

## Übungen und Praktikum zu Datenbanken und Informationssysteme

### A PROGRAMMIERUNG VON DATENBANKZUGRIFFEN

#### 1. Daten lesen mit JDBC

Schreiben Sie eine Java-Anwendung, die die Tabelle `Books` in der Datenbank `Azamon` ausgibt. Verwenden Sie dabei die SQL-Anweisung `select * from Books`. Angezeigt werden soll eine Kopfzeile mit den Namen der in der Tabelle enthaltenen Spalten, sowie der Inhalt der Tabelle.

#### 2. Daten lesen mit ADO.NET

Schreiben Sie ein `C#`-Programm, das die Tabelle `Books` der Datenbank `Azamon` ausgibt. Verwenden Sie auch hier die SQL-Anweisung `select * from Books`. Angezeigt werden soll eine Kopfzeile mit den Namen der in der Tabelle enthaltenen Spalten, sowie der Inhalt der Tabelle. Zum Lesen der Daten verwenden Sie einen `DataReader`.

#### 3. Daten lesen mit der Java Persistence API

Schreiben Sie ein Java-Programm, das die Bücher in der Tabelle `Books` ausgibt. Angezeigt werden sollen die Inhalte der Zeilen der Tabelle.

Sie benötigen eine Implementierung der JPA, zu finden z.B. bei <http://www.oracle.com/technology/products/ias/toplink/jpa/index.html>.

Es folgen drei alternative Blöcke mit je zwei Aufgaben zu JDBC, ADO.NET bzw der Java Persistence API. Sie entscheiden sich für *eine* der Techniken.

### Block 1: JDBC

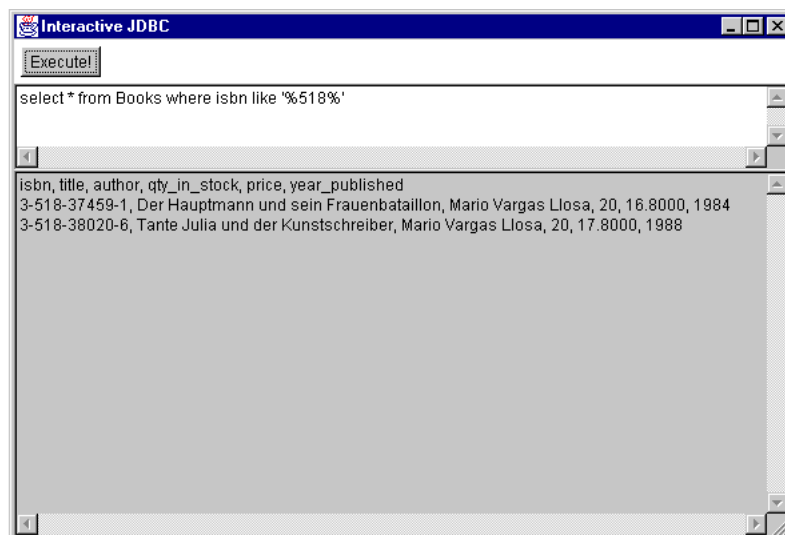


Abbildung 1: Interactive JDBC

#### 4. Interaktives SQL mit JDBC

Schreiben Sie eine Java-Anwendung *Interactive JDBC*, mit der man beliebige SQL-Anweisungen (anfragende und modifizierende Anweisungen) durchführen kann.

Es soll möglich sein, in einem Dialog die Datenquelle zu wählen, mit der man sich verbinden möchte.

Das Programm sollte etwa so aussehen wie das in Abb. 1

#### 5. Datenbankexport mit JDBC

Schreiben Sie eine Java-Anwendung, mit der man einen einfachen Datenbank-Export durchführen kann. Man gibt die Tabellen an, die man exportieren möchte und das Programm erzeugt zwei SQL-Skripten: eines, das die DDL-Anweisungen zum Erzeugen der Tabellen enthält, und eines, das die Insert-Anweisungen zum Füllen der Tabellen enthält.

Zusatzaufgabe: Es sollen auch Anweisungen, die die referentielle Integrität betreffen, erzeugt werden.

### Block 2: ADO.NET

#### 6. Interaktives SQL mit ADO.NET

Schreiben Sie eine C#-Anwendung *Interactive ADO.NET*, mit der man beliebige SQL-Anweisungen (anfragende und modifizierende Anweisungen) durchführen kann.

Es soll möglich sein, in einem Dialog die Datenquelle zu wählen, mit der man sich verbinden möchte.

Das Programm sollte ähnlich aussehen wie das Java-Programm in Abb. 1

#### 7. TableViewer mit ADO.NET

Schreiben Sie eine C#-Anwendung *TableViewer*, mit der man eine beliebige Tabelle einer Datenbank durchblättern kann.

