SELECT Statements

Agenda

- ✓ SQL Server Database Overview
- ✓ Types of Database
- ✓ Working with Queries
- ✓ Aggregates and Group By

SQL Server Database Overview

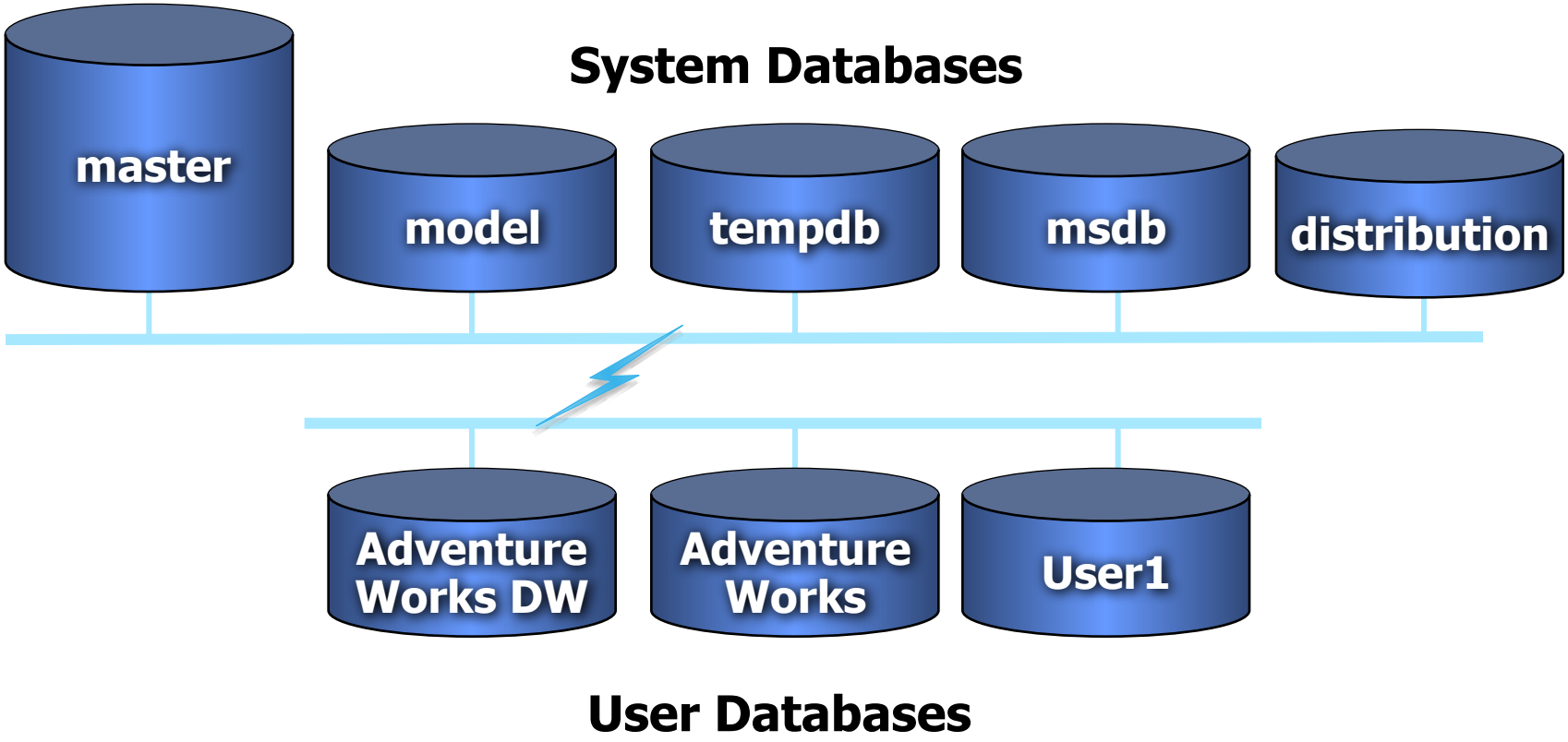
Database

- Database
  - Collection of organized data in a particular order, preferably in rows and columns

- Types of Database
  - There are two main types of database; flat-file and relational
    - The typical flat-file database is split up using a common delimiter
    - The "relation" comes from the fact that the tables can be linked to each other

- There are 5 system databases and 2 user databases are available with SQL server

Types of Databases



Master Database

The master database contains the following crucial information:

- All logins, or roles, that the user IDs belong to
- Every system configuration setting (e.g., data sorting information, security implementation, default language)
- The names of , location and information about the databases within the server
- Specific system tables holding the system information (this list is not exhaustive)
- System error and warning messages

MS DB

- msdb is a system database that contains information used by SQL Server agent, log shipping, SSIS, and the backup and restore system for the relational database engine.
- The database stores all the information about jobs, operators, alerts, and job history.

