

COMP9311: DATABASE SYSTEMS

Term 1 2024

Week 10 – Revision

By Xiaoyang Wang, CSE UNSW

*Disclaimer: the course materials are sourced from previous offerings of
COMP9311 and COMP3311*

A solid orange horizontal bar at the bottom of the slide.

Final Exam

Time: 2pm - 5pm Sat 04-May Sydney time.

Exam paper: will be released on our course website (Moodle and Ed) around 1:50pm on the exam day, allowing you for an extra 10min to download the paper and upload your solutions.

How to submit: via Moodle

- Same procedure as what you did for the assignments.
- You can submit multiple times and we will mark the last one.
- Accepted format: directly answer using word or handwriting and convert to word/pdf. As long as the file is in .doc or .pdf format and clear.

Final Exam

Length: 3 hrs + 10 mins for downloading and uploading (i.e., the exam paper will be released around 10 mins before the exam starts)

Special Considerations

The exam is covered by UNSW's Fit-to-Sit policy. That means that by sitting this exam, you are declaring yourself well enough to do so. You will be unable to apply for special consideration after the exam for circumstances affecting you before it began.

Consultation

There will be consultations (both in person and online consultation) before the final exam.

Detailed schedule will be released later.

Overview

Data models

- ER, Relational Data Model and their mapping

Relational Algebra

- Use relational algebra to answer question

Database Languages

- SQL

Relational Database Design

- Functional Dependency
- Normal Forms
- Database design

Overview

Data Storage

- Record Format, Buffer Management
- Index structure
- Performance tuning

Transaction Management

- Concurrency Control
- Recovery

NoSQL

- NoSQL Concept
- Different Data Model: Key-Value, Document, Column-family, Graph

Data Model

ER, Relational Data Model and their mapping



ER Model

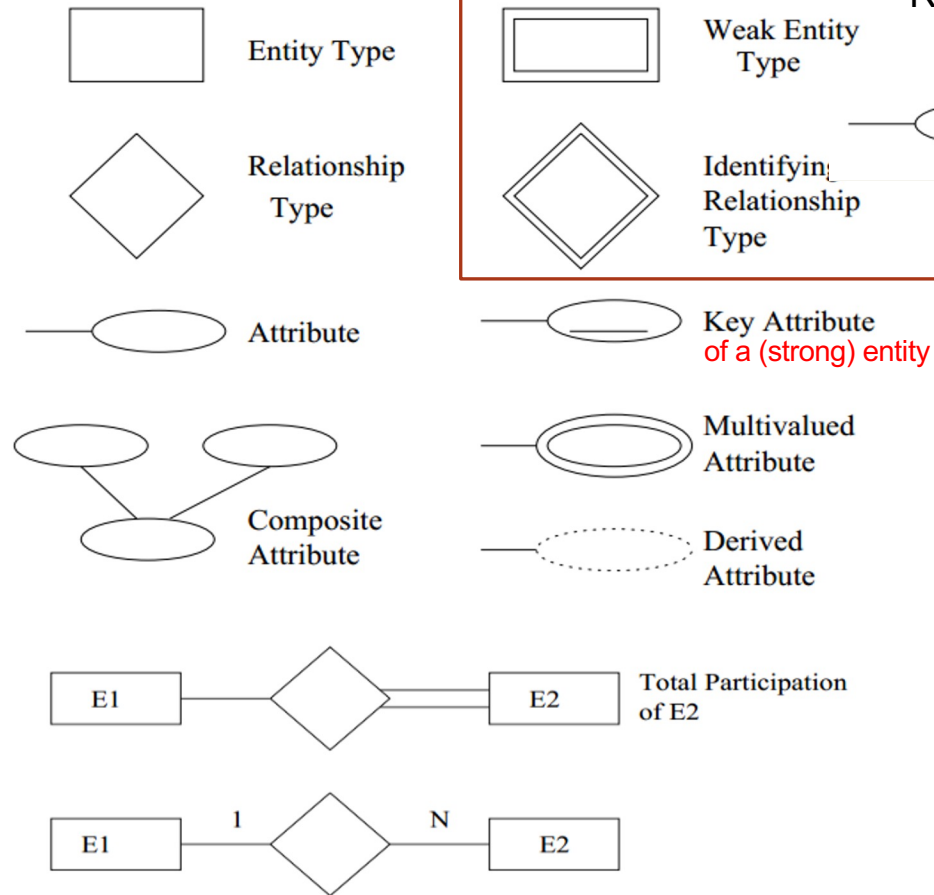
Entity type: Group of object with the same properties

Entity: member of an entity type - analogous to an object.

