BELGIUM CAMPUS
iTversity
It's the way we're wired

DATABASE DEVELOPMENT

ARE YOU READY?

DBD371/381

DATA AND ACCESS CONTROL

Semantic Data Control

LEARNING OBJECTIVES

BELGIUM CAMPUS iTversity
It's the way we're wired

- Semantic Data Control
- View Management
- Data Security
- Semantic Integrity Control

DATA AND ACCESS CONTROL

- A DBMS has the ability to support semantic data control (data and access control using high-level semantics)
- Semantic data control includes
  - ❑ view management,
  - ❑ security control,
  - ❑ semantic integrity control.
- These functions ensure authorized users perform correct operations on the database, making sure database integrity is maintained.
- Violation of some rules by a user program implies the rejection of the effects of that program or propagating some effects to preserve the database integrity.
- Cost of enforcing semantic data control, is high in terms of resource utilization in a centralized DBMS,
- In the Ddbase this can be prohibitive since these must be stored in a catalog (distributed directory) which in itself is a distributed database and managing this can be a challenge.

DATA AND ACCESS CONTROL

- Ways to store semantic data control definitions:
  - ❑ According to the way the directory is managed.
  - ❑According to its type; i.e, either fully replicated or distributed.

VIEW MANAGEMENT

- The relational model provides full logical data independence.
- This is an advantage because external schemas enable user groups to have their own view of the database.
- A view is a virtual relation, defined as the result of a query on base relations
- A view is a dynamic window in the sense that it reflects all updates to the database
- views hide some data and therefore secure.
- In a distributed DBMS, a view can be derived from distributed relations, and the access to a view requires the execution of the distributed query corresponding to the view definition

VIEWS IN CENTRALIZED DBMSS

- A view is a relation derived from base relations, as a result of the relational query
- This relation is defined by associating the name of the view with the retrieval query that specifies it.
- Eg view of system analysts (SYSAN) derived from relation EMP
- (ENO,ENAME,TITLE)

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

- The effect of this statement is the storage of the view definition and no other information needs to be recorded.
- The result of the query defining the view is not produced.
- The view can be manipulated as a base relation.

SYSAN

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

View Management

- Views can be defined using complex relational queries involving selection,projection, join and aggregate functions
- All views can be interrogated as base relations, but not all views can be manipulated as such (eg updates)
- Updates through views can be handled automatically only if they can be propagated correctly to the base relations.
- In a Centralized relational database we classify views as being updatable and not updatable
- A view is updatable only if
    1. the updates to the view can be propagated to the base relations without ambiguity.
    2. they are derived from a single relation by selection and projection.
    3. Views derived by join are updatable if they include the keys of the base relations.

Views in Distributed DBMSs

1. The definition of a view is similar in a distributed DBMS and in centralized systems
2. A view in a distributed system may be derived from fragmented relations stored at different sites.
3. When a view is defined, its name and its retrieval query are stored in the catalog.
4. View definition should be stored in the directory(catalog) in the same way as the base relation descriptions
5. View definitions can be centralized at one site, partially duplicated, or fully duplicated.
6. The information associating a view name to its definition site should be duplicated.
7. If the view definition is not present at the site where the query is issued, remote access to the view definition site is necessary
8. The qualification defining the view is found in the distributed database catalog and then merged with the query to provide a query on base relations.

Views in Distributed DBMSs

- The query processor maps the distributed query into a query on physical fragments.
- Evaluating views derived from distributed relations may be costly.(many users may access the same view which must be recomputed for each user.)
- An alternative solution is to avoid view derivation is by maintaining actual versions of the views, called materialized views.

