CS 327E Lecture 3

Shirley Cohen

February 1, 2016

Agenda

- Announcements
- Homework for today
- Reading Quiz
- Concept Questions
- Homework for next time

Announcements

- Class participation points
- Midterm #1 will take place on 02/17
- Short review on 02/15

Homework for Today

- Chapter 4 from the Learning SQL book
- Exercises at the end of Chapter 4

Quiz Question 1

Which of the following operators **may not** be used to separate conditions in a `WHERE` clause?

A. `ALL`
B. `AND`
C. `OR`
D. All of the above operators may be used.

Quiz Question 2

```
mysql> select * from account;
+------------+----------------+---------------+
| account_id | open_branch_id | avail_balance |
+------------+----------------+---------------+
|          1 |              2 |       1057.75 |
|          2 |           NULL |        500.00 |
|          3 |           NULL |       3000.00 |
|          4 |              0 |       2258.02 |
+------------+----------------+---------------+
```

How many rows does the following query return?

```
SELECT * FROM account
WHERE open_branch_id = NULL;
```

A. 0 B. 2 C. 3 D. 4

Quiz Question 3

Which of the following queries filters rows with a `start_date` between January 1, 2007 and January 1, 2008?

A. `IF start_date > '2007-01-01' AND start_date < '2008-01-01' THEN SELECT * from employee;`
B. `SELECT * FROM employee WHERE start_date > '2007-01-01' AND < '2008-01-01';`
C. `SELECT * FROM employee WHERE start_date BETWEEN '2007-01-01' AND '2008-01-01';`
D. None of the above.

Quiz Question 4

```
mysql> select fname, lname
from employee;
+----------+-----------+
| fname    | lname     |
+----------+-----------+
| Michael  | Smith     |
| Susan    | Barker    |
| Susan    | Hawthorne |
| Sarah    | Parker    |
| Jane     | Grossman  |
| Paula    | Roberts   |
| Thomas   | Ziegler   |
| Samantha | Jameson   |
| Frank    | Portman   |
| Theresa  | Markham   |
| Alex     | Barth     |
+----------+-----------+
```

How many rows are produced from the following query?

```
SELECT fname
FROM employee
WHERE fname like '%a%';
```

A. 0
B. 3
C. 7
D. 10

Concept Question 1

Recall the retail store that keeps information about its products in a table called SKU_Data. How can we look up all the products that are sold by the camping department or climbing department?

A. SELECT * FROM SKU_Data WHERE Department = 'Camping' OR 'Climbing'

B. SELECT * FROM SKU_Data WHERE Department IN ('Camping', 'Climbing')

C. SELECT * FROM SKU_Data WHERE Department = 'Camping' OR Department = 'Climbing'

D. All of the above

E. Only B and C

SKU_Data (SKU, SKU_Description, Department)

SELECT * FROM SKU_Data

SKU	SKU_Description	Department
100100	Std. Scuba Tank, Yellow	Water Sports
100200	Std. Scuba Tank, Magenta	Water Sports
101100	Dive Mask, Small Clear	Water Sports
101200	Dive Mask, Med Clear	Water Sports
201000	Half-dome Tent	Camping
202000	Half-dome Tent Vestibule	Camping
301000	Light Fly Climbing Harness	Climbing
302000	Locking carabiner, Oval	Climbing

Concept Question 2

We have extended the retail store schema to allow tracking the vendors who supply products to the store. We want to obtain a list of the vendors, but we are only interested in those who are in Austin. What SQL query can we use to retrieve all vendors that have a presence in Austin?

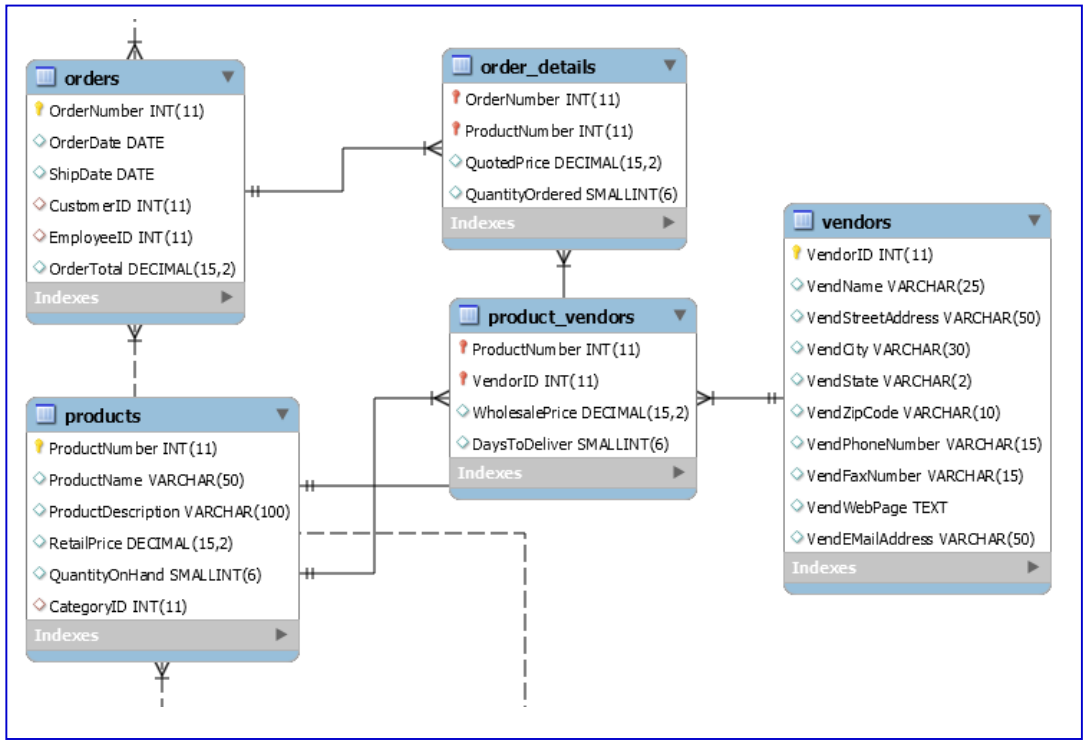
A. select vendName
from vendors
where vendCity = 'AUSTIN'