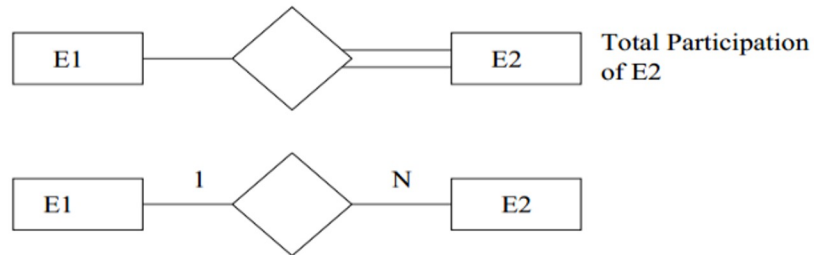
Attribute: property of object

Relationship: among objects

Notation



Relating to Weak Entities



Relational Data Model

- In the relational model, everything is described using relations.
- A relation can be thought of as a named table.
- Each column of the table corresponds to a named attribute.
- The set of allowed values for an attribute is called its domain.
- Each row of the table is called a tuple of the relation.

Keys

- Keys are used to identify tuples in a relation.
- A superkey is a set of attributes that uniquely determines a tuple.
- Note that this is a property of the relation that does not depend on the current relation instance.
- A candidate key is a superkey, none of whose proper subsets is a superkey.
- Keys are determined by the applications.

Integrity Constraints

There are several kinds of integrity constraints that are an integral part of the relational model

- Key constraint: candidate key values must be unique for every relation instance.
- Entity integrity: an attribute that is part of a primary key cannot be NULL.
- Referential integrity: The third kind has to do with “foreign keys”.

Foreign Keys

Foreign keys are used to refer to a tuple in another relation.

A set, FK, of attributes from a relation schema R1 may be a foreign key if

- the attributes have the same domains as the attributes in the primary key of another relation schema R2, and
- a value of FK in a tuple t1 of R1 either occurs as a value of PK for some tuple t2 in R2 or is null.

Referential integrity: The value of FK must occur in the other relation or be entirely NULL.

ER to Relational Model Mapping

- One technique for database design is to first design a conceptual schema using a high-level data model, and then map it to a conceptual schema in the DBMS data model for the chosen DBMS.
- Here we looked at a way to do this mapping from the ER to the relational data model. (see details in the lecture notes of Relational Data Model).
- Composite and multivalued attributes are allowed in ER model, but not allowed in relational data model.

Relational Algebra

Relational Algebra is a procedural data manipulation language (DML).

It specifies operations on relations to define new relations:

- Unary Relational Operations: Select, Project
- Operations from Set Theory: Union, Intersection, Difference, Cartesian Product
- Binary Relational Operations: Join, Divide.

Database Languages

Database Languages: SQL

SELECT attributes

FROM relations

WHERE condition

The result of this statement is a table, which is typically displayed on output.

The SELECT statement contains the functionality of select, project and join from the relational algebra.

Relational Database Design

Functional dependency, normal forms, decomposition algorithms for 3NF and BCNF

- Decide whether a particular relation R is in “good” form.
- If a relation R is not in “good” form, decompose it into a set of relations $\{R_1, R_2, \dots, R_n\}$ such that
 - each relation is in good form
 - the decomposition is a lossless

Functional Dependencies

A functional dependency describes a **relation** between attributes

Whenever any two tuples t_1 and t_2 of r agree on one attribute α , they also agree on another attribute β .

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

This relation is denoted $\alpha \rightarrow \beta$.

