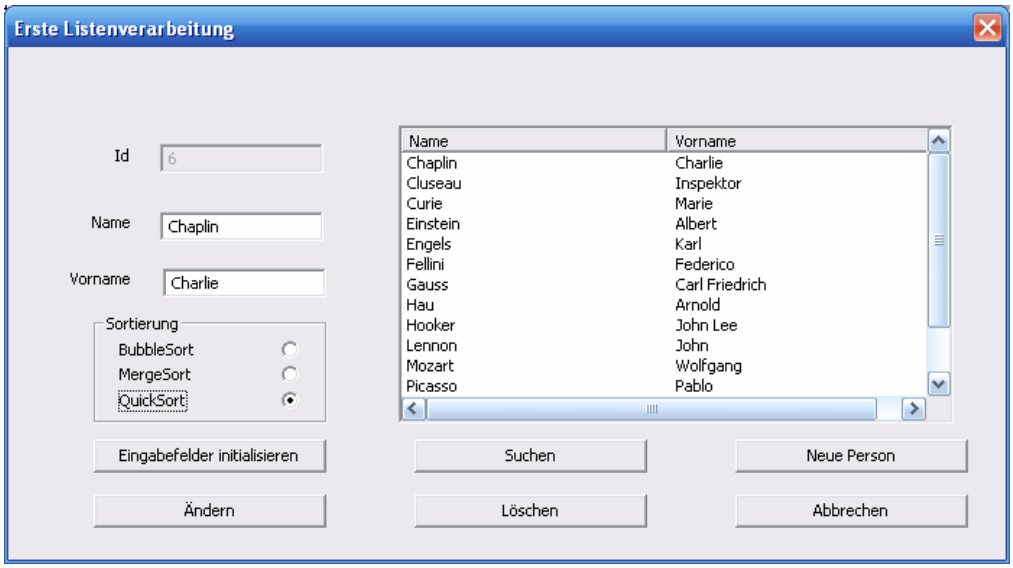
Lösungen der Übungsaufgaben von Kapitel 4

1. Machen Sie einen Entwurf für die Oberfläche einer Applikation, mit der Sie Ihr Adressbuch verwalten wollen. Wenn Sie können, entwerfen und realisieren Sie sogar einen Prototypen in einer Ihnen genehmen Entwicklungsumgebung wie z.B.NET. Was für Funktionen brauchen Sie? Welche Zugänge zu den Daten (Einzelansichten, Listen usw.) sehen Sie vor? Definieren Sie Ereignisse wie das Betätigen eines Buttons mit dem Titel „*Löschen*“ und den vollständigen Ablauf Ihrer Applikation nach solch einem Ereignis.



Ablauf der Verarbeitung Löschen:

- Ein gültiger Satz muss ausgewählt sein.
- Nach dem Betätigen des Buttons Löschen wird der Benutzer gefragt, ob er „wirklich sicher“ ist, dass dieser Satz gelöscht wird.
- Der Satz wird aus der Datenbank gelöscht
- Die verbleibenden Sätze werden angezeigt.

2. Was Sie in Aufgabe 1 entworfen haben, ist **Ihre** Gestaltung einer externen Ebene für die Tabelle Adressbuch. Fragen Sie einen Freund oder eine Freundin, wie er oder sie solch eine externe Ebene gestalten würde. Sicherlich anders als Sie es getan haben. Grundsätzlich gibt es für eine Datenbank im Allgemeinen sehr viele Applikationen, die mit dieser Datenbank arbeiten. Sie alle definieren jeweils eine eigenständige externe Ebene, die alle mit derselben konzeptionellen Ebene, mit demselben DBMS kommunizieren.

3. Überlegen Sie, ob es nicht Benutzer Ihrer Adressbuch-Applikation geben muss, die weniger (oder mehr) können dürfen als andere. Sollen andere Benutzer, die mit Ihrer Tabelle arbeiten, auch Sätze anlegen, verändern oder löschen können? Wie kann man ein solches benutzerspezifisches Berechtigungskonzept realisieren?

In fast allen Fällen, wo mehrere Benutzer auf eine Datenbank zugreifen können, ist es sinnvoll, verschiedene Benutzergruppen einzurichten:

- Administratoren, die neue Benutzer anlegen können, Passwörter vergeben und Rechte zuordnen und ändern können
- Benutzer, die die Daten der Datenbank nicht nur lesen, sondern auch vermittels Einfügen, Ändern oder Löschen manipulieren können
- Benutzer, die alle Sätze und auch alle Attribute einer Datei sehen können
- Benutzer, die nur bestimmte Sätze bzw. bestimmte Attribute einer Datenbank sehen können

Solche Benutzerrechte können in einer eigenen Berechtigungsdatei abgelegt werden, die dann beim Zugriff auf die Datenbank mit abgefragt wird. Sehr sinnvoll sind hierarchisch strukturierte Berechtigungskonzepte, die leichter zu verwalten sind als individuell ausgestaltete Profile.

