MIS2502:
Review for Exam 2

Jing Gong
gong@temple.edu
http://community.mis.temple.edu/gong
Overview

• **Date/Time:** Thursday, March 24, in class (1 hour 20 minutes)
• **Place:** Regular classroom

Please arrive 5 minutes early!

• Multiple-choice and short-answer questions
• Closed-book, closed-note
• No computer
Coverage

Check the **Exam 2 Study Guide**

*Not every item on this list may be on the exam, and there may be items on the exam not on this list.*
Example: Which order has the highest order amount? (Display the order ID and amount, and assume there is no tie)

```
MyDB.Order
<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>10308</td>
<td>2</td>
<td>1996-09-18</td>
<td>100</td>
</tr>
<tr>
<td>10309</td>
<td>1</td>
<td>1996-09-19</td>
<td>10</td>
</tr>
<tr>
<td>10310</td>
<td>1</td>
<td>1996-09-20</td>
<td>63</td>
</tr>
</tbody>
</table>
```

```
SELECT OrderID, Amount
FROM MyDB.Order
ORDER BY Amount DESC
LIMIT 1;
```
SQL Joins

- Used to combine two or more tables, based on the common fields between them.
- Suppose we have....

<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>10308</td>
<td>2</td>
<td>1996-09-18</td>
<td>100</td>
</tr>
<tr>
<td>10309</td>
<td>1</td>
<td>1996-09-19</td>
<td>10</td>
</tr>
<tr>
<td>10310</td>
<td>1</td>
<td>1996-09-20</td>
<td>63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>LastName</th>
<th>FirstName</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Futterkiste</td>
<td>Alfreds</td>
<td>Germany</td>
</tr>
<tr>
<td>2</td>
<td>Trujillo</td>
<td>Ana</td>
<td>US</td>
</tr>
<tr>
<td>3</td>
<td>Moreno</td>
<td>Antonio</td>
<td>Mexico</td>
</tr>
</tbody>
</table>
A Correct, Simple Join

SELECT * FROM MyDB.Order, MyDB.Customer
WHERE Order.CustomerID=Customer.CustomerID;

<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>10308</td>
<td>2</td>
<td>1996-09-18</td>
<td>100</td>
</tr>
<tr>
<td>10309</td>
<td>1</td>
<td>1996-09-19</td>
<td>10</td>
</tr>
<tr>
<td>10310</td>
<td>1</td>
<td>1996-09-20</td>
<td>63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>LastName</th>
<th>FirstName</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Futterkiste</td>
<td>Alfreds</td>
<td>Germany</td>
</tr>
<tr>
<td>2</td>
<td>Trujillo</td>
<td>Ana</td>
<td>US</td>
</tr>
<tr>
<td>3</td>
<td>Moreno</td>
<td>Antonio</td>
<td>Mexico</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>10308</td>
<td>2</td>
<td>1996-09-18</td>
<td>100</td>
</tr>
<tr>
<td>10309</td>
<td>1</td>
<td>1996-09-19</td>
<td>10</td>
</tr>
<tr>
<td>10310</td>
<td>1</td>
<td>1996-09-20</td>
<td>63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>LastName</th>
<th>FirstName</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Trujillo</td>
<td>Ana</td>
<td>US</td>
</tr>
<tr>
<td>1</td>
<td>Futterkiste</td>
<td>Alfreds</td>
<td>Germany</td>
</tr>
<tr>
<td>1</td>
<td>Futterkiste</td>
<td>Alfreds</td>
<td>Germany</td>
</tr>
</tbody>
</table>
What If We Don’t Have the WHERE condition?

```sql
SELECT * FROM MyDB.Order, MyDB.Customer
WHERE Order.CustomerID = Customer.CustomerID;
```

<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>Amount</th>
<th>CustomerID</th>
<th>LastName</th>
<th>FirstName</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>10308</td>
<td>2</td>
<td>1996-09-18</td>
<td>100</td>
<td>1</td>
<td>Futterkiste</td>
<td>Alfreds</td>
<td>Germany</td>
</tr>
<tr>
<td>10309</td>
<td>1</td>
<td>1996-09-19</td>
<td>10</td>
<td>2</td>
<td>Trujillo</td>
<td>Ana</td>
<td>US</td>
</tr>
<tr>
<td>10310</td>
<td>1</td>
<td>1996-09-20</td>
<td>63</td>
<td>3</td>
<td>Moreno</td>
<td>Antonio</td>
<td>Mexico</td>
</tr>
</tbody>
</table>

It will fetch every possible combination (pair) of records from the two tables.

