> >

COMP9315 Week 01

^ >>

COMP9315 25T1
DBMS Implementation

(Data structures and algorithms inside relational DBMSs)

Lecturer: **Xiaoyang Wang**

Web Site: http://www.cse.unsw.edu.au/~cs9315/

(If WebCMS unavailable, use
http://www.cse.unsw.edu.au/~cs9315/25T1/)

<< ^ >>

❖ Lecturer

Name: Xiaoyang Wang

Office: K17-501D

Email: xiaoyang.wang1@unsw.edu.au

Research: Database and data mining
Fintech and financial network analysis
LLM with graph

<< ^ >>

❖ Course Admin and Tutor

Course admin:	Xingyu Tan
Email:	xingyu.tan@unsw.edu.au
Tutor:	Peiting Xie and Xiangjun Zai
Office:	K17-201/501 closed area

Mostly research students from the Data and Knowledge Research Group

<< ^ >>

❖ Plan for the delivery of this course

Lectures

- Describe all syllabus topics in some detail, with exercises and examples
- Start from Week 1: 9 AM – 11 AM (Tue) and 11 AM – 1 PM (Thu)
- In-person lecture and online streaming at the same time
- There will be tutor to answer questions in the online chat panel
- In-person (Physics Theatre and Keith Burrows Theatre)
- Online streaming (Moodle → Blackboard Collaborate)
- Recorded and uploaded to Echo360 (Moodle → Lecture Recordings)

<< ^ >>

❖ Plan for the delivery of this course (cont)

Weekly consultation: Tutor present to answer course related questions

Section	Time	Location
In-person	Wednesday 17:00 - 18:00 (Week 1-5, 7-10)	K17 Rm203
Online	Friday 14:00-15:00 (Week 1-5, 7-10)	collab

❖ Plan for the delivery of this course (cont)

How to access Online Consultations

- Log into Moodle (https://moodle.telt.unsw.edu.au/)
- Go to course (COMP9315 - 2025 T1)
- Click Blackboard Collaborate
- Click the corresponding consultation session to join

Practice/Theory execises

- Sample answers are provided
- To be released on course website at every interval

❖ Support your learning

Couse Website

- https://webcms3.cse.unsw.edu.au/COMP9315/25T1/
- All course announcements, content and links to other tools/platforms

Online forum

- Ed forum (register yourself with https://edstem.org/au/join/PNfqdk)
- Tutors will visit the forum regularly to answer questions

Moodle

- Access Echo360, submit assignment, MyExperience and Blackboard Collaborate

Email

- If you need any help, email me (xiaoyang.wang1@unsw.edu.au) or Xingyu (xingyu.tan@unsw.edu.au)

<< ^ >>

❖ For Other Enrolment Issues

- The course enrolment process isn't something lecturers have direct control over.
- Matters such as the number of students that can take a course, etc.
- Students always adjust their courses during prior to the census date.
- Checking daily for openings is still recommended.

<< ^ >>

❖ Course Goals

Introduce you to:

- architecture of relational DBMSs (e.g. PostgreSQL)
- algorithms/data-structures for data-intensive computing
- representation of relational database objects
- representation of relational operators (sel,proj,join)
- techniques for processing SQL queries
- techniques for managing concurrent transactions
- concepts in distributed and non-relational databases

Develop skills in:

- analysing the performance of data-intensive algorithms
- the use of C to implement data-intensive algorithms

❖ Pre-requisites

We assume that you are already familiar with

- the C language and programming in C (or C++)
  (e.g. completed ≥ 1 programming course in C)
- developing applications on RDBMSs
  (SQL, [relational algebra] e.g. an intro DB course)
- basic ideas about file organisation and file manipulation
  (Unix `open, close, lseek, read, write, flock`)
- sorting algorithms, data structures for searching
  (sorting, trees, hashing e.g. a data structures course)

If you don't know this material very well, don't take this course.