B. select vendName
from vendors
where vendCity = 'Austin'

C. select vendName
from vendors
where UPPER(vendCity) = 'AUSTIN'

D. Any of the above

E. Not enough information



Concept Question 3

Continuing with the same example database, we now want to see a list of all vendors who are **not** based in Austin. Which SQL query will give us the answer?

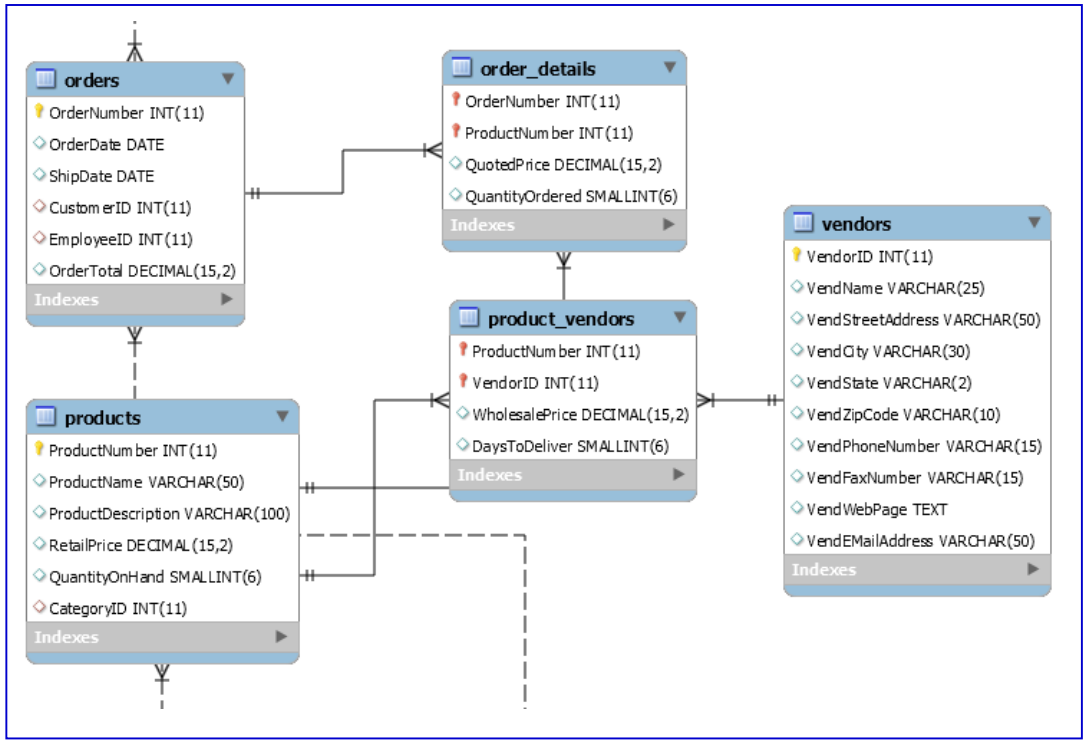
A. select vendName
from vendors
where UPPER(vendCity) != 'AUSTIN'

B. select vendName
from vendors
where UPPER(vendCity) <> 'AUSTIN'

C. select vendName
from vendors
where UPPER(vendCity) <> 'AUSTIN' or vendCity is null

D. Any of the above

E. None of the above



Concept Question 4

Suppose we have a pool of printers and a set of **registered** users who have been given access to a printer. We now want to allow a **guest** user who is not in the table to use one of the **common** printers. How can we come up with a table definition that lets us assign common printers to guest users without losing existing functionality?
Hint: we want the same SQL query that works for registered users to also work for guest users and we want the load balancing logic for common printers to reside in the database.