Das Programm soll etwa so aussehen, wie in Abb 2.

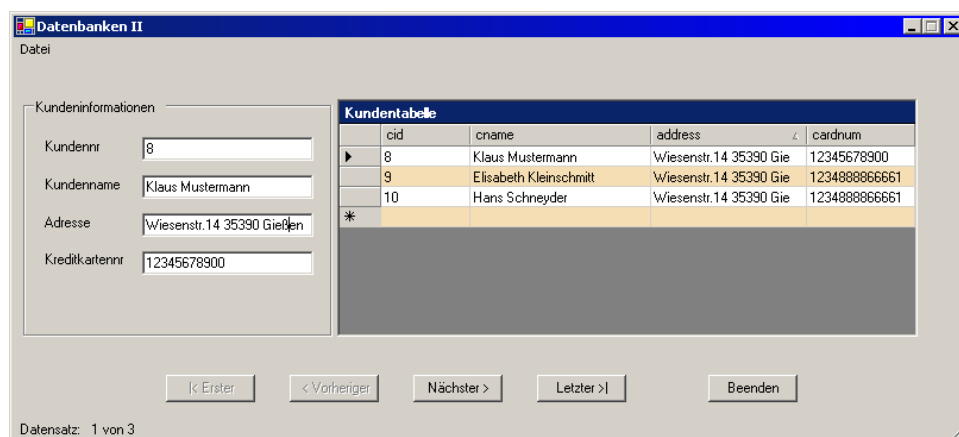


Abbildung 2: TableViewer

Verwenden Sie für die Implementierung das DataSet und das DataGrid von ADO.NET.

Optional: Erweitern Sie den TableViewer so, dass man die Daten auch ändern kann.

### Block 3: JPA

#### 8. Objekt-relationales Mapping mit JPA

Gegeben ist die Datenbank „amazon“ mit folgendem Schema:

```
author      char(40),
price       numeric(6,2),
year_published integer
);

create table Customers
(
  cid        integer primary key,
  cname     char(40),
  address   char(200),
  cardnum   char(16)
);

create table Orders
(
  ordernum  integer primary key,
  cid       integer references Customers(cid),
  order_date date
);

create table Orderitems
(
  ordernum  integer references Orders(ordernum),
  isbn      char(13) references Books(isbn),
  qty       integer,
  primary key( ordernum, isbn )
);
```

Schreiben Sie ein Java-Programm, das mit Hilfe der JPA es ermöglicht, einen Kunden und zum Kunden Aufträge anzulegen.

#### 9. Objektstrukturen laden mit JPA

Schreiben Sie ein Java-Programm, das auf Basis des OR-Mappings aus der vorherigen Aufgabe alle Kunden samt ihren Aufträgen anzeigt.

---

Weitere Aufgaben zum Thema Programmieren von Datenbank-Zugriffen:

#### 10. Metainformationen mit JDBC

In JDBC gibt es zwei Interfaces für Metainformationen:  
DatabaseMetaData und ResultSetMetaData.

Beschreiben Sie jeweils, welche Informationen über diese Interfaces bereitgestellt werden und geben Sie an, welchen *Konzepten* einer SQL-Datenbank diese Interfaces entsprechen.

(a) DatabaseMetaData

(b) ResultSetMetaData

### 11. DBMS-unabhängige Programmierung in JDBC und ADO.NET

Man möchte APIs zum Zugriff auf Datenbankmanagementsysteme oft gerne so verwenden, dass man möglichst unabhängig vom jeweiligen DBMS ist: das übersetzte, ausführbare Programm soll mit jedem beliebigen DBMS verwendet werden können. Erläutern Sie die grundlegenden Konzepte.

- (a) Welchen Mechanismus sieht JDBC vor, um die Unabhängigkeit vom DBMS zu erreichen?
- (b) Welchen Mechanismus sieht ADO.NET (ab Version 2.0) vor, um die Unabhängigkeit vom DBMS zu erreichen?
- (c) Worauf muss man außer den in (a) und (b) genannten Mechanismen achten, um Unabhängigkeit vom DBMS zu erreichen?

### B TRANSAKTIONEN UND NEBENLÄUFIGKEIT

#### 12. Dauer serieller Transaktionen

100 Kunden einer Bank wollen gleichzeitig an 100 verschiedenen Geldautomaten Geld abheben. Nehmen wir an, dass die Transaktion auf dem Rechner der Bank 0,3 Sekunden dauert und dass die Transaktionen *seriell* durchgeführt werden.

Wie lange muss der Kunde warten, dessen Abhebung als letzte bearbeitet wird? Wie lange ist die durchschnittliche Antwortzeit des Systems?