4. Finden Sie andere Beispiele für Applikationen, wo es verschiedene Berechtigungsstufen für Benutzer einer Datenbank gibt. Wie ist das bei Anbietern von Produkten im Internet, wo das Artikelsortiment in Datenbanken gespeichert ist?

   Ganz offensichtlich sollte der Kunde oder interessierte „Besucher“ eines Online-Anbieters nur lesenden Zugriff auf dessen Produktdatenbank haben, während die entsprechenden Mitarbeiter „vor Ort“ beispielsweise neue Produkte und Produktbeschreibungen einfügen müssen, Preise und Mengenangaben ändern müssen und Artikel löschen müssen.

5. Machen Sie sich auch hier wieder klar, wie wichtig es ist, Fragen, welche die Korrektheit Ihrer Daten betreffen, durch das **eine** DBMS der konzeptionellen Ebene behandeln zu können und sie nicht jeder externen Ebene zur individuellen Verarbeitung zu überlassen. Stellen Sie sich vor, dass DBMS könnte nicht überprüfen, ob die Eingabe beim Attribut ***Name*** leer ist. Welche Funktionen Ihres Entwurfs aus Aufgabe 1 wären davon betroffen?

   Einfügen und Ändern

6. Angenommen, Sie richten an unsere Adressbuchtabelle die Anfrage:

   - Zeige alle Personen, deren Nachname mit Z anfängt und die in Deutschland wohnen

   Geben Sie drei verschiedene Möglichkeiten an, wie man diese Abfrage Schritt für Schritt in einer Programmiersprache wie COBOL, JAVA oder C++ programmieren könnte.

1. Möglichkeit

Lies den ersten Eintrag der Personentabelle

Prüfe, ob der Eintrag angezeigt werden soll und zeige ihn gegebenenfalls an

Lies den nächsten Eintrag der Datenbank

Solange man nicht am Ende der Datei angekommen ist

{

    Prüfe, ob der Eintrag angezeigt werden soll und zeige ihn gegebenenfalls an.

    Lies den nächsten Eintrag der Datenbank

}

2. Möglichkeit

Sortiere absteigend nach Anfangsbuchstaben des Nachnamens (also Z zuerst), bei gleichem Anfangsbuchstaben sortiere aufsteigend nach Land. (Beispielsweise mit Quicksort)

Lies den ersten Eintrag der Personentabelle

Wenn der Nachname nicht mit Z anfängt, beende die Verarbeitung

Solange gilt (man ist nicht am Ende der Datei angekommen und das Land ist nicht Deutschland)

{

Lies den nächsten Eintrag der Datenbank

}

Prüfe, ob der Eintrag angezeigt werden soll und zeige ihn gegebenenfalls an, Andernfalls beende die Verarbeitung.

Lies den nächsten Eintrag der Datenbank

Solange gilt (man ist nicht am Ende der Datei angekommen und der Anfangsbuchstabe des Nachnamens ist Z und das Land ist Deutschland)

{

zeige den Satz an.

Lies den nächsten Eintrag der Datenbank

}

3. Möglichkeit

Sortiere aufsteigend nach Land und bei gleichem Land absteigend nach Anfangsbuchstaben des Nachnamens (also Z zuerst). (Beispielsweise mit Quicksort)

Lies den ersten Eintrag der Personentabelle

Solange gilt (man ist nicht am Ende der Datei angekommen und das Land ist nicht Deutschland)

{

Lies den nächsten Eintrag der Datenbank

}

Prüfe, ob der Eintrag angezeigt werden soll und zeige ihn gegebenenfalls an, Andernfalls beende die Verarbeitung.

Lies den nächsten Eintrag der Datenbank

Solange gilt (man ist nicht am Ende der Datei angekommen und der Anfangsbuchstabe des Nachnamens ist Z und das Land ist Deutschland)

{

zeige den Satz an.

Lies den nächsten Eintrag der Datenbank

}

Welches wäre das optimale Vorgehen? Ist das stets ein und dasselbe Verfahren oder hängt es vom Datenbestand ab, welches Vorgehen schneller ist?

