Datenbankabfrage - eine Einführung am Beispiel der

Terra

Inhaltsverzeichnis

1 Vorwort

Ziel ist es, Schülerinnen und Schülern ein Script zum Thema „Datenbanken“ zur Verfügung zu stellen. Dieser Text soll insbesondere die *Abfrage* von Datenbanken am Beispiel der *Terra-Datenbank*, einer umfangreichen, relationalen Datenbank, einführen.

Ich habe dabei den Lehrplan „Informatik am Gymnasium“ im Freistaat Sachsen zugrunde gelegt. Besonderer Augenmerk liegt auf der Behandlung von „Datenbank- und Informationssystemen“ im Grundkurs der Jahrgangsstufe 12. Sicherlich können auch jüngere Schüler die Problematik Datenbanken verstehen und mit der Terra-Datenbank arbeiten.

Im Text wird der Aufbau und die Abfrage einer Datenbank illustriert. Die Terra-Datenbank[1] hat den Vorteil, dass sie einen großen Datenbestand zur Geographie liefert, Daten mit denen jeder Schüler umgehen kann, an denen aber auch die Aufgaben und Notwendigkeit der Verwendung von Datenbanken erkennbar wird.

Es ist nicht mein Anliegen das Thema umfassend und vollständig zu behandeln. Ich möchte lediglich den Schülerinnen und Schülern ein Mittel an die Hand geben, mit dem sie selbständig, unterrichtsbegleitend arbeiten können. Die einzelnen Kapitel sind soweit möglich unabhängig voneinander und können in unterschiedlicher Reihenfolge miteinander kombiniert oder auch ganz ausgelassen werden. Da jeder Lehrer eigene Prioritäten setzt und diese im Verlauf seines Unterrichts oftmals den Gegebenheiten anpassen muss, ist dies die **Version 1.1** mit Stand Juni 2011. Ich weiß, dass einig dieser Texte dringend aktualisert bzw. verbessert werden müssen. Für Ihre Anregungen oder Mitarbeit bin ich stets dankbar. F. Müller

1.1 Lehrplanbezug

Wahlgrundkurs 12/I: Anwendungen der Informatik 30 Std.

Für das erste Halbjahr des 12. Schuljahres besteht die Möglichkeit, zwischen den Varianten "Datenbank- und Informationssysteme", "Prozessdatenverarbeitung" und "Computergrafik" zu wählen. ...
Auf diesen Gebieten ist für die Schülerinnen und Schüler eine praxisnahe Behandlung komplexer Anwendungsfälle zu realisieren, um die Wechselwirkungen zwischen dem Menschen und den Methoden und Werkzeugen der Informatik zu verinnerlichen. Die Auswahl der Themen sollte je nach Kurszusammensetzung, Profil und Ausrüstung der Schule mit Hardware und Software erfolgen[2].
Neben den fachlichen Fragestellungen sollen sich die Schülerinnen und Schüler mit ihren eigenen Erfahrungen als Anwender und Betroffene von Computertechniken und Medien auseinandersetzen. Möglichkeiten von Betriebsbegehungen sollten genutzt werden. Die Schülerinnen und Schüler lernen exemplarisch Auswahlkriterien und Konfigurationsmöglichkeiten für Standardsoftware kennen.

Variante a: Datenbank- und Informationssysteme

Mit der Einführung in ein kommerzielles Datenbanksystem werden die Methoden des Zugangs zu Informationen und die Informationsverarbeitung sowie die Arbeit in Informationsnetzen vermittelt. Schwerpunkte sind die Beschaffung und Strukturierung von Informationen und die Präsentation von Wissen, wobei mit der Benutzeroberfläche, ohne Programmiersprache des Systems, gearbeitet wird.

1 Die Terra-Datenbank stammt aus: Dürr, M., Radermacher, K.: " Einsatz von Datenbanksystemen - Ein Leitfaden für die Praxis". Berlin u.a.: Springer-Verlag 1990.

2 Insofern ist auch die Unterteilung des praktischen Teils in SQL-Abfragen per Internet-Datenbank und Abfragen mittels benutzerfreundlicher Eingabe der SQL-Befehle in Applikationen wie MS Access oder StarOffice und Verwendung einer lokalen Datenbank gerechtfertigt.

Analysieren von Daten *selektieren* *sortieren* *verknüpfen* *Struktur einer Datenbank* *erstellen, ergänzen, verändern* *Datenfernübertragung* *Vernetzte Systeme im persönlichen Alltag* *Vergleich der Geräte und Verfahren zur Datenerfassung* *Nutzung von Datenbanken in der Textverarbeitung*	*Nutzen einer vorgegebenen branchenspezifischen Datenbank (z. B. Periodensystem, Kontenverwaltung)* *Beispiele aus dem Erfahrungsbereich, die eine relationale Anwendung ermöglichen* *Nutzung bestehender simulierter oder realer Informationsnetze* *Tastatur, Scanner, Markierungs- und Magnetkartenleser, Symbol- und Objekterkennung (EAN, ISBN)* *Serienbriefe, Berichte*

[LPSa]

1.2 Inhalt

Der Text ist in Theorie, Beschreibung der Terra-Datenbank und Betrachtungen zum praktischen Vorgehen geteilt. Die Praxis wiederum enthält zwei mögliche Vorgehensweisen:

1. die Abfrage mit SQL per Internetanbindung[3] unter http://marvin.schule.de/terra bzw. dieser Text unter http://sn.schule.de/~reimegym/terra und
2. die Abfrage in einer benutzerfreundlichen Datenbankumgebung.
   Da in meiner Schule StarOffice 5.2[4] verwendet wird, beziehen sich alle Angaben darauf. Würde man allerdings das Vorgehen mit anderen Datenbank-Managementsystemen vergleichen, wie z. B. MS Access, so würde man keine großen Unterschiede bemerken.

Der Entwurf eigener Datenbanken kann aus Zeitgründen leider nicht besprochen werden. Insbesondere wäre das Datenbank-Design mit den Normalisierungsregeln ein eigenes Thema.

Dieses Script liegt in folgenden Formaten vor:

- als terra.pdf – das ist die immer aktuelle Variante,
- als terraOL.pdf – der gleiche Text wie terra.pdf, jedoch ohne Lösungen und
- als HTML-Seite.
  Leider ist das Layout der HTML-Version nicht in allem stimmig. Es entspricht im Wesentlichen den Möglichkeiten, die StarOffice beim Export von HTML-Seiten bietet. Wer die Texte als Script verwenden möchte sollte sich unbedingt an die PDF-Version halten[5].
  Ein Vorteil der HTML-Version ist jedenfalls, dass sie zur Abfrage der Datenbank genutzt werden kann. In den Aufgabenstellungen gibt es entsprechende Eingabemöglichkeiten. Einige Besonderheiten werden dann ersichtlich und müssen unbedingt beachtet werden. Zum Beispiel sind Tabellennamen immer groß zu schreiben.

---

3 Der Vorteil für die Schulen besteht darin, dass es nicht notwendig wird ein DBMS zu installieren. Die Abfrage der Terra-Datenbank erfolgt über die allen zugänglichen Internetseiten auf dem Sächsischen Schulserver, die zum Zeitpunkt der Erstellung dieses Textes unter der Verwaltung von Herrn Fabianski standen. Dass die Datenbank ohne eigenes Zutun so immer auf einem aktuellen Stand bleiben kann, ist sicherlich ein weiterer Vorteil, der den Nachteil der eingeschränkten Nutzung aufwiegen sollte. Die Internet-Datenbank kann zwar abgefragt, jedoch nicht geändert werden.

4 StarOffice ist Freeware und kann zu schulischen Zwecken kostenlos eingesetzt werden. Außerdem bietet StarOffice die gleichen Möglichkeiten wie andere Officepakete bei gleichzeitiger Integration aller Komponenten in einer Benutzeroberfläche, läuft unter Linux und ist zu MS-Office kompatibel.

5 Vielleicht möchte sich jemand die Mühe machen und das Layout verbessern – das wäre schön.

2 Kurze Einführung in die Theorie der Datenbanken

Um das Abfragen von Datenbanken genauer zu verstehen, werden wir uns zuerst den Aufbau an einem Beispiel genauer ansehen. Doch bevor wir das tun, müssen wir ein paar wichtige Begriffe kennen. Eine Datenbank besteht aus

Daten -
: den integrierten Datensätzen und

Software -
: dem Datenbank Management System (database management system (DBMS))

2.1 Von der Realwelt zum Modell

Stellen wir uns vor, wir wollten eine Übersicht über die Kennzahlen von Städten, Ländern, Bergen, Seen und Meeren unserer Welt anfertigen. Um die Daten bereitzustellen, würden wir einen Atlas[6] nehmen und dort alle für uns notwendigen Angaben finden. Um die Informationen anschaulich darzustellen, gibt es zwei Möglichkeiten:

3. Wir kennen bereits eine konkrete Fragestellung, wie z. B. „Welche Städte haben mehr als 800 000 Einwohner und liegen an einem See?“. In diesem Fall können wir unsere Suche einschränken und eine entsprechende Tabelle erstellen.

4. Wir wissen nicht genau was gefragt sein wird. Wir möchten unsere Daten allerdings so aufbereiten, dass alle möglichen Fragen in kurzer Zeit beantwortet werden können. In diesem Fall sollten wir uns wohl am Atlas ein Beispiel nehmen und die Daten unter verschiedenen Stichworten in einzelne Tabellen einordnen. Alle Angaben von Städten kommen in die Tabelle „Stadt“, alle Daten zu Ländern in die Tabelle „Land“ usw.
   Dieses Vorgehen führt uns zum Begriff der **relationalen Datenbank**, denn zwischen den von uns angefertigten Tabellen bestehen Beziehungen (**Relationen**), wie zum Beispiel jede Stadt liegt in einem bestimmten Land und umgekehrt liegen verschiedene Städte in einem Land.

Anders als mit den Daten im Atlas können wir unter neuer Fragestellung unsere Tabellen durch Nutzen entsprechender Relationen schnell neu zusammenstellen. Weitere Vorteile sind im Folgenden aufgeführt.

2.2 Merkmale eines Datenbankbetriebssystems

- *Redundanzarme Speicherung (Integration)*
- *Trennung der Datenbeschreibung von den Anwendungsdaten (Katalog)*
- *Mehrnutzerzugriff*
- *Nutzerfreundlichkeit*
- *Gewährleistung der Datensicherheit*
  - *Gewährleistung der physischen Datenintegrität*
  - *Gewährleistung der operationalen Datenintegrität*
  - *Gewährleistung der semantischen Datenintegrität*
  - *Gewährleistung der Datensicherung*
- *akzeptables Antwort-Zeit-Verhalten*

[Keller99]

Die wichtigsten heute üblichen Datenbankmodelle sind:

- *das hierarchische Datenbankmodell*
- *das Netzwerk-Modell*
- *das relationale Datenbankmodell*

[Payer97] - mit Erklärungen zu den Datenbankmodellen

6 Zugegeben – wir nehmen einen idealen Atlas.

Um auf eine Vielzahl von Fragen antworten zu können, bietet sich also die Nutzung eines relationalen Datenbankmodells an. Es soll ein möglichst exaktes Abbild der realen Welt sein. Eine solche Miniwelt besteht aus gewissen **Objekte**n (sogenannten **Entities**) wie Stadt, Land, Kontinent usw. Zwischen diesen Objekten existieren Beziehungen, die bestimmte Abläufe oder Abhängigkeiten in der Miniwelt darstellen (z. B. welche Stadt in welchem Land liegt.) Um zu einem Datenmodell zu kommen, fasst man gleichartige Objekte und gleichartige Beziehungen (sogenannten **Relationships**) jeweils zu Klassen zusammen. Somit ergeben sich

- **Objekttypen** wie STADT, LAND, KONTINENT und
- **Beziehungstypen** wie UMFASST, LIEGT_AN.

K_NAME	L_ID	PROZENT
Europa	A	100
Asien	ADN	100
Asien	AFG	100
Europa	AL	100
Europa	AND	100
Amerika	AUB	100
Australien	AUS	100

Tabelle 1: Ausschnitt aus UMFASST

K_NAME	FLAECHE
Afrika	48,00
Amerika	39,34
Asien	30,00
Australien	13,20
Europa	10,50

Tabelle 2: KONTINENT

EINWOHNER	FLAECHE	HAUPTSTADT	L_NAME	L_ID	LT_ID
7862000	83845	Wien	Oesterreich	A	WIE
2360000	332968	Aden	Jemen	ADN	ADN
18630000	647497	Kabul	Afghanistan	AFG	AFG
3389000	28748	Tirana	Albanien	AL	AL
61000	468	Andorra_la_Vella	Andorra	AND	AND
81500	442	Saint_Johns	Antigua_und_Barbuda	AUB	AUB
16028400	7686420	Canberra	Australien	AUS	ACT

Tabelle 3: Auszug aus LAND

Im Gegensatz zu den nicht-relationalen Datenbank-Management-Systemen ist eine der Anforderungen an ein relationales Datenbank-Management-System (RDBMS), dass es die Daten in einfachen Tabellen abspeichert.

[Sauer98]

2.3 Relationale Datenbanksysteme

Eines der ersten Datenbanksysteme kam 1968 auf den Markt. Es war das hierarchische Datenbanksystem IMS von IBM. Sehr bald stellte man jedoch fest, dass die Möglichkeiten, die reale Welt in diesem DBMS abzubilden, nicht ausreichten.

[Sauer98]

2.3.1 Datenbank vs. Dateisystem

In den Anfängen der elektronischen Datenverarbeitung kannte man noch keine Datenbanksysteme. Jedes Programm speicherte seine Daten in einfachen Dateien. Diese Dateien wurden den Programmen fest zugeordnet; somit waren die gleichen Daten in der Regel mehrmals gespeichert (redundant). Falls ein Programm die Daten eines anderen Programms zur Weiterverarbeitung benötigte, wurden diese Daten in der Regel für diese spezielle Anwendung kopiert.
Damals wurden die einzelnen Programme noch sequentiell nacheinander abgearbeitet. D.h. erst, wenn das eine Programm komplett abgearbeitet war, wurde das nächste Programm gestartet. Doch im Laufe der Weiterentwicklung der Datenverarbeitungsanlagen wurden die Anforderungen an die Applikationen immer anspruchsvoller. Man erwartete von den Daten einen immer aktuelleren Stand. Somit musste z.B. der Lagerbestand eines Unternehmens sofort aktualisiert werden, wenn ein Auftrag ausgeführt wurde. Es konnte ja vorkommen, dass ein anderes Programm schon kurze Zeit später' den Lagerbestand wissen wollte, um einem Kunden Auskunft über die Liefermöglichkeiten eines Teils zu geben. Außerdem sollten diese Informationen direkt (online) vom Rechner abrufbar sein.
Gleichzeitig wurde die Datenstruktur immer komplexer. Große Unternehmen hatten plötzlich Tausende von Dateien zu verwalten. Es wurden ganze Aktenschränke mit Dateibeschreibungen manuell verwaltet, die sich nie auf dem aktuellen Stand befanden.
Dies war die Zeit, in der man über »zentralisierte Datenverwaltungsprogramme« nachdachte. Das waren Programme, die sowohl die Struktur (Satzaufbau) der Dateien als auch die dazugehörigen physischen Speichermedien und die Programme, die mit diesen Daten operierten, verwalteten. Somit waren die ersten Ansätze eines Datenbanksystems entstanden.
Im Laufe der Zeit wurde die Funktionalität dieser Datenverwaltungsprogramme immer weiter ausgebaut. Heute haben Datenbanksysteme nicht nur das »Wissen« über Zusammenhänge der Daten untereinander, sondern sie übernehmen auch Tätigkeiten wie Überwachung vor unberechtigtem Zugriff, Datensicherung, Kontrolle des gleichzeitigen Zugriffs mehrerer Programme usw.
Die Vorteile von Datenbanksystemen gegenüber herkömmlichen Dateisystemen sind vor allem in der einfacheren Programmierung zu sehen. Als die Programme noch dateiorientiert arbeiteten, musste in jedem einzelnen Programm die Datenstruktur aller benutzten Dateien definiert werden, die Programmlogik nicht nur vorwärts, sondern für den Fall des Programmabbruchs auch noch einmal rückwärts programmiert werden und falls das Programm beabsichtigte, die Daten zu ändern, musste sichergestellt sein, dass die Daten für andere Programme gesperrt waren.
Da beim Einsatz eines Datenbanksystems alle diese Programmteile an einer zentralen Stelle zur Verfügung gestellt werden, ergibt sich eine immense Vereinfachung der Anwendungsprogramme. Somit werden Programmfehler vermieden. Ein weiterer Vorteil eines Datenbanksystems ist, dass die Datenbeschreibungen (Datentyp und Zusammenhänge der Daten) an zentraler Stelle in einem sogenannten Data Dictionary verwaltet werden. Dadurch werden die Anwendungsprogramme unabhängiger von Änderungen der Datenstruktur.

[Sauer98]

2.3.2 Grundbegriffe des relationalen Datenmodells

Das relationale Modell besteht aus der Definition von Objekten, Operationen und Regeln.

2.3.3 Relationale Objekte

Domain

Wertebereich - eine Menge möglicher Werte, die ein Attribut annehmen kann, z.B. INTEGER, MONEY oder CHAR(32).

Attribut

Spalte - eine bezeichnete Eigenschaft. Attribute basieren immer auf einem Wertebereich.

Relation

Tabelle - Menge von (gleichartigen) Tupeln mit folgenden Eigenschaften

- keine doppelten Tupel in einer Relation
- die Reihenfolge, mit der die Tupel gespeichert sind, ist nicht definiert
- die Reihenfolge, mit der die Attribute gespeichert sind, ist nicht definiert
- die Attributwerte sind atomar (keine Mengen)

Degree

Ausdehnungsgrad der Tabelle

Tupel

Datensatz - ein Tupel umfasst mehrere Attribute mit je einem zugehörigen Wertebereich.

Ein Tupel fasst die interessierenden Eigenschaften eines Objektes, einer Person oder einer Beziehung zusammen.

Beziehung (relationship)

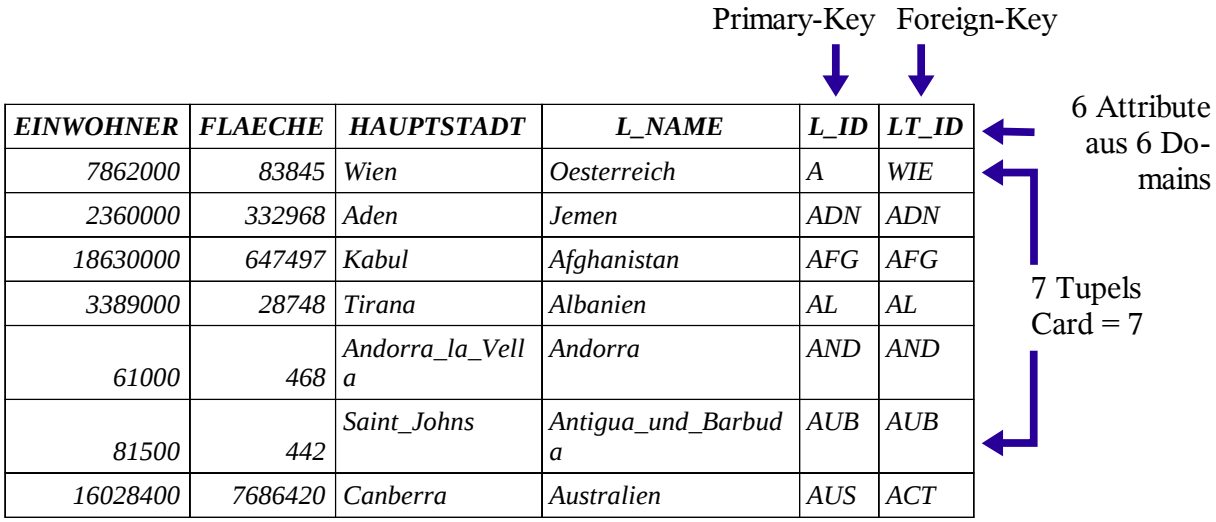
Beziehung zwischen Tupeln, wie zum Beispiel: eine Student besucht mehrere Vorlesungen. Beziehungen werden selbst wieder als Tupel dargestellt.