Views in Distributed DBMSs

MATERIALIZED VIEWS

1. A materialized view stores the tuples of a view in a database relation, like the other database tuples, possibly with indices
2. Access to a materialized view is much faster than deriving the view, in particular, in a distributed DBMS where base relations can be remote.
3. Materialized views come in handy in the context of data warehousing to speed up On Line Analytical Processing (OLAP) applications because they provide and store compact database summaries.(Involving aggregate and grouping operators (sum,count and group by)

Views in Distributed DBMSs

- view over relation PROJ(PNO,PNAME,BUDGET,LOC) gives, for each location, the number of projects and the total budget.

```
CREATE VIEW PL(LOC, NBPROJ, TBUDGET)
AS SELECT LOC, COUNT(*),SUM(BUDGET)
FROM PROJ
GROUP BY LOC
```

Maintenance of Materialized Views

- A materialized view is a copy of some base data and thus must be kept consistent with that base data which may be updated.
- View maintenance is the process of updating (or refreshing) a materialized view to reflect the changes made to the base data.
- Issues related to view materialization are to some extent similar to those of database replication but differ in that:
  1. Materialized view expressions, are more complex than replica definitions and may include join, group by and aggregate operators.
  2. Database replication is concerned with more general replication configurations, e.g., with multiple copies of the same base data at multiple sites.

Maintenance of Materialized Views

- A view maintenance policy allows a DBA to specify when and how a view should be refreshed..
- A view can be refreshed in two modes:**immediate or deferred.**
- **immediate mode**,
  - a view is refreshed immediately as part of the transaction that updates the base data used by the view.
  - If the view and the base data are managed by different DBMSs, possibly at different sites, this requires the use of a distributed transaction, for instance, using the two-phase commit (2PC) protocol

Maintenance of Materialized Views

- The main advantages of immediate refreshment are :
  1. the view is always consistent with the base data
  2. read-only queries can be fast.
- Disadvantages:
  1. Increased transaction time to update both the base data and the views within the same transactions.
  2. Using distributed transactions may be difficult.

Maintenance of Materialized Views

- **Deferred mode**
- In the **deferred** mode the view is refreshed in separate (refresh) transactions, without performance penalty on the transactions that update the base data.
- The refresh transactions can be triggered at different times:
    1. **Lazily,** just before a query is evaluated on the view;
    2. **Periodically,** at predefined times, e.g., Every day;
    3. **Forced,** after a predefined number of updates to the base data.

MAINTENANCE OF MATERIALIZED VIEWS

- **Lazy refreshment** enables queries to see the latest consistent state of the base data but at the expense of increased query time to include the refreshment of the view.
- **Periodic and forced refreshment** allow queries to see views whose state is not consistent with the latest state of the base data.
- The view managed with these strategies are also called **snapshots**

How to refresh a view(Methods of Refresh)

- Recompute it from scratch using the base data. Advantageous if a large subset of the base data has been changed.
- Incremental view maintenance relies on the concept of differential relation. Computing only the changes to the view.
  - Where only a small subset of view needs to be changed then a better strategy is to compute the view incrementally

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
- Approaches to Data protection
  - **Data encryption:** Encrypted data can be decrypted only by authorized users who “know” the code.
    - **Two main schemes**
      - Data Encryption Standard
      - public-key encryption

Data Security

- Access control must guarantee that only authorized users perform operations they are allowed to perform on the database.
- Access control in database systems differs in several aspects from that in traditional file systems
- Authorizations must be refined so that different users have different rights on the same database objects → implies ability to specify subsets of objects more precisely than by name and to distinguish between groups of users.
- decentralized control of authorizations is of particular importance in a distributed context.
- In relational systems, authorizations can be uniformly controlled by database administrators using high-level constructs

Approaches to database access control

- **Discretionary (or authorization control) :** defines access rights based on the users, the type of access (e.g., SELECT, UPDATE) and the objects to be accessed.
- **Mandatory (or multilevel):** further increases security by restricting access to classified data to cleared users (quite resent as a result of threats from the internet)
- While both can be implemented on centralized DB, Distributed DBs bring in additional complexity which stems from the fact that objects and users can be distributed.