<< ^ >>

❖ Exercise: SQL (revision)

Given the following schema:

```
Students(sid, name, degree, ...)
e.g. Students(3322111, 'John Smith', 'MEngSc', ...)
Courses(cid, code, term, title, ...)
e.g. Courses(1732, 'COMP9311', '12s1', 'Databases', ...)
Enrolments(sid, cid, mark, grade)
e.g. Enrolments(3322111, 1732, 50, 'PS')
```

Write an SQL query to solve the problem

- find all students who enrolled COMP9315 in 18s2
- for each student, give (student ID, name, mark)

Solution

<< ^ >>

❖ Exercise: Unix File I/O (revision)

Write a C program that reads a file, block-by-block.

Command-line parameters:

- block size in bytes
- name of input file

Use low-level C operations: `open`, `read`.

Count and display how many blocks/bytes read.

Solution

<< ^ >>

❖ Learning/Teaching

What's available for you:

- Textbooks: describe some syllabus topics in detail
- Lecture slides: summarise topics and contain exercises
- Lecture videos: for review or if you miss a lecture, or are in WEB stream
- Readings: research papers on selected topics

The onus is on you to use this material.

Note: Exercises and videos will be available only *after* the lecture.

<< ^ >>

❖ Learning/Teaching (cont)

Things that you need to **do**:

- Exercises: tutorial-like questions
- Prac work: lab-class-like exercises
- Assignments: large/important practical exercises
- On-line quizzes: for self-assessment

Dependencies:

- Exercises → Exam (theory part)
- Prac work → Assignments → Exam (prac part)

There are **no** tute/lab classes; use Ed Forum, Email, Help Sessions

- debugging is best done in person (can see full context)
- at the very least, send error messages (**not** screenshots)

<< ^ >>

❖ Rough Schedule

Week 01	intro, dbms review, RA, catalogs
Week 02	storage: disks, buffers, pages, tuples
Week 03	RA ops: scan, sort, projection
Week 04	selection: heaps, hashing, indexes
Week 05	selection: N-d matching, similarity
Week 06	no lectures
Week 07	joins: naive, sort-merge, hash join
Week 08	query processing, optimisation
Week 09	transactions: concurrency, recovery
Week 10	distributed and non-SQL databases

<< ^ >>

❖ Textbooks

No official text book; several are suitable ...

- Garcia-Molina, Ullman, Widom
  "Database Systems: The Complete Book"
- Ramakrishnan, Gehrke
  "Database Systems Management"
- Silberschatz, Korth, Sudarshan
  "Database System Concepts"
- Kifer, Bernstein, Lewis
  "Database Systems: An algorithmic-oriented approach"
- Elmasri, Navathe
  "Database Systems: Models, languages, design ..."

but not all cover all topics in detail

<< ^ >>

❖ Prac Work

In this course, we use PostgreSQL v15.11

Prac Work requires you to compile PostgreSQL from source code

- instructions explain how to do this on Linux at CSE
- also works easily on Linux and Mac OSX at home
- PostgreSQL docs describe how to compile for Windows

Make sure you do the first Prac Exercise when it becomes available.

Sort out any problems ASAP.

<< ^ >>

❖ Prac Work (cont)

PostgreSQL is a large software system:

- > 2400 source code files in the core engine/clients
- > 1,400,000 lines of C code in the core

You won't be required to understand all of it :-)

You will need to learn to navigate this code effectively.

Will discuss relevant parts in lectures to help with this.

PostgreSQL books?

- tend to add little to the manual, and cost a lot

<< ^ >>

❖ Assignments

Schedule of assignment work:

Ass	Description	Due	Marks
1	New Data Type	Week 5	15%
2	Query Processing	Week 10	20%

Assignments will be carried out individually

Assignments will require up-front code-reading (see Pracs).

<< ^ >>

❖ Assignments (cont)

Don't leave assignments to the last minute

- they require significant code reading
- as well as code writing and testing
- and, you can submit early.

Cheating will be penalised with mark of zero for that assignment

- submitting work copied from another student
- submitting work copied from an online code repo
- submitting work derived from generative AI
- submitting work written by someone else for $$$

You are only cheating yourself ... losing an opportunity to learn

<< ^ >>

❖ Quizzes

Over the course of the semester ...

- five online quizzes; released Monday, due Friday
- taken in your own time (but there are deadlines)
- each quiz is worth a small number of marks

Quizzes are primarily a review tool to check progress.

But they contribute 15% of your overall mark for the course.