Primary-Key

eindeutiger Schlüssel

Foreign-Key

Fremdschlüssel – repräsentieren Beziehungen zwischen Tabellen



EINWOHNER	FLAECHE	HAUPTSTADT	L_NAME	L_ID	LT_ID
7862000	83845	Wien	Oesterreich	A	WIE
2360000	332968	Aden	Jemen	ADN	ADN
18630000	647497	Kabul	Afghanistan	AFG	AFG
3389000	28748	Tirana	Albanien	AL	AL
61000	468	Andorra_la_Vella	Andorra	AND	AND
81500	442	Saint_Johns	Antigua_und_Barbuda	AUB	AUB
16028400	7686420	Canberra	Australien	AUS	ACT

Abbildung 1: Auszug aus LAND

2.3.3.1 Relation, Degree, Attribut-Tupel

Alle Daten einer Datenbank werden in sogenannten Relationen gespeichert. Als ***Relation*** *(Tabelle) bezeichnet man eine logisch zusammenhängende Einheit von Informationen. Sie besteht aus einer festen Anzahl von Attributen (Spalten) und einer variablen Anzahl* ***Tupel*** *(Zeilen). Die Anzahl der Attribute einer Relation wird als* ***Degree*** *(Ausdehnungsgrad) bezeichnet. Eine Relation mit nur einem Attribut wird* ***unär,*** *eine mit zwei Attributen* ***binär*** *genannt. Die Relation LAND hat den Degree 6. Eine Relation mit n Attributen wird n-ary genannt.*

Mit ***Kardinalität*** *bezeichnet man die Anzahl der Tupel in einer Relation. Diese ist zeitabhängig und kann auch gleich Null sein, wenn die Relation leer ist.*

Eine Relation hat zusätzlich folgende Eigenschaften:

- *Keine doppelten Tupel*
  D.h. es gibt zu keinem Zeitpunkt zwei Tupel, deren Attributwerte den gleichen Inhalt haben.
- *Tupelreihenfolge*
  Die Reihenfolge, mit der die Tupel in einer Relation gespeichert sind, ist nicht definiert. Man darf sich also nie auf eine bestimmte Reihenfolge der Tupel in einer Relation verlassen!
- *Attributreihenfolge*
  Die Reihenfolge der Attribute in einer Relation ist nicht definiert. Es ist also nicht möglich, das n-te Attribut einer Relation anzusprechen. Man kann ein bestimmtes Attribut nur mit seinem Namen ansprechen.
- *Attributwerte sind atomar*
  Die Werte eines Attributes unterliegen einer Domäne. Da alle Elemente einer Domäne atomar sind, sind auch die Elemente eines Attributes atomar. Wenn eine Relation diese Eigenschaft hat, erfüllt sie die Bedingung der ersten Normalform (siehe Normalformen).

[Sauer98]

2.3.3.2 Primary-Key, Foreign-Key

Jede Relation besitzt genau einen ***Primary-Key (Primärschlüssel),*** *um ein Tupel der Relation eindeutig zu identifizieren. Er wird definiert, indem eine der eindeutigen (****unique****) Attributmengen zum »Primary-Key« erklärt oder ein neues*

Attribut speziell für diesen Zweck eingeführt wird (z.B. eine Nummer). Ein solcher Primary-Key darf keine Null-Werte enthalten. ... Der Primary-Key soll mindestens einen Zugriffspfad garantieren, mit dem exakt ein Tupel angesprochen werden kann. Mit Hilfe dieses Primary-Keys werden meist die Verknüpfungen (Joins) der Relationen in der Datenbank hergestellt.

[Sauer98]

Betrachten wir noch einmal die Tabellen 1, 2 und 3 von Seite 6. Wir können die Namen aller Länder, die nicht nur auf einem Kontinent liegen, heraussuchen, indem wir die Tabelle LAND mit der Tabelle Umfasst verbinden. Dazu nutzen wir den Primärschlüssel L_ID aus LAND und den Fremdschlüssel L_ID aus UMFASST. Der SQL-Befehl[7] für die Abfrage der beiden Tabellen lautet:

```
SELECT L_NAME
FROM UMFASST, LAND
WHERE UMFASST.L_ID = LAND.L_ID AND UMFASST.PROZENT < 100.
```

*In der Relation UMFASST wurde ein **Foreign-Key** definiert: L_ID. Er wird Foreign-Key genannt, da seine Wertemenge (Domäne) in einer anderen Relation als Primary-Key definiert ist. Beide Attribute (oder Attributmengen, wenn der Key aus mehr als einem Attribut besteht) müssen zu den gleichen Domäne gehören.*
Es ist wichtig, dass ein Datenbanksystem weiß, welche Attributmenge den Primary-Key definiert und welche Foreign-Keys existieren. Nur so kann das DBMS die referenzielle Integrität der Datenbank gewährleisten.
*Aus Performancegründen können zusätzlich beliebig viele Keys in einer Relation definiert werden. Diese werden dann **Alternate-Keys** oder auch **Secondary-Keys (Zweitschlüssel)** genannt.*

[Sauer98]

2.3.4 Relationale Operationen

Restriction
Zeilenselektion (siehe Abb. 2)

Projection
Spaltenselektion (siehe Abb. 2)

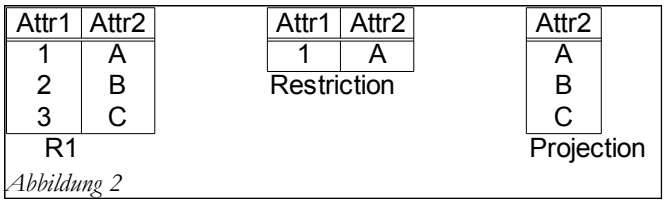


Abbildung 2

Product
kartesisches Produkt – jede Zeile der einen Tabelle wird mit jeder Zeile der anderen Tabelle verknüpft (siehe Abb. 3)

Union
Vereinigung (siehe Abb. 3)

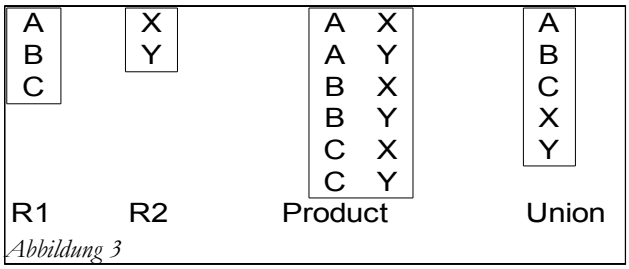


Abbildung 3

Intersection
Schnittmenge (siehe Abb. 4)

Difference
Differenz (siehe Abb. 4)

Join
Verbindung (siehe Abb. 5)

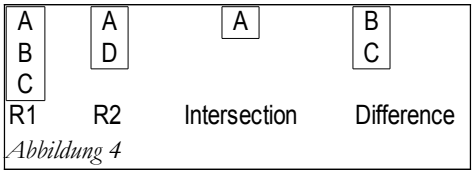


Abbildung 4

7 Auf die Verwendung von SQL-Befehlen werden wir später noch ausführlicher eingehen.

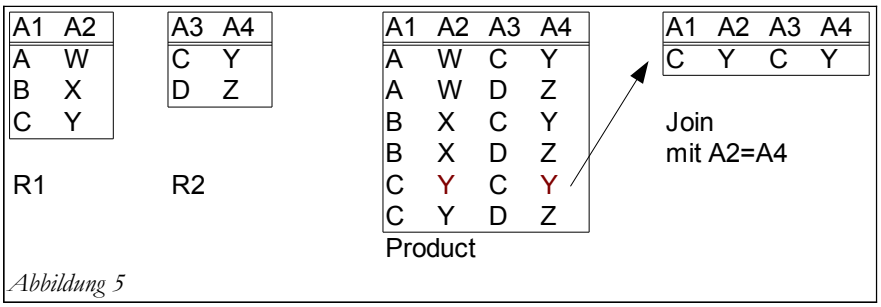

A1 A2
A W
B X
C Y
R1
A3 A4
C Y
D Z
R2
A1 A2 A3 A4
A W C Y
A W D Z
B X C Y
B X D Z
C Y C Y
C Y D Z
Product
A1 A2 A3 A4
C Y C Y
Join
mit A2=A4

Abbildung 5

3 Terra-Datenbank

Die relationale Datenbank TERRA enthält Informationen über die politische Geographie und die Topographie der Erde[8].

3.1 Formaler Aufbau

TERRA =
{LAND, LANDTEIL, STADT, KONTINENT, BERG, EBENE, SEE, MEER, FLUSS, INSEL, WUESTE, ORGANISATION, IST_BENACHBART_ZU, IST_MITGLIED_VON, HAT_SITZ_IN, GEHT_UEBER_IN, LIEGT_AN, UMFASST, GEO_FLUSS, GEO_SEE, GEO_MEER, GEO_INSEL, GEO_WUESTE, GEO_EBENE, GEO_BERG}

LAND
(L_NAME, L_ID, EINWOHNER, ZUWACHS, FLAECHE, BSP, HAUPTSTADT, LT_ID, SYSTEM, REGCHEF)
LANDTEIL
(LT_NAME, LT_ID, L_ID, EINWOHNER, LAGE, HAUPTSTADT)
STADT
(ST_NAME, L_ID, LT_ID, EINWOHNER, BREITE, LAENGE)
KONTINENT
(K_NAME, FLAECHE)
BERG
(B_NAME, GEBIRGE, HOEHE, JAHR, LAENGE, BREITE)
EBENE
(E_NAME, HOEHE, FLAECHE)
SEE
(S_NAME, TIEFE, FLAECHE)
MEER
(M_NAME, TIEFE)
FLUSS
(F_NAME, FLUSS, SEE, MEER, LAENGE, LAENGEU, BREITEU, LAENGEM, BREITEM)
INSEL
(I_NAME, INSELGRUPPE, FLAECHE, LAENGE, BREITE)
WUESTE
(W_NAME, FLAECHE, WUESTENART)
ORGANISATION
(O_NAME, ABKUERZUNG, ART)
IST_BENACHBART_ZU
(LAND1, LAND2)
IST_MITGLIED_VON
(L_ID, ABKUERZUNG, ART)
HAT_SITZ_IN
(ST_NAME, LT_ID, L_ID, ABKUERZUNG)
GEHT_UEBER_IN
(MEER1, MEER2)
LIEGT_AN
(ST_NAME, LT_ID, L_ID, F_NAME, S_NAME, M_NAME)
UMFASST
(L_ID, K_NAME, PROZENT)
GEO_FLUSS
(LT_ID, L_ID, F_NAME)
GEO_SEE
(LT_ID, L_ID, S_NAME)
GEO_MEER
(LT_ID, L_ID, M_NAME)
GEO_INSEL
(LT_ID, L_ID, I_NAME)
GEO_WUESTE
(LT_ID, L_ID, W_NAME)
GEO_EBENE
(LT_ID, L_ID, E_NAME)
GEO_BERG
(LT_ID, L_ID, B_NAME).

Die Relation LIEGT_AN enthält keine Schlüssel, da das Tripel (ST_NAME, LT_ID, L_ID) mehrfach vorkommen kann, z.B. wenn mehrere Flüsse durch ein und dieselbe Stadt fließen.
Andererseits können bezüglich F_NAME, S_NAME oder M_NAME Nullwerte auftreten.

8 Unterstrichene Attribute der Tabellen sind Schlüssel der Relation.

Die verwendeten Tabellen und Merkmalsnamen sind zum größten Teil selbsterklärend. Insbesondere bedeuten: SP - Bruttosozialprodukt, LAENGEU, BREITEU, LAENGEM, BREITEM - geographische Länge und Breite des Ursprunges bzw. der Mündung eines Flusses.
Beachten Sie, dass V(LAND1) = V(LAND2) = V(L_ID) und V(MEER1) = V(MEER2) = V(M_NAME). Als Datentypen der verwendeten Merkmale sind je nach Semantik TEXT oder ZAHL als gegeben zu betrachten.
Für die folgenden Anfragen sind SQL-Anweisungen zu formulieren und mit der Praktikumsdatenbank TERRA zu testen.

[Keller99]

3.2 Aufbau der Datenbank

Die Terra-Datenbank besteht aus rund 20 Tabellen, die in zwei Gruppen unterteilt werden können. Einerseits gibt es Tabellen wie LAND, STADT, BERG usw. Diese enthalten eigenständige, von anderen unterscheidbare Objekte (**Entities** ↑Objekttypen Kapitel 2.2). Anderseits stellen Tabellen wie GEO_BERG, IST_MITGLIED_VON und LIEGT_AN Beziehungen (**Relationships** ↑Beziehungstypen Kapitel 2.2) zwischen zwei oder mehreren Entities dar[9].

In Abbildung 6 wird anhand des Attributes L_ID einige der möglichen Verbindungen aufgezeigt. L_ID kommt in jeder der genannten Tabellen vor. In LAND ist L_ID der Primärschlüssel. Das heißt, in LAND kann jeder Datensatz anhand L_ID eindeutig identifiziert werden. In anderen Tabellen kann ein Wert von L_ID mehrfach vorkommen. Liegen z. B. mehrere Städte in einem Land, so wird L_ID in STADT auch mehrfach vorkommen.

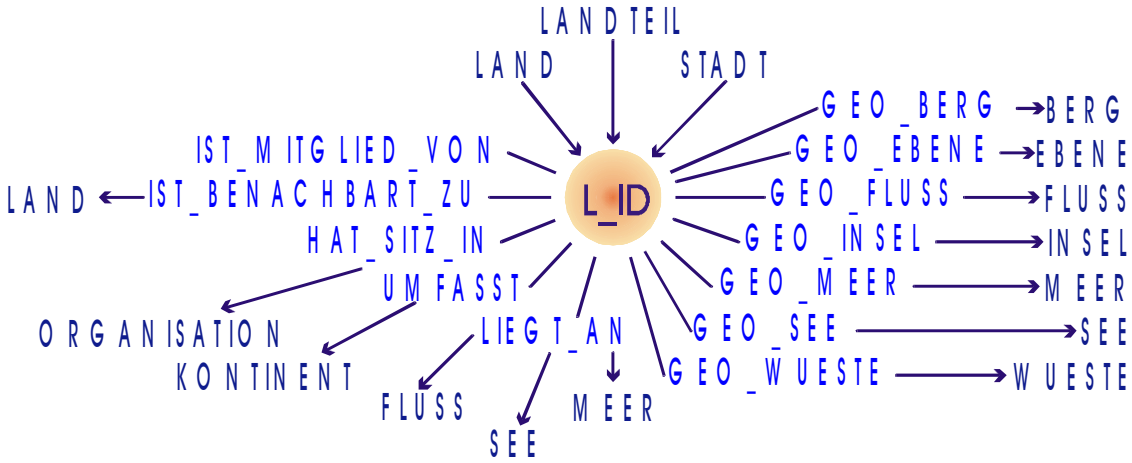


Abbildung 6: Verbindung von Tabellen mittels L_ID

Möchte man Angaben zu den Inseln eines Landes haben, so müssen die Entities LAND und INSEL zusammengeführt werden. Da sie aber keine gemeinsamen Attribute haben, ist das nur über die Relationships z. B. GEO_INSEL möglich. In GEO_INSEL kommt sowohl L_ID als auch I_NAME vor. Zum Beispiel kann die Zuordnung welche Insel zu welchem Land gehört, nur so erfolgen:

```
select l.l_name, i.i_name
 from Land l, Insel i, Geo_Insel g
 where l.l_id=g.l_id and g.i_name=i.i_name
```

Weiterhin ist es kein Problem, mehrere Tabellen miteinander zu verknüpfen.
Ein häufiger Fehler ist es, Tabellen nicht richtig oder gar nicht in Beziehung zueinander zu

9 Die Trennung in Entity- und Relationship-Tabellen ist stark vereinfacht. Normalerweise findet man in fast allen Tabellen Relationships und in vielen Tabellen kommen Entities vor.
Die Tabelle Stadt enthält zum Beispiel die Beziehung „Welche Stadt in welchem Land liegt.“, indem sie die Attribute ST_NAME und L_ID enthält. Trotzdem bezeichne ich sie hier als Objekt-Tabelle, denn sie sammelt alle Daten zum Objekt „Stadt“.

setzen.

```
select l.l_name, i.i_name
 from Land l, Insel i, Geo_Insel g
```

Dabei entsteht das Kreuzprodukt der Tabellen (↑Product Kapitel 2.3.4). Dies würde 190 · 175 · 174 = 5.785.500 Einträge besitzen. Selbst eine schnelle Datenbank benötigt einige Minuten, wenn nicht gar Stunden, um alle Datensätze zusammenzustellen. Bei Nutzung der Internet-Datenbank käme noch der Aufwand zur Übertragung der Datenmenge hinzu.

3.3 Grafische Darstellungen des Entity-Relationship-Modell

Eine Verdeutlichung für die Kombinationsmöglichkeiten verschiedener Tabellen können solche Zeichnungen geben.

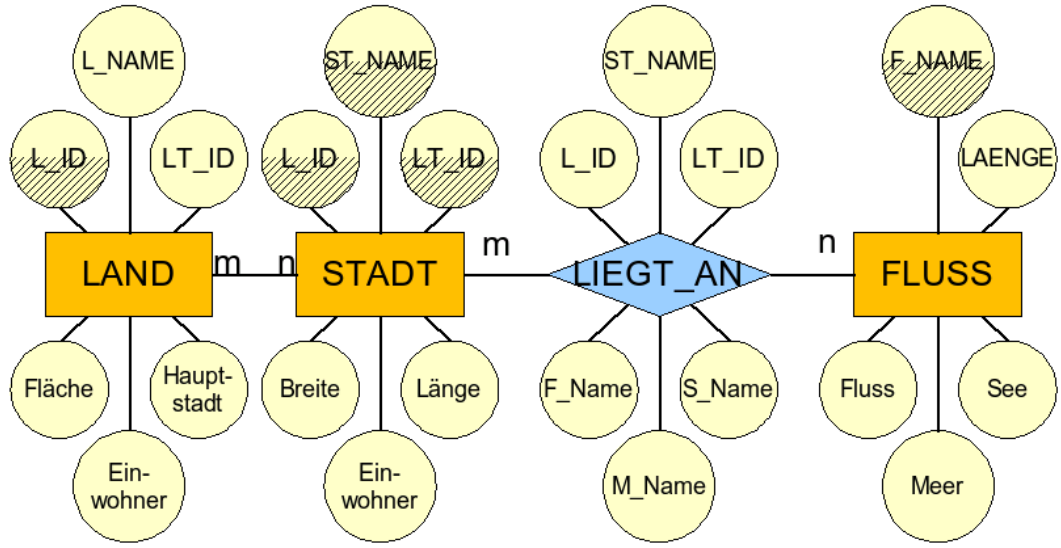


Abbildung 7: ER-Schema

Im Internet finden Sie umfangreichere Zeichnungen:

- http://www.dbis.informatik.hu-berlin.de/fileadmin/lectures/WS2005_06/DBS1_Praktikum/Informationsblatt2.pdf (Stand: 22.11.06) und
- http://www.informatik.uni-jena.de/dbis/lehre/ss2006/dbprak/projekt-0.pdf (Seite 9; Stand: 22.11.06)

wobei allerdings anzumerken ist, dass nicht alle Darstellungen zu 100% auf die an der TU-Dresden vorzufindenden Datenbank anwendbar sind. Sowohl die Benennung als auch der Inhalt der Tabellen unterscheiden sich zuweilen.

