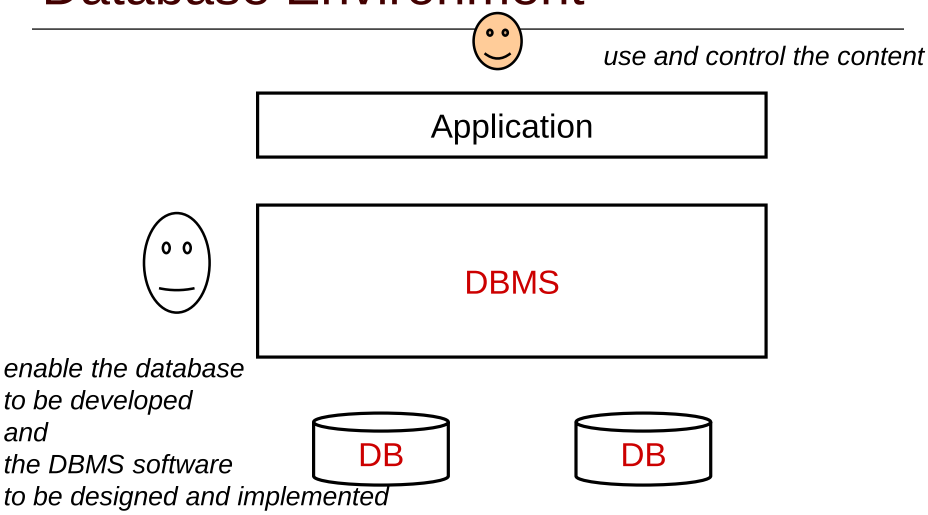
Database
(some reminders)

Vu Tuyet Trinh
trinhvt@soict.hust.edu.vn

Department of Information Systems
School of Information Technology and Communication
Hanoi University of Science andTechnology

Database Environment



Database

A shared collection of related data designed to meet the information needs of an organisation

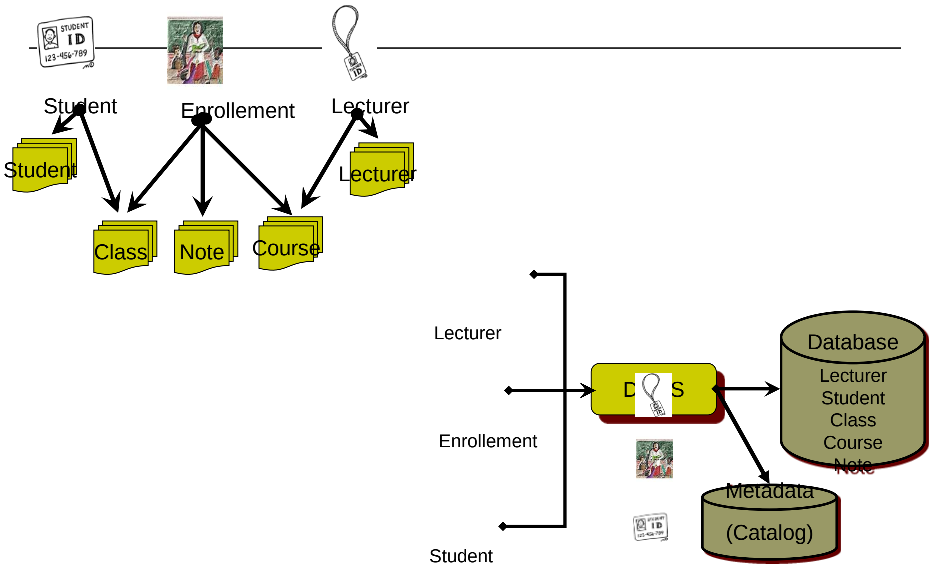
- Logically coherent
- Internally consistent
- Specific purpose
- Representation of the real world

Database Management System

A software to facilitate the creation and maintenance of a database

- Defining ~ specifying *types* of data
- Constructing ~ storing & populating
- Manipulating ~ querying, updating, reporting

