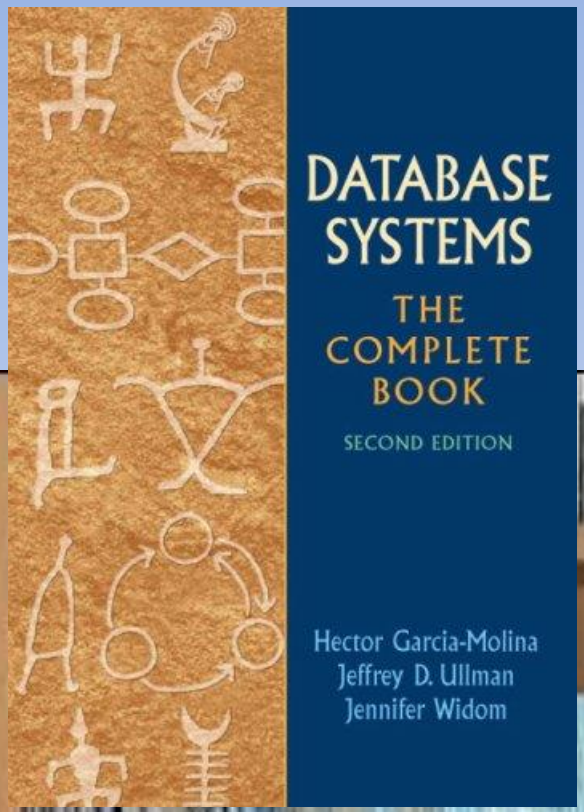
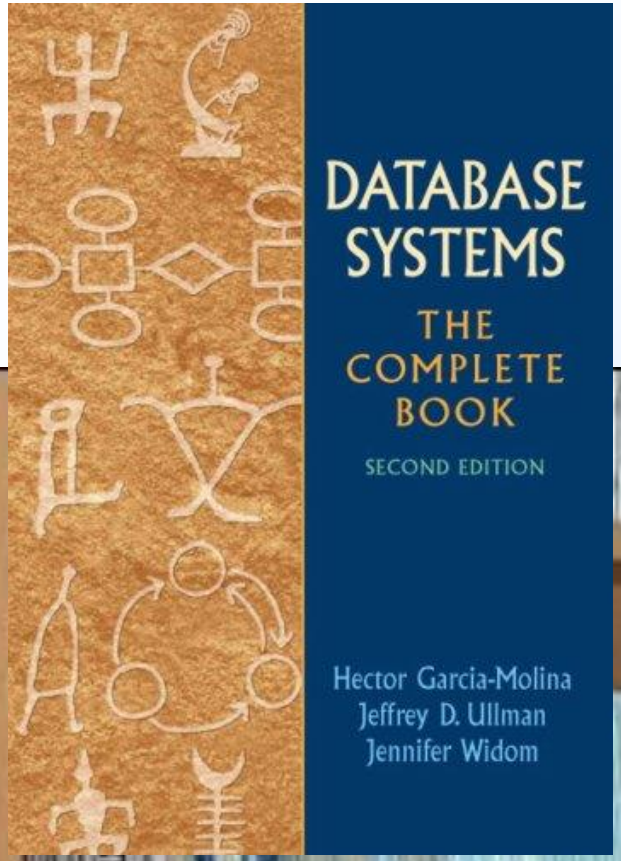
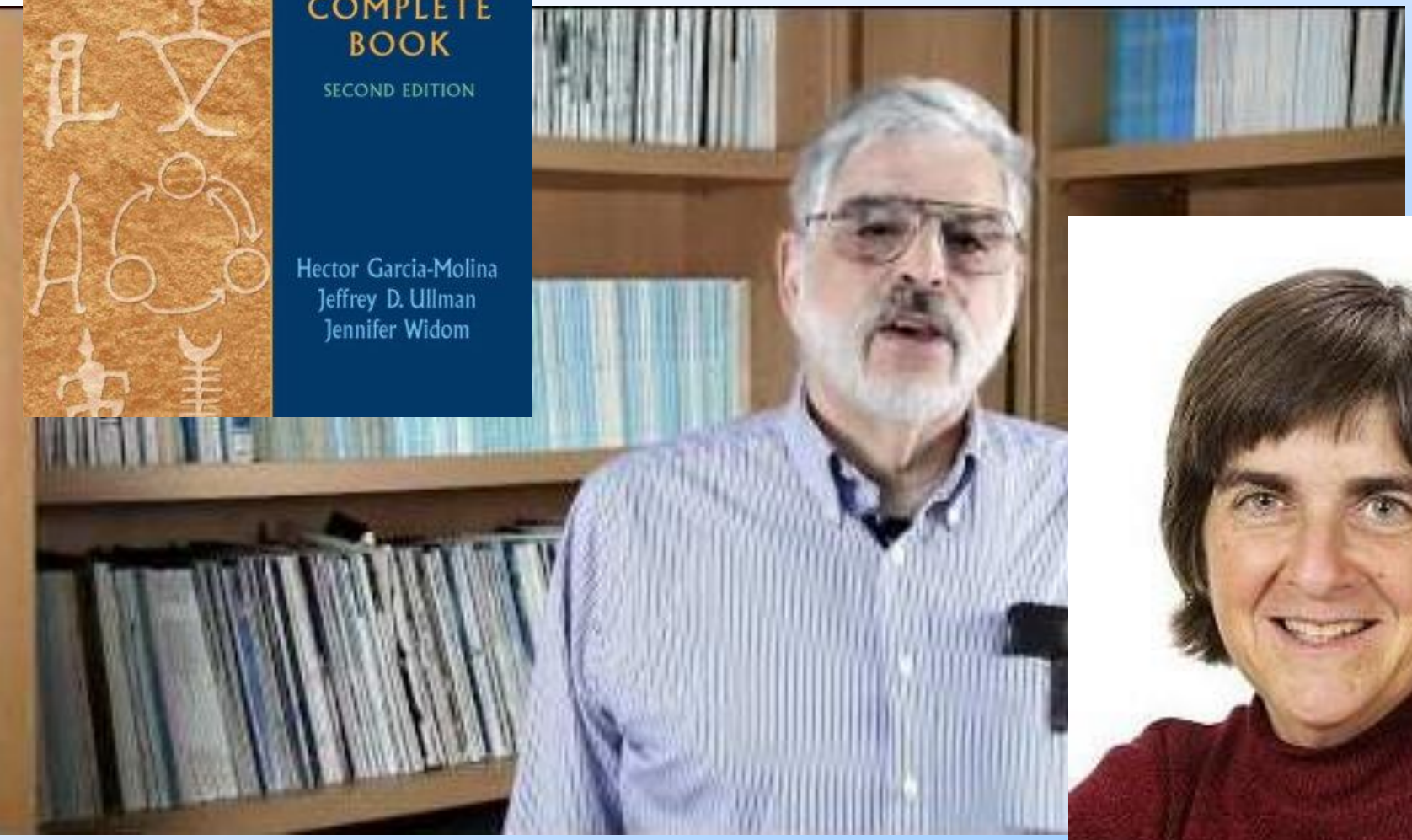
Chapter 10: Advanced Topics in Relational Databases



