6. SELECT mit Gruppenbildung

Während die im vorigen Abschnitt behandelten virtuellen Attribute so zustande kommen, dass für jede Zeile einzeln aus vorhandenen Attributwerten neue berechnet werden, fasst die Gruppenbildung die Werte eines Attributs für verschiedene Tupel der Relation zusammen.

COUNT

Wie viele Kunden gibt es? Wir zählen die Zeilen der Kundentabelle:

```
SELECT COUNT(*)
   FROM T_Kunden;
```

Die Funktion COUNT(*), die hier angewandt wird, ist eine Aggregatfunktion. Sie zählt in diesem Fall die Anzahl aufgefundener Tupel. Wenn wir statt * den Bezeichner einer Spalte bei COUNT angeben, wird die Anzahl der vorhandenen Werte ohne Nullmarken gezählt. Daher erhalten wir bei der Relation T_Bestellungen verschiedene Zahlen, wenn wir die Werte in Bestelldatum und Lieferdatum zählen lassen.

```
SELECT COUNT(Bestelldatum), COUNT(Lieferdatum)
   FROM T_Bestellungen;
```

Auffällig sind erstens die verschiedenen Werte, zweitens die Tatsache, dass die Ergebnisrelation nur eine einziges Tupel aufweist. Die Funktion COUNT berechnet also aus allen Tupeln einer Relation genau einen Wert.

Schließlich kann mit der COUNT-Funktion auch die Anzahl verschiedener Werte eines Attributs ermittelt werden. Beispielsweise lassen wir mit folgender Anweisung die Anzahl verschiedener Postleitzahlen bei unseren Kunden anzeigen:

```
SELECT COUNT(DISTINCT PLZ)(*)
   FROM T_Kunden;
```

Zusammengefasst stellen wir fest, dass die Aggregatfunktion COUNT drei verschiedene Formen hat:

```
COUNT(*)
COUNT(Attribut)
COUNT(DISTINCT Attribut)
```

Weitere Aggregatfunktionen

In SQL sind genau fünf Aggregatfunktionen definiert, die dort als set functions bezeichnet werden:

COUNT	Anzahl von Tupeln, Werten oder verschiedenen Werten
SUM	Summe der Werte
AVG	Durchschnitt der Werte
MAX	Größter Wert
MIN	Kleinster Wert

Im Übrigen kann - wie bei COUNT - jeweils das Schlüsselwort DISTINCT vor der Attributbezeichung stehen.

(*) Diese Form wird von MS-ACCESS nicht unterstützt.

Alle Aggregatfunktionen lassen sich in einer einzigen SELECT-Anweisung unterbringen, wie das folgende Beispiel zeigt. Wir fragen nach der Anzahl aller Artikel, der Summe und dem Durchschnitt ihrer Einzelpreise sowie nach dem größten und kleinsten Einzelpreis.

```
SELECT
        COUNT(Artikel_Nr) AS ANZAHL,
        SUM(Listenpreis) AS PREISSUMME,
        AVG(Listenpreis) AS DURCHSCHNITT,
        MIN(Listenpreis) AS 'KLEINSTER PREIS',
        MAX(Listenpreis) AS 'GROESSTER PREIS'
    FROM T_Artikel;
```

Einschränkungen der Projektion bei Anwendung von Aggregatfunktionen

Die Aggregatfunktionen in den bisher vorgestellten Beispielen fassen eine ganze Relation in einem Tupel zusammen. Aus diesem Grund ist es nicht möglich, zusätzlich noch einzelne Attributwerte auszugeben, denn dann müsste die Ergebnisrelation mindestens so viele Tupel enthalten, wie es verschiedene Werte in dem gewünschten Attribut gibt.

Das heißt:

Wenn in der SELECT-Klausel mindestens eine Aggregatfunktion auftritt, dann dürfen keine elementaren Attribute mehr auftreten!

A06-1

Warum liefert im obigen Fall die Funktion COUNT(Artikel_Nr) mit Sicherheit dasselbe Ergebnis wie COUNT(*) ? Was würde bei Anwendung von COUNT(DISTINCT Artikel_Nr) herauskommen?

