Telerik Academy

Advanced SQL

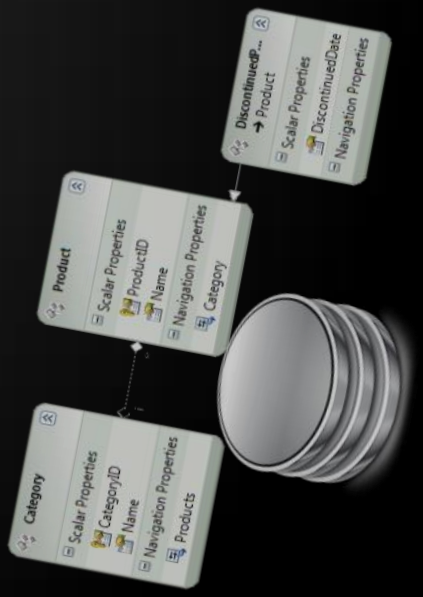
Aggregations, Grouping, SQL Functions, DDL

Databases

Telerik Software Academy
http://academy.telerik.com

Telerik Academy

Table of Contents



Telerik Academy

SQL Language

Nested SELECT Statements

Telerik Academy

Nested SELECT Statements

- SELECT statements can be nested in the where clause

```
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary =
  (SELECT MAX(Salary) FROM Employees)
```

```
SELECT FirstName, LastName, DepartmentID, Salary
FROM Employees
WHERE DepartmentID IN
  (SELECT DepartmentID FROM Departments
   WHERE Name='Sales')
```

  - Note: always prefer joins to nested SELECT statements for better performance

Telerik Academy

Nested SELECT Statements with Table Aliases

- Tables from the main SELECT can be referred in the nested SELECT by aliases
- Example:
  - Find the maximal salary for each department and the name of the employee that gets it

```
SELECT FirstName, LastName, DepartmentID, Salary
FROM Employees e
WHERE Salary =
  (SELECT MAX(Salary) FROM Employees
   WHERE DepartmentID = e.DepartmentID)
ORDER BY DepartmentID
```

Telerik Academy

Using the EXISTS Operator

- Using the EXISTS operator in SELECT statements
  - Find all employees with managers from the first department

```
SELECT FirstName, LastName, EmployeeID, ManagerID
FROM Employees e
WHERE EXISTS
  (SELECT EmployeeID
   FROM Employees m
   WHERE m.EmployeeID = e.ManagerID
     AND m.DepartmentID = 1)
```

Telerik Academy

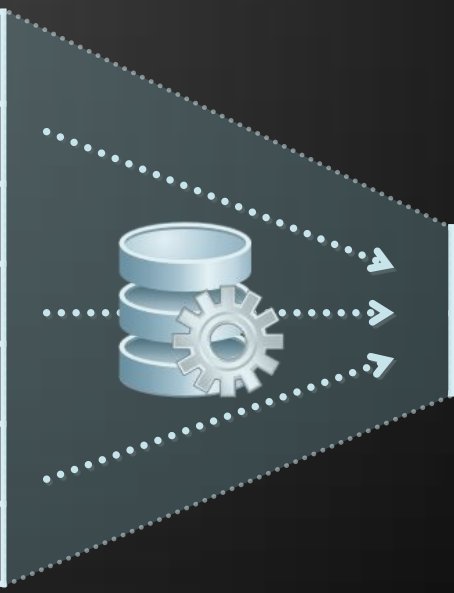
SQL Language

Aggregating Data with Group Functions

Group Functions

- Group functions operate over sets of rows to give one single result (per group)

EmployeeID	Salary
1	12500,00
2	13500,00
3	43300,00
4	29800,00
5	25000,00
…	…



MAX(Salary)
125500,00

Group Functions in SQL

- `COUNT(*)` – count of the selected rows
- `SUM(column)` – sum of the values in given column from the selected rows
- `AVG(column)` – average of the values in given column
- `MAX(column)` – the maximal value in given column
- `MIN(column)` – the minimal value in given column

Telerik Academy

AVG() and SUM() Functions

- You can use AVG() and SUM() only for numeric data types

```
SELECT
  AVG(Salary) [Average Salary],
  MAX(Salary) [Max Salary],
  MIN(Salary) [Min Salary],
  SUM(Salary) [Salary Sum]
FROM Employees
WHERE JobTitle = 'Design Engineer'
```

Average Salary	Max Salary	Min Salary	Salary Sum
32700.00	32700.00	32700.00	98100.00

