WHERE ARE WE NOW?

W34: Introduction, DBMSs and Relational Databases
W35: Developing Database Systems
W36: SQL –Part I
W37: SQL –Part II and Relational Algebra
W38: Data Modelling
W39: Data Modelling
W40: Database Design
W41: Normalisation and Stored Procedures
W42: XML and Web Technology
W43: Processing XML Data
W44: XML Validation
W45: Beyond relational databases and XML
W46: File Organisations and Indexes
W47: Database Security and Administration
W48: Transaction Processing and Wrap-up

GOAL

- Monday:
  - *PHP/PDO revisit*
  - *Security in web database application*
  - *Database privileges*
- Today:
  - *Database administration*
  - *Introduction to database transactions*
  - *Database backup and recovery*

CONTENT

- Data and database administration
- Introduction to transactions
- Database recovery:
  - *The log file*
  - *Recovery techniques*
  - *Database backup*

DATA ADMINISTRATION

- The management of the data resource:
    - *Database planning*
    - *Development and maintenance of standards, policies, and procedures*
    - *Conceptual and logical design*

DATABASE ADMINISTRATION

- The management of the physical database:
  - *Physical database design and implementation*
  - *Implementing security and integrity controls*
  - *Monitor and tune system performance*
  - *Perform backups routinely*
  - *Ensuring that recovery mechanisms and procedures are in place*
  - *Keeping system HW and SW up to date*

TRANSACTIONS

- Transaction:
  - *Series of actions which reads or updates database contents*
- Logical unit of work on the database
- Application program is series of transactions:
  - *Usually mixed with non-database operations*
- Transform database from one consistent state to another
  - *Consistency may be violated during transaction*

TRANSACTION SUPPORT

- Can have one of two outcomes:
  - *Success - transaction commits*
    - » Database reaches a new consistent state
  - *Failure - transaction aborts*
    - » Database must be restored to its previous consistent state,
    - » the transaction is rolled back (undone)
- Committed transaction cannot be aborted
- Aborted transactions that are rolled back can be restarted later

PROPERTIES OF TRANSACTIONS

- Four basic (**ACID**) properties of a transaction are:
  - ***A****tomicity*
    - ‘All or nothing’ property
  - ***C****onsistency*
    - Must transform database from one consistent state to another
  - ***I****solation*
    - Partial effects of incomplete transactions should not be visible to other transactions
  - ***D****urability*
    - Effects of a committed transaction are permanent and must not be lost because of later failure

TRANSACTIONS IN MySQL

- By default, MySQL runs with `autocommit` mode enabled:
  - *MySQL stores the update on disk immediately*
  - *The change cannot be rolled back*

- With `START TRANSACTION`, `autocommit` is disabled
  - *until end of transaction (i.e., next `COMMIT` or `ROLLBACK`)*
  - *`autocommit` mode then reverts to its previous state.*

- Statements that cause implicit commit and thus cannot be rolled back:
  - *DDL statements:*
    - » `CREATE/DROP/ALTER TABLE`
    - » …

A NUT TO CRACK

- Can you think of cases in which rollback is not initiated by the end-user application?

CONTENT

- Data and database administration
- Introduction to transactions
- Database recovery:
  - *The log file*
  - *Recovery techniques*
  - *Database backup*

