EXAM

- 3 Dec 2007 (Monday), 7.30pm-9.30pm
- Venue: SR3A/B, COM1 Level 2
- Open book!
- Covers all lectures
- 40 marks
  - Section A **(L1 to L8)**
    - 20 marks
    - 4 open questions
  - Section B **(ALL lectures)**
    - 20 marks
    - 20 MCQs

Review (L1-L8)

Design

- Consider D(Dno,Head,Faculty) and E(Eno, Ename, Dno, Salary). Let D be horizontally partitioned into D1, D2 and D3 such that
  - D1 = D with Dname < 10;
  - D1 = D with 10 <= Dno <= 20;
  - D3 = D with Dno > 20.
- Let the horizontal fragmentation of E be derived from that of D, based on Dno.
- Given the following data, compute D1, D2, D3, E1, E2, E3.

Design

D

Dno	Head	Faculty
11	Jaffar	SoC
12	Wei	SoC
13	Lee	Engr
1	Tan	Sci
15	Ong	Sci
22	Brian	Busi
25	Hui	Engr
7	Lou	Sci

E

Eno	Ename	Dno	Salary
1	Toh	13	50000
2	Seah	11	60000
3	Lee	22	45000
4	Chu	15	50000
5	Tan	11	65000
6	Ang	1	35000
7	Chia	7	75000
8	Ali	13	60000

Design

D1

Dno	Head	Faculty
1	Tan	Sci
7	Lou	Sci

D2

Dno	Head	Faculty
11	Jaffar	SoC
12	Wei	SoC
13	Lee	Engr
15	Ong	Sci

D3

Dno	Head	Faculty
22	Brian	Busi
25	Hui	Engr

E1

Eno	Ename	Dno	Salary
6	Ang	1	35000
7	Chia	7	75000

E2

Eno	Ename	Dno	Salary
1	Toh	13	50000
2	Seah	11	60000
4	Chu	15	50000
5	Tan	11	65000
8	Ali	13	60000

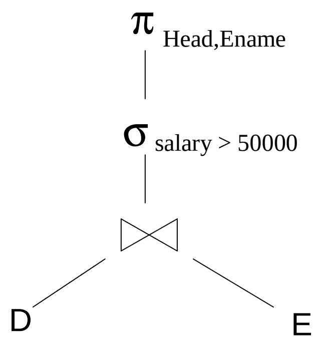
E3

Eno	Ename	Dno	Salary
3	Lee	22	45000

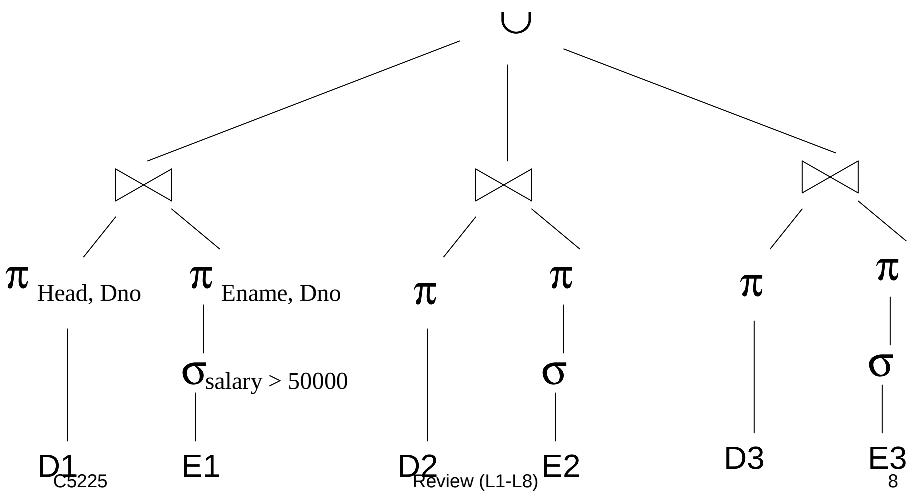
Query Processing

- SELECT Head, Ename FROM D, E WHERE D.Dno = E.Eno AND Salary > 50000. Push projections and selections as down as possible.

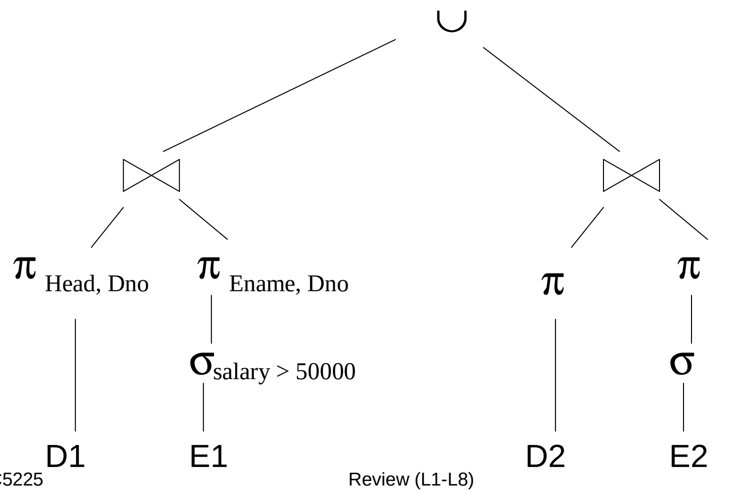
Query Processing



Query Processing



Query Processing



Concurrency Control

- Two sites A and B; x1 and x2 stored at A, x3 and x4 at B
- Two global transactions, T1 and T2
  - T1 = [r1(x1), w1(x1), r1(x3), w1(x3)]
  - T2 = [r2(x2), w2(x2), r2(x4), w2(x4)]
- Two local transactions, L3 and L4
  - L3 = [r3(x1), r3(x2)] at site A
  - L4 = [r4(x3), r4(x4)] at site B
- Two local schedules
  - SA = [r3(x1), r1(x1), w1(x1), r2(x2), w2(x2), r3(x2)]
  - SB = [r4(x3), r1(x3), w1(x3), r2(x4), w2(x4), r4(x4)]
- Are the schedules globally serializable?