Armstrong's Axioms

These are the inference rules for functional dependencies

- Rule 1 (reflexivity)

if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

- Rule 2 (augmentation)

if $\alpha \rightarrow \beta$, then $\gamma \alpha \rightarrow \gamma \beta$

- Rule 3 (transitivity)

if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

- Where α , β , γ are all (nonempty) sets of attributes

Armstrong's Axioms

Additional Rules we inferred from Armstrong's axioms.

- Rule 4 (**additivity**):

If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta \gamma$ holds

- Rule 5 (**projectivity**):

If $\alpha \rightarrow \beta \gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds

- Rule 6 (**pseudo-transitivity**):

If $\alpha \rightarrow \beta$ holds and $\gamma \beta \rightarrow \delta$ holds, then $\alpha \gamma \rightarrow \delta$ holds

Closure of F

Definition. the set of all dependencies that can be inferred from F is called the **closure** of F .

F^+ denotes the closure of F .

F^+ includes dependencies in F .

Note: We typically reserve F to denote the set of functional dependencies that are specified on relation schema R .

Closure of Attribute Sets

An **algorithm** for you to follow step by step

```
X := X;  
change := true;  
while change do  
begin  
    change := false;  
    for each FD  $W \rightarrow Z$  in F do  
    begin  
        if  $(W \subseteq X^+) \text{ and } (Z \notin X^+)$  then do  
        begin  
             $X^+ := X^+ \cup Z$ ;  
            change := true;  
        end  
    end  
end  
end
```

Procedurally Determine Keys

How to compute a candidate key of a relation R based on the FD's belonging to R

Algorithm

- Step 1 : Assign a super-key of R in F to X.
- Step 2 : Iteratively remove attributes from X while retaining the property $X^+ = R$ till no reduction on X is possible.
- The remaining X is a key.

Compute All the Candidate Keys

Given a relational schema R and a set F of functional dependencies on R , the algorithm to compute all the candidate keys is as follows:

$T := \emptyset$

Main:

$X := S$ where S is a super key which does not contain any candidate key in T

remove := true

While remove do

 For each attribute $A \in X$

 Compute $\{X-A\}^+$ with respect to F

 If $\{X-A\}^+$ contains all attributes of R then

$X := X - \{A\}$

 Else

 remove := false

$T := T \cup X$

Repeat Main until no available S can be found. Finally, T contains all the candidate keys.

Normal Forms

Normal Forms for relational databases:

- 1NF, 2NF, 3NF (Codd 1972)
- Boyce-Codd NF (1974)
- 1NF: This simply means that attribute values are **atomic** and is part of the definition of the relational model.
- 2NF: A relation schema R is in 2NF if every nonprime attribute A in R is not partially dependent on any key of R
- 3NF: A relation scheme is in 3NF if for all non-trivial FD's of the form $X \rightarrow A$: either X is a superkey or A is a prime attribute. It disallows transitive dependencies.
- BCNF: A relation scheme is in BCNF if whenever $X \rightarrow A$ holds and $X \rightarrow A$ is non-trivial, X is a superkey.

Relational Database Design

- Anomalies can be reduced by decomposing the relation into a normal form.
- A decomposition of a relation scheme, R , is a set of relation schemes $\{R_1, \dots, R_n\}$ such that $R_i \subseteq R$ for each i , and $\bigcup_{i=1}^n R_i = R$
- A decomposition $D = \{R_1, \dots, R_n\}$ of R is dependency-preserving wrt a set F of FDs if $(F_1 \cup \dots \cup F_n)^+ = F^+$, where F_i means the projection of F onto R_i .
- A decomposition $\{R_1, \dots, R_n\}$ of R is a lossless join decomposition with respect to a set F of FD's if for every relation instance r that satisfies F : $r = \pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_n}(r)$.

Property	3NF	BCNF
Elimination of redundancy due to functional dependency	Most	Yes
Lossless Join	Yes	Yes
Dependency preservation due to functional dependency	Yes	Maybe

Lossless Decomposition into BCNF

Algorithm TO_BCNF

- $D := \{R_1, R_2, \dots, R_n\}$
- **While** (there exists a $R_i \in D$ and R_i is not in BCNF) **Do**
 - 1 . find a $X \rightarrow Y$ in R_i that **violates** BCNF;
 - 2. replace R_i in D by $(R_i - Y)$ and $(X \cup Y)$;

Algorithm for Minimal Cover

Algorithm Min_Cover

Input: a set F of functional dependencies.

