

Tuesday, 22 October 2013

General Assembly

Practical SQL

Table Manipulations



INSERT Table Data

1. Adding President Obama to presidents table

```
1 INSERT INTO
2   sampdb.president
3
4   SET
5     last_name   = 'Obama',
6     first_name  = 'Barack',
7     suffix      = NULL,
8     city        = 'Honolulu',
9     state       = 'HI',
10    birth       = '1961-08-04',
11    death       = NULL
12
13 ;
14
```

2. Check:

```
1 SELECT * FROM sampdb.president WHERE last_name = 'Obama';
2
```

1	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----
2		last_name		first_name		suffix		city		state		birth
3	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----
4		Obama		Barack		NULL		Honolulu		HI		1961-08-04
5	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----
6												

Alternate INSERT Syntax

1. That entry is wrong, so let's add the right one
2. **VALUES** provides another syntax for adding data

```
1 INSERT INTO
2   sampdb.president
3 VALUES ('Obama', 'Barack', NULL, 'Chicago', 'IL', '1961-08-04',
4 ;
5
```

3. **Check:**

```
1 SELECT * FROM sampdb.president WHERE last_name = 'Obama';
2
```

1	last name	first name	suffix	city	state	birth
2						
3						
4	Obama	Barack	NULL	Honolulu	HI	1961-08-04
5	Obama	Barack	NULL	Chicago	IL	1961-08-04
6						
7						

4. **VALUES** statement have the nice property of allowing multiple usages

so we can load lots of data with them.

see the database setup files for examples of this

Editing database data

1. We could also fix the problem with UPDATE

```
1 UPDATE
2     sampdb.president
3 SET
4     state = 'IL',
5     city = 'Chicago'
6 WHERE
7     last_name = 'Obama'
8 ;
9
10
11
```

1	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----
2		last_name		first_name		suffix		city		state		birth
3	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----
4		Obama		Barack		NULL		Chicago		IL		1961-08-04
5		Obama		Barack		NULL		Chicago		IL		1961-08-04
6	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----
7												

DELETE -- Removing Records

1. Now we have two entries, let's remove one

```
1 DELETE FROM sampdb.president WHERE last_name = 'Obama' LIMIT 1;  
2
```

```
1 SELECT * FROM sampdb.president WHERE last_name = 'Obama';  
2
```

```
1 +-----+-----+-----+-----+-----+-----+  
2 | last_name | first_name | suffix | city    | state | birth  
3 +-----+-----+-----+-----+-----+-----+  
4 | Obama     | Barack    | NULL   | Chicago | IL    | 1961-08-04  
5 +-----+-----+-----+-----+-----+-----+  
6
```

DELETE -- Removing Many Records

1. Without the LIMIT, DELETE takes all records

1	DELETE FROM sampdb.president WHERE last_name = 'Obama';
2	

1	SELECT * FROM sampdb.president WHERE last_name = 'Obama';
2	

1	
2	

DELETE -- Removing LOTS Many Records

1. Be careful with DELETE!!

```
1 DELETE FROM sampdb.president;
2
1 SELECT
2   COUNT(*)
3 FROM
4   sampdb.president;
5
1  +-----+
2  | COUNT(*) |
3  +-----+
4  |         0 |
5  +-----+
6
```

2. Here, not a big deal:

```
1 INSERT INTO sampdb.president
2   VALUES
3   ('Washington','George',NULL,'Wakefield','VA','1732-02-22'
4   ('Adams','John',NULL,'Braintree','MA','1735-10-30','1826-
5   ('Jefferson','Thomas',NULL,'Albemarle County','VA','1743-
6   ('Madison','James',NULL,'Port Conway','VA','1751-03-16','
7   ('Monroe','James',NULL,'Westmoreland County','VA','1758-0
8   ' ' '
9 ;
10
```

3. This is why databases have permissions.

4. I don't leave uncommented DELETE statements lying around.

DROP neither.

Databases can protect against these mistakes

```
1 DROP TABLE IF EXISTS sampdb.president;  
2 CREATE TABLE sampdb.president  
3 (  
4     last_name  VARCHAR(15) NOT NULL,  
5     first_name VARCHAR(15) NOT NULL,  
6     suffix     VARCHAR(5) NULL,  
7     city       VARCHAR(20) NOT NULL,  
8     state      VARCHAR(2) NOT NULL,  
9     birth      DATE NOT NULL,  
10    death      DATE NULL,  
11    UNIQUE (last_name, first_name)  
12 );  
13
```

1. **UNIQUE** provides a data consistency rule

Prevents insertion of a record with a first_name, last_name tuple matching an extant record

Thus prevents addition of data inconsistent with data already entered

2. **"Consistent" != "Right"**

If we enter the Hawaii values first, the Illinois ones are prevented

But, we are forced to solve the problem in some way that doesn't risk mistakes like two duplicating records

