DATABASDESIGN FÖR INGENJÖRER - 1DL124

Sommar 2007

En introduktionskurs i databassystem

http://user.it.uu.se/~udbl/dbt-sommar07/
alt. http://www.it.uu.se/edu/course/homepage/dbdesign/st07/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Active Database Technology

Elmasri/Navathe ch 24.1
Padron-McCarthy/Risch ch 15

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Active Databases

• General principles of Conventional Database Systems
Conventional (Passive) Databases

- Data model
- Transaction model
  - ACID principle
- Examples of real world problems:
  - Inventory control
    - reordering of items when quantity in stock falls below threshold.
  - Travel waiting list
    - book ticket as soon as right kind is available
  - Stock market
    - buy/sell stocks when price below/above threshold
  - Maintenance of master tables (view materialization)
    - maintain table that contain sum of salaries for each department
Conventional Databases

- In a *passive* database system, the application will periodically poll the DBMS:
  - Frequent polling $\Rightarrow$ expensive
  - Infrequent polling $\Rightarrow$ might miss the right time to react
  - The DBMS does not know that application is polling
Active Databases

• In an active database system, the DBMS recognize predefined situations in database, application or environment
• Trigger predefined actions when situations occur, such as DB operations, program executions

T1: 25 copies of Elmasri/Navathe sold

Order 10 copies

ADBMS

Records

Rules

on update of Sales when quantity<5 order 100 copies
Active Databases

• The general idea is that ADBMS provides:
  – Regular DBMS primitives
  – + definition of application-defined situations
  – + triggering av application-defined reactions
Active Databases

• Possible applications of ADBMS:
  • Consistency enforcement
    – Reaction to violations
      • e.g. ROLLBACK when constraint violated
    – Connection to time
      • e.g. check some constraint violations every midnight
  • Computation of derived data
    – View materialization of derived data
      • e.g. incremental recomputation of view of sum of salaries per department, salsum(dno,salary), computed from employee(ssn,dno,salary)
    – … or invalidation of materialized view when relevant update (e.g. ssalary) occurs => rematerialize view when accessed next time
  • Automatic travel booking
    – Order currently not available ticket with desired properties wherever one gets available
Active databases - rule models

- ECA (event-condition-action) rules:
  - Semantics of ECA rules (most common model presently)
    - ECA: WHEN event occurs - IF condition holds - DO execute action
  - Event:
    - Update of database record(s)
    - Parameterised using pseudo tables OLD and NEW
  - Condition:
    - Query on database state, e.g. a database query
    - empty result => condition is FALSE / non-empty result => condition is TRUE
  - Action:
    - Database update statement(s)
    - Stored procedure execution
- Unconditioned EA (event-condition) rules, as in example:
  - ON ... DO
  - Natural in C++/Java based object stires
  - Can make arbitrary C++/Java test in DO part!
- CA (condition-action) rules:
  - Not common in databases
Active databases

- Active DBMSs embed situation-action rules in database
- Support many functionalities:
  - e.g. integrity control, access control, monitoring, derived data, change notification
- Some ADBMS functionality commercially available as triggers: Sybase, Oracle, Interbase, etc.
Triggers

- A **trigger** is a statement that is executed automatically by the system as a side effect of a modification to the database.

- To design a trigger mechanism, we must:
  - Specify the conditions under which the trigger is to be executed.
  - Specify the actions to be taken when the trigger executes.

- The SQL-92 standard does not include triggers, but many implementations support triggers. SQL:99 has specified triggers.
Trigger example

• Suppose that instead of allowing negative account balances, the bank deals with overdrafts by
  – setting the account balance to zero
  – creating a loan in the amount of the overdraft
  – giving this loan a loan number identical to the account number of the overdrawn account
• New values after an update are represented by the keyword new and old values by the keyword old.
• The condition for executing the trigger is a check to see if the balance after an update to the account relation (represented by new.balance) results in a negative balance.
Trigger example cont’d

@

create trigger overdraft after update on account
referencing new row as n
for each row
when (n.balance < 0)
begin atomic
    insert into loan
        values(n.account-number, n.branch-name, - n.balance);
    insert into borrower
        select customer-name, account-number
        from depositor d
        where n.account-number = d.account-number;
    update account a set a.balance = 0
        where a.account-number = n.account-number;
end;
@

@
Active databases - example

employee(ssn, dno, salary)
dept(dno, mgrssn)
salsum(dno, total)  (<- Materialized)

@
create trigger employee after update on employee  <--- Event
referencing new table as n old table as o  <--- No condition
for each statement
begin atomic
  update salsum s
  set total = total - o.salary
  where s.dno = o.dno;
  update salsum s
  set total = total + n.salary
  where s.dno = n.dno
end;
@

