Weitere Vergleichsoperatoren

|  |  |
| --- | --- |
| Bereichsabfrage | Spaltenname BETWEEN untere\_Grenze AND obere\_Grenze  inkl. Der angegebenen Grenzen  Hinweis: Die untere Grenze wird zuerst angegeben. |
| Liste | Spaltenname IN ( wert { , wert }) |
| Mustervergleich | Spaltenname LIKE 'muster'  Muster: % steht für 0, 1, 2, … , N Zeichen  \_ steht für genau ein Zeichen  Bemerkung: DOS- dir: \* bzw. ? |
| NULL-Wert | Spaltenname IS [ NOT ] NULL |

12.4. ORDER BY-Klausel

Dient zum Sortieren

SQL-Syntax:



SELECT select\_klausel  
 FROM from\_klausel  
 [ WHERE where\_klausel ]  
 ORDER BY sortierung  
;



Sortierung ::= spaltenname | spaltenaliasname [ ASC | DESC ]

{ , spaltenname | spaltenaliasname [ ASC | DESC ] }

12.5. Funktionen

Funktionen sind Unterprogramme, d.h. sie bestehen aus einer Menge von Anweisungen. An die Funktion kann etwas übergeben werden, dann macht die Funktion etwas und gibt genau einen Wert zurück.



12.5.1. Single Row Funktionen

12.5.1.1. Zeichenfunktionen

upper() - Umwandlung in Großstaben

lower() - Umwandlung in Kleinbuchstaben

instr() - Kommt ein String IN einem anderen String vor

substr() - Teilstring ab einer Position und einer festen Anzahl an Zeichen

Beispiel: substr( String, Position [ , Anzahl ] )

length() - Anzahl der Zeichen im Text

12.5.1.2. Numerische Funktionen

power() - Potenzieren

sqrt() - Quadratwurzel

round() - mathematische Runden

12.5.1.3. Datumsfunktionen

sysdate - liefert das System-Datum und die –Zeit zurück

12.5.1.4. Konvertierungsfunktionen



to\_char ( date\_wert , 'Formatmaske' )

to\_date ( char\_wert , 'Formatmaske' )

to\_number ( char\_wert , 'Formatmaske' )

12.5.1.5. Weitere Funktionen

nvl () - Ersetzt einen NULL-Wert durch einen bestimmten Wert

Beispiel: nvl (commission\_pct, -100)

nvl ( last\_name, 'unbekannt')

Achtung: Beide Parameter müssen vom selben Datentyp sein.

12.5.2. Multiple Row Funktionen

min () - Minimum der Spalte ermitteln

max () - Maximum der Spalte ermitteln

sum() - Summiert die Werte der angegebenen Spalte

avg() - Berechnet den Durchschnittswert der angegebenen Spalte

count() - Zählt die Werte der angegebenen Spalte

count(\*) – Zählt die einzelnen Datensätze

Achtung: Gruppenfunktionen ignorieren NULL-Werte!

Hinweis: Verwende nvl()!

12.6. GROUP BY-Klausel

Dient zum Gruppieren von Datensätzen

SQL-Syntax:

SELECT select\_klausel



FROM from\_klausel



[ WHERE where\_klausel ]



GROUP BY gruppenkriterium



[ ORDER BY order\_by\_klausel ]  
;



Gruppenkriterium ::= spaltenliste | ( )

Spaltenliste ::= spaltenname { , spaltenname }

( ) ::= Die gesamte Menge an Datensätzen, die nach der WHERE-Klausel übrig bleibt, wird als eine einzige Gruppe definiert.

Achtung: In der WHERE-Klausel sind Gruppenfunktionen verboten!