Chapter 10: Advanced Topics in Relational Databases





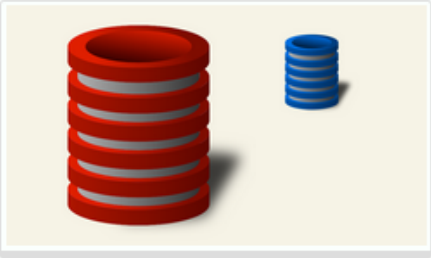
SQL Authorization

Privileges

Grant and Revoke

Grant Diagrams

Stanford Online



Databases

Course Started - Jun 09, 2014 at 15:00 UTC

DB12 Views and Authorization

Your final grade: **90%**.

Download Your Statement (PDF)

View Course

Email Settings *Unenroll*

Authorization

- A file system identifies certain privileges on the objects (files) it manages.
  - Typically read, write, execute.
- A file system identifies certain participants to whom privileges may be granted.
  - Typically the owner, a group, all users.

Privileges – (1)

- SQL identifies a more detailed set of privileges on objects (relations) than the typical file system.
- Nine privileges in all, some of which can be restricted to one column of one relation.

Privileges – (2)

- Some important privileges on a relation:
  1. SELECT = right to query the relation.
  2. INSERT = right to insert tuples.
     - May apply to only one attribute.
  3. DELETE = right to delete tuples.
  4. UPDATE = right to update tuples.
     - May apply to only one attribute.

Example: Privileges

- For the statement below:

```
INSERT INTO Ships(sname, class)
  SELECT class, class FROM Classes C
  WHERE NOT EXISTS
      (SELECT * FROM Ships
       WHERE sname = C.class);
```

Classes that do not appear in Ships. We add them to Ships with a NULL launch.

- We require privileges SELECT on Classes and Ships, and INSERT on Ships or Ships.sname.

Database Objects

- The objects on which privileges exist include stored tables and views.
- Other privileges are the right to create objects of a type, e.g., triggers.
- Views form an important tool for access control.

Example: Views as Access Control

- We might not want to give the SELECT privilege on Emps(name, addr, salary).
- But it is safer to give SELECT on:

```
CREATE VIEW SafeEmps AS
  SELECT name, addr FROM Emps;
```

- Queries on SafeEmps do not require SELECT on Emps, just on SafeEmps.

Authorization ID's

- A user is referred to by *authorization ID*, typically their login name.
- There is an authorization ID PUBLIC.
  - Granting a privilege to PUBLIC makes it available to any authorization ID.

Granting Privileges

- You have all possible privileges on the objects, such as relations, that you create.
- You may grant privileges to other users (authorization ID's), including PUBLIC.
- You may also grant privileges WITH GRANT OPTION, which lets the grantee also grant this privilege.

The GRANT Statement

- To grant privileges, say:

  GRANT <list of privileges>

  ON <relation or other object>

  TO <list of authorization ID's>;

- If you want the recipient(s) to be able to pass the privilege(s) to others add:

  WITH GRANT OPTION

Example: GRANT

- Suppose you are the owner of Sells. You may say:

```
GRANT SELECT, UPDATE(price)
ON Sells
TO sally;
```

- Now Sally has the right to issue any query on Sells and can update the price component only.

Example: Grant Option

- Suppose we also grant:

```
GRANT UPDATE ON Sells TO sally
WITH GRANT OPTION;
```

- Now, Sally not only can update any attribute of Sells, but can grant to others the privilege UPDATE ON Sells.
  - Also, she can grant more specific privileges like `UPDATE(price)ON Sells`.

Revoking Privileges

REVOKE <list of privileges>
ON <relation or other object>
FROM <list of authorization ID's>;

- Your grant of these privileges can no longer be used by these users to justify their use of the privilege.
  - But they may still have the privilege because they obtained it independently from elsewhere.

REVOKE Options

- We must append to the REVOKE statement either:
  1. CASCADE. Now, any grants made by a revokee are also not in force, no matter how far the privilege was passed.
  2. RESTRICT. If the privilege has been passed to others, the REVOKE fails as a warning that something else must be done to "chase the privilege down."

