UNIT-5

Semantic Data Control

Outlines…

- Introduction of Semantic Data Control
- View Management
- Authentication Control
- Semantic Integrity Control
- Cost of Enforcing Semantic Integrity

Data Security

- Data security is an important function of a database system that protects data against unauthorized access.
- Data security includes two aspects: **data protection** and **access control**.
- **Data protection** is required to prevent unauthorized users from understanding the physical content of data.
- The main data protection approach is data encryption.
- Access control must be refined so that different users have different rights on the same database objects.
- There are two main approaches to database access control **discretionary** (or authorization control) **mandatory or multilevel**.

Discretionary Access Control

- Three main actors are involved in discretionary access control.

1. The **subject** (e.g., users, groups of users) who trigger the execution of application programs.
2. **Operations**, which are embedded in application programs.
3. The **database objects**, on which the operations are performed.

Authorization control

- Authorization control consists of checking whether a given triple (subject, operation, object) can be allowed to proceed.
- The introduction of a subject in the system is typically done by a pair (user name, password).
- The objects to protect are subsets of the database. Relational systems provide finer and more general protection granularity than do earlier systems.
- A right expresses a relationship between a subject and an object for a particular set of operations.

GRANT <operation type(s)> ON <object> TO <subject(s)>

REVOKE <operation type(s)> FROM <object> TO <subject(s)>

Multilevel Access Control

- Discretionary access control has some limitations. One problem is that a malicious user can access unauthorized data through an authorized user.
- For instance, consider user A who has authorized access to relations R and S and user B who has authorized access to relation S only. If B somehow manages to modify an application program used by A so it writes R data into S, then B can read unauthorized data without violating authorization rules.
- Multilevel access control answers this problem and further improves security by defining different security levels for both subjects and data objects.

Multilevel Access Control

- process has a security level also called clearance derived from that of the user.
- In its simplest form, the security levels are Top Secret (TS), Secret (S), Confidential (C) and Unclassified (U), and ordered as TS > S >C >U, where ">" means "more secure".
- Access in read and write modes by subjects is restricted by two simple rules:

Rule 1 (called "no read up")

- protects data from unauthorized disclosure, i.e., a subject at a given security level can only read objects at the same or lower security levels.

Rule 2 (called "no write down")

- protects data from unauthorized change, i.e., a subject at a given security level can only write objects at the same or higher security levels.

Distributed Access Control

- The additional problems of access control in a distributed environment stem from the fact that objects and subjects are distributed and that messages with sensitive data can be read by unauthorized users.
- These problems are: remote user authentication, management of discretionary access rules, handling of views and of user groups, and enforcing multilevel access control.
- Remote user authentication is necessary since any site of a distributed DBMS may accept programs initiated, and authorized, at remote sites.

Distributed Access Control

- Three solutions are possible for managing authentication

1. Authentication information is maintained at a central site for global users which can then be authenticated only once and then accessed from multiple sites.
2. The information for authenticating users (user name and password) is replicated at all sites in the catalog.
3. Intersite communication is thus protected by the use of the site password. Once the initiating site has been authenticated, there is no need for authenticating their remote users.

Semantic Integrity Control

- Another important and difficult problem for a database system is how to guarantee database consistency.
- A database state is said to be consistent if the database satisfies a set of constraints, called semantic integrity constraints.
- Maintaining a consistent database requires various mechanisms such as concurrency control, reliability, protection, and semantic integrity control, which are provided as part of transaction management.
- **Semantic integrity control ensures database consistency by rejecting update transactions that lead to inconsistent database states, or by activating specific actions on the database state, which compensate for the effects of the update transactions.**

Semantic Integrity Control

- Two main types of integrity constraints can be distinguished: structural constraints and behavioral constraints.
- Structural constraints express basic semantic properties inherent to a model. Examples of such constraints are unique key constraints in the relational model, or one-to-many associations between objects in the object-oriented model.
- Behavioral constraints are essential in the database design process. They can express associations between objects, such as inclusion dependency in the relational model, or describe object properties and structures.