Constraints are nice when operations are large

```
1 INSERT INTO
2   sampdb.student
3 SELECT
4   first_name AS name,
5   'M' AS sex,
6   NULL as student_id
7 FROM
8   sampdb.president
9 ;
10
```

1. We're adding 43 records to the data

Getting harder to see what consistency assumptions we might be violating

2. Notice that expressions in SELECT fields are handy

We can use these to handle data requirements of a target table unaddressed by our source table

And, to handle edge cases

```
1 INSERT INTO
2     sampdb.student
3 SELECT
4     first_name AS name,
5     IF(first_name='Hillary', 'F', 'M') AS sex,
6     NULL as student_id
7 FROM
8     sampdb.president
9 ;
10
```

More Query Syntax



UNION combines query results

1. Say we want a list of names used in all our
more_syntax

```
1 SELECT name AS first_name FROM sampdb.student
2 UNION
3 SELECT first_name FROM sampdb.president
4 UNION
5 SELECT first_name FROM sampdb.member
6 ORDER BY first_name ASC
7 LIMIT 5
8 ;
9
10
11
```

```
1 +-----+
2 | first_name |
3 +-----+
4 | Abby      |
5 | Abraham   |
6 | Alan      |
7 | Alma     |
8 | Amanda    |
9 +-----+
10
```

Again, field manipulation bridges data

1. Getting last_name is a problem:

```
1  -- FAILS
2  SELECT name AS first_name FROM sampdb.student
3  UNION
4  SELECT first_name, last_name FROM sampdb.president
5  UNION
6  SELECT first_name, last_name FROM sampdb.member
7  ORDER BY first_name ASC
8  LIMIT 5
9  ;
10
11
12
```

2. Solved by generating a 'new' table on the fly:

```
1  SELECT name AS first_name, 'NA' AS last_name FROM sampdb.student
2  UNION
3  SELECT first_name, last_name FROM sampdb.president
4  UNION
5  SELECT first_name, last_name FROM sampdb.member
6  ORDER BY first_name ASC
7  LIMIT 5
8  ;
9
10
11
```

```
1  +-----+-----+
2  | first_name | last_name |
3  +-----+-----+
4  | Abby      | NA       |
5  | Abraham   | NA       |
6  | Abraham   | Lincoln  |
7  | Alan      | Camosy   |
8  | Alma      | Schauer  |
9  +-----+-----+
10
```

Data Cleaning



Remember our AAPL data?

1						
2						
3						
4	1-Apr-13	442	444	428	429	13918927
5	1-Aug-13	456	457	453	457	7366046
6	1-Feb-13	459	459	448	454	19266727
7	1-Jul-13	403	412	401	409	13970435
8	1-Mar-13	438	438	430	430	19730256
9						
10						

1. That ain't the right order

Date data isn't typed properly

1	DESCRIBE csv_load.aapl;
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Field	Type	Null	Key	Default	Extra
date	varchar(20)	YES		NULL	
open	int(4)	YES		NULL	
high	int(4)	YES		NULL	
low	int(4)	YES		NULL	
close	int(4)	YES		NULL	
volume	int(12)	YES		NULL	

Date data isn't type properly

Server is sorting alphabetically, because that's how it sorts strings

So: Get a date typed table

```
1 DROP TABLE IF EXISTS csv_load.aapl_dated;  
2 CREATE TABLE csv_load.aapl_dated (  
3     -- date          VARCHAR(10),  
4     date             DATE,  
5     open             DECIMAL(5, 2),  
6     high             DECIMAL(5, 2),  
7     low              DECIMAL(5, 2),  
8     close            DECIMAL(5, 2),  
9     volume           INTEGER(12)  
10 );  
11
```

1. Notice that field date is here assigned type DATE

But to load that field, we need YYYY-MM-DD formatted strings

Create those strings in a query

```
1 SELECT
2   CONCAT(
3     CONCAT(
4       '20',
5       RIGHT(date, 2)
6     ),
7     .,
8     MONTHS(
9       STR_TO_DATE(
10        MID(
11          date,
12          LOCATE('-', date) + 1,
13          ( LOCATE('-', date, 4) - LOCATE('-', date) - 1 )
14        ),
15        'ab'
16      )
17    ),
18    .,
19    CONCAT(
20      IF(
21        LENGTH(
22          LEFT(
23            date, LOCATE('-', date) - 1
24          ) = 1, '0', ''
25        ),
26        LEFT(date, LOCATE('-', date) - 1 )
27      )
28    ) AS date,
29    open,
30    high,
31    low,
32    close,
33    volume
34 FROM
35   csv_load.aapl
36 LIMIT 5
37 ;
```

date	open	high	low	close	volume
2013-10-01	478	489	478	488	12638665
2013-9-30	477	482	474	477	9291344
2013-9-26	486	489	484	486	8472169
2013-9-25	489	490	481	482	11319881
2013-9-24	495	495	488	489	13612249

