UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Prof. Dr.-Ing. G. Weikum
Dipl.-Inform. R. Schenkel

# Datenbanksysteme WS 1999/2000

## Praktische Übung 4

Physischer Datenbankentwurf

**Zu bearbeiten bis spätestens 11. Februar 2000**

**Anwendung: Videothek**

Herr Glotzkist, der Betreiber einer modernen Videothek, arbeitet mit dem Datenbankschema, das Sie aus der dritten praktischen Übung kennen. Er hat jedoch ein schwerwiegendes Performance-Problem mit seinen Datenbankanwendungen. Ein Freund von ihm, der leider nicht in Saarbrücken studiert hat, hatte Herrn Glotzkist für seine Anwendung das Schema erzeugt und die entsprechenden Anwendungsprogramme geschrieben. Der Normalbetrieb hat ergeben, dass die Anfragen, wie sie in den Anwendungsmodulen anfallen, nicht optimal vom physischen Datenbankentwurf unterstützt werden. Da der Freund von Herrn Glotzkist mittlerweile als Schafzüchter arbeitet, wendet sich Herr Glotzkist nun an *SIE*!

### Hinweise zu Schema und Datenbank

Das Schema der Videotheks-Datenbank is auf dem Beiblatt *Relationenschema der Videothek* gegeben. Die Primärschlüssel-Attribute sind unterstrichen. Eine Analyse der Semantik der Anwendung sowie des bisherigen Inhalts der Datenbank ergibt folgende Zusatzinformation:

(1) Relation KUNDEN

Die Relation KUNDEN enthält 40.000 Einträge, wobei nur die Hälfte der Kunden unterschiedliche Namen tragen und diese Namen unter den Kunden gleichmäßig verteilt sind. In der Relation kommen außerdem ca. 1.000 verschiedene Orte mit unterschiedlichen Postleitzahlen sowie ca. 1.300 verschiedene Straßen vor, die ebenfalls gleichmäßig über die Kunden verteilt sind.

(2) Relation FILME

In der Videothek sind aktuell 40.000 Filme gespeichert. Das Attribut *ProdJahr* enthält das Produktionsjahr des Filmes. In der Datenbank sind Filme mit Produktionsjahren von 1940 bis1995. 20% der Filme sind zwischen 1940 und 1970 produziert, 30% zwischen 1971 und 1985 und 50% zwischen 1986 und 1995. Alle Filme sind über 100 Genres und 10 verschiedene *Preiskategorien* gleichverteilt. Jeder Film besitzt eine *Plazierung* innerhalb der Hitliste. Man beachte aber, dass dieselben Hitparadenplätze mehrfach vorkommen können. So könnten z.B. zwei Filme dieselbe Plazierung besitzen, falls sich für sie dieselbe Anzahl von Ausleihen innerhalb eines Zeitraumes ergeben hat.

(3) Relation PERSONEN

Die Relation PERSONEN enthält eine Auswahl von *Personen* zu einem Film, die in diesem Film eine bestimmte *Funktion* bzw. Rolle haben. So hat z.B. die Person ‚Wolfgang Petersen' in dem Film ‚Das Boot' die Funktion ‚Regisseur'. Personen kommen in der Relation normalverteilt vor. Die 30 verschiedenen Funktionen und alle Filme kommen gleichmäßig verteilt vor.

(4) Relation BAENDER

Die Anzahl der Bänder pro Film ist normalverteilt. Im Mittel sind jedem Film 10 Bänder zugeteilt, mit einer Standardabweichung von 3 Bändern. Die Bänder sind gleichmäßig über 300 verschiedene Standorte verteilt. Das Attribut *Ausleihen* enthält die Information, wie oft ein Band bereits ausgeliehen wurde. Zu einem Film wird jeweils das Band vergeben, das bisher am wenigsten ausgegeben wurde.

(5) Relation AUSLEIHEN

Die Attribute *Von* und *Bis* enthalten konzeptionell nur Tagesnummern zwischen 0 und undendlich, da die Videothek hier an keiner feineren Auflösung interessiert ist. Die Werte von Von sind gleichmäßig verteilt. Die Differenz zwischen *Bis* und *Von* (Leihdauer) ist normalverteilt mit einem Mittelwert von einem Tag und einer Standardabweichung von ebenfalls einem Tag. Die Relation enthält 100.000 Ausleihen und 5.000 Reservierungen. Kunden kommen innerhalb von Ausleihen ungefähr normalverteilt vor. Alle Einträge mit einem *Bis*-Datum vor dem aktuellen Datum sind zurückgegeben (*Status*=‘z‘), während alle Einträge mit einem *Bis*-Datum nach dem aktuellen Datum als noch ausgeliehen gelten (*Status*=‘a‘). Im Mittel sind die Hälfte aller Einträge mit einem *Bis*-Datum gleich dem aktuellen Datum schon zurückgegeben, die andere Hälfte ist ausgeliehen. Reservierungen haben ein *Von*-Datum, das gleich dem aktuellen Datum ist oder danach liegt. Reservierungen haben einen *Status*-Wert ‘r‘.