#### 13. Serialisierbarkeit

Geben Sie an, welche der folgenden Abläufe serialisierbar sind – mit Begründung

- (a)  $r_1(x); r_2(y); r_1(z); r_3(z); r_2(x); r_1(y);$
- (b)  $r_1(x); w_2(y); r_1(z); r_3(z); w_2(x); r_1(y);$
- (c)  $r_1(x); w_2(y); r_1(z); r_3(z); w_1(x); r_2(y);$
- (d)  $r_1(x); r_2(y); r_1(z); r_3(z); w_1(x); w_2(y);$
- (e)  $r_1(x); r_2(y); w_2(x); w_3(x); w_3(y); r_1(y);$
- (f)  $w_1(x); r_2(y); r_3(z); r_1(x); w_2(y);$
- (g)  $r_1(z); w_2(x); r_2(y); w_1(x); w_3(z); w_1(y); r_3(x);$

#### 14. 2PL und Serialisierbarkeit

Beweisen Sie folgende Aussage:

Gegeben sei ein Ablauf, der dem 2PL-Protokoll entspricht. Dann ist ein äquivalenter serieller Ablauf derjenige, bei dem die Transaktionen in der Reihenfolge ausgeführt werden, wie sie im gegebenen Ablauf den ersten Lock freigeben.

#### 15. Beispiele für Abläufe – 2PL?

Geben Sie bei den folgenden Abläufen an, ob sie dem 2-Phasen-Lock-Protokoll genügen oder sogar dem strikten 2PL. ( $b_i$  steht für den Beginn der Transaktion  $T_i$ ,  $c_i$  für Commit von  $T_i$ .)

(a)  $b_1; l_1(x); r_1(x); r_1(y); l_1(y); l_1(z); w_1(x); w_1(y); u_1(z); u_1(x); c_1;$

nicht 2PL    2PL, nicht strikt    striktes 2PL

(b)  $b_1; l_1(x); r_1(x); l_1(y); b_2; l_2(z); w_1(x); w_1(y); w_2(z); u_1(y); r_1(x);$   
 $u_1(x); u_2(z); c_2; c_1;$

nicht 2PL    2PL, nicht strikt    striktes 2PL

(c)  $b_1; b_2; l_1(y); w_1(y); l_2(x); r_1(x); u_1(y); u_2(x); c_1; c_2;$

nicht 2PL    2PL, nicht strikt    striktes 2PL

(d)  $b_1; l_1(x); l_1(y); l_1(z); r_1(x); u_1(x); w_1(y); u_1(y); r_1(z); u_1(z); c_1;$

nicht 2PL    2PL, nicht strikt    striktes 2PL

(e)  $b_2; b_3; b_1; l_1(x); r_1(x); l_2(y); r_2(y); w_1(x); u_1(x); l_2(z); w_2(y); w_2(z);$   
 $c_1; l_3(x); w_3(x); u_3(x); u_2(z); u_2(y); c_2; c_3$

nicht 2PL    2PL, nicht strikt    striktes 2PL

## 16. Protokoll für Modus-Sperren

In der Vorlesung haben wir ein simples Modell eines Datenbanksystems behandelt, das binäre Sperren  $l_i(x)$  (Lock) und  $u_i(x)$  verwendet. Wir haben gesehen, wie ein Sperrprotokoll definiert wird, in dem man das Verhalten der Transaktionen einerseits und des Systems, genauer des Lockmanagers andererseits definiert.

Ein etwas realistischeres Modell eines Datenbanksystems verwendet sogenannte Modus-Sperren:

$lr_i(x)$  bezeichnet einen Read-Lock der Transaktion  $T_i$  auf dem Datenobjekt  $x$  (besser ist die Bezeichnung „shared lock“).

$lw_i(x)$  bezeichnet einen Write-Lock der Transaktion  $T_i$  auf dem Datenobjekt  $x$  (besser ist die Bezeichnung „exclusive lock“).

$u_i(x)$  bezeichnet die Freigabe einer Sperre auf Datenobjekt  $x$  durch Transaktion  $T_i$ .

Beschreiben Sie das Sperrprotokoll für solche Modus-Sperren.

## 17. Verklemmung & Wartegraph

Gegeben sei folgende Situation: Die Transaktionen  $T_1 - T_7$  halten Sperren auf den Datenobjekten  $x_i$  und warten auf die Freigabe der Sperren auf Datenobjekten  $x_i$  entsprechend folgender Aufstellung.