Discretionary (or authorization control)

- Three main actors are involved in discretionary access control:
  1. **Subject** (e.g., users, groups of users) who trigger the execution of application programs;
  2. **The operations**, which are embedded in application programs;
  3. **The database objects**, on which the operations are performed
- Authorization control consists of checking whether a given triple (subject, operation, object) can be allowed to proceed
- To control authorizations properly, the DBMS requires the definition of subjects,objects, and access rights
- The introduction of a subject in the system is done by a pair (user name, password).
- A user name uniquely identifies the users of that name in the system, while the password, authenticates the users.
- User name and password prevents people who do not know the password from entering the system with only the user name.

Discretionary (or authorization control)

- The objects to protect are subsets of the database.
- A right expresses a relationship between a subject and an object for a particular set of operations.
- In an SQL-based relational DBMS, an operation is a high-level statement such as SELECT, INSERT, UPDATE, or DELETE,
- Rights are defined (granted or revoked) using such statements:
- As
  - ❑GRANT (operation type(s) ON (object) TO (subject(s))
  - ❑REVOKE (operation type(s)) ON (object) FROM (subject(s))
- The keyword public for subject can be used to mean all users.

Discretionary (or authorization control)

- Authorization control can be characterized based on who (the grantors) can grant the rights.
  - Centralized: where a single user or user class, (DBAs), has all privileges on the database objects and is the only one allowed to use the GRANT and REVOKE statements
  - Decentralized: allows the owner to grant access to his/her objects
  - The GRANT, may also be transferred by the grantor to the grantee all the rights of the grantor performing the statement to the specified subjects
  - Privileges of the subjects over objects are recorded in the catalog (directory) as authorization rules in the form of an authorization matrix, in which a row defines a subject, a column an object, and a matrix entry, the authorized operations.

AUTHORIZATION MATRIX

- The authorization matrix can be stored in three ways:
  1. **by row** each subject is associated with the list of objects that may be accessed together with the related access rights and all the rights of the logged-on user are together (in the user profile)
  2. **by column : each object is associated with the list of subjects who may access it with the corresponding access rights**
  3. **by element: This is a combined approach** in which the matrix is stored by element, that is, by relation (subject, object,right).

Example of Authorization Matrix

	EMP	ENAME	ASG
Casey	UPDATE	UPDATE	UPDATE
Jones	SELECT	SELECT	SELECT WHERE RESP ≠ "Manager"
Smith	NONE	SELECT	NONE

Multilevel Access Control

- Discretionary access control has some limitations.
- One problem is that a malicious user can access unauthorized data through an authorized user
- Multilevel access control answers this problem and further improves security by defining different security levels for both subjects and data objects
- This approach is based on Bell and LapaduLa model designed for operating system security
- In this model,subjects are processes acting on a user's behalf; a process has a security level also called clearance derived from that of the user.
- 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

Multilevel Access Control

- Access in read and write modes by subjects is restricted by two rules:
  1. A subject S is allowed to read an object of security level l only if level(S) >= l.
  2. A subject S is allowed to write an object of security level l only if class(S) <= l.

1. **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.
2. **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.

MULTILEVEL ACCESS CONTROL

Bell and Lapaduda model:

- A subject with top-secret clearance can only write top-secret data but cannot write secret data (which could then contain top-secret data).
- Data objects can be relations, tuples or attributes.
- A relation can be classified at different levels:
    1. Relation (i.e., all tuples in the relation have the same security level)
    2. Tuple (i.e., every tuple has a security level),
    3. Attribute (i.e., every distinct attribute value has a security level).
- A classified relation called multilevel relation, to reflect that it will appear differently (with different data) to subjects with different clearances.
- Eg
    - a multilevel relation classified at the tuple level can be represented by adding a security level attribute to each tuple.
    - a multilevel relation classified at attribute level can be represented by adding a corresponding security level to each attribute.