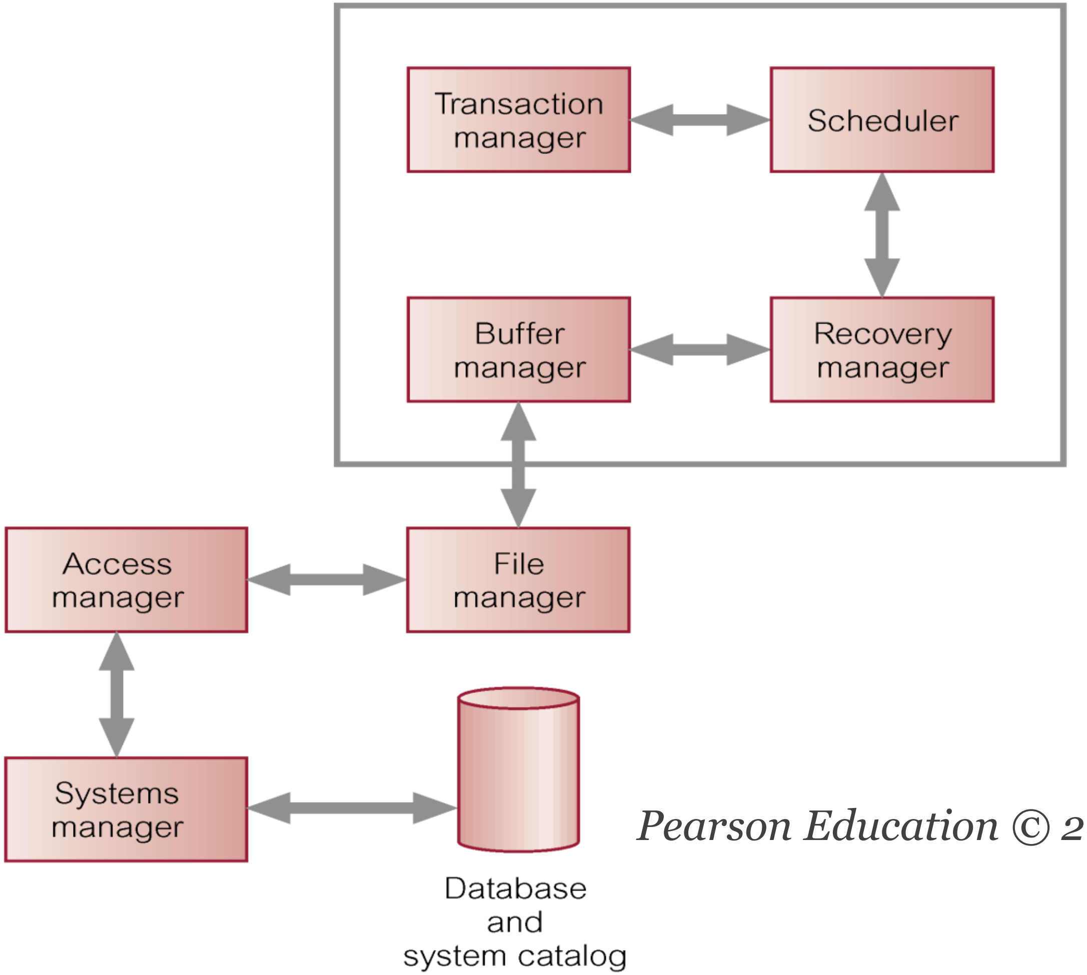
DATABASE RECOVERY

- Process of restoring database to a correct state after a failure
- Need for recovery control:
  - *Two types of storage: volatile (main memory) and nonvolatile*
  - *Volatile storage does not survive system crashes*
  - *Stable storage:*
    - » When information has been replicated in several nonvolatile storage media with independent failure modes

TYPES OF FAILURES

- System crashes, resulting in loss of main memory
- Media failures, resulting in loss of parts of secondary storage
- Application software errors
- Natural physical disasters
- Carelessness or unintentional destruction of data or facilities
- Sabotage

DBMS TRANSACTION SUBSYSTEM



Pearson Education © 2009

WHEN IS A FILE ACTUALLY WRITTEN TO DISK?

