank_account
SET Balance = Balance + 100
WHERE AccountNo = 678453;
```

- If the transaction failed between these operations, the database would be left in an inconsistent state.

Transaction ACID Properties

- **Atomicity** : An “All or Nothing” unit either executed in its entirety or not at all.
- **Consistency** : Transforms the database from one consistent state into another consistent state.
- **Isolation** : Executes independently of other transactions.
- **Durability** : The effects of a successfully completed transaction are permanently recorded in the database and must not be lost.

COMMIT/ROLLBACK protocol

- A transaction has one of two outcomes:
  - It completes successfully and data is committed to the database,

    or
  - It is unsuccessful and data is rolled back to restore the database to its original consistent state.

- The user must define the boundaries of a transaction:
  - Begin the transaction : START TRANSACTION
  - Make changes to database permanent : COMMIT

    or
  - Undo effects of transaction : ROLLBACK

Transaction Example

```
CREATE PROCEDURE bank_payment_transfer (intAccNoFrom
  INT, intAccNoTo INT, numPayment NUMERIC(10,2))

BEGIN

  DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK;

  START TRANSACTION;

    UPDATE bank_account
      SET dbBalance = dbBalance - numPayment
      WHERE dbAccNo = intAccNoFrom;
    UPDATE bank_account
      SET dbBalance = dbBalance + numPayment
      WHERE dbAccNo = intAccNoTo;

  COMMIT;