3×3 = 9 rows
More Variations to Join

Step 1: We start with a simple join

```sql
SELECT *
FROM MyDB.Order, MyDB.Customer
WHERE Order.CustomerID=Customer.CustomerID;
```

<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>Amount</th>
<th>CustomerID</th>
<th>LastName</th>
<th>FirstName</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>10308</td>
<td>2</td>
<td>1996-09-18</td>
<td>100</td>
<td>2</td>
<td>Trujillo</td>
<td>Ana</td>
<td>US</td>
</tr>
<tr>
<td>10309</td>
<td>1</td>
<td>1996-09-19</td>
<td>10</td>
<td>1</td>
<td>Futterkiste</td>
<td>Alfr...</td>
<td>Germany</td>
</tr>
<tr>
<td>10310</td>
<td>1</td>
<td>1996-09-20</td>
<td>63</td>
<td>1</td>
<td>Futterkiste</td>
<td>Alfr...</td>
<td>Germany</td>
</tr>
</tbody>
</table>

Step 2: We then end up with -

```sql
SELECT Customer.Country, SUM(Order.Amount)
FROM MyDB.Order, MyDB.Customer
WHERE Order.CustomerID=Customer.CustomerID
GROUP BY Customer.Country;
```

<table>
<thead>
<tr>
<th>Country</th>
<th>SUM(Amount)</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>100</td>
</tr>
<tr>
<td>Germany</td>
<td>73</td>
</tr>
</tbody>
</table>
SQL (Subselects)

- Subselect query can return
  - One single value (one column, one row)
  - A temporary table (one or multiple columns, one or multiple rows)
One single value: Used With Comparison Operators

- SELECT `column_name1` 
  `FROM` `schema_name.table_name1` 
  `WHERE` `column_name2` `comparison_operator` 
  `(SELECT `column_name3` 
   `FROM` `schema_name.table_name2` 
   `WHERE` condition);`

`comparison_operator` could be equality operators such as =, >, <, >=, <=.

Treated as a single value
Subselect as One Single Value: Example 1

Q: What are the order IDs with order amount below average?

- Step 1. We start by writing the subselect query that returns the average order amount:

```
SELECT AVG(Amount) FROM MyDB.Order
```

- Step 2. We treat the subselect query as a single value:

```
SELECT OrderID, Amount FROM MyDB.Order
WHERE Amount <
(SELECT AVG(Amount) FROM MyDB.Order);
```
Q: Which German customer placed the order(s) with the highest order amount?

Recall the simple join query...

```
SELECT * FROM MyDB.Order, MyDB.Customer
WHERE Order.OrderID=Customer.CustomerID;
```
Subselect as One Single Value: Example 2

Q: Which German customer placed the order(s) with the highest order amount?

• Step 1. We start by write the subselect query that returns the highest order amount for German customers:

```sql
SELECT MAX(Order.Amount) 
FROM MyDB.Order, MyDB.Customer 
WHERE Order.CustomerID=Customer.CustomerID 
AND Customer.Country='Germany';
```

• Step 2. We treat the subselect query as a single value:

```sql
SELECT Customer.LastName, Customer.FirstName 
FROM MyDB.Order, MyDB.Customer 
WHERE Order.OrderID=Customer.CustomerID 
AND Customer.Country='Germany' 
AND Amount= (SELECT MAX(Order.Amount) 
FROM MyDB.Order, MyDB.Customer 
WHERE Order.CustomerID=Customer.CustomerID 
AND Customer.Country='Germany');
```

<table>
<thead>
<tr>
<th>LastName</th>
<th>FirstName</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Futterkiste</td>
<td>Alfreds</td>
<td>63</td>
</tr>
</tbody>
</table>
As a temporary table

• SELECT column_name(s)
  FROM
    (SELECT column_name(s)
     FROM schema_name.table_name2
     WHERE condition)
  WHERE condition;
CREATE TABLE schema_name.table_name (columnName1 datatype [NULL][NOT NULL], columnName2 datatype [NULL][NOT NULL], PRIMARY KEY (KeyName));

CREATE TABLE schema_name.table_name (columnName1 datatype [NULL][NOT NULL], columnName2 datatype [NULL][NOT NULL], PRIMARY KEY (KeyName), FOREIGN KEY (ForeignKeyName) REFERENCES reference_table_name(ReferenceKeyName));

When to use NOT NULL?

