

# Architecture and Implementation of Database Systems – Exercises

Teaching Assistant: Marcos Antonio Vaz Salles

[marcos.vazsalles@inf.ethz.ch](mailto:marcos.vazsalles@inf.ethz.ch)

<http://www.dbis.ethz.ch/people/vmarcos>

Institute of Information Systems



03.Feb.2006

## Overview of Today's Class

- Third Programming Task: Correction
- Last Theoretical Exercise: Correction

## Third Programming Task: Correction Criteria

- Implementation of in-memory hash join (30%)
  - 21%: produce the right results with the right algorithm
  - 2%: documentation
  - 8%: coding style
    - Nice hash function – hashCode ok
    - Keep tuple list in the hash map to produce duplicates

## Third Programming Task: Correction Criteria

- What are outer joins and what are they good for? (10%)
- Extension of in-memory hash join to process outer joins (20%)
  - 14%: produce the right results with the right algorithm
  - 1%: documentation
  - 5%: coding style
    - Annotate hash map entries with matching information
    - Produce output tuples anyway for outer tuple in right, full cases

## Third Programming Task: Correction Criteria

- Grace Hash Join
  - 28%: produce the right results with the right algorithm
  - 2%: documentation
  - 10%: coding style
    - Nice hash function for partitioning (e.g., hashCode mod W or first\_random mod W)
    - No writing of join results to disk, only of partitions
    - Use SimpleHashOuterJoin to join partition pairs

## Last Theoretical Task: Correction

- Exercise 1 – What is the process a SQL query goes through from submission until execution is started?
  - Query rewriting
    - Predicate expansion
    - Unnesting of views/subqueries
    - Heuristic transformations (projection and selection push-down, cross-product elimination, etc)
    - Query blocks

## Last Theoretical Task: Correction

- Exercise 1 – What is the process a SQL query goes through from submission until execution is started?
  - Example: 

```
SELECT p.id, p.name
FROM patient p
WHERE p.sex = 'male' AND
      EXISTS ( SELECT *
                FROM medical_records r
                WHERE r.patient_id = p.id AND
                      r.date < '02.02.1992' AND
                      ( r.diagnosis = 'malaria' OR
                        r.diagnosis = 'smallpox' ) )
```

## Last Theoretical Task: Correction

- Exercise 1 – What is the process a SQL query goes through from submission until execution is started?
  - Cost-based optimization
    - Use of cost model to estimate selectivity and cost for physical operators (access paths and join methods)
    - Determine join order

## Last Theoretical Task: Correction

- Exercise 2 – Force / No Force, Steal / No Steal

**¬force ⇔ redo**  
**steal ⇔ undo**

## Last Theoretical Task: Correction

- Exercise 2 – Force / No Force, Steal / No Steal
  - a. When a force and no-steal strategy is employed, one must implement a scheme for the redo of operations from committed transactions whose changes have not yet been written to the database, but there is no need for a scheme for the undo of operations from uncommitted transactions that have already updated the database.
  - b. When a no-force and no-steal strategy is employed, one must implement a scheme for the undo of operations from uncommitted transactions that have already updated the database, but there is no need for a scheme for the redo of operations from committed transactions whose changes have not yet been written to the database.
  - c. When a force and steal strategy is employed, there is no need to implement schemes for the redo of operations from committed transactions whose changes have not yet been written to the database or for the undo of operations from uncommitted transactions that have already updated the database.
  - d. When a no-force and steal strategy is used, one must implement schemes both for the redo of operations from committed transactions whose changes have not yet been written to the database and for the undo of operations from uncommitted transactions that have already updated the database.

## Last Theoretical Task: Correction

- Exercise 3 – WAL
  - a. the writing of a data item should be done ahead of any logging operation.
  - b. the log record for an operation should be written before the actual data is written.
  - c. all log records should be written before a new transaction begins execution.
  - d. the log never needs to be written to disk.
- NOTE: also means that by commit all log entries should be written to stable storage

## Last Theoretical Task: Correction

- Exercise 4 – Properties of physical logging operations
  - a. commutative.
  - b. associative.
  - c. idempotent.
  - d. distributive.
- Why is that important?

## Last Theoretical Task: Correction

- Exercise 5: Serializability
  - Schedule 1
    - R(T1, A); W(T1, A); R(T2, A); R(T3, A); W(T2, B); C(T1); C(T2); C(T3)
  - Schedule 2
    - R(T1, A); W(T1, A); R(T3, C); W(T3, C); R(T2, A); R(T2, B); R(T3, B); W(T3, B); R(T1, C); W(T1, C); C(T1); C(T2); C(T3)
  - Schedule 3
    - R(T2, C); R(T2, B); W(T2, B); R(T3, B); R(T3, C); R(T1, A); W(T1, A); W(T3, B); W(T3, C); R(T2, A); R(T1, B); W(T1, B); W(T2, A); C(T1); C(T2); C(T3)
- Only schedule 1 is (conflict) serializable (a)

Questions?