Step 1: Reduce right side.

Apply Algorithm Reduce Right to F .

Step 2: Reduce left side.

Apply Algorithm Reduce Left to the output of Step 1.

Step 3: Remove redundant FDs.

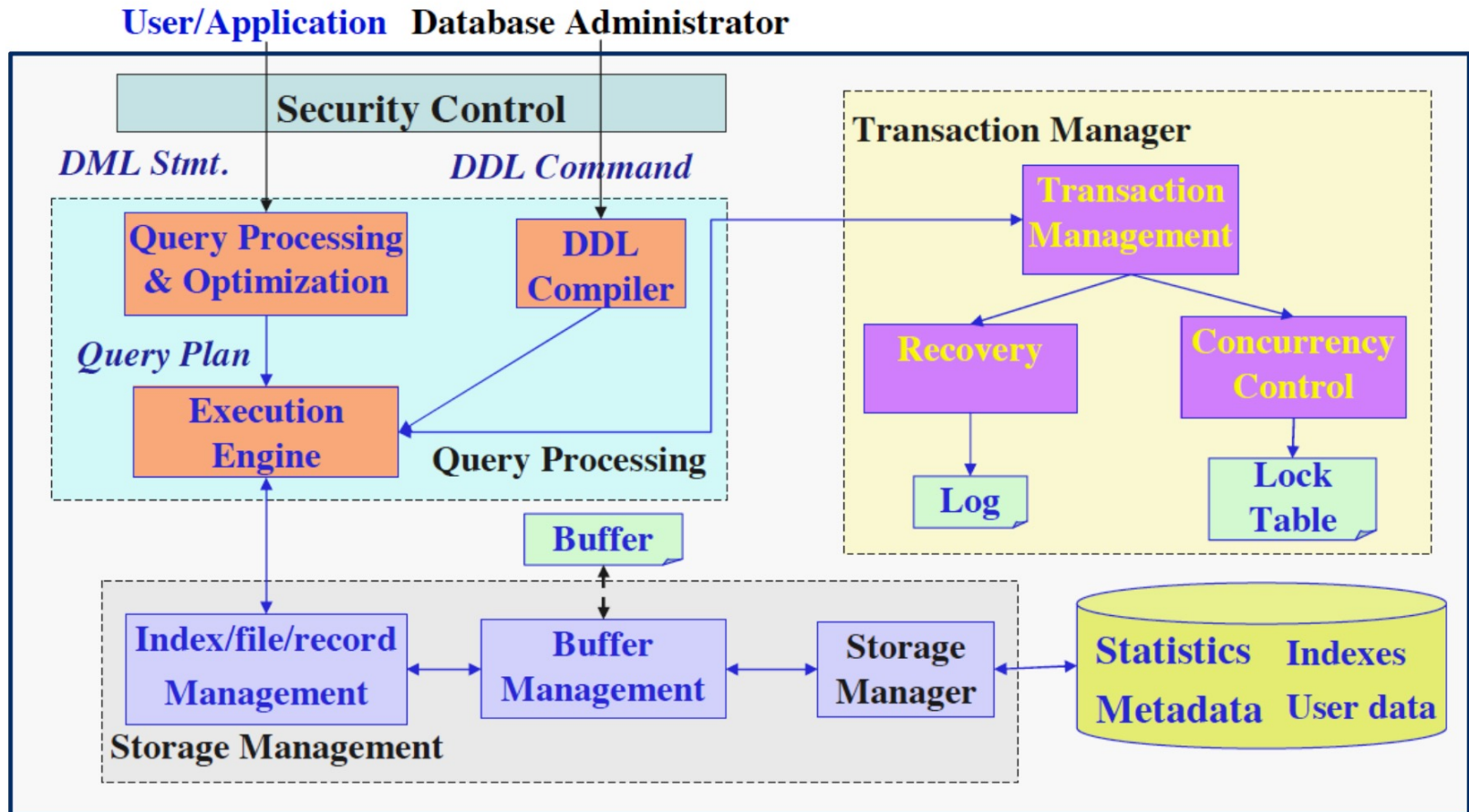
Apply Algorithm Remove_redundancy to the output of Step 2.