1. Debugging that is where your text editor matters

Load that

```
1 INSERT INTO
2   csv_load.aapl_dated
3 SELECT
4   CONCAT(
5     CONCAT(
6       '20',
7       RIGHT(date, 2)
8     ),
9     '-',
10    MONTH(
11      STR_TO_DATE(
12        MID(
13          date,
14          LOCATE('-', date) + 1,
15          ( LOCATE('-', date, 4) - LOCATE('-', date) - 1 )
16        ),
17        '%b'
18      )
19    ),
20    '-',
21    CONCAT(
22      IF(
23        LENGTH(
24          LEFT(
25            date, LOCATE('-', date) - 1
26          )
27        ) = 1, '0', ''
28      ),
29      LEFT(date, LOCATE('-', date) - 1 )
30    )
31  ) AS date,
32  open,
33  high,
34  low,
35  close,
36  volume
37 FROM
38   csv_load.aapl
39 ;
40
```

Voila

```
1 SELECT * FROM csv_load.aapl_dated ORDER BY date LIMIT 5 ;  
2
```

```
1 +-----+-----+-----+-----+-----+-----+  
2 | date      | open  | high  | low   | close  | volume |  
3 +-----+-----+-----+-----+-----+-----+  
4 | 2012-10-04 | 671.00 | 674.00 | 666.00 | 667.00 | 13241259 |  
5 | 2012-10-05 | 665.00 | 666.00 | 651.00 | 653.00 | 21214444 |  
6 | 2012-10-08 | 647.00 | 648.00 | 636.00 | 638.00 | 22784981 |  
7 | 2012-10-09 | 639.00 | 640.00 | 624.00 | 636.00 | 29949841 |  
8 | 2012-10-10 | 640.00 | 645.00 | 637.00 | 641.00 | 18226990 |  
9 +-----+-----+-----+-----+-----+-----+  
10
```



Advanced MySQL

1. Put that in a user defined function

```
1  -----
2  DROP FUNCTION IF EXISTS csv_load.reformatter;
3
4  -----
5
6  delimiter $$
7
8  CREATE FUNCTION
9  csv_load.reformatter(
10     date_str VARCHAR(255)
11 )
12     RETURNS VARCHAR(100)
13
14 BEGIN
15
16     DECLARE date_var VARCHAR(100);
17
18     SET date_var = CONCAT(
19         CONCAT(
20             '20',
21             RIGHT(date_str, 2)
22         ),
23         '-',
24         MONTH(
25             STR_TO_DATE(
26                 MID(
27                     date_str,
28                     LOCATE('-', date_str) + 1,
29                     ( LOCATE('-', date_str, 4) - LOCATE('-',
30
31                     'bb' )
32                 )
33             ),
34             '%Y-%m-%d'
35         ),
36         '-',
37         CONCAT(
38             IF(
39                 LENGTH(
40                     LEFT(
41                         date_str, LOCATE('-', date_str) - 1
42                     ) = 1, '0', ''
43                 ),
44                 LEFT(date_str, LOCATE('-', date_str) - 1 )
45             )
46         )
47     );
48
49     RETURN date_var;
50
51 END$$
52
53 -----
54
```

Now the call is much simpler

```
1 SELECT
2     csv_load.reformatter(date) AS date,
3     open,
4     high,
5     low,
6     close,
7     volume
8 FROM
9     csv_load.aapl
10 ORDER BY
11     date
12 LIMIT 5
13 ;
14
```

```
1 +-----+-----+-----+-----+-----+-----+
2 | date      | open  | high  | low   | close  | volume |
3 +-----+-----+-----+-----+-----+-----+
4 | 2012-10-04 | 671   | 674   | 666   | 667   | 13241259 |
5 | 2012-10-05 | 665   | 666   | 651   | 653   | 21214444 |
6 | 2012-10-08 | 647   | 648   | 636   | 638   | 22784981 |
7 | 2012-10-09 | 639   | 640   | 624   | 636   | 29949841 |
8 | 2012-10-10 | 640   | 645   | 637   | 641   | 18226990 |
9 +-----+-----+-----+-----+-----+-----+
10
```

This is "Abstraction" or "Encapsulation"

1. **We "hide" complexity in some "object"**

Call on the "object" when we need that complex task

2. **Here, "functions" and "views" are "objects"**

We use them to make references simpler

Data manipulation is irreducibly complex

1. **Tools like Excel abstract a lot of that for us**
2. **But, we are confined to the tools they offer us**
3. **Learning syntax allows us to shape our own tools**
4. **Learning concepts gives us new vision on using those tools**
5. **That's what programming is about, and why it matters**