Selektion und Projektion

Was ist das?

- Selektion Filtern nach Zeilen
- Projektion Filtern nach Spalten
- Das SELECT-Kommando selektiert und projiziert

Klausel	Ausdruck	Nutzen
SELECT	<Spaltenliste>	Legt Spaltenliste für Ergebnismenge fest
FROM	<Tabellenname>	Sagt aus, welche Tabellen betrachtet werden
WHERE	<Filterausdruck>	Filtert unerwünschte Daten heraus
GROUP BY	<Gruppierungskriterium>	Gemeinsame Spaltenwerte zusammenfassen
HAVING	<Filterausdruck>	Filtert unerwünschte Gruppen hinaus
ORDER BY	<Spaltenliste>	Sortiert nach Spalte(n), aufsteigend / absteigend

Reihenfolge der Abfrage

- Die Auswertungsreihenfolge von SQL Server ist nicht die Reihenfolge, in der die Abfrage geschrieben wird

Auswertungsreihenfolge	Klausel (Syntaxreihenfolge)	Ausdruck
5	SELECT	<Spaltenliste>
1	FROM	<Tabellenname>
2	WHERE	<Filterausdruck>
3	GROUP BY	<Gruppierungskriterium>
4	HAVING	<Filterausdruck>
6	ORDER BY	<Sortierreihenfolge>

Wesentliche Operationen

- Projektion: „blendet Spalten aus“
- Selektion: „blendet Zeilen aus“
- Kreuzprodukt: „Jede Zeile von A mit jeder Zeile von B“
- Umbenennung: Spalten bekommen andere Attributnamen
- Vereinigung: Relationen mit gleichem Schema zusammenfügen
- Differenz: „Aus A alles entfernen, was in B ist“

Erste SELECT-Anweisungen

Beispiel 1

```
SELECT *
FROM pubs.dbo.jobs;
```

* für alle Spalten

Spaltenliste

Beispiel 2

```
SELECT job_id, job_desc
FROM pubs.dbo.jobs;
```

Semikolon um Anweisung abzuschliessen

Schema und Tabellenname

Erstellen berechneter Spalten

```
SELECT UnitPrice, UnitPrice + (UnitPrice * 0.19)
FROM Northwind.dbo.Products;
```

Operatoren sind +, -, *, /, %

Verknüpfte Texte

```
SELECT CompanyName + ‘ from ‘ + City
FROM Northwind.dbo.Customers;
```

Verwenden von Aliasnamen I

- Aliase (latein ‚sonst‘) sind (Alternativ-)Bezeichnungen für Tabellen oder Spalten
- Beispiele für Spaltenaliase

```
SELECT UnitPrice, UnitPrice + (UnitPrice * 0.19) AS PriceWithTax
FROM Northwind.dbo.Products;
```

```
SELECT UnitPrice, ProductName Produkt
FROM Northwind.dbo.Products;
```

Verwenden von Aliasnamen II

- Beispiele für Tabellenaliase

```
SELECT UnitPrice, ProductName
FROM Northwind.dbo.Products AS NP;
```

```
SELECT UnitPrice, ProductName
FROM Northwind.dbo.Products NP;
```

```
SELECT NP.UnitPrice, NP.ProductName
FROM Northwind.dbo.Products AS NP;
```

Wesentliche Operationen

- Projektion: „blendet Spalten aus“
- Selektion: „blendet Zeilen aus“
- Kreuzprodukt: „Jede Zeile von A mit jeder Zeile von B“
- Umbenennung: Spalten bekommen andere Attributnamen
- Vereinigung: Relationen mit gleichem Schema zusammenfügen
- Differenz: „Aus A alles entfernen, was in B ist“

