>>

COMP9315 Week 2 Sessions

^ >>

❖ Things to Note

Course Notes

- more detailed than Slides ... need upgrading ... coming soon

Quiz 1

- due before Friday 23:59 ... so far 90/470 submissions

Dropping/Enrolling

- if you drop COMP9315, will need help enrolling in replacement

Unix skills

- Home Computing playlist on https://www.youtube.com/channel/UCi3Kf5eONIwV6QgNHiYqVzg

<< ^ >>

❖ Exercise: Relation Scan Cost

Consider a table *R(x,y,z)* with 10^5 tuples, implemented as

- number of tuples *r = 10,000*
- average size of tuples *R = 200* bytes
- size of data pages *B = 4096* bytes
- time to read one data page T_r *= 10msec*
- time to check one tuple *1 usec*
- time to form one result tuple *1 usec*
- time to write one result page T_r *= 10msec*

Calculate the total time-cost for answering the query:

```
insert into S select * from R where x > 10;
```

if 50% of the tuples satisfy the condition.

COMP9315 21T1 ♢ Week 2 Sessions ♢ [2/11]

<< ^ >>

❖ Exercise: PostgreSQL Files

In your PostgreSQL server

- examine the content of the **`$PGDATA/base`** directory
- find the directory containing the **`pizza`** database
- find the file in this directory for the **`People`** table
- examine the contents of the **`People`** file
- what are the other files in the directory?
- are there forks in any of your databases?

<< ^ >>

❖ Exercise: Buffer Cost Benefit (i)

Assume that:

- the **`Customer`** relation has b_C pages (e.g. 10)
- the **`Employee`** relation has b_E pages (e.g. 4)

Compute how many page reads occur ...

- if we have only 2 buffers (i.e. effectively no buffer pool)
- if we have 20 buffers
- when a buffer pool with MRU replacement strategy is used
- when a buffer pool with LRU replacement strategy is used

For the last two, buffer pool has $n=3$ slots ($n < b_C$ and $n < b_E$)

<< ^ >>

❖ Exercise: Buffer Cost Benefit (ii)

If the tables were larger, the above analysis would be tedious.

Write a C program to simulate buffer pool usage

- assuming a nested loop join as above
- **`argv[1]`** gives number of pages in "outer" table
- **`argv[2]`** gives number of pages in "inner" table
- **`argv[3]`** gives number of slots in buffer pool
- **`argv[4]`** gives replacement strategy (LRU,MRU,FIFO-Q)

<< ^ >>

❖ Exercise: Clock-sweep Page Replacement

Suing the following data type for buffer frame descriptors:

```
struct FrameDesc {
    Tag pid;     // ID of page in frame e.g. "R0"
    int pin;     // number tx's using this page
    int usage;   // clock-sweep usage counter
}
```

Show how the buffer pool changes for

- n = 4, b_R = 3, b_S = 4, b_T = 6
- when executing **`select * from T`** via sequential scan
- when executing **`select * from R join S`** using nested-loop join

<< ^ >>

❖ Exercise: Fixed-length Records

Give examples of table definitions

- which result in fixed-length records
- which result in variable-length records

```
create table R ( ...);
```

What are the common features of each type of table?

<< ^ >>

❖ Exercise: Inserting/Deleting Fixed-length Records

For each of the following Page formats:

- compacted/packed free space
- unpacked free space (with bitmap)

Implement

- a suitable data structure to represent a **`Page`**
- a function to insert a new record
- a function to delete a record

<< ^ >>

❖ Exercise: Inserting Variable-length Records

For both of the following page formats

1. variable-length records, with compacted free space
2. variable-length records, with fragmented free space

implement the **`insert()`** function.

Use the above page format, but also assume:

- page size is 1024 bytes
- tuples start on 4-byte boundaries
- references into page are all 8-bits (1 byte) long
- a function **`recSize(r)`** gives size in bytes

<< ^ >>

❖ Exercise: PostgreSQL Pages

Draw diagrams of a PostgreSQL heap page

- when it is initially empty
- after three tuples have been inserted
  with lengths of 60, 80, and 70 bytes
- after the 80 byte tuple is deleted (but before vacuuming)
- after a new 50 byte tuple is added

Show the values in the tuple header.

Assume that there is no special space in the page.

<< ^

❖ Exercise: Space Utilisation

Consider the following page/record information:

- page size = 1KB = 1024 bytes = 2^{10} bytes
- records: **`(a:int,b:varchar(20),c:char(10),d:int)`**
- records are all aligned on 4-byte boundaries
- **`c`** field padded to ensure **`d`** starts on 4-byte boundary
- each record has 4 field-offsets at start of record (each 1 byte)
- **`char(10)`** field rounded up to 12-bytes to preserve alignment
- maximum size of **`b`** values = 20 bytes; average size = 16 bytes
- page has 32-bytes of header information, starting at byte 0
- only insertions, no deletions or updates

Calculate *c* = average number of records per page.

Produced: 25 Feb 2021