JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

Data Dictionary

INSY Übung
4AHITM 2015/16

Kanyildiz Muhammedhizir & Seyyid Tiryaki

Version 1.0

Note:

Betreuer: Prof. Borko

Begonnen am 4. Mai 2016

Beendet am 11. Mai 2016

Inhaltsverzeichnis

1 Einführung

1.1 Ziele

- Zitierrechtlinien beachten
- Qualitätsreiche Quellen finden:
  - Bücher und Zeitschriften
  - Unterrichtsmaterialien
  - elektronische Online-Bibliotheken
  - spezielle wissenschaftliche Suchmaschinen
- Besseres Verstehen was bei dem DBMS im Hintergrund passiert

1.2 Voraussetzungen

- Im Unterreicht zuhören
- Die Referenz-Manuels durchforsten

1.3 Aufgabenstellung

- Wo liegt der große Unterschied zwischen den Data Dictionaries der einzelnen DBMS und wie erfolgt der Zugriff (MySQL, PostgreSQL, Oracle)?
- Kann eine Performancesteigerung durch Manipulation am System Katalog erzielt werden?
- Wie könnte man über den System Katalog den Datentypen eines Attributes einer bestimmten Tabelle ändern? Tun Sie dies und erläutern Sie was dabei nach einem SELECT auf dieses Attribut passiert!
- Wo liegt der Unterschied der einzelnen Index-Typen von PostgreSQL? Listen Sie diese tabellarisch auf!
- Wie sind B-Bäume grundsätzlich aufgebaut?
- Wieso kann bei B-Bäumen stets die maximale Zugriffszeit berechnet werden und in welchen Zusammenhang steht die Ordnungszahl mit den Knoten?
- Wie verläuft die Suche bei B-Bäumen?
- Welche weiteren Bäume werden bei Datenbank-Managementsystemen verwendet? Erläutern Sie kurz die Unterschiede!

2 Data-Dictionary [1]

„Ein Data-Dictionary ist ein Katalog von Metadaten, der die Definitionen und Darstellungsregeln für alle Anwendungsdaten eines Unternehmens und die Beziehungen zwischen den verschiedenen Datenobjekten enthält, damit der Datenbestand redundanzfrei und einheitlich strukturiert wird. Es ist ein Anwendungsfall eines spezifischen Datenmodells.

Bei einer relationalen Datenbank ist ein Datenwörterbuch eine Menge von Tabellen und Ansichten, die bei Abfragen (etwa mit der Sprache SQL) nur gelesen werden (read-only). Das Data-Dictionary ist wie eine Datenbank aufgebaut, enthält aber nicht Anwendungsdaten, sondern Metadaten, das heißt Daten, welche die Struktur der Anwendungsdaten beschreiben (und nicht den Inhalt selbst). Aufbau und Pflege eines solchen Datenkatalogs erfolgen üblicherweise über einen interaktiven Dialog oder mit Hilfe einer Datendefinitionssprache (DDL).“

2.1 PostgreSQL [2]

Erklärung

Im Data Directory speichern relationale Datenbanksysteme ihre Schemadaten, wie Informationen über Tabellen und Spalten, sowie interne Verwaltungsinformationen. In PostgreSQL sind die Systemkataloge normale Tabellen. Sie können die Tabellen löschen und neu erzeugen, Werte einfügen und ändern und auf diese Weise ziemlich einfach das Datenbanksystem komplett durcheinanderbringen. Normalerweise sollte man die Data Directory nicht von Hand ändern, denn dafür gibt es immer SQL-Befehle. (Zum Beispiel fügt CREATE DATABASE eine Spalte in den Katalog pg_database ein – und erzeugt die Datenbank auch tatsächlich auf der Festplatte.) Es gibt einige Ausnahmen für besonders esoterische Operationen, wie das Erzeugen einer neuen Indexmethode.

Data Dictionary erscheinen für den Benutzer als normale Tabellen, aber das Datenbanksystem speichert darin seine internen Informationen. Ein wichtiger Unterschied zwischen PostgreSQL und normalen relationalen Datenbanksystemen ist, dass PostgreSQL viel mehr Informationen in seinen Katalogen speichert: nicht nur Informationen über Tabellen und Spalten, sondern auch Informationen über Datentypen, Funktionen, Zugriffsmethoden

und so weiter. Diese Tabellen können vom Benutzer verändert werden, und da die Operation von PostgreSQL von diesen Tabellen gesteuert wird, bedeutet das, dass PostgreSQL von Benutzern erweitert werden kann. Im Gegensatz dazu können herkömmliche Datenbanksysteme nur erweitert werden, indem der Quellcode selbst verändert wird oder indem spezielle Module vom DBMS-Hersteller geladen werden.

Zugriff

PostgreSQL bietet die Möglichkeit einer Veränderung - pg_*

Der Zugriff auf genaue Informationen über die jeweiligen Catalogs erfolgt durch Eingabe der folgenden Zeile im psql:

SELECT * FROM pg_CATALOGNAME;

Table 8-1. PostgreSQL System Catalogs

Catalog Name	Description
pg_database	databases
pg_class	tables
pg_attribute	table columns
pg_index	indexes
pg_proc	procedures/functions
pg_type	data types (both base and complex)
pg_operator	operators
pg_aggregate	aggregate functions
pg_am	access methods
pg_amop	access method operators
pg_amproc	access method support functions
pg_opclass	access method operator classes

2.2 MySQL [3]

Erklärung

INFORMATION_SCHEMA gibt Zugriff auf Datenbank-Metadaten. Metadaten sind Informationen über Daten, also beispielsweise der Name einer Datenbank oder Tabelle, der Datentyp einer Spalte, oder Zugriffsberechtigungen. Manchmal werden diese Informationen auch als Data Dictionary oder Systemkatalog bezeichnet.
INFORMATION_SCHEMA ist die Informationsdatenbank, also der Ort, an den Informationen über alle anderen auf dem betreffenden MySQL Server gepflegten Datenbanken gespeichert werden.
Im INFORMATION_SCHEMA gibt es eine Reihe von schreibgeschützten Tabellen. Da diese in Wirklichkeit keine Basistabellen, sondern Views sind, werden Sie keine Dateien sehen, die mit ihnen verbunden sind.