File vs. Database



Database Advantages

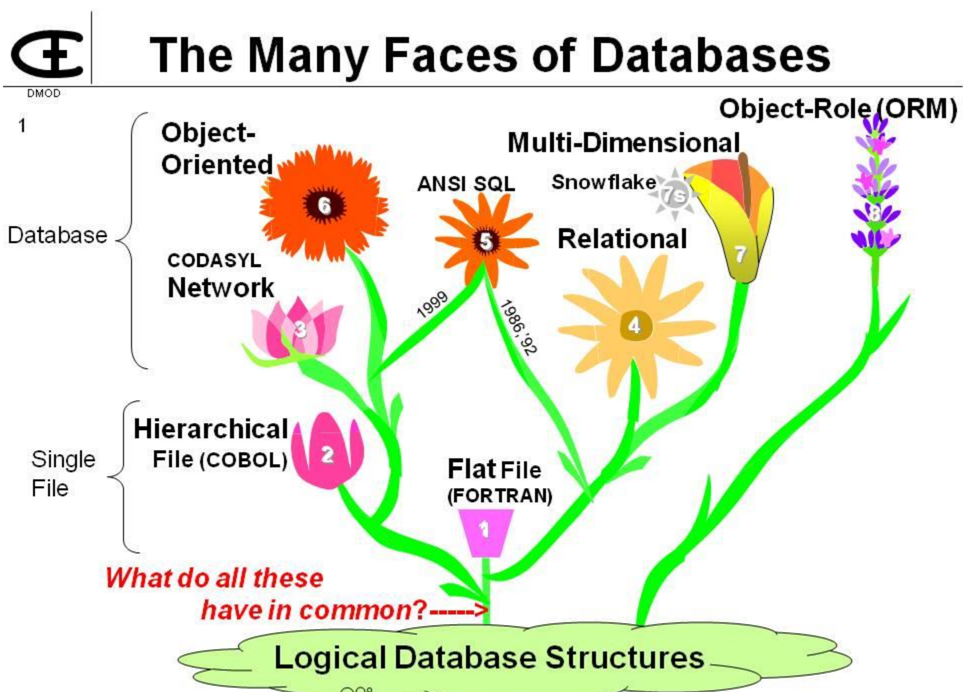
- Controlled redundancy
  - consistency of data & integrity constraints
- Integration of data
  - self-contained & represents semantics of application
- Data and operation sharing
  - multiple interfaces
- Flexibility
  - data independence
  - data accessibility
  - reduced program maintenance
- Services & Controls
  - security & privacy controls
  - backup & recovery
  - enforcement of standards
- Ease of application development

Data Models

- a set of concepts used to describe the database structure
  - data types
  - constraints
- Some existing database models
  - Hierarchical model
  - Network model
  - Relational model
  - Object-Oriented model

"More than 90% of current database applications are built on relational database systems which uses relational model as its underlying data model"*

* R. Elmasri and S. Navathe. Fundamentals of Database Systems



(c) Gordon C. Everest

Model vs. Schema vs. Instance

- Data Model
  - set of concepts used to describe the structure of a database: data types, relationships, constraints, semantics, …
  - tool for data abstraction
- Schema
  - data structure fulfilled all features of the parts of the real world which is of interest to the users
- Instance
  - Data itself

Example

- Data Model

```
type <type_name> = record
    <field_name> : <data_type>;
    <field_name> : <data_type>;
    …
end;
```

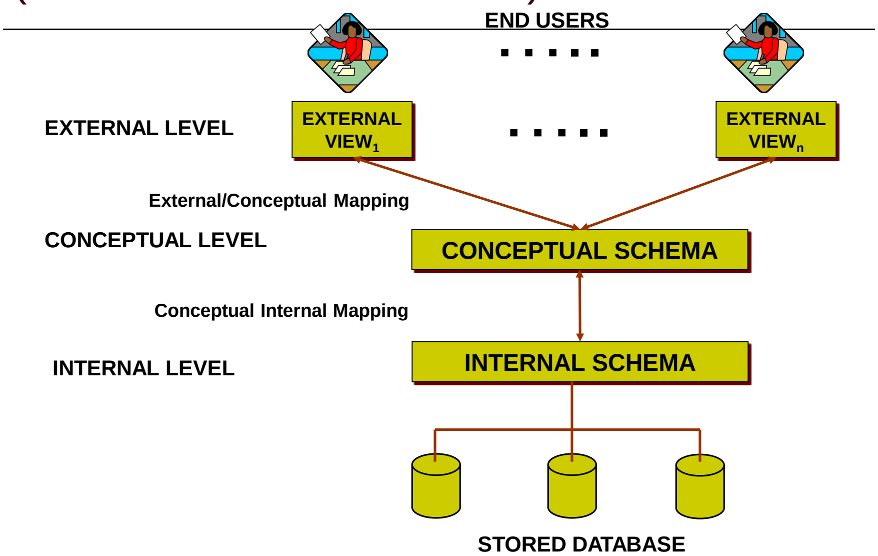
- Schema

```
type student = record
    ID : string;
    fullName: string;
    Birthday: date;
    Address: string ;
    Class: string;
end;
```