- File systems store data in internal buffers to optimise disk utilization

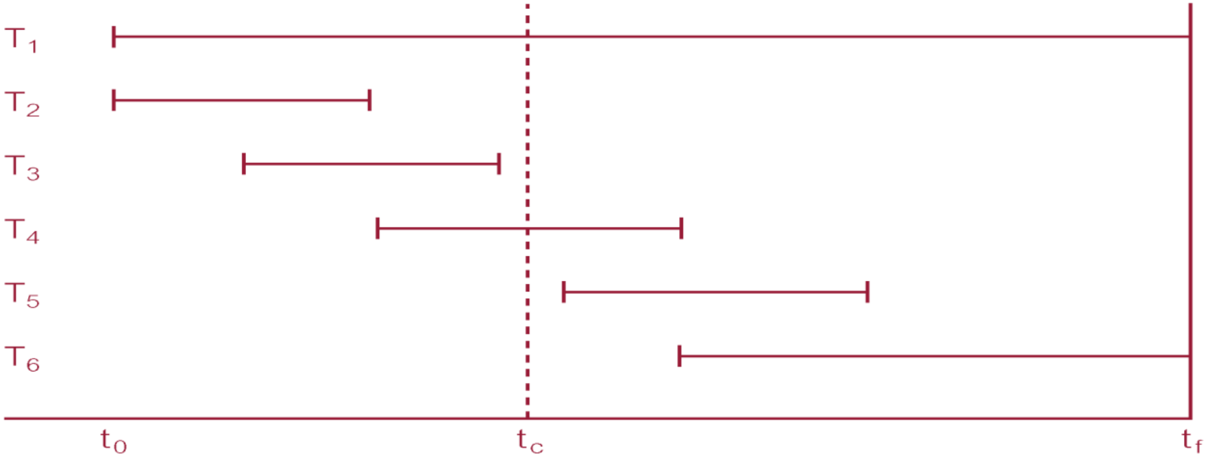
- File write operation divided in to steps:
  1. *Data buffered updated immediately when file is written to*
  2. *The file system writes the data to the disk "on a regular basis"*

- An application can request a flush of data buffers:
  - *forcing the operating system to write the contents of the data buffers to the disk*

TRANSACTIONS AND RECOVERY

- Transactions represent basic unit of recovery
- Recovery manager responsible for atomicity and durability
- Failure occurs between commit and database buffers being flushed to secondary storage:
  - *Recovery manager has to redo (rollforward) the transaction's updates*
- If transaction had not committed at failure time:
  - *Recovery manager has to undo (rollback) any effects of that transaction for atomicity*

EXAMPLE



Pearson Education © 2009

- DBMS starts at time t_0, but fails at time t_f
- Assume:
  - *Data for transactions T_2 and T_3 have been written to secondary storage buffers only*
- Then:
  - *T_1 and T_6 have to be undone*
  - *Recovery manager has to redo T_2, T_3, T_4, and T_5*

RECOVERY FACILITIES

- Backup mechanisms:
  - *Make periodic backup copies of database*
- Logging facilities:
  - *Keep track of current state of transactions and database changes*
- Checkpoint facility:
  - *Enables updates to database in progress to be made permanent*