Die folgenden Aufgaben stimmen nicht in jedem Fall mit den Aufgaben im Internet überein. Die in eckige Klammern gesetzten Zahlen geben die Anzahl der Zeilen in der Ergebnistabelle an. Auch sie weichen von denen im Internet ab, den der Datenbestand wird ab und zu überarbeitet. Dann können sich auch die Ergebnismengen ändern.

3.4 Aufgaben zur Terra-Datenbank

1. *Gesucht sind alle Länder mit L_NAME, L_ID, EINWOHNER. [190][10]*
2. *Gesucht ist die Langform des Landes mit L_ID = 'D' und die entsprechende Einwohnerzahl. [1]*
3. *Gesucht sind alle Inseln (Name) der Inselgruppe der Philippinen. [9]*
4. *Ermitteln Sie die Namen aller Kontinente. [5]*
5. *Gesucht ist die Einwohnerzahl der deutschsprachigen Länder (D, A, CH). [1]*
6. *Gesucht ist die Summe der Stadteinwohner je Land. [185]*
7. *Gesucht sind alle Berge der Alpen (Beachten Sie, dass die Alpen in Teilgebirge aufgegliedert sind). [4]*
8. *Gesucht sind alle Hochgebirge (Gebirge mit Bergen über 3000 m Höhe). [28]*
9. *Gesucht sind alle Flüsse (Name), die in die Ostsee oder Nordsee münden und über 1000 km lang sind. [4]*
10. *Gesucht sind alle Wüsten (Name) der Art "Sandwueste" mit einer Fläche größer als 25000 km². [9]*
11. *Geben Sie von jedem Land in Afrika den Namen, die Anzahl der Einwohner, die Fläche an. [53]*
12. *Geben Sie von jedem Land in Afrika den Namen, die Anzahl der Einwohner, die Fläche sowie den prozentualen Flächenanteil am Kontinent an. [53]*
13. *Welche Länder (L_ID) haben mehr als 5 Millionenstädte? [5]*
14. *Geben Sie für alle Millionenstädte, die in den Tropen liegen, die Namen und ihre Koordinaten an (Die Tropen liegen zwischen 23,27 Grad nördlicher und 23,27 Grad südlicher Breite). [28]*
15. *Gesucht sind Namen und Hauptstädte aller Länder, die nicht Mitglied der UNO sind.*
16. *Ermitteln Sie die Namen aller Flüsse, die sowohl durch Deutschland als auch durch Österreich fließen. Es kann davon ausgegangen werden, dass die Landesidentifikatoren für Deutschland (D) und für Österreich (A) bekannt sind.*
17. *Geben Sie die Namen, die jeweilige Länderkennung und die Zahl der Einwohner für alle Länder an, die mehr als 45 Millionen Einwohner haben. Ordnen Sie dabei die Länder in absteigender Reihenfolge nach der Einwohnerzahl. [19]*
18. *Gesucht ist der Anteil der Meere an der Erdoberfläche (Angabe in Prozent). [1]*
    *Hinweise: Oberflächenformel: $O = 4\pi r^2$, Erdradius gleich 6370 km, Flaeche in Tabelle Kontinent liegt normiert vor (Faktor 1 Mill.), damit 6.37 * 6.37 anstelle 6370 * 6370.*
19. *Gesucht sind alle Inselgruppen sowie die Anzahl der zugehörigen Inseln. [35]*
20. *Gesucht sind alle Flüsse mit mehr als zwei an ihnen liegenden Städten, sortiert nach dieser Anzahl. [21]*
21. *Gesucht sind alle Städte, in denen mehrere Organisationen ihren Sitz haben. [5]*
22. *Ermitteln Sie die Namen aller Flüsse, die sowohl durch Deutschland als auch durch Österreich fließen. [1]*
23. *Gesucht ist die größte Insel der Welt (Name und Fläche der Insel). [1]*
24. *Geben Sie von jedem Land in Afrika den Namen, die Anzahl der Einwohner, die Fläche, das Bruttosozialprodukt sowie den prozentualen Flächenanteil auf dem Kontinent (1. Teilaufgabe) [56] bzw. am Kontinent (2. Teilaufgabe) [52] an.*

[Keller99]

10 Angegeben wird der Grad der resultierenden Relation. Die Lösungen haben Stand Juni 2011. Sollten Aktualisierungen an der Datenbank durchgeführt werden, so kann sich das auch auf die Ergebnismengen auswirken (↑Lösungsvorschläge zu den Aufgaben).

4 Praxis SQL

Der Begriff »**S**tructured **Q**uery **L**anguage« bezeichnet eine strukturierte **Abfragesprache** für relationale Datenbanken. Sie wurde in den 70ern von IBM entwickelt, um die Selektion und Modifikation von Daten in einer Datenbank zu beschreiben. Vielleicht aufgrund der Einfachheit der Sprache setzte sich SQL in der Datenbank-Szene schnell und erfolgreich durch. Inzwischen gibt es wohl keinen namhaften Datenbankanbieter, der nicht auf die eine oder andere Art diesen Standard unterstützt.

Die Spezifikationen von SQL wurden von ANSI und ISO im ANSI-SQL-Standard festgehalten. Sie finden im Internet die aktuelle Spezifikation auf verschiedenen Servern, z.B. unter SQL Standards und weitere Links im Inhaltsverzeichnis.

4.1 SQL-Befehlsübersicht

Anweisungen zur Veränderung der logischen Struktur der Datenbank oder der Tabellen der Datenbank sind

4.1.1 Befehle zur Datendefinition

CREATE TABLE:
: neue Basistabelle erstellen,

CREATE VIEW:
: Definieren einer logischen Tabelle,

GRANT:
: Benutzerberechtigungen vergeben.

Die vier wichtigsten Kommandos, auch „queries“ genannt, die den Zugriff auf die Daten in einer Datenbanktabelle ermöglichen, sind

4.1.2 Befehle zur Datenmanipulation

SELECT:
: für die Suche nach Datensätzen, auf welche die angegebenen Suchkriterien zutreffen,

INSERT,:
: für das Einfügen neuer Datensätze in eine Tabelle,

DELETE:
: für das Löschen existierender Datensätze aus einer Tabelle und

UPDATE:
: um Änderungen an bestehenden Datensätzen durchführen zu können.

Da wir SQL im Zusammenhang mit der Terra-Datenbank im Internet verwenden wollen, sind unsere Einflussmöglichkeiten beschränkt. So ist es uns nicht gestattet, Datensätze in Tabellen einzufügen, zu verändern oder zu löschen. Ebensowenig ist erlaubt Tabellen zu erstellen oder zu löschen. Was von den SQL-Befehlen übrig bleibt ist die SELECT-Anweisung. Diese Anweisung ist allerdings so umfangreich und mächtig, dass anhand dessen genug über den Aufbau und die Abfrage von Datenbanken lernen werden.

4.2 SELECT-Befehl

Der wohl wichtigste und komplexeste SQL-Befehl verändert keine Daten. Mit ihm werden Daten aus einer oder mehreren Tabellen lesend verarbeitet. Das Ergebnis des SELECT sind Daten, die wiederum wie eine Tabelle strukturiert sind.

Wir wollen verschiedene Beispiele betrachten, um die Verwendung der Anweisung zu untersuchen.

4.2.1 Spaltenselektion (projection)

Beispiel 1

Frage: Welche Länder sind in der Terra Datenbank eingetragen?
SQL: SELECT l_name FROM LAND
Resultat: *[190]*

L_NAME
Oesterreich
Jemen
Afghanistan
Albanien
Andorra
⋮

Beispiel 2

Frage: Was ist in der Tabelle LAND überhaupt eingetragen?
SQL: SELECT * FROM LAND
Resultat: *[190]*

EINWOHNER	FLAECHE	HAUPTSTADT	L_ID	L_NAME	LT_ID
7862000,00	83845,00	Wien	A	Oesterreich	WIE
2360000,00	332968,00	Aden	ADN	Jemen	ADN
18630000,00	647497,00	Kabul	AFG	Afghanistan	AFG
3389000,00	28748,00	Tirana	AL	Albanien	AL
⋮	⋮	⋮	⋮	⋮	⋮

4.2.2 Filtern

Beispiel 3

Frage: Von welchem Land ist Bridgetown die Hauptstadt?
SQL: SELECT l_name FROM LAND WHERE hauptstadt = 'bridgetown'
Resultat: *[1]*

L_NAME
Barbados

Beispiel 4

Frage: Wie heißen die Hauptstädte der östereichischen Bundesländer?
SQL: SELECT lt_name, hauptstadt FROM LANDTEIL WHERE l_id = 'a'
Resultat: *[9]*

LT_NAME	*HAUPTSTADT*
Burgenland	*Eisenstadt*
Kaernten	*Klagenfurt*
Niederoesterr.	*St.Poelten*
Oberoesterr.	*Linz*
Salzburg	*Salzburg*
Steiermark	*Graz*
Tirol	*Innsbruck*
Voralberg	*Bregenz*
Wien	*Wien*

4.2.3 Syntax I

```
SELECT [ALL | DISTINCT] spaltenausdruck | *
  FROM tabellenausdruck
  [WHERE bedingungen]
```

DISTINCT unterdrückt doppelte Zeilen im Ergebnis. ALL (die Standardeinstellung) bewirkt, dass alle, also auch gleiche Zeilen, im Ergebnis erscheinen.

4.2.3.1 Spaltenausdruck

Mit SPALTENAUSDRUCK wird angegeben, welche Spalten in die Ergebnisrelation (Tabelle) aufgenommen werden. Mit * werden alle mögliche Spalten einer Tabelle ausgewählt. Die Spaltenüberschrift kann durch Eingabe eines Spaltenbezeichners geändert werden.

Beispiel 5

SELECT lt_name AS Landteil, hauptstadt FROM LANDTEIL WHERE l_id = 'a'

(Vergleiche mit Beispiel 4)

Landteil	*HAUPTSTADT*
Burgenland	*Eisenstadt*
...	...

Standard-SQL erlaubt die Benutzung von fünf Funktionen im SPALTENAUSDRUCK:

Funktion	*Bedeutung*
COUNT	*Anzahl der Werte in der Spalte*
SUM	*Summe der Werte in der Spalte*
AVG	*Mittelwert der Spalte*
MAX	*größter Wert in der Spalte*
MIN	*kleinster Wert in der Spalte*

Beispiel 6

Frage: Wie viele Städte sind in Spanien eingetragen?
SQL: SELECT COUNT(st_name) FROM STADT WHERE l_id = 'e'
Resultat: *[1]*

Expr1000
4

Beispiel 7

Frage: Wie viele Einwohner hat die Stadt Deutschlands mit den wenigsten Einwohner?
SQL: SELECT MIN(einwohner) AS MINIMUM FROM STADT WHERE l_id = 'd'
Resultat: *[1]*

Minimum
21530

Und als Kombination der beschrieben Möglichkeiten:

Beispiel 8

Frage: Welche deutsche Stadt hat die wenigsten Einwohner?
SQL: SELECT st_name AS Stadt
FROM STADT
WHERE einwohner = (SELECT MIN(einwohner) FROM STADT WHERE l_id = 'd'))
Resultat: *[1]*

Stadt
Dillingen

oder alle Angaben

SQL: SELECT *
FROM STADT
WHERE einwohner = (SELECT MIN(einwohner) FROM STADT WHERE l_id = 'd'))
Resultat: *[1]*

ST_NAME	***EINWOHNER***	***L_ID***	***LT_ID***	***LAENGE***	***BREITE***
Dillingen	*21530,00*	*D*	*SAR*		

4.2.3.2 Tabellenausdruck

Mit TABELLENAUSDRUCK wird festgelegt, in welchen Tabellen nach den Daten gesucht wird.

Bevor wir mit der formalen Syntax des Tabellenausdrucks fortfahren, hier zunächst eine Übersicht seiner Einzelteile:

FROM	definiert die Eingangstabellen
WHERE	selektiert aufgrund einer Bedingung die Zeilen der Eingangstabellen
GROUP BY	gruppiert Zeilen auf der Basis gleicher Spaltenwerte
HAVING	selektiert nur Gruppen im GROUP-BY-Teil laut einer Bedingung

Beispiel 9 – Join

Frage: Welche Länder (Namen, Einwohnerzahl) liegen nicht nur auf einem Kontinent?
SQL:
```
SELECT DISTINCT l.l_name, l.einwohner
FROM LAND l, UMFASST u
WHERE u.l_id = l.l_id AND u.prozent <> 100
```
Resultat: *[3]*

L_NAME	*EINWOHNER*
Aegypten	*42000000*
Russland	*148700000*
Tuerkei	*59597000*

4.2.3.3 Suchbedingungen

werden in der WHERE- und HAVING-Klausel benötigt, um bestimmte Spalten zu selektieren bzw. zu gruppieren. Eine solche Suchbedingung besteht meist aus einer Reihung von Prädikaten, die mit AND oder OR verbunden werden. Durch Klammerung wird eine bestimmte Abarbeitungsreihenfolge erzwungen.

Man unterscheidet sieben Arten von Prädikaten:

- Vergleichsprädikat (=)
- Intervallprädikat (BETWEEN):

Beispiel 10

Frage: Welche deutschen Kleinstädte sind eingetragen?
SQL:
```
SELECT st_name, einwohner
FROM STADT
WHERE  l_id = 'd' AND  einwohner BETWEEN 50000 AND 100000
```
Resultat: *[28]*

ST_NAME	*EINWOHNER*
Aschaffenburg	*66152*
Bernburg	*71690*
Brandenburg	*84493*
...	

- Ähnlichkeitsprädikat (LIKE):

Beispiel 11

Frage: Welche Städte besitzen „furt“ im Namen?
SQL:
```
SELECT st_name
FROM STADT
WHERE  st_name LIKE '%furt%'
```
Resultat: *[5]*

st_name
Erfurt

Syntax
expr [Not] Like *pattern*
Beschreibung:
Der Like-Operator gibt True zurück, wenn der angegebene Zeichenkettenausdruck (expr) mit dem angegebenen Muster aus der Zeichenkette (pattern) übereinstimmt. Mit dem Not-Operator kann das Ergebnis des Like-Operators umgekehrt werden.

st_name
Frankfurt/ M
Frankfurt/O
Klagenfurt
Schweinfurt

- Test auf leere Einträge (NULL)
  Welche Städte haben keine Einwohner? [25]
  SELECT st_name FROM STADT WHERE einwohner IS NULL
- IN-Prädikat
  Gesucht sind alle Städte in Deutschland und Österreich. [122]
  SELECT st_name FROM STADT WHERE l_id IN ('d', 'a')

Während die Wirkung der meisten Prädikatarten selbsterklärend ist, bedürfen das ALL-oder-ANY-Prädikat und das EXISTS-Prädikat einer zusätzlichen Erklärung:

- ALL- oder-ANY-Prädikat:
  Das **ALL-oder-ANY-Prädikat** läßt sich am besten an einem Beispiel verdeutlichen:

Beispiel 12

Frage: Welche Länder haben größere Städte als Japan?
SQL:
```
SELECT DISTINCT l.l_name, l.l_id
FROM STADT s, LAND l
WHERE s.l_id = l.l_id
  AND s.einwohner > ALL ( SELECT einwohner FROM STADT WHERE l_id = 'j' )
```

oder einfacher:

```
SELECT DISTINCT l_id
FROM STADT
WHERE einwohner > ALL ( SELECT einwohner FROM STADT WHERE l_id = 'j' )
```

Resultat[11]: *[2]*

L_NAME
Mexiko
Vereinigte_Staaten_von_Amerika

Zunächst wird die Ergebnismenge des Subselects (SELECT einwohner FROM STADT WHERE l_id = 'j') – die Einwohnerzahlen der japanischen Städte – errechnet. Danach werden im äußeren SELECT die Länderkennungen der Länder selektiert, deren Städte Einwohnerzahlen größer als der japanischer Städte haben.

Hinweis: Wer mit MySQL arbeitet[12], hat bisher nicht die Möglichkeit der Verwendung von Subselects. Die Online-Abfrage der Datenbank in www.sn.schule.de kann über Kreuztabellen abfragen. Unter der oben genannten Aufgabenstellung könnte das so aussehen:

11 Leider gibt es inzwischen verschieden aktuelle Versionen der Datenbank. Das lässt sich auch an dieser Aufgabe erkennen. Zum Beispiel ergibt die einfachere Version der Abfrage 3 Zeilen, aber China ist nicht in der Land-Tabelle und wird deshalb nicht mit angezeigt.

12 zumindest bis MySQL Version 4.0.12

Beispiel 13

Frage: Welche Länder haben größere Städte als Japan?
MySQL:
```
SELECT l.l_name, s.l_id, t.l_id, max(t.einwohner)
FROM STADT s, STADT t, LAND l
WHERE s.l_id<>t.l_id and t.l_id=l.l_id
GROUP BY s.l_id, t.l_id
HAVING s.l_id='j' and max(s.einwohner)<max(t.einwohner)
```
Resultat: *[7] bzw. [9]*

l.l_name	*s.l_id*	*t.l_id*	*max(t.einwohner)*
Brasilien	*J*	*BR*	*10099100*
Indien	*J*	*IND*	*15000000*
Mexiko	*J*	*MEX*	*9815785*
Indonesien	*J*	*RI*	*9341400*
Suedkorea	*J*	*ROK*	*10231217*
Russland	*J*	*RUS*	*8600000*
Vereinigte_Staaten_von_Amerika	*J*	*USA*	*18200000*
	J	*RA*	*13076300*
	J	*VRC*	*14640000*

- EXISTS-Prädikat

Beispiel 14

Frage: Welche deutschen Städte liegen an einem Fluss? Wieviele Einwohner haben diese Städte?
SQL:
```
SELECT st_name, einwohner
FROM STADT
WHERE EXISTS ( SELECT * FROM LIEGT_AN WHERE NOT f_name ISNULL ) AND
l_id = 'd'
```
Resultat: *[113]*

ST_NAME	*EINWOHNER*
Aachen	*247792*
Aschaffenburg	*66152*
Augsburg	*258457*
...	...
Zittau	*29373*

4.2.4 Übung

25. Wie viele Städte liegen in Bayern? [21]
26. Welche Städte haben mehr als 850.000 Einwohner? [166]
27. Welche Städte haben ein „tz“ im Namen? [3]
28. Welche Städte haben keine Einwohner? [25]

29. Welche Länder haben Hauptstädte, die mit „m“ beginnen? [21]
30. Welche Länder haben mehr 1 Mio., aber weniger als als 10 Mio. Einwohner und eine Hauptstädte, die mit „m“ beginnt. [7]

4.2.5 Sortieren

<< Dieser Absatz ist noch unvollständig. >>

4.2.6 Gruppieren

<< Dieser Absatz ist noch unvollständig. >>

4.2.7 Syntax II

Die umfassende Beschreibung der SELECT-Anweisung in diesem Kapitel entstammt der Online-Hilfe zu [Freeze] und der Dokumentation zu [MySQL]

4.2.7.1 Einfache Form

```
SELECT [DISTINCT | DISTINCTROW | ALL] select_expression,...
 [FROM table_references
 [WHERE where_definition]
 [GROUP BY {unsigned_integer|col_name|formula} [ASC|DESC], ...]
 [HAVING where_definition]
 [ORDER BY {unsigned_integer|col_name|formula} [ASC|DESC] ,...]
 [LIMIT [offset,] rows]
```

4.2.7.2 Komplexe Form

```
{ simpleselect | ( complexselect ) | valueclause }
[ Union [ All ] | Except [ All ] | Intersect [ All ]
  { simpleselect | ( complexselect ) | valueclause } ]
```

4.2.7.3 Beschreibung

*Die **Select**-Anweisung verfügt über zwei Varianten: einfach und komplex. Die einfache **Select**-Anweisung ist am gebräuchlichsten und wird bei der Durchführung einzelner Abfragen verwendet. Die komplexe **Select**-Anweisung besteht aus mehreren **Select**-und **Value**-Anweisungen mit kombiniertem Resultat.*
*Die einfache Select-Anweisung gewinnt eine oder mehrere Reihen mit Informationen aus der Datenbank. Das Schlüsselwort **All** bedeutet, dass alle qualifizierenden Reihen aus der Datenbank ausgewählt werden. Mit dem Schlüsselwort **Distinct** werden keine doppelten Reihen ausgewählt. Ohne Angabe wird **All** angenommen.*
Es folgt eine Liste von Spaltenausdrücken (columnexpr), die den zurückgegebenen Spalten entsprechen. columnexpr hat die folgende Syntax:

```
{ [schemaname.]tablename.columnname |
tabledesignator.columnname | columnname | expr }
```

*Dabei bezieht sich schemaname auf den Namen des Schemas, das sich im Besitz der Tabelle oder Sicht befindet, die den angegebenen Spaltennamen (columnname) enthält. Fehlt diese Angabe, wird Ihr Vorgabeschema als Vorgabe eingesetzt. tablename bezieht sich auf die Tabelle oder Sicht, die die Spalte enthält. Wenn diese Angabe fehlt, ermittelt das Datenbanksystem den passenden Namen der Tabelle oder Sicht auf Grundlage der in der **From**-Klausel enthaltenen Informationen.*
*correlation stammt ebenfalls aus der **From**-Klausel und bietet eine alternative Referenzierungsmöglichkeit für die Tabelle oder Sicht. Wenn nur columnname angegeben wird, dann wird die zugehörige Tabelle oder Sicht aus der **From**-Klausel bestimmt.*
Sie können für expr auch einen Ausdruck angeben, der einen einzelnen Wert zurückgibt. Dabei kann es sich sowohl um eine einfache Konstante oder auch einen komplexen Ausdruck handeln, der mehrere verschiedene Spalten in die Berechnungen einbezieht.
Schließlich können Sie immer dann ein Sternchen (*) verwenden, wenn die Angabe eines Spaltennamens gefragt ist. Dieses spezielle Zeichen bedeutet, dass alle Spalten zurückgeliefert werden, die zur Tabelle oder Sicht gehören.