**Hinweise zu Anwendungsmodulen**

Eine Analyse von Herrn Glotzkists Anwendung hat nun ergeben, daß sie aus zehn Modulen besteht. Jedes Modul besteht aus einer Menge von SQL-Queries, die innerhalb eines Moduls eine gewisse Wahrscheinlichkeit besitzen, dass sie bei Aufruf des Moduls zur Ausführung kommen. Das Beiblatt *Anwendungsmodule und Queries* listet alle Module zusammen mit der Query-Nummer sowie der genauen SQL-Query auf. Innerhalb der SQL-Query bezeichnen Namen, die mit einem Doppelpunkt beginnen, Variablen, die zur Laufzeit der Anwendung festgelegt werden.

Die dritte Spalte enthält jeweils die Wahrscheinlichkeit, dass die Query beim Betreten des Moduls ausgeführt wird. Beispielsweise hat im Anwendungsmodul *1:Kundenidentifikation* die Query 1.1 eine Wahrscheinlichkeit von 0.8, während die Query 1.3 bei jedem Aufruf von Modul 1 zur Ausführung kommt.

**Erwartete Last**

Um nun einen möglichst optimalen physischen Datenbankentwurf machen zu können, müssen Sie zunächst Informationen über die erwartete Last an die Datenbank haben. Durch Beobachtung des Nornalbetriebs über einen repräsentativen Zeitraum in der Videothek hat sich folgende Lastverteilung ergeben:

| Query-Typ | Aufgerufenes Modul | Anteil an Gesamttageslast |
|---|---|---|
| Online-Queries | 1 | 1 Prozent |
| | 2 | 11 Prozent |
| | 3 | 24 Prozent |
| | 4 | 4 Prozent |
| | 5 | 24 Prozent |
| | 6 | 5 Prozent |
| | 7 | 11 Prozent |
| | 8 | 20 Prozent |
| Batch-Queries | 9 | 1 Aufruf pro Tag |
| | 10 | 1 Aufruf pro Tag |

Der erste Teil der Tabelle enthält Online-Modulaufrufe, die zufällig über den Beobachtungszeitraum verteilt sind. Die dritte Spalte in diesem Teil der Tabelle bezeichnet den Anteil eines entsprechenden Modules an allen Modulaufrufen des Beobachtungszeitraumes. Die Module 9 und 10 werden nur einmal im Batch-Betrieb aufgerufen. Online-Queries und Batch-Queries wechseln sich im Normalbetrieb immer ab (Tag- vs. Nachtbetrieb).

**Aufgabenstellung**

Für die oben beschriebene Anwendung sollen Sie nun einen physischen Datenbankentwurf erstellen (Indizes, Index-Cluster, Hash-Cluster), der die folgenden Zielsetzungen hat:

**a) Die Anzahl der Blocktransfers für den spezifizierten Mix von Online-Queries soll minimiert werden, so daß sich minimale Antwortzeiten ergeben.**

**b) Die Anzahl der Blocktransfers für die Batch-Queries darf 50.000 Blocktransfers nicht überschreiten, damit die Tagesabrechnungen der Videothek in einem vorgegebenen Zeitraum abgewickelt werden können.**

c) **Die Datenbank darf durch Ihren Entwurf nicht größer als 200MB werden. Entwürfe, die dieses Limit überschreiten, werden nicht zu Ende ausgewertet und gelten als ungültig.**

Um den optimalen Entwurf zu finden, haben Sie **3 Versuche**, die **spätestens bis zum 11. Februar** abgeschlossen sein müssen. Aus Kapazitätsgründen können wir Ihnen leider nicht die Möglichkeit geben, Ihren Entwurf selbst zu evaluieren. Deshalb haben wir vorgesehen, daß Sie uns ein SQL-Skript per Email an **schenkel@cs.uni-sb.de** schicken **(Subject: Entwurf zu P4**), das aus den 'CREATE CLUSTER ...'- und 'CREATE INDEX ...'-Statements Ihres Entwurfs besteht.

**Wichtig:** Verteilen Sie die Abgabe Ihrer drei Versuche möglichst gleichmäßig über die Zeit, denn sonst kann es passieren, dass wir bei Häufungen mit der Auswertung nicht nachkommen. Geben Sie Ihren ersten Versuch also möglichst kurzfristig ab, um schnell eine Basis für Ihre weiteren Entwürfe zu haben.

