INDEX UND SELECT IN SQL

1. Beispiel DATENBANK

```
create table Ware
        (Artikel# SmallInt,
             Bezeichnung CHAR(20),
             Preis SmallInt,
             ME  CHAR(3),
             Bestand SmallInt
            );

create table Kunde
            (Kunden# SmallInt,
             Name CHAR(20),
             Adresse CHAR(40),
             PLZ SmallInt,
             Ort CHAR(20)
            );

create table Auftrag
            (Re# SmallInt,
             Kunden# SmallInt,
             Datum date,
             status CHAR(3)
            );

create table Pos
            (Re# SmallInt,
             Pos# SmallInt,
             Artikel SmallInt,
             Menge SmallInt
            );
```

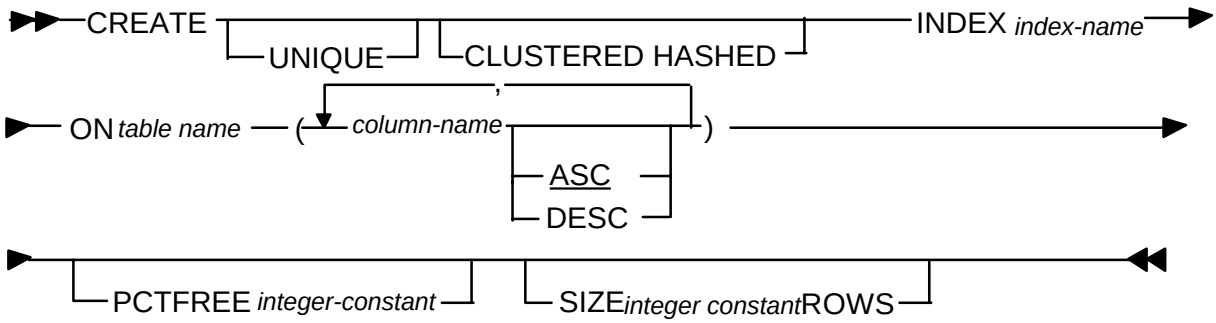
2. Der Index

2.1. Allgemeines

Wird eine Zeile in einer realen Tabelle gesucht, die in einer Spalte bzw. Spaltenkombination vorgegebene Werte enthält, so muß die gesamte Tabelle durchsucht werden. die realen Implementationen sind bemüht, die Zugriffe auf den externen Speicher möglichst gering zu halten, da diese Zugriffe langsam sind. Deshalb sind üblicherweise Methoden implementiert, die es gestatten, Zeilen direkt zu adressieren. Dies erreicht man durch Errichtung eines Indexes. Ist für die betrachtete Tabelle ein Index auf die Spalte bzw. Spaltenkombiantion errichtet worden, so muß nur im Index nachgeschaut werden, wo die betreffende Zeile steht (es können auch mehrere Zeilen sein, die die Bedingung erfüllen). ein Index ist in der Regel wesentlich kleiner als

die Tabelle und zudem so organisiert, daß er schnell durchsucht werden kann. Dadurch
wird beim Suchen die Zahl der Zugriffe auf den externen Speicher erheblich verringert.

2.2. Syntax



6 + Anzahl der Spalten + Summe der Länge der Spalten ≤ 255

Die Länge der einzelnen Spalten hängt von ihren Datentypen ab.

Datentyp	Länge
Character	pro Zeichen 1 Byte
Numeric	12 Bytes
Date/Time	12 Bytes

Ein Beispiel für die Berechnung von Indexes.

Nachname CHAR(20)
Vorname CHAR(20)
Geschl CHAR(1)
41
daraus ergibt sich:

6 + 3 + 41 = 50

SEHR_GROSS CHAR(249)

6 + 1 + 249 = 256

256 > 255

Es kommt zur Fehlermeldung:

Index key is too large

2.2.1. index-name

Der Indexname muß in der gesamten Datenbank eindeutig sein.

2.2.2. table-name

Auf Views können keine Indexe gelegt werden.

2.2.3. UNIQUE

Erzeugt einen Schlüssel - Index. Die Wertekombination der einzelnen Elemente darf in der Tabelle nur ein einziges mal vorkommen. Existieren bereits zur Indexerstellung doppelte Werte kommt es zu einer Fehler Meldung. Ebenso bei Update oder Insert.