Transaktion	hat Lock auf	wartet auf
$T_1$	$x_2$	$x_1, x_3$
$T_2$	$x_3, x_{10}$	$x_7, x_8$
$T_3$	$x_8$	$x_4, x_5$
$T_4$	$x_7$	$x_1$
$T_5$	$x_1, x_5$	$x_3$
$T_6$	$x_4, x_9$	$x_6$
$T_7$	$x_6$	$x_5$

Zeichnen Sie einen Wait-for-Graph und ermitteln Sie Deadlocks.

## 18. Auflösung des Deadlocks

Verwenden Sie den Wait-for-Graph aus der vorherigen Aufgabe und wenden Sie den Algorithmus aus der Vorlesung an, um die Deadlocks zu erkennen und aufzulösen.

## 19. DBMS installieren

In der folgenden Aufgabe möchten wir die Phänomene erzeugen, die im SQL-Standard der Definition der Isolationenlevel zugrunde liegen. Diese Aufgabe sollten Sie in einer Gruppe von vier Studierenden durchführen.

Zur Vorbereitung der Aufgabe soll jeder von Ihnen ein Datenbankmanagementsystem auf seinem Rechner installieren. Wir möchten folgende DBMS in Bezug auf das Verhalten bei Isolationenleveln untersuchen:

- PostgreSQL <http://www.postgresql.org/>
- MySQL <http://www.mysql.de/>
- Microsoft SQL Server Express <http://www.microsoft.com/germany/express/products/database.aspx>
- Oracle Database Express Edition <http://www.oracle.com/technetwork/database/express-edition/overview/index.html>
- Apache Derby <http://db.apache.org/derby>
- IBM DB 2 <http://www-01.ibm.com/software/data/db2/express/>

Installieren Sie also eines dieser DBMS.

## 20. Phänomene

Die Isolationenlevel in SQL-Datenbanksystemen werden dadurch definiert, dass angegeben wird, welche Phänomene der Beeinflussung von Transaktionen garantiert ausgeschlossen werden. In dieser Aufgabe sollen Sie in den wechselseitigen Einfluss zweier Transaktionen auf den verschiedenen Isolationenleveln erproben.

Sie arbeiten in einer Vierergruppe und jeder von Ihnen verwendet eines der in der vorigen Aufgabe genannten DBMS (natürlich jeder ein anderes).

Legen Sie eine Datenbank an und schreiben Sie mit JDBC oder ADO.NET kleine Programme, mit denen Sie die Phänomene erzeugen können. Spielen Sie alle Kombinationsmöglichkeiten der Isolationenlevel durch.

Vergleichen Sie die Ergebnisse und notieren Sie die Unterschiede der DBMS in Bezug auf die Synchronisation konkurrierender Zugriffe.

## 21. Eine endlose Transaktion

Ein etwas abwegiges Beispiel soll die Konzeption der Isolationenlevel veranschaulichen: Gegeben sei eine Relation PC(model, speed, ram, price) und wir nehmen an, eine Transaktion enthält eine Endlosschleife, die immerzu nachschaut, ob es mittlerweile einen PC mit 2 Gigahertz unter 1000 Euro gibt. Währenddessen finden andere Transaktionen statt, die mit verschiedenen Isolationenleveln laufen, etwa eine solche, die ein gesuchtes PC-Modell in die Relation einfügt.

Pseudocode:

```
begin transaction;  
forever  
{
```

```
sleep( 1 sec );  
select * from PC;  
untersuche ergebnismenge;  
if ( found ) goto ende;  
}  
ende:  
commit();  
ausgabe;
```

Erläutern Sie was passiert, wenn die endlose Transaktion in folgendem Isolationslevel läuft:

- (a) SERIALIZABLE
- (b) REPEATABLE READ
- (c) READ COMMITTED
- (d) READ UNCOMMITTED

## 22. Wirkung des Isolationslevels

Wir gehen von folgender Relation aus: M(MId, Gehalt). In der Relation sind zwei Tupel gespeichert: (A, 1000) und (B, 2000). Nun werden folgende Transaktionen durchgeführt:

```
Transaktion 1:  
begin transaction;  
update M set Gehalt = Gehalt*2 where MId = 'A';  
update M set Gehalt = Gehalt+100 where MId = 'B';  
commit;
```

```
Transaktion 2:  
begin transaction;  
select sum(Gehalt) as G1 from M;  
select sum(Gehalt) as G2 from M;  
commit;
```

Die erste Transaktion wird im Isolationslevel SERIALIZABLE durchgeführt.

Berechnen Sie alle möglichen Werte für G1 und G2, wenn

- (a) die zweite Transaktion mit dem Isolationslevel SERIALIZABLE durchgeführt wird.
- (b) die zweite Transaktion mit dem Isolationslevel READ COMMITTED durchgeführt wird.
- (c) die zweite Transaktion mit dem Isolationslevel READ UNCOMMITTED durchgeführt wird.