Den Aufbau der Befehle sowie die Semantik der Parameter, die Sie bestimmen sollen, entnehmen Sie bitte dem Beiblatt *SQL-Befehle für Schemaentwurf*. Wir werden dann die Testdatenbank basierend auf Ihrem Entwurf aufbauen, die Query-Ausführungs-Pläne für jede einzelne Query ermitteln sowie die Anzahl der akkumuliert übertragenen Blöcke für jede Query innerhalb des Last-Query-Mixes messen. Das Ergebnis schicken wir Ihnen dann automatisch zurück. Sie können nach der Analyse dieses Ergebnisses Ihren Entwurf dann nochmals modifizieren und uns erneut zur Auswertung zuschicken, soweit Sie noch einen weiteren von den drei Versuchen übrig haben.

**Wichtig:** Das Skript mit Ihrem Entwurf wird automatisch ausgeführt und bewertet. Bitte stellen Sie unbedingt sicher, dass das Skript auch erfolgreich ausgeführt werden kann (keine Syntaxfehler, keine ungültigen Parameter-Werte, keine Fehler in der Erzeugungsreihenfolge usw.). Als Feedback bekommen Sie die Ausführungspläne jeder einzelnen Query in der Auswertung mitgeschickt.

**Belohnung!**

Der Entwurf mit dem besten Ergebnis wird am Semesterende prämiert. Ein Entwurf, der fehlerhaft ist oder das ‚Zeitlimit' für die Batch-Queries überschreitet, ist ungültig und gilt als verbraucht (tja, in der Realität haben Sie als Datenbankadministratorin bzw. -administrator auch gegen begrenzte Zeitressourcen und Deadlines zu kämpfen).

In jedem Falle bekommen Sie jedoch die Auswertung Ihres Entwurfs zugeschickt.

# SQL-Befehle für Schemaentwurf

(siehe dazu auch die entsprechenden Seiten im Skript)

Dieses Beiblatt beschreibt die Syntax der Oracle-SQL-Befehle zum Erzeugen von Tabellen, Indizes und Clustern. Es werden hier nur diejenigen Parameter erklärt, die Sie im Rahmen Ihres Entwurfes bestimmen sollten.

Weitere Informationen finden Sie im *Oracle7 Server SQL Reference Manual* unter der URL
*http://www-dbs.cs.uni-sb.de/~schenkel/oradoc/*

**Erzeugen von Clustern**

Oracle bietet die Möglichkeit, Cluster-Indizes und Hash-Clustern zu erzeugen. Tupel mit gleichen Werten im Clusterschlüssel bzw. mit gleichem Hash-Wert werden in Clustern möglichst in einem Block abgespeichert. Ein Cluster wird durch folgenden Befehl erzeugt:

**CREATE CLUSTER** *clustername* (attr1 datentyp1 [,attr2 datentyp2] ... )
[**PCTUSED** integer] [**PCTFREE** integer]
[**SIZE** integer [**K**|**M**] ]
[**INDEX** | [**HASH IS** attr] **HASHKEYS** integer]

Die Attribute attr1, attr2,... geben den Clusterschlüssel an. Die Parameter PCTUSED und PCTFREE steuern den Füllgrad von Datenblöcken. Der Parameter **PCTUSED** bezeichnet dabei den Anteil (Angabe in Prozent) eines Datenblocks, der mindestens belegt sein sollte. Wird dieser Anteil unterschritten, so dürfen wieder Tupel in den Clusterblock eingefügt werden. **PCTFREE** gibt an, wieviel Prozent des Datenblocks mindestens frei sein sollten. Wird dieser Wert unterschritten, so dürfen in den Datenblock keine weiteren Tupel eingefügt werden, bis der minimale Füllgrad (PCTUSED %) wieder erreicht ist. Die Summe aus PCTUSED und PCTFREE darf den Wert 100 nie überschreiten. Der Parameter **SIZE** bestimmt, wieviele Bytes innerhalb eines Datenblockes für denselben Clusterschlüssel-Wert bzw. Hash-Wert verwendet werden. Der Wert darf die Größe eines Datenblockes (in unserem Fall 2048) nicht überschreiten. Wird die Angabe von SIZE weggelassen, so reserviert Oracle einen Datenblock pro Schlüsselwert. Die Angabe von SIZE erfolgt in Bytes, wobei **K** und **M** für 'Kilobytes' bzw. 'Megabytes' stehen. Die Angabe von **INDEX** erzeugt einen Index-Cluster, während durch **HASHKEYS *n*** ein Hash-Cluster mit n verschiedenen Hash-Werten (Buckets) erzeugt wird. Der Wert n wird zuvor intern auf die nächst größere Primzahl aufgerundet. Der Default-Cluster ist INDEX. Durch **HASH IS *attr*** kann bei einem Hash-Cluster das Attribut *attr* direkt als Hash-Funktion verwendet werden. Es muß jedoch vom Typ NUMBER sein, muss äquivalent zum Cluster-Schlüssel sein und darf nur nicht-negative Werte enthalten. Wird diese Angabe weggelassen, so verwendet Oracle eine Standard Hash-Funktion. Der Erzeugung eines Index-Clusters muss der Aufbau des entsprechenden Index folgen, damit auf die Daten zugegriffen werden kann. Die Tabellen, die in einem Cluster abgelegt werden sollen, müssen ebenfalls nach dem Cluster erzeugt werden.