2.2.4. CLUSTERED HASHED

Ein Clustered Hashed Index erlaubt einen schnellern Zugriff auf die Daten. Besteht ein Index aus unique und clustered hashed kann mit einem Plattenzugriff eine Zeile ausgelesen werden.

Ist kein clustered hashed definiert wird der Index als B-Baum erzeugt.

2.2.5. ASC,DESC

Definiert ob der index aufsteigend oder Absteigend sortiert wird. Diese Klausel ist nur für B-Bäume interessant. Wird die Klausel nicht angeben wird default mäßig ASC genommen.

2.2.6. PCTFREE

Die PCTFREE (prozent frei) klausel definiert, wieviel Platz in jeder Indexseite sein muß wenn der Index erstellt wird. Wird diese klausel nicht angegeben ist der default Wert 10%

2.2.7. SIZE *integer-constatnt* ROWS

Diese Angabe kontrolliert die größe des Indexes und wird in Anzahl der Reihen angegeben. Ist diese größe zu klein angeben kann es overflow pages kommen. Ist die angebene größe zu groß, wird das overflow page nicht verwendet, aber der Platz auf der Platte ist verbraucht. Diese Klausel wird nur bei clustered hashed Indexe benötigt. und ist dort unbedingt erforderlich.

2.3. Funktionen in Index

Ein Index kann für mehrere Spalten gebildet werden. Diese Spalten können durch Funktionen verbunden werden. Es können jedoch nicht alle Funktionen in Index eingebaut werden.

@CHAR
@CODE
@DATEVALUE
@DAY
@HOUR
@LEFT
@LENGTH
@LICS
@LOWER
@MICROSECOND
@MID
@MINUTE
@MONTH
@MONTHBEG
@PROPER
@QUARTER
@QUARTERBERG
@RIGHT
@SECOND
@STRING
@SUBSTRING
@TIMEVALUE
@TRIM
@UPPER
@VALUE
@WEEKBEG
@WEEKDAY
@YEAR
@YEARBEG
@YEARNUM

