SYSTEMS IMPLEMENTATION TECHNIQUES

- TRANSACTION PROCESSING
- DATABASE RECOVERY
- DATABASE SECURITY
- CONCURRENCY CONTROL

TRANSACTION PROCESSING

- Def:
  A Transaction is a program unit (deletion, creation, updating etc) whose execution preserves the consistency of a database.

TRANSACTION PROCESSING

- To ensure that the above is met a transaction must be
  - Atomic
  - Execute to completion
  - Not execute at all

PROPERTIES OF A TRANSACTION

- 4 Basic properties
- Also known as the ACID properties
  - Atomicity
  - Consistency
  - Isolation
  - Durability

PROPERTIES OF A TRANSACTION

- atomicity
  - Also known as the all nothing property
  - A transaction is an individual unit that is either performed in its entirety or not performed at all.

PROPERTIES OF A TRANSACTION

- Consistency
  - The transaction must transform the database from one consistency state to another consistency state

PROPERTIES OF A TRANSACTION

- Isolation
  - Transactions execute independently of one another.
- Durability
  - The effects of a successfully completed transaction are permanently recorded in the database and must never be lost due to subsequent failure

Transactions Management

- Scenario:
- transferring money from one account to another in one bank requires the SQL commands:

```
UPDATE ACCOUNTS
SET BALANCE = BALANCE - 100
WHERE ACCOUNT = 1234;
```

```
UPDATE ACCOUNTS
SET BALANCE = BALANCE + 100
WHERE ACCOUNT = 4567;
```

Transactions Management

- Transferring £100 from account 1234 to Account 4567.
- Together they comprise a **single** transaction.
- Potential problem:
  - Database crash !!

Transaction integrity

- Crash may leave the Database in inconsistent state
- in the example, it would be better if neither of the commands had been executed.
- Transaction integrity
  - demands that the effects of a transaction should be either complete or not enacted at all.

Commit/Rollback protocol

- exist to support transaction integrity
- Commit
  - is when changes to a Database are made permanent
  - when a Database crashes, any changes that have not been committed will be lost

Commit

- We can issue an explicit commit command when both of these update commands have been issued

```
UPDATE ACCOUNTS
SET BALANCE = BALANCE - 100
WHERE ACCOUNT = 1234;

UPDATE ACCOUNTS
SET BALANCE = BALANCE + 100
WHERE ACCOUNT = 4567;
COMMIT;
```

Rollback

- a mechanism to undo the effects of a transaction.
- when issued all of the Database changes since last commit are undone.

```
1. Select name from customers where refno = 1; {returns 'P Abdul'}
2. Update customers set name = 'J Jones' where refno =1;
3. Select name from customers where refno = 1; {returns 'J Jones'}
4. ROLLBACK;
5. Select name from customers where refno = 1; {returns 'P Abdul'}
```

Rollback

- the Rollback in 4 undoes the effect of the Update in 2
- because the Update has not been committed
- – Suppose we issue a Commit command

Commit/Rollback

```
1. Select name from customers where refno = 1; {returns 'P Abdul'}
2. Update customers set name = 'J Jones' where refno =1;
3. COMMIT ;
4. Select name from customers where refno = 1; {returns 'J Jones'}
5. ROLLBACK ;
6. Select name from customers where refno = 1; {returns 'J Jones'}
```

- The Commit command in 3 makes the change permanent

DATABASE RECOVERY

- Def:
  - This is the process of restoring the database to a consistency state after a failure .

DATABASE RECOVERY

- Types of failure
  - System failure – system entering an undesirable state, like an infinite loop or deadlock.
  - Logic Errors – Bad programmes
  - Hardware failures

Recovery Facilities

- The DBMS provides the following facilities to recover from failure.
  - Backup Mechanism -: Periodical backup of the system
  - Logging Facility -: Keeps truck of the current state of the transaction and the database

Recovery Facilities

- Checkpoint Facility -: enables update to the database to be made permanent
- Recovery manager -: Allows the system t o restore the database to a consistency state following a failure

Recovery Techniques

- Deferred updates :
  - This were you use a log to record all new transactions and the log will be used to update the database at a later stage.

Recovery Techniques

- Immediate updates :
  - This is where updates are made to the records immediately and the update is kept in both the log and the database

Recovery Techniques

- Shadow Paging
  - Two page tables are maintained during the life of a transaction
  - The current page and the shadow page
  - When the transaction starts the two tables are the same

Recovery Techniques

- The shadow page is not changed and is used to restore the database in the event of a failure
- The current page is used to record all updates to the database
- When the transaction completes the current page becomes the shadow page and the shadow page is garbage collected.

DATABASE SECURITY

- DEF:
  - Mechanism that protects the database against intentional or accidental threats.
  - It encompasses hardware , software , people and data

DATABASE SECURITY

- It is considered in relation to the following situations:
  - Theft
  - Loss of Confidentiality
  - Loss of privacy
  - Loss of Integrity
  - Loss of Availability

THEFT/FRAUD

- This the acquisition of data illegally

Confidentiality

- Refers to the need to maintain secrecy over the data usually that which is critical to an organization

Privacy

- Refers to the need to protect data about individuals , loss would lead to legal action taken against the organization

Integrity

- Loss results in invalid and corrupted data

Availability

- Data must be available to authorized persons at an appropriate time (when as required)
- Loss leads to the inability to access data.

