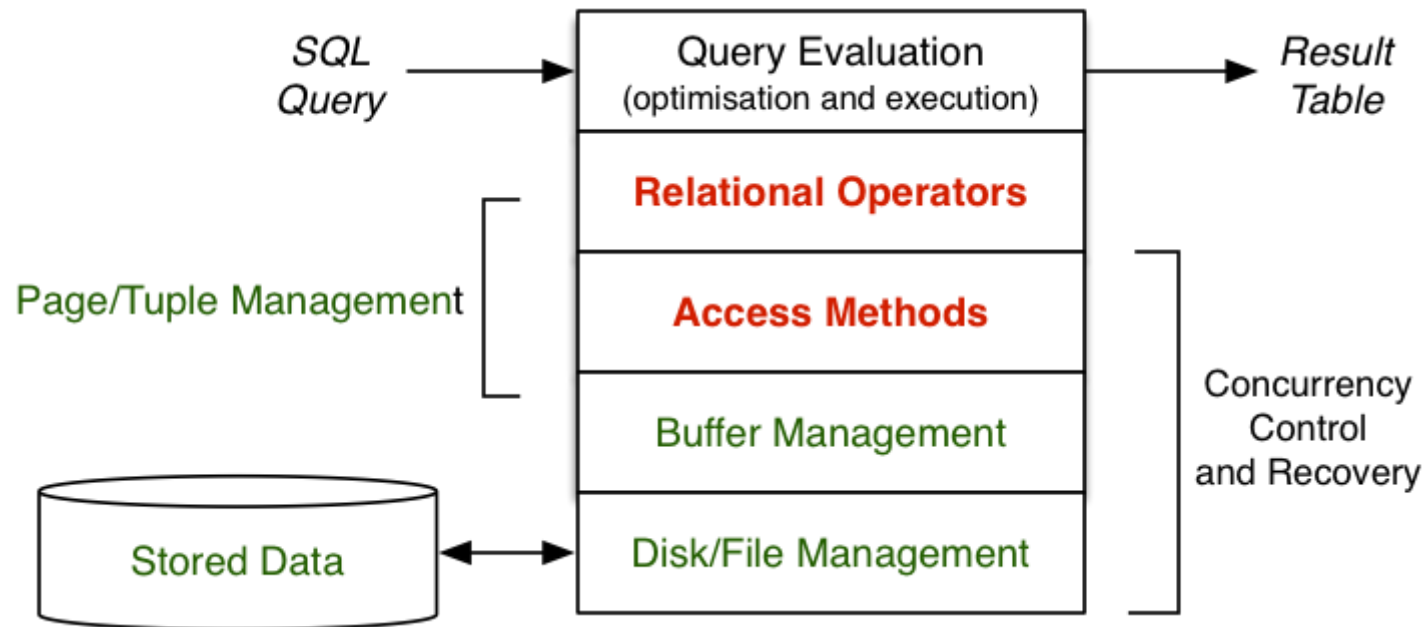


Relational Operations

- DBMS Architecture (revisited)
- Relational Operations
- Cost Models
- Query Types

❖ DBMS Architecture (revisited)

Implementation of relational operations in DBMS:



❖ Relational Operations

DBMS core = relational engine, with implementations of

- selection, projection, join, set operations
- scanning, sorting, grouping, aggregation, ...

In this part of the course:

- examine methods for implementing each operation
- develop cost models for each implementation
- characterise when each method is most effective

Terminology reminder:

- tuple = collection of data values under some schema \cong record
- page = block = collection of tuples + management data = i/o unit
- relation = table \cong file = collection of tuples

❖ Relational Operations (cont)

In order to implement relational operations the low-levels of the system provides:

- **Relation** `openRel(db, name)`
 - get handle on relation **name** in database **db**
- **Page** `request_page(rel, pid)`
 - get page **pid** from relation **rel**, return buffer containing page
- **Record** `get_record(buf, tid)`
 - return record **tid** from page **buf**
- **Tuple** `mkTuple(rel, rec)`
 - convert record **rec** to a tuple, based on **rel** schema

❖ Relational Operations (cont)

Example of using low-level functions

```
// scan a relation Emps
Page p; // current page
Tuple t; // current tuple
Relation r = relOpen(db, "Emps");
for (int i = 0; i < nPages(r); i++) {
    p = request_page(rel, i);
    for (int j = 0; j < nRecs(p); j++)
        t = mkTuple(r, get_record(p, j));
        ... process tuple t ...
    }
}
```

❖ Relational Operations (cont)

Two "dimensions of variation":

- which relational operation (e.g. Sel, Proj, Join, Sort, ...)
- which access-method (e.g. file struct: heap, indexed, hashed, ...)

