
Advanced Data Base (8trd157)

Lab3 (phase 3 of 4) Validation of the Multi-User Data Model (1 week)

Paul Girard, Ph.D.

Objectives

- Based on the tables and views created in lab2 and the 5 partial relational data models of lab1, create a file *privilege.sql* to grant the required privileges to each application analyst. These privileges will give them the type of access needed to support the programming of their own user transactions corresponding to their partial data model (*ref. transaction table of each user type*). Each user will work in his own account but he will be able to access to database objects owned by another user of the team.
- Create a command file *.sql* for each user transaction mentioned further and give it an appropriate name. These commands files will be interactive.
- Test all user transactions at the same time on different sessions (*sqlplus server*)

Methodology

1. Each student in a team should have a given responsibility :
 - DBA,
 - Application programmer for the technician in the maintenance department,
 - Application programmer for the storekeeper in the store department,
 - Application programmer for the storekeeper supervisor in the store department,
 - Application programmer for the purchasing agent in the purchasing department,
 - Application programmer for the purchasing department supervisor.

A student may be responsible of more than a task but he must have the required privileges to do his work by checking the transaction table and the relational partial data model for each user type supported. Each team may have between 3 (*min*) and 5 (*max*) students.

Example of task assignment with a team of 4 students

analyst 1 : DBA, PDS and SKS transactions
analyst 2 : Purchasing agents transactions
analyst 3 : Storekeepers transactions
analyst 4 : Maintenance technicians transactions

Example of task assignment with a team of 3 students

analyst 1 : DBA, PDS, SKS, technicians transactions
analyst 2 : Purchasing agents transactions
analyst 3 : Storekeepers transactions

- Only one student should be responsible for any modification to the database (*tables, views, index, privileges, load, ...*). The database will be in the DBA user account. Any modification requested by another team member will be done by the DBA. Use email to describe your needs. The file **privilege.sql** controlled by the DBA will contain all grant for all tables and all his team members. Create another file **no_privilege.sql** to revoke all privileges. See *Annex 2* for examples.
- From your own user account, create a command file *.sql* for each transaction listed under a user type. The name of the transaction will identify the command file : *lisagent.sql, creagent.sql, quant.sql, respon.sql, modresp.sql* are the 5 command files for the *Purchasing Department Supervisor*

Purchasing Department Supervisor

Transaction	Description	File
lisagent	Display all purchasing agents (<i>emp_num, emp_name</i>)	lisagent.sql
creagent	Create a new agent (<i>emp_num, emp_name</i>)	creagent.sql
quant	Display the number of purchase orders done by a given agent (<i>emp_num</i>) and the total value of these purchase orders.	quant.sql
respon	Display for each agent (<i>emp_num, emp_name</i>) the list of all parts (<i>part_id, part_name</i>) under his responsibility	respon.sql
modresp	Change the responsibility of a given part (<i>part_id</i>) from a agent (<i>emp_num1</i>) to another agent (<i>emp_num2</i>)	modresp.sql

Maintenance Technician

Transaction	Description	File
lispart	Display all attributes (<i>part_id, name, unit, unit_price, stock_qty, order_qty, min_qty</i>) of a given part (<i>part_id</i>)	lispart.sql
explosion	Display all components (<i>part_id, name</i>) of a given part (<i>part_id</i>)	explosion.sql
implosion	Display all parts (<i>part_id, name</i>) where a given part (<i>part_id</i>) is a component	implosion.sql
crecomp	Creation of a component (<i>part_id</i>) of a given part (<i>part_id</i>)	crecomp.sql
cancomp	Cancel a component (<i>part_id</i>) of a given part (<i>part_id</i>)	cancomp.sql

Storekeepers Supervisor

Transaction	Description	File
invent	Display the number of parts having a different serial number AND the total number of parts in stock	invent.sql
value	Display the total value of parts in stock	value.sql

Storekeeper

Transaction	Description	File
lispart	Display all attributes (<i>part_id, name, unit, unit_price, stock_qty, order_qty, min_qty</i>) of a given part (<i>part_id</i>)	lispart.sql
part_out	Part (<i>part_id, qty</i>) is updated when some parts (n) are taken out from the store to the technician	part_out.sql
prod_in	Products (<i>product_id, qty_received</i>) is delivered by the supplier following a purchase order (<i>purchase_order_number</i>); the corresponding part is updated (<i>stock_qty, order_qty</i>)	prod_in.sql
list_po	Display the contents of a purchase order (<i>po_number</i>): <i>po_number, date purchasing agent_name, supplier number, name, address, contact</i> , list of all ordered products (<i>product_id, name, unit, unit_price, qty ordered, qty_received</i>), <i>total, status</i> of a purchase order.	lis_po.sql

Purchasing agent

Transaction	Description	File
crepart	Creation of a part (<i>part_id, name, unit, unit_price, qty_min, qty_order qty_stock</i>). Appropriate default values to some attributes.	crepart.sql
lispart	Display all attributes (<i>part_id, name, unit, unit_price, stock_qty, order_qty, min_qty</i>) of a given part (<i>part_id</i>)	lispart.sql
cresupp	Creation of a supplier (<i>supplier-id, name, addr, contact</i>)	cresupp.sql
creprod	Creation of a product (<i>prod_id, prod_name, unit_prod, unit_prod_cost</i>) of a given supplier (<i>supplier_id</i>) that could be ordered for a given part (<i>part_id</i>)	creprod.sql
low_part	Display the list of parts (<i>part_id, name</i>) under the responsibility of a given purchasing agent (<i>emp_num</i>) and having, $stock_qty + ordered_qty \leq qty_min$	low_part.sql
pot_supp	Display the list of potential suppliers products (<i>supplier_id, product_id, unit_prod, unit_prod_cost</i>) for a given part (<i>part_id</i>)	pot_supp.sql
cre_po	Creation of a purchase order (<i>po_number, pa_name, ...date, supplier info</i> , and for each product (<i>product_id, qty ordered</i>); calculate the total of the purchase order. The status is a default (<i>not completed</i>).	cre_po.sql