## 23. Phantome

Gegeben sei ein Datenbankschema mit zwei Tabellen T1 und T2, die jeweils drei Attribute a1, a2, a3 haben.

Betrachten Sie die SQL-Anweisung

```
select T1.a1, T2.a1  
from T1, T2  
where T1.a2 = T2.a2  
and T1.a3 = 5 and T2.a3 = 7
```

Formulieren Sie eine SQL-Anweisung, die zu einem Phantom führen kann.

**24. Snapshot-Isolation**

Erklären Sie, weshalb bei Snapshot-Isolation folgende Phänomene nicht auftreten können:

- (a) Dirty Reads
- (b) Lost Updates
- (c) Nonrepeatable Reads
- (d) Phantom Rows

Ist Snapshot-Isolation mit dem Isolationslevel SERIALIZABLE identisch?

**25. SERIALIZABLE**

Gegeben seien zwei nebenläufige Transaktionen  $T_1$  und  $T_2$ . Beweisen Sie folgende Aussage:

Wird  $T_1$  im Isolationslevel SERIALIZABLE ausgeführt und  $T_2$  in einem beliebigen Isolationslevel, dann sieht  $T_1$  entweder alle Änderungen, die  $T_2$  gemacht hat oder keine.

**26. READ COMMITTED**

Wir haben in der Vorlesung zwei Implementierungen des Isolationslevels READ COMMITTED kennengelernt: durch Sperrmechanismen und durch Multiversionierung. Konstruieren Sie ein Beispiel, bei dem die beiden Implementierungen unterschiedliche Ergebnisse haben.

**27. Statistische Auswertung**

Eine Transaktion für eine statistische Auswertung liest Daten aus der Datenbank, die im vergangenen Monat eingegeben wurden und erstellt aus diesen Daten einen Bericht.

Welches Isolationslevel kann man für diese Transaktion verwenden?

**28. Kompensatorische Transaktionen**

Geben Sie zu folgenden Datenbankänderungen an, ob es eine kompensatorische Transaktion gibt. Wenn ja, was müsste sie tun?

- (a) Erhöhung des Gehalts aller Professoren um 10%.
- (b) Erhöhung des Gehalts aller Mitarbeiter um 10%, sofern sie weniger als 3000 Euro verdienen.
- (c) Setze die Note von Student Max auf 2.
- (d) Füge einen Datensatz mit der Matrikelnummer (Primärschlüssel) 65432 und den Angaben 'Max', 'Schneider' für Vorname, Name ein.
- (e) Setze einen Wert auf das Quadrat des bisherigen Wertes.

**29. Synchronisation in der Java Persistence API**

Lesen Sie das Kapitel 3.4 der Spezifikation der Java Persistence API 1.0 („Optimistic Locking and Concurrency“). Quelle: <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>

Vergleichen Sie mit dem Kapitel 3.4 der Version 2.0 Quelle: <http://jcp.org/aboutJava/communityprocess/final/jsr317/index.html>

### 30. Persistenz von Objekten – ein Code-Beispiel

In SQL hat man die Anweisungen INSERT und UPDATE. In einem Programm möchte man diesen Unterschied oft verbergen. Hat man etwa eine Klasse Kunde mit den Attributen kNr und kName, dann möchte man eine Methode save der Klasse Kunde zum Speichern von Objekten programmieren, die selbst überprüft, ob in der Datenbank ein INSERT oder ein UPDATE nötig ist.

Wir nehmen an, dass in der Datenbank eine Tabelle Kunde mit den Feldern Knr (Primärschlüssel) und Kname angelegt ist.

Neulich habe ich zu diesem Thema in einer *Multiuser-Anwendung* folgenden Code gesehen (vereinfachter Pseudocode):

Pseudocode für Methode save der Klasse Kunde:

```
Connection con = DriverManager.getConnection(...);
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(
    "SELECT Knr FROM Kunde WHERE Knr = " + this.kNr );

boolean found =
    ... wenn in rs die Knr gefunden wird, wird die Variable found
    ... auf true gesetzt

if ( found ) {
    stmt.executeUpdate( "UPDATE Kunde set Kname = " + this.kName +
        "WHERE Knr = " + this.kNr );
} else {
    stmt.executeUpdate( "INSERT into Kunde values( " + this.kNr +
        ", '" + this.kName + "'" );
}

con.close();
```

Aufgabe:

- (a) Worin besteht der konzeptionelle Fehler in diesem Vorgehen?
- (b) Wie kann man das Vorgehen verbessern?