- Instance

(« Stud001 », « Nguyen », 1/4/1983, «1 Dai Co Viet », « 1F VN K50 »)

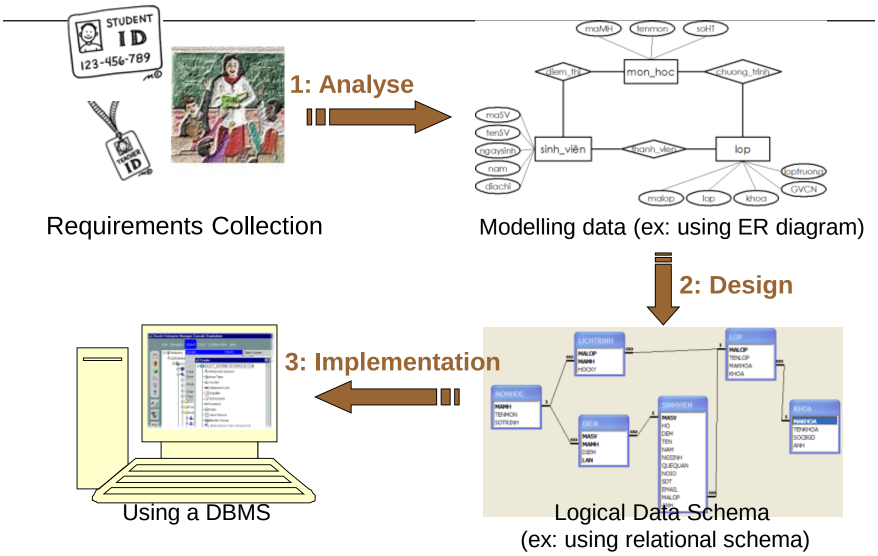
3-tier Schema Model (ANSI-SPARC Architecture)



Database Design

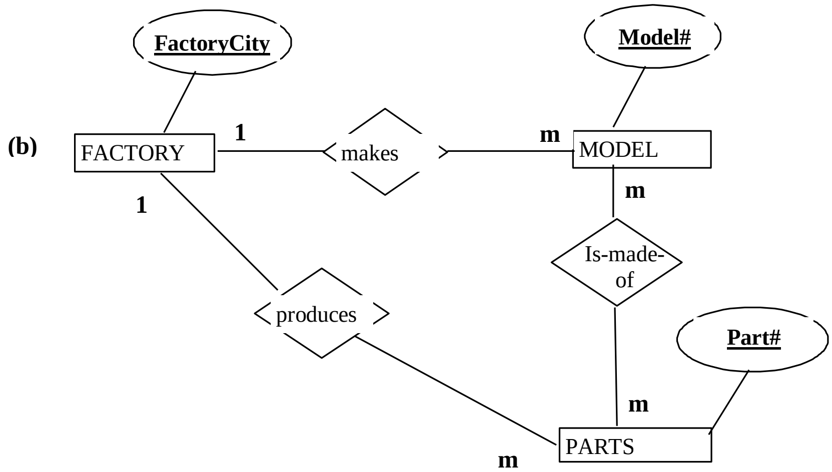
- Extended Entity Relationship
  - Top Down
  - Conceptual/Abstract View
- Functional Dependencies
  - Bottom Up
  - Implementation View
  - Synthesise relations