<< ^ >>

❖ UNSW Standard Late Penalty

5% reduction per day for late submission

- 0 marks after 5 days late
- 1 second late = 1 day late
- Submit wrong files = Late
- Double check to make sure your submission is correct and on time!

<< ^ >>

❖ Special Consideration

- We will grant no-penalty extensions due to extreme circumstances (e.g., medical emergencies)
- Apply via myUNSW as soon as possible
- Evidence is needed, application process and details in https://www.student.unsw.edu.au/special-consideration
- No other excuses are accepted (e.g., network down, too busy, forgot to submit)

<< ^ >>

❖ Exam

Three-hour exam in the exam period.

Exam is held in CSE Labs (learn the environment, VLab)

In-person exam only, don't take the course if you cannot make in-person.

PostgreSQL and C documentation (only) will be available in the exam.

Things that we can't reasonably test in the exam:

- writing large programs, running major experiments

Everything else is potentially examinable.

Contains: descriptive questions, analysis, small programming exercises.

Exam contributes 50% of the overall mark for this course.

<< ^ >>

❖ Exam (cont)

If you cannot attend the final exam ...

- because of documented illness/misadventure

then you will be offered a Supplementary Exam

You get one chance at passing the exam

- unsw's fit-to-sit rule applies

Exam hurdle 20/50 (which is 40%)

You must attend the exam in-person

- no online exams are available ... be in Sydney ... be at UNSW

❖ Assessment Summary

Your final mark/grade is computed according to the following:

```
quiz   = mark for on-line quizzes    (out of 15)
ass1   = mark for assignment 1       (out of 15)
ass2   = mark for assignment 2       (out of 20)
exam   = mark for final exam         (out of 50)
okExam = exam > 20/50             (after scaling)

mark   = ass1 + ass2 + quiz + exam
grade  = HD|DN|CR|PS,   if mark ⩾ 50 && okExam
       = FL,            if mark < 50 && okExam
       = UF,            if !okExam
```

<< ^ >>

❖ What We Expect From You

Importantly

- We want you to feel welcome and safe in asking questions/help online
- This can only happen if we all behave respectfully towards each other when we interact online
- No judgement. Everybody in the class is here to learn something and everybody in the class will help each other to have the best learning experience.
- Summary: work hard and be nice to each other.

<< ^ >>

❖ Relational Database Revision

<< ^ >>

❖ Relational DBMS Functionality

Relational DBMSs provide a variety of functionalities:

- storing/modifying data and meta-data (data defintions)
- constraint definition/storage/maintenance/checking
- declarative manipulation of data (via SQL)
- extensibility via views, triggers, stored procedures
- query re-writing (rules), optimisation (indexes)
- transaction processing, concurrency/recovery
- etc. etc. etc.

Common feature of all relational DBMSs: relational model, SQL.

<< ^ >>

❖ Data Definition

Relational data: relations/tables, tuples, values, types, e.g.

```
create domain WAMvalue float
   check (value between 0.0 and 100.0);

create table Students (
   id          integer,   -- e.g. 3123456
   familyName  text,      -- e.g. 'Smith'
   givenName   text,      -- e.g. 'John'
   birthDate   date,      -- e.g. '1-Mar-1984'
   wam         WAMvalue,  -- e.g. 85.4
   primary key (id)
);
```

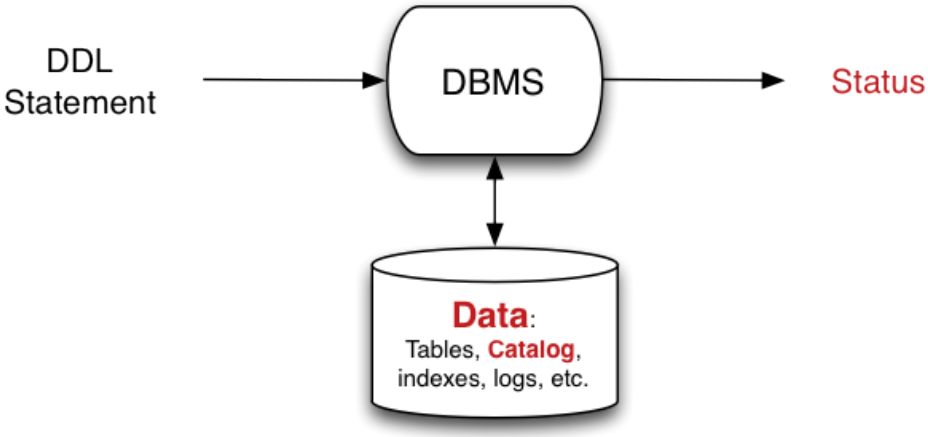
The above adds meta-data to the database.