Users w/ MySQL

- First, lets create a new user

CREATE USER *user*
IDENTIFIED BY '*password*'

Users w/ MySQL

- First, lets create a new user

CREATE USER *user*
IDENTIFIED BY '*password*'

CREATE user anonymous@'%'
IDENTIFIED BY 'test';

Users w/ MySQL

- Let’s create a user
- But first, let’s list current users:

```
SELECT User,Host FROM mysql.user;
```

Users w/ MySQL

```
424  #user stuff
425  select User,Host from mysql.user;
```

Result Set Filter: Edit: Export/Import:

User	Host
cs126	%
root	127.0.0.1
root	::1
root	localhost
NULL	NULL

Users w/ MySQL

- Now, lets create a new user

CREATE USER *user*
IDENTIFIED BY '*password*'

CREATE user test@localhost
IDENTIFIED BY 'test';

Users w/ MySQL

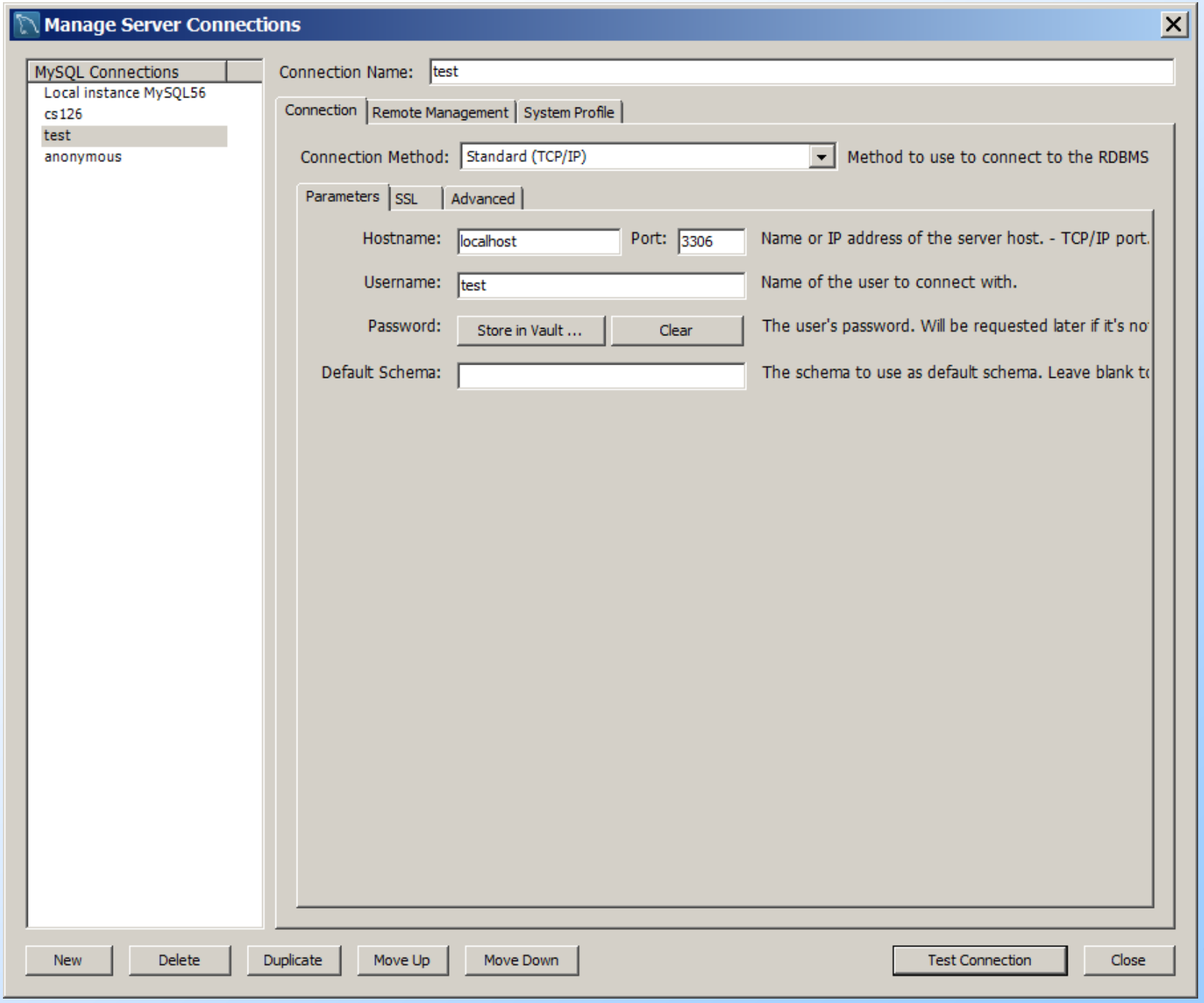
```
#user stuff
select User,Host from mysql.user;
```