Tip

Auch wenn die Verwendung eines Sternchens zur Gewinnung aller Spalten einer Tabelle bei der Arbeit mit interaktiven Abfragen sehr nützlich sein kann, sollten Sie innerhalb von Programmen alle gewünschten Spalten aufführen.

Die **Into***-Klausel enthält eine Folge von Host-Variablen, die die Informationen einer einzelnen Reihe aufnehmen. Jeder Host-Variablen muss ein Doppelpunkt vorangehen. Diese Klausel ist nur verfügbar, wenn SQL-Anweisungen aus einem Programm heraus mit der Anweisung* **Exec SQL** *ausgeführt werden.*

Die **From***-Klausel enthält eine Liste von Tabellen und Sichten, die für die Abfrage benötigt werden, und entspricht der folgenden Syntax:*

```
[schemaname.]tablename | tabledesignator
```

Dabei bezieht sich schemaname auf den Namen des Schemas, das die Tabelle oder Sicht besitzt. Wenn schemaname nicht angegeben wird, wird das Vorgabeschema verwendet. tablename bezieht sich auf den Namen der Tabelle oder Sicht, die bei der Erfüllung der Abfrage verwendet wird. tabledesignator wird verwendet, wenn ein alternativer Name für die Tabelle in der Abfrage zur Verfügung gestellt werden soll.

Tip

Setzen Sie tabledesignator in komplexen, wie zum Beispiel verschachtelten Abfragen ein, in denen auf unterschiedliche Weise auf dieselbe Tabelle Bezug genommen wird.

Die **Where***-Klausel enthält einen logischen Ausdruck, der den Wert* **True** *annehmen muss, damit die Reihe zurückgeliefert wird.*

Die Klausel **Order By** *enthält eine Folge von Spaltennamen oder Zahlen, die zur Sortierung der Abfrageergebnisse verwendet wird. Sie können auch angeben, ob die Informationen der entsprechenden Spalten in aufsteigender (***Asc***) oder absteigender (***Desc***) Reihenfolge sortiert werden sollen. Die Angabe einer Zahl für orderkey bezieht sich auf die relative Position von columexpr in der* **Select***-Anweisung. Wenn die Klausel* **Order By** *nicht angegeben wird, werden die Reihen unsortiert zurückgegeben.*

Die Klausel **Group By** *enthält eine Folge von Angaben darüber, wie die ausgewählten Reihen zusammenzufassen sind. Um die Klausel* **Group By** *zu verwenden, müssen Sie eine Liste der zusammenzufassenden Spalten in der* **Select***-Anweisung sowie eine oder mehrere Ausdrücke (zum Beispiel* **Count** *oder* **Max***) angeben, die für die Datenaggregation sorgen. Dann geben Sie in der Klausel* **Group By** *dieselbe Liste der zusammenzufassenden Spalten an. Die* **Select***-Anweisung sortiert dann die Daten in der in der Klausel* **Group By** *angegebenen Reihenfolge und gibt eine Summenzeile für jede unverwechselbare Wertekombination zurück.*

groupexpr besteht aus einem Spaltennamen, der in der **Select***-Klausel aufgeführt ist, oder aus einer Folge von Spaltennamen in Klammern. Die Klausel* **Grouping Sets** *erlaubt die Angabe mehrerer Gruppierungsebenen.* **Grouping Sets** *((a,b)) ist gleichbedeutend mit* **Group By** *a,b. Die Klausel* **Rollup** *erlaubt die Angabe einer Liste von Spalten und führt zu einer Gruppierung der Reihen von rechts nach links.* **Rollup** *(a,b) ist äquivalent mit* **Grouping Sets** *((a,b), (a), ()).* **Cube** *erzeugt eine Übersichtstabelle mit allen möglichen Kombinationen der angegebenen Spalten.*

```
Cube(a,b)
```

Diese Anweisung ist dementsprechend äquivalent mit **Grouping Sets** *((a,b), (a),(b),()). Beachten Sie, dass es sich bei groupexpr auch um ein leeres Klammernpaar () handeln kann, wenn innerhalb der Klauseln* **Grouping Sets**, **Rollup** *oder* **Cube** *die Gesamtsumme repräsentiert werden soll.*

Die **Having***-Klausel enthält einen Ausdruck, der bestimmt, ob aggregierte Ergebnisse zurückgegeben werden sollen. Typischerweise wird sie eingesetzt, um zu ermitteln, ob die zwischenzeitliche Gruppierung einer* **Group By***-Klausel zurückgegeben werden soll. Wenn die Klausel* **Group By** *nicht angegeben wird, werden die Ergebnisse für die gesamte* **Select***-Anweisung zurückgegeben. In havingexpr muss eine Aggregatfunktion enthalten sein, die sich auf eine zu gruppierende Spalte bezieht.*

Die Klauseln **For Read Only** *und* **For Fetch Only** *bedeuten dasselbe. Sie können eine cursor-positionierte* **Delete***- oder* **Update***-Anweisung auf die Ergebnisse einer* **Select***-Anweisung anwenden, wenn eine der beiden Klauseln angegeben wird. Dies kann der Leistungssteigerung dienen, weil der Datenbank-Manager keine Exklusivsperren vornehmen muss.*

Die Klausel **For Update** *legt die Spalten der ausgewählten Reihen fest, die aktualisiert werden können. Wenn keine Spalten angegeben werden, wird von allen aktualisierbaren Spalten ausgegangen. Diese Information wird von der Abfrageoptimierung zur Leistungsverbesserung ausgewertet.*

Die Klausel **Optimize For** *gibt die erwarteterweise zurückgegebene Anzahl von Reihen an. Ohne Angabe geht die Abfrageoptimierung davon aus, dass alle Reihen der Tabelle ausgewählt werden. Die Angabe einer kleineren Anzahl von Reihen kann zu einer erheblichen Leistungssteigerung führen. Wenn die Anzahl der zurückgelieferten Reihen den angegebenen Wert überschreitet, läuft die Abfrage zwar weiterhin, jedoch gegebenenfalls nur unter deutlichem Geschwindigkeitsverlust.*

4.2.7.4 Argumente

columnexpr: Ein Spaltenausdruck, der sich auf eine einzelne Spalte bezieht und der folgenden Syntax entspricht:

```
{ [schemaname.]tablename.columnname |
tabledesignator.columnname | columnname | expr [ [ As ] newcolummn ] }
```

Dabei bezieht sich schemaname auf den Namen des Schemas, das sich im Besitz der Tabelle oder Sicht befindet, die den angegebenen Spaltennamen (columnname) enthält. Fehlt diese Angabe, wird Ihr Vorgabeschema als Vorgabe eingesetzt. tablename bezieht sich auf die Tabelle oder Sicht, die die Spalte enthält. Wenn diese Angabe fehlt, ermittelt das Datenbanksystem den passenden Namen der Tabelle oder Sicht auf Grundlage der in der **From***-Klausel enthaltenen Informationen. Alle Spalten in einer Tabelle lassen sich durch ein Sternchen (*) anstelle von columnname angeben. Die Klausel* **As** *newcolumn kann verwendet werden, um einem Ausdruck einen Spaltennamen zu geben.*
hostvar: Eine Variable in einem Anwendungsprogramm.
tablename: Der Name einer Tabelle oder Sicht.
correlation: Ein alternativer Name für table.
whereexpr: Ein logischer Ausdruck, der wahr sein muss, damit die entsprechende Reihe ausgewählt wird.
orderkey: Hier handelt es sich entweder um den Namen einer Spalte, einen ganzzahligen Wert, der die relative Position einer Spalte in der Reihe angibt (1 ist die erste Spalte, 2 die zweite usw.) oder einen Ausdruck, der einen einzelnen Wert zurückgibt.
groupexpr: Diese Angabe besteht aus einem Spaltennamen, der in der **Select***-Klausel aufgeführt ist oder einer Folge von Spaltennamen in Klammern. Die Klausel* **Grouping Sets** *erlaubt die Angabe mehrerer Gruppierungsebenen.* **Grouping Sets** *((a,b)) ist gleichbedeutend mit* **Group By** *a,b. Die Klausel* **Rollup** *erlaubt die Angabe einer Liste von Spalten und führt zu einer Gruppierung der Reihen von rechts nach links.* **Rollup** *(a,b) ist äquivalent mit* **Grouping Sets** *((a,b), (a), ()).* **Cube** *erzeugt eine Übersichtstabelle mit allen möglichen Kombinationen der angegebenen Spalten.*

```
Cube(a,b)
```

Diese Anweisung ist dementsprechend äquivalent mit **Grouping Sets** *((a,b), (a),(b),()). Beachten Sie, dass es sich bei groupexpr auch um ein leeres Klammernpaar () handeln kann, wenn innerhalb der Klauseln* **Grouping Sets**, **Rollup** *oder* **Cube** *die Gesamtsumme repräsentiert werden soll.*
*havingexpr: Ein logischer Ausdruck, der dazu führt, dass nur Reihen, für die dieser wahr (***True***) ist, zurückgegeben werden.*
numrows: Die Anzahl der erwarteterweise von der Abfrage zurückgegebenen Reihen. Ohne Angabe gilt die Anzahl der Reihen in der Tabelle für diesen Wert als Vorgabe. Das Überschreiten dieses Wertes kann zu Leistungseinbußen führen. Wenn Sie jedoch die Rückgabe relativ weniger Reihen erwarten, können Sie die Leistung unter Umständen signifikant verbessern.

[Freeze]

5 Praxis OpenOffice

Das Folgende bezieht sich auf OpenOffice 1.1 RC1 (oO).

Leider ist es zurzeit nicht möglich, Datenbanken mit oO zu entwerfen bzw. zu erstellen. Das Paket von StarOffice Version 6 enthält dagegen weiterhin das DBMS Adabas. Abgesehen davon können bestehende Datenbanken abgefragt oder geändert werden. Zum Einbinden der Terra-Datenbank muss man allerdings andere Schritte befolgen.

Wenn Sie StarOffice verwenden überspringen Sie diesen Absatz. Verwenden Sie oO, lesen Sie hier weiter und überspringen 6 bis zu 6.5.

5.1 Datenbank importieren

Öffnen Sie zunächst die ANSICHT – DATENQUELLEN. Füllen Sie die Felder wie in den Abbildungen zu sehen ist. Gehen Sie über das Kontextmenü DATENQUELLEN VERWALTEN weiter mit den Schaltflächen NEUE DATENQUELLE - DATEQUELLEN-URL ... (Abbildung 8) - VERWALTEN (Abbildung 9) zum ODBC-Datenquellen-Administrator. Legen Sie dort eine Verknüpfung auf die Terra-Datenbank mit dem passenden Treiber über die Schaltfläche HINZUFÜGEN an. Geben Sie den Pfad zur Datei terrra.mdb an und verwenden Sie den MS-Access-Treiber (Abbildung 10).

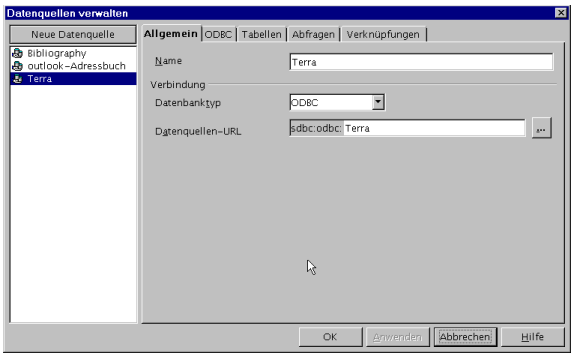


Abbildung 8

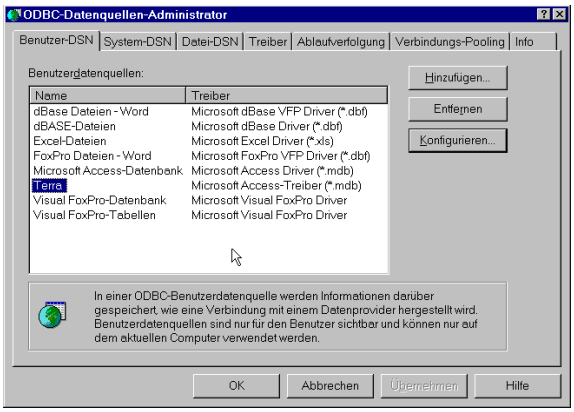


Abbildung 9

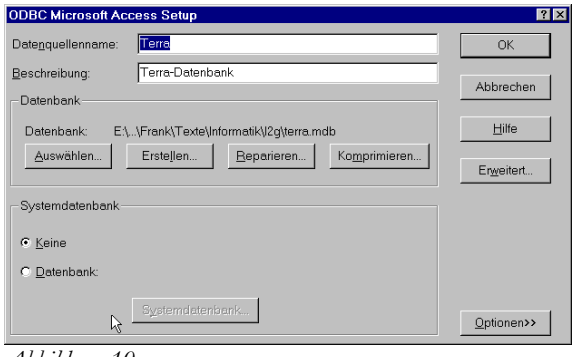


Abbildung 10

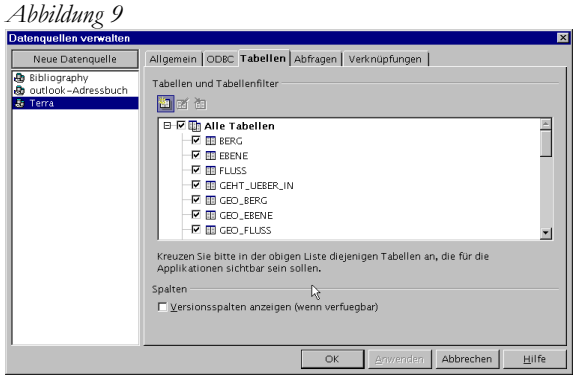


Abbildung 11

Die Karteikarte ODBC betrifft uns nicht. Wenn alles geklappt hat, sehen Sie wie in Abbildung 11 die Tabellen der Datenbank. Verlassen Sie das Fenster. Nun können Sie mit der Terra-Datenbank arbeiten.

5.2 Besonderheiten

Natürlich ändert sich mit jeder neuen Version auch das Aussehen und das Verhalten des Programms. Einiges ist mir aufgefallen, dass ich hier nun noch kurz anhand eines Beispiels vorstellen möchte. Überspringen Sie diesen Absatz, wenn Sie nicht wissen was eine Gruppierung ist.

Das Beispiel: *Gesucht sei das Verhältnis von den Einwohnern aller Städte zu den (gesamt) Einwohnern eines Landes.* Wir müssen also die Tabellen Stadt und Land verbinden eine Gruppierung vornehmen um die Summe der Stadteinwohner je Land zu ermitteln. Das zeigen die Bilder 16-17. Weniger wichtig ist, dass manche Länder mehr Stadteinwohner als Einwohner des gesamten Landes haben. Aber die Sortierung wird in der Design-Ansicht der Abfrage zwar eingestellt aber nicht wirksam. Sie muss erst in der SQL-Ansicht nachträglich von Hand eingefügt werden. Woran man sieht, dass die SQL-Befehle auch weiterhin große Bedeutung haben.

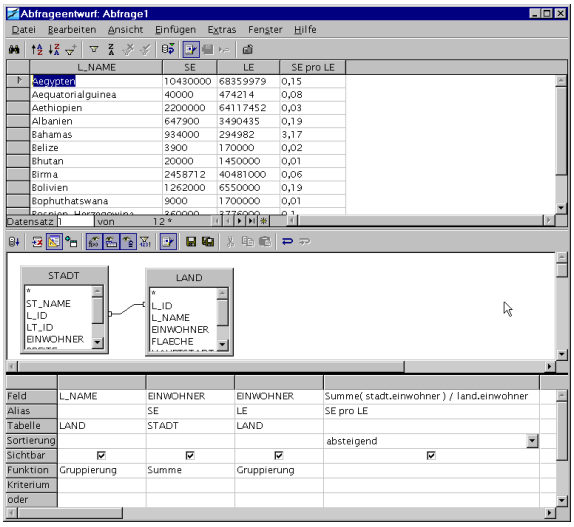


Abbildung 12

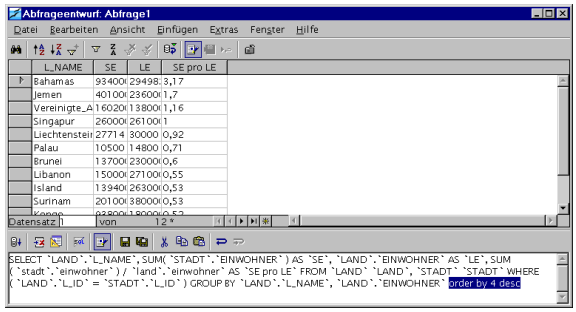


Abbildung 13

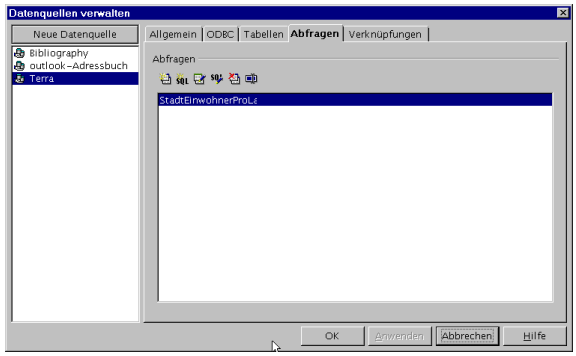


Abbildung 14

6 Praxis StarOffice

Der unten beschriebene Sachverhalt bezieht sich auf Star Office 5.2 (SO). Beachten Sie, dass Sie die Online-Hilfe verwenden können. Dazu wählen sie im Hilfe-Menü den Punkt INHALT. Suchen Sie dann nach dem Eintrag STAROFFICE BASE. Das unten beschriebene Vorgehen finden Sie in der Hilfe ausführlicher beschrieben.