Process of Database Design



ER Model

- wellsuited to data modelling for use with databases
  - easy to represent and explain
  - readily translated to relations.
- Basic concepts
  - Attribute
    - represent a property/characteristic of an object in real world
  - Entity
    - defined as a set of attributes
  - Entity Set
    - Set of all entity instances of the same entity type
  - Relationship Set
    - Set of all relationship instances of the same relationship type



studno	name	tutor	roomno	courseno	labmark	subject
s1	jones	bush	2.26	cs250	65	prog
s1	jones	bush	2.26	cs260	80	graphics
s1	jones	wibby	2.26	cs270	47	elecs
s2	brown	kahn	IT206	cs250	67	prog
s2	brown	kahn	IT206	cs270	65	elecs
s3	smith	goble	2.82	cs270	49	comms
s4	blogg	goble	2.82	cs280	50	design
s5	jones	zobel	2.34	cs250	0	prog
s6	peters	kahn	A17	cs250	2	prog
null	*null*	capon	A14	*null*	*null*	*null*
null	*null*	*null*	*null*	cs290	*null*	specs

F
studno → name, tutor
tutor → roomno
roomno → tutor
courseno → subject
studno, courseno → labmark

F+
studno, courseno → name *partial*

studno → roomno *transitive*

Using functional dependencies to…Synthesise relations

studno → studno
studno → familyname
studno → givenname
studno → hons
studno → tutor
studno → slot
studno → year

STUDENT
(studno,givenname,familyname,hons,tutor,slot,year)

studno, courseno → labmark
studno, courseno → exammark

ENROL(studno,courseno,labmark,exammark)

courseno → courseno
courseno → subject
courseno → equip

COURSE(courseno,subject,equip)

lecturer → lecturer
lecturer → roomno
lecturer → appraiser
roomno → lecturer
roomno → roomno
roomno → appraiser

STAFF(lecturer,*roomno*,appraiser)
(L01, R01, 3)
(L02,R01, 4)

hons → faculty
hons → hons

year → year
year → yeartutor
yeartutor → year
yeartutor → yeartutor

YEAR(year,*yeartutor*)

SCHOOL(hons,faculty)

STUDENT_DETAILS
(studno, name, tutor, roomno, {courseno, labmark, subject})
studno → name, tutor
tutor → roomno, roomno → tutor
courseno → subject
studno, courseno → labmark

STUDENT
(studno, name, tutor, roomno)
studno → name, tutor
tutor → roomno,
roomno → tutor

ENROL (studno, courseno, subject, labmark)
courseno → subject
studno, courseno → labmark

COURSE (courseno, subject)
courseno → subject

ENROL' (studno, courseno, labmark)
studno, courseno → labmark

STUDENT (studno, name, tutor)
studno → name, tutor

TUTOR (tutor, roomno)
tutor → roomno
roomno → tutor

Languages of DBMS

- Data Definition Language (DDL)
  - define the logical schema (relations, views, …) and storage schema stored in a Data Dictionary
- Data Manipulation Language (DML)
  - Manipulative populate schema, update database
  - Retrieval querying content of a database
- Data Control Language (DCL)
  - permissions, access control, ...