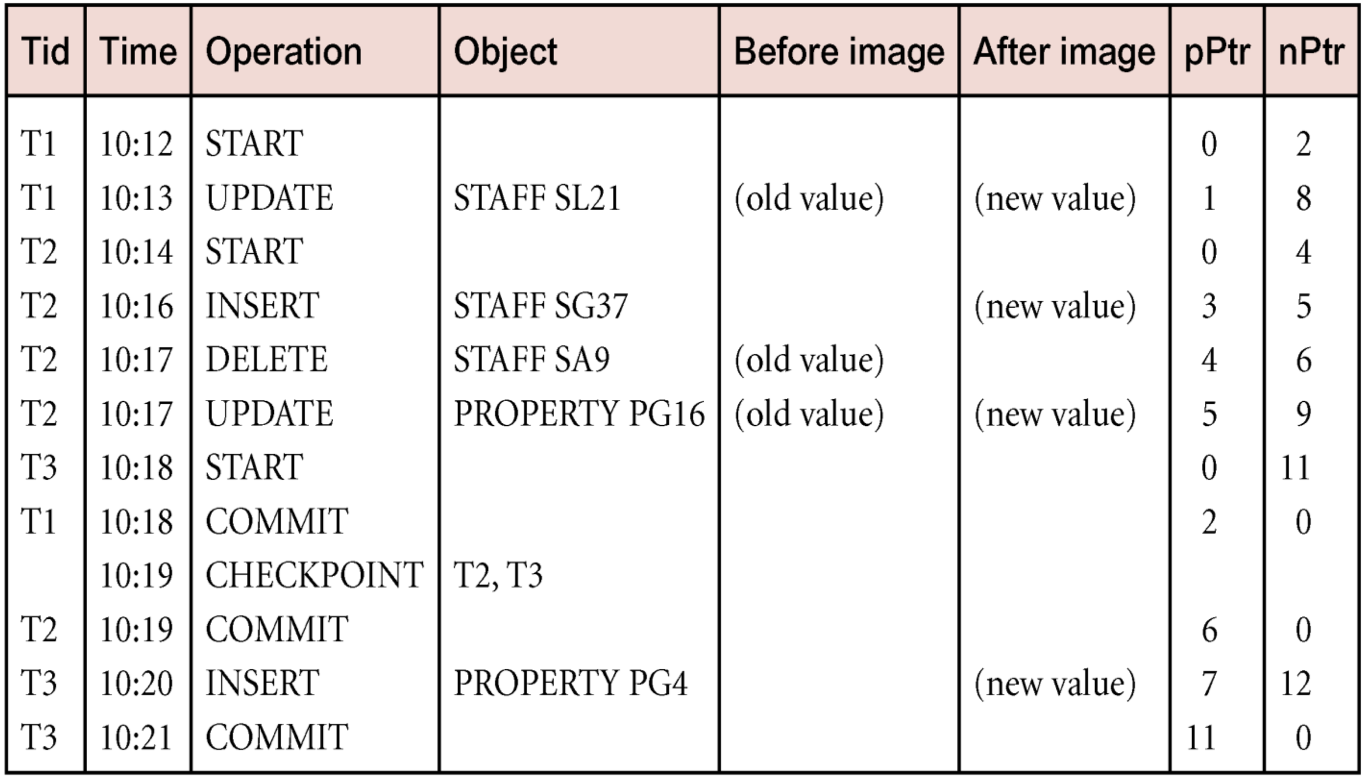
LOG FILE

- Contains information about all updates to database:
  - *Transaction records*
  - *Checkpoint records*
- Potential bottleneck
  - *Critical in determining overall performance*

LOG FILE - TRANSACTION RECORD

- Transaction records contain:
  - *Transaction identifier*
  - *Type of log record (transaction start, insert, update, delete, abort, commit)*
  - *Identifier of data item affected by database action*
  - *Before-image of data item*
  - *After-image of data item*
  - *Log management information*

SAMPLE LOG FILE



Tid	Time	Operation	Object	Before image	After image	pPtr	nPtr
T1	10:12	START				0	2
T1	10:13	UPDATE	STAFF SL21	(old value)	(new value)	1	8
T2	10:14	START				0	4
T2	10:16	INSERT	STAFF SG37		(new value)	3	5
T2	10:17	DELETE	STAFF SA9	(old value)		4	6
T2	10:17	UPDATE	PROPERTY PG16	(old value)	(new value)	5	9
T3	10:18	START				0	11
T1	10:18	COMMIT				2	0
	10:19	CHECKPOINT	T2, T3				
T2	10:19	COMMIT				6	0
T3	10:20	INSERT	PROPERTY PG4		(new value)	7	12
T3	10:21	COMMIT				11	0