DBMSs typically store meta-data as special tables (catalog).

<< ^ >>

❖ Data Definition (cont)

Input: DDL statement (e.g. `create table`)



Result: meta-data in catalog is modified

<< ^ >>

❖ Data Modification

Critical function of DBMS: changing data

- `insert` new tuples into tables
- `delete` existing tuples from tables
- `update` values within existing tuples

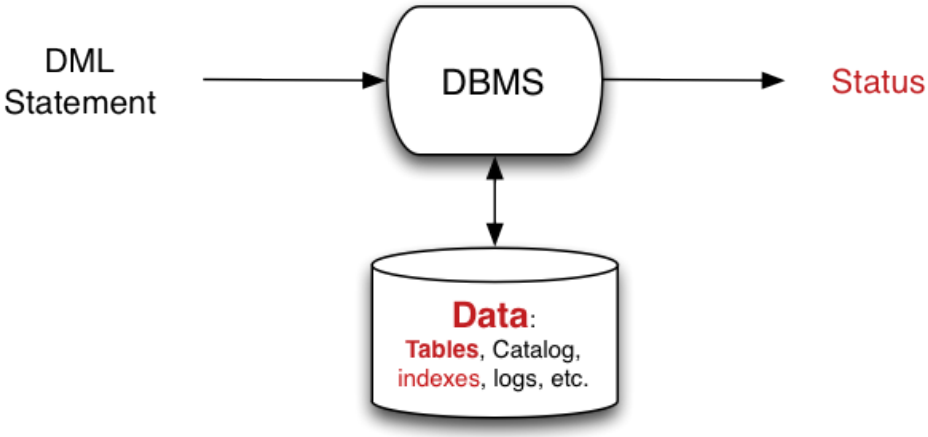
E.g.

```
insert into Enrolments(student,course,mark)
values (3312345, 5542, 75);

update Enrolments set mark = 77
where  student = 3354321 and course = 5542;

delete Enrolments where student = 3112233;
```

<< ^ >>

❖ Data Modification (cont)

Input: DML statements



Result: tuples are added, removed or modified

<< ^ >>

❖ Query Evaluator

Most common function of relational DBMSs

- read an SQL query
- return a table giving result of query

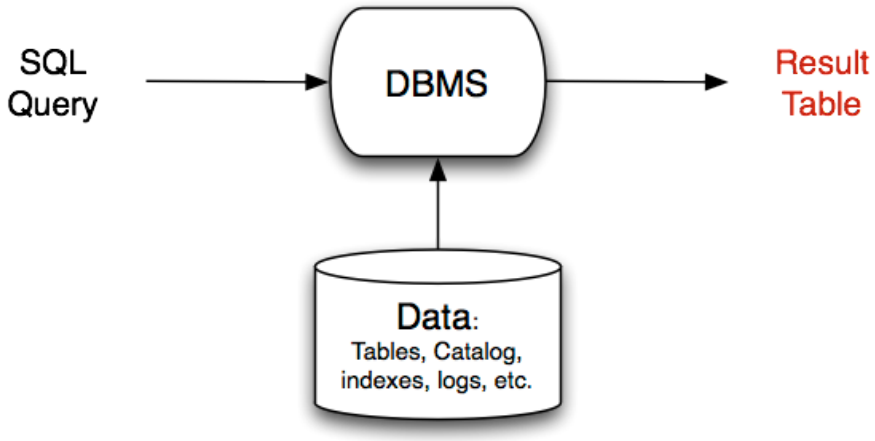
E.g.

```
select s.id, c.code, e.mark
from   Students s
       join Enrolments e on s.id = e.student
       join Courses c on e.course = c.id;
```

<< ^ >>

❖ Query Evaluator (cont)

Input: SQL query



Output: table (displayed as text)

<< ^ >>

❖ DBMS Architecture

The aim of this course is to

- look inside the DBMS box
- discover the various mechanisms it uses
- understand and analyse their performance