## C VERTEILTE DATENBANKEN

### 31. Unterschied horizontaler und vertikaler Zerlegung

Beschreiben Sie den Unterschied zwischen *horizontaler* und *vertikaler Zerlegung* in einer verteilten Datenbank. Durch welche relationalen Operatoren werden die beiden Typen der Zerlegung in der Regel realisiert: beschreiben Sie die Zerlegung und das Zusammenführen der Daten als relationale Operationen.

### 32. Horizontale Zerlegung einer Tabelle

Ein Unternehmen verwendet in drei Werken Teile. Die Teile mit den Nummern 1 bis 300 werden in Werk I verwendet, in Werk II werden die Teile mit den Nummern 301 bis 500, aber mit Ausnahme von 399 eingesetzt und in Werk III werden alle übrigen Teile verwendet.

Konzipieren Sie die horizontale Zerlegung der globalen Tabelle

```
Teile( TeileNr, Bez, LieferNr, Preis )
```

unter der Voraussetzung, dass in der Regel in der verteilten Datenbank im jeweiligen Werk auf die Teile zugegriffen wird, die dort verwendet werden.

Zeigen Sie außerdem, wie die globale Tabelle aus den Fragmenten entsteht.

### 33. Beispiel für Zerlegung

Die Relation

```
Angest( PersNr, Name, Gehalt, MgrNr, AbtNr, Anschrift )
```

soll so zerlegt werden, dass ein Fragment Namen und Anschrift, eines die Personalnummer des Vorgesetzten und die Abteilungsnummer enthält, schließlich soll das dritte Fragment Namen und Gehalt umfassen.

Wie kann man die globale Relation aus den Fragmenten definieren?

### 34. Schema einer verteilten Datenbank

Ein Softwarehaus möchte eine verteilte Datenbank für das Projektmanagement in ihren Geschäftsstellen Hamburg (HH), Darmstadt (DA) und München (M) einrichten. Die im Moment in der zentralisierten Datenbank verwendete Datenstruktur ist so aufgebaut:

```
Angest( PersNr, Name, Gehalt, AbtNr, Anschrift )  
Abt( AbtNr, Name, MgrNr, BereichNr, RegionNr )  
Projekt( ProjNr, Name, Kontraktvolumen, ProjMgrNr, AbtNr )  
Leistung( PersNr, ProjNr, Stunden )  
Bereich( BereichNr, Name )  
Region( RegionNr, Name )
```

dabei sind

Bereiche: Consulting (SC), Entwicklung (SE) und Rechenzentrumsbetrieb (RZ)

Regionen: Hamburg (HH) mit den Bereichen SC, SE; Darmstadt (DA) mit dem Bereich SE und München mit den Bereichen SE und RZ.

Die Zentrale der Firma ist in München, wo auch die Personalabteilung (HR) ist.

Üblicherweise werden folgende Informationen benötigt:

- Informationen über die Projekte in den Bereichen, wobei jedes Projekt mit Mitarbeiter aus der Region durchgeführt wird.
- Informationen der zentralen Personalabteilung über alle Mitarbeiter.

- Informationen der Abteilung über die Arbeit der Mitarbeiter an Projekten für die Rechnungsstellung.

Erarbeiten Sie einen Vorschlag für die Zerlegung und Replikation von Daten, die der geschilderten Situation gerecht wird. Begründen Sie Ihre Vorgehensweise:

- Erstellen Sie ein Entity-Relationship-Diagramm für die Datenstruktur.
- Erstellen Sie das Design für Verteilung und Replikation.
- Zeigen Sie wie die globalen Relationen aus den Fragmenten rekonstruiert werden können.
- Überlegen Sie Anfragestrategien für die oben skizzierten typischen Anfragen in der verteilten Datenbank

### 35. Semijoin

Betrachten Sie folgende Situation in einer verteilten Datenbank:

An Site 1 ist die Tabelle T gespeichert, die Zeilen mit den ids 1, 3, 5, 8, 14 enthält. Jede Zeile ist 1024 Bytes lang, 4 Bytes für die id und 1020 Bytes für die Nutzdaten.

An Site 2 ist die Tabelle S gespeichert, deren Zeilen analog aufgebaut sind, und die ids 1, 2, 4, 7, 14, 20 haben.

An Site 1 soll der Join durchgeführt werden:

```
select * from T, S where T.id = S.id.
```

Spielen Sie die Schritte durch, die erforderlich sind, wenn man den Join mittels des *Semijoins* ermittelt. Zeigen Sie, welche (Teile von) Tabellen von Site 1 an Site 2 und umgekehrt übertragen werden müssen.

Wieviele Bytes müssen bei diesem Vorgehen über das Netz übertragen werden? Vergleichen Sie den Wert mit der Vorgehensweise, dass die komplette Tabelle S an Site 1 übertragen wird und dort der Join durchgeführt wird.