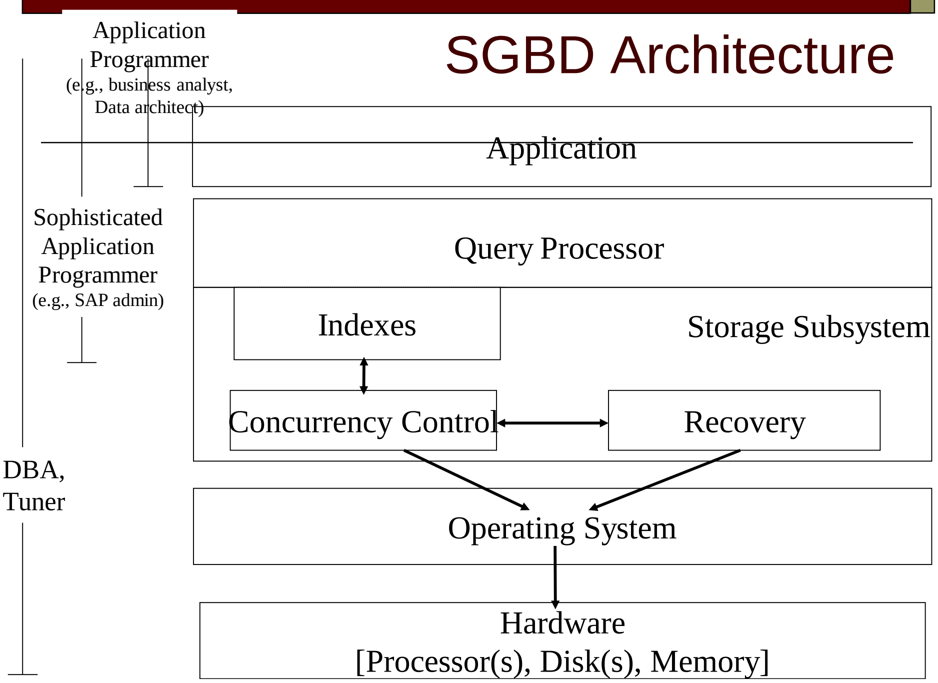
Data Manipulation Language

- Structured Query Language (SQL)
- A brief history
  - SQL 1
    - The first standard for SQL defined in 1986
    - adopted as an international by Standards Organisation (ISO) in 1987
  - SQL2
    - revised version of the processor (also called SQL 92).
    - adopted as the formal standard language for defining and manipulating relational database.
  - SQL 3
    - extension with additional features such as user-defined data types, triggers, user-defined functions and other Object Oriented features

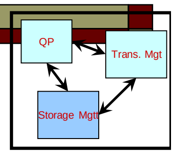
SQL Retrieval Statement

```
SELECT[all|distinct]
      {*|{table.*|expr[alias]|view.*}
      [,{table.*|expr[alias]}]...}
FROM table [alias][,table[alias]] ...
[WHERE condition]
[GROUP BY expr [,expr] ...]
[HAVING condition]
[{UNION|UNION ALL|INTERSECT|MINUS}
      SELECT ...]
[ORDER BY {expr|position} [ASC|DESC]
         [,expr|position}[ASC|DESC].
```

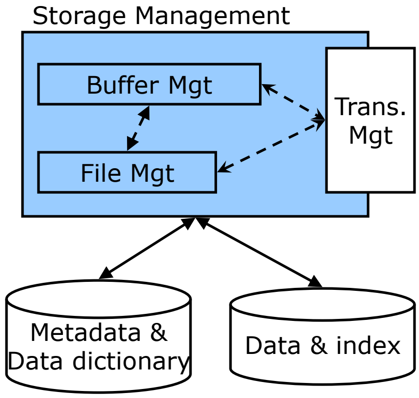
SGBD Architecture



Storage Management



- Responsible for storing and accessing data.
- Buffer manager
  - responsible for partitioning the available main memory into buffers
- File Management
  - responsible for interacting with file system



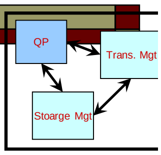
Indexing technique

- Search key
  - Any subset of the fields of a relation can be the search key
  - *Search key* may not be the *key* in relation
- Index
  - a collection of k *data entries*
  - supports efficient retrieval with a given key value **k**.

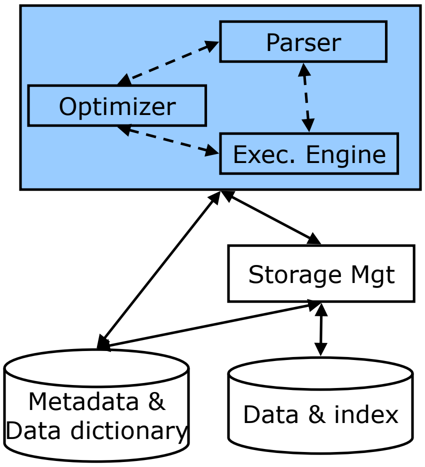
Classes of Indexes

- Primary vs. secondary: primary has primary key
- Clustered vs. unclustered: order of records and index approximately same
  - Alternative 1 implies clustered, but not vice-versa
  - A file can be clustered on at most one search key
- Dense vs. Sparse: dense has index entry per data value; sparse may "skip" some
  - Alternative 1 always leads to dense index
  - Every sparse index is clustered!
  - Sparse indexes are smaller;
    however, some useful optimizations are based on dense indexes