**Erzeugen von Indizes**

Indizes werden durch folgende Anweisung erzeugt:

**CREATE INDEX** *indexname* **ON** { tabellenname(attr1 [,attr2]...) | **CLUSTER** clustername }
[**PCTFREE** integer]

Durch die Angabe eines Tabellennamens und der zu indizierenden Attribute der Tabelle wird ein mittelbarer Index auf der entsprechenden Relation aufgebaut. Durch Angabe von **ASC** bzw. **DESC** wird das Attribut innerhalb des Index aufsteigend bzw. absteigend sortiert. Wird statt eines Tabellennamens **CLUSTER *clustername*** angegeben, so wird ein Cluster-Index über dem Cluster mit der Bezeichnung *clustername* aufgebaut. Der Cluster *clustername* muß

als Index-Cluster bereits erzeugt worden sein. Der Wert von **PCTFREE** bestimmt, wieviel Prozent der Index-Datenblöcke beim Aufbau des Index mindestens unbelegt bleiben.

### Erzeugen von Tabellen

Die Tabellen, die Sie in Ihrem Entwurf verwenden, sind bereits in ihrem Grundgerüst durch das konzeptionelle Schema gegeben. Die grundlegenden Tabellen-Definitionen finden Sie in der Datei

*http://www-dbs.cs.uni-sb.de/dbs99/uebungen/Grundschema.sql*

Sie dürfen keine der hier bereits spezifizierten Integritätsbedingungen entfernen, auch wenn dadurch die Performance verbessert werden würde (das ist eben der Preis, den man für Aufrechterhaltung der Integrität zahlen muß). Sie dürfen jedoch die im folgenden Befehlsaufbau beschriebenen Parameter zur Speicherung der Relationen ergänzen.

**CREATE TABLE** tabellenname( attr1 datentyp1 [constraints]
[, attr2 datentyp2 [column_constraint] ]...)
[ [**PCTFREE** integer] [**PCTUSED** integer] | [**CLUSTER** clustername (attr1 [, attr2]...)] ]

Als Constraints wollen wir nur diejenigen vom Typ **NOT NULL**, **PRIMARY KEY** und **REFERENCES Relation(attr)** betrachten, deren Semantik dem Skript entnommen werden können. Wichtig ist hierbei, daß durch Angabe von **PRIMARY KEY** intern ein Index über dem Primärschlüssel aufgebaut wird, der auch vom Optimierer verwendet wird.

Soll die Tabelle in einem Cluster abgelegt werden, so muß durch **CLUSTER clustername** der die Tabelle enthaltende Cluster angegeben werden, zusammen mit den Attributen (attr1, attr2...) der Tabelle, die dem Clusterschlüssel entsprechen. Es muß hier beachtet werden, daß das an einer bestimmten Position angegebene Tabellenattribut im Typ identisch ist zum Clusterschlüssel-Attribut an derselben Position. Wird die Tabelle nicht in einem Cluster abgelegt, so wird durch PCTUSED festgelegt, wieviel Prozent der Datenblöcke minimal belegt sein sollten. Fällt der Füllgrad unter diesen Wert, so können wieder Tupel in den Datenblock eingefügt werden. Durch PCTFREE wird vorgegeben, wieviel Prozent des Datenblockes mindestens freigehalten werden sollen. Wird der Wert unterschritten, so dürfen solange keine Tupel in den Block eingefügt werden, bis weniger als PCTUSED Prozent des Blockes belegt sind. Die Summe aus beiden Angaben darf 100 nicht überschreiten. Der Default-Wert von PCTFREE ist 10. von PCTUSED 40.

### Ermitteln des Query-Plans

Sie können den Query-Plan des Oracle-Optimierers, wie in der dritten praktischen Übung, durch Aufruf des Skriptes

*http://www-dbs.cs.uni-sb.de/dbs99/uebungen/sp.sql*

in SQLPlus ermitteln (sp für 'show plan'). Durch Aufruf von *@sp 'SELECT * FROM KUNDEN WHERE KundenNr=123'* wird beispielsweise der Query-Plan für Query 1.1 der Anwendungsmodule ermittelt.