3NF Decomposition Algorithm

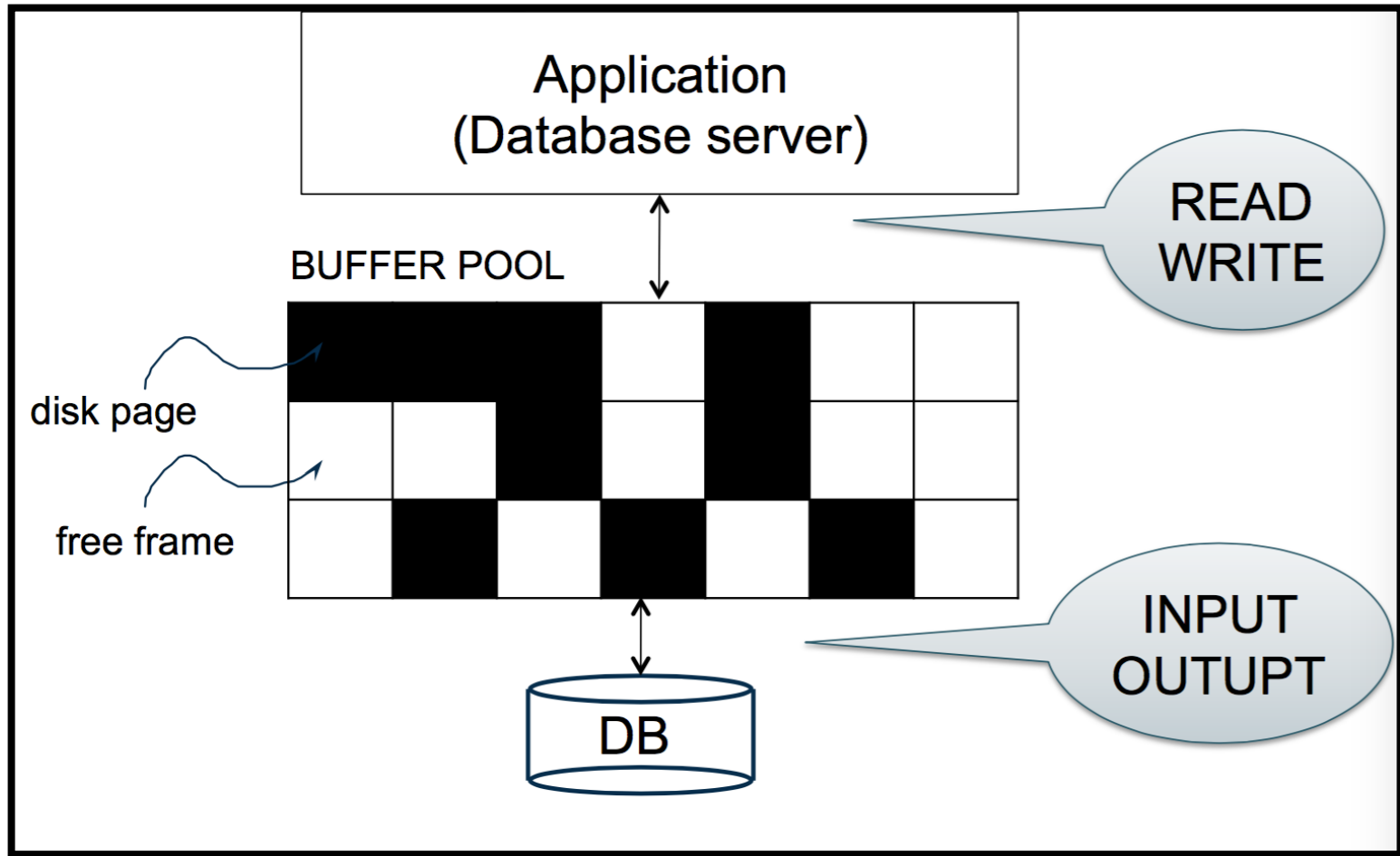
Algorithm 3NF decomposition

- Find a minimal cover G for F .
- For each left-hand-side X of a functional dependency that appears in G , create a relation schema in D with attributes $\{X \cup \{A_1\} \cup \{A_2\} \dots \cup \{A_k\}\}$, where $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$ are the only dependencies in G with X as left-hand-side (X is the key to this relation).
- If none of the relation schemas in D contains a key of R , then create one more relation schema in D that contains attributes that form a key of R .
- Eliminate redundant relations from the resulting set of relations in the relational database schema. A relation R is considered redundant if R is a projection of another relation S in the schema; alternately, R is subsumed by S .

Storing Data: Disk, File and Index



Buffer Management in a DBMS



Buffer Replacement Policies

Least Recently Used (LRU)

- release the frame that has not been used for the longest period.
- intuitively appealing idea but can perform badly