Pearson Education © 2009

CHECK-POINTING

- Checkpoint:
  - *Point of synchronization between database and log file*
  - *All buffers are force-written to secondary storage*
- Checkpoint records contain identifiers of all active transactions
- When failure occurs:
  - *Redo all transactions that committed since the checkpoint*
  - *Undo all transactions active at time of crash.*
  - *In previous example, with checkpoint at time t_c, changes made by T_2 and T_3 have been written to secondary storage*
    - » only redo T_4 and T_5
    - » undo transactions T_1 and T_6

A NUT TO CRACK

- Which operation is faster:
  - *Adding the update event to the log, or*
  - *Updating the database itself?*

RECOVERY ACTIONS

- If database has been damaged:
  - *Need to restore last backup copy of database and*
  - *reapply updates of committed transactions using log file.*
- If database is only inconsistent:
  - *Undo changes that caused inconsistency.*
  - *Redo transactions not yet written to secondary storage*
  - *Do not need to load backup files*
    - » Restore database using before- and after-images in the log file

RECOVERY TECHNIQUES

- Three main recovery techniques:
  - *Deferred Update*
  - *Immediate Update*
  - *Shadow Paging*

DEFERRED UPDATE

- Updates are written to log as they occur
    - *Database files updated only after commit*
- Need to redo updates of committed transactions not yet written to disk in the case of failure
- No updates to undo

IMMEDIATE UPDATE

- Updates are applied to database files as they occur
- Need to redo updates of committed transactions
- Need to undo effects of transactions that had not committed
  - *Undo operations are performed in reverse order*
- Essential that log records are written before write to database
  - *Write-ahead log protocol*
- If no “transaction commit” record in log:
  - *The transaction was active at failure and must be undone*

SHADOW PAGING

- Maintain two page tables during life of a transaction
  - *Current page table*
  - *Shadow page table*
- When transaction starts, two pages are the same
- During transaction:
  - *Shadow page table is kept in original state*
  - *Current page table records all updates to database*
- When transaction completes
  - *Current page table becomes shadow page table*
- On recovery:
  - *Shadow page table restored as current table*

A NUT TO CRACK

- Given the stable storage "definition":
  - *When information has been replicated in several nonvolatile storage media with independent failure modes*
- Can a RAID system be considered stable storage?

CONTENT

- Data and database administration
- Introduction to transactions
- Database recovery:
  - *The log file*
  - *Recovery techniques*
  - *Database backup*

A NUT TO CRACK

- The database is stored in files on the file system.
  - *Why cannot we simply back up all database files as we back up every other file?*

MySQL BACKUP TECHNIQUES (1)

- HotBackup:
  - *MySQL Entreprise Backup*
  - *Database backed up while database is running*
- mysqldump:
  - *Dumps databases to file as SQL statements*
- Backing up table files:
  - *Backs up raw files*
  - *Database needs to stopped*
  - *Tables need to be locked for read*
  - *Buffers need to be flushed*

MySQL BACKUP TECHNIQUES (2)

- Incremental Backups by Enabling the Binary Log:
  - *Copy to the backup location all binary logs created since last full backup*
- …

FULL VS INCREMENTAL BACKUPS

- Full backups are necessary, but it is not always convenient to create them:
  - *Produce large backup files*
  - *Take time to generate*
  - *The database may not have changed much since last backup*
- Incremental backups may be more efficient:
  - *Incremental backups are smaller*
  - *Take less time to produce.*
  - *More time and processing required during recovery:*
    - » Restore previous full backup
    - » Apply incremental backups one-by-one

ESTABLISHING A BACKUP POLICY

- Backups must be scheduled regularly
- A mix of full and incremental backups is usually best, e.g.,:
  - *Full backup once a week*
  - *Incremental backup once a day between the full backups*

RESOURCES

- C&B 20.6
- MySQL Reference Manual, 7. Backup and Recovery
- MySQL Reference Manual, 5.2.4 The Binary Log
- MySQL Reference Manual, 13.3 MySQL Transactional and Locking Statements