Um einzelne Phrasen zu finden verwenden Sie das Fernglas-Symbol.

Datenbanken bestehen in SO aus Tabellen, Abfragen, Berichten und Formularen. Um das Problem der Abfragen zu demonstrieren, verwenden wir die **Terra**-Datenbank. Deren Aufbau wird in Kapitel 3 beschrieben.

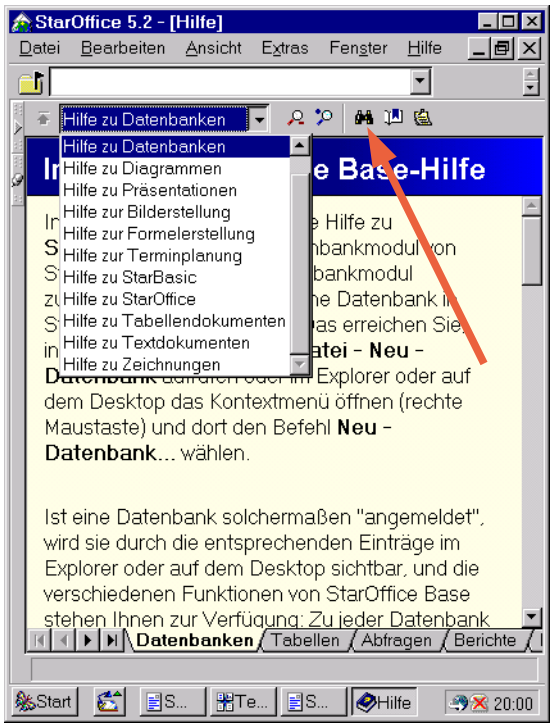


Abbildung 15

6.1 Datenbank importieren

Die Terra-Datenbank befindet sich in Ihrem Multididac-Verzeichnis. So können Sie vorgehen:

- Wählen Sie DATEI – NEU - DATENBANK.
- Geben Sie der Datenbank den Namen **Terra**.
- Wählen Sie die Karteikarte TYP. Suchen Sie MS ACCESS 97 und suchen Sie die Datenquelle in c:\multididac\informatik_12x\nName_vName\terra.mdb und bestätigen Sie mit OK.

Sie erreichen die Datenbank ab sofort über den Explorer. Das Kapitel 6.2 können Sie überspringen, wenn Sie keine neue Datenbank anlegen müssen.

6.2 Datenbank neu erstellen

31. Der Befehl DATEI - NEU - DATENBANK... öffnet den Eigenschaften-Dialog, in dem Sie einige Einstellungen vornehmen müssen.
32. Vergeben Sie im Register Allgemein als Namen für Ihre Datenbank z. B. "Adressen".
33. Im Register Typ können Sie dBase wählen. Dieses Dateiformat ist weit verbreitet und deshalb gut geeignet, die Datenbank auch mit andern Datenbankbetriebssystemen zu bearbeiten.

6.3 Datenbankdesign

Unter dem „Design einer Datenbank“ versteht man das Aufteilen der Daten in getrennte Relationen. Im Prinzip besteht es aus zwei Hauptschritten. Dem **logischen Datenbankdesign**, dem Aufteilen der Attribute in unterschiedliche Relationen und dem **physischen Datenbankdesign** bei dem der logische Entwurf in ein konkretes Datenbank-Managementsystem umgesetzt wird. Beide Schritte sind durchaus nicht trivial und wir werden hier nicht weiter darüber sprechen. Sollten Sie weitere Hinweise dazu benötigen, sehen Sie sich die Video-Datenbank in der Online-Hilfe zu SO an oder suchen Sie z. B. im Internet nach den Begriffen **Entity-Relationship-Mo-**

dell und **Normalisierung** im **CODDschen-Relationenmodell**. In Zukunft setzen wir voraus, dass die Struktur der Tabellen klar ist.

6.4 Tabellenentwurf

- Sie müssen im Explorer die Zeile TABELLEN unter einer Datenbank sehen.
- Wählen Sie im Kontextmenü NEU – TABELLE - TABELLENENTWURF, um eine neue Tabelle anzulegen.
- Im Tabellenentwurf können Sie nun alle notwendigen Attribute (Spalten/Felder der Tabelle) definieren.
- In jeder Spalte stehen Daten eines Typs. Legen Sie deshalb für jedes Attribut den Datentyp fest. Je nach Datenbankformat gibt es unterschiedliche Möglichkeiten. Folgende Überlegungen könnten für die Adressdatenbank zutreffen:

Feldname	*Datentyp*	*Länge*
Vorname	*Text*	*25*
Nachname	*Text*	*25*
Straße	*Text*	*25*
Ort	*Text*	*20*
PLZ	*Zahl oder Text*	*5*
Telefon	*Zahl oder Text*	*11*
Fax	*Zahl oder Text*	*11*

Tabelle 4

- Um den gewünschten Datentyp festzulegen können Sie das Drop-Down-Menü nutzen.
- Sollten noch weitere Daten benötigt werden, so müssen diese hinzugefügt werden.
- Nachdem die Definition abgeschlossen ist und der Entwurf geschlossen wird, ist es noch erforderlich der Tabelle einen Namen zu geben.

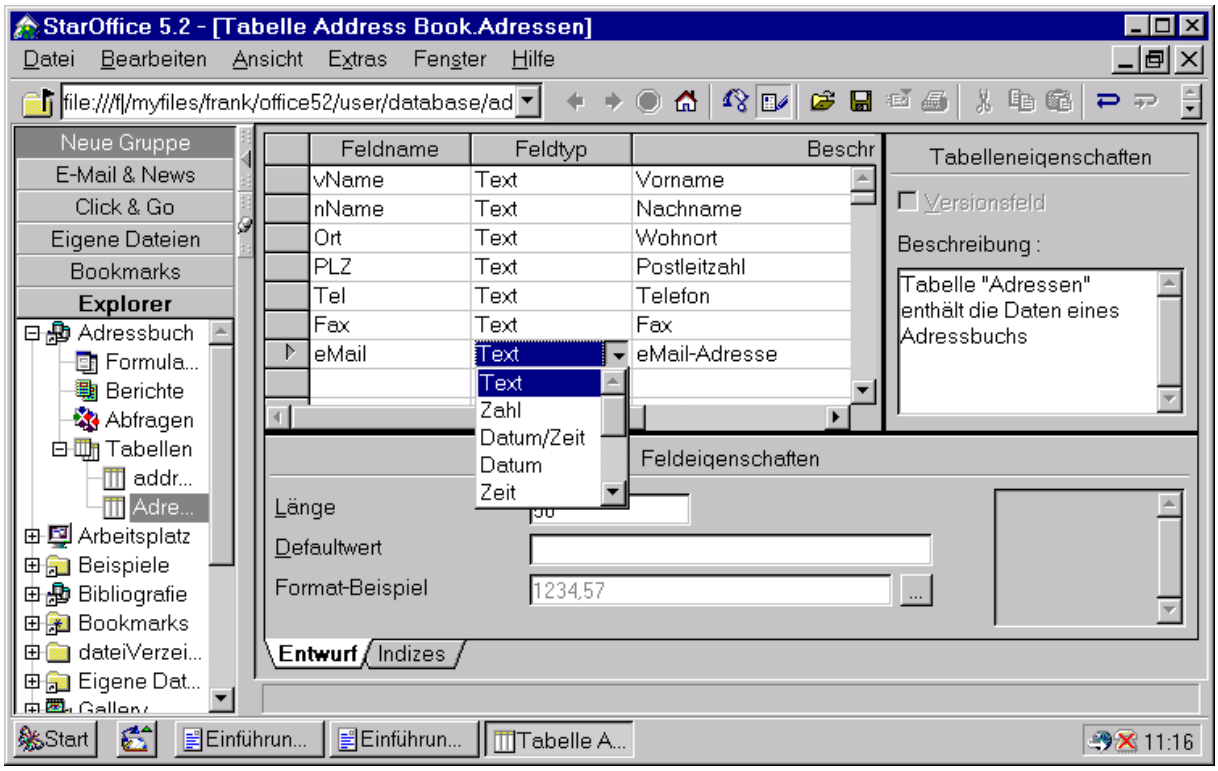


Abbildung 16: Tabellenentwurf

6.5 Operationen auf Datensätzen

Vergleichen Sie dazu au
den Aufbau und die wic
zen werden auch in Kapi

6.5.1 Hinzufügen, Ändern, Löschen

Die angelegte Tabelle kann, wie Sie das aus den „normalen“ StarOffice-Tabellen kennen bearbeitet werden. Um einen Datensatz zu löschen, muss die Zeile selektiert werden. Danach rufen Sie im Kontextmenü LÖSCHEN auf. Die Reihenfolge der Zeilen und Spalten kann beliebig geändert werden, aber ein neuer Datensatz kann stets nur in der untersten Zeile eingefügt werden.

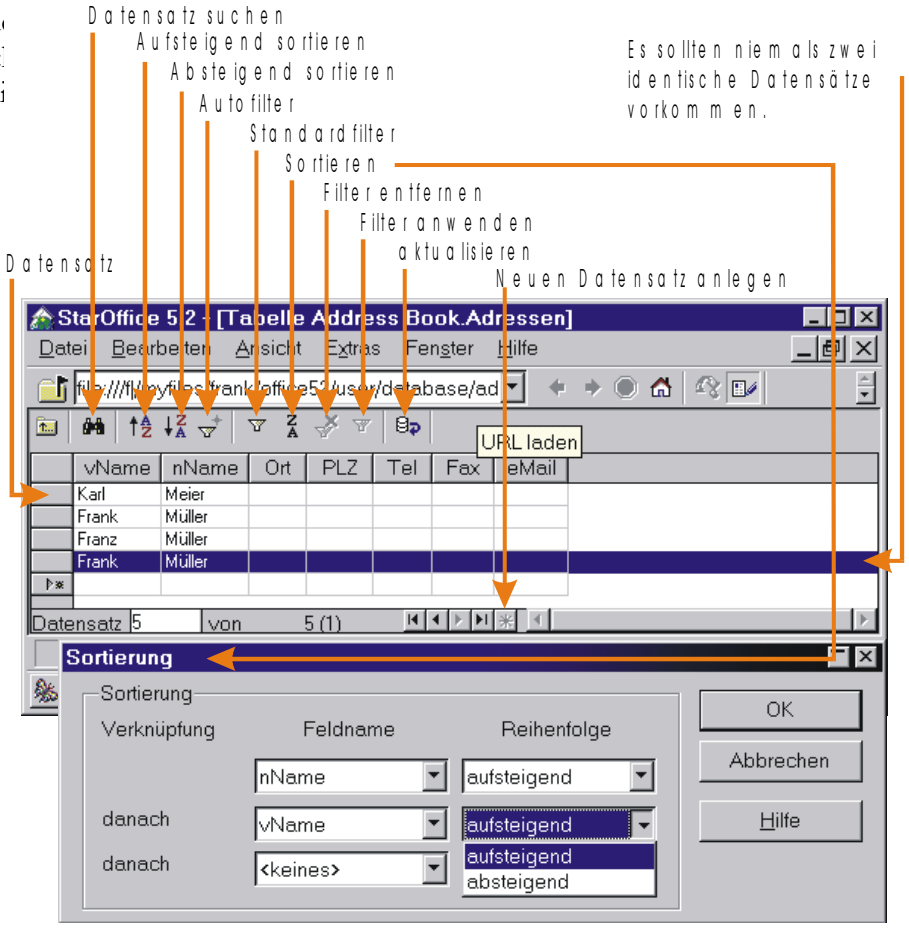


Abbildung 17

6.5.2 Sortieren

Die Sortierung kann nach unterschiedlichen Kriterien erfolgen (siehe Abb. 17).

6.5.3 Filtern

Bei längeren Datenlisten sind die Filterfunktionen hilfreich. Sie verwenden sie, um Datensätze, die bestimmten Kriterien entsprechen, anzuzeigen.

In Abbildung 18 und 19 sehen Sie, wie in der Tabelle Stadt der Terra-Datenbank im Attribut L_ID der Attributwert DK ausgewählt wurde. Mit der AUTOFILTERN-Funktion werden nun nur noch Datensätze, deren Wert L_ID=“DK“ entspricht, angezeigt. In Abbildung 19 wurde außerdem nach der Einwohnerzahl aufsteigend sortiert. Das geschieht, indem Sie die Spalte „Einwohner“ durch Klicken auf den Spaltenkopf markieren und anschließend das Symbol AUFSTEIGEND SORTIEREN betätigen.

Beachten Sie auch, dass nur die jeweils aktuelle Anzahl von Datensätzen angezeigt wird.

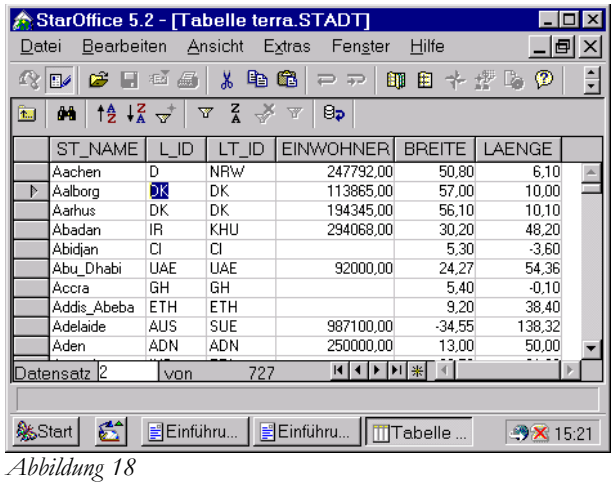


Abbildung 18

Bei der Verwendung des Standardfilters können mehrere Filterkriterien angewendet werden. Sehen Sie dazu Abbildung 20 und 21.

In der gefilterten Tabelle werden alle Städte Deutschlands, deren Namen mit einem **n** enden, aufgelistet.

Abbildung 20

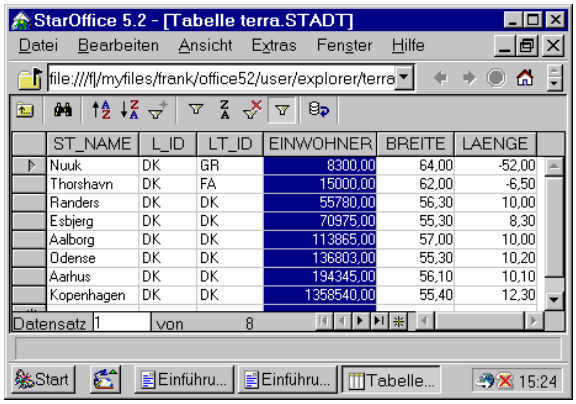


Abbildung 19

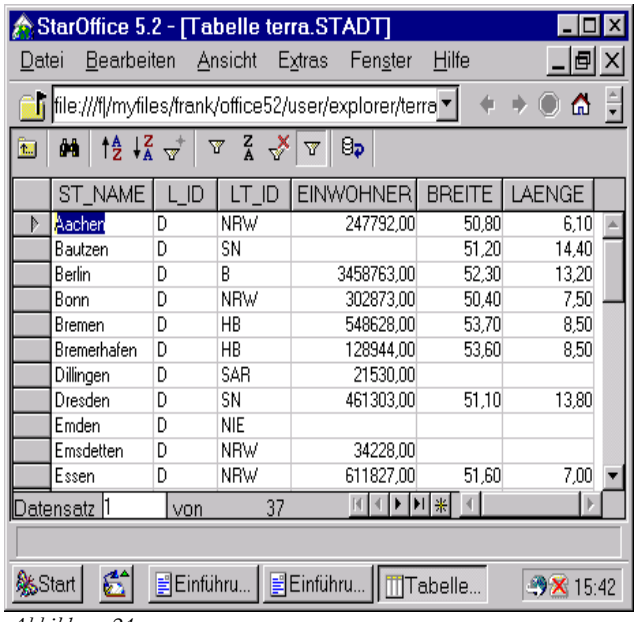


Abbildung 21

6.5.3.1 Aufgaben

- Wie viele Städte liegen in Bayern? [21]
- Welche Städte haben mehr als 850.000 Einwohner? [166]
- Welche Städte haben ein „tz“ im Namen? [3]
- Welche Städte haben keine Einwohner? [25]
- Welche Länder haben Hauptstädte, die mit „m“ beginnen? [21]
- Welche Länder haben mehr als 1 Mio., aber weniger als 10 Mio. Einwohner und Hauptstädte, die mit „m“ beginnen. [7]

6.5.4 Guppieren und Rechnen

Ähnlich wie Sie bei längeren Datenlisten mit den Filterfunktionen von SO bestimmte Datensätze angezeigt haben, können Sie durch das Gruppieren und Berechnen von Teilergebnissen mehr Informationen gewinnen. << Dieser Absatz ist noch unvollständig. >>

6.6 Abfragen

6.6.1 Erstellen mit dem Autopiloten

1. *Wählen Sie*
   1. *DATEI – AUTOPILOT – ABFRAGE, um das Fenster AUTOPILOT ABFRAGE: TABELLENAUSWAHL zu erhalten oder*
   2. *Klicken Sie im Kontextmenü des EXPLORER – ABFRAGEN auf NEU – ABFRAGE – AUTOPILOT.*
2. *Wählen Sie in der Liste die Option DATENBANK au, werden die im entsprechenden Ordner vorhandenen Tabellen aufgeführt. Wählen Sie nun eine Tabelle aus.*
3. *Suchen sie aus den vorhandenen Feldern die für die Abfrage benötigten aus.*
4. *Nach einem Klick auf WEITER können Sie die Filterbedingungen in mehreren Stufen durch die Wahl aus den Angaben der Listen einstellen. Tragen Sie als KRITERIUM ein, welcher Vorgabe die Feldinhalte entsprechen sollen. Wird z. B. ein Ort gesucht, schreiben Sie den Namen hin. Sie müssen sich nicht um Angaben wie Gleichheits- oder Anführungszeichen kümmern. Diese werden vom Programm automatisch eingesetzt. Unterhalb der Zeile KRITERIUM sehen Sie weitere Zeilen mit der Bezeichnung ODER. Mit ihnen und mit der Zeile KRITERIUM stellen Sie zusammengesetzte Bedingungen auf. Alle Bedingungen, die Sie in derselben Spalte! untereinander eintragen, werden durch ein logisches ODER verknüpft. Für alle Bedingungen in derselben Zeile nebeneinander gilt ein logisches UND.*

5. *In diesem Schritt können Sie schon die Schaltfläche VORSCHAU anklicken. Betrachten Sie das Ergebnis im Beamer, und wählen Sie WEITER.*
6. *Im nächsten Schritt legen Sie in bis zu vier Stufen das Sortieren fest. Mit einem Klick auf die Schaltflächen neben den Sortierfeldern stellen Sie von A-Z auf Z-A um und umgekehrt.*
7. *Geben Sie noch einen Abfragetitel an, oder übernehmen Sie die Vorgabe. Mit der Wahl FERTIGSTELLEN beenden Sie die Einstellungen.*

Abbildung 22: Mehrere Bedingungen

Sie können bei der Betrachtung eines Filterergebnisses den Beamer und das Desktop-Dokument gemeinsam verwenden, um so an-dop-Bea-

ST_NAME	EINWOHNER
Bautzen	
Chemnitz	259187
Goerlitz	

iehe Abb. 22

wohl die gesamte Tabelle als auch die durch die Abfrage gefilterte Tabelle zugleich zusehen. Klicken Sie dazu im Explorer auf die betreffende Zeile der Abfrage und pelt auf den Namen der Tabelle. Im Ergebnis sehen Sie die gefilterte Abfrage im mer und die gesamte Tabelle im Desktop-Dokument darunter.