Erinnerung WHERE I

Art des Operators	Operator	Bedeutung
Unär	+	Gibt den Wert des folgenden Ausdrucks als positiven Wert zurück
	-	Gibt den Wert des folgenden Ausdrucks als negativen Wert zurück
	~	Gibt den Wert der folgenden Ganzzahl bitweise umgekehrt zurück
Arithmetische Operatoren	+, -, *, /	Addition, Subtraktion, Multiplikation und Division
	%	Modulo, ganzzahliger Rest einer Division
Verknüpfung von Zeichenketten	+	5 + 9 erzeugt 14, 5 + ‘9‘ erzeugt ebenfalls 14, ‘5‘ + ‘9‘ erzeugt 59, ‘5.2‘ + 9 scheitert, ‘5.2‘ + 9.0 erzeugt 14.2, ‘5,2‘ + 9 scheitert
Bitweise Operatoren	&, |, ^	Bitweise AND, OR, XOR
Vergleichsoperatoren	=, <, >, <=, >=, <>, !=, !<, !>	Gleich, kleiner, größer, gleiner oder gleich, größer oder gleich, ungleich, ungleich, nicht kleiner, nicht größer

Erinnerung WHERE II

Art des Operators	Operator	Bedeutung
Erweiterte Vergleichsoperatoren	BETWEEN	5 BETWEEN 3 AND 7 ergibt TRUE
	IN	5 IN (3, 5, 7) ergibt TRUE
	LIKE	Musterüberprüfung der Ausdrücke, kann Platzhalter (%, _) und Gruppenzeichen ([], [^]) enthalten
	EXISTS	TRUE bei wenigstens einem Vorkommen; manchmal schneller als COUNT
	ALL, ANY, SOME	<erster Ausdruck> <Standardoperator> ALL|ANY|SOME (<Unterabfrage>) Liefert TRUE oder FALSE
Logische Operatoren	AND, OR, NOT	Gibt Wahrheitswerte zurück

Filtern von Daten mit WHERE

Klausel	Ausdruck	Nutzen
SELECT	<Spaltenliste>	Legt Spaltenliste für Ergebnismenge fest
FROM	<Tabellenname>	Sagt aus, welche Tabellen betrachtet werden
WHERE	<Filterausdruck>	Filtert unerwünschte Daten heraus
GROUP BY	<Gruppierungskriterium>	Gemeinsame Spaltenwerte zusammenfassen
HAVING	<Filterausdruck>	Filtert unerwünschte Gruppen hinaus
ORDER BY	<Sortierreihenfolge>	Sortiert nach Spalte(n)

Die WHERE-Klausel

- Folgt der FROM-Klausel, steht aber vor anderen Klauseln
- Kann die in der SELECT-Klausel definierten Aliase nicht erkennen
- Muss als logische Bedingung ausgedrückt werden
- Es werden nur Zeilen angezeigt, welche nach Überprüfen der Bedingung mit „TRUE“ ausgewertet werden

Beispiel WHERE

```
SELECT UnitPrice, ProductName
FROM Northwind.dbo.Products
WHERE UnitPrice < 10;
```

```
SELECT *
FROM Northwind.dbo.Orders
WHERE orderdate BETWEEN ‘1997-02-02‘ AND ‘1998-07-07‘;
```

```
SELECT *
FROM Northwind.dbo.Orders
WHERE orderdate >= ‘1998-01-07‘ AND orderdate < ‘1999.01.01‘;
```

NULL beim Filtern

- NULL ist die Kennzeichnung fehlender Werte
- Ohne fehlende Werte sind die Prädikatausgaben nur TRUE oder FALSE
- Mit fehlenden Werten können Aussagen auch UNKNOWN sein
- Prüfen **immer** mit IS NULL oder IS NOT NULL, **niemals** = NULL oder <> NULL oder != NULL verwenden!

