CS344
Database Management Systems

Lecture Notes : 11th AUGUST

Topic:- INTRODUCTION TO SQL

Submitted by -

VIRENDRA KUMAR
09010158

ANIK CHATTOPADHYAY
09010160

Background:-

IBM developed the original version of SQL at its San Jose Research Laboratory (now the Almaden Research Center).IBM implemented the language, originally called Sequel, as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query Language). Today SQL is the most influencial commercially marketed query language. SQL uses a combination of relational-algebra and relational-calculus constructs.
Although we refer to the SQL language as a “query language,” it can do much more than just query a database. It can define the structure of the data, modify data in the database, and specify security constraints. Here we discuss the fundamental constructs and concepts of SQL.

How to use **create table**:-

Here we give an example creating a table using SQL commands.The Table name is "customer" and its different attributes are written with proper types.We also specify the primary key, candidate key and foreign key at the bottom(although it is not mandatory).

create table *customer*
(*customer-name* **char**(20),
customer-street **char**(30),
customer-city **char**(30),

primary key (*customer-name*))

How to Use **alter table** :-

alter table command is used to add attributes to an existing relation and drop attributes from an existing relation.

Command to ADD attributes:-

alter table *r* **add** *A D*

Command to DROP attributes:-

alter table *r* **drop** *A*

where, *r* - the name of an existing relation,
A - the name of the attribute to be added or dropped,
D- the domain of the added attribute

Insertion:-

The simplest **insert** statement is a request to insert one tuple. Suppose that we wish to insert the fact that there is an account A-9732 at the Perryridge branch and that has a balance of $1200.We write

insert into *account*
values ('A-9732', 'Perryridge', 1200)

In this example, the values are specified in the order in which the corresponding attributes are listed in the relation schema.

For the benefit of users who may not remember the order of the attributes, SQL allows the attributes to be specified as part of the **insert** statement. For example, the following SQL **insert** statements are identical in function to the preceding one:

insert into *account* (*account-number, branch-name, balance*)
values (’A-9732’, ’Perryridge’, 1200)

insert into *account* (*branch-name, account-number, balance*)
values (’Perryridge’, ’A-9732’, 1200)

Updates:-

Suppose that annual interest payments are being made, and all balances are to be increased by 5 percent.For this updation in account table we write the following command

update *account*
set *balance* = *balance* * 1.05

If interest is to be paid only to accounts with a balance of $1000 or more, we can write

update *account*

set *balance* = *balance* * 1.05
where *balance* >= 1000

Deletion:-

We can delete only whole tuples; we cannot delete values on only particular attributes. SQL expresses a deletion by

delete from *r*
where *P*

where,*P* represents a predicate and *r* represents a relation. And to delete the whole table we need to write like:
drop table table_name

The Union Operation:-

To find all customers having a loan, an account, or both at the bank, we write

(**select** *customer-name*
from *depositor*)
union
(**select** *customer-name*
from *borrower*)

The **union** operation automatically eliminates duplicates.

If we want to retain all duplicates, we must write **union all** in place of **union**:

(**select** *customer-name*
from *depositor*)
union all
(**select** *customer-name*
from *borrower*)

The from Clause:-

The **from** clause by itself defines a Cartesian product of the relations in the clause.
Since the natural join is defined in terms of a Cartesian product,a selection, and a projection.
Now,write an SQL expression for the natural join.
Let an relational-algebra expression:-

Π*customer-name, loan-number, amount* (*borrower* ⋈ *loan*) for the query “For all customers who have a loan from the bank, find their names, loan numbers and loan amount.”
In SQL, this query can be written as:

select *customer-name, borrower.loan-number, amount*
from *borrower, loan*
where *borrower.loan-number = loan.loan-number*

Notice that SQL uses the notation *relation-name.attribute-name*, as does the relational algebra, to avoid ambiguity in cases where an attribute appears in the schema of more than one relation.

The Projection Operation:-

The **select** clause corresponds to the projection operation of the relational algebra. For example

select dept_name from Student

- this statement extracts the column named **dept_name** from **Student** table. This may include duplicates. But the clause **select distinct** avoids duplicates.

select distinct dept_name from Student

this will bring out only distinct **dept_name**s from **student** table.

The asterisk symbol " * " can be used to denote "all attributes." Like,

select * from student where id > 50 **and** student_name = 'Rahul'

Will entire rows where the condition matches. Also note that SQL uses the logical connectives **and**, **or**, and **not.**

Summary:-

Apart from selection,projection and other relatioal operations we can apply some mathematical functions on the values of attributes in a table in SQL.These are called aggregate functions.SQL offers five built-in aggregate functions:

- Average: **avg**
- Minimum: **min**
- Maximum: **max**
- Total: **sum**
- Count: **count**

As an illustration, consider the query “Find the average account balance at the Perryridge branch.”We write this query as follows:

select avg (*balance*)
from *account*
where *branch-name* = ’Perryridge’

==========END==========