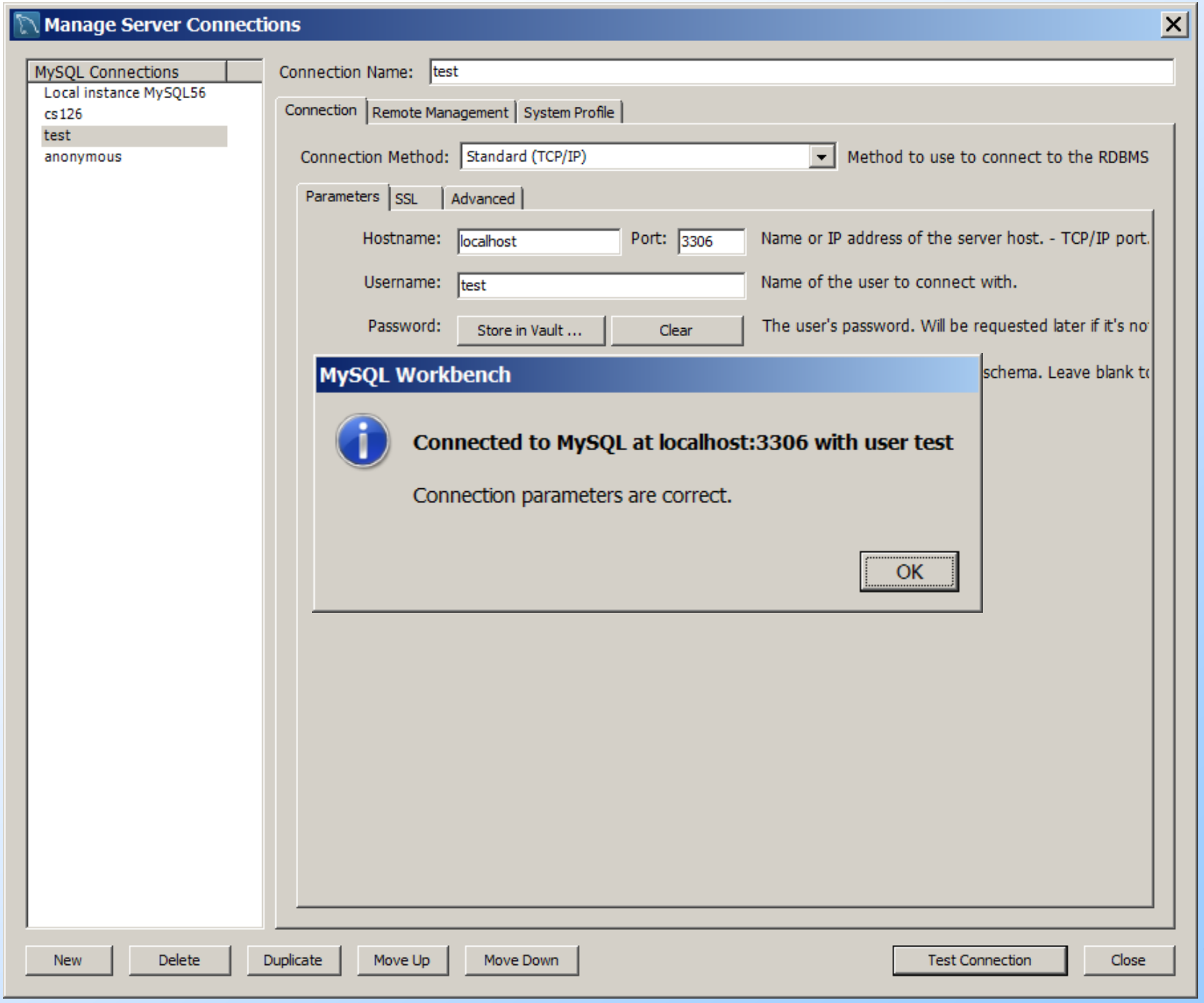
Result Set Filter: Edit: Export/Im

User	Host
cs126	%
root	127.0.0.1
root	::1
root	localhost
test	localhost
NULL	NULL

Connecting as User

- To login to database as our new user:
  - Connection established using new user.


Manage Server Connections
MySQL Connections
Local instance MySQL56
cs126
test
anonymous
Connection Name:
test
Connection
Remote Management
System Profile
Connection Method:
Standard (TCP/IP)
Method to use to connect to the RDBMS
Parameters
SSL
Advanced
Hostname:
localhost
Port:
3306
Name or IP address of the server host. - TCP/IP port.
Username:
test
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's no
Default Schema:
The schema to use as default schema. Leave blank t
New
Delete
Duplicate
Move Up
Move Down
Test Connection
Close


Manage Server Connections
MySQL Connections
Local instance MySQL56
cs126
test
anonymous
Connection Name:
test
Connection
Remote Management
System Profile
Connection Method:
Standard (TCP/IP)
Method to use to connect to the RDBMS
Parameters
SSL
Advanced
Hostname:
localhost
Port:
3306
Name or IP address of the server host. - TCP/IP port.
Username:
test
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's no
schema. Leave blank t
MySQL Workbench
Connected to MySQL at localhost:3306 with user test
Connection parameters are correct.
OK
New
Delete
Duplicate
Move Up
Move Down
Test Connection
Close

User Test

- Lets access computers as ‘test’

User Test

- Lets access computers as ‘test’
- Test needs Privileges

```
#testing
use computers;
```

Output

Action Output

	Time	Action	Message
1	10:10:02	use computers	Error Code: 1044. Access denied for user 'test'@'localhost' to database 'computers'

Example: GRANT

- Suppose you are the owner of Sells. You may say:

```
GRANT SELECT, UPDATE(price)
ON Sells
TO sally;
```

- Now Sally has the right to issue any query on Sells and can update the price component only.

Example: GRANT

- Lets give Test SELECT privilege on Computers.Product to test

```
GRANT select
ON computers.product
TO test@localhost;
```

Now Test has the right to issue any query on Product

```
23 • select * from product;
24 • select model, color from printer;
25
26 • select * from pc where speed >= 3.00;
```

Result Set Filter: Export: Wrap Cell Content:

maker	model	ctype
A	1001	pc
A	1002	pc
A	1003	pc
B	1004	pc
B	1005	pc
B	1006	pc
C	1007	pc
D	1008	pc
D	1009	pc

product 1

Output

Action Output

	Time	Action	Message
1	10:10:02	use computers	Error Code: 1044. Access denied for user 'test'@'localhost' to database 'computers'
2	10:17:49	grant select on computers.product to test...	Error Code: 1142. SELECT, GRANT command denied to user 'test'@'localhost' for table 'product'
3	10:18:54	use computers	0 row(s) affected
4	10:21:13	select * from product LIMIT 0, 1000	31 row(s) returned

Example: GRANT

- Lets give Test SELECT privilege on Computers.Product to test

```
GRANT select
ON computers.product
TO test@localhost;
```

Now Test has the right to issue any query on Product

But not PC table!