First in First Out (FIFO)

- need to maintain a queue of frames
- enter tail of queue when read in

Most Recently Used (MRU):

- release the frame used most recently

Random

No one is guaranteed to be better than the others. Quite dependent on applications.

Record Formats

Records are stored within fixed-length blocks.

Fixed-length: each field has a fixed length as well as the number of fields.

33357462	Neil Young	Musician	0277
4 bytes	40 bytes	20 bytes	4 bytes

- Easy for intra-block space management.
- Possible waste of space.

Variable-length: some field is of variable length.

33357462	Neil Young	Musician	0277
4 bytes	10 bytes	8 bytes	4 bytes

- complicates intra-block space management
- does not waste (as much) space.

Index

Choice orthogonal to indexing technique used to locate data entries with a a given key value.

There may be several indexes on a given file of data records, each with a different search key.

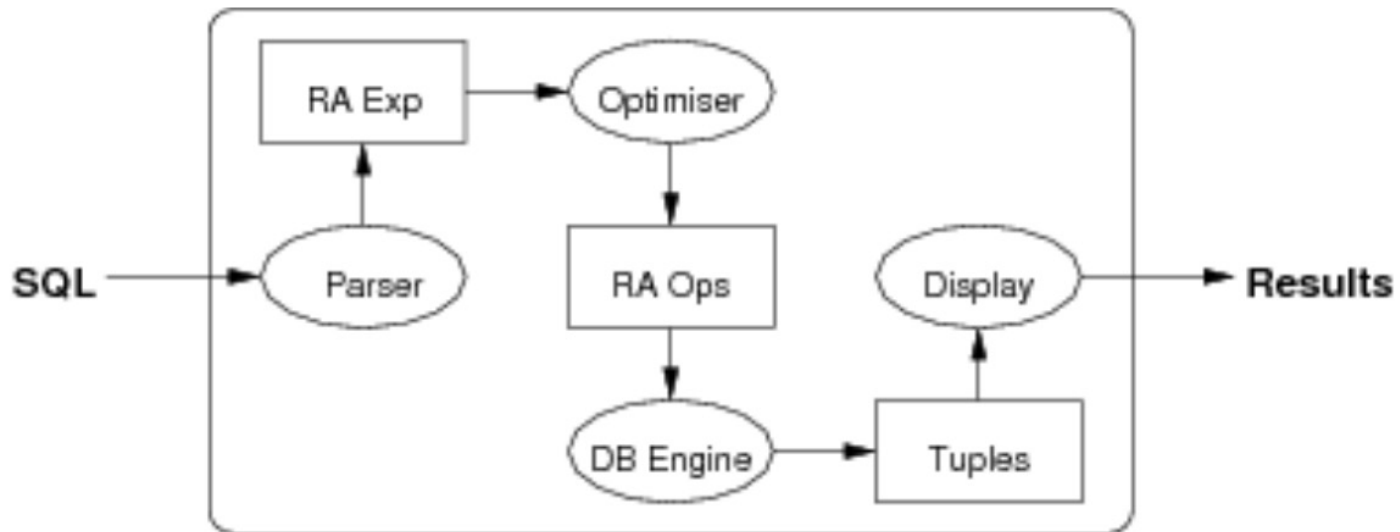
Indexes can be classified as

- clustered vs. unclustered
- dense vs. sparse.

