



## More SQL & DB Security Overview

1

## Back to SQL: Commands for advanced utilities



- ❖ Index structures – needed for performance tuning
  - SQL standards do not require this utility but all vendors provide options
- ❖ Transaction definition
- ❖ Lock granularity
- ❖ Security

2

## Index Creation



- ❖ CREATE [UNIQUE/BITMAP] INDEX <name> ON <table-name> (attribute) [WITH STRUCTURE=<type> ]
- ❖ CREATE UNIQUE INDEX cust-name ON CUSTOMER (cname)
- ❖ Unique index usually creates B+tree index
- ❖ Bitmap creates bitmap index
  - Create this index on columns with low cardinality that are likely to have few distinct values

3

## Transactions



- ❖ COMMIT [work];
  - End of sql query treated as end of transaction
- ❖ MS SQL Server allows for BEGIN
  - BEGIN TRAN
  - select \* from customer UPDATE customer SET salary=1.5\*salary
  - COMMIT TRAN
- ❖ ROLLBACK [work] TO SAVEPOINT <name>

4

## *Locks*



- ❖ Depends of the DBMS
  - Oracle, MS SQL server provide different options
- ❖ LOCK TABLE <table name>  
IN <mode> [NOWAIT]
  - Mode: EXCLUSIVE, SHARE,
    - ROW SHARE, ROW EXCLUSIVE
  - NOWAIT: instructs Xact to move on if table is already locked

5

## *SQL Authorization*



- ❖ Next..

6

## *Database Security*



7

## *Databases provide*



- ❖ Shared access
- ❖ Controlled access
- ❖ Data consistency
- ❖ Data integrity
- ❖ Minimal redundancy

8

## DB Security Requirements



- ❖ **Secrecy:** Users should not be able to see things they are not supposed to.
  - E.g., A student can't see other students' grades.
- ❖ **Integrity:** Users should not be able to modify things they are not supposed to.
  - E.g., Only instructors can assign grades.
- ❖ **Availability:** authorized users should be able to see and modify things they are allowed to

9

## DB Security



- ❖ **Non-repudiation:**
  - DB cannot deny getting requests for changes
- ❖ **Auditability**
  - Helps determine if inappropriate disclosure has occurred
  - Helps track divulged information to prevent inference
  - Difficult to record access of fields; often a field is reported to have been accessed when it has not - e.g. SELECT all entries with ZIP 20007

10

## DB Security and Access Controls



- ❖ A **security policy** specifies who is authorized to do what.
- ❖ A **security mechanism** allows us to enforce a chosen security policy.
  - Two main mechanisms at the DBMS level:
    - Discretionary access control
    - Mandatory access control
- ❖ Note: when DB is a backend of an application, also need to be able to **authenticate** users
  - Logging in to Amazon through a network

11

## Means of Achieving Security Requirements



- ❖ Access Control
  - Inference is a problem (fields are related; knowledge of race or gender can be a good predictor of salary, for example)
  - Size and granularity different from access control in OS
- ❖ User Authentication
  - DBMS does its own authentication because no trusted path between DBMS and OS when DBMS is an application program on top of OS

12

## Discretionary Access Control



- ❖ Based on the concept of access rights or **privileges** for objects and mechanisms for giving users privileges (and revoking privileges).
- ❖ In Databases, objects refer to Tables and Views
  - Creator of a table or a view automatically gets all privileges on it.
  - DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed.

13

## SQL GRANT Command



**GRANT** privileges **ON** object **TO** users [**WITH GRANT OPTION**]

- ❖ The following **privileges** can be specified:
  - ❖ **SELECT**: Can read all columns (including those added later via ALTER TABLE command).
  - ❖ **INSERT**(col-name): Can insert tuples with non-null or non-default values in this column.
    - ❖ INSERT means same right with respect to all columns.
  - ❖ **DELETE**: Can delete tuples.
  - ❖ **REFERENCES** (col-name): Can define foreign keys (in other tables) that refer to this column.
- ❖ If a user has a privilege with the **GRANT OPTION**, can pass privilege on to other users (with or without passing on the **GRANT OPTION**).
- ❖ Only owner can execute CREATE, ALTER, and DROP.

14

## GRANT and REVOKE of Privileges



- ❖ GRANT INSERT, SELECT ON Sailors TO Ryan
  - Ryan can query Sailors or insert tuples into it
- ❖ GRANT DELETE ON Sailors TO Peter WITH GRANT OPTION
  - Peter can delete tuples, and also authorize others to do so
- ❖ GRANT UPDATE (*rating*) ON Sailors TO Sarah
  - Sarah can update (only) the *rating* field of Sailors tuples
- ❖ GRANT SELECT ON AllCustomers TO Guppy, Yuppy
  - This does NOT allow the 'uppies to query Deposit or Loan directly!
    - AllCustomers is a View over Deposit and Loan tables
- ❖ **REVOKE**: When a privilege is revoked from X, it is also revoked from all users who got it *solely* from X

15

## Views: Example



- ❖ Create view of all branch names and IDs of customers with loan or deposit

```
CREATE VIEW AllCustomers AS
(SELECT branchname, CustID
FROM Deposit)
UNION
(SELECT branchname, CustID
FROM Loan);
```

16

## GRANT/REVOKE on Views



- ❖ If the creator of a view loses the SELECT privilege on an underlying table, the view is dropped!
- ❖ If the creator of a view loses a privilege held with the grant option on an underlying table, (s)he loses the privilege on the view as well; so do users who were granted that privilege on the view!