Antwort:

Nullmarken bei Aggregatfunktionen

Nullmarken erfahren bei Anwendung der Aggregatfunktionen keine Berücksichtigung. Dies führt bei der Summenbildung zu demselben Resultat, als wenn NULL die Zahl 0 repräsentierte. Bei der Minimum- und Maximumbildung und bei der Durchschnittsbildung fallen Nullmarken ganz heraus. Bei leeren Ergebnismengen sind MIN, AVG, MAX nicht definiert. Die COUNT-Funktion, auf ein Attribut angewandt, zählt die tatsächlich vorhandenen Werte. Bei Spalten, für die Nullmarken zulässig sind, kann deshalb das Ergebnis COUNT(Attribut) kleiner sein als COUNT(*).

7. SELECT mit GROUP BY

Die oben diskutierten Aggregatfunktionen erlauben weitergehende Auswertungen, indem bestimmte Teilmengen der Tupel einer Relation zu Gruppen zusammengefasst werden. Das Zählen, die Durchschnitts- und Summenbildung, die Ermittlung von Minimal- und Maximalwerten kann beispielsweise bezogen werden auf

- jeweils alle Bestellungen eines Kunden,
- jeweils alle Positionen einer Bestellung,
- alle Positionen, in denen ein bestimmter Artikel bestellt oder berechnet wird,
- alle Kunden einer Stadt.

Entscheidend für die Gruppenbildung sind gleiche Werte in einem bestimmten Attribut. Für Bestellungen desselben Kunden ist der Wert der Spalte KundenNr in Bestellungen immer gleich. Bei Positionen, die denselben Artikel betreffen, ist der Wert der Spalte ArtikelNr in der Tabelle T_Position immer derselbe.

Eine Abfrage-Anweisung mit GROUP BY hat folgende Struktur:

```
SELECT gruppenausdrucksliste
    FROM relation
    GROUP BY attributliste
```

Die hinter GROUP BY angegebene Attributliste wird als Gruppierungskriterium benutzt, d.h. alle Zeilen der Tabelle, die in diesen Spalten denselben Wert haben, werden zu einer Gruppe zusammengefasst. Im einfachsten Fall besteht die Attributliste aus einem einzigen Attribut.

Beispiele

Wie oft hat jeder Kunde bestellt?

```
SELECT F_Kunden_Nr, COUNT(*)
    FROM T_Bestellungen
    GROUP BY F_Kunden_Nr;
```

Für jeden Artikel ist die Anzahl aller Bestellungen zu ermitteln:

```
SELECT F_Artikel_Nr, SUM(Bestellmenge)
    FROM T_Positionen
    GROUP BY F_Artikel_Nr;
```

Wann war die jeweils letzte Bestellung der Kunden?

```
SELECT F_Kunden_Nr, MAX(Bestelldatum)
    FROM T_Bestellungen
    GROUP BY F_Kunden_Nr;
```

Auch auf virtuelle Attribute können Gruppierungsfunktionen angewandt werden. Das folgende Beispiel berechnet für jeden Lagerplatz die durchschnittliche Differenz zwischen Lagerbestand und Mindestbestand aller Artikel.

```
SELECT Lagerplatz, AVG(Bestand - Mindestbestand)
    FROM T_Artikel
    GROUP BY Lagerplatz;
```

Es ist klar, dass die Werte des Gruppierungsattributs jeweils eindeutig in der Ergebnisrelation sind. Somit können insbesondere keine Tupel mehrfach auftreten. Es ist sinnvoll, alle Gruppierungsattribute in der SELECT-Klausel anzugeben - sonst könnten wir die übrigen Informationen ja gar nicht zuordnen.

Die Gruppierung fasst in der Regel mehrere Tupel der Ausgangsrelation zu einem einzigen Tupel der Ergebnisrelation zusammen. Daraus ergibt sich, dass die Attribute, die innerhalb einer Gruppe verschiedene Werte aufweisen, nicht mehr einzeln angezeigt werden können. Es kann pro Gruppe und pro Attribut nur noch einen Wert geben, der eben durch Anwendung der Aggregatfunktionen erzeugt wird.