Query Processing



- Parser
  - verifying query syntax and semantic
- Optimizer
  - responsible for performing query plan transformation for the best evaluation
- Execution engine
  - Responsible for executing each of steps in the chosen query plan.
  - interacts with most of the other components of the DBMS



Relational Algebra Operations

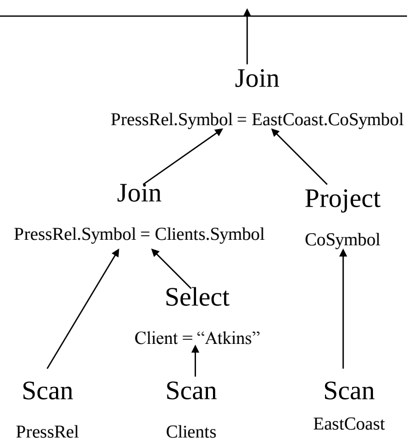
- Projection
- Selection
- Join
- Division

- Union
- Intersection
- Difference
- Cartesian product

Query Plans

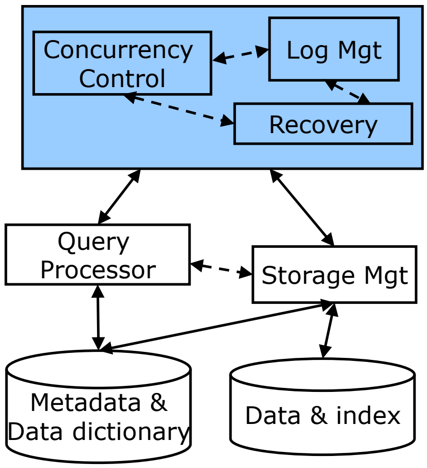
- Data-flow graph of relational algebra operators
- *Typically:* determined by optimizer

```
SELECT *
FROM PressRel p, Clients C
WHERE p.Symbol = c.Symbol
 AND c.Client = 'Atkins'
 AND c.Symbol IN
(SELECT CoSymbol FROM EastCoast)
```



Transaction Management

- Ensure ACID properties
- The transaction processor performs the following tasks:
  - Logging
  - Recovery
  - Concurrency control



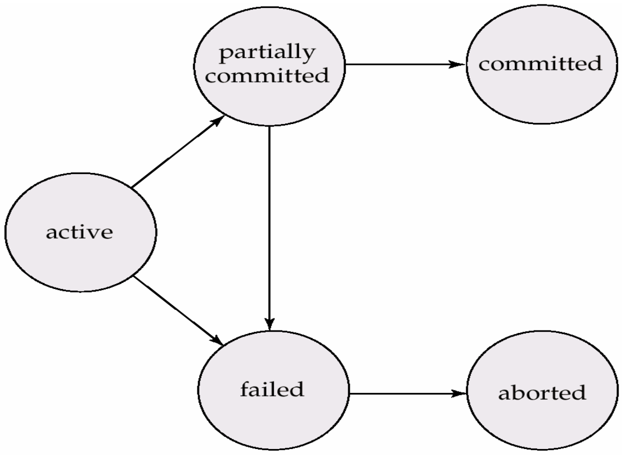
Transaction

- A sequence of read and write operations on data items that logically functions as one unit of work
  - Assuring data integrity and correction
- ACID Properties
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability

Concurrency Control

Recovery

Transaction States



Transaction Management Interfaces

- Begin Trans
- Commit ()
- Abort()

- Savepoint Save()
- Rollback (savepoint)
  (savepoint = 0 ==> Abort)

Concurrency Control

- Objective:
  - ensures that database transactions are performed concurrently without the concurrency violating the data integrity
  - guarantees that no effect of committed transactions is lost, and no effect of aborted (rolled back) transactions remains in the related database.
- Example

T0: read(A);
A := A -50;
write(A);
read(B);
B := B + 50;
write(B);
	T1: read(A);
	temp := A *0.1;
	A := A -temp;
	write(A);
	read(B);
	B := B + temp;
	write(B);

Scheduling

T_0	T_1
read(A)	
A := A - 50	
write(A)	
read(B)	
B := B + 50	
write(B)	
	read(A)
	temp := A * 0.1
	A := A -temp
	write(A)
	read(B)
	B := B + temp
	write(B)

(1)

