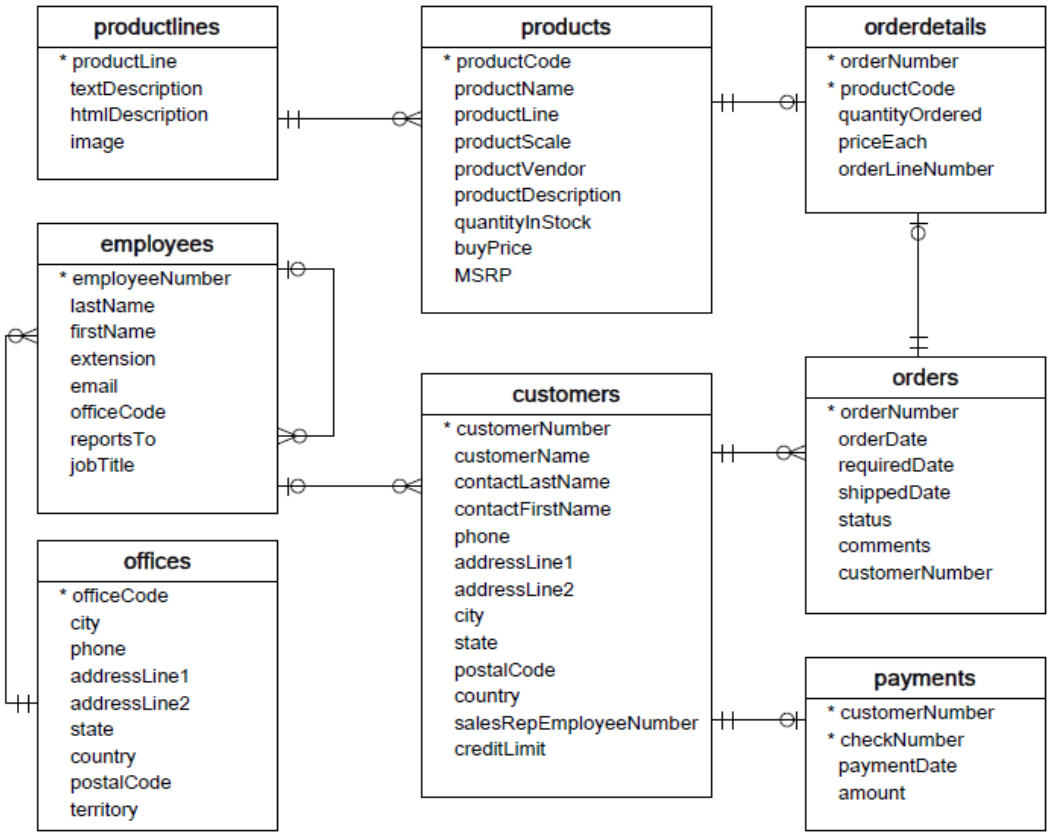
Subqueries

Topics Covered

- What are Subqueries?
- Where are Subqueries Used?
- FROM Subqueries
- WHERE Subqueries
- HAVING Subqueries
- Correlated Subqueries
- Subqueries with
  - INSERT
  - UPDATE
  - DELETE

Example Database



The examples used in this presentation use the sample database provided by MySqlTutorial.org.

What are Subqueries?

Subqueries are SQL SELECT statements nested within another SQL statement.

Subqueries can be used in the FROM clause, the WHERE clause, and/or the HAVING clause. Subqueries make SQL more *dynamic*. For example:

```
select checkNumber, amount from payments
where amount> (select avg(amount) from payments);
```

We might know at a given time that the average salary is 81,000, and could have written “where salary > 81000”. But that query would need to be updated every time a salary changed. The version using the subquery is *dynamic*.

Subqueries in the FROM Clause

We are accustomed to seeing tables in the FROM clause, but FROM can include any item that returns a rowset, including a subquery.

Here is a query that examines the distribution of item counts in all orders:

```
SELECT MAX(items), MIN(items), FLOOR(AVG(items))
    FROM (SELECT orderNumber, COUNT(orderNumber) AS items
    FROM orderdetails
    GROUP BY orderNumber) AS lineitems;
```

As always, the **subquery** is contained within a pair or parentheses. An alias is used to select a specific field out of the subquery.

Subqueries in the WHERE Clause

Probably the most common use of subqueries is in the WHERE clause. Many of these subqueries use the IN operator.

This query returns all employees working in the European offices:

```
select * from employees
where officeCode in
(select officeCode from offices where territory='EMEA');
```

Notice that we could have achieved the same result using a join.

Subqueries in the HAVING Clause

HAVING subqueries can be used to generate rather sophisticated statistics. Here is a query that returns the top 10% of revenue-generating orders:

```
select orderNumber, sum(quantityOrdered * priceEach) NET
from orderdetails
group by orderNumber
having NET >= (
        select 0.9 * max(NET) from (
                select orderNumber, sum(quantityOrdered * priceEach) NET
                from orderdetails
                group by orderNumber) as nets);
```

Notice that this subquery uses a subquery in its FROM clause – a subquery within a subquery.

Correlated Subqueries

Each of the subqueries used in the prior examples can run stand-alone. SQL also allows us to construct subqueries which reference table(s) in the outer query. This query returns products with prices > the average price in their product line:

```
SELECT productname, buyprice
FROM products p1
WHERE buyprice > (
    SELECT AVG(buyprice)
    FROM products
    WHERE productline = p1.productline);
```

In contrast to stand-along subqueries which are executed just one time, correlated subqueries must execute once for every row processed in the outer query. This can lead to poor performance.

Subqueries with DML

All of the above examples use SELECT. Subqueries are also used with INSERT, UPDATE and DELETE statements:

```
INSERT into employeeBackup (select * from employees);

UPDATE orders set status='Backorder'
where status='In Process' and orderNumber in (
        select distinct od.orderNumber
        from orderDetails od join products p using(productCode)
        where p.quantityInStock<10);

DELETE from cart where customerNumber in (
        select distinct customerNumber from (
                select customerNumber, max(orderDate) from orders
                group by customerNumber
                having max(orderDate) < date_sub(now(), interval 3 year)) outdated);
```

Topics Covered

- What are Subqueries?
- Where are Subqueries Used?
- FROM Subqueries
- WHERE Subqueries
- HAVING Subqueries
- Correlated Subqueries
- Subqueries with
  - INSERT
  - UPDATE
  - DELETE