Distributed Database

Munawar, PhD

Semantic Integrity Control
Sentralized & Distributed

Semantic Data Control

Involves Semantic Data Control ;

- View management
- Security control
- Integrity control

View Management

View – virtual relation

- generated from base relation(s) by a query
- not stored as base relations

Example :

```
CREATE VIEW   SYSAN(ENO,ENAME)
AS            SELECT    ENO,ENAME
              FROM      EMP
              WHERE     TITLE= "Syst. Anal."
```

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

SYSAN

ENO	ENAME
E2	M.Smith
E5	B.Casey
E8	J.Jones

View Management

Views can be manipulated as base relations

Example :

```
SELECT    ENAME, PNO, RESP
FROM      SYSAN, ASG
WHERE     SYSAN.ENO = ASG.ENO
```

Query Modification

Queries expressed on views

Queries expressed on base relations

Example :

```
SELECT ENAME, PNO, RESP
FROM   SYSAN, ASG
WHERE  SYSAN.ENO = ASG.ENO

SELECT ENAME,PNO,RESP
FROM   EMP, ASG
WHERE  EMP.ENO = ASG.ENO
AND    TITLE = "Syst. Anal."
```

ENAME	PNO	RESP
M.Smith	P1	Analyst
M.Smith	P2	Analyst
B.Casey	P3	Manager
J.Jones	P4	Manager

View Management

- To restrict access

```
CREATE  VIEW    ESAME
AS      SELECT  *
        FROM    EMP E1, EMP E2
        WHERE   E1.TITLE = E2.TITLE
        AND     E1.ENO = USER
```

- Query

```
SELECT  *
FROM    ESAME
```

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	L. Chu	Elect. Eng

View Updates

- Updatable

```
CREATE   VIEW     SYSAN(ENO,ENAME)
AS       SELECT   ENO,ENAME
         FROM     EMP
         WHERE    TITLE="Syst. Anal."
```

- Non-updatable

```
CREATE   VIEW     EG(ENAME,RESP)
AS       SELECT   ENAME,RESP
         FROM     EMP, ASG
         WHERE    EMP.ENO=ASG.ENO
```

Data Security

- **Data protection**
  - Prevents the physical content of data to be understood by unauthorized users
  - Uses encryption/decryption techniques (Public key)
- **Access control**
  - Only authorized users perform operations they are allowed to on database objects
  - Discretionary access control (DAC)
    - Long been provided by DBMS with authorization rules
  - Multilevel access control (MAC)
    - Increases security with security levels

Discretionary Access Control

- **Main actors**
  - Subjects (users, groups of users) who execute operations
  - Operations (in queries or application programs)
  - Objects, on which operations are performed
- **Checking whether a subject may perform an op. on an object**
  - Authorization= (subject, op. type, object def.)
  - Defined using GRANT OR REVOKE
  - Centralized: one single user class (admin.) may grant or revoke
  - Decentralized, with op. type GRANT
    - More flexible but recursive revoking process which needs the hierarchy of grants

Multilevel Access Control

- **Different security levels (*clearances*)**
  - *Top Secret* > *Secret* > *Confidential* > *Unclassified*
- **Access controlled by 2 rules:**
  - No read up
    - subject *S* is allowed to read an object of level *L* only if *level(S)* ≥ *L*
    - Protect data from unauthorized disclosure, e.g. a subject with secret clearance cannot read top secret data
  - No write down:
    - subject *S* is allowed to write an object of level *L* only if *level(S)* ≤ *L*
    - Protect data from unauthorized change, e.g. a subject with top secret clearance can only write top secret data but not secret data (which could then contain top secret data)

Distributed Access Control

- **Additional problems in a distributed environment**
  - Remote user authentication
    - Typically using a directory service
      - Should be replicated at some sites for availability
  - Management of DAC rules
    - Problem if users' group can span multiple sites
      - Rules stored at some directory based on user groups location
      - Accessing rules may incur remote queries
  - Covert channels in MAC

Semantic Integrity Control

Maintain database consistency by enforcing a set of constraints defined on the database.

- **Structural constraints**
  - basic semantic properties inherent to a data model e.g., unique key constraint in relational model
- **Behavioral constraints**
  - regulate application behavior, e.g., dependencies in the relational model
- **Two components**
  - Integrity constraint specification
  - Integrity constraint enforcement

Semantic Integrity Control

- **Procedural**

  control embedded in each application program

- **Declarative**

  assertions in predicate calculus

  - easy to define constraints
  - definition of database consistency clear
  - inefficient to check assertions for each update
    - limit the search space
    - decrease the number of data accesses/assertion
    - preventive strategies
    - checking at compile time

Thank You !

Munawar, PhD