END;
```

Log files

- Log files keep track of database transactions.
- Contain information about all changes to a database.
- Support the COMMIT/ROLLBACK protocol
- Log files contain:
  - **Before-image** of a data item, i.e. value before change.
  - **After-image** of a data item, i.e. value after change.
  - Lots of other information …
- ROLLBACK uses before-image values backwards.
- COMMIT uses after-image values forwards.

Concurrency Control

- The process of managing simultaneous transactions on the database without having them interfere with one another.
- Concurrency allows much more efficient use of resources, resulting in improved database performance.
- However, concurrent access can cause interference and result in data inconsistencies

Concurrency Problems

- Three classic types of potential problems caused by concurrency:
  - **Lost update**
  - **Uncommitted dependency**
  - **Inconsistent analysis**

Lost Update problem

Time	Transaction 1	Transaction 2	Acc
t1	SELECT Acc		£100
t2		SELECT Acc	£100
t3	UPDATE Acc+£100		£200
t4		UPDATE Acc-£100	£0

- What happens to Transaction 1's update?

Uncommitted Dependency problem

Time	Transaction 3	Transaction 4	Acc
t1		UPDATE Acc+£200	£150
t2	SELECT Acc		£150
t3		ROLLBACK	-£50

- Transaction 3 is dependent on an uncommitted change at time t2.
- What is the consequence?

Inconsistent Analysis problem

Three accounts with three amounts (£):
1. ACC_1 **400** 2.ACC_2 **500** 3. ACC_3 **300** sum = **1200**

Time	Transaction 5	Transaction 6
t1	SELECT ACC_1 sum = **400**	
t2	SELECT ACC_2 sum = **900**	
t3		SELECT ACC_3
t4		UPDATE ACC_3 **300** >> **200**
t5		SELECT ACC_1
t6		UPDATE ACC_1 **400** >> **500**
t7		COMMIT
t8	SELECT ACC_3 sum = **1100 !!**	

Serialisation

- If transactions were only allowed to execute in series then these concurrency problems would not exist.
- The Serialisation principle:

  When two transactions operate in a parallel interleaved manner, then their effects should be the same as if they operated in a purely serial manner.
- We can guarantee serialisability by using the two-phase locking protocol ...

Locks

- A transaction's lock on data can prevent access to other transactions to prevent incorrect results.
- Two types of lock:
  - Shared read lock (S-lock)
    - shared reading but not writing.
  - Exclusive write lock (X-lock)
    - exclusive access.
- A transaction may have to wait for a lock to be released until placing its own lock on the data item.
- COMMIT & ROLLBACK release locks.

Two Phase Locking

- All locking operations must precede the first unlock operation in the transaction.
- There are two phases:
  - Growing phase - acquires all locks, but releases none.
  - Shrinking phase - releases all locks, but acquires none.

Lost Update solution

Time	Transaction 1	Transaction 2	Acc
t1	SELECT Acc (S-lock)		£100
t2		SELECT Acc (S-lock)	£100
t3	UPDATE Acc+100 (X-lock ?)		
t4	wait	UPDATE Acc-100 (X-lock ?)	
t5	wait	wait	

- No lost update, but a deadlock (more later)

Uncommitted Dependency solution

Time	Transaction 3	Transaction 4	Acc
t1		UPDATE Acc+200 (X-lock)	£150
t2	SELECT Acc (S-lock?)		
t3	wait	ROLLBACK (release X-lock)	-£50