Example: GRANT

- Lets give Test SELECT privilege on

```
26 • select * from pc where speed >= 3.00;
27 • select model from pc where speed >= 3.00;
```

Output

Action Output

	Time	Action	Message
1	10:10:02	use computers	Error Code: 1044. Access denied for user 'test'@'localhost' to database 'computers'
2	10:17:49	grant select on computers.product to test...	Error Code: 1142. SELECT, GRANT command denied to user 'test'@'localhost' for table 'product'
3	10:18:54	use computers	0 row(s) affected
4	10:21:13	select * from product LIMIT 0, 1000	31 row(s) returned
5	10:22:47	select * from pc where speed >= 3.00 LIM...	Error Code: 1142. SELECT command denied to user 'test'@'localhost' for table 'pc'

Now Test has the right to issue any query on Product

But not PC table!

Changing Password

- Set Password can be used to change passord for a user:

SET PASSWORD [FOR *user*] = PASSWORD('*cleartext password*')

Changing Password

- Set Password can be used to change passord for a user:

SET PASSWORD [FOR *user*] = PASSWORD(‘*cleartext password*’)

- For Example:

set password = password('test99');

Changing Password

- Set Password can be used to change passord for a user:

SET PASSWORD [FOR *user*] = PASSWORD(‘*cleartext password*’)

- For Example:

SET password

FOR test@localhost = password('test99');

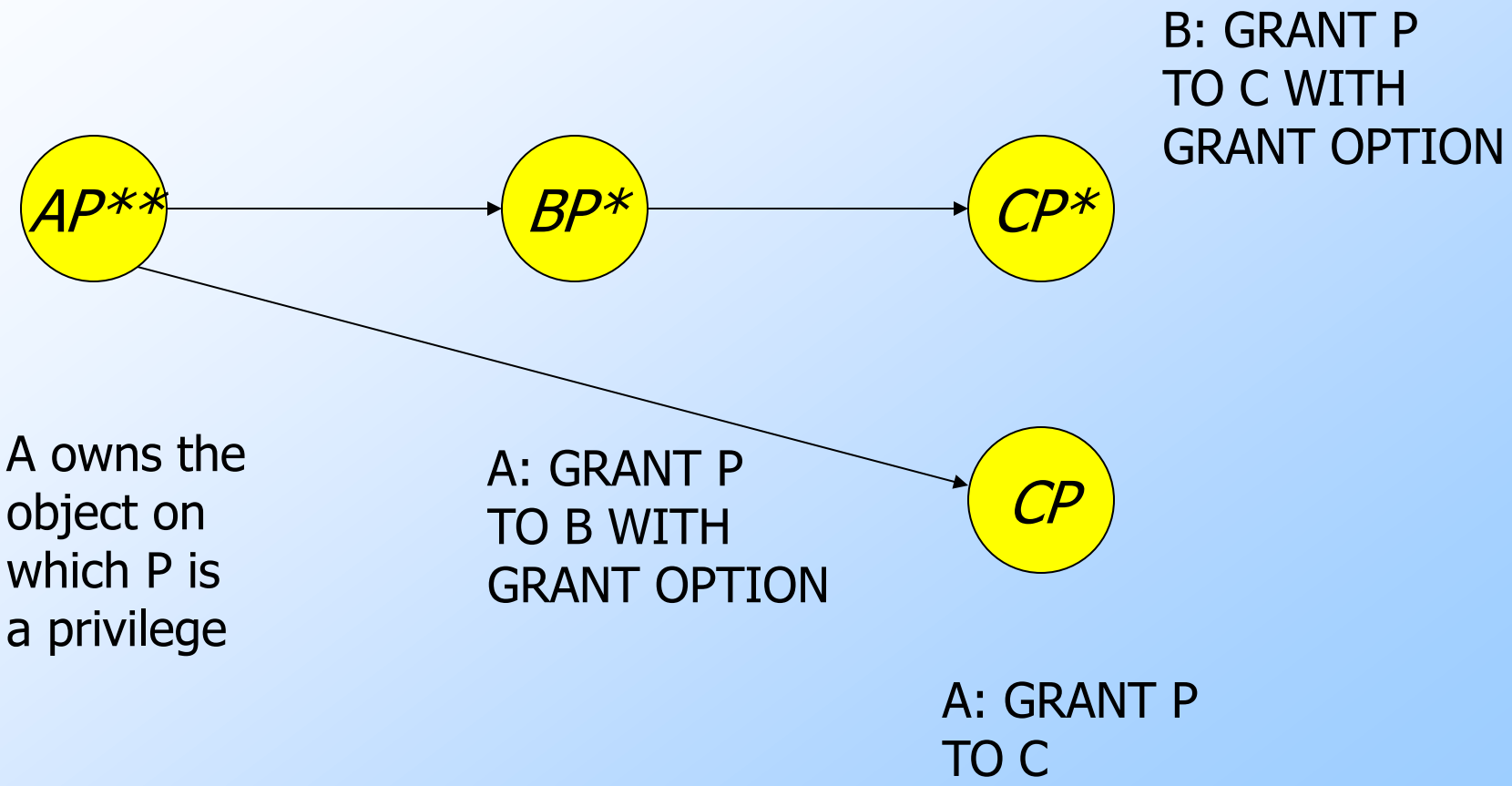
Remember Privileges

- Some important privileges on a relation:
  1. SELECT = right to query the relation.
  2. INSERT = right to insert tuples.
     - May apply to only one attribute.
  3. DELETE = right to delete tuples.
  4. UPDATE = right to update tuples.
     - May apply to only one attribute.

Grant Diagrams

- Nodes = user/privilege/grant option?/is owner?
  - UPDATE ON R, UPDATE(a) on R, and UPDATE(b) ON R live in different nodes.
  - SELECT ON R and SELECT ON R WITH GRANT OPTION live in different nodes.
- Edge *X* -> *Y* means that node *X* was used to grant *Y*.