Telerik Academy

MIN() and MAX() Functions

- You can use `MIN()` and `MAX()` for almost any data type (`int`, `datetime`, `varchar`, ...)

```
SELECT MIN(HireDate) MinHD, MAX(HireDate) MaxHD
FROM Employees
```

MinHD	MaxHD
1996-07-31	2003-06-03

- Displaying the first and last employee's name in alphabetical order:

```
SELECT MIN(LastName), MAX(LastName)
FROM Employees
```

Telerik Academy

The COUNT(…) Function

- COUNT(*) returns the number of rows in the result record set

```
SELECT COUNT(*) Cnt FROM Employees
WHERE DepartmentID = 3
```

Cnt
18

- COUNT(expr) returns the number of rows with non-null values for the expr

```
SELECT COUNT(ManagerID) MgrCount,
  COUNT(*) AllCount
FROM Employees
WHERE DepartmentID = 16
```

MgrCount	AllCount
1	2

Telerik Academy

Group Functions and NULLs

- Group functions ignore NULL values in the target column

```
SELECT AVG(ManagerID) Avg,
  SUM(ManagerID) / COUNT(*) AvgAll
FROM Employees
```

Avg	AvgAll
108	106

- If each NULL value in the ManagerID column were considered as 0 in the calculation, the result would be 106

Telerik Academy

Group Functions in Nested Queries

- Find the earliest hired employee for each department

```
SELECT e.FirstName, e.LastName, e.HireDate, d.Name
FROM Employees e
  JOIN Departments d
    ON e.DepartmentID = d.DepartmentID
WHERE e.HireDate =
  (SELECT MIN(HireDate) FROM Employees
   WHERE DepartmentID = d.DepartmentID)
```

FirstName	LastName	HireDate	Name
Guy	Gilbert	1998-07-31 00:00:00	Production
Kevin	Brown	1999-02-26 00:00:00	Marketing
Roberto	Tamburello	1999-12-12 00:00:00	Engineering

Telerik Academy

SQL Language

Group Functions and the GROUP BY Statement

Creating Groups of Data

Employees

DepartmentID	Salary
12	10300
12	16800
12	16800
12	10300
12	17800
2	28800
2	25000
2	29800
2	25000
16	125500
16	60100
...	...

72000

108600

185600

Depart-mentID	SUM (Salary)
12	72000
2	108600
16	185600
...	...

Telerik Academy

The GROUP BY Statement

- We can divide rows in a table into smaller groups by using the GROUP BY clause
- The SELECT + GROUP BY syntax:

```
SELECT <columns>, <group_function(column)>
FROM   <table>
[WHERE <condition>]
[GROUP BY <group_by_expression> ]
[HAVING   <filtering_expression>]
[ORDER BY <columns>
```

- The <group_by_expression> is a list of columns

Telerik Academy

The GROUP BY Statement (2)

- Example of grouping data:

```
SELECT DepartmentID, SUM(Salary) as SalariesCost
FROM Employees
GROUP BY DepartmentID
```

DepartmentID	SalariesCost
12	72000
2	108600
16	185600
…	…

- The GROUP BY column is not necessary needed to be in the SELECT list

Telerik Academy

Grouping by Several Columns

Depart-mentID	JobTitle	Salary
11	Network Manager	39700
11	Network Administrator	32500
11	Network Administrator	32500
11	Database Administrator	38500
11	Database Administrator	38500
10	Accountant	26400
10	Accountant	26400
10	Finance Manager	43300
…	…	…

39700

65000

77000

52800

43300

Depart mentID	JobTitle	Salary
11	Network Manager	39700
11	Network Administrator	65000
11	Database Administrator	77000
10	Accountant	52800
10	Finance Manager	43300
…	…	…

Grouping by Several Columns – Example

- Example of grouping data by several columns:

```
SELECT DepartmentID, JobTitle,
  SUM(Salary) as Salaries, COUNT(*) as Count
FROM Employees
GROUP BY DepartmentID, JobTitle
```

DepartmentID	JobTitle	Salaries	Count
2	Senior Tool Designer	58600	2
2	Tool Designer	50000	2
7	Production Supervisor	525000	21
7	Production Technician	1926000	157
…	…	…	…

Telerik Academy

Illegal Use of Group Functions

- This SELECT statement is illegal:

```
SELECT DepartmentID, COUNT(LastName)
FROM Employees
```