Sortieren von Daten mit ORDER BY

Klausel	Ausdruck	Nutzen
SELECT	<Spaltenliste>	Legt Spaltenliste für Ergebnismenge fest
FROM	<Tabellenname>	Sagt aus, welche Tabellen betrachtet werden
WHERE	<Filterausdruck>	Filtert unerwünschte Daten heraus
GROUP BY	<Gruppierungskriterium>	Gemeinsame Spaltenwerte zusammenfassen
HAVING	<Filterausdruck>	Filtert unerwünschte Gruppen hinaus
ORDER BY	<Spaltenliste>	Sortiert nach Spalte(n), ASC (Standard), DESC

ORDER BY

- Sortiert Zeilen; ohne gibt es keine garantierte Reihenfolge
- Letzte Klausel, welche logisch verarbeitet wird
- Kann auf Spalten nach Name oder Alias verweisen
- Kann auf nicht zur SELECT-Liste gehörigen Spalten verweisen
- Kann auf Rückgabewerte von Skalarfunktionen verweisen
- Sortierreihenfolge mit ASC (aufsteigend) oder DESC (absteigend) festlegen

Syntax ORDER BY

- Mithilfe von Spaltennamen

```
SELECT <Spaltenliste>
FROM <Tabelle>
ORDER BY <Spalte 1>, <Spalte 2>;
```

- Mithilfe von Aliasen und Sortierreihenfolge

```
SELECT <Spaltenliste> AS <Alias>
FROM <Tabelle>
ORDER BY <Spaltenname|Alias> ASC|DESC;
```

Beispiele

```
SELECT OrderID, OrderDate, CustomerID, Freight
FROM Northwind.dbo.Orders
ORDER BY Orderdate;
```

```
SELECT CustomerID, ShipVia, YEAR(OrderDate) AS Orderyear
FROM Northwind.dbo.Orders
ORDER BY Orderdate DESC;
```

TOP

- TOP beschränkt auf Anzahl oder Prozentsatz
- Funktioniert mit ORDER BY-Klausel, nicht immer eindeutig!
  Um Eindeutigkeit zu erzielen, TOP WITH TIES verwenden
- Syntax:

```
SELECT TOP (N) | TOP (N) PERCENT
Oder:
SELECT TOP (N) WITH TIES (mit Duplikaten, nicht deterministisch)
```

- TOP ist eine Eigenfunktion von Microsoft SQL Server

Beispiele I

```
SELECT
      TOP 5 WITH TIES
      ProductName,
      UnitPrice
FROM Northwind.dbo.Products
ORDER BY UnitPrice DESC;
```

Wählt die Produkte mit den 5 teuersten Preisen aus, füllt aber noch auf, bis der Preis sich ändert

```
SELECT
      TOP 5
      ProductName,
      UnitPrice
FROM Northwind.dbo.Products
ORDER BY UnitPrice DESC;
```

Die 5 teuersten Produkte anzeigen

Beispiele II

```
SELECT
      TOP 5 PERCENT
      ProductName,
      UnitPrice
FROM Northwind.dbo.Products
ORDER BY UnitPrice DESC;
```

Wählt die teuersten 5 % der Produkte aus

Bei Prozentwerten wird die Anzahl der Zeilen aufgerundet

OFFSET - FETCH

- Erweiterung der ORDER BY-Klausel, ermöglicht Filtern nach Zeilenbereichen
- Ermöglicht Blättern durch Ergebnisse
- Syntax:

```
ORDER BY <Spaltenliste>
OFFSET <offset-Wert> ROW(S)
FETCH FIRST|NEXT <Abruf-Wert> ROW(S) ONLY
```

Beispiele

- Abrufen der ersten 20 Zeilen

```
SELECT OrderID, CustomerID, OrderDate, ShipName
FROM Northwind.dbo.Orders
ORDER BY OrderDate, OrderID DESC
OFFSET 0 ROWS FETCH FIRST 20 ROWS ONLY;
```

- Abruf der nächsten 20 Zeilen

```
SELECT OrderID, CustomerID, OrderDate, ShipName
FROM Northwind.dbo.Orders
ORDER BY OrderDate, OrderID DESC
OFFSET 20 ROWS FETCH FIRST 20 ROWS ONLY;
```

Regeln für OFFSET - FETCH

- OFFSET-Wert muss angegeben werden
- Optionale FETCH-Klausel gibt alle Zeilen nach dem OFFSET-Wert zurück
- ROW und ROWS, FIRST und NEXT austauschbar
- OFFSET-Wert und Abrufwert können Konstanten, Ausdrücke, Variablen und Parameter sein