Differences have important consequences for utility/performance

Query Processing

mapping SQL to relational algebra (RA)



RA Expressions → Optimiser → concrete RA operations
(e.g., JOIN on empid) (e.g., HASH JOIN on empid)

Query Optimization Problem

The query optimizer start with an RA expression, then

- generates a set of equivalent expressions
- generates possible execution plans for each
- estimates cost of each plan, chooses cheapest

The cost of evaluating a query is determined by:

- size of relations (database relations and temporary relations)
- access mechanisms (indexing, hashing, sorting, join algorithms)
- size/number of main memory buffers (and replacement strategy)

Analysis of costs involves estimating:

- the size of intermediate results
- then, based on this, cost of disk storage accesses (i.e., I/O - page read/write)

Performance Tuning

Performance can be considered at two times:

during schema design

- typically towards the end of schema design process
- requires schema transformations such as **denormalization**

after schema design

- requires adding extra data structures such as indexes

ACID

A **transaction** is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data, the database system must ensure the ACID property.

- **Atomicity:** Either all operations of the transaction are properly reflected in the database, or none are.
- **Consistency:** Every transaction sees a consistent database.
- **Isolation:** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
- That is, for every pair of transactions T_i and T_j , it must appear to T_i that either T_j , finished execution before T_i started, or T_j started execution after T_i finished.
- **Durability:** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

Concurrent Control

Question: why not run only serial schedules?

- It is desirable to interleave the operations of transactions in an **appropriate way**.

We can fully utilise **resources**.

For example, if one transaction is waiting for I/O to complete, another transaction can use the CPU.

Point:

- serial schedules are considered unacceptable in practice
- **executing multiple transactions concurrently has significant benefits.**

Conflict Serializability

We say that a schedule S is **conflict serializable** if it is conflict equivalent to a serial schedule

Algorithm

Step 1: Construct a precedence graph.

Step 2: Check if the graph is cyclic:

- Cyclic: non-serializable.
- Acyclic: serializable.

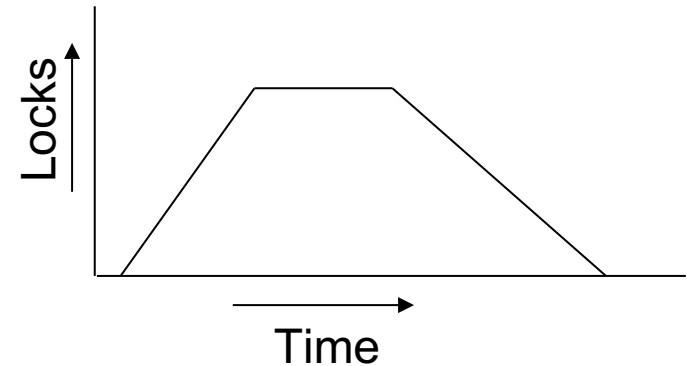
Two Phase Locking (2PL)

Phase 1: Growing Phase

- Transaction obtains locks
- Transaction does not release any locks

Phase 2: Shrinking Phase

- Transaction releases locks
- Transaction does not obtain new locks



This protocol ensures serializability: and produces **conflict-serializable schedules**.

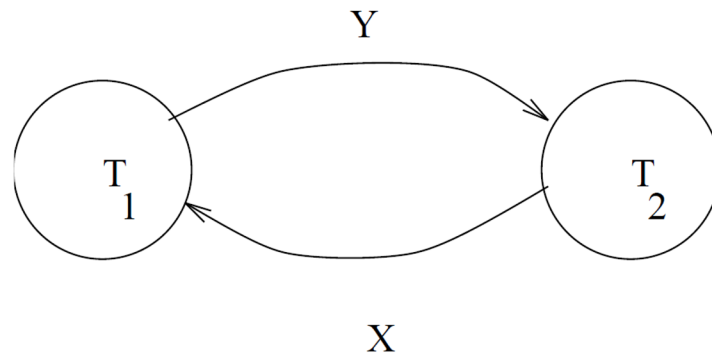
Testing for Deadlocks

Create a **wait-for graph** for currently **active** transactions:

- create a vertex for each transaction; and
- an arc from T_i to T_j if T_i **is waiting** for an item locked by T_j .