### 36. Bloomjoin

Gehen Sie wieder von der Situation der vorherigen Aufgabe aus, spielen Sie nun den *Bloomjoin* durch, bei dem Sie die Hashfunktion  $h(x) = x \bmod 7$  verwenden.

Berechnen Sie wieder, wieviele Bytes über das Netz übertragen werden müssen.

### 37. Vorgehen in einer verteilten Datenbank I

Betrachten Sie folgende Datenbank:

```
Mitarbeiter( MId, MName, WId, Gehalt)  
Werk( WId, Sitz)
```

Setzen Sie voraus, dass die Tabelle *Mitarbeiter* horizontal fragmentiert ist nach der *WId* am jeweiligen Sitz des Werks.

Beschreiben Sie Strategien für folgende Statements:

- `update Mitarbeiter set Gehalt = Gehalt * 1.05`
- `update Mitarbeiter set WId = 12 where MId = 2314`
- `insert into Mitarbeiter values(4567, 'Schneider', 12, 5000)`

### 38. Vorgehen in einer verteilten Datenbank II

Verwenden Sie die Datenbank und ihre Verteilung aus der vorherigen Aufgabe. Setzen Sie zusätzlich voraus, dass folgende Replikationsstrategie eingeführt ist: Jedes Fragment hat 2 Replika: eines in der Zentrale in Frankfurt, das andere am Sitz des jeweiligen Werkes.

Beschreiben Sie gute Suchstrategien für folgende Abfragen, wenn Sie sich in Hamburg befinden:

- Alle Angestellten in Bremen
- Durchschnittsgehalt aller Beschäftigten

### 39. Vorgehen in einer verteilten Datenbank III

Wir wandeln unsere Datenbank etwas ab:

```
Mitarbeiter( MId, MName, AId, Gehalt, ...)  
Abteilung( AId, MgrId, Sitz, Status, ...)  
(MgrId ist die MId des Managers der Abteilung.)
```

Die Abfrage, die uns interessiert, lautet:

```
select * from Mitarbeiter M, Abteilung A  
where M.MId = A.MgrId and A.Status > 50
```

und wir wissen, dass etwa 1 Prozent der Mitarbeiter Abteilungsleiter sind und etwa die Hälfte der Abteilungen den gewünschten Status hat.

Setzen Sie voraus, dass die Tabelle der Mitarbeiter in Frankfurt, die der Abteilungen in München gespeichert ist. Sie selbst sind in Hamburg.

Die Tabelle *Mitarbeiter* hat 100.000 Zeilen à 4 KBytes, die Tabelle *Abteilung* 1.000 Zeilen à 4 KBytes. Die Schlüssel sind jeweils 4 Bytes lang. Für die Übertragungsrate im Netz nehmen wir 100 KBytes pro Sekunde an.

Diskutieren Sie folgende Strategien und machen Sie Aussagen über die Übertragungskosten:

- Transferieren Sie beide Tabellen nach Hamburg und berechnen den Join dort.
- Transferieren Sie *Abteilung* nach Frankfurt und berechnen den Join dort, übermitteln Sie das Ergebnis nach Hamburg.
- Berechnen Sie den Join mit der Strategie des *Bloomjoin* in Frankfurt und übermitteln Sie das Ergebnis nach Hamburg.
- Berechnen Sie den Join mit der Strategie des *Semijoin* in München und übermitteln Sie das Ergebnis nach Hamburg.

### 40. Arten der Replikation

Erläutern Sie den Unterschied zwischen *synchroner* und *asynchroner* Replikation in verteilten Datenbanken. Nennen Sie Gründe, weshalb bei den heute verfügbaren Datenbank-Systemen in der Regel die asynchrone Replikation verwendet wird?

### 41. 2PC Absturz des Teilnehmers

Im 2PC sieht der Ablauf (bei globalem Commit) für Teilnehmer und Koordinator (zusammengefasst) so aus:

Koordinator:

schreibe [T,prepare] in Log  
sende <prepare T> an alle Teilnehmer  
nach Erhalt der (positiven) Antwort: schreibe [T,commit] in Log  
sende <commit T> an alle Teilnehmer  
nach Bestätigung: schreibe [T,eot] in Log

Teilnehmer:

nach <prepare T>: schreibe [T,ready] in Log  
sende <ready T> an Koordinator  
nach <commit T>: schreibe [T,commit] in Log  
sende <ack T> an Koordinator

Erläutern Sie, wie das 2PC-Protokoll verläuft, wenn der Teilnehmer in folgender Situation abstürzt:

- vor Erhalt der Nachricht <prepare T>
- nach Erhalt von <prepare T>, aber vor der Antwort an den Koordinator
- nach Antwort <ready T> an Koordinator
- nach der Nachricht <ack T> an Koordinator