Wenn also, um auf das letzte Beispiel zurückzukommen, die letzte Bestellung eines Kunden gefragt ist, können nicht alle einzelnen Bestelldaten angezeigt werden. Ein solcher Versuch wird vielmehr mit einer Fehlermeldung quittiert, wie im folgenden Beispiel:

Illegale Anweisung

```
SELECT F_Kunden_Nr, Bestelldatum
    FROM T_Bestellungen
    GROUP BY F_Kunden_Nr;
```

Die Verwendung der Gruppierung muss also auf Fälle beschränkt bleiben, in denen die Einzelwerte innerhalb einer Gruppe tatsächlich nicht relevant sind. Sonst gibt man besser eine sortierte Liste aus. Die folgende Ausgabe erlaubt sowohl die Feststellung, wann Kunden im Einzelnen bestellt haben, wie oft sie bestellt haben und wann die letzte Bestellung war. Diese Daten müssen aber durch Interpretation aus der Liste gewonnen werden, sie werden (bis auf die Einzelwerte) nicht direkt angezeigt.

```
SELECT F_KundenNr, Bestelldatum
    FROM T_Bestellungen
    ORDER BY F_Kunden_Nr, Bestelldatum;
```

Regel für die Projektion bei Gruppenbildung

Im Zusammenhang mit einer GROUP BY-Klausel kann auf folgende Attribute projiziert werden:

- die Gruppierungsattribute selbst
- Aggregatfunktionen, angewendet auf Attribute der Tabelle
- Aggregatfunktionen, angewendet auf virtuelle Attribute

Nullmarken in den Gruppierungsspalten

Die Gruppierung auf eine Spalte mit Nullmarken macht aus allen Nullmarken eine Gruppe. Allgemeiner gesagt bildet bei einer Gruppierung mit mehreren Attributen jede Kombination - auch unter Einbeziehung der Nullmarken - eine Gruppe.

Als Beispiel wird die Frage gestellt, wie viele Kunden bei jeweils einer Werbeaktion betreut wurden. Auch die Gruppen, die noch nie bei einer Werbeaktion angesprochen wurden, bilden dann eine Gruppe.

```
SELECT letzteWerbeakt, COUNT(*)
    FROM T_Kunden
    GROUP BY letzteWerbeakt;
```

A07-1
Lassen Sie pro Mwst-Satz die Anzahl der davon betroffenen Artikel anzeigen.

A07-2
Geben Sie jeweils die größte Kundennummer der Kunden mit demselben Status aus.

A07-3
Zeigen Sie je Lagerplatz den kleinsten, den größten und den durchschnittlichen Listenpreis der dort gelagerten Artikel an.

Reihenfolge der Komponenten der SELECT-Anweisung

Da nunmehr alle Klauseln der SELECT-Anweisung vorgestellt worden sind, kann ein Resümee gezogen werden.

Syntaktisch gesehen sind die einzelnen Klauseln der SELECT-Anweisung zwingend in folgender Reihenfolge anzuordnen:

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. ORDER BY

Jede andere Reihenfolge führt zu einer Fehlermeldung.

Von der logischen Bearbeitung her gilt allerdings etwas anderes:

Die FROM-Klausel wird zuerst ausgewertet. Damit stehen die Relationen fest, auf die sich die Abfrage richtet. Als nächstes folgt die Selektionsklausel WHERE. Es schließt sich die Gruppierung aufgrund der GROUP BY-Klausel an. Zum Schluss werden die Attribute der Ergebnisrelation mit der SELECT-Klausel festgelegt. Etwaige Berechnungen von virtuellen Spalten werden also auch erst dann vorgenommen, wenn feststeht, welche Tupel die Ergebnisrelation umfasst.

Die technische Abarbeitung einer SELECT-Anweisung kann durchaus von der hier gezeigten logischen Abfolge abweichen.