Query Results

- Query results are either a table or a value
  - E.g. `select * from products` or `select count(*) from products`
- Query results can be used in places where a table/value can be used
  * A value can also be considered as a table with only one row and one column

Subquery Example 1

- Find the most expensive products
  ```sql
  select * from products where price =
  ( select max(price) from products );
  ```

Subquery Example 2

- List the ID's of the products sold on 2011/9/1
  ```sql
  select d.product_id from order_details d,
  (select * from orders
   where date_ordered = '2011-09-01') as o
  where d.order_id = o.id;
  ```

Subquery Example 3

- List the ID's of the products sold on 2011/9/1
  ```sql
  select product_id from order_details
  where order_id in
  (select id from orders
   where date_ordered = '2011-09-01');
  ```

About IN

- Checks whether a value is in a set of values
- Only works on single column
- Returns NULL if
  - The value is NULL, or
  - No match found and there’s a NULL in the set
More Subquery Examples

- Find the CPU products that are cheaper than Intel Pentium D
- Find the products that have never been ordered
  - NOT IN

Correlated Subquery

- The inner query uses column(s) from the outer query
  - E.g. find the products that are cheaper than the average price of their category

```
select * from products p where p.price <
  ( select avg(price) from products
    where category = p.category );
```

How Correlated Subqueries Work

```
Outer query
  (1, CPU, Intel Core 2 Duo, 200)
  (2, CPU, Intel Pentium D, 98.99)
  ***
  (6, HD, Maxtor 250G, 60.89)

Inner Query
  WHERE conditions

WHERE conditions

result
```

Correlated Subquery Using EXISTS

- Find the customers who have ordered from our store before

```
select * from customers c where exists
  ( select * from orders
    where customer_id = c.id );
```

About EXISTS

- A unary operator
- Returns true if the subquery returns at least one row
- NOT EXISTS

ANY and ALL

- Find the CPU products that are more expensive than all HD products
- Find the HD products that are more expensive than at least one CPU product

Can we write these queries without using ANY or ALL?
Set Operations

- **Union**
  - \{1,2,3\} \cup \{4,5,6\} = \{1,2,3,4,5,6\}

- **Intersect**
  - \{1,2,3\} \cap \{2,3,4\} = \{2,3\}

- **Difference**
  - \{1,2,3\} - \{2,3,4\} = \{1\}

Set Operations in Database - UNION

<table>
<thead>
<tr>
<th>vendors</th>
<th>customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendor</td>
<td>customer</td>
</tr>
<tr>
<td>Intel</td>
<td>John</td>
</tr>
<tr>
<td>AMD</td>
<td>Jane</td>
</tr>
<tr>
<td>Seagate</td>
<td>Tom</td>
</tr>
<tr>
<td>MAXTOR</td>
<td></td>
</tr>
</tbody>
</table>

- List all the zip codes from both vendors and customers table

About UNION

- Combine result tables of SELECT statements
- The result tables must have the same number of columns
- The corresponding columns must have the same (or at least “compatible”) type
- Duplicates in union results
  - **UNION** – automatically remove duplicates
  - **UNION ALL** – keep duplicates

INTERSECT and DIFFERENCE

- Same syntax as UNION
- MySQL does not support INTERSECT and DIFFERENCE
- So how we implement intersection and difference without INTERSECT and DIFFERENCE??

Summary

- Syntax
  - Subquery (regular and correlated)
  - IN, EXISTS, ANY, ALL
- A different way of thinking (vs. Joins)