Why should we care? (apart from passing the exam)

Practical reason:

- if we understand how query processor works,
  we can (maybe) do a better job of writing efficient queries

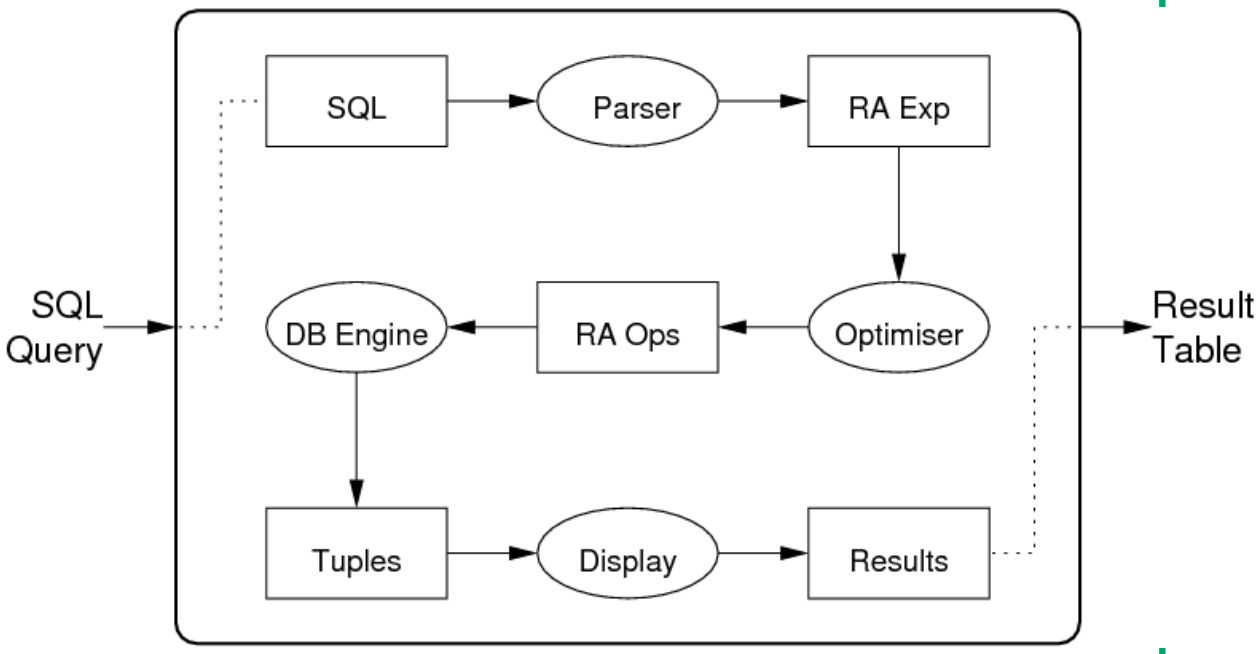
Educational reason:

- DBMSs contain interesting data structures + algorithms
  which may be useful outside the (relational) DBMS context

<< ^ >>

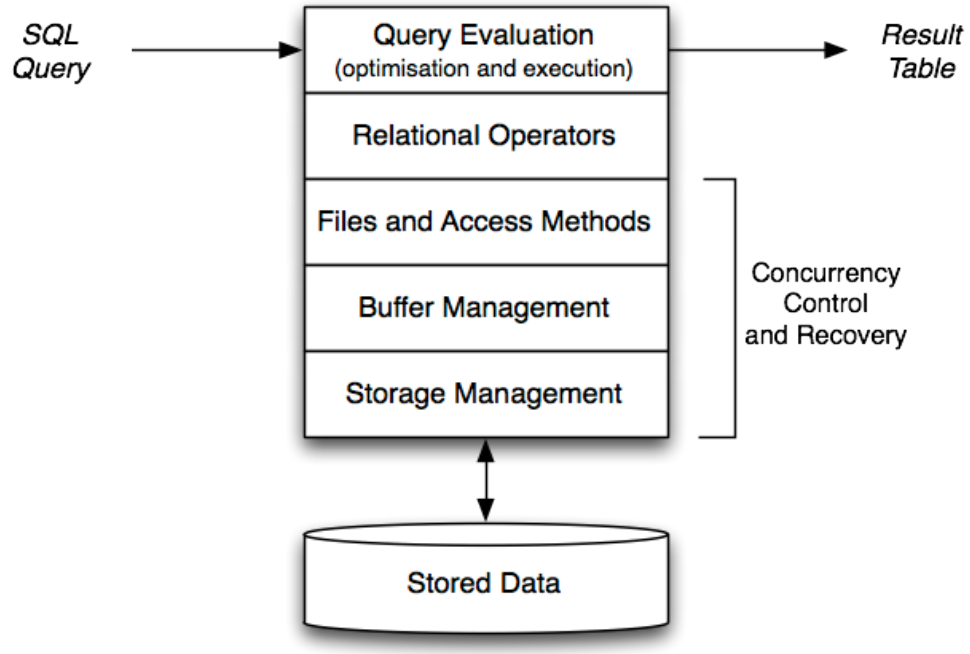
❖ DBMS Architecture (cont)

