Universität Heidelberg
Lehrstuhl für Datenbanksysteme
Prof. Dr. Michael Gertz
Sebastian Lackner

17. Juni 2019
Datenbanken
Sommersemester 2019

Übungsblatt 7: "SQL, Tupelkalkül und physische Datenorganisation"

Abgabe bis Montag, 1. Juli 2019, entweder zu **Beginn** der Vorlesung, oder bis 14:00 Uhr in den Briefkästen vor dem Sekretariat Informatik (Raum 1/308) im Mathematikon

---

Wichtige Hinweise zur Abgabe

- **Abgabe:** Zu jeder Anfrage ist der SQL-Ausdruck **und das Ergebnis der Anfrage** anzugeben. Hierbei gilt: Hat das Ergebnis mehr als 15 Tupel, ist nur die Anzahl der Tupel anzugeben. Hat das Ergebnis 15 oder weniger Tupel, so sind alle Ergebnistupel explizit anzugeben. Nicht ausführbare Anfragen (Syntaxfehler) werden mit 0 Punkten bewertet. Für semantische Fehler oder fehlende Ergebnisse gibt es Abzüge. Kopieren Sie die SQL-Ausdrücke und die Ergebnisse am besten direkt aus der Konsole bzw. aus pgAdmin in ihr Dokument, um Syntaxfehler zu vermeiden.
- **Online-Abgabe:** Bitte geben Sie zu allen Aufgaben, in denen Sie SQL-Anfragen formulieren müssen (mit ✉ markiert), immer auch eine digitale Version in Form einer **einfachen Textdatei** ab. Diese muss mit Ihrer schriftlichen Abgabe übereinstimmen und dient dazu, die Tutoren bei der Korrektur komplexer Anfragen zu unterstützen. Die digitale Abgabe erfolgt per **E-Mail** direkt an den jeweiligen Tutor - die E-Mail-Adressen hierzu finden Sie unten auf dem Übungsblatt und in Moodle. Es ist ausreichend wenn ein Gruppenmitglied die Anfragen digital abgibt.

Aufgabe 7-1 Weitere SQL-Anfragen — 3 + 3 + 6 + 6 = 18 Punkte ✉

Die folgende Anfragen sind in SQL an die MusicDB zu formulieren.

1. Die Namen der Personen, die mindestens 3 Bands gegründet haben (`founder = 'Y'`). Die Anzahl der Bands soll mit ausgegeben, und das Ergebnis absteigend nach der Anzahl der Bands sortiert werden.
2. Die Namen der Bands, die im Jahre 1990 gegründet wurden, und deren erstes Konzert in Deutschland stattfand.
3. Die Paare von Namen der Bands, die exakt die gleichen Bandmitglieder haben bzw. hatten. Die Zeiträume, in denen Bandmitglieder in der Gruppe gespielt haben, sollen zur Vereinfachung ignoriert werden.
   Hinweis: Zwei Mengen r_1 und r_2 stimmen genau dann überein, wenn $|r_1 \cap r_2| = |r_1| = |r_2|$ gilt.
4. Die Namen der Songs, die während eines einzelnen Konzertes zwei oder mehrfach unmittelbar in Folge gespielt wurden. Für jeden Song soll der Name des Konzertes, sowie die Anzahl, wie häufig der jeweilige Song mehrfach hintereinander gespielt wurde, mit ausgegeben werden.
   Hinweis: Zwei Songs wurden unmittelbar in Folge gespielt, wenn sich `SONGS.pos` um 1 unterscheidet.

Aufgabe 7-2 Tupelkalkül **2 + 3 = 5 Punkte**

Gegeben sind die folgenden Relationenschemata:

```
BESUCHER = {Gast, Restaurant}   ▷ Welcher Gast besucht welches Restaurant?
ANGEBOT  = {Restaurant, Wein}   ▷ Welches Restaurant bietet welchen Wein an?
MAG      = {Gast, Wein}         ▷ Welcher Gast mag welchen Wein?
```

Geben Sie die folgenden Anfragen jeweils im Tupelkalkül an:

1. Gesucht sind die Gäste, welche ein Restaurant besuchen, das auch einen Wein anbietet, den sie mögen.
2. Gesucht sind die Gäste, welche ausschließlich Restaurants besuchen, die auch einen Wein anbieten, den sie mögen.

Aufgabe 7-3 Seiten und Sätze **2 + 2 + 2 = 6 Punkte**

1. Berechnen Sie die durchschnittliche Satzlänge eines Tupels der Relation `ARTIST` auf Grundlage der Datentypen im `create table` Statement. Verwenden Sie als Datentyp-Größe (Bytes) die Werte aus der PostgreSQL Dokumentation:

   `https://www.postgresql.org/docs/current/datatype.html`

   Nehmen Sie zur Vereinfachung an, dass `character varying(n)` und `character(n)` jeweils genau `n` Bytes benötigen.

2. Bei PostgreSQL stehen pro Seite 8192 Bytes für die Speicherung der Tupel zur Verfügung. Vor jedem Tupel wird außerdem ein 23 Byte großer Header mit zusätzlichen Informationen abgespeichert. Angenommen es werden ausschließlich Nichtspannsätze als Blockungsstrategie verwendet, wie viele Seiten werden benötigt, um alle Tupel der Relation `ARTIST` abzuspeichern?