  - Can not combine columns with groups functions unless when using GROUP BY

- This SELECT statement is also illegal

```
SELECT DepartmentID, AVG(Salary)
FROM Employees
WHERE AVG(Salary) > 30
GROUP BY DepartmentID
```

  - Can not use WHERE for group functions

Restrictions for Grouping

- When using groups we can select only columns listed in the GROUP BY and grouping functions over the other columns

```
SELECT DepartmentID, JobTitle,
  SUM(Salary) AS Cost, MIN(HireDate) as StartDate
FROM Employees
GROUP BY DepartmentID, JobTitle
```

  - Can not select columns not listed in the GROUP BY clause
  - It is allowed to apply group functions over the columns in the GROUP BY clause, but has no sense

Telerik Academy

Using GROUP BY with HAVING Clause

- HAVING works like WHERE but is used for the grouping functions

```
SELECT DepartmentID, COUNT(EmployeeID) as
  Count, AVG(Salary) AverageSalary
FROM Employees
GROUP BY DepartmentID
HAVING COUNT(EmployeeID) BETWEEN 3 AND 5
```

DepartmentID	Count	AverageSalary
2	4	27150
12	5	14400
…	…	…

Telerik Academy

Using Grouping Functions and Table Joins

- Grouping function can be applied on columns from joined tables

```
SELECT COUNT(*) AS EmpCount, d.Name AS DeptName
FROM Employees e JOIN Departments d
     ON e.DepartmentID = d.DepartmentID
WHERE e.HireDate BETWEEN '1999-2-1' AND '2002-12-31'
GROUP BY d.Name
HAVING COUNT(*) > 5
ORDER BY EmpCount DESC
```

EmpCount	DeptName
95	Production
8	Finance
8	Information Services

Telerik Academy

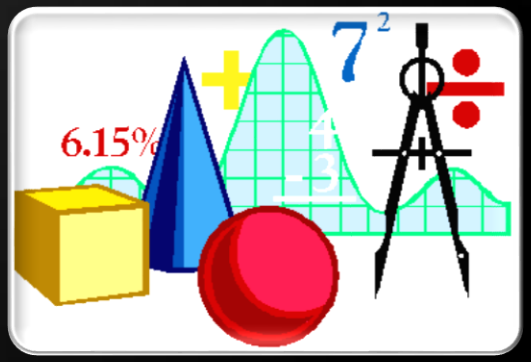
SQL Language

SQL Server Functions

Telerik Academy

Standard Functions in Microsoft SQL Server

- Single-row functions
  - String functions
  - Mathematical functions
  - Date functions
  - Conversion functions
- Multiple-row functions
  - Aggregate functions



COALESCE() Functions

- ISNULL(<value>,<default_value>) – converts NULL values to given default value
- COALESCE(<value>,<value>,<value>) – returns the first that is not NULL

```
SELECT Name AS [Projects Name],
  ISNULL(EndDate, GETDATE()) AS [End Date]
FROM Projects
```

Projects Name	End Date
Classic Vest	2006-07-02 08:19:43.983
Cycling Cap	2003-06-01 00:00:00.000
Full-Finger Gloves	2003-06-01 00:00:00.000
Half-Finger Gloves	2003-06-01 00:00:00.000
…	…

String Functions

- Changing the casing – LOWER, UPPER
- Manipulating characters – SUBSTRING, LEN, LEFT, RIGHT, TRIM, REPLACE

```
SELECT LastName, LEN(LastName) AS LastNameLen,
  UPPER(LastName) AS UpperLastName
FROM Employees
WHERE RIGHT(LastName, 3) = 'son'
```

LastName	LastNameLen	UpperLastName
Erickson	8	ERICKSON
Johnson	7	JOHNSON
Munson	6	MUNSON
…	…	…

Other Functions

- Mathematical Functions – ROUND, FLOOR, POWER, ABS, SQRT, …

```
SELECT FLOOR(3.14) → 3
SELECT ROUND(5.86, 0) → 6.00
```

- Date Functions – GETDATE, DATEADD, DAY, MONTH, YEAR, …
- Conversion Functions – CONVERT, CAST

```
SELECT CONVERT(DATETIME, '20051231', 112)
→ 2005-12-31 00:00:00.000
-- 112 is the ISO formatting style YYYYMMDD
```

Combining Functions

- We can combine functions to achieve more complex behavior

```
SELECT Name AS [Projects Name],
  COALESCE(CONVERT(nvarchar(50), EndDate),
  'Not Finished') AS [Date Finished]
FROM Projects
```

