1. SELECT - Anweisung mit Projektion

Wir stellen zunächst dar, wie die Projektion mit der Abfrage-Anweisung realisiert wird. Alle anderen Operationen werden zunächst ausgeklammert. Die Abfrage - Anweisung für die Projektion auf eine Relation gerät unter diesen Voraussetzungen recht kurz:

```
SELECT [DISTINCT] attribut(e)
   FROM relation
```

SELECT

Diese Klausel leitet die Projektionsangabe ein. Hier werden die Attribute aufgelistet, die die Ergebnisliste enthalten soll. Dabei kann der Platzhalter * anstelle einer expliziten Attributliste verwendet werden, um alle Attribute der Relation zu projizieren.

FROM

Hier wird die gewünschte Tabelle angegeben, aus der die Daten entnommen werden sollen.

Beispiel

Eine Kundenliste kann durch Projektion der Relation T_Kunden auf die Attribute Kunden_Nr und Nachname erzeugt werden.

```
SELECT Kunden_Nr, Nachname
   FROM T_Kunden;
```

Übung: Führen Sie diese Abfrage aus, verändern Sie die projizierten Attribute, projizieren Sie mit * auf alle Attribute. Verdeutlichen Sie sich die Wirkung der Abfrage durch einen Vergleich von Ausgangs- und Ergebnisrelation

DISTINCT

Durch die Angabe von DISTINCT werden identische Zeilen in der Ergebnistabelle zusammengefasst. Zwei Tupel sind identisch, wenn sie in allen Attributen denselben Wert haben.

Beispiel

Welche Zahlungsarten sind tatsächlich Kunden zugeordnet? (Möglich wären fünf verschiedene Werte.)

```
SELECT DISTINCT Zahlungsart
   FROM T_Kunden;
```

Übung: Führen Sie diese Abfrage mit und ohne DISTINCT aus.

2. Sortieren der Ergebnistabelle

Meistens erwartet der Benutzer eine Ausgabe, die nach irgendeinem Merkmal sortiert ist. Dafür muss mit ORDER BY eine Sortieranweisung gegeben werden. Ohne ORDER BY - Klausel erzeugt SQL eine undefinierte Reihenfolge der Tupel. Die Syntax lautet:

```
SELECT [DISTINCT] attribut(e)
   FROM relation
   ORDER BY attribut(e)
```

ORDER BY

Diese Klausel dient zur sortierten Ausgabe der Liste. Die Ausgabe wird nach den Werten der angegebenen Attribute sortiert. Bei mehreren Attributen wird hierarchisch sortiert: Zuerst nach dem erstgenannten Attribut, bei Wertgleichheit nach dem zweitgenannten Attribut usw.

Beispiel

```
SELECT Kunden_Nr, Vorname, Nachname
    FROM T_Kunden
    ORDER BY Nachname
```

Übung: Führen Sie diese Abfrage mit und ohne ORDER BY - Klausel aus. Lassen sie abwechselnd nach Vor- und Nachname sortieren.

Beispiel

Sortieren Sie alle Kunden nach dem Attribut Zahlung und dann als zweites nach der PLZ.

```
SELECT Zahlungsart, PLZ, Nachname
    FROM T_Kunden
    ORDER BY Zahlungsart, PLZ;
```

Übung: Führen Sie diese Abfrage aus, ändern sie die Sortierreihenfolge.

A02-1

Lassen Sie zu allen Positionen die Bestellnummer und die Artikelnummer ausgeben. Sortieren Sie hierarchisch nach Bestellnummer und Artikelnummer.

A02-2

An welchen (verschiedenen) Lagerplätzen haben Sie Artikel?

A02-3

Lassen Sie die verschiedenen Paare von PLZ und Ort der Kundentabelle ausgeben.

---

3. SELECT mit Projektion und Selektion

Eine der wichtigsten Operationen ist die Selektion, also das Auffinden und Anzeigen einer Teilmenge der Tupel einer Relation. Beispielsweise interessieren wir uns speziell für die Stammkunden. Um eine Liste zu erhalten, in der nur Stammkunden vorkommen, muss die Teilmenge von Tupeln der Kundenrelation ausgewählt und angezeigt werden, für die das Attribut Status den Wert „S“ hat.

Diese Operation erfordert eine WHERE - Klausel, in der die Bedingung Status = 'S' formuliert wird.

```
SELECT *
    FROM T_Kunden
    WHERE Status = 'S';
```

Übung: Führen Sie diese Abfrage aus, experimentieren sie mit den Attributen der WHERE - Klausel. Kombinieren Sie die Abfrage mit einer Projektion auf Kunden_Nr, Nachname und Status.