Database Security

- Measures that can be used to safeguard databases from anticipated threats
  - Authorization
  - Authentication
  - Views or subschema
  - encryption

Authentication

- Mechanisms that determines whether a user is s/he what s/he claims to be
- Establishing proof of identity
  - Physical traits
  - Pin codes
  - Cards etc

Authorization

- Also known as Access control
  - This is the granting of rights and privileges that enables a user to have access to the system

Views or subschema

- A view is a virtual table that does not exist in the database but is produced upon request by particular user

Encryption

- This is the encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key.

CONCURRENCY CONTROL

- Concurrency
  - The process describing two or more users accessing the database at the same time and transactions are interleaved.
  - Undesirable results may occur, hence the need for concurrency control

Concurrency problems

- **The Lost Update Problem**
- • The following situation might arise:
  - **1) TA reads Account record 1234. Value of balance is 150.**
  - **2) TB reads Account record 1234. Value of balance is 150.**
  - **3) TA increases to 250 (150+100).**

The Lost Update Problem

- **4) TB increases to 350 (150+200).**
- **5) TA writes back balance of 250.**
- **6) TB writes back balance of 350.**
- • The account should have a balance of 450, not 350.
- • The update performed by **TA** has been lost

The uncommitted dependency

- • When does it occur?
  - Another transaction may start using data that has not yet been committed.
  - – Effects: the 2nd transaction will use false information.

The uncommitted dependency

- – Example
- **TA**
- **Update Accounts**
- **Set Balance = Balance - 100**
- **Where Accno = 1234;**
- **If Balance < 0.00 Then Rollback Else Commit;**
- **TB**
- **Delete from Accounts**
- **Where Balance < 0.00;**

The uncommitted dependency

- • Example
  - **1) TA retrieves Account 1234. Value of balance is 50.**
  - **2) TA reduces balance by 100. Leaving it as -50.**
  - **3) TA writes back value of -50.**

uncommitted dependency

- **4) TB retrieves Account 1234. Balance is -50.**
- **5) TB deletes Account 1234 as it has negative balance.**
- **6) TA rolls back update. Too late! the account has been deleted**
- TB used uncommitted data.

Inconsistent Analysis

- A transaction accesses records while are they being updated by another transaction.
- Example
  - 2nd transaction transfers money from one account to another.
  - Hence, should have no effect on **TA** result.

Inconsistent Analysis

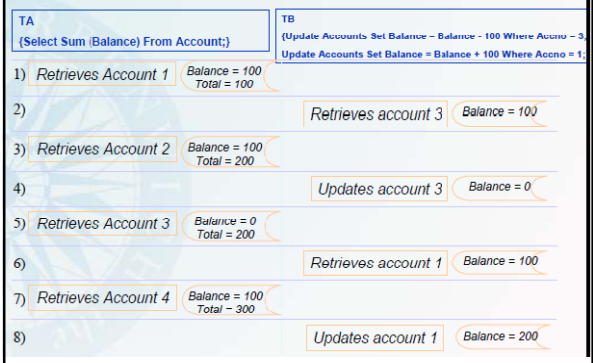
- TA
- Select Sum (Balance)
- From Account;

- **TB**
- **Update Accounts**
- **Set Balance = Balance - 100 Where Accno = 3;**
- **Update Accounts**
- **Set Balance = Balance + 100 Where Accno = 1;**

Inconsistent Analysis

- 2nd transaction transfers money from one account to another.
  - Hence, should have no effect on **TA** result.

Inconsistent Analysis



Locking

- How to avoid all previous problems?
  - ***Lock*** the object to prevent access by other transactions
  - A transaction releases the object when it finishes with it
  - Other transactions need to queue until the object is released
- The lock could be shared or exclusive

Shared Locks

- A Shared lock ***S*** is placed on an object that is being accessed for read only purposes
  - many S locks may be placed
  - an X lock must wait

Exclusive Locks

- An exclusive lock ***X***, when an object is being altered
  - No other lock may be placed
  - All transactions must wait

The Locking Protocol

- Relate this to SQL:
- Many 'read-only' operations (e.g. Select)
- One 'update' operation (e.g. Delete)

The Locking Protocol

- The Lost Update Problem:
  - TA will place an X lock on Account 1234 before it starts update

The Locking Protocol

- The uncommitted dependency:
  - TA will lock TB out from Account 1234 until it has completed the rollback

The Locking Protocol

- The inconsistent analysis:
  - TA will place an S lock on all of the account records.

Problems with Locking

- Appropriate locking can guarantee correctness, However, it also introduces potential undesirable effects:
- Deadlock
  - No transactions can proceed; each waiting on lock held by another.

- Starvation
  - One transaction is permanently "frozen out" of access to data.
  - reduced performance
- Locking introduces delays while waiting for locks to be released.

Two-Phase Locking

- A transaction follows a 2 phase locking protocol if all operations precede the first unlock operations in the transaction.
- According to this protocol every transaction can be divided into two phases

Two-Phase Locking

- Growing phase
  - A transaction acquires all the locks needed but can not release any locks
- Shrinking Phase
  - A transaction releases its locks but cannot acquire any locks

Two-Phase Locking

	TA: S lock on Obj1; S on Obj2; X on Obj1	TB: S lock on Obj1; X on Obj1; S on Obj2
1)	Place X on Obj1 — *Request X not S*	
2)		~~Place X on Obj1~~ — *Requests denied, Wait*
3)	Place S on Obj2	*Wait*
4)	Release all locks	*Wait*
5)		Place X on Obj1
6)		Place S on Obj2
7)		Release all locks

- Acquiring locks phase & releasing locks phase
- Two-Phase locking