Model DB

- model is a system database that serves as a template when SQL Server creates a new database.
- As each database is created, SQL Server copies the model database as the new database.
- The only time this does not apply is when you restore or attach a database from a different server.

Temp DB

- It's used to hold temporary objects created by users, temporary objects needed by the database engine, and row-version information.
- The tempdb database is created each time you restart SQL Server.
- The database will be recreated to be its original database size when the SQL Server is stopped.

Resource DB

- This database contains all the read-only critical system tables, metadata, and stored procedures that SQL Server needs to run.
- It does not contain any information about your instance or your databases, because it is only written to during an installation of a new service pack.

Data Organization

- **Logical Structure**
  - The data in a database is organized into the logical components visible to users.
  - Logical components include tables, views, procedures, and users.
- **Physical Structure**
  - Data is physically stored on the disk system with in the data files
  - Clustered indexed actually forces the physical ordering of the data within the data files
- **Meta Data & Catalog**
  - Metadata describes the database's structure, components, users, security, and so on.
  - Catalog includes system catalog and database catalog.

Metadata Catalogs

- System Tables Store Information (Metadata) About the System and Database Objects
- Database Catalog Stores Metadata About a Specific Database
- System Catalog Stores Metadata About the Entire System and All Other Databases
- System Stored Procedures

```
EXEC sp_help Employees
```

- System and Metadata Functions

```
SELECT USER_NAME(10)
```

- Information Schema Views

```
SELECT * FROM INFORMATION_SCHEMA.TABLES`
```

SELECT Queries

Objectives

At the end of this sub-module, you should be able to:

- Identify how to retrieve information form database using select statement
- Recognize how to restrict rows in query result
- Illustrate sorting of query result
- Use of Top Clause

Using the SELECT Statement

- Select List Specifies the Columns
- FROM Clause Specifies the Table
- WHERE Clause Specifies the Condition Restricting the Query

Syntax

```
SELECT [ALL | DISTINCT] <select_list>
 FROM {<table_source>} [,...n]
 WHERE <search_condition>
```

Select Command-Examples

```
SELECT employeeid, lastname, firstname, title
FROM EMPLOYEES                          -- select a few columns
```

```
SELECT employeeid, lastname, firstname, title
FROM EMPLOYEES WHERE employeeid = 5   --  select with condition
```

```
SELECT companyname                      -- select with matching values
FROM CUSTOMERSWHERE companyname LIKE '%Restaurant%‘
```

```
SELECT productid, productname, supplierid, unitprice
FROM  PRODUCTS                          -- select with match & conditions
 WHERE (productname LIKE 'T%' OR productid = 46)
  AND  (unitprice > 16.00)
```

Select - Range of Values

- Range of Values can be selected using BETWEEN, IN, IS or LIKE operator

```
SELECT productname, unitprice
FROM PRODUCTS WHERE unitprice BETWEEN 10 AND 20
```

```
SELECT companyname, country
FROM SUPPLIERS WHERE country IN ('Japan', 'Italy')
```

```
SELECT companyname, fax
FROM SUPPLIERS WHERE fax IS NULL
```

```
SELECT  firstname AS First, lastname AS Last,
employeeid AS 'Employee ID:' FROM EMPLOYEES
```

Column and Table alias

- Alias A column alias is useful sometimes in cutting down the clutter in an SQL statement:

```
SELECT  firstname AS First, lastname AS Last,
employeeid AS 'Employee ID:' FROM EMPLOYEES
```

- The readability of a SELECT statement can be improved by giving a table an alias, also known as a correlation name or range variable.
-
- A table alias can be assigned either with or without the AS keyword:

```
SELECT  e.firstname AS First, e.lastname AS
Last,
employeeid AS 'Employee ID' FROM EMPLOYEES AS e
```

TOP & Order BY

- Lists Only the First n Rows of a Result Set
- Specifies the Range of Values in the ORDER BY Clause
- Returns Ties if WITH TIES Is Used

```
USE Northwind
SELECT TOP 5 orderid, productid, quantity
 FROM [ORDER DETAILS] ORDER BY quantity DESC
```

```
SELECT TOP 5 WITH TIES orderid, productid, quantity
 FROM [ORDER DETAILS]
 ORDER BY quantity DESC
```