Path of a query through a typical DBMS:



<< ^ >>

❖ DBMS Architecture (cont)



<< ^ >>

❖ DBMS Architecture (cont)

Important factors related to DBMS architecture

- data is stored permanently on large slow devices**
- data is processed in small fast memory

Implications:

- data structures should minimise storage utilisation
- algorithms should minimise memory/disk data transfers

Modern DBMSs interact with storage via the O/S file-system.

** SSDs change things a little, but most high volume bulk storage still on disks

<< ^ >>

❖ DBMS Architecture (cont)

Implementation of DBMS operations is complicated by

- potentially multiple concurrent accesses to data structures
  (not just data tables, but indexes, buffers, catalogues, ...)
- transactional requirements (atomicity, rollback, ...)
- requirement for high reliability of raw data (recovery)

Require "concurrency-tolerant" data structures.

Transactions/reliability require some form of logging.

<< ^ >>

❖ Database Engine Operations

DB engine = "relational algebra virtual machine".

Common operators of relational algebra (RA):

selection(σ)	projection (π)	join (⋈)
union (∪)	intersection (∩)	difference (-)
sort	group	aggregate

For each of these operations:

- various data structures and algorithms are available
- DBMSs may provide only one, or may provide a choice

<< ^ >>

❖ Relational Algebra

All RA operators return a result of type relation.

For convenience, we can name a result and use it later.

E.g. database *R1(x,y)*, *R2(y,z)*,

```
Tmp1(x,y)   = Sel[x>5]R1
Tmp2(y,z)   = Sel[z=3]R2
Tmp3(x,y,z) = Tmp1 Join Tmp2
Res(x,z)    = Proj[x,z] Tmp3

-- which is equivalent to
Tmp1(x,y,z) = R1 Join R2
Tmp2(x,y,z) = Sel[x>5 & z=3] Tmp1
Res(x,z)    = Proj[x,z]Tmp2
```

Each "intermediate result" has a well-defined schema.

<< ^ >>

❖ Exercise: Relational Algebra

Assume a schema: R(a,b,c), S(x,y)

Translate each of the following SQL statements to RA

- select * from R
- select a,b from R
- select * from R where a > 5
- select * from R join S on R.a = S.y