If the graph has a cycle, then a deadlock has occurred.

Example:



System Log

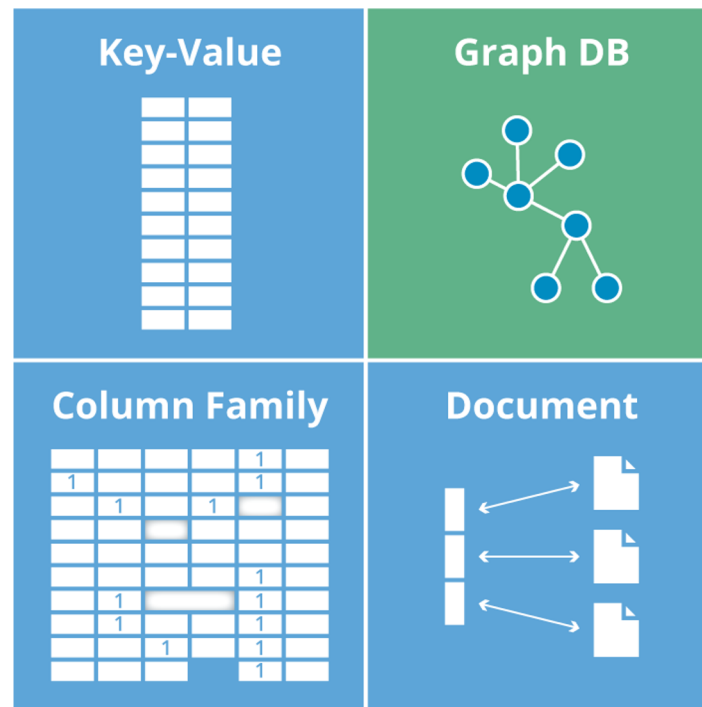
To be able to recover from failures that affect transactions, the system maintains a **log** to keep track of all transaction operations that affect the values of database items.

- The system needs to record the states information to recover failures correctly.
- The information is maintained in a log (also called journal).
- The system log is kept in hard disk but maintains its current contents in main memory.

NoSQL

Why NoSQL?

How to choose a database model given your requirements?



Note 1

Computer Updates

- You must ensure that auto-updates are disabled on your computer prior to the online assessment.
- Special consideration will NOT be awarded on the grounds that your computer performed an update during an online assessment.

Note 2

If you accidentally upload the wrong document or wrong version of your exam

- Students are responsible for uploading the correct version of the correct document. Once uploaded, there will be no opportunity to replace or re-upload your exam papers AFTER the end of the exam.
- The documents submitted will be the documents that are marked. There is NO provision for students who upload incorrect or incomplete documents.
- Therefore, you must check the work before you submit.

Note 3

Communication during the exam

- Students are NOT permitted to communicate with other people during the exam (including the reading and submission periods).
- Attempts to communicate with other students will be considered to be serious academic misconduct.
- This includes communication in person, by email, text, message, telephone, or internet.
- i.e., do the work yourself

Note 4

Sharing answers with others or posting them online

- Any attempts to collaborate or share your answers with others will be considered a very serious case of academic misconduct

Note 5

Checklist

- Be logged in at your computer and ready to go 20 minutes before the exam commences.
- Ensure your device has power, and the charger is plugged in.
- If applicable remind your roommates or family that you'll be taking an exam to avoid interruptions.

Tips

- You can attempt the questions in any order, arrange your time wisely.
- Read all questions carefully
- Be fully prepared before the exam: don't forget to eat lunch, take break and relax yourself, no need to panic

myExperience Survey

The UNSW myExperience survey is still open. Please submit your feedback.

“Please participate in the myExperience Survey and take the opportunity to share your constructive thoughts on your learning experience. Your contributions help your teachers and shape the future of education at UNSW.”

You can access the survey by logging into Moodle or accessing myexperience.unsw.edu.au directly.

Thank you and all the best!