MULTILEVEL ACCESS CONTROL

PROJ*

PNO	SL1	PNAME	SL2	BUDGET	SL3	LOC	SL4
P1	C	Instrumentation	C	150000	C	Montreal	C
P2	C	Database Develop.	C	135000	S	New York	S
P3	S	CAD/CAM	S	250000	S	New York	S

Multilevel relation PROJ* based on relation PROJ which is classified at the attribute level.

Note that the additional security level attributes may increase significantly the size of the relation.

The entire relation has a security level which is the lowest security level of any data it contains.(relation PROJ* has security level C)

A relation can then be accessed by any subject having a security level which is the same or higher

A subject can only access data for which it has clearance.

MULTILEVEL ACCESS CONTROL

Attributes for which a subject has no clearance will appear to the subject as null values with an associated security level which is the same as the subject.

PROJ*

PNO	SL1	PNAME	SL2	BUDGET	SL3	LOC	SL4
P1	C	Instrumentation	C	150000	C	Montreal	C
P2	C	Database Develop.	C	135000	S	New York	S
P3	S	CAD/CAM	S	250000	S	New York	S

PROJ*C

PNO	SL1	PNAME	SL2	BUDGET	SL3	LOC	SL4
P1	C	Instrumentation	C	150000	C	Montreal	C
P2	C	Database Develop.	C	Null	C	Null	C

An instance of relation PROJ* as accessed by a subject at a confidential security level.

MULTILEVEL ACCESS CONTROL

- Multilevel access control has strong impact on the data model because users do not see the same data and have to deal with unexpected side-effects.
- One major side-effect is called polyinstantiation which allows the same object to have different attribute values depending on the users' security level.

MULTILEVEL ACCESS CONTROL

PROJ**

PNO	SL1	PNAME	SL2	BUDGET	SL3	LOC	SL4
P1	C	Instrumentation	C	150000	C	Montreal	C
P2	C	Database Develop.	C	135000	S	New York	S
P3	S	CAD/CAM	S	250000	S	New York	S
P3	C	Web Develop.	C	200000	C	Paris	C

Tuple of primary key P3 has two instantiations, each one with a different security level.

This may result from a subject S with security level C inserting a tuple with key =“P3” in relation PROJ*

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:
  1. remote user authentication,
  2. management of discretionary access rules,
  3. handling of views and of user groups,
  4. 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
- To prevent remote access by unauthorized users or applications users must also be identified and authenticated at the accessed site.
- Instead of using passwords that could be obtained from sniffing messages, encrypted certificates could be used.

Possible solutions 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. Local programs, initiated at a remote site, must also indicate the user name and password.
3. All sites of the distributed DBMS identify and authenticate themselves similar to the way users do. Intersite communication is protected by the use of the site password. Once the initiating site has been authenticated, there is no need for authenticating their remote users.

COMMENTS ON THE POSSIBLE SOLUTIONS

- **Solution one(**Authentication information is maintained at a central site) simplifies password administration and enables single authentication but the central authentication site can be a single point of failure and a bottleneck
- **Solution two (**information for authenticating users replicated at all sites in the catalog) is more costly in terms of directory management given that the introduction of a new user is a distributed operation however, users can access the distributed database from any site.
- **Solution 3 (**All sites of the distributed DBMS identify and authenticate themselves is necessary if user information is not replicated.
  - ❑ If user names and passwords are not replicated, they should be stored at the sites where the users access the system
  - ❑The solution is based on the realistic assumption that users are more static, or at least they always access the distributed database from the same site.

COMMENTS ON THE POSSIBLE SOLUTIONS

- Distributed authorization rules are expressed in the same way as centralized ones.
- view definitions must be stored in the catalog
- Views can be either fully replicated at each site or stored at the sites of the referenced objects
- Main advantage of the fully replicated approach is that authorization can be processed by query modification at compile time although directory management is more costly because of data duplication
- Views are composite objects, composed of other underlying objects. Therefore,granting access to a view translates into granting access to underlying objects

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.

