Intermediate SQL

Engr. Laraib Siddiqui

View

- It is not always desirable for all users to see the entire set of relations in the database.

- We may wish to create a personalized collection of "virtual" relations that is better matched to a certain user's intuition of the structure of the enterprise.

View

We define a view in SQL by using the create view command. To define a view, we must give the view a name and must state the query that computes the view.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
```

Example

```
CREATE VIEW agentview AS
SELECT *
FROM agents;
```

Practice

Customers

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007

salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

1. Write a query to create a view for all salesmen with columns salesman_id, name and city.
2. Write a query to create a view to getting a count of how many customers we have at each level of a grade.
3. Write a query to create a view that shows the average and total orders for each salesman after his or her name. (Assume all names are unique)

Orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001

Index

- Many queries reference only a small proportion of the records in a file. For example, a query like "Find all instructors in the Physics department".
- It is inefficient for the system to read every record and to check the *building* field for the value "Physics". Here comes the role of index.
- An index on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation.
- For example, if we create an index on attribute *dept name* of relation *instructor*, the database system can find the record with any specified *dept name* value, such as "Physics", or "Music", directly, without reading all the tuples of the *instructor* relation.

Index

We create an index with the create index command and delete index through drop index command.

CREATE INDEX *index_name*
ON *table_name* (*column1*, *column2*, ...);

Example:

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

DROP INDEX *table_name.index_name;*

Example:

```
DROP INDEX Persons.idx_pname;
```

Practice

driver (id, full_name, year_born, city, rating, ride_count)

1. Create an index to search for drivers based on their name.
2. Create an index to filter for drivers who have completed a certain number of rides.

Authorization

- We may assign a user several forms of authorizations on parts of the database. Authorizations on data include:
  - Authorization to read data.
  - Authorization to insert new data.
  - Authorization to update data.
  - Authorization to delete data.
- Each of these types of authorizations is called a privilege. We may authorize the user all, none, or a combination of these types of privileges on specified parts of a database, such as a relation or a view.

Granting and Revoking of Privileges

- The SQL standard includes the privileges select, insert, update, and delete. It includes commands to grant and revoke privileges.
- The grant statement is used to confer authorization. The basic form of this statement is:

  grant <privilege list>
  on <relation name or view name>
  to <user/role list>;

  Example: grant select on *department* to *Ali*;
- To revoke an authorization, we use the revoke statement.

  revoke <privilege list>
  on <relation name or view name>
  from <user/role list>;

  Example: revoke update (budget) on *department* from *Asad*;

Role

- Consider the real-world roles of various people in a university. Each instructor must have the same types of authorizations on the same set of relations. Whenever a new instructor is appointed, she will have to be given all these authorizations individually.
- A better approach would be to specify the authorizations that every instructor is to be given, and to identify separately which database users are instructors.
- The system can use these two pieces of information to determine the authorizations of each instructor. When a new instructor is hired, a user identifier must be allocated to him, and he must be identified as an instructor.
- Individual permissions given to instructors need not be specified again.

Role

- Any authorization that can be granted to a user can be granted to a role. Roles are granted to users just as authorizations are.
- Roles can be created in SQL as follows:

  create role *instructor*;
- Roles can then be granted privileges just as the users can, as illustrated in this statement:

  grant select on *takes*

  to *instructor*;

Practice

Hotel (hotelno, hotelname, city)
Room (roomno, hotelno, Type, price)
Booking (hotelno, guestno, datefrom, dateto, roomno)
Guest (guestno, guestname, guestaddress)

1. Create a view containing the hotel name and the names of the guests staying at the hotel.
2. Create a view containing the account for each guest at the Gorsvenor Hotel.
3. Give the users Manager and Director full access to these view, with the privilege to pass the access on to other users.
4. Give the user Accounts SELECT access to these views. Now revoke the access from this user.