Kjell Orsborn  8/7/07
Active databases - example
(triggers as constraints)

@
create trigger salary_constraint
after update of salary on employee
when new.salary >
(select m.salary
from employee m,department d
where new.dno = d.dno and d.mgr = m.ssn)
begin
update employee e
set salary = old.salary
end;
@

UU - IT - UDBL
Active databases - example

@  
CREATE TRIGGER SALARY_CONSTRAINT2
   AFTER UPDATE ON EMPLOYEE
   WHEN NEW.SALARY <
       (SELECT E.SALARY
        FROM EMPLOYEE E, DEPARTMENT D
        WHERE E.DNO = D.DNO AND D.MGR = M.SSN)
   BEGIN
       ROLLBACK
   END;

@
Active Databases

• Cautions:
  • Very powerful mechanism:
    – Small statement => massive behaviour changes.
    – Rope for programmer.
    – Requires careful design
    – Activity design + traditional database design.
• Trace consequences of rule specification/changes:
  – Make sure indefinite triggering cannot happen.
• Help user design meaningful rules:
  – Tool, e.g. simulator
• Higher abstraction level needed for ordinary users:
  – Assertions.
  – Specification of materialized views.
  – Graphical tools.
Stored proc. vs. triggers

- Triggers are used when one should monitor updates of tables where one do not who, or how, the table will be updated. Should be used with carefulness!!
- Stored procedures are used to update tables where one knows that the table update will allways be done through the procedure instead of through a direct update.
- Stored procedures are precompiled and can be used as efficient views (views are normally not optimized before they are refererenced), but are usually used for updates.
Active Databases

• Workflow Control Application
• Goal: Control long running activity
  – multiple application steps
  – may be of long duration
  – may be executed by different servers in a distributed system
• Issues:
  – how to sequence the steps (workflow)?
  – if a step fails, committed earlier steps cannot be rolled back: how to compensate?
• Approach: use triggers/rules
  – model application steps by transactions
  – use rules to check constraints, chain steps, and handle exceptions
  – flexible response to exceptions, rather than a fixed compensation policy
Active Databases

- Advanced level SQL-92 (SYBASE, ORACLE) has assertions too:

```sql
CREATE ASSERTION SALARY_CONSTRAINT
CHECK(NOT EXISTS
(SELECT *
FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D
WHERE E.SALARY > M.SALARY AND
E.DNO = D.DNO AND
D.MGRD = M.SSN))
```
Trigger example cont’d

create trigger overdraft for account before update as
begin
  if (new.balance < 0) then
    begin
      insert into loan values
        (branch-name, old.account-number, new.balance);
      insert into borrower
        (select customer-name, account-number
         from depositor
         where old.account-number = depositor.account-number);
      update account S set S.balance = 0
        where S.account-number = old.account-number;
    end
  end
end
Active databases - example

- Example in Oracle syntax
  - EMPLOYEE(SSN, DNO, SALARY)
  - DEPT(DNO, MGRSSN)
  - SALSUM(DNO, TOTAL) <= Materialized
  - CREATE TRIGGER EMPLOYEE_SALARY_MATERIALIZATION
    AFTER UPDATE ON EMPLOYEE
    FOR EACH ROW
    BEGIN
      UPDATE SALSUM S
      SET TOTAL = TOTAL - OLD.SALARY
      WHERE S.DNO = OLD.DNO
    UPDATE SALSUM S
      SET TOTAL = TOTAL + NEW.SALARY
      WHERE S.DNO = NEW.DNO
    END;
Active databases - example

- Example of triggers for constraints, Oracle

- `CREATE TRIGGER SALARY_CONSTRAINT
  AFTER UPDATE OF SALARY ON EMPLOYEE  
  WHEN NEW.SALARY > 
  (SELECT M.SALARY  
   FROM EMPLOYEE M,DEPARTMENT D  
   WHERE NEW.DNO = D.DNO AND D.MGR = M.SSN) 
  BEGIN 
  UPDATE EMPLOYEE E  
  SET SALARY = OLD.SALARY
  END;`
Active databases - example

- NOTICE! SALARY_CONSTRAINT needed for managers:

- CREATE TRIGGER SALARY_CONSTRAINT2
  AFTER UPDATE ON EMPLOYEE
  WHEN NEW.SALARY <
    (SELECT E.SALARY
     FROM EMPLOYEE E, DEPARTMENT D
     WHERE E.DNO = D.DNO AND D.MGR = M.SSN)
  BEGIN
    ROLLBACK
  END;

- NOTICE! SALARY_CONSTRAINT needed for departments too!
- NOTICE! SALARY_CONSTRAINT needed for insertions too!
- Solution: Integrity constraints!