Example: Grant Diagram



Notation for Nodes

- Use *AP* for the node representing authorization ID *A* having privilege *P*.
  - *P* * = privilege *P* with grant option.
  - *P* ** = the source of the privilege *P*.
    - I.e., *A* is the owner of the object on which *P* is a privilege.
    - Note ** implies grant option.

Manipulating Edges – (1)

- When *A* grants *P* to *B*, We draw an edge from *AP* * or *AP* ** to *BP*.
  - Or to *BP* * if the grant is with grant option.
- If *A* grants a subprivilege *Q* of *P* [say UPDATE(a) on R when *P* is UPDATE ON R] then the edge goes to *BQ* or *BQ* *, instead.

Manipulating Edges – (2)

- Fundamental rule: User *C* has privilege *Q* as long as there is a path from *XP*** to *CQ*, *CQ**, or *CQ***, and *P* is a superprivilege of *Q*.
  - Remember that *P* could be *Q*, and *X* could be *C*.

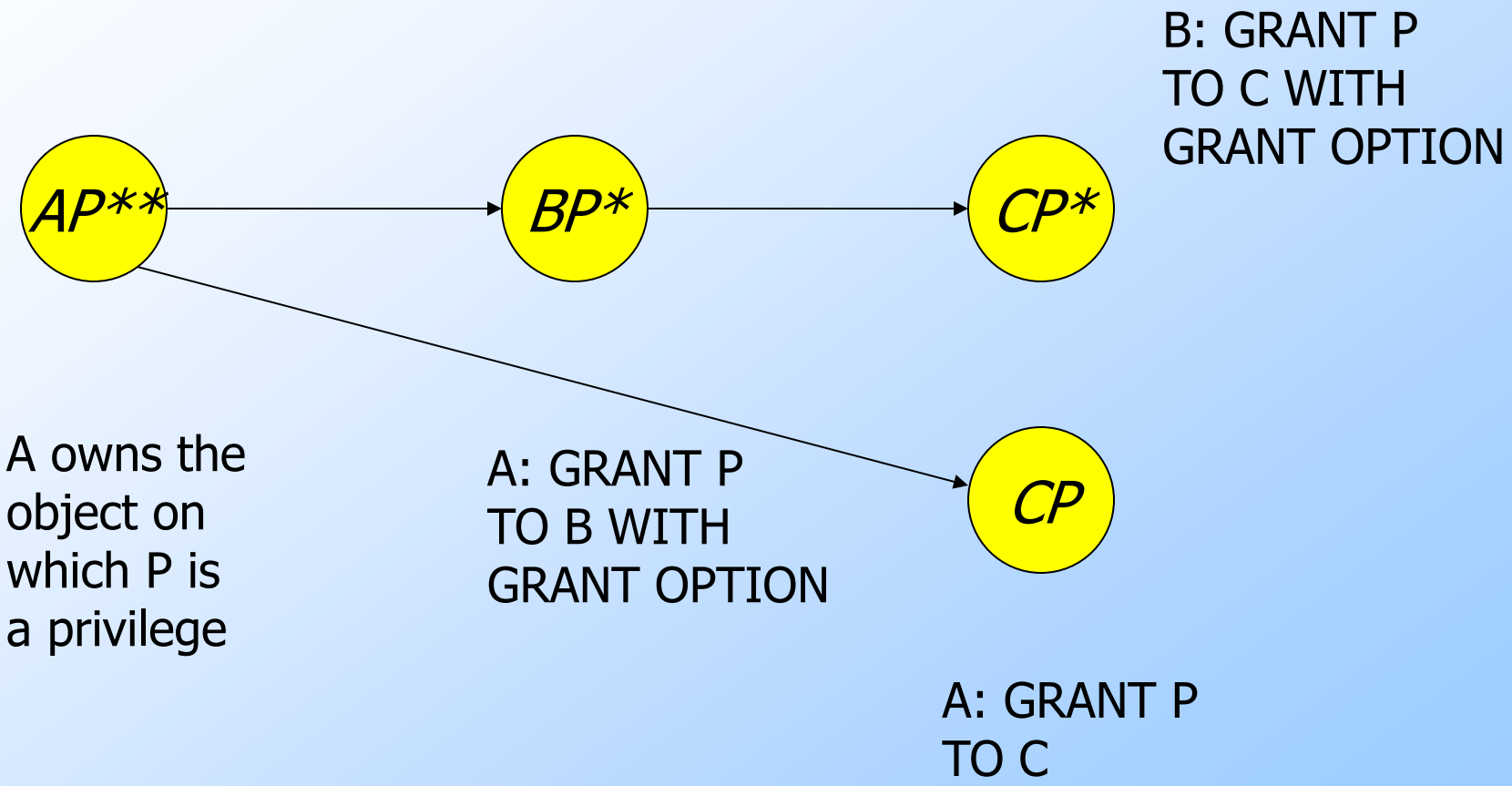
Manipulating Edges – (3)

- If *A* revokes *P* from *B* with the CASCADE option, delete the edge from *AP* to *BP*.
- But if *A* uses RESTRICT instead, and there is an edge from *BP* to anywhere, then reject the revocation and make no change to the graph.