t4	resume SELECT Acc (S-lock)		-£50

- Transaction 3 reads correct value of Account

Inconsistent Analysis solution

Three accounts with three amounts:
1. ACC_1= **400** 2. ACC_2= **500** 3. ACC_3=**300** sum = **1200**

Time	Transaction 5	Transaction 6
t1	SELECT ACC_1 **(S-lock)**	
	sum = **400**	
t2	SELECT ACC_2 **(S-lock)**	
	sum = **900**	
t3		SELECT ACC_3 **(S-lock)**
t4		UPDATE ACC_3 **(X-lock)**
		300 >> **200**
t5		SELECT ACC_1 **(S-lock)**
t6		UPDATE ACC_1 **(X-lock?)**
t7		**wait wait**
t8	SELECT ACC_3 **(S-lock?)**	
t9	**wait**	**wait**

Deadlock

- When two or more transactions are in a simultaneous wait state, each waiting for locks held by the other to be released.
- Breaking the deadlock involves the "lock manager" choosing one of the transactions & rolling it back – this releases its locks and allows the other transaction to proceed.
- Some systems will then re-execute the rolled back transaction.

Starvation

- When a transaction waits for an indefinite period of time while others continue normally.
- For example, a transaction wants an X-lock which it can never get because of new transactions continually gaining S-locks
- Solutions include locking in the order requested or increasing priority according to length of waiting time and number of previous aborts.

Recovery

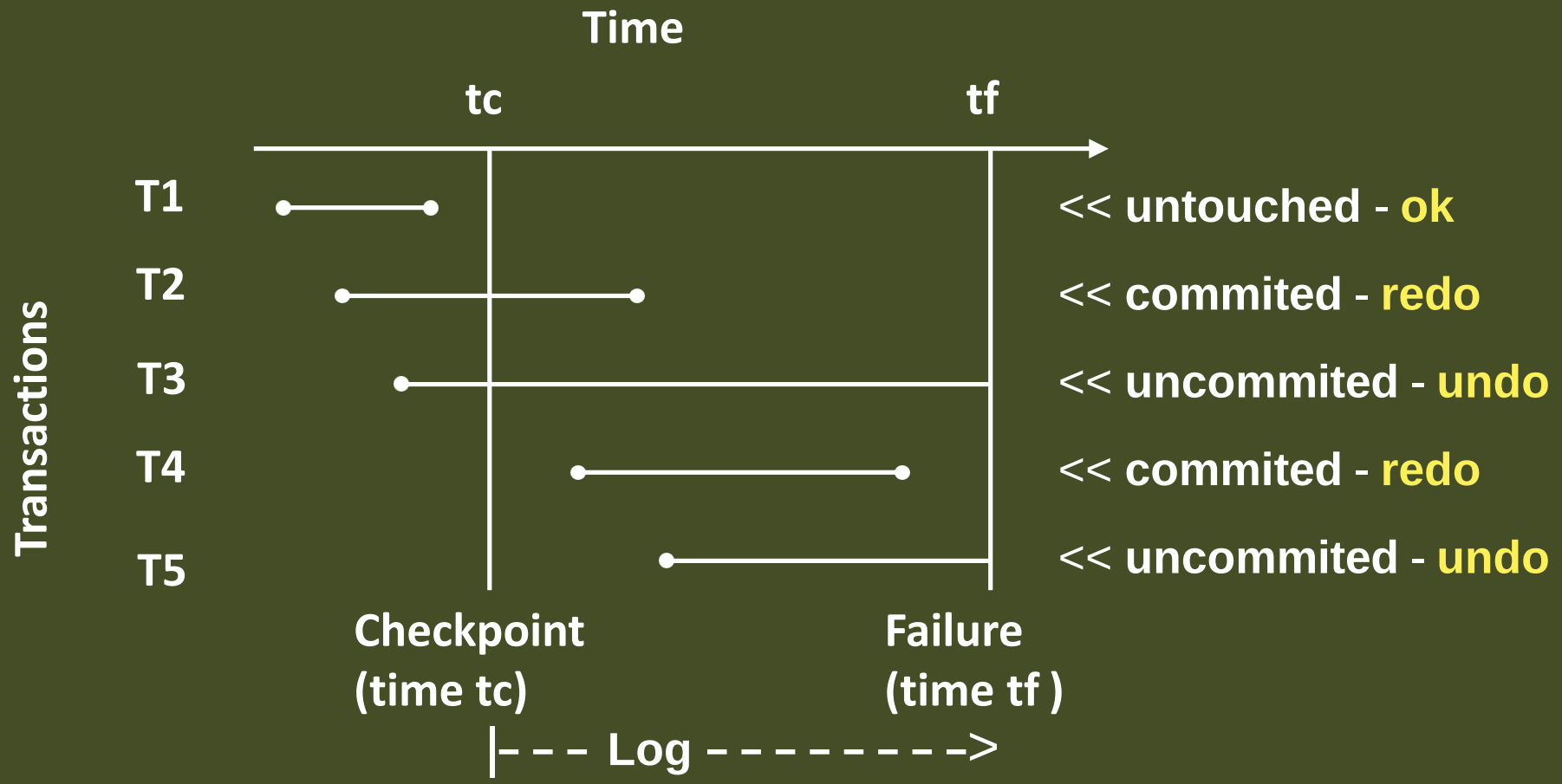
- Restoring the database to a correct state in the event of failure.
- The transaction represents the basic unit of recovery.
- Recovery must ensure *atomicity* by undoing uncommitted transactions and *durability* by redoing committed transactions after failure.
- Recovery makes use of log files to restore the database.

Checkpoint

- A synchronisation point between the database and log file. All buffers are forced-written to secondary storage.

- Involves the following operations:
  - Write all log records to secondary storage.
  - Write contents of buffers to secondary storage.
  - Write a checkpoint record to the log file. This includes a list of all transactions in progress.

Using Checkpoints



Recovery technique

- Write ahead to the log file before the database as follows:
  - When transaction starts, register in log.
  - When write operation performed, write data record to log.
  - Once the log record is written, write the update to the database buffers.
  - The updates to the database itself are written when the buffers are next flushed to secondary storage.
  - When the transaction commits, register in log.
- Recovery uses the log to undo and redo transactions.

Summary

- Transactions and transaction control.
  - function, properties, COMMIT/ROLLBACK protocol, log file.
- The problems of transaction concurrency and how they can be controlled.
  - three common problems, two phase locking & deadlock
- The recovery of lost transactions.
  - checkpoint & write-ahead log protocol

Further Reading

- ROLLAND, F.D. (1998) *The Essence of Databases*, London, Prentice-Hall, Chapter 9.

- DATE, C.J. (2004) *An Introduction to Database Systems*, 8th ed., Reading, Addison-Wesley, Chapters 15 & 16.

- CONNOLLY, T.M. & BEGG, C.E. (2010) *Database Systems*, 5th. ed., Reading, Addison-Wesley, Chapter 20.