Aufgabe: Gib den Vornamen, den Nachnamen und das größte Alter aller Nachnamen-Gruppen aus.

t\_person

|  |  |  |
| --- | --- | --- |
| **vorname** | **nachname** | **alt** |
| Donald | Duck | 34 |
| Daisy | Duck | 34 |
| Fred | Feuerstein | 567 |

SELECT vorname, nachname, max(alt)  
 FROM t\_person  
 GROUP BY nachname;

|  |  |  |
| --- | --- | --- |
| **vorname** | **nachname** | **Max(alt)** |
| Donald / Daisy | Duck | 34 |
| Fred | Feuerstein | 567 |

Problem: Abfrage ist nicht ausführbar!

Regel:

Wird die GROUP BY-Klausel verwendet, so muss in der SELECT- (HAVING-) Klausel jede Spalte für jede Gruppe genau einen Wert liefern.

Folgerung:

Wird die GROUP BY-Klausel verwendet, so dürfen innerhalb der SELECT- (HAVING-) Klausel nur Spalten aufgeführt werden,

* Die durch den Gruppenausdruck referenziert werden oder



* Die zusammen mit einen Gruppenfunktion auftreten.



SELECT vorname, nachname, max(alt)  
 FROM t\_person  
 GROUP BY nachname;



Lösung: Sämtliche Lösungen erfüllen nicht die gestellte Aufgabe.

1. Vorname aus der SELECT-Klausel entfernen.

SELECT nachname, max(alt)  
 FROM t\_person  
 GROUP BY nachname;

2. Vorname bei GROUP BY aufnehmen

SELECT vorname, nachname, max(alt)  
 FROM t\_person  
 GROUP BY nachname, vorname;

3. Vorname mit einer Gruppenfunktion angeben

SELECT min(vorname), nachname, max(alt)  
 FROM t\_person  
 GROUP BY nachname;

Hinweis:

Gruppenfunktion können nur bis zur Ebene 2 verschachtelt werden.

Beispiel: SELECT min(sum(salary))

12.7. HAVING-Klausel

Gruppierte Datensätze zu filtern

SQL-Syntax:

SELECT select\_klausel



FROM from\_klausel



[ WHERE where\_klausel ]



[ GROUP BY group\_by\_klausel ]



HAVING gruppenkriterium



[ ORDER BY order\_by\_klausel ]  
;



Gruppenkriterium ::= Bedingung, aber Gruppenfunktion sind erlaubt

12.8. Unterabfragen

Eine SELECT-Anweisung in einer SELECT-Anweisung.

Eine Unterabfrage ist eine SELECT-Anweisung, die einer anderen SQL-Anweisung untergeordnet ist.



Die Unterabfrage wird genau einmal

a) entweder für die gesamte Hauptabfrage oder

b) für jede Zeile der Hauptabfrage ( korrelierende Unterabfrage )

ausgeführt.

Zu a)

Schritte:

1. Die Unterabfrage wird vollständig bearbeitet und das Ergebnis wird temporär zwischengespeichert.

2. Die Hauptabfrage wird unter Einbeziehung des temporär gespeicherten Ergebnisses der Unterabfrage bearbeitet.

Zu b)

Schritte:

1. Abrufen der in Frage kommenden Zeile der Hauptabfrage

2. Ausführen der Unterabfrage basieren auf der in Frage kommenden Zeile der Hauptabfrage und das Ergebnis wird temporär zwischengespeichert.

3. Beantwortung der in Frage kommenden Zeile der Hauptabfrage unter Einbeziehung der temporär gespeicherten Ergebnismenge.

Single Row-Operatoren: = < <= != <> (Vergleichsoperatoren)

Multiple Row-Operatoren:

* Vergleichsoperator ANY (Bedingung muss irgend einen Wert in der Unterabfrage erfüllt sein)
* Vergleichsoperator ALL (Bedingung muss alle Werte in der Unterabfrage erfüllt sein)
* IN (Bedingung muss irgend einen Wert in der Unterabfrage erfüllt sein)  
  IN entspricht =ANY

EXISTS:

* Wird bei korrelierenden Unterabfragen eingesetzt
* EXISTiert in der Unterabfrage ein Ergebnis?
* Ja: Ausgabe der Zeile Nein: keine Ausgabe der Zeile