#### 42. 2PC Absturz des Koordinators

Erläutern Sie, wie das 2PC-Protokoll verläuft, wenn der Koordinator in folgender Situation abstürzt:

- vor dem Schreiben von [T,prepare]
- nach dem Senden von <prepare T> an alle Teilnehmer
- nach Erhalt der Antwort <ready T> von einem Teil der Teilnehmer
- nach dem Schreiben von [T,commit]
- nach dem Senden von <commit T> an einen Teil der Teilnehmer
- nach dem Senden von <commit T> an alle Teilnehmer

## D INFORMATION RETRIEVAL

[Einige der folgenden Aufgaben stammen von Norbert Fuhr, Uni Duisburg-Essen]

#### 43. Anwendungen des Information Retrievals

Beispiele für Systeme des Information Retrieval sind:

- Web-Suchmaschinen
- Suche in Online-Dokumentationen
- Digitale Bibliotheken
- Suche in Bildarchiven

Finden Sie jeweils ein Beispiel und untersuchen Sie

- Welche Dokumente oder Informationsobjekte können in dem jeweiligen System gefunden werden?
- Nach welchen Kriterien können die Objekte gesucht werden?

- Wie sieht das Ergebnis einer Anfrage aus?
- Wie beurteilen Sie die Antworten in Bezug auf Effizienz und Effektivität?

#### 44. Vektorraummodell

Gegeben seien folgende Dokumentrepräsentationen (entstanden durch Extraktion aus den drei Original-Dokumenten). Dabei gibt die Zahl beim Term an, wie oft er im Dokument vorkommt.

$D_1$  „retrieval, digital libraries, interface (3), evaluation“

$D_2$  „evaluation (2), retrieval, interface, user, service“

$D_3$  „digital libraries (3), agents, access, retrieval (2), distributed“

Die Terme des Vokabulars sind:

„access, agents, digital libraries, distributed, evaluation, interface, retrieval, service, user“

Aufgabe:

- Bestimmen Sie  $N_t$  (*document frequency*), die Zahl der Dokumente, die Term  $t$  enthalten.
- Ermitteln Sie die Vektoren für die Dokumente (mit Berücksichtigung der Termhäufigkeit)
- Betrachten Sie folgende Anfragen:
  - $Q_1$  „retrieval, evaluation“
  - $Q_2$  „digital libraries, interface, evaluation“Berechnen Sie die „Ähnlichkeit“ zwischen diesen Anfragen und den Dokumenten.
- Interpretieren Sie die Ergebnisse.

#### 45. Invertierter Index

Bilden Sie einen invertierten Index mit dem Aufbau

Term:  $\langle \text{Dok}_1: \text{Pos}_1, \text{Pos}_2, \dots; \text{Dok}_2: \text{Pos}_1, \dots \rangle$

aus folgenden Dokumenten:

- 1: Frankreich ist ein europäisches Land
- 2: Deutschland und Frankreich sind benachbart
- 3: Paris ist die Hauptstadt von Frankreich
- 4: Paris ist eine Weltstadt, Frankfurt auch

Als Liste der *stop words* sei „ist, ein, und, sind, die, von, eine, auch“ vorgegeben.

#### 46. Suchen im invertierten Index

Gegeben sei ein invertierter Index mit dem Aufbau

Term:  $\langle \text{Dok}_1: \text{Pos}_1, \text{Pos}_2, \dots; \text{Dok}_2: \text{Pos}_1, \dots \rangle$ ,

nämlich:

was:           〈 1: 1,14,102,302; 3: 11,53,233,401; 6: 26,43,82; 8: 23,49,401 〉  
du:            〈 2: 63,105,282; 3: 12,88,143; 6:27,128,169,482 〉  
heute:         〈 1: 211,234,311; 3: 13; 4: 100,122; 6: 28,234 〉  
kannst:        〈 2: 179,284; 3: 14,87,156; 6: 29,70 〉  
besorgen:     〈 3: 15; 6: 30,155; 7:67,166 〉  
verschiebe:   〈 3: 17,53; 5: 40,99,120; 8: 45,132 〉  
nicht:         〈 3: 18,44,217; 4: 34,97; 8: 1,46,156 〉  
auf:            〈 3: 19,61,101,189; 8: 47,386 〉  
morgen:        〈 3: 20,111,273; 4: 24,103; 8:48,430 〉

Ermitteln Sie die Fundstellen für folgende Anfragen:

- (a) besorgen
- (b) besorgen and was
- (c) „verschiebe nicht auf morgen“ (als Phrase)
- (d) „was du heute kannst“
- (e) „heute kannst besorgen“ and „verschiebe nicht auf morgen“