Data Control Language in SQL

by

Joe Celko

copyright 2006
DCL in SQL

- DCL = Data Control Language
- This is the third and least respected sub-language in SQL
- It is not really a security system, but more of an access control language
- Encryption is not part of the Standards
DCL in SQL

- There are two kinds of people in the Schema
- Users: they work with the data, but cannot change the structure of the schema. They write DML (Data Manipulation Language).
- Admin: they can change the structure of the schema and control access to the schema objects. They write DDL (Data Declaration Language).
- In actual products, the lines might blur a bit
  - SQL Server allows users to create temporary tables on the fly in their sessions.
- In shops, the lines might blur a bit
  - One employee wears several hats
Access is a four-part relationship
Grantor = a user or admin who controls the Privileges on a Schema Object
Privileges = the actions that can be done on a Schema Object
Schema Object = TABLE, VIEW, DOMAIN, COLLATION, stored procedure, trigger, etc.
Grantee = a user who is given Privileges on a Schema Object
- GRANT <privilege> [ { , <privilege>}... ]
  ON <object name>
  TO <grantee> [ { , <grantee>}... ]
  [ WITH GRANT OPTION ] [ WITH ADMIN ]
- <object name> ::= [ TABLE ] <table name> | ...
  - The other objects are DOMAIN, COLLATION, CHARACTER SET, and TRANSLATION
  - Not widely implemented and not that important for this discussion
  - Big changes in SQL-99 and various product differ slightly
Privileges = SELECT, INSERT, DELETE, UPDATE, or REFERENCES and USAGE
- INSERT, UPDATE, and REFERENCES can be qualified with a list of columns in the table. DELETE is done to rows.
- USAGE applies only to DOMAIN, COLLATION, CHARACTER SET, and TRANSLATION - skip them for now

REFERENCES means that you can create a FOREIGN KEY reference to the keys in this table.
- Security issue: if I have access to a table that references table X, I can experiment around with inserts or updates to discover the contents of table X.

GRANT ALL PRIVILEGES = shorthand for all those the grantor has!
GRANT

- Grantee = a user log-in id or PUBLIC

- PUBLIC = everyone on the system, present and future

- Some products and SQL-99 have ROLEs, which are groups of related grantees
  - Usually a department or functional group within the enterprise

- SQL does not have a DENY concept, so a grantee can collect privileges from multiple sources.

- The model is a downward flow from a schema owner
  - If your grantor’s privileges are cut off, so are your derived privileges.
• Grantor = implied by the current user log-in id

• The Grantor can only grant what privileges he has

• The model is a downward flow from a mythical schema owner who is the ultimate Admin
  - If your grantor’s privileges are cut off, so are your derived privileges.

• WITH GRANT OPTION allows new grantee to become a grantor

• WITH ADMIN OPTION allows new grantee to alter the schema

• Circular grants used to be a problem
  - Give yourself privileges and nobody else can take them away
REVOKE

- The same model as GRANT, but “in reverse”
- REVOKE <privileges> ON <object name> FROM <grantee> [ { , <grantee> }... ] [RESTRICT | CASCADE]
- Only the grantor can execute this and only on the grantees to whom he gave those privileges
- The grantee can be PUBLIC
- A grantee can have multiple paths from many grantors to a privilege
  - There is no global “DENY” statement in SQL that would prevent a user from ever getting a privilege.
• If this grantee had WITH GRANT OPTION and passed along his privileges, what happens when his privileges are revoked?
• RESTRIC T = Send up an error message about the chain of privileges and do nothing
• CASCADE = Revoke everyone who is “downstream” from the grantee. Revoke all privileges that were derived from the original GRANT action (VIEWs, stored procedures, etc.)
  - This can be dangerous
If this grantee had WITH GRANT OPTION and passed along his privileges, what happens when his privileges are revoked?

- RESTRICT = Send up an error message about the chain of privileges and do nothing
- CASCADE = Revoke everyone who is “downstream” from the grantee. Revoke all privileges that were derived from the original GRANT action (VIEWS, stored procedures, etc.)
  - This can be dangerous
Security vs Access Control - 1

- Security stinks in Standard SQL
  - and most products
  - and most installations

- If you know the schema, make a copy on a new machine, take the existing database and copy the file that contains the target to the new machine.

- You can loop thru values in the columns of tables to which you have REFERENCES privileges until you get a hit

- You and conspirators can set up circular GRANTs
Security stinks in Standard SQL
- and most products
- and most installations

If you know the schema, make a copy on a new machine, take the existing database and copy the file that contains the target to the new machine.

You can loop thru values in the columns of tables to which you have REFERENCES privileges until you get a hit.

You and conspirators can set up circular GRANTs.
Security vs Access Control -2

- Standard SQL does not lie to users
  - If you are Perry White, then you are told Clark Kent is “A mild mannered reporter for a great Metropolitan Newspaper.”
  - If you are Lois Lane, then you are told Clark Kent is “Superman, a strange visitor from another planet.”

- Standard SQL does not encrypt data
  - A vendor option or third party product on most products now

- Standard SQL does not challenge users log-ins, change passwords, etc.
  - A vendor option or third party product on most products now