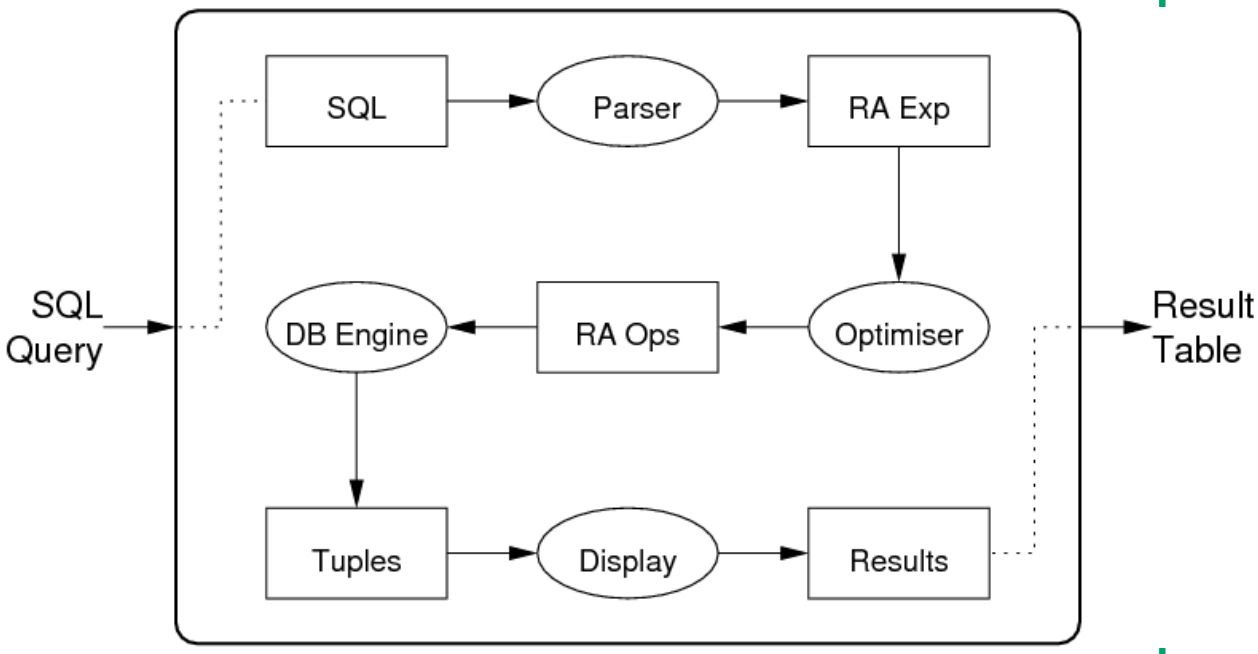
Indicate: the fields and # tuples in the result

Solution

<< ^ >>

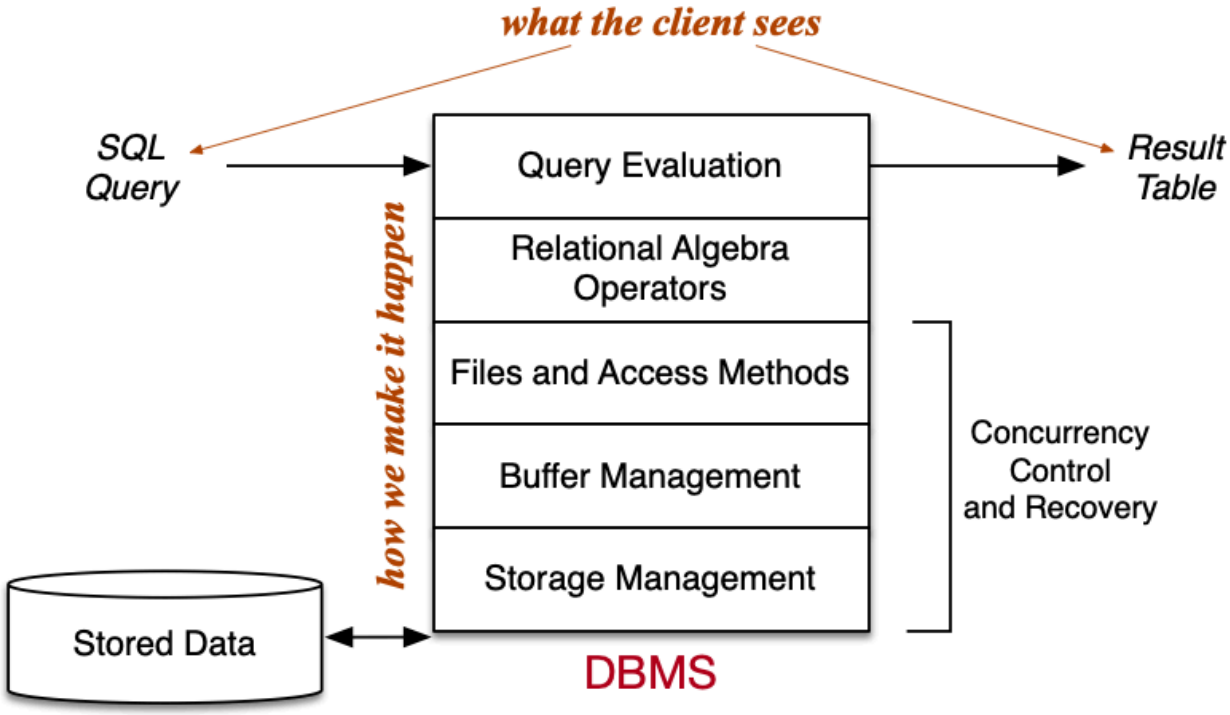
❖ DBMS Query Evaluation

Path of a query through a typical DBMS:



<< ^ >>

❖ A Relational Algebra Engine



<< ^ >>

❖ Installing/Using PostgreSQL

<< ^ >>

❖ Installing PostgreSQL

PostgreSQL is available via the COMP9315 web site.

Provided as tar-file in `~cs9315/web/25T1/postgresql/`

File: `postgresql-15.11.tar.bz2` is ~23MB **

Unpacked, source code + binaries is ~210MB **

If using on CSE, do not put it under your home directory

Place it under `/localstorage/YOU/` which has 600MB quota

<< ^ >>

❖ Before Installing ...

If you have databases from previous DB courses

- the databases will no longer work under v15.11
- to preserve them, use dump/restore

E.g.

```
... login to vxdb
... run your old server for the last time ...
$ pg_dump -O -x myFavDB > /localstorage/YOU/myFavDB.dump
... stop your old server for the last time ...
... remove data from your old server ...
$ rm -fr /localstorage/YOU/pgsql
... install and run your new PostgreSQL 15.11 server ...
$ createdb myFavDB
$ psql myFavDB -f /srvr/YOU/myFavDB.dump
... your old database is restored under 15.11 ...
```

<< ^ >>

❖ Installing/Using PostgreSQL

Environment setup for running PostgreSQL in COMP9315:

```
# Must be "source"d from sh, bash, ksh, ...

export PGDATA=/localstorage/$USER/pgsql/data
export PGHOST=$PGDATA
export LD_LIBRARY_PATH=/localstorage/$USER/pgsql/lib

export PATH=/localstorage/$USER/pgsql/bin:$PATH

alias p0="pg_ctl stop"
alias p1="pg_ctl -l $PGDATA/log start"
```

Will probably work (with tweaks) on home laptop if Linux or MacOS

<< ^ >>

❖ Installing/Using PostgreSQL (cont)

Brief summary of installation:

```
$ tar xfj ..../postgresql/src.tar.bz2
  # create a directory postgresql-11.3
  # set up environment variables
$ configure --prefix=$PGHOME
$ make
$ make install
$ source your/environment/file
$ initdb
  # set up postgresql configuration ... done once?
$ edit postgresql.conf
$ pg_ctl start -l $PGDATA/log
  # do some work with PostgreSQL databases
$ pg_ctl stop
```

❖ Exercise: Install PostgreSQL

Follow instruction from previous slide (or P01)

- install a PostgreSQL server
- run it without setting environment
- try to use it without running server
- try to close it while a job is running
- examine the files/directories under $PGDATA

<< ^ >>

❖ Using PostgreSQL for Assignments

If changes don't modify storage structures ...

```
$ edit source code
$ pg_ctl stop
$ make
$ make install
$ pg_ctl start -l $PGDATA/log
  # run tests, analyse results, ...
$ pg_ctl stop
```

In this case, existing databases will continue to work ok.

<< ^ >>

❖ Using PostgreSQL for Assignments (cont)

If changes modify storage structures ...

```
$ edit source code
$ save a copy of postgresql.conf
$ pg_dump testdb > testdb.dump
$ pg_ctl stop
$ make
$ make install
$ rm -fr $PGDATA
$ initdb
$ restore postgresql.conf
$ pg_ctl start -l $PGDATA/log
$ createdb testdb
$ psql testdb -f testdb.dump
  # run tests and analyse results
```

Old databases will not work with the new server.

≪ ∧

❖ Using PostgreSQL for Assignments (cont)

Troubleshooting ...

- read the $PGDATA/log file
- which socket file are you trying to connect to?
- check the $PGDATA directory for socket files
- remove postmster.pid if sure no server running
- ...

Prac Exercise P01 has useful tips down the bottom

Produced: 17 Feb 2025