

Lecture 3: Continuing SQL

Monday, February 2, 2015

Announcements

- Homework #1 has been posted on Canvas and is due by **4pm next Monday**
- FoCS Career Night this Wednesday from 5:00 pm to 7:30 pm
- CNS Spring Career Fair at Frank Erwin Center this Thursday from 1:00 pm to 6:00 pm

Agenda for today

- Continue SQL: Finish Chapter 3 and start on Chapter 4
- Take Quiz #1

Reviewing your questions from last week

- Question #1: What is a **NULL** value?
- Question #2: Why do **column aliases** not work in the WHERE clause?
- Question #3: How does **DISTINCT** work with multiple columns?

NULL Values

- NULL can mean value does not exist or exists but is unknown
- Schema specifies if an attribute is nullable or not nullable
- Some Persons are not included. Why?

```
select *  
from Persons  
where age < 25 or age >= 25
```

Joins

```
Vendors(vendor_id, vendor_name, vendor_address1, ...)  
Invoices(invoice_id, vendor_id, invoice_number,  
         invoice_date, invoice_total, payment_total ...)
```

```
SELECT i.invoice_number, i.invoice_date, i.invoice_total,  
       i.payment_total  
FROM vendors v, invoices i  
WHERE v.vendor_id = i.vendor_id  
AND i.invoice_total >= 500  
AND v.vendor_name = 'Costco'  
ORDER BY v.vendor_name, i.invoice_total DESC
```

Are these two SQL statements equivalent?

```
SELECT vendor_name, invoice_number, invoice_date,  
invoice_total  
FROM vendors JOIN invoices  
ON vendors.vendor_id = invoices.vendor_id  
WHERE invoice_total >= 500  
ORDER BY vendor_name, invoice_total DESC
```

```
SELECT vendor_name, invoice_number, invoice_date,  
invoice_total  
FROM vendors, invoices  
WHERE vendors.vendor_id = invoices.vendor_id  
AND invoice_total >= 500  
ORDER BY vendor_name, invoice_total DESC
```

4-way table join

```
SELECT vendor_name, invoice_number, invoice_date,
       line_item_amt, account_description
FROM   vendors v, invoices i, invoice_line_items li,
       general_ledger_accounts gl
WHERE  v.vendor_id = i.vendor_id
       AND i.invoice_id = li.invoice_id
       AND li.account_number = gl.account_number
       AND (invoice_total - payment_total - credit_total) > 0
ORDER BY vendor_name, line_item_amt DESC
```

| | VENDOR_NAME | INVOICE_NUMBER | INVOICE_DATE | LINE_ITEM_AMT | ACCOUNT_DESCRIPTION |
|---|-------------------------------|----------------|--------------|---------------|--------------------------------|
| 1 | Abbey Office Furnishings | 203339-13 | 02-MAY-14 | 17.5 | Office Supplies |
| 2 | Blue Cross | 547481328 | 20-MAY-14 | 224 | Group Insurance |
| 3 | Blue Cross | 547480102 | 19-MAY-14 | 224 | Group Insurance |
| 4 | Blue Cross | 547479217 | 17-MAY-14 | 116 | Group Insurance |
| 5 | Cardinal Business Media, Inc. | 134116 | 01-JUN-14 | 90.36 | Card Deck Advertising |
| 6 | Coffee Break Service | 109596 | 14-JUN-14 | 41.8 | Meals |
| 7 | Compuserve | 21-4748363 | 09-MAY-14 | 9.95 | Books, Dues, and Subscriptions |
| 8 | Computerworld | 367447 | 31-MAY-14 | 2433 | Card Deck Advertising |

(44 rows selected)

Readability of SELECT statements

```
select invoice_number, invoice_date, invoice_total,  
payment_total, credit_total, invoice_total - payment_total -  
credit_total as balance_due from invoices where invoice_total  
- payment_total - credit_total > 0 order by invoice_date
```

```
SELECT invoice_number, invoice_date, invoice_total,  
       payment_total, credit_total,  
       invoice_total - payment_total - credit_total  
       AS balance_due  
FROM invoices  
WHERE invoice_total - payment_total - credit_total > 0  
ORDER BY invoice_date
```