Summary

In this sub-module, we have discussed:

- Simple Select Statements
- Column Aliasing
- Table Aliasing
- WHERE Clause
- ORDER BY Clause
- TOP Operator

Aggregates and Group By

Objectives

At the end of this sub-module, you should be able to:

- Recognize how to use Aggregate Functions in query statements
- Applying Group by Clause in Select Statement
- Illustrate Having Clause
- Illustrate Compute and Compute By Clause

Using Aggregate Functions

Function	Description
AVG	Average of values in a numeric expression
COUNT	Number of values in an expression
COUNT (*)	Number of selected rows
MAX	Highest value in the expression
MIN	Lowest value in the expression
SUM	Total values in a numeric expression
COMPUTE	Displays Grand Total at the end
COMPUTE BY	Displays sub Total at each group & Grand Total at end

- Aggregate functions operate on sets of rows to give one result per group

- The group may be formed on the whole table or on a group

Using Aggregate Functions (Contd.).

Syntax:

SELECT [*column*,] *group_function(column), ...*
FROM *table*
[WHERE *condition*]
[GROUP BY *column*]
[ORDER BY *column*]

```
SELECT AVG(Price), MAX(Price), MIN(Price), SUM(Price)
FROM   Titles
WHERE  Type In('Mod_Cook', 'Business')

SELECT MIN(pubdate), MAX(pubdate)
FROM Titles

SELECT COUNT(advance)
FROM   Titles
WHERE  type='Business'
```

GROUP BY Clause

- Each group summarizes the data for all the rows in the table that have the same value
- When you group data, you can display only summary or grouped data
- You cannot display values from individual rows
- You can group by more than one column, each group in the query shows the aggregate values for all grouping columns

GROUP BY Clause (Contd.).

- Group By is added to SQL because aggregate functions (like SUM) return the aggregate of all column values every time they are called, and without the GROUP BY function it was impossible to find the sum for each individual group of column values

- You can use the GROUP BY clause to divide the rows in a table into groups

```
USE Northwind
SELECT productid,SUM(quantity) AS total_quantity
 FROM ORDERHIST
 WHERE productid = 2
 GROUP BY productid
```

GROUP BY Clause (Contd.).

Examples:

```
USE pubs

SELECT pub_id ,SUM(price) as Total
FROM titles
GROUP BY pub_id

SELECT pub_id, type, SUM(price) Total_price
FROM titles
GROUP BY pub_id, type

SELECT type, AVG(price)
FROM titles
WHERE advance > $5000
GROUP BY type

SELECT royalty, AVG(price * 2) AS AveragePrice
FROM pubs.dbo.titles
WHERE royalty IS NOT NULL
GROUP BY royalty
```

HAVING Clause

- The HAVING clause sets conditions on the GROUP BY clause similar to the way WHERE interacts with SELECT
- You can limit the groups that appear in a query by specifying a condition that applies to groups as a whole

```
SELECT column, group_function
FROM table
[WHERE condition]
[GROUP BY   group_by_expression]
[HAVING group_condition]
[ORDER BY column]
```

```
USE Northwind
SELECT productid, SUM(quantity)
   AS total_quantity
 FROM ORDERHIST
 GROUP BY productid
 HAVING SUM(quantity)>=30
```

HAVING Clause (Contd.).

```
USE Northwind
SELECT productid, SUM(quantity) AS total_quantity
 FROM orderhist
 GROUP BY productid
 HAVING SUM(quantity)>=30

SELECT pub_id, AVG(price)
FROM titles
GROUP BY pub_id
HAVING (AVG(price) > 10)

USE pubs
SELECT pub_id, total = SUM(ytd_sales)
FROM titles GROUP BY pub_id
HAVING SUM(ytd_sales) > 40000
```

Summary

In this sub-module, we have learnt:

- Aggregate Functions Native to SQL Server
- Using Aggregate Functions with NULL Values
- CLR Integration, Assemblies
- Implementing Custom Aggregate Functions
- Using the GROUP BY clause
- Filtering Grouped Data by Using the HAVING Clause
- Building a Query for Summarizing Grouped Data – GROUP BY
- Examining How the ROLLUP and CUBE Operators Work
- Using the ROLLUP and CUBE Operators
- Using the COMPUTE and COMPUTE BY Clauses
- Building a Query for Summarizing Grouped Data – COMPUTE
- Using GROUPING SETS