Types of integrity constraints

- Two main types exist:
  1. **Structural constraints:** express basic semantic properties inherent to a model. Eg unique key constraints in the relational model, or one-to-many associations between objects in the object-oriented model.
  2. **Behavioral constraints.** regulate the application behavior.
     - ❑They 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

- A semantic integrity manager has two main components:
  1. a language for expressing and manipulating integrity assertions,
     - standard SQL language through triggers, CASCADING clause, Precondition constraints, NEW and OLD, are implicitly defined.
  2. an enforcement mechanism that performs specific actions to enforce database integrity upon update transactions

Distributed Semantic Integrity Control

- Definition of Distributed Integrity Constraints
- Since 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

1. **Individual constraints:** single-relation single-variable constraints. They refer only to tuples to be updated independently of the rest of the database. For instance, the domain constraint
2. **Set-oriented constraints:** include single-relation multivariable constraints such as functional dependency and multirelation multivariable constraints such as foreign key constraints
3. **Constraints involving aggregates:** require special processing because of the cost of evaluating the aggregates. The assertion is representative of a constraint of this class.

Individual constraints.

- The constraint definition is sent to all other sites that contain fragments of the relation involved in the constraint.
- The constraint must be compatible with the relation data at each site.
- Compatibility can be checked at two levels:
  1. **predicate:** verified by comparing the constraint predicate with the fragment predicate

  A constraint C is not compatible with a fragment predicate p if “C is true” implies that “p is false,” and is compatible with p otherwise.

  If noncompatibility is found at one of the sites, the constraint definition is globally rejected because tuples of that fragment do not satisfy the integrity constraints.

INDIVIDUAL CONSTRAINTS

- Say EMP, is horizontally fragmented across three sites using the predicates
- p1 : 0 < ENO < “E3”
- p2 : “E3” <= ENO < =“E6”
- p3 : ENO >“E6”
- If the domain constraint C is ENO < “E4”.
    1. Constraint C is compatible with p1 →if C is true, p1 is true
    2. Constraint C is compatible with p2 →if C is true, p2 is not necessarily false
    3. Constraint C is not compatible with p3 → if C is true, then p3 is false.

Constraint C should be globally rejected because the tuples at site 3 cannot satisfy C,

Individual constraints.

2 **data:** If predicate compatibility has been found, the constraint is tested against the instance of the fragment. (If it is not satisfied by that instance, the constraint is also globally rejected. If compatibility is found, the constraint is stored at each site.)

Set-oriented constraints.

- Set-oriented constraint are multivariable; that is, they involve join predicates.
- The constraint definition can be sent to all the sites that store a fragment referenced by these variables.
- Compatibility checking involves fragments of the relation used in the join predicate.
- Predicate compatibility is useless because it is impossible to infer that a fragment predicate p is false if the constraint C (based on a join predicate) is true. Therefore C must be checked for compatibility against the data.

Enforcement of Distributed Integrity Assertions

- Enforcing distributed integrity assertions is more complex than needed in centralized DBMSs
- The main problem is to decide where (at which site) to enforce the integrity constraints.
- The choice depends on the class of the constraint, the type of update, and the nature of the site where the update is issued (called the query master site).
- This site may, or may not, store the updated relation or some of the relations involved in the integrity constraints
- The critical parameter we consider is the cost of transferring data, including messages, from one site to another.

ENFORCEMENT OF DISTRIBUTED INTEGRITY ASSERTIONS(Individual constraints)

- Two cases are considered.
    1. If the update transaction is an insert statement, all the tuples to be inserted are explicitly provided by the user. In this case, all individual constraints can be enforced at the site where the update is submitted.
    2. If the update is a qualified update (delete or modify statements), it is sent to the sites storing the relation that will be updated.

Set-oriented constraints.