Manipulating Edges – (4)

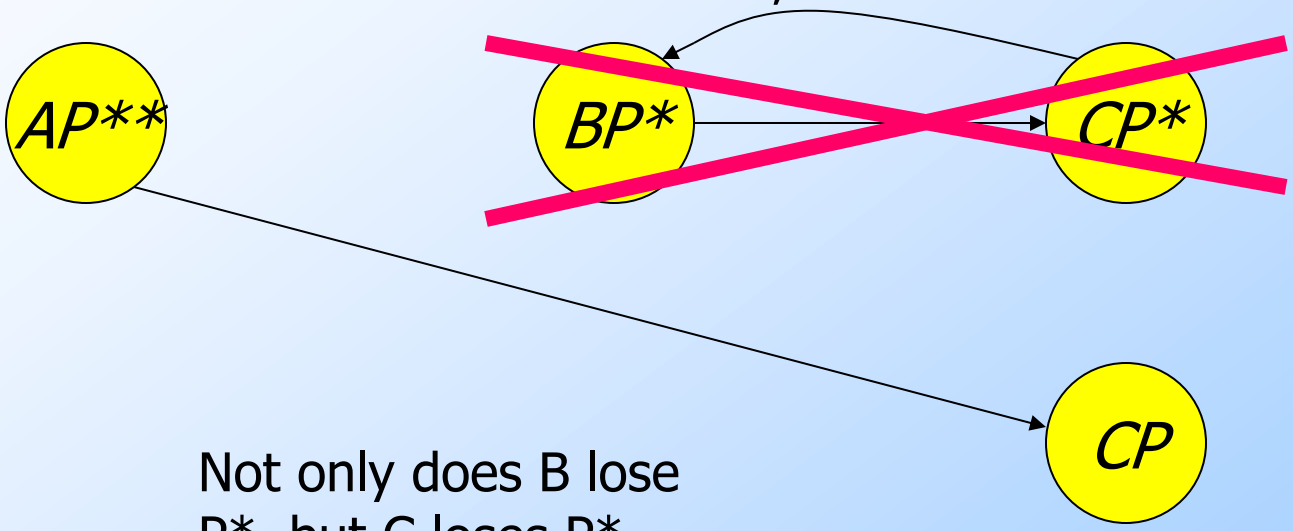
- Having revised the edges, we must check that each node has a path from some ** node, representing ownership.
- Any node with no such path represents a revoked privilege and is deleted from the diagram.

Example: Grant Diagram



Example: Grant Diagram

A executes
REVOKE P FROM B CASCADE;



Even had C passed P to B, both nodes are still cut off.

Not only does B lose P*, but C loses P*. Delete BP* and CP*.

However, C still has P without grant option because of the direct grant.

Example: GRANT

- Lets give Test SELECT privilege on Computers.Product WITH GRANT OPTION

```
GRANT select
ON computers.product
TO test@localhost
WITH GRANT OPTION;
```

Now Test has the right to issue any query on Computers.Product

AND give that right to others!

Example: GRANT

- Lets give Test SELECT privilege on

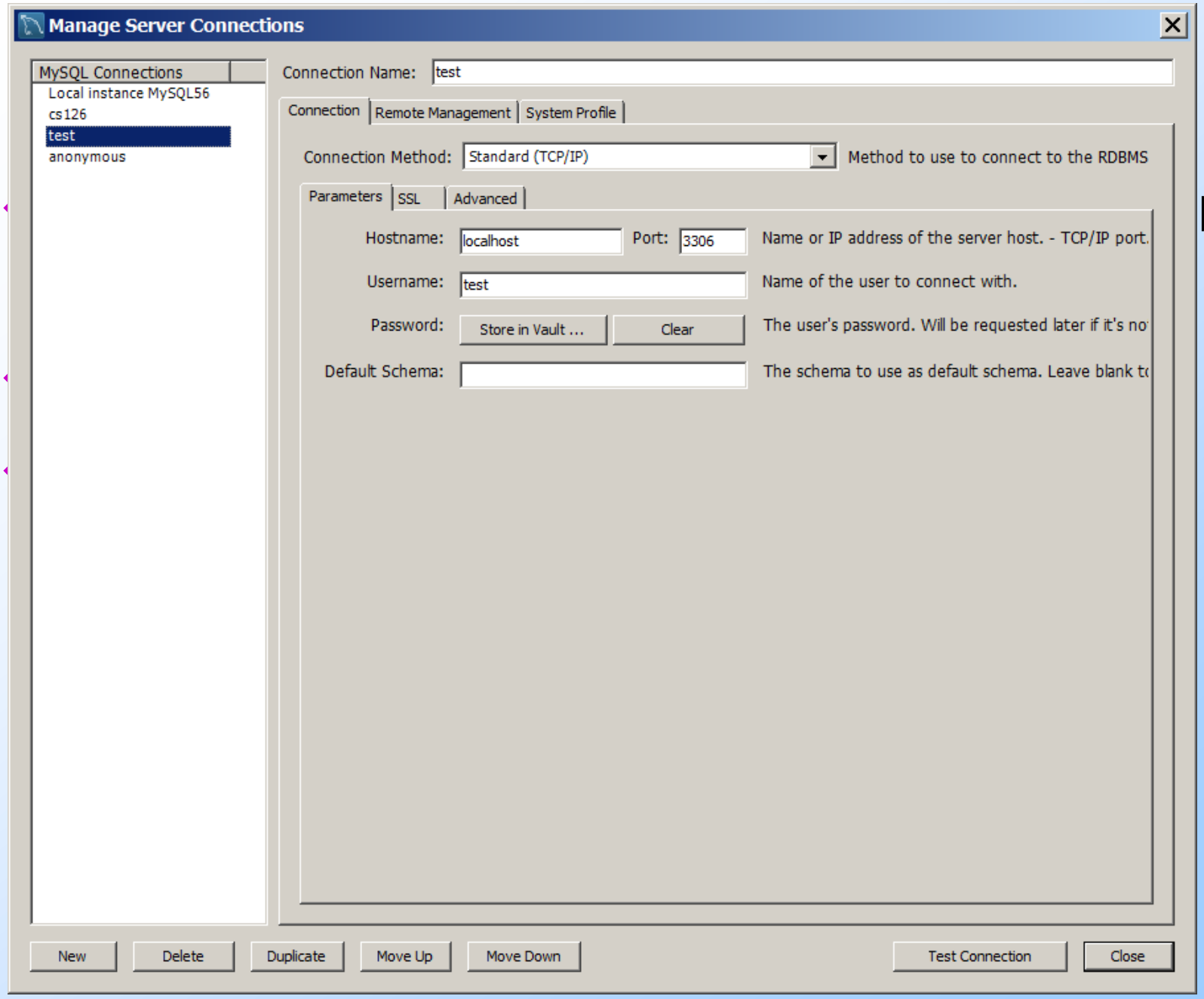
```
show grants for test@localhost;
```

Result Set Filter: Export: Wrap Cell Content:

Grants for test@localhost
GRANT USAGE ON *.* TO 'test'@'localhost' IDENTIFIED BY PASSWORD '*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29'
GRANT SELECT ON `computers`.`product` TO 'test'@'localhost' WITH GRANT OPTION

Now Test has the right to issue any query on Computers.Product

AND give that right to others!