If there is no foreign key:

If there is foreign key:
DROP TABLE, ALTER TABLE

**DROP TABLE** schema_name.table_name;

**ALTER TABLE** schema_name.table_name

**ADD COLUMN** column_name_name datatype 
[NULL][NOT NULL];

or

**ALTER TABLE** schema_name.table_name

**DROP COLUMN** column_name_name;

or

**ALTER TABLE** schema_name.table_name

**CHANGE COLUMN** old_column_name_name new_column_name_name datatype 
[NULL][NOT NULL];
INSERT INTO, UPDATE, DELETE FROM

• To insert a record into a table:
  
  ```sql
  INSERT INTO schema_name.table_name
  (columnName1, columnName2, columnName3)
  VALUES (value1, value2, value3);
  ```

• To change data in a row:
  
  ```sql
  UPDATE schema_name.table_name
  SET columnName1=value1, columnName2=value2
  WHERE condition;
  ```

• To delete a row from a table:
  
  ```sql
  DELETE FROM schema_name.table_name
  WHERE condition;
  ```

What to use in the WHERE condition?

How about this WHERE condition?
# Data Types

<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Integer</td>
<td>3, -10</td>
</tr>
<tr>
<td>DECIMAL(p,s)</td>
<td>Decimal. Example: decimal(5,2) is a number that has 3 digits before decimal and 2 digits after decimal</td>
<td>3.23, 3.14159</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>String (numbers and letters) with maximum length n</td>
<td>'Hello, I like pizza, MySQL!'</td>
</tr>
<tr>
<td>DATETIME, DATE</td>
<td>Date/Time, or just Date</td>
<td>'2011-09-01 17:35:00', '2011-04-12'</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>Boolean value</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>

Q: What does DECIMAL(4, 1) indicate?  
A: The value cannot go beyond -999.9~999.9
ETL and Assignment 4

| Extract data from the operational data store | Transform data into an analysis-ready format | Load it into the analytical data store |

- **What is it? Why is it important?**
  - Data consistency
  - Data quality

- **Explain the purpose of each component (Extract, Transform, Load)**

- **Excel functions** **VLOOKUP** and **CONCATENATE**
Dimensional Data Modeling

- Data warehouse vs data mart vs data cube
- Data Cube
- Star schema
- Kimball’s four step process for data mart design
  1. Choose the business process
  2. Identify the fact
  3. Decide on the level of granularity
  4. Identify the dimensions
- “non-volatility” of data cubes
Pivot Tables and Assignment 5

Given a question about a set of data, be able to identify the fields required to create a pivot table

- Identify which fields are assigned as VALUES and which ones are assigned as ROWS
- Identify the correct function for aggregation: i.e., SUM, COUNT, AVERAGE
Pivot Tables vs. data cubes

Equivalent to “dimensions” in a data cube

Equivalent to “measured facts” in a data cube
Data Visualization

• Data visualization principles.
  – Tell a story
  – Graphical integrity (lie factor)
  – Minimize graphical complexity (data ink, chartjunk)
Bonus Assignment

Chart #1:

Issues:

• Tell a Story
  – The vertical axis isn’t labeled. We don’t know the unit.

• Graphical Integrity
  – The vertical axis does not start from zero

• Graphical Complexity
  – Horizontal and vertical lines are unnecessary (Chartjunk)
Bonus Assignment

Chart #1:

Issues:

- **Tell a Story**
  - The vertical axis isn’t labeled. We don’t know the unit.

- **Graphical Integrity**
  - The vertical axis does not start from zero

- **Graphical Complexity**
  - Horizontal and vertical lines are unnecessary (Chartjunk)

This also works
Bonus Assignment

Chart #2:

Quantity Sold by Customer Type

Issues:

- Graphical Integrity
  - The 3D chart makes it difficult to compare the sizes

- Graphical Complexity
  - The 3D chart requires more ink (Chartjunk)
Bonus Assignment

Chart #3:

Total Sales by State

Issues:

• Tell a Story
  – Vertical axis isn’t labeled. We don’t know the units
  – Because there are many states to compare, horizontal lines may be helpful

• Graphical Integrity
  – The 3D chart makes it difficult to compare sizes
  – The cone-shaped bars make it even harder to compare sizes

• Graphical Complexity
  – The 3D chart requires more ink (Chartjunk)
  – The number labels are unnecessary
Good Luck!