T_0	T_1
	read(A)
	temp := A * 0.1
	A := A -temp
	write(A)
	read(B)
	B := B + temp
	write(B)
read(A)	
A := A - 50	
write(A)	
read(B)	
B := B + 50	
write(B)	

(2)

T_0	T_1
read(A)	
A := A - 50	
	read(A)
	temp := A * 0.1
	A := A -temp
	write(A)
	read(B)
write(A)	
read(B)	
B := B + 50	
write(B)	
	B := B + temp
	write(B)

(3)

Serializability

- A schedule of a set of transactions is a linear ordering of their actions
  - e.g. for the simultaneous deposits example:

    R1(X) R2(X) W1(X) W2(X)

- A serial schedule is one in which all the steps of each transaction occur consecutively

- A serializable schedule is one which is equivalent to some serial schedule

Lock

- Definition
  - a synchronization mechanism for enforcing limits on access to DB in concurrent way.
  - one way of enforcing concurrency control policies
- Lock types
  - Shared lock (LS) readable but can not write
  - Exclusive lock (LX): read and write
  - UN(D): unlock

- Compatibility

	LS	LX
LS	true	false
LX	false	false

How can constraints be violated?

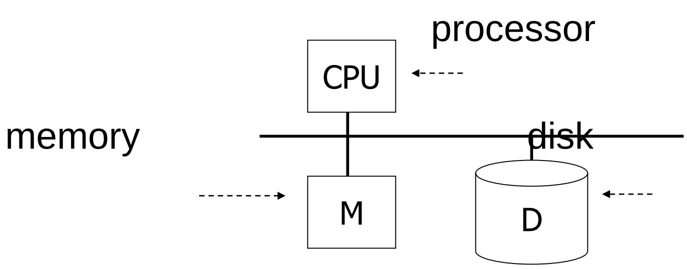
- Transaction bug
- DBMS bug
- Hardware failure

  e.g., disk crash
- Data sharing

  e.g., T1 and T2 in parallel

Failures

Events — Desired
Events — Undesired — Expected
Events — Undesired — Unexpected



Recovery

- Maintaining the consistency of DB by ROLLBACK to the last consistency state.

- Ensuring 2 properties
  - Atomic
  - Durability

➢ Using LOG

Transaction Log

- A sequence of log record keeping trace of actions executed by DBMS

  <start T>

    Log the beginning of the transaction execution

  <commit T>

    transaction is already finished

  <abort T>

    Transaction is calcel

  <T, X, v, w>

    Transaction makes an update actio, before update X=v, after update x = w

- Read(A)
- If A > 50 then display(“so du hop le”)
- Else {

  A:=A+50

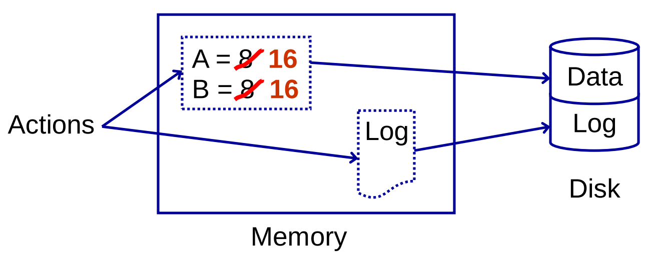
  ==========➔CRASH

  display (“ghi no tai khoan A”)

  }

Transaction Log

- Handled in main memory and put to external memory (disk) when possible



Checkpoint

- Definition:
  - moment where intermediate results and a log record are saved to disk.
  - being initiated at specified intervals
- Objective
  - minimize the amount of time and effort wasted when restart
  - the process can be restarted from the latest checkpoint rather than from the beginning.
- Log record

  <checkpoint> or <ckpt>

Discussion

- Undo Logging
  - need to write to disk as soon transaction finishes
  - ➢ Access disk
- Redo Logging
  - need to keep all modified blocks in memory until commit
  - ➢ Use memory

Undo/Redo Logging Recovery Rules

- Backwards pass (end of log → latest valid checkpoint start)
- Constructing set S of committed transactions
- undo actions of transactions not in S
- undo pending transactions
- follow undo chains for transactions in (checkpoint active list) – S
- Forward pass (latest checkpoint start → end of log)
- redo actions of S transactions