3. Vergleichen Sie diese Abschätzung mit dem tatsächlichen Speicherverbrauch der Relation `ARTIST`. In PostgreSQL lässt sich der Wert mit der folgenden SQL-Anfrage abrufen:

   ```
   select relname, relpages from pg_class
   ```

   Wie viele Seiten werden benötigt? Falls es eine Abweichung gibt, welche Gründe könnte dies haben? *Hinweis:* Weitere Informationen über die Seitenverwaltung von PostgreSQL finden Sie hier:

   `https://www.postgresql.org/docs/current/storage-page-layout.html`

Aufgabe 7-4 B-Bäume **5 + 2 = 7 Punkte**

Gegeben ist die folgende Sequenz von Zahlen:

39, 68, 35, 98, 27, 15, 5, 21, 77, 81, 17, 79

1. Konstruieren Sie einen B-Baum der Ordnung $m = 2$, indem Sie die obigen Werte in der angegebenen Reihenfolge in einen leeren Baum einfügen.
2. Löschen Sie aus dem so konstruierten Baum den Wert *21*.

Es genügt, wenn Sie jeweils den finalen Baum nach dem vollständigen Einfügen bzw. Löschen der Werte angeben, d.h. abzugeben sind zwei Bäume. Achten Sie auf eine nachvollziehbare Darstellung und geben Sie alle relevanten Features der B-Bäume an.

Aufgabe 7-5 B$^+$-Bäume **5 + 2 = 7 Punkte**

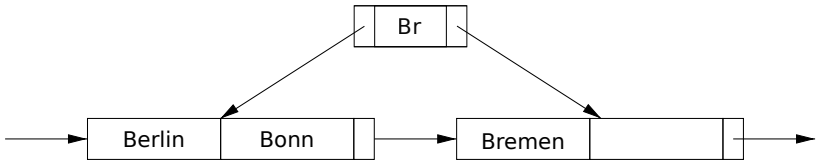
Gegeben ist die folgende Sequenz von Elementen:

Berlin, Bremen, Bonn, Hamburg, Essen, Dresden, Stuttgart,
Frankfurt, Duesseldorf, Leipzig, Duisburg, Dortmund

1. Konstruieren Sie einen B$^+$-Baum der Ordnung $(x = 1,\ y = 1)$, d.h. mit mindestens einem Eintrag pro Indexseite (x) und einem Eintrag pro Datensatz-Seite (y), indem Sie die obigen Werte in der angegebenen Reihenfolge in einen leeren Baum einfügen. Verwenden Sie als Zugriffsattribute möglichst kurze eindeutige String-Präfixe. Die Blätter seien durch eine einfach verkettete Liste miteinander verbunden.

   Die Präfixe sollen jeweils beim Aufteilen einer Datensatz-Seite generiert werden, ohne Berücksichtigung der danach noch folgenden Elemente. Alle Elemente des linken Knotens sollen stets kleiner als der Präfix sein, alle Elemente des rechten Knotens größer oder gleich. Falls sich der Inhalt einer Seite nicht gleichmäßig aufteilen lässt, so soll der linke Knoten einen Eintrag mehr bekommen als der rechte Knoten.

   Zum Vergleich, nach den ersten drei Einfügungen sollte der Baum wie folgt aussehen:



   Es genügt, wenn Sie den finalen Baum nach dem vollständigen Einfügen der Elemente angeben. Achten Sie auf eine nachvollziehbare Darstellung und geben Sie alle relevanten Features der B$^+$-Bäume an.

2. Nehmen Sie an, die Sequenz von Elementen sei vor der Eingabe bereits sortiert. Beschreiben Sie eine Strategie, mit der der Baum effizienter aufgebaut werden kann, als durch das einzelne Einfügen aller Elemente.

Informationen zur Abgabe. Die Aufgaben können in Gruppen bis zu **drei** Studierende bearbeitet und abgegeben werden. Bitte schreiben Sie die Namen aller Mitglieder ihrer Gruppe sowie die Nummer ihrer Übungsgruppe (1, 2 oder 3) auf das Frontblatt ihrer Abgabe! Zur Erinnerung, hier die Übungsgruppen:

Gruppe 1:	Donnerstag, 16:00 - 18:00 Uhr, Dominic Gargya	→	dominic@gargya.de
Gruppe 2:	Freitag, 11:00 - 13:00 Uhr, Fabian Kaiser	→	Fabian.Kaiser@stud.uni-heidelberg.de
Gruppe 3:	Freitag, 14:00 - 16:00 Uhr, Philip Hausner	→	Hausner@stud.uni-heidelberg.de

Schreiben Sie klar und deutlich und verwenden Sie keinen Bleistift. **Tackern** Sie Ihre Lösungsblätter zusammen (keine Büroklammern, keine Origami-Kunstwerke). Sie können die Lösungen in dem ihrer Übungsgruppe entsprechenden Briefkasten vor dem Sekretariat Informatik (Raum 1/308) im Gebäude INF 205 (Mathematikon) einwerfen oder zu Beginn der Vorlesung abgeben. Eine komplett elektronische Abgabe ist prinzipiell nicht möglich. **Vergessen Sie nicht die Online-Abgabe ihres SQL Codes!**