Each **query method** involves an operator and a file structure:

- e.g. primary-key selection on hashed file
- e.g. primary-key selection on indexed file
- e.g. join on ordered heap files (sort-merge join)
- e.g. join on hashed files (hash join)
- e.g. two-dimensional range query on R-tree indexed file

We are interested in *cost* of query methods (and insert/delete operations)

❖ Relational Operations (cont)

SQL vs DBMS engine

- **select ... from R where C**
 - find relevant tuples (satisfying C) in file(s) of R
- **insert into R values(...)**
 - place new tuple in some page of a file of R
- **delete from R where C**
 - find relevant tuples and "remove" from file(s) of R
- **update R set ... where C**
 - find relevant tuples in file(s) of R and "change" them

❖ Cost Models

An important aspect of this course is

- analysis of cost of various query methods

Cost can be measured in terms of

- *Time Cost*: total time taken to execute method, or
- *Page Cost*: number of pages read and/or written

Primary assumptions in our cost models:

- memory (RAM) is "small", fast, byte-at-a-time
- disk storage is very large, slow, page-at-a-time

❖ Cost Models (cont)

Since *time cost* is affected by many factors

- speed of i/o devices (fast/slow disk, SSD)
- load on machine

we do not consider time cost in our analyses.

For comparing methods, *page cost* is better

- identifies workload imposed by method
- BUT is clearly affected by buffering

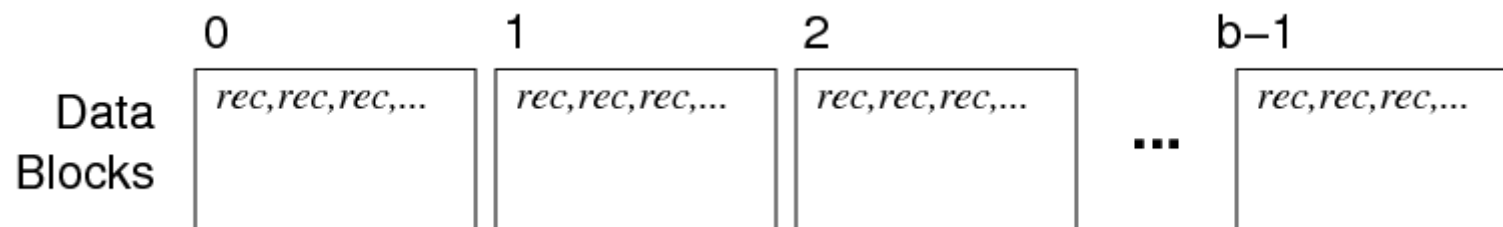
Estimating costs with multiple concurrent ops *and* buffering is difficult!!

Additional assumption: every page request leads to some i/o

❖ Cost Models (cont)

In developing cost models, we also assume:

- a relation is a set of r tuples, with average tuple size R bytes
- the tuples are stored in b data pages on disk
- each page has size B bytes and contains up to c tuples
- the tuples which answer query q are contained in b_q pages
- data is transferred disk \leftrightarrow memory in whole pages
- cost of disk \leftrightarrow memory transfer $T_{r/w}$ is very high



❖ Cost Models (cont)

Our cost models are "rough" (based on assumptions)

But do give an $O(x)$ feel for how expensive operations are.

Example "rough" estimation: how many piano tuners in Sydney?

- Sydney has $\approx 4\,000\,000$ people
- Average household size $\approx 3 \therefore 1\,300\,000$ households
- Let's say that 1 in 10 households owns a piano
- Therefore there are $\approx 130\,000$ pianos
- Say people get their piano tuned every 2 years (on average)
- Say a tuner can do 2/day, 250 working-days/year
- Therefore 1 tuner can do 500 pianos per year
- Therefore Sydney would need $\approx 130000/2/500 = 130$ tuners

Actual number of tuners in Yellow Pages = 120

Example borrowed from Alan Fekete at Sydney University.

COMP9315 21T1 ◇ Relational Operations ◇ [10/11]

❖ Query Types

Type	SQL	RelAlg	a.k.a.
Scan	select * from R	R	-
Proj	select x,y from R	$Proj[x,y]R$	-
Sort	select * from R order by x	$Sort[x]R$	<i>ord</i>
Sel ₁	select * from R where id = k	$Sel[id=k]R$	<i>one</i>
Sel _n	select * from R where a = k	$Sel[a=k]R$	-
Join ₁	select * from R,S where R.id = S.r	$R Join[id=r] S$	-

Different query classes exhibit different query processing behaviours.

Produced: 27 Feb 2021