Projects Name	Date Finished
HL Mountain Front Wheel	Jun 1 2003 12:00AM
LL Touring Handlebars	Not Finished
HL Touring Handlebars	Not Finished
LL Road Front Wheel	Jun 1 2003 12:00AM
…	…

Telerik Academy

SQL Language

Data Definition Language (DDL)

Telerik Academy

Data Definition Language

- DDL commands for defining / editing objects
  - CREATE
  - ALTER
  - DROP

- Data Control Language (DCL) for managing access permissions
  - GRANT
  - REVOKE
  - DENY

Telerik Academy

Creating Database Objects

- CREATE command
  - CREATE TABLE <name> (<field_definitions>)
  - CREATE VIEW <name> AS <select>
  - CREATE <object> <definition>

```
CREATE TABLE Persons (
  PersonID int IDENTITY,
  Name nvarchar(100) NOT NULL,
  CONSTRAINT PK_Persons PRIMARY KEY(PersonID)
)
GO

CREATE VIEW [First 10 Persons] AS
SELECT TOP 10 Name FROM Persons
```

Telerik Academy

Creating Objects – More Examples

```
CREATE TABLE Countries (
  CountryID int IDENTITY,
  Name nvarchar(100) NOT NULL,
  CONSTRAINT PK_Countries PRIMARY KEY(CountryID)
)

GO

CREATE TABLE Cities (
  CityID int IDENTITY,
  Name nvarchar(100) NOT NULL,
  CountryID int NOT NULL,
  CONSTRAINT PK_Cities PRIMARY KEY(CityID)
)
```

Telerik Academy

Modifying Database Objects

- ALTER command
  - ALTER TABLE <name> <command>
  - ALTER <object> <command>

```
-- Add a foreign key constraint Cities --> Country
ALTER TABLE Cities
ADD CONSTRAINT FK_Cities_Countries
  FOREIGN KEY (CountryID)
  REFERENCES Countries(CountryID)

-- Add column Population to the table Country
ALTER TABLE Countries ADD Population int

-- Remove column Population from the table Country
ALTER TABLE Countries DROP COLUMN Population
```

Deleting Database Objects

- DROP command
  - DROP TABLE <name>
  - DROP TRIGGER <name>
  - DROP INDEX <name>
  - DROP <object>

```
DROP TABLE Persons

ALTER TABLE Cities
DROP CONSTRAINT FK_Cities_Countries
```

Telerik Academy

Managing Access Permissions

- GRANT command

```
GRANT <persmission> ON <object> TO <role>
```

  - Example:

```
GRANT SELECT ON Persons TO public
```

- REVOKE command

```
REVOKE <persmission> ON <object> FROM <role>
```

  - Example:

```
REVOKE SELECT ON Employees FROM public
```

Telerik Academy

Creating Tables in SQL Server

Best Practices

Telerik Academy

Creating Tables in SQL Server

- Creating new table:
  - Define the table name
    - Should have good name
  - Define the columns and their types
    - Use proper data type
  - Define the table primary key
    - Use `IDENTITY` for enabling auto increment of the primary key
  - Define foreign/keys and constraints

Creating Tables in SQL Server – Examples

```
CREATE TABLE Groups (
  GroupID int IDENTITY,
  Name nvarchar(100) NOT NULL,
  CONSTRAINT PK_Groups PRIMARY KEY(GroupID)
)

CREATE TABLE Users (
  UserID int IDENTITY,
  UserName nvarchar(100) NOT NULL,
  GroupID int NOT NULL,
  CONSTRAINT PK_Users PRIMARY KEY(UserID),
  CONSTRAINT FK_Users_Groups FOREIGN KEY(GroupID)
    REFERENCES Groups(GroupID)
)
```

Telerik Academy

Transactions

Begin / Commit / Rollback Transactions in SQL Server

Telerik Academy

What Is Concurrency Control?