– SA = [r3(x1), r1(x1), w1(x1), r2(x2), w2(x2), r3(x2)]
  - T2 < L3 < T1 where “<“ denotes “precedes”
– SB = [r4(x3), r1(x3), w1(x3), r2(x4), w2(x4), r4(x4)]
  - T2 < L4 < T1

Since both schedules are locally serializable and the order of global transactions are the same in the local schedules, the transactions are globally serializable.

Concurrency Control

- What if SB’ = [r1(x3), w1(x3), r2(x4), r4(x3), r4(x4), w2(x4)]?
  – T1 < L4 < T2
- The two global transactions are not serializable!

Locking

- Suppose there are 5 sites, each with a copy of a database element X. Suppose we adopt the ROWA protocol.
- Now, a node P generates 52% of all accesses – 50% of which are read-only accesses, and 2% are write accesses
- Each of the remaining node generates 10% of read and 2% of write.
- What is the average number of messages needed to obtain a lock?

- Since every site has a local copy, reads incur no communication/message cost
- For each write, the write action must be performed each site with a copy. Each write to each site incurs 3 messages: request, grant, release.
- Since there are 5 sites, it means a write request must be sent to 4 other sites, i.e., 12 messages needed per write.
- Average cost = 90%*0 + 10%*12 = 1.2

- What if we use primary-copy locking with P containing the primary copy?
  - Try it out!

Is the termination protocol correct?

- Global decision to abort, failed participant (FP) had committed?

Is the protocol correct?

- Global decision to abort, failed participant (FP) had committed?
  - FP can commit ONLY IF it receives global commit
  - This is ONLY possible if all the other operational participants had received and acknowledged global precommit
  - This means the decision cannot be abort

Is the protocol correct?

- Global decision to commit, failed participant (FP) had aborted?

Is the protocol correct?

- Global decision to commit, failed participant (FP) had aborted?
- To abort, FP must have aborted unilaterally prior to voting, voted abort, or received a global abort prior to failure
  - If it had voted commit and had received a global abort, then the global decision could not have been commit
  - If it had aborted, then there could not have been a global pre-commit and hence …

Parallel DBMS

- Given two tables
  - Emp(**eid**: int, did: int, sal: real)
  - Dept(**did**: int, mgrid: int, budget: int)
- Each table contains 20-byte tuples; sal and budget are uniformly distributed in the range 0 to 1 million.
- Emp contains 100,000 pages, Dept 5000
- Each processor has 100 4000-byte buffers
- Each I/O cost is td, cost to ship one page of data is ts.

Parallel DBMS (Cont)

- Data are initially partitioned using round-robin on 10 processors; no indexes.
- Assume hash-join.
- What's the *elapsed time* to answer the following queries:
  - Find the highest paid employee
  - Find the highest paid employee over all departments with budget less than 100,000

Query 1

- Evaluation strategy
  - All nodes read their portion of Emp
    - This incurs 10K pages of I/O = 10000*td (Since there are 100K pages, and there are 10 processors, and data are placed using round-robin strategy)
  - As each tuple is scanned, the highest salary is maintained
    - We ignore CPU cost (since question is silenced about it)
  - Once done, the page containing the tuple is sent to a coordinator (can assume to be an arbitrary node)
    - This incurs communication cost of ts
  - Assuming bandwidth is sufficient and it takes only ts to transmit the page from each node to the coordinator
    - Elapsed time = 10000*td + ts (if bandwidth is small, then we may need to incur at least 9*ts – assuming one of the processor is the coordinator)
  - Finally, the coordinator computes the highest salary from the data received
    - Ignore CPU cost.

Query 2

- Evaluation strategy
  - Build hash table for dept
    - Read dept
      - Cost = 500 td (5000 pages, 10 processors)
    - 10% of the data have budget < 100,000
      - Cpu ignored
      - Communication cost
        - » The qualified tuples have to be partitioned and shipped to the respective processor
        - » Since data are uniformly distributed, about 10% of the qualified tuples will remain in the same node, while 90% will be transmitted to other processors
        - » Comm cost = 9/10*50 ts
    - At receiving processor, build the hash table
      - There is no need to store the arriving tuples of Dept since the memory of each processor is 100 pages – more than enough to contain the hash tables for dept partition (only 50 pages)
      - Only cpu cost – so ignored

Query 2 (Cont)

- Evaluation strategy
  - Probe hash table with emp
    - Read emp
      - I/O cost = 10,000 td (100,000 pages, 10 processors)
  - As each tuple is scanned, they are partitioned and transmitted to the relevant processor
    - Comm cost = 9/10*10,000 ts
  - As tuples are received at the corresponding processor
    - There is no need to store the tuples; instead, they are used to probe the hash table immediately
    - As results are generated, only the highest salary tuple need to be retained; so join result also do not need to be stored
    - All cpu cost - ignored
  - Finally, one tuple is transmitted to coordinator

Query 2 (Cont)

- Evaluation Strategy
  - Coordinator receives a tuple from each node
  - Only cost is comm (same as Query 1)
- So elapsed time = I/O + Comm Cost

Parallel DBMS (Cont)

- What if data were initially range or hash partitioned?
- 100 processors?
- Different join methods?
  - Try them!