Zunächst gilt: Bei großen Datenbeständen, die nur wenig Sätze enthalten, die den angegebenen Bedingungen genügen, ist natürlich die Bearbeitung sortierter Sätze viel performanter. Denn ohne Sortierung muss man auf jeden Fall die Tabelle bis zum Ende durchprüfen, während die Prüfung sortierter Bestände sofort nach Erreichen des ersten ungültigen Satzes beendet werden kann. Andererseits wird bei einer Sortierung jeder Satz einer Tabelle meistens mehrmals „angefasst“, dieser Nachteil kann nur aufgewogen werden, wenn es für die zu sortierenden Felder so genannte Indexdateien gibt, die einem die Sortierung ermöglichen ohne dass der gesamte Satz in den Hauptspeicher geladen wird. (Darüber lernen Sie erst im 9. Kapitel mehr)

Da wir mit dem Buchstaben Z einen markanten Punkt in der Sortierungsreihenfolge haben (der erste Punkt bei absteigender Sortierung) scheint die zweite Möglichkeit auf jeden Fall die beste Möglichkeit sein, falls der Datenbestand groß genug ist und falls Sortierungsunterstützungen wie Indexdateien zur Verfügung stehen

Wie könnte man einen Optimizer für diese Abfrage konzipieren?

Falls der Datenbestand genügend groß ist (Anzahl der Sätze multipliziert mit der Größe eines einzelnen Satzes) und falls Indexdateien für die Attribute Name und Land vorhanden sind gehe nach Methode 2 vor, andernfalls nach Methode 1.

7. Betrachten Sie noch einmal das Beispiel zur Wiederherstellung von Daten. Wir hatten da den Fall konstruiert, dass ein Benutzer den Ort *Frankfurt* aus der Ortstabelle und korrespondierend dazu alle Personen aus Frankfurt aus der Personentabelle löschen will.

   - Dieser Löschprozess wurde so organisiert, dass zuerst der Satz aus der Ortstabelle und dann die Sätze aus der Personentabelle gelöscht wurden.

   Ein Systemabsturz während dieser Verarbeitung konnte dann den Datenbestand inkonsistent machen.

Gibt es eine andere Organisation dieses Löschvorgangs, bei der ohne den Einsatz eines Transaktions-Managers bei allen möglichen Abbrüchen dieser Verarbeitung die Konsistenz des Datenbestands nie gefährdet ist?

> Ja. Man löscht **zuerst** die Sätze aus der Personentabelle und dann den Satz aus der Ortstabelle.

8. Nehmen Sie an, ein Warenhaus hat eine Tabelle ARTIKEL für sein Artikelsortiment, in der für jeden Artikel die ***Bestandsmenge*** geführt wird. Außerdem gibt es eine Tabelle BESTELLUNGEN, in der jede Bestellung eines Artikels mit der zugehörigen ***Bestellmenge*** gespeichert ist. Betrachten Sie nun eine Verarbeitung „BESTELLUNG WIRD AUSGEFÜHRT“, bei der sowohl der Satz in der Tabelle BESTELLUNGEN als „*verarbeitet*“ gekennzeichnet wird als auch die ***Bestandsmenge*** in der Tabelle ARTIKEL um die ***Bestellmenge*** verringert wird.

   Gibt es eine Organisation dieser Verarbeitung, bei der ohne den Einsatz eines Transaktions-Managers bei allen möglichen Abbrüchen dieser Verarbeitung die Konsistenz des Datenbestands nie gefährdet ist?

   > Die Aufgabe ist schlecht formuliert. Ich habe es zu spät gemerkt und muss nun leider bis zur dritten Auflage warten, bis ich es korrigieren kann. Die Konsistenz (also pure Datenbanklogik) ist bei diesem Vorgang nie gefährdet, egal wann es Abstürze gibt. Die **inhaltliche Korrektheit** des Datenbestandes ist allerdings stets in Gefahr, wenn es zu einem Absturz nach der ersten Verarbeitung (Bestellung wird als verarbeitet gekennzeichnet **oder** Bestandsmenge wird verringert) und vor der zweiten Verarbeitung (Bestandsmenge wird verringert **oder** Bestellung wird als verarbeitet gekennzeichnet) kommt. Da hilft keine clevere Umorganisation.

9. Geben Sie noch einmal in eigenen Worten eine Zusammenfassung über die hier aufgeführten Verantwortlichkeiten eines DBMS.

http://www.springer.com/978-3-8351-0163-0

Datenbanken
Theorie, Entwurf und Programmierung relationaler Datenbanken
Schubert, M.
2007, XII, 344 S. Mit Online-Extras., Softcover
ISBN: 978-3-8351-0163-0