4. In your own account and using EditPlus, create each file one by one to support one transaction at a time and test it with **sqlplus server**. When all user transactions of one type is completed, test them in parallel using many sessions with the database to see the locks (*normal or deadlock*) that could be created. Many users of the same user type will execute them in any order. You must have the privileges to access your tables and views from your DBA.

note: Do not forget that the last purchase order always fixes the unit price and the ordered quantity of each part affected by each ordered product of that purchase order. You have to consider the product unit (*integer*) which a multiple of the part unit (alphanumeric). The delivery of the products by the supplier will update the *qty received* for each ordered product and update *qty ordered* and *stock qty* of the correspondind part unit.

-
5. After all transactions have been tested by each application analyst, you should simulate a production database in a multi-user environment. The following transactions could be tested in 3-4 different sessions in parallel : crepart, respon, lispart, cresupp, creprod corresponding to the last part, pot_supplier on that given part, cre_po, lispart , lis_po, prod_in, lispart, prod_out, lispart, crecomp, explosion, implosion, cancomp, explosion, quant, value, invent.

Format of the team report

Keep your documentation up to date (*E/R and relational data models, global and partials*), insert all *transaction.sql* for each user type immediately after their own transaction table with an execution example (*spool file*). Make sure that creation files for tables and views, load and initialization files are correct. Display the contents of your security files *privilege.sql* and *no_privilege.sql*.

Annex 1

sqlplus useful commands and examples

sqlplus> clear buffer	deletes all lines
> get file	place the contents of a command <i>.sql</i> in the buffer,
> list	lists all lines in the SQL buffer
> @ file	execute the file <i>file.sql</i>
> set verify on (off)	display (<i>or not</i>) the lines before and after the substitution of variables
> prompt text	display contents of <i>text</i>
> pause text	display contents of <i>text</i> and wait for a keyboard "return"
> accept variable format	receive the input <i>variable</i> from the user
> accept variable prompt 'text'	display a text and receive the input <i>variable</i> from the user
> column col format format heading 'title'	modify the display format and title of the column <i>col</i>
> set define on	Defines the substitution character (<i>by default the ampersand "&"</i>) and turns substitution on and off.
> exit	exit sqlplus
> /* */	comments on one or multiple lines
> --	comments on one line

Example 1 ==> *lispart.sql*

```
-- Example of reformatting and input variable

set define on
set verify off
prompt Enter a part number please
accept num number prompt 'number : '
column unit_price format $99,999.99 heading 'price'
column part_name heading 'Name of part'
-- execution de la requete sql
select part_id "Number", part_name, stock_qty "quant in stock", unit_price
from part where part_id=&num;
```

Execution of *lispart.sql* with *set verify off*

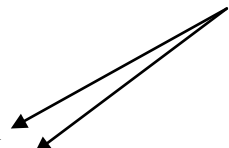
```
SQL> @lispart
Enter a part number please
number : 1001

Number          Name of part  quant in stock  price
-----
1001            'motor 455'      0             $2,450.45
```

Execution of *lispart.sql* with *set verify on*

```
SQL> @lispart
Enter a part number please
number : 1001
old 2: from part where part_id=&num
new 2: from part where part_id= 1001
```

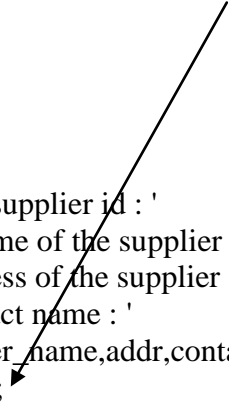
Number	Name of part	quant in stock	price
1001	'motor 455'	0	\$2,450.45



Example 2 ==> *cresupp.sql*

(note that a character datatype needs 2 apostrophes in values

```
-- Request to create a new supplier
set define on
set verify off
prompt Creation of a new supplier
prompt
accept num number prompt 'Enter the supplier id : '
accept name char prompt 'Enter the name of the supplier : '
accept adr char prompt 'Enter the address of the supplier : '
accept rep char prompt 'Enter the contact name : '
insert into supplier (supplier_id,supplier_name,addr,contact)
values (&num,'&name','&adr','&rep');
```



Execution of *cresupp.sql*

```
SQL> @cresupp
Creation of a new supplier

Enter the supplier id : : 3
Enter the name of the supplier : Reno Depot
Enter the address of the supplier : Tianjin
Enter the contact name : Leo Bouchard

1 row created.
```

Annex 2

Access privilege to table and view

1. The command **GRANT** give the required privileges on a table or a view to a group of users

```
sql> grant priv1, priv2, ... on objet to user1, user2,...
```

Ex1 *sql> grant delete, select, update on part to ora11000, ora11062;*

==> ora11000 and ora11062 will be able to do a delete, a select and an update on the table (or view) part.

However, for example, ora11000 will have to mention the schema owner of the table part in his sql request :

select * from ora11004.part; *if ora11004 is the owner of the table part.*

2. Revoke all or some privileges already given to a group of users for a table or a view

```
sql> revoke priv1, priv2, ... on objet from user1, user1,...
```

Ex2 *sql> revoke all on part from ora11000, ora11062;*

Ex3 *sql> revoke delete, update on part from ora11062;*

3. The following privileges may be granted for a table or a view

Privilege	Table	View
ALTER	X	
DELETE	X	X
INDEX	X	
INSERT	X	X
REFERENCES	X	
SELECT	X	X
UPDATE	X	X