Manage Server Connections
MySQL Connections
Local instance MySQL56
cs126
test
anonymous
Connection Name:
test
Connection
Remote Management
System Profile
Connection Method:
Standard (TCP/IP)
Method to use to connect to the RDBMS
Parameters
SSL
Advanced
Hostname:
localhost
Port:
3306
Name or IP address of the server host. - TCP/IP port.
Username:
test
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's no
Default Schema:
The schema to use as default schema. Leave blank t
New
Delete
Duplicate
Move Up
Move Down
Test Connection
Close

ry on

Example: GRANT

MySQL Workbench

cs126 × test ×

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT
- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE
- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

Filter objects

- computers
- test

SQL File 1 testing ×

```
440 • grant insert on searching.activitylog to anonymous@'%';
441 • grant insert on searching.SearchResultsRatings to anonymous@'%';
442
443 • set password = password('test99');
444 • set password for anonymous@'%' = password('test1');
445 • flush privileges;
446
447 • select current_user();
448 • drop user anonymous@'%';
```

Result Set Filter: Export: Wrap Cell Content:

current_user()
test@localhost

Result 1 ×

Output

Action Output

	Time	Action	Message
1	11:16:13	select current_user() LIMIT 0, 1000	1 row(s) returned

Example: GRANT

- Now Test has the right to issue any query on Computers.Product
- AND give that right to others!
- Now we'll Login as Test:
- Give SELECT privilege on computers.product to Anonymous@'%'

Example: GRANT

- Now Test has the right to issue any query on Computers.Product
- AND give that right to others!

```
452 • grant select on computers.product to anonymous@'%';
453
454 • SET password
455   FOR test@localhost = password('test');
456
457
458
```

Output

Action Output

	Time	Action	Message
1	11:16:13	select current_user() LIMIT 0, 1000	1 row(s) returned
2	11:19:50	grant select on computers.product to anonymous@'%'	0 row(s) affected

Example: GRANT

```
454 • show grants for test@localhost;
455 • show grants for anonymous@'%';
456
```

esult Set Filter: Export: Wrap Cell Content:

Grants for anonymous@%
GRANT USAGE ON *.* TO 'anonymous'@'%' IDENTIFIED BY PASSWORD '*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC2
GRANT SELECT ON `computers`.`product` TO 'anonymous'@'%'

Result 2

Output

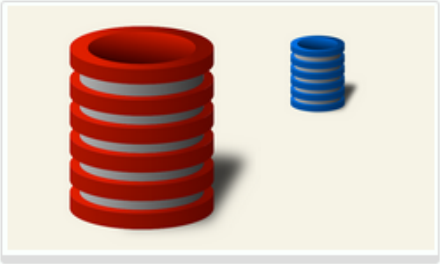
Action Output

	Time	Action	Message
2	11:11:58	grant select on computers.product to test@localhost with grant option	0 row(s) affected
3	11:12:03	show grants for test@'127.0.0.1'	Error Code: 1141. There is
4	11:12:44	show grants for test@localhost	2 row(s) returned
5	11:15:51	create user anonymous@'%' IDENTIFIED BY 'test'	0 row(s) affected
6	11:22:56	show grants for anonymous@'%'	2 row(s) returned

ry on

duct

Stanford Online



Databases

Course Started - Jun 09, 2014 at 15:00 UTC

DB12 Views and Authorization

Your final grade: **90%**.

Download Your Statement (PDF)

View Course

Email Settings *Unenroll*

Question

[Q1] The following SQL statement over tables R(a,b), S(b,c), and T(a,c) requires certain privileges to execute:

```
UPDATE R
SET a = 10
WHERE b IN (SELECT c FROM S)
AND NOT EXISTS (SELECT a FROM T WHERE T.a = R.a)
```

Which of the following privileges is **not** useful for execution of this SQL statement?

- SELECT ON T
- SELECT ON R(b)
- SELECT ON S(c)
- INSERT ON R(a)

Question

[Q1] The following SQL statement over tables R(a,b), S(b,c), and T(a,c) requires certain privileges to execute:

```
UPDATE R
SET a = 10
WHERE b IN (SELECT c FROM S)
AND NOT EXISTS (SELECT a FROM T WHERE T.a = R.a)
```

Which of the following privileges is **not** useful for execution of this SQL statement?

- ○ SELECT ON T
- ○ SELECT ON R(b)
- ○ SELECT ON S(c)
- ◉ INSERT ON R(a) ✔

Question

[Q2] Consider a set of users A, B, C, D, E. Suppose user A creates a table T and thus is the owner of T. Now suppose the following set of statements is executed in order:

```
1. User A: grant update on T to B,C with grant option
2. User B: grant update on T to D with grant option
3. User C: grant update on T to D with grant option
4. User D: grant update on T to E
5. User A: revoke update on T from C cascade
```

After execution of statement 5, which of the following is true?

> D has privilege UPDATE ON T, but without grant option
>
> D and E do not have privilege UPDATE ON T, but B does
>
> E has privilege UPDATE ON T
>
> D no longer has privilege UPDATE ON T