Centralized Semantic Integrity Control

- **Specification of Integrity Constraints**
- triggers (event-condition-action rules) can be used to automatically propagate updates, and thus to maintain semantic integrity.
- We can distinguish between three types of integrity constraints: **predefined**, **precondition**, or **general constraints**.
- EMP(ENO, ENAME, TITLE)
- PROJ(PNO, PNAME, BUDGET)
- ASG(ENO, PNO, RESP, DUR)

Centralized Semantic Integrity Control

- **Predefined constraints** are based on simple keywords. Through them, it is possible to express concisely the more common constraints of the relational model, such as non-null attribute, unique key, foreign key, or functional dependency.
- Employee number in relation EMP cannot be null.

ENO NOT NULL IN EMP

- The project number PNO in relation ASG is a foreign key matching the primary key PNO of relation PROJ.

PNO IN ASG REFERENCES PNO IN PROJ

Centralized Semantic Integrity Control

- **Precondition constraints** express conditions that must be satisfied by all tuples in a relation for a given update type. The update type, which might be INSERT, DELETE, or MODIFY, permits restricting the integrity control.
- Precondition constraints can be expressed with the SQL CHECK statement enriched with the ability to specify the update type.

CHECK ON <relation name > WHEN <update type>

(<qualification over relation name>)

- The budget of a project is between 500K and 1000K.

CHECK ON PROJ (BUDGET+ >= 500000 AND BUDGET <= 1000000)

- Only the tuples whose budget is 0 may be deleted.

CHECK ON PROJ WHEN DELETE (BUDGET = 0)

Centralized Semantic Integrity Control

- **General constraints** are formulas of tuple relational calculus where all variables are quantified. The database system must ensure that those formulas are always true.

CHECK ON list of <variable name>:<relation name>,(<qualification>)

- The total duration for all employees in the CAD project is less than 100.
- **CHECK ON g:ASG, j:PROJ (SUM(g.DUR WHERE g.PNO=j.PNO)<100 IF j.PNAME="CAD/CAM")**

Distributed Semantic Integrity Control

- **Definition of Distributed Integrity Constraints**
- Assertions can involve data stored at different sites, the storage of the constraints must be decided so as to minimize the cost of integrity checking. There is a strategy based on a taxonomy of integrity constraints that distinguishes three classes:
- **Individual constraints**: single-relation single-variable constraints. They refer only to tuples to be updated independently of the rest of the database.
- **Set-oriented constraints**: include single-relation multivariable constraints such as functional dependency and multirelation multivariable constraints such as foreign key constraints
- **Constraints involving aggregates**: require special processing because of the cost of evaluating the aggregates.

Individual constraints

- Consider relation EMP, horizontally fragmented across three sites using the predicates and the domain constraint C: ENO < "E4".
- p1 : 0 ENO < "E3"
- p2 : "E3" ENO "E6"
- p3 : ENO > "E6"
- Constraint C is compatible with p1 (if C is true, p1 is true) and p2 (if C is true, p2 is not necessarily false), but not with p3 (if C is true, then p3 is false). Therefore, constraint C should be globally rejected because the tuples at site 3 cannot satisfy C, and thus relation EMP does not satisfy C.

Set-oriented constraints.

- Set-oriented constraint are multivariable; that is, they involve join predicates.
- Three cases, given in increasing cost of checking, can occur:

1. The fragmentation of R is derived from that of S based on a semi join on the attribute used in the assertion join predicate.
2. S is fragmented on join attribute.
3. S is not fragmented on join attribute.

Set-oriented constraints.

- In the first case, compatibility checking is cheap since the tuple of S matching a tuple of R is at the same site.

- In the second case, each tuple of R must be compared with at most one fragment of S, because the join attribute value of the tuple of R can be used to find the site of the corresponding fragment of S.

- In the third case, each tuple of R must be compared with all fragments of S. If compatibility is found for all tuples of R, the constraint can be stored at each site.

Constraints involving aggregates

- These constraints are among the most costly to test because they require the calculation of the aggregate functions.
- The aggregate functions generally manipulated are MIN, MAX, SUM, and COUNT.
- Each aggregate function contains a projection part and a selection part.