17

## Views and Security



- ❖ Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - Given AllCustomers, but not Deposit or Loan, we can find customers at branch, but not their account numbers.
- ❖ Creator of view has a privilege on the view if (s)he has the privilege on all underlying tables.
- ❖ Together with GRANT/REVOKE commands, views are a very powerful access control tool.
- ❖ How about GUI based views
  - Take the user to a different GUI based on their role?..banner?
    - This is what you are doing in your project!

18

## Role-Based Authorization



- ❖ In SQL-92, privileges are actually assigned to **authorization ids**, which can denote a single user or a group of users.
  - CREATE ROLE and DROP ROLE
  - GRANT role and REVOKE role
- ❖ In SQL:1999 (and in many current systems), privileges are assigned to **roles**.
  - Roles can then be granted to users and to other roles.
  - Reflects how real organizations work.
  - Illustrates how standards often catch up with “de facto” standards embodied in popular systems.

19

## Security to the Level of a Field!



- ❖ Can create a view that only returns one field of one tuple. (How?)
  - Then grant access to that view accordingly.
- ❖ Allows for *arbitrary* granularity of control, *but*:
  - Clumsy to specify
  - Performance is unacceptable (Too many view creations and look-ups.)

20

## Problem ?



- ❖ Discretionary control has some flaws, e.g., the **Trojan horse** problem:
  - Kyle creates Horsie and gives INSERT privileges to Ryan (who doesn't know about this).
  - Kyle modifies the code of an application program, to update grades, used by Ryan, to additionally write some secret data (Kyle's grades) to table Horsie.
  - Now, Kyle can see the secret info.
- ❖ The modification of the code is beyond the DBMSs control, but it can try and prevent the use of the database as a **channel** for secret information.

21

## Mandatory Access Control



- ❖ Based on system-wide policies that cannot be changed by individual users.
  - Each **DB object** is assigned a **security class**.
  - Each **subject** (user or user program) is assigned a **clearance** for a security class.
  - Rules based on security classes and clearances govern who can read/write which objects.
- ❖ Most commercial systems do not support mandatory access control. Versions of some DBMSs do support it; used for specialized (e.g., military) applications.

22

## Mandatory Access Control: Intuition



- ❖ Idea is to ensure that information can never flow from a higher to a lower security level.
- ❖ E.g., If Narahari has security class C, Peter has class S, and the secret table has class S:
  - Narahari's table, Horsie, has Narahari's clearance, C.
  - Peter's application has his clearance, S.
  - So, the program cannot write into table Horsie.
- ❖ The mandatory access control rules are applied in addition to any discretionary controls that are in effect.

23

## Multilevel Security (MLS): Bell-LaPadula Model



- ❖ Objects (e.g., tables, views, tuples)
- ❖ Subjects (e.g., users, user programs)
- ❖ Security classes:
  - Top secret (TS), secret (S), confidential (C), unclassified (U):  $TS > S > C > U$
- ❖ Each object and subject is assigned a class.
  - Subject S can read object O only if  $class(S) \geq class(O)$  (**Simple Security Property**)
  - Subject S can write object O only if  $class(S) \leq class(O)$  (**\*-Property**)

24

## Internet-Oriented Security



- ❖ **Key Issues: User authentication and trust.**
  - When DB must be accessed from a secure location, password-based schemes are usually adequate.
- ❖ For access over an external network, trust is hard to achieve.
  - If someone with Peter's credit card wants to buy from you, how can you be sure it is not someone who stole his card?
  - How can Peter be sure that the screen for entering his credit card information is indeed yours, and not some rogue site spoofing you (to steal such information)? How can he be sure that sensitive information is not "sniffed" while it is being sent over the network to you?
- ❖ *Encryption* is a technique used to address these issues.

25

## Database Security and Privacy



- ❖ How to ensure privacy in the data stored in the database ?
- ❖ How to enforce the legal policies of access into the system ?
  - Example: medical records databases
- ❖ What about privacy preserving mining
  - After mining the data, we can learn something about the person that we are not supposed to!

26

## More security....



- ❖ Security is a major concern in all information systems today
- ❖ Has many aspects to consider
  - Is a challenging software engineering problem
- ❖ Want to learn more...take the security courses!

27

## Exam 2: April 29<sup>th</sup>



- ❖ All materials after Exam 1
  - upto(including) Transactions and basic Security definitions
  - Focus on materials after Exam 1
    - But need to remember relational algebra and SQL
- ❖ Go over Homeworks 5,6 and Transaction processing examples from book and notes
  - Yes, you will be expected to remember the 'basic' time complexities of the various operations we covered
- ❖ Similar format to Exam 1
  - Multiple choice and short-answers

28