- Pessimistic concurrency (default in SQL Server)
  - Locks table data at each data modification
  - Concurrent users are blocked until the lock is released
- Optimistic concurrency (default in MySQL)
  - No locks are performed when data is being read or changed
  - Concurrent users don't see the changes until they are committed / rolled-back
  - Supported with SNAPSHOT isolation in SQL Server

Transactions

- Transactions start by executing `BEGIN TRANSACTION` (or just `BEGIN TRAN`)
- Use `COMMIT` to confirm changes and finish the transaction
- Use `ROLLBACK` to cancel changes and abort the transaction
- Example:

```
BEGIN TRAN
DELETE FROM EmployeesProjects;
DELETE FROM Projects;
ROLLBACK TRAN
```

Telerik Academy

The Implicit Transactions Option

- What is implicit transactions mode?
  - Automatically start a new transaction after each commit or rollback
  - Nested transactions are not allowed
  - Transaction must be explicitly completed with `COMMIT` or `ROLLBACK TRANSACTION`
- By default, `IMPLICIT_TRANSACITONS` setting is switched off

```
SET IMPLICIT_TRANSACTIONS ON
```

Telerik Academy

Advanced SQL

Exercises

1. Write a SQL query to find the names and salaries of the employees that take the minimal salary in the company. Use a nested SELECT statement.
2. Write a SQL query to find the names and salaries of the employees that have a salary that is up to 10% higher than the minimal salary for the company.
3. Write a SQL query to find the full name, salary and department of the employees that take the minimal salary in their department. Use a nested SELECT statement.

Exercises (2)

4. Write a SQL query to find the average salary in the department #1.
5. Write a SQL query to find the average salary in the "Sales" department.
6. Write a SQL query to find the number of employees in the "Sales" department.
7. Write a SQL query to find the number of all employees that have manager.
8. Write a SQL query to find the number of all employees that have no manager.
9. Write a SQL query to find all departments and the average salary for each of them.

Exercises (3)

10. Write a SQL query to find the count of all employees in each department and for each town.
11. Write a SQL query to find all managers that have exactly 5 employees. Display their first name and last name.
12. Write a SQL query to find all employees along with their managers. For employees that do not have manager display the value "(no manager)".
13. Write a SQL query to find the names of all employees whose last name is exactly 5 characters long. Use the built-in `LEN(str)` function.

Exercises (4)

14. Write a SQL query to display the current date and time in the following format "day.month.year hour:minutes:seconds:milliseconds". Search in Google to find how to format dates in SQL Server.
15. Write a SQL statement to create a table `Users`. Users should have username, password, full name and last login time. Choose appropriate data types for the table fields. Define a primary key column with a primary key constraint. Define the primary key column as identity to facilitate inserting records. Define unique constraint to avoid repeating usernames. Define a check constraint to ensure the password is at least 5 characters long.

Exercises (5)

16. Write a SQL statement to create a view that displays the users from the `Users` table that have been in the system today. Test if the view works correctly.
17. Write a SQL statement to create a table `Groups`. Groups should have unique name (use unique constraint). Define primary key and identity column.
18. Write a SQL statement to add a column `GroupID` to the table `Users`. Fill some data in this new column and as well in the `Groups` table. Write a SQL statement to add a foreign key constraint between tables `Users` and `Groups` tables.

Exercises (6)

19. Write SQL statements to insert several records in the `Users` and `Groups` tables.
20. Write SQL statements to update some of the records in the `Users` and `Groups` tables.
21. Write SQL statements to delete some of the records from the `Users` and `Groups` tables.
22. Write SQL statements to insert in the `Users` table the names of all employees from the `Employees` table. Combine the first and last names as a full name. For username use the first letter of the first name + the last name (in lowercase). Use the same for the password, and `NULL` for last login time.

Exercises (7)

23. Write a SQL statement that changes the password to `NULL` for all users that have not been in the system since `10.03.2010`.
24. Write a SQL statement that deletes all users without passwords (`NULL` password).
25. Write a SQL query to display the average employee salary by department and job title.
26. Write a SQL query to display the minimal employee salary by department and job title along with the name of some of the employees that take it.
27. Write a SQL query to display the town where maximal number of employees work.

Exercises (8)

28. Write a SQL query to display the number of managers from each town.
29. Write a SQL to create table `WorkHours` to store work reports for each employee (employee id, date, task, hours, comments). Don't forget to define identity, primary key and appropriate foreign key.

    Issue few SQL statements to insert, update and delete of some data in the table.

    Define a table `WorkHoursLogs` to track all changes in the `WorkHours` table with triggers. For each change keep the old record data, the new record data and the command (insert / update / delete).

Exercises (9)

30. Start a database transaction, delete all employees from the 'Sales' department along with all dependent records from the pother tables. At the end rollback the transaction.
31. Start a database transaction and drop the table `EmployeesProjects`. Now how you could restore back the lost table data?
32. Find how to use temporary tables in SQL Server. Using temporary tables backup all records from `EmployeesProjects` and restore them back after dropping and re-creating the table.

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com