- The site where the update is submitted must receive for each site a message indicating that this constraint is satisfied and that it is a condition for all sites.
- If the constraint is not true for one site, this site sends an error message indicating that the constraint has been violated.
- The update is then invalid, and it is the responsibility of the integrity manager to decide if the entire transaction must be rejected using the global transaction manager.
- As with single-relation constraints, the update is computed at the site where it was submitted.
- The enforcement is done at the query master site, using the ENFORCE given in Algorithm

Constraints involving aggregates.

- These constraints are among the most costly to test because they require the calculation of the aggregate functions.(MIN, MAX, SUM, and COUNT)
- To enforce these constraints efficiently, it is possible to produce pretest that isolate redundant data which can be stored at each site storing the associated relation
- This data is what we called materialized views

QUESTIONS

a. Show in a Diagram
   1. ANSI/SPARC three-level architecture
   2. Components of a DBMS

b. Distinguish between the terms ‘external schema’, ‘conceptual schema’ and ‘internal schema’.

c. • Explain the role of an application server in a 3-tier client-server network.

d. • Distinguish between the terms ‘fragmentation’ and ‘replication’ in a distributed database environment.

e. What problems are associated with data replication and fragmentation?

f. Distinguish between Logical data independence and physical data independence

QUESTIONS

- **The external schema**
- The external schemas describe the database as it is seen by the user, and the user applications. The external schema maps onto the conceptual schema, which
- is described below.
- There may be many external schemas, each reflecting a simplified model of the world, as seen by particular applications. External schemas may be modified, or new ones created, without the need to make alterations to the physical storage of data. The interface between the external schema and the conceptual schema can be amended to accommodate any such changes.
- The external schema allows the application programs to see as much of the data as they require, while excluding other items that are not relevant to that application. In this way, the external schema provides a view of the data that corresponds to the nature of each task.
- The external schema is more than a subset of the conceptual schema. While items in the external schema must be derivable from the conceptual schema, this could be a complicated process, involving computation and other activities.

QUESTIONS

- **The conceptual schema**

The conceptual schema describes the universe of interest to the users of the database system. For a company, for example, it would provide a description of all of the data required to be stored in a database system. From this organisation-wide description of the data, external schemas can be derived to provide the data for specific users or to support particular tasks.

At the level of the conceptual schema we are concerned with the data itself, rather than storage or the way data is physically accessed on disk. The definition of storage and access details is the preserve of the internal schema.

QUESTIONS

- **The internal schema**

A database will have only one internal schema, which contains definitions of the way in which data is physically stored. The interface between the internal schema and the conceptual schema identifies how an element in the conceptual schema is stored, and how it may be accessed.

If the internal schema is changed, this will need to be addressed in the interface between the internal and the conceptual schemas, but the conceptual and external schemas will not need to change. This means that changes in physical storage devices such as disks, and changes in the way files are organised on storage devices, are transparent to users and application programs.

In distinguishing between **'logical' and 'physical**' views of a system, it should be noted that the difference could depend on the nature of the user. While 'logical'

describes the user angle, and 'physical' relates to the computer view, database designers may regard relations (for staff records) as logical and the database itself as physical. This may contrast with the perspective of a systems programmer,who may consider data files as logical in concept, but their implementation on magnetic disks in cylinders, tracks and sectors as physical.

QUESTIONS

- **Physical data independence**

In a database environment, if there is a requirement to change the structure of a particular file of data held on disk, this will be recorded in the internal schema.

The interface between the internal schema and the conceptual schema will be amended to reflect this, but there will be no need to change the external schema.

This means that any such change of physical data storage is not transparent to users and application programs. This approach removes the problem of physical data dependence.

QUESTIONS

- **Logical data independence**
- Any changes to the conceptual schema can be isolated from the external schema and the internal schema; such changes will be reflected in the interface between the conceptual schema and the other levels. This achieves logical data independence.

What this means, effectively, is that changes can be made at the conceptual level, where the overall model of an organisation's data is specified, and these changes can be made independently of both the physical storage level, and the external level seen by individual users. The changes are handled by the interfaces between the conceptual, middle layer, and the physical and external layers.