SELECT statement with a block comment

```
/*  
  Author: Shirley Cohen  
  Date: 01/29/2015  
  The fourth column calculates the balance due  
*/  
SELECT invoice_number, invoice_date, invoice_total,  
       invoice_total - payment_total - credit_total  
       AS balance_due  
FROM invoices
```

A SELECT statement with a single-line comment

```
-- The fourth column calculates the balance due  
SELECT invoice_number, invoice_date, invoice_total,  
       invoice_total - payment_total - credit_total  
       AS balance_due  
FROM invoices
```

Data Manipulation Language (DML) statements

- SELECT
- INSERT
- UPDATE
- DELETE

A statement that adds a row to the Invoices table

```
INSERT INTO invoices
  (invoice_id, vendor_id, invoice_number, invoice_date,
   invoice_total, terms_id, invoice_due_date)
VALUES
  (invoice_id_seq.NEXTVAL, 12, '3289175', '18-JUL-14',
   165, 3, '17-AUG-14')
```

Are these two inserts equivalent?

```
INSERT INTO customers(customer_id, customer_last_name,  
customer_first_name, customer_address, customer_city,  
customer_state, customer_zip) VALUES (26, 'Smith', 'John',  
'1234 Main St', 'Austin', 'TX', '78705')
```

```
INSERT INTO customers VALUES (26, 'Smith', 'John', '1234  
Main St', 'Austin', 'TX', '78705', NULL)
```

Are these two inserts equivalent?

```
INSERT INTO customers(customer_id, customer_last_name,  
customer_first_name, customer_address, customer_city,  
customer_state, customer, zip, customer_phone) VALUES (26,  
'Smith', 'John', '1234 Main St', 'Austin', 'TX', '78705',  
NULL)
```

```
INSERT INTO customers(customer_id, customer_last_name,  
customer_first_name, customer_address, customer_city,  
customer_state, customer, zip, customer_phone) VALUES (26,  
'Smith', 'John', '1234 Main St', 'Austin', 'TX', '78705',  
'')
```

Warning: we get inconsistent behavior across different DBMS systems

A statement that changes one value in one row

```
UPDATE invoices
SET credit_total = 35.89
WHERE invoice_number = '367447'
```

A statement that changes one value in multiple rows

```
UPDATE invoices
SET invoice_due_date = invoice_due_date + 30
WHERE terms_id = 4
```

A statement that deletes a selected invoice

```
DELETE FROM invoices  
WHERE invoice_number = '4-342-8069'
```

A statement that deletes all paid invoices

```
DELETE FROM invoices  
WHERE invoice_total - payment_total - credit_total = 0
```


Types of Database Workloads

- OLTP (online transaction processing)
 - Lots of small updates
 - Access record by key
- OLAP (online analytical processing)
 - Aggregate group-by queries
 - Long-running queries used for data analysis
- Mixed (OLTP and OLAP)

Transactions

- Recovery + Concurrency Control
- ACID =
 - Atomocity (all or nothing)
 - Consistency
 - Isolation (= concurrency control)
 - Durability

Quiz #1

Q1: What is a relation?

Q2: Give an example of a many-to-many relationship

Q3: What are the 4 clauses of a SQL statement that we've seen in class?

Q4: Given the table customers(customer_id, customer_last_name, customer_first_name, customer_address, customer_city, customer_state, customer_zip, customer_zip, customer_phone),

- a) write a select statement that returns all the columns from this table
- b) write the same select in a), but this time also sort by last name
- c) write the same select in b), but this time also only return customers who reside in Austin

Q5: Given that a primary key on table uniquely identifies records on that table, give an example in SQL of a primary key violation?

Next class

- SQL: Chapter 4 in Murach textbook
- Class exercises