WHERE

Die allgemeine Form einer WHERE -Klausel sieht wie folgt aus:

```
WHERE ausdruck1 vergleichsoperator ausdruck2
```

Alle folgenden Beispiele sind gleichwertig und nach dem SQL-Standard zulässig:

```
WHERE Mindestbestand = 50
WHERE 50 = Mindestbestand
WHERE Mindestbestand * 2 = 400 / 4
```

Beispiel

In einer WHERE - Klausel können auch die Werte zweier Attribute miteinander verglichen werden. Die nächste Abfrage beantwortet die Frage, bei welchen Bestellpositionen die gelieferte Menge kleiner als die bestellte Menge ist.

```
SELECT F_Bestell_Nr, F_Artikel_Nr, Bestellmenge, Liefermenge
    FROM T_Positionen
    WHERE Liefermenge < Bestellmenge;
```

Übung: Führen Sie diese Abfrage aus, verändern Sie die den Vergleichsoperator.

Die nächste Variante beantwortet die Frage, bei welchen Positionen die Liefermenge mindestens 60 % unter der Bestellmenge liegt.

```
SELECT F_Bestell_Nr, F_Artikel_Nr, Bestellmenge, Liefermenge
    FROM T_Positionen
    WHERE Liefermenge <= Bestellmenge * 0.4;
```

Übung: Führen Sie diese Abfrage aus, verändern Sie die den Vergleichsoperator und den Faktor.

Vergleichsoperatoren	
kleiner	<
größer	>
kleiner gleich	<=
größer gleich	>=
ungleich	<>

Logische Operatoren

Bedingungen können in der WHERE - Klausel logisch miteinander verknüpft werden. Folgende logische Operatoren sind verfügbar:

Logische Operatoren	
Ausschluss aller Tupel, die die Bedingung erfüllen.	NOT
Beide Bedingungen müssen erfüllt sein.	AND
Mindestens eine bedingung muss erfüllt sein.	OR

Bei den logischen Operatoren gelten folgende Prioritätsregeln:

➔ NOT geht vor …

➔ AND geht vor …

➔ OR.

➔ Klammern gehen in jedem Fall vor.

Beispiel

Die folgenden Beispiele sollen die Verwendung von logischen Operatoren verdeutlichen.

Zeige eine Liste der Artikel mit F_Mwst_Nr = 1 und zugleich Bestand > 100:

```
SELECT Artikel_Nr, F_Mwst_Nr, Bestand
    FROM T_Artikel
    WHERE F_Mwst_Nr = 1 AND Bestand > 100;
```

Übung: Führen Sie diese Abfrage aus, ändern Sie die den Operator auf OR.

Beispiel

Liste alle mit F_Mwst_Nr ungleich 1 oder Bestand > 100 auf:

```
SELECT Artikel_Nr, F_Mwst_Nr, Bestand
    FROM T_Artikel
    WHERE NOT F_Mwst_Nr = 1 OR Bestand > 100;
```

Übung: Führen Sie diese Abfrage aus, vergleichen Sie das Ergebnis mit dem Ergebnis der nächsten Abfrage.

```
SELECT Artikel_Nr, F_Mwst_Nr, Bestand
    FROM T_Artikel
    WHERE NOT (F_Mwst_Nr = 1 OR Bestand > 100);
```

Nullmarken in Vergleichen

In relationalen Datenbanken ist es möglich, dass für ein Tupel der Wert eines Attributs fehlt, man spricht dann von einer Nullmarke. Wir sagen auch: „*Das Attribut ist NULL*“. Eine Nullmarke in einer Datenbank sagt aus, dass der Wert des Attributs unbekannt (UNKNOWN) ist! Eine Nullmarke ist nicht identisch mit dem numerischen Wert 0 oder dem Leerzeichen!

Die Tatsache, dass kein Attribut vorhanden ist, wird durch eine Markierung in der Datenbank festgehalten. Diese Markierung kann abgefragt werden. Für die Feststellung, ob in einer Spalte eine Nullmarke vorhanden oder nicht vorhanden ist, müssen zunächst einmal spezielle Operatoren her, denn die Aussage

```
liefermenge = NULL
```

liefert immer den Wahrheitswert UNKNOWN, da die Liefermenge mit einem unbekannten Wert verglichen wird, der gleich oder auch ungleich sein könnte.

Unter SQL ist die Abfrage nach den Bestellpositionen, bei denen für die Liefermenge noch nichts eingetragen wurde, folgendermaßen zu formulieren:

```
SELECT F_Bestell_Nr, F_Artikel_Nr, Bestellmenge, Liefermenge
    FROM T_Positionen
    WHERE Liefermenge IS NULL;
```

Der Vergleichsoperator für Nullmarken heißt:

```
IS NULL
```

und sein Gegenstück:

```
IS NOT NULL
```

A03-1

Ändern Sie die beiden Beispiele vom Anfang der Seite so ab, dass man ohne den NOT - Operator zur gleichen Ergebnismenge kommt.

A03-2

Bei welchen Positionen sind Artikel mit den Nummern 'G001', 'G002' oder 'G003' geliefert worden? (**A03-2-1:** Bei welchen dieser Positionen ist die Liefermenge größer als 2?)

A03-3

Bei welchen Positionen sind keine Artikel mit den oben genannten Nummern geliefert worden?

A03-4

Welche Kunden sind keine Stammkunden? Geben Sie die Kundennummer, den Namen, den Ort und den Status aus.

A03-5

Welche Kunden sind keine Stammkunden und nicht aus Husum?