Eine einmal angefertigte Abfrage können Sie immer wieder verwenden. Sie wird im Ordner ABFRAGE gespeichert. Um sie erneut zu starten, müssen Sie die Zeile im Explorer suchen und darauf doppelklicken. Die Abfrage wird sofort gestartet und des Ergebnis nach einer Wartezeit eingeblendet. Ein neuer Aufruf blendet ein vorher angezeigtes Ergebnis aus.

[Huttel97]

6.6.2 Manuell aufbauen

Reicht das im vorausgehenden Abschnitt vorgestellte Angebot des AutoPiloten für komplexe Abfragen nicht aus bauen Sie eine Abfrage gezielt manuell auf. Gehen Sie so vor:

1. *Starten Sie den Explorer. Markieren Sie den Ordner ABFRAGE, und beginnen Sie mit NEU – ABFRAGE aus dem Kontextmenü.*
2. *Entscheiden Sie sich gegen den AutoPiloten und für die ENTWURFSANSICHT, um eine manuelle Abfrage aufzubauen.*
3. *Tragen Sie einen Namen ein, und bestätigen Sie.*
4. *Im Fenster TABELLEN HINZUFÜGEN müssen Sie sich entscheiden, auf Basis welcher Tabelle(n) die Abfrage durchgeführt werden soll. Markieren Sie die gewünschte Zeile, und wählen Sie HINZUFÜGEN. Sie können die Daten einer Abfrage mehreren Tabellen entnehmen. Fügen Sie in diesem Fall die neuen Daten hinzu.*
5. *Schalten Sie mit SCHLIESSEN zum Fenster des Abfrageentwurfs.*
6. *Klicken Sie auf das Icon TABELLEN HINZUFÜGEN, und wählen Sie aus der Liste, falls noch keine Liste mit Feldnamen angezeigt wird.*
7. *Im nächsten Arbeitsgang ziehen Sie aus der Liste in der oberen Fensterhälfte Feldnamen in die Zeile FELD. Haben Sie in der ersten Zeile einen Namen angeordnet, wird der zugehörige Tabellenname automatisch in der zweiten Zeile eingefügt. Sie können die Anzeige ändern, wenn Sie auf den Pfeil rechts im Feld klicken und eine andere Wahl treffen. Wiederholen Sie die Einstellungen für weitere Felder.*
8. *Klicken Sie unter einem Feld in die dritte Zeile, und wählen Sie bei Bedarf eine Sortierform.*
9. *Klicken Sie die Markierfelder in der Zeile SICHTBAR nach Wunsch an.*
10. *In den letzten Zeilen haben Sie Platz zur genauen Definition, welche Kriterien für einen Erfolg der Abfrage zutreffen müssen. Da Sie diese über fünf Ebenen staffeln können, lassen sich Abfragen hier präzise formulieren.*

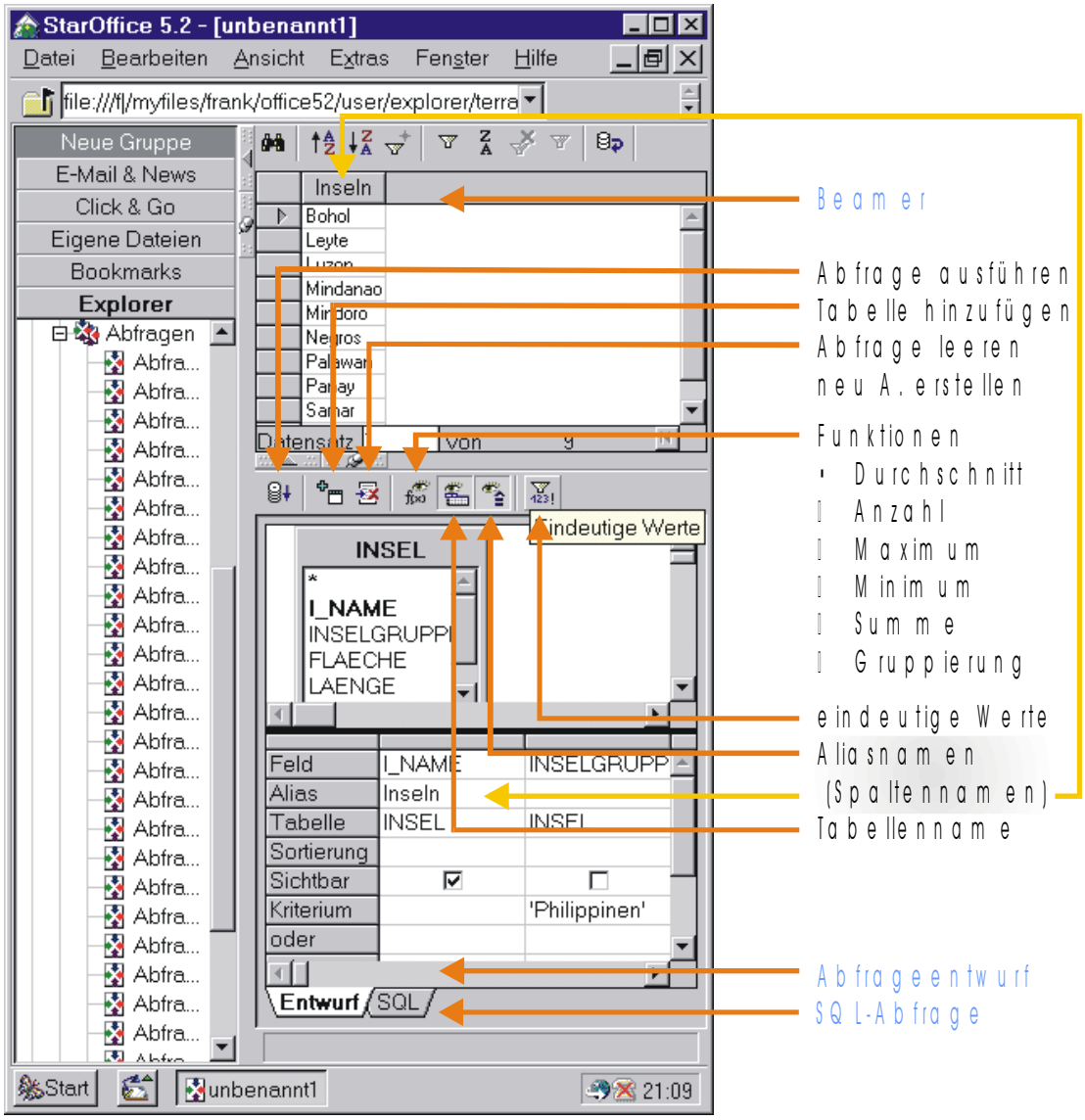


Abbildung 23: siehe Aufgabe 3 (Gesucht sind alle Inseln (Name) der Inselgruppe der Philippinen.)

11. *Um die Abfrage zu testen, klicken Sie auf das Icon ABFRAGE AUSFÜHREN. Nach der Untersuchung können Sie das Fenster im Menü DATEI oder mit dem Icon SCHLIESSEN ausblenden.*
12. *Nach einem Klick auf das Icon ABFRAGE AUSFÜHREN wird das Ergebnis im eigenen Fenster und mit einer Objektleiste zur Bedienung angezeigt. Diese Anzeige lässt sich mit den an gebotenen Icons weiter organisieren. Wenn Sie zunächst wissen wollen, wie viele Ergebnisse die Abfrage erbracht hat, klicken Sie auf das Icon ANZAHL DATENSÄTZE.*
13. *Stehen in der Abfrage nur Ergebnisse aus dem Feld mit den angegebenen Sortierkriterien, müssen Sie noch die Feldnamen einfügen, deren Inhalte zusätzlich angezeigt werden sollen. Legen Sie nur auf die Anzeige Wert, geben Sie keine Kriterien an. Klicken Sie nach einer solchen Änderung nochmals auf das Icon ABFRAGE AUSFÜHREN. Wollen Sie neu beginnen, aktivieren Sie zuvor das Icon FILTER ENTFERNEN.*

[Huttel97]

6.6.3 SQL

Der Begriff »Structured Query Language« bezeichnet eine strukturierte ***Abfragesprache*** *für Datenbanken. In den meisten Fällen werden Sie Abfragen mit dem AutoPiloten anlegen. Sie können aber auch direkt oder interaktiv SQL per Maus und Abfrageformular zum Beschreiben von Datenbankabfragen benutzen. Aus den Einträgen im Formular legen Sie die zugehörige SQL-Abfrage an. Diese können Sie bearbeiten oder direkt eingeben.*
Nutzen Sie den AutoPiloten, und gehen Sie folgendermaßen vor, um eine entsprechende Abfrage aufzubauen:

1. *Aktivieren Sie im Explorer die Datenbank, für die die Abfrage gelten soll.*
2. *Wählen Sie die Zeile ABFRAGEN und im Kontextmenü NEU – ABFRAGE. Legen Sie einen Namen fest, stellen Sie im Abfrage formular oben zunächst die benötigte Tabelle ein, und tragen Sie unten in tabellarischer Form die Abfrage ein.*
3. *Übernehmen Sie aus dem oberen Teil des Abfrageformulars Datenfelder in die Tabelle.*
4. *Die im Abfrageformular durch Klicken und wenige Einträge zusammengestellte Abfrage ist gleichzeitig in SQL übersetzt worden. Das Abfrageformular besteht aus zwei Registern. Wenn Sie durch Anklicken der Registerzunge SQL am unteren Rand um schalten, sehen Sie die gleiche Abfrage in der Formulierung durch die SQL-Sprache.*
5. *Klicken Sie auf das Icon ABFRAGE AUSFÜHREN, wird das Ergebnis im Beamer angezeigt.*

Verwenden Sie das Register SQL am unteren Rand des Abfrageformulars, wenn Sie die Abfrage direkt in der Abfragesprache SQL formulieren wollen.

Der in Abbildung 22 gezeigten Abfrage entspricht die folgende SQL-Anweisung:

```
SELECT I_NAME AS Inseln FROM INSEL WHERE INSELGRUPPE = 'Philippinen'.
```

Wenn Sie Abfragen durch Anklicken zusammenstellen und zur SQL- Darstellung umschalten, werden Sie feststellen, dass die SQL-Formulierungen leicht zu verstehen sind. Die möglichen Inhalte sind allerdings sehr umfangreich und werden ausführlich in eigenen Publikationen behandelt. Für die tägliche Arbeit wird im allgemeinen die Verwendung des AutoPiloten die effektivere Technik sein.

Weitere Ausführungen zu SQL finden Sie in 4.

6.6.4 Daten als Bericht herausgeben

Gespeicherte Datenbanken können Sie auf dem Bildschirm anzeigen oder drucken. Am elegantesten ist dies mit einem Bericht möglich. Ein Bericht ist eine Zusammenfassung von Ausgabedaten. Mit dem Aufbau, der Gestaltung und Ausgabe eines Berichtes können Sie die folgenden Arbeiten erledigen:

- *Daten aus einer Datenbank wählen und in bestimmter Anordnung in Berichten ausgeben,*
- *Daten können in einem Bericht in logische Gruppen gegliedert und sortiert werden, um so neu Zusammenhänge zu zeigen,*
- *Berichte können mit grafischen Elementen gestaltet werden, so dass die Datensammlungen leichter verständlich werden.*

Berichte können außerdem am Bildschirm angezeigt, für die weitere Verwendung gespeichert oder auf Papier ausgegeben werden.

[Keller99]

<< Dieser Absatz ist noch unvollständig. >>

Literaturverzeichnis

[Keller99] Keller, Dr.-Ing. B.: Lehrmaterial zur Lehrveranstaltung "Theorie und Praxis relationaler Datenbanksysteme", TU-Dresden, http://wwwdb.inf.tu-dresden.de/~keller

[Sauer98] Hermann Sauer: Relationale Datenbanken – Theorie und Praxis, 4. aktualisierte und erweiterte Auflage – Addison-Wesley 1998, ISBN 3-8273-1381-3

[Payer97] **Payer, Margarete <1942 - >:** Datenbankaufbau : Skript / Margarete Payer & Alois Payer. -- Kapitel 1: Grundkonzepte von Datenbanken. -- Fassung vom 2. Juni 1997. -- URL: http://machno.hbi-stuttgart.de/~payer/dbauf01.html.

[Huttel97] Huttel, Klaus Peter: StarOffice 4.0 optimal einsetzen – Die Office-Komplettlösung, ECON Verlag 1997, ISBN 3-612-28165-8

[Freeze] Freeze, Wayne S.: Die SQL-Referenz; Buchbeilage auf CD

[SQL21] Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH. Elektronische Fassung des Titels: SQL in 21 Tagen, ISBN: 3-8272-2020-3 URL: http://www.mut.de/media/buecher/SQL/Inhalt.htm

[LPSa] Lehrplan Informatik des Freistaates Sachsen URL: http://www.sn.schule.de

[MySQL] MySQL Language Reference – Copyright © 1997-2001 MySQL AB

„Datenbanken“ im Internet

- Eine einfache Einführung mit MS Access: http://www.teleschule.de/ts/Access95/01/theorie/01.shtml sehr gut für Einsteiger und
- eine etwas tiefergehende Einführung: http://fishmac.phonetik.uni-muenchen.de/db_seminar/
- SQL-Einführung
- SQL Tutorial
- SQL Hilfe
- Spezifikationen von SQL – ANSI-SQL-Standard
  http://www.inf.fu-berlin.de/lehre/SS94/einfdb/
  http://www.inf.fu-berlin.de/lehre/SS94/einfdb/SQL3/sqlindex.html
- Übersicht (englisch):
  http://www.jcc.com/sql_stnd.html
- Hilfe zu verschiedenen Themen (englisch):
  http://www.inquiry.com/techtips/thesqlpro
- umfangreiche Link-Sammlung (Database Resources):
  http://www.unifx.com/links.html
- Datenbank für Spezialisten:
  http://www.microsoft.com/data
  http://www.javasoft.com
  http://www.php.net
  http://www.stars.com/Authoring/DB.

Stand: Mai 2001

7 Lösungsvorschläge zu den Aufgaben aus 3.4

Die Lösungen haben Stand Juni 2011. Sollten Aktualisierungen an der Datenbank durchgeführt werden, so kann sich das auch auf die hier gezeigten Bilder auswirken.

Abfrage: A001

```
SELECT L_NAME, L_ID, EINWOHNER FROM LAND
```

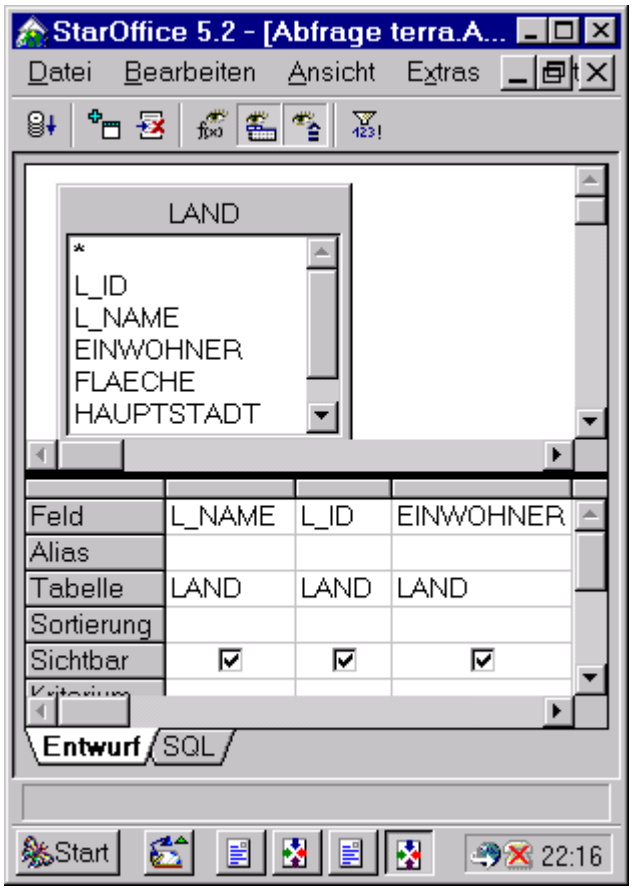


Abbildung 24

Abfrage: A002

```
SELECT L_NAME, EINWOHNER FROM LAND WHERE L_ID = 'D'
```

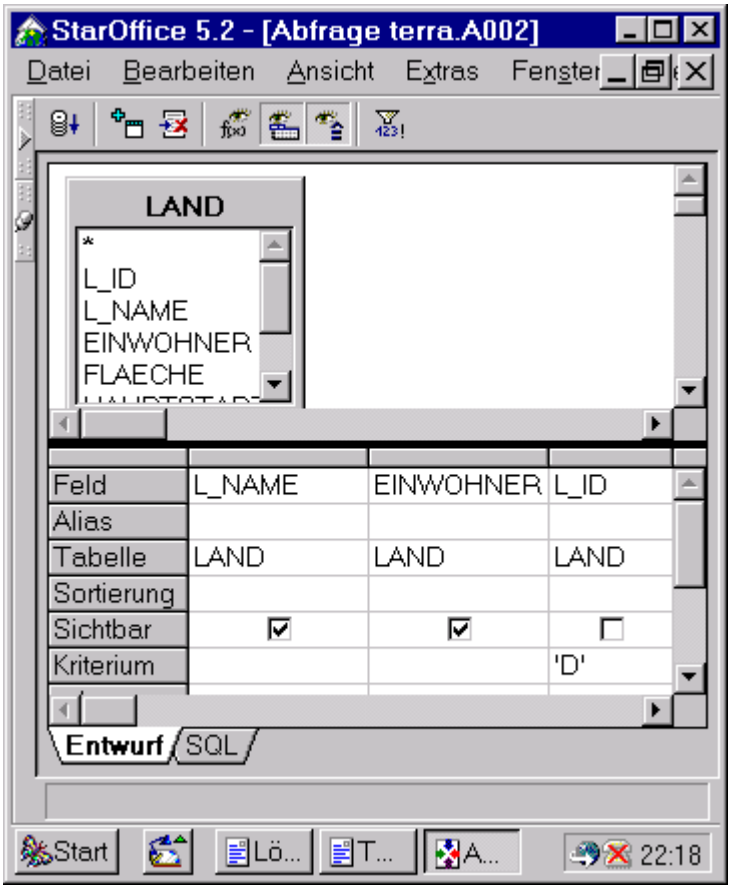


Abbildung 25

Abfrage: A003

```
SELECT I_NAME AS Inseln FROM INSEL WHERE INSELGRUPPE = 'Philippinen'
```

oder

```
SELECT I_NAME AS Inseln FROM INSEL WHERE INSELGRUPPE LIKE 'phil*'
```

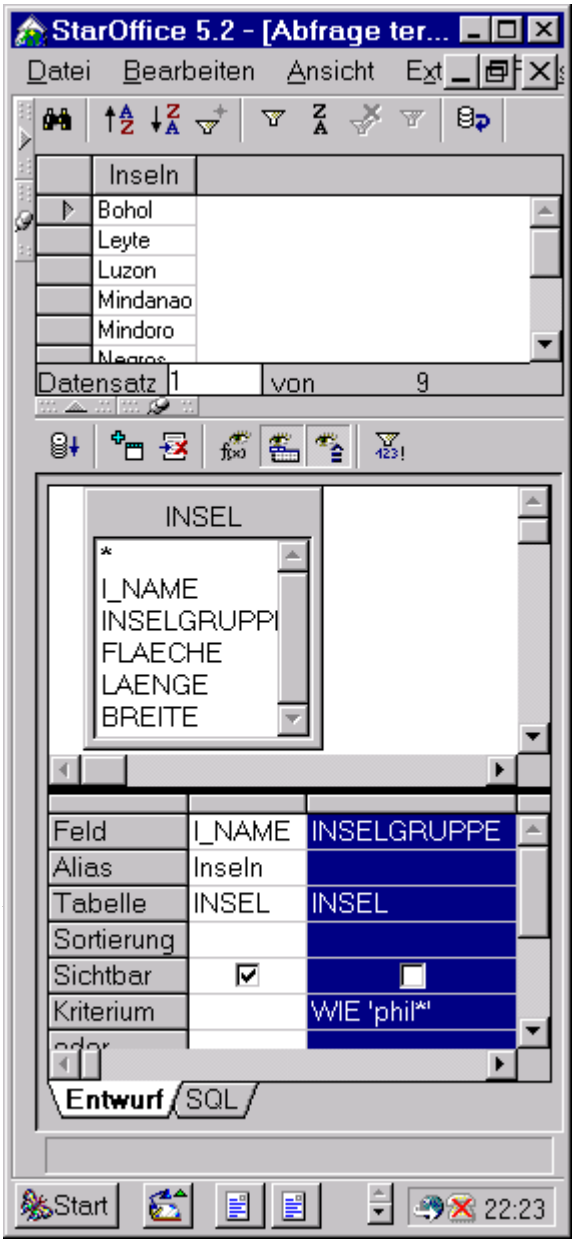


Abbildung 26

Abfrage: A004

```
SELECT K_NAME FROM KONTINENT
```

Abfrage: A005

```
SELECT SUM(EINWOHNER) AS Anzahl FROM LAND WHERE (L_ID='D' OR L_ID='A'
OR L_ID='CH')
```

oder

```
SELECT SUM( EINWOHNER ) AS Anzahl FROM LAND WHERE  L_ID in
( 'D','A','CH' )
```