2.4. Beispiele

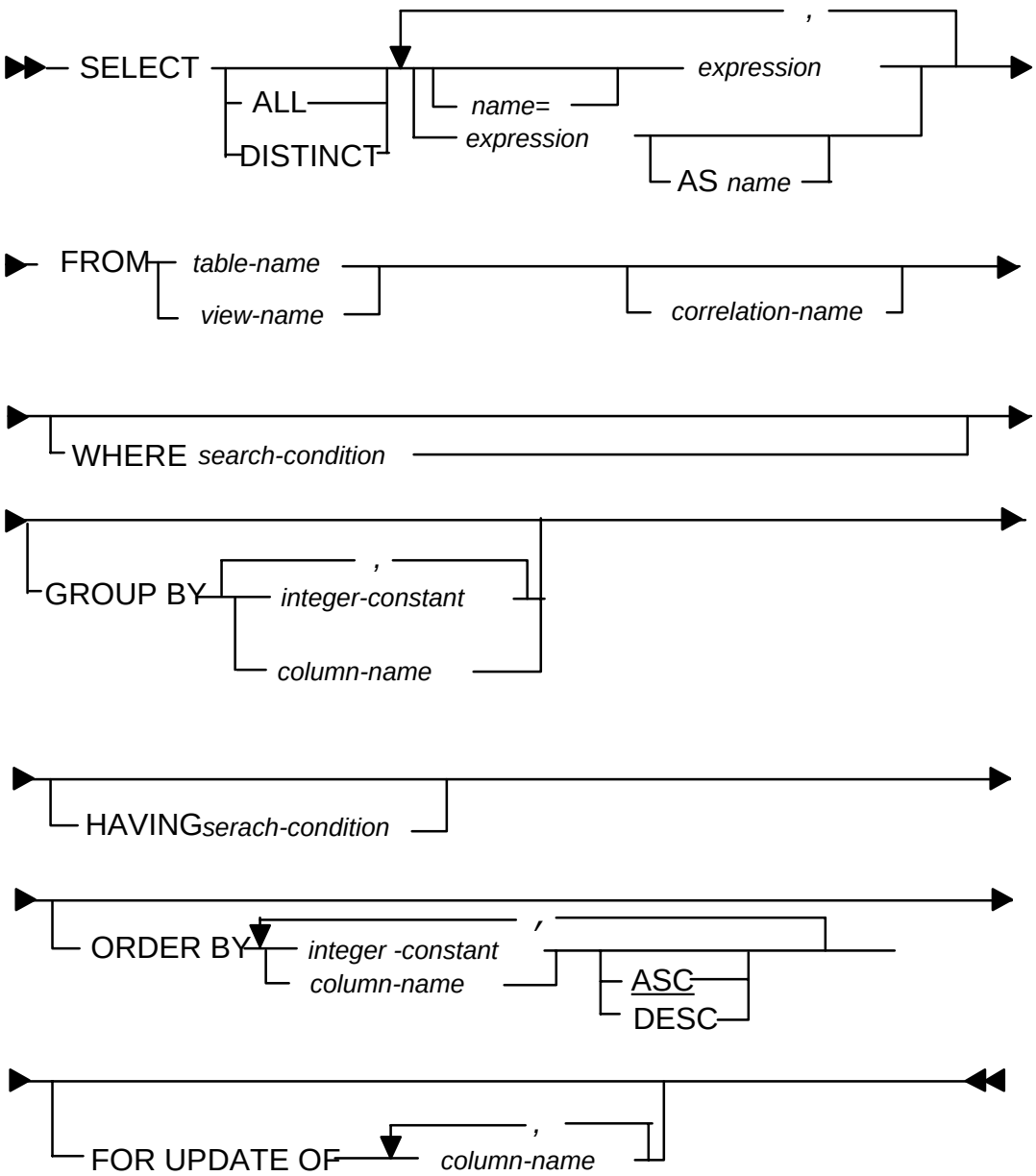
```
CREATE UNIQE INDEX sWare ON Ware (Artikel#)
CREATE UNIQE INDEX sKunde ON Kunde (Kundenl#)
CREATE UNIQE INDEX sAuftrag ON Auftrag(RE#)
CREATE UNIQE INDEX sPos ON Pos (RE#,POS#)

CREATE INDEX KUINDEX ON KUNDE (@UPPER(name))

SELECT name FROM KUNDE
        WHERE @UPPER(name) = 'HUBER'
```

3. SELECT

3.1. Syntax



Im Prinzip besteht das SELECT Statement aus sechs Komponenten

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

Zwei Punkte sind äußerst wichtig und sollten immer vor Augen gehalten werden:

• Die Reihenfolge der Kompnenten ist fix vorgegben; eine GROUP BY Komponente darf nicht vor einer WHERE oder vor einer FROM Klausel stehen.

• Eine HAING-Komponente darf nur dann verwendet werden wenn eine GROUP BY Komponente verwendet wird.

FROM
definiert die Ausgangstabellen

WHERE
selektiert die Reihen, die der Bedingung genügen

GROUP BY
gruppiert Reihen auf Basis gleicher Werte in Spalten

HAVING
selektiert Gruppen, die der Bedingung genügen

SELECT
selektiert Spalten

ORDER BY
sortiert Reihen auf der Basis von Spalten

3.2. Verarbeitung einer SELECT ANWEISUNG

Ausgabe aller Kundennummer die 1993 mehr als einen Einkauf getätigt haben.

```
SELECT Kunden#
FROM Auftrag
WHERE DATUM > 01011993
GROUP BY Kunden#
HAVING COUNT(RE#) >1
ORDER BY Kunden#
```

3.2.1. FROM

In der FROM-Komponente wird nur die Tablle Auftrag genannt. Die bedeutet für SQL, daß mit der Tabelle AUFTRAG gearbeitet werden muß. Das Zwischenergebnis dieser Operation ist eine exakte Kopie der Tabelle AUFTRAG

Zwischenergebnis:

RE#	Kunden#	Datum	status
1	1	1990-01-01	fa
2	1	1993-05-19	fa
3	3	1993-05-19	fa
4	2	1993-05.17	fa
5	3	1993-07-19	fa