A. (printer_name, printer_description, **printer_type**, userid)

B. (printer_name, printer_description, **userid_start**, **userid_end**)

C. (printer_name, printer_description, **registered_userid**, **guest_userid**)

D. None of the above

Current table definition:

```
create table PrinterControl
(
  printer_name CHAR(4) PRIMARY KEY,
  printer_description CHAR(4),
  userid CHAR(10)
)
```

select * from PrinterControl

printer_name	printer_description	userid
'LPT1'	'First floor's printer'	'blake'
'LPT2'	'Second floor's printer'	'lee'
'LPT3'	'Third floor's printer'	'smith'
'LPT4'	'Common printer for new user'	NULL
'LPT5'	'Common printer for new user'	NULL

Solution for Concept 4

Previous table definition:

```
create table PrinterControl
(
  printer_name CHAR(4) PRIMARY KEY,
  printer_description CHAR(4),
  userid CHAR(10)
)
```

New table definition:

```
create table PrinterControl
(
  printer_name CHAR(4) PRIMARY KEY,
  printer_description CHAR(4),
  userid_start CHAR(10),
  userid_end CHAR(10)
)
```

Query over new table:

```
SELECT printer_name
FROM PrinterControl
WHERE $userid BETWEEN userid_start
AND userid_end;
```

printer_name	printer_description	userid_start	userid_end
'LPT1'	'First floor's printer'	'blake'	'blake'
'LPT2'	'Second floor's printer'	'lee'	'lee'
'LPT3'	'Third floor's printer'	'smith'	'smith'
'LPT4'	'Common printer for new user'	'a'	'l'
'LPT5'	'Common printer for new user'	'm'	'z'

Concept Question 5

Suppose we have a database that tracks software bugs. What is the relationship between the Bugs entity and the other entities according to the conceptual diagram?

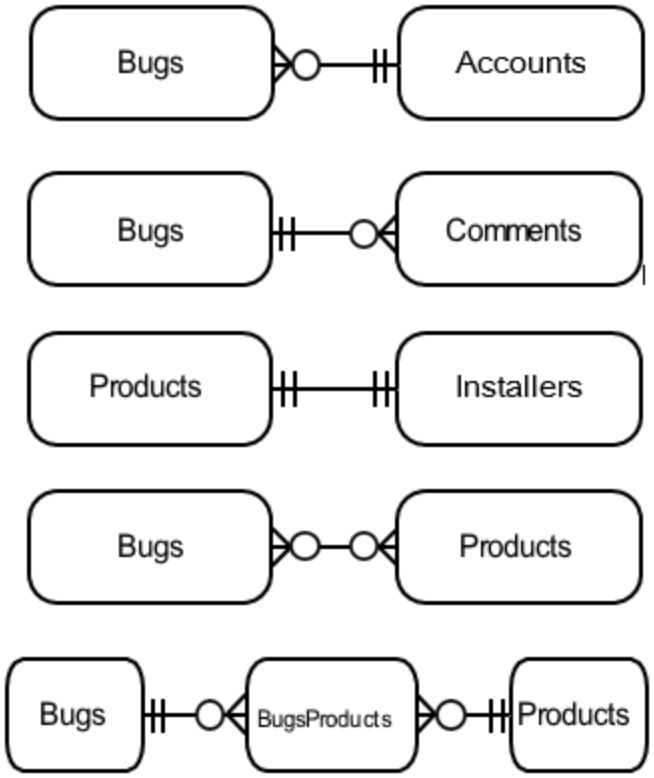
A. Bugs has a many-to-one relationship with Accounts

B. Bugs has a one-to-many relationship with Comments

C. Bugs has a many-to-many relationship with Products

D. Bugs has a one-to-many relationship with BugsProducts

E. All of the above



Concept Question 6

How can we find all the bugs that are both **unassigned** and **active**? Assume that the `assigned_to` field identifies if a bug has been assigned and an **active** bug equals `status` of not 'CLOSED'.

A.
```
select * from Bugs
where assigned_to IS NULL
and (status <> 'CLOSED'
or status IS NULL)
```

B.
```
select * from Bugs
where assigned_to IS NULL
and status <> 'CLOSED'
```

C.
```
select * from Bugs
where assigned_to = NULL
and (status <> 'CLOSED'
or status = NULL)
```

D.
```
select * from Bugs
where assigned_to IS NULL
and status NOT IN
('CLOSED')
```

E. None of the above

Table definitions:

```
CREATE TABLE Accounts (
 account_id INT PRIMARY KEY,
 account_name VARCHAR(20),
 first_name VARCHAR(20),
 last_name VARCHAR(20),
 email VARCHAR(100),
 password_hash CHAR(64),
 ...);

CREATE TABLE Bugs (
 bug_id INT PRIMARY KEY,
 date_reported DATE NOT NULL,
 summary VARCHAR(80),
 reported_by INT NOT NULL,
 assigned_to INT,
 status enum('NEW', 'OPEN', 'QA', 'CLOSED'),
 ...
 FOREIGN KEY (reported_by) REFERENCES
Accounts(account_id),
 FOREIGN KEY (assigned_to) REFERENCES
Accounts(account_id));
```

Homework

- Read chapter 5 from the Learning SQL book
- Exercises at the end of chapter 5