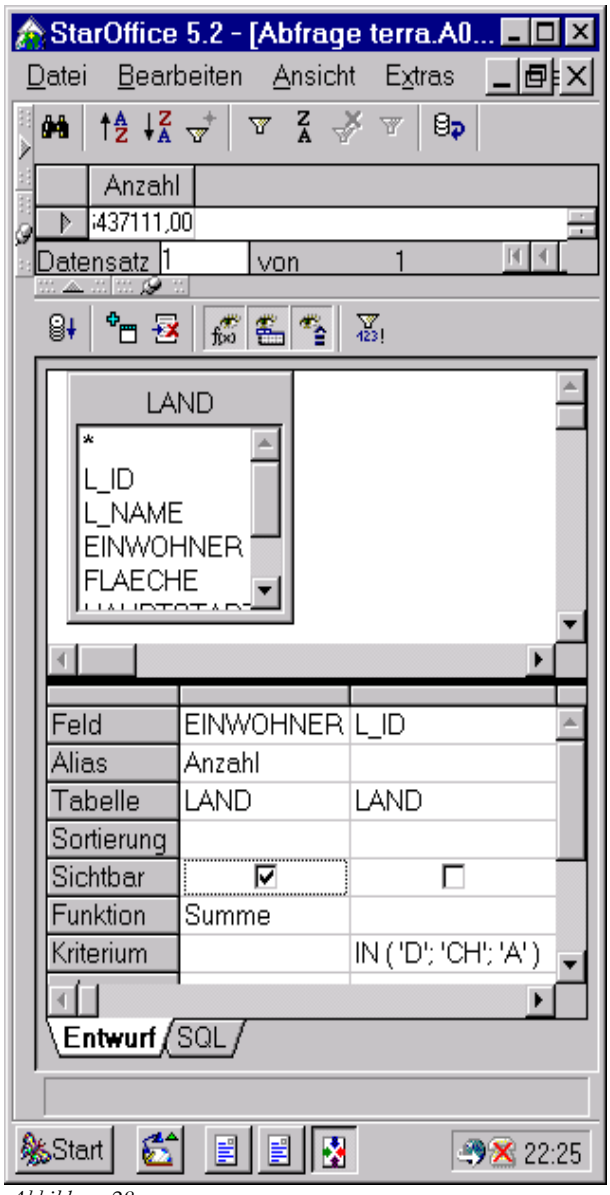


Abbildung 28

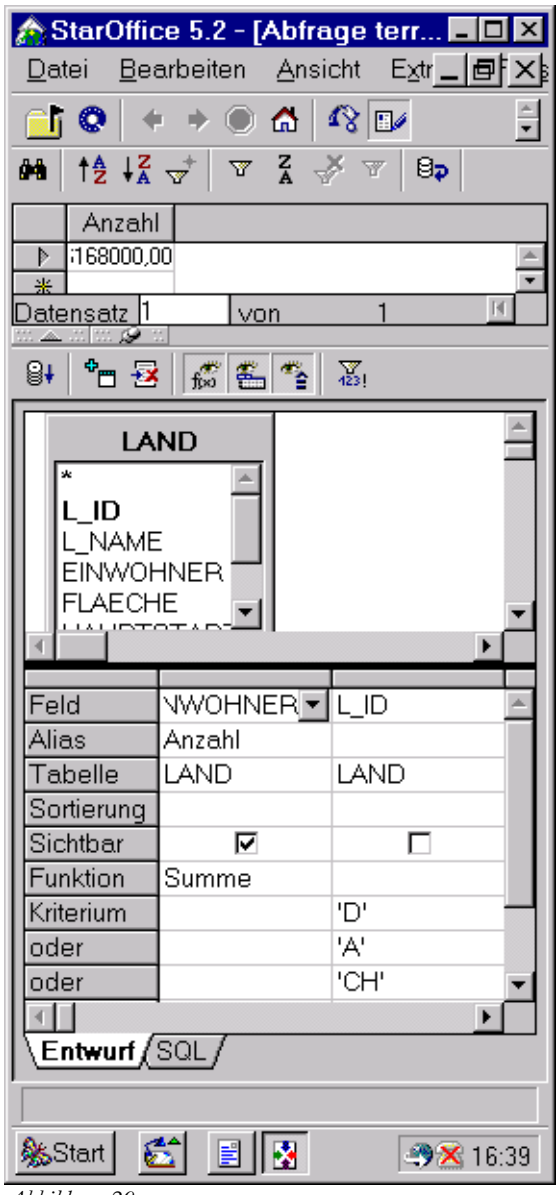


Abbildung 29

Abfrage: A006

```
SELECT L_ID, SUM( EINWOHNER ) FROM STADT GROUP BY L_ID
SELECT L_ID, SUM( EINWOHNER ) AS Summe Einwohner FROM STADT GROUP BY
L_ID
```

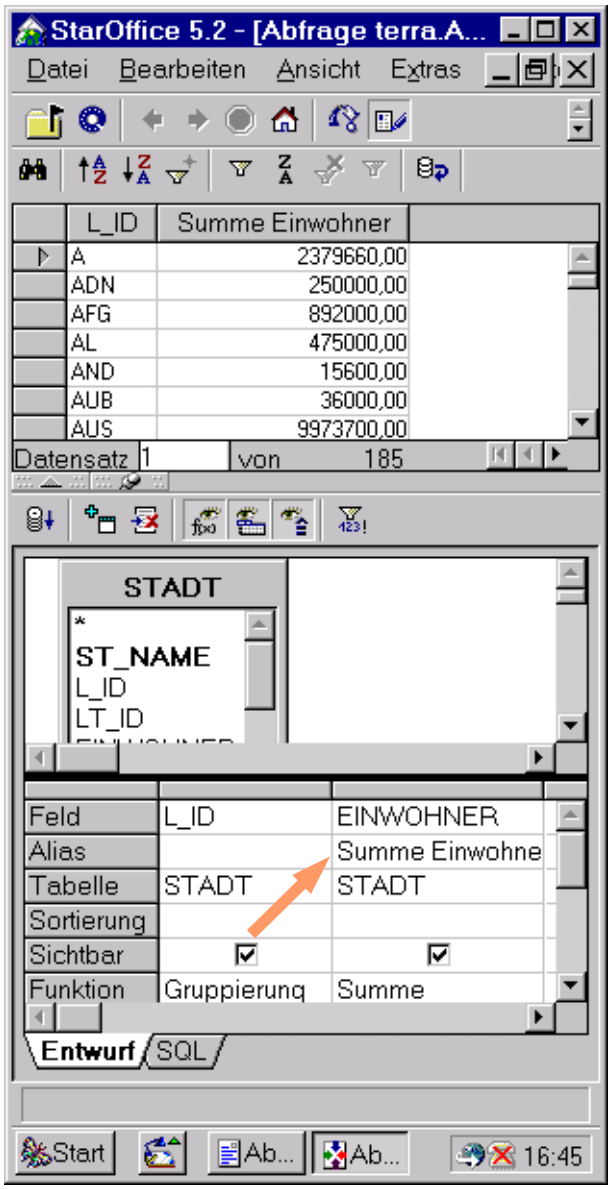


Abbildung 30

Abfrage: A006a

Wird die Fragestellung geringfügig geändert, so sollten einige Besonderheiten von OpenOffice 1.1 beachtet werden. Dazu mehr im Abschnitt „Praxis OpenOffice“.

Frage: Wie ist das (prozentuale) Verhältnis zwischen den Einwohnern aller Städte zu den Einwohnern eines Landes?

```
SELECT `LAND`.`L_NAME`, SUM( `STADT`.`EINWOHNER` ) AS `SE`,
`LAND`.`EINWOHNER` AS `LE`, SUM( `stadt`.`einwohner` ) /
`land`.`einwohner` AS `SE pro LE` FROM `LAND` `LAND`, `STADT` `STADT`
WHERE ( `LAND`.`L_ID` = `STADT`.`L_ID` ) GROUP BY `LAND`.`L_NAME`,
`LAND`.`EINWOHNER` ORDER BY 4 DESC
```

Diese Abfrage wird im Design-Modus nicht richtig formuliert. Scheinbar hat oO ein Problem die Syntax richtig zu parsen.

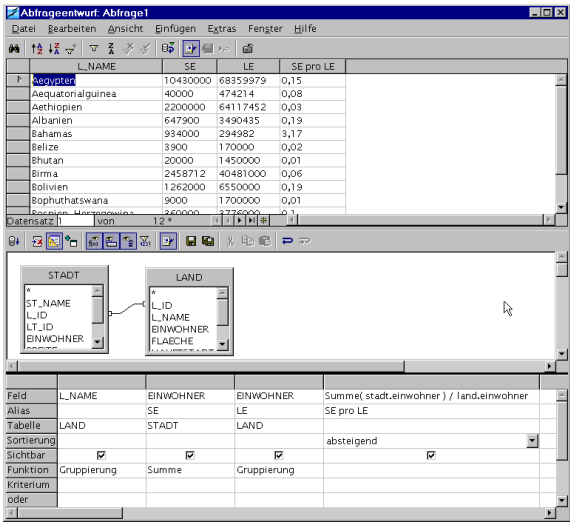


Abbildung 31

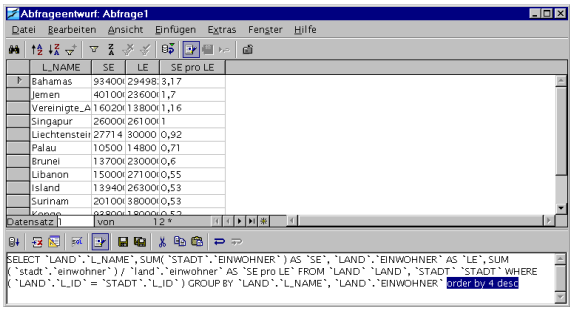


Abbildung 32

Abfrage: A007

```
SELECT B_NAME, GEBIRGE FROM BERG WHERE GEBIRGE LIKE '*alpen*'
```

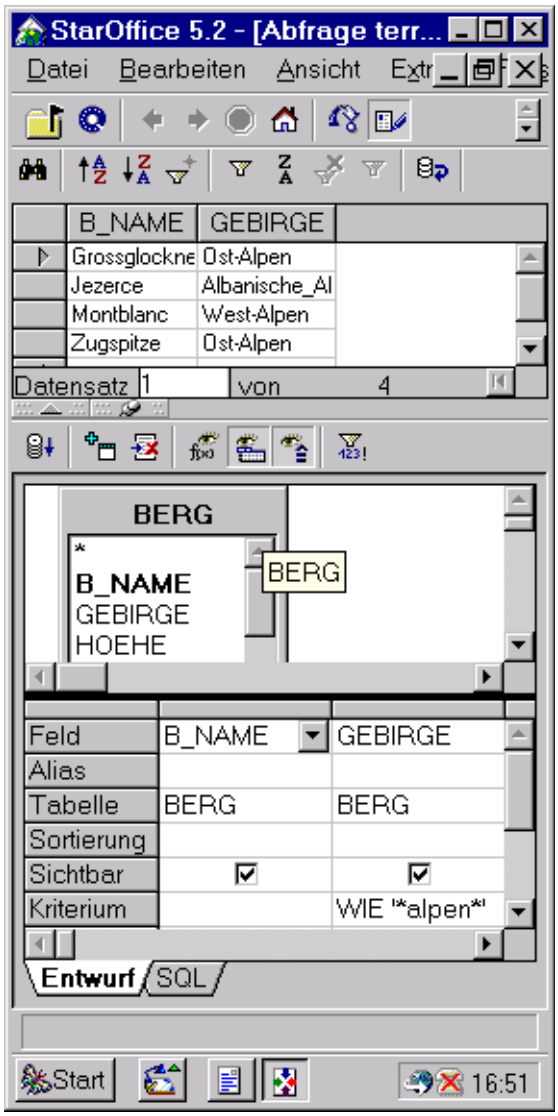


Abbildung 33

Abfrage: A008

```
SELECT GEBIRGE FROM BERG WHERE HOEHE > 3000 GROUP BY GEBIRGE HAVING
( GEBIRGE IS NOT NULL )
```

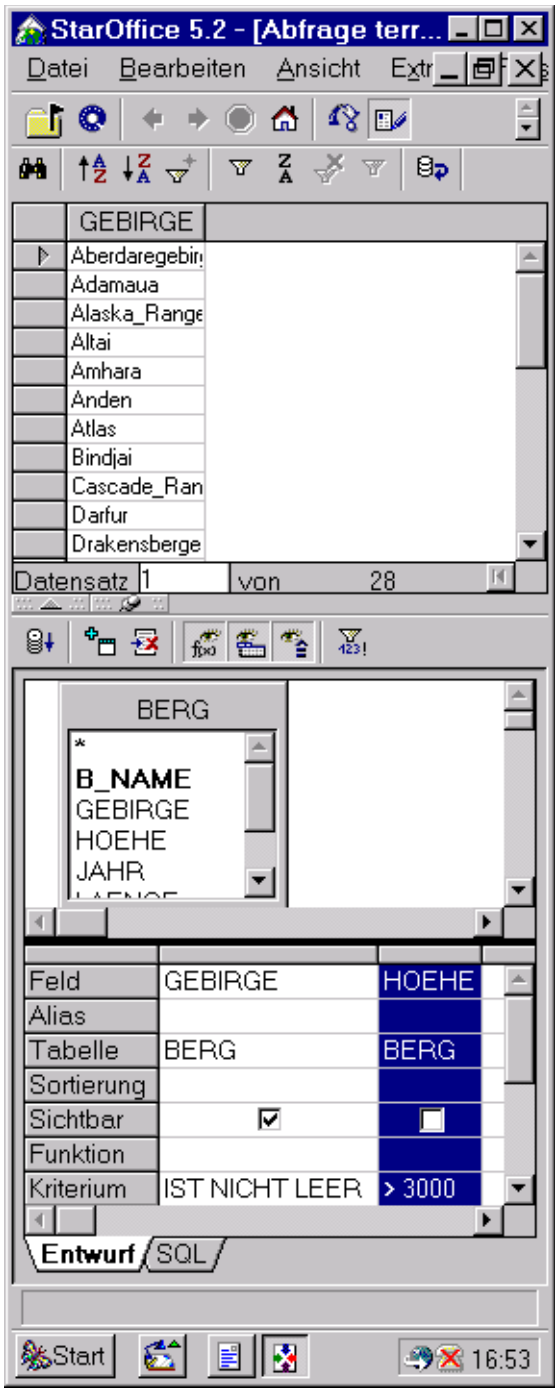


Abbildung 34

Abfrage: A009

```
SELECT F_NAME, LAENGE, MEER FROM FLUSS FLUSS WHERE LAENGE > 1000 AND (
MEER = 'Ostsee' OR MEER = 'Nordsee' )
```

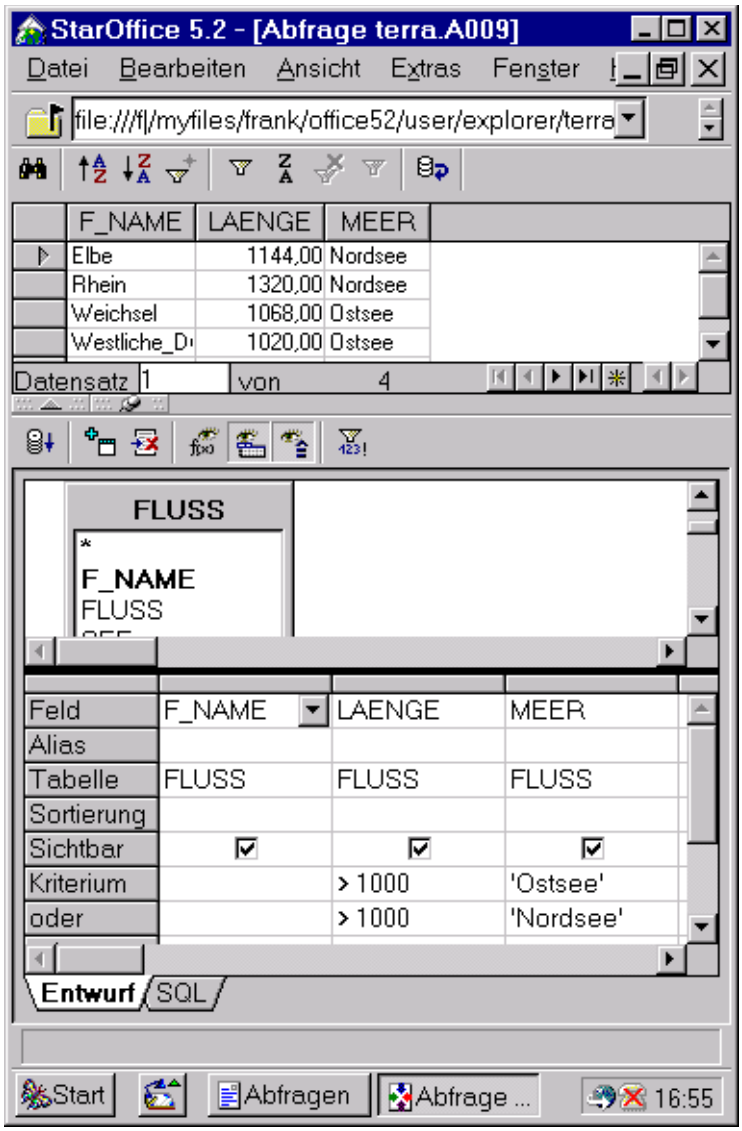


Abbildung 35

Abfrage: A010

```
SELECT W_NAME, FLAECHE FROM WUESTE WUESTE WHERE WUESTENART =
'Sandwueste' AND FLAECHE > 25000
```

Abfrage: A011

Aufgabe gestrichen - wird hier nicht weiter besprochen.

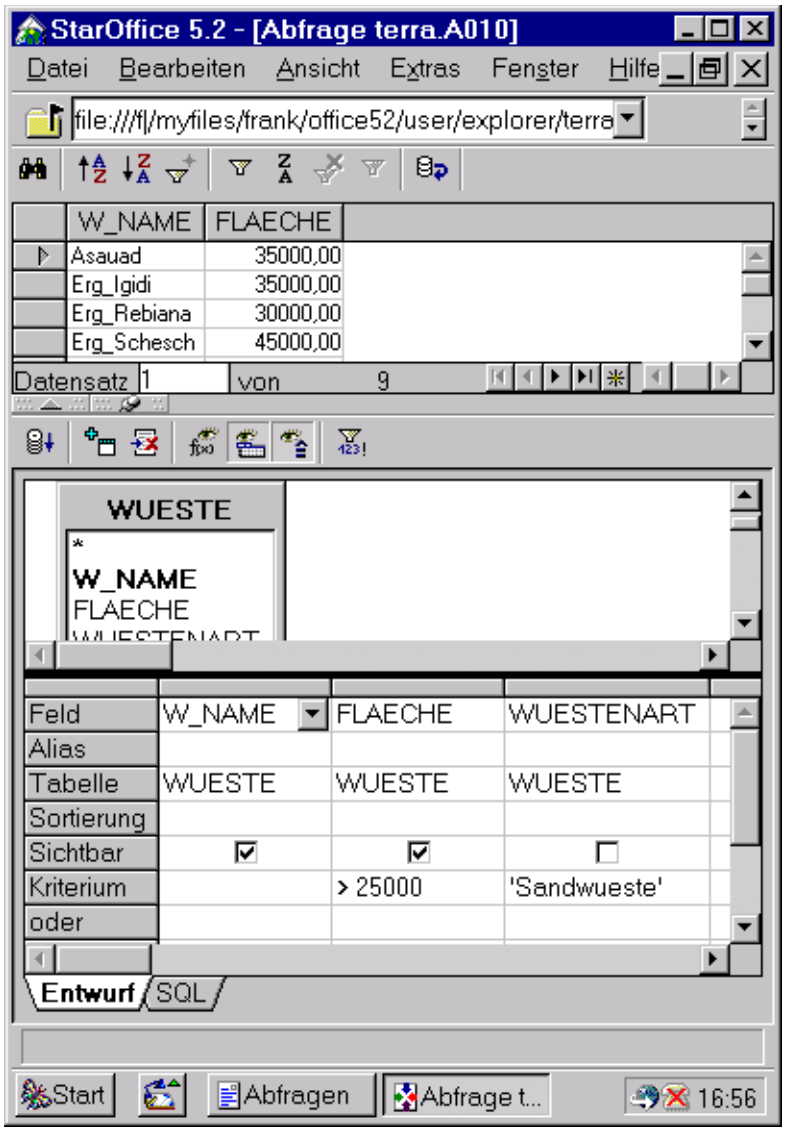


Abbildung 36

Abfrage: A012

```
SELECT LAND.L_NAME, LAND.EINWOHNER, (LAND.FLAECHE * UMFASST.PROZENT)
AS LFlaeche FROM UMFASST UMFASST, LAND LAND WHERE UMFASST.L_ID =
LAND.L_ID AND UMFASST.K_NAME = 'afrika'
```

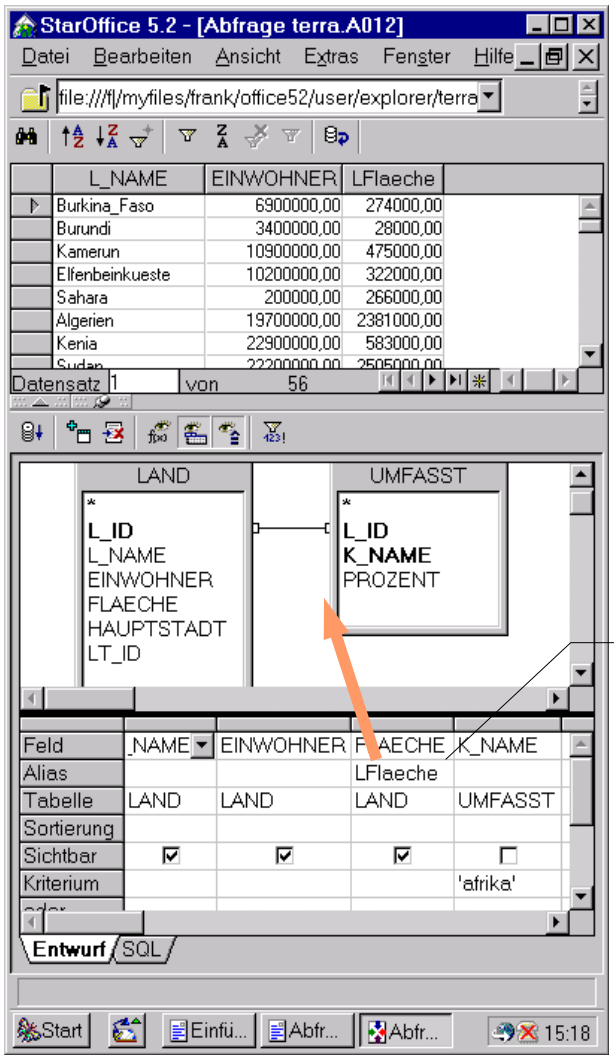


Abbildung 37

Um die beiden Tabellen zu verbinden, muss eine „Relation“ eingefügt werden. Das geschieht durch Ziehen des Attributes der einen Tabelle auf das zweite Attribut (mit der Maus).

Abfrage: A013

```
SELECT LAND.L_NAME, LAND.EINWOHNER, LAND.FLAECHE AS LFlaeche,
KONTINENT.FLAECHE * 1000000 AS KFlaeche,
(LAND.FLAECHE*UMFASST.PROZENT) / KFlaeche AS Flaeche in Prozent FROM
UMFASST UMFASST, LAND LAND, KONTINENT KONTINENT WHERE UMFASST.L_ID =
LAND.L_ID AND UMFASST.K_NAME = KONTINENT.K_NAME AND UMFASST.K_NAME =
'afrika' ORDER BY LAND.FLAECHE DESC
```

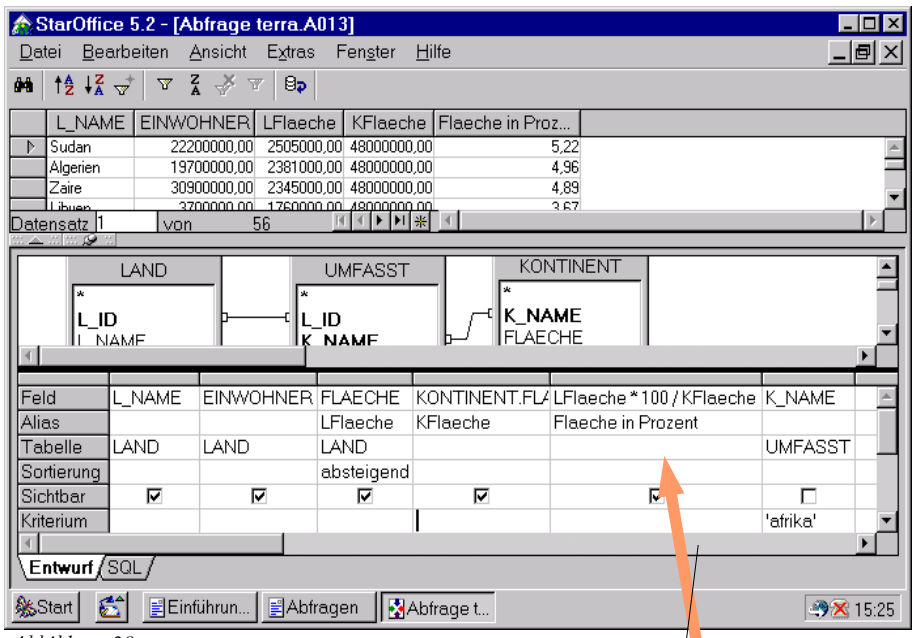


Abbildung 38

Berechnungen unter Verwendung von selbst definierten Alias-Namen.

Abfrage: A013a (Variante I)

```
SELECT LAND.L_NAME, LAND.EINWOHNER, (LAND.FLAECHE*UMFASST.PROZENT) AS
LFlaeche, KONTINENT.FLAECHE * 1000000 AS KFlaeche, LFlaeche / KFlaeche
AS PFlaeche, PFlaeche * 100 AS Fläche in Prozent FROM LAND LAND,
UMFASST UMFASST, KONTINENT KONTINENT WHERE LAND.L_ID = UMFASST.L_ID
AND KONTINENT.K_NAME = UMFASST.K_NAME AND UMFASST.K_NAME = 'afrika'
ORDER BY LAND.FLAECHE DESC
```

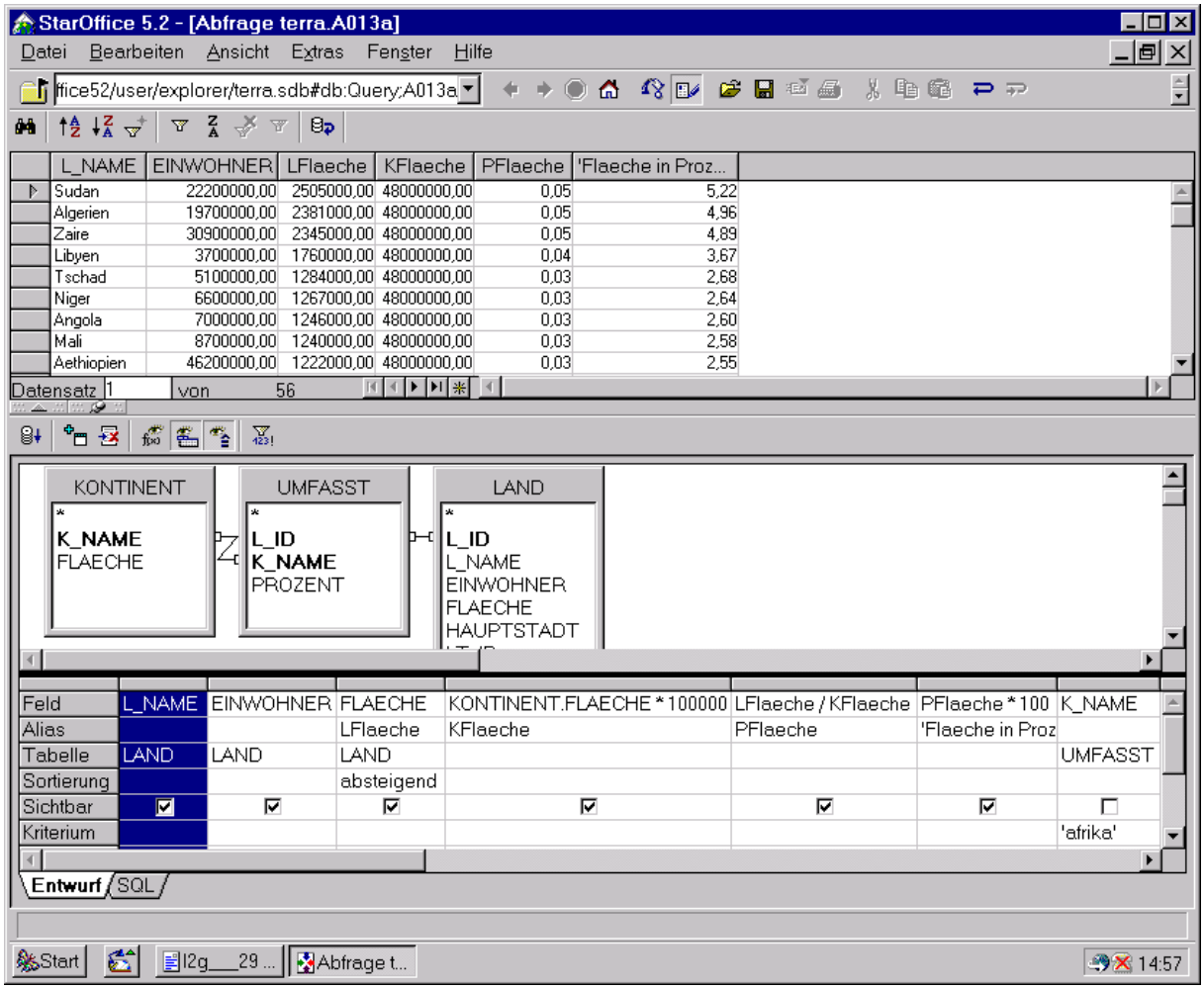


Abbildung 39

Abfrage: A013b (Variante II)

```
SELECT LAND.L_NAME, LAND.EINWOHNER, (LAND.FLAECHE*UMFASST.PROZENT) /
( KONTINENT.FLAECHE * 1000000 )[13] AS Fläche in Prozent FROM LAND LAND,
UMFASST UMFASST, KONTINENT KONTINENT WHERE LAND.L_ID = UMFASST.L_ID
AND KONTINENT.K_NAME = UMFASST.K_NAME AND UMFASST.K_NAME = 'afrika'
```

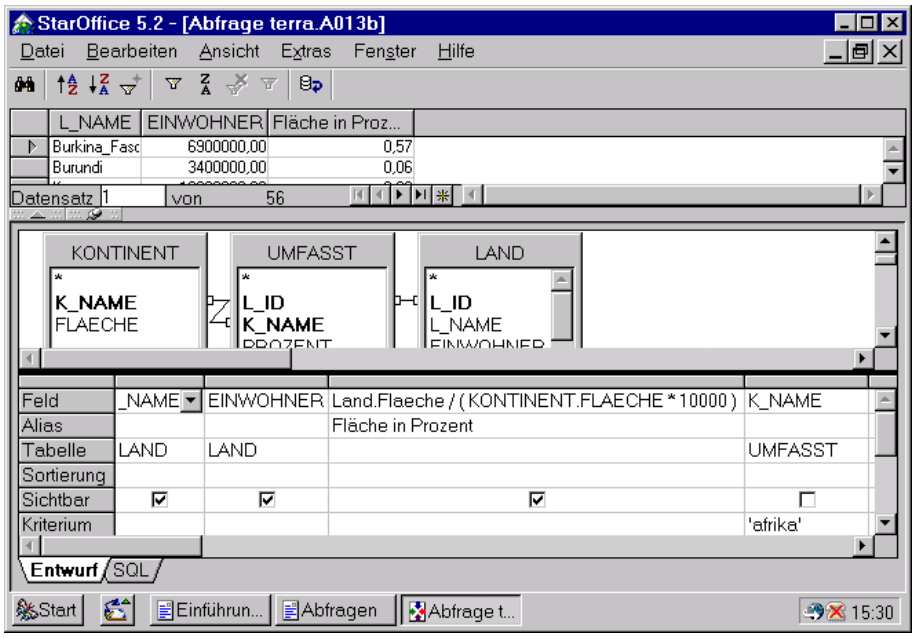


Abbildung 40

13 Berechnung durch direkten Tabellenbezug. Es wurden keine Aliasnamen definiert.

Abfrage: A013c (Variante IV)

```
SELECT LAND.L_NAME, LAND.EINWOHNER, (LAND.FLAECHE*UMFASST.PROZENT) / (
SELECT flaeche * 10000 FROM kontinent WHERE k_name = 'afrika' ) AS
'Fläche in Prozent', LAND.FLAECHE FROM LAND LAND, UMFASST UMFASST
WHERE LAND.L_ID = UMFASST.L_ID AND UMFASST.K_NAME = 'afrika' ORDER BY
LAND.FLAECHE DESC
```

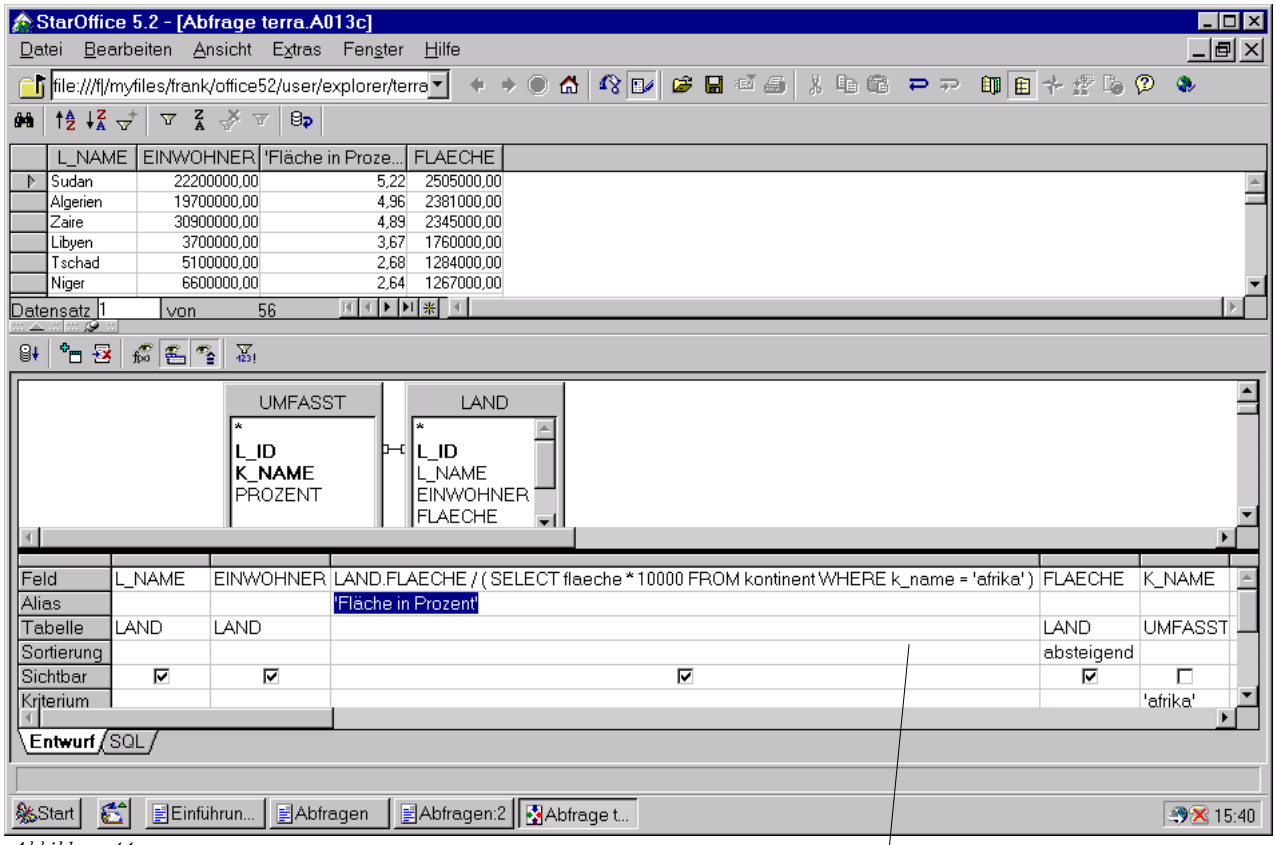


Abbildung 41

Verwendung einer Sub-Query. Dabei bezieht sich die Abfrage auf die Ergebnisse, die aus der Tabelle Kontinent entnommen werden. Die Tabelle Kontinent ist damit kein direkter Bestandteil der Abfrage und taucht auch nicht mehr im Tabellenfenster auf.