More SQL Data Definition

Database Systems Lecture 6

Munawar, PhD

In this Lecture

- More SQL
  - DROP TABLE
  - ALTER TABLE
  - INSERT, UPDATE, and DELETE
  - Data dictionary
  - Sequences
- For more information
  - Connolly and Begg chapters 5 and 6

Creating Tables

- From last lecture…
  - CREATE TABLE
  - Columns
    - Data types
    - [NOT] NULL, DEFAULT values
  - Constraints
    - Primary keys
    - Unique columns
    - Foreign keys

```
CREATE TABLE
<name> (
  <col-def-1>,
  <col-def-2>,
        :
  <col-def-n>,
  <constraint-1>,
        :
  <constraint-k>)
```

Deleting Tables

- To delete a table use
- `DROP TABLE`
- `  [IF EXISTS]`
- `  <name>`
- Example:
- `DROP TABLE Module`

- **BE CAREFUL** with any SQL statement with DROP in it
  - You will delete any information in the table as well
  - You won't normally be asked to confirm
  - There is no easy way to undo the changes

Changing Tables

- Sometimes you want to change the structure of an existing table
  - One way is to DROP it then rebuild it
  - This is dangerous, so there is the ALTER TABLE command instead
- ALTER TABLE can
  - Add a new column
  - Remove an existing column
  - Add a new constraint
  - Remove an existing constraint

ALTERing Columns

To add or remove columns use

```
ALTER TABLE <table>
 ADD COLUMN <col>

ALTER TABLE <table>
 DROP COLUMN <name>
```

Examples

```
ALTER TABLE Student
  ADD COLUMN
  Degree VARCHAR(50)

ALTER TABLE Student
  DROP COLUMN Degree
```

ALTERing Constraints

To add or remove columns use

```
ALTER TABLE <table>
   ADD CONSTRAINT
         <definition>

ALTER TABLE <table>
   DROP CONSTRAINT
          <name>
```

Examples

```
ALTER TABLE Module
   ADD CONSTRAINT
   ck UNIQUE (title)

ALTER TABLE Module
   DROP CONSTRAINT ck
```

INSERT, UPDATE, DELETE

- `INSERT` - add a row to a table
- `UPDATE` - change row(s) in a table
- `DELETE` - remove row(s) from a table
- `UPDATE` and `DELETE` use '`WHERE` clauses' to specify which rows to change or remove
- BE CAREFUL with these - an incorrect `WHERE` clause can destroy lots of data

INSERT

```
INSERT INTO
   <table>
   (col1, col2, …)
   VALUES
   (val1, val2, …)
```

- The number of columns and values must be the same
- If you are adding a value to every column, you don’t have to list them
- SQL doesn’t require that all rows are different (unless a constraint says so)

INSERT

Student

ID	Name	Year
1	John	1

```
INSERT INTO Student
 (ID, Name, Year)
 VALUES (2, 'Mary', 3)
```

Student

ID	Name	Year
1 2	John Mary	1 3

```
INSERT INTO Student
 (Name, ID)
 VALUES ('Mary', 2)
```

Student

ID	Name	Year
1 2	John Mary	1

```
INSERT INTO Student
 VALUES (2, 'Mary', 3)
```

Student

ID	Name	Year
1 2	John Mary	1 3

UPDATE

```
UPDATE <table>
SET col1 = val1
  [,col2 = val2…]
[WHERE
  <condition>]
```

- All rows where the condition is true have the columns set to the given values
- If no condition is given all rows are changed so BE CAREFUL
- Values are constants or can be computed from columns

UPDATE

Student

ID	Name	Year
1	John	1
2	Mark	3
3	Anne	2
4	Mary	2

```
UPDATE Student
SET Year = 1,
    Name = 'Jane'
WHERE ID = 4
```

Student

ID	Name	Year
1	John	1
2	Mark	3
3	Anne	2
4	Jane	1

```
UPDATE Student
SET Year = Year + 1
```

Student

ID	Name	Year
1	John	2
2	Mark	4
3	Anne	3
4	Mary	3

DELETE

- Removes all rows which satisfy the condition

```
DELETE FROM
   <table>
   [WHERE
    <condition>]
```

- If no condition is given then ALL rows are deleted - BE CAREFUL
- Some versions of SQL also have **`TRUNCATE TABLE <T>`** which is like **`DELETE FROM <T>`** but it is quicker as it doesn't record its actions

DELETE

Student

ID	Name	Year
1	John	1
2	Mark	3
3	Anne	2
4	Mary	2

```
DELETE FROM
  Student
  WHERE Year = 2
```

Student

ID	Name	Year
1	John	1
2	Mark	3

```
DELETE FROM Student
```

or

```
TRUNCATE TABLE Student
```

Student

ID	Name	Year

SELECT

- The SQL command you will use most often
  - Queries a set of tables and returns results as a table
  - Lots of options, we will look at many of them
  - Usually more than one way to do any given query
- SQL’s SELECT is different from the relational algebra’s selection σ
  - SELECT in SQL does all of the relational algebra
  - But it is a bit different because SQL differs from the relational model

SQL SELECT Overview

```
SELECT
  [DISTINCT | ALL] <column-list>
  FROM <table-names>
  [WHERE <condition>]
  [ORDER BY <column-list>]
  [GROUP BY <column-list>]
  [HAVING <condition>]
```

- *([]- optional, | - or)*

Simple SELECT

```
SELECT <columns>
  FROM <table>
```

`<columns>` can be

- A single column
- A comma-separated list of columns
- `*` for ‘all columns’

- Given a table Student with columns
  - stuID
  - stuName
  - stuAddress
  - stuYear

Sample SELECTs

```
SELECT * FROM Student
```

stuID	*stuName*	*stuAddress*	*stuYear*
1	Anderson	15 High St	1
2	Brooks	27 Queen's Rd	3
3	Chen	Lenton Hall	1
4	D'Angelo	Derby Hall	1
5	Evans	Lenton Hall	2
6	Franklin	13 Elm St	3
7	Gandhi	Lenton Hall	1
8	Harrison	Derby Hall	1

Sample SELECTs

```
SELECT stuName FROM Student
```

stuName
Anderson
Brooks
Chen
D'Angelo
Evans
Franklin
Gandhi
Harrison

Sample SELECTs

```
SELECT stuName, stuAddress
  FROM Student
```

stuName	*stuAddress*
Anderson	15 High St
Brooks	27 Queen's Rd
Chen	Lenton Hall
D'Angelo	Derby Hall
Evans	Lenton Hall
Franklin	13 Elm St
Gandhi	Lenton Hall
Harrison	Derby Hall

Being Careful

- When using DELETE and UPDATE
  - You need to be careful to have the right WHERE clause
  - You can check it by running a SELECT statement with the same WHERE clause first

Before running

```
DELETE FROM Student
    WHERE Year = 3
```

run

```
SELECT * FROM Student
    WHERE Year = 3
```

Sequences

- Often we want to assign each row a unique number
  - These are useful as primary keys
  - Using integers to reference rows is more efficient
  - We would like the DBMS to do this
- In most versions of SQL we can use autoincrementing fields to do this
  - Details differ between versions
  - Usually the first entry is assigned 1, the next 2, and so on, but Oracle lets you change this

Sequences

- In Oracle we use a *Sequence*
    - A sequence is a source of numbers
    - We can declare several sequences, giving each a name, a start point, and a step size
    - We can then generate unique numbers by asking for the next element from a sequence

Sequences in Oracle

- To declare a sequence:

```
CREATE SEQUENCE <name>
    [START WITH <value>]
    [INCREMENT BY <value>]
```

  - If no **`START WITH`** or **`INCREMENT BY`** values are given they default to 1
- To get the next value from a sequence

```
<sequence name>.nextVal
```

Sequence Example

- Creating a sequence

```
CREATE SEQUENCE mySeq START WITH 1
```

- Using a sequence

```
SELECT mySeq.nextVal FROM DUAL;
INSERT INTO Student
   (stuID, stuName, stuAddress)
VALUES
   (mySeq.nextVal, 'Steve Mills',
    '13 Elm Street')
```

SQL and the Data Dictionary

- The *data dictionary* or *catalogue* stores
  - Information about database tables
  - Information about the columns of tables
  - Other information - users, locks, indexes, and more
  - This is 'metadata'

- Some DBMSs let you query the catalogue
  - In Oracle you can access the metadata in several ways
  - There are 'system tables' with metadata in them
  - You can also **`DESCRIBE`** tables

Oracle Data Dictionary

- To find out what tables and sequences you have defined use

```
SELECT table_name
   FROM user_tables
```

  - The user_tables table is maintained by Oracle
  - It has *lots* of columns, so don't use

```
    SELECT * FROM user_tables
```

Oracle Data Dictionary

- To find the details of a table use

```
DESCRIBE <table name>
```

- Example:

```
SQL> DESCRIBE Student;
 Name         Null?    Type
 ------------ -------- ----------
 STUID        NOT NULL NUMBER(38)
 STUNAME      NOT NULL VARCHAR2(50)
 STUADDRESS            VARCHAR2(50)
 STUYEAR               NUMBER(38)
```

This Lecture in Exams

Track

cID	Num	Title	Time	aID
1	1	Violent	239	1
1	2	Every Girl	410	1
1	3	Breather	217	1
1	4	Part of Me	279	1
2	1	Star	362	1
2	2	Teaboy	417	2

CD

cID	Title	Price
1	Mix	9.99
2	Compilation	12.99

Artist

aID	Name
1	Stellar
2	Cloudboy

This Lecture in Exams

Add $2.50 to the price of all CDs that cost more than $10.00.

(2 marks)

Add a new column, Genre, to the CD table. This column should hold a string of up to 100 characters, and if no genre is provided then it should default to the value “Unknown”.

(3 marks)

Add a track titled “Runnin” by the artist “Fat Freddy’s Drop” which is 12 minutes and 27 second long to the CD titled “Compilation”. For this part only, you may assume that the tables contain exactly the information shown above.

(3 marks)

Next Lecture

- SQL SELECT
  - WHERE clauses
  - SELECT from multiple tables
  - JOINs
- For more information
  - Connolly and Begg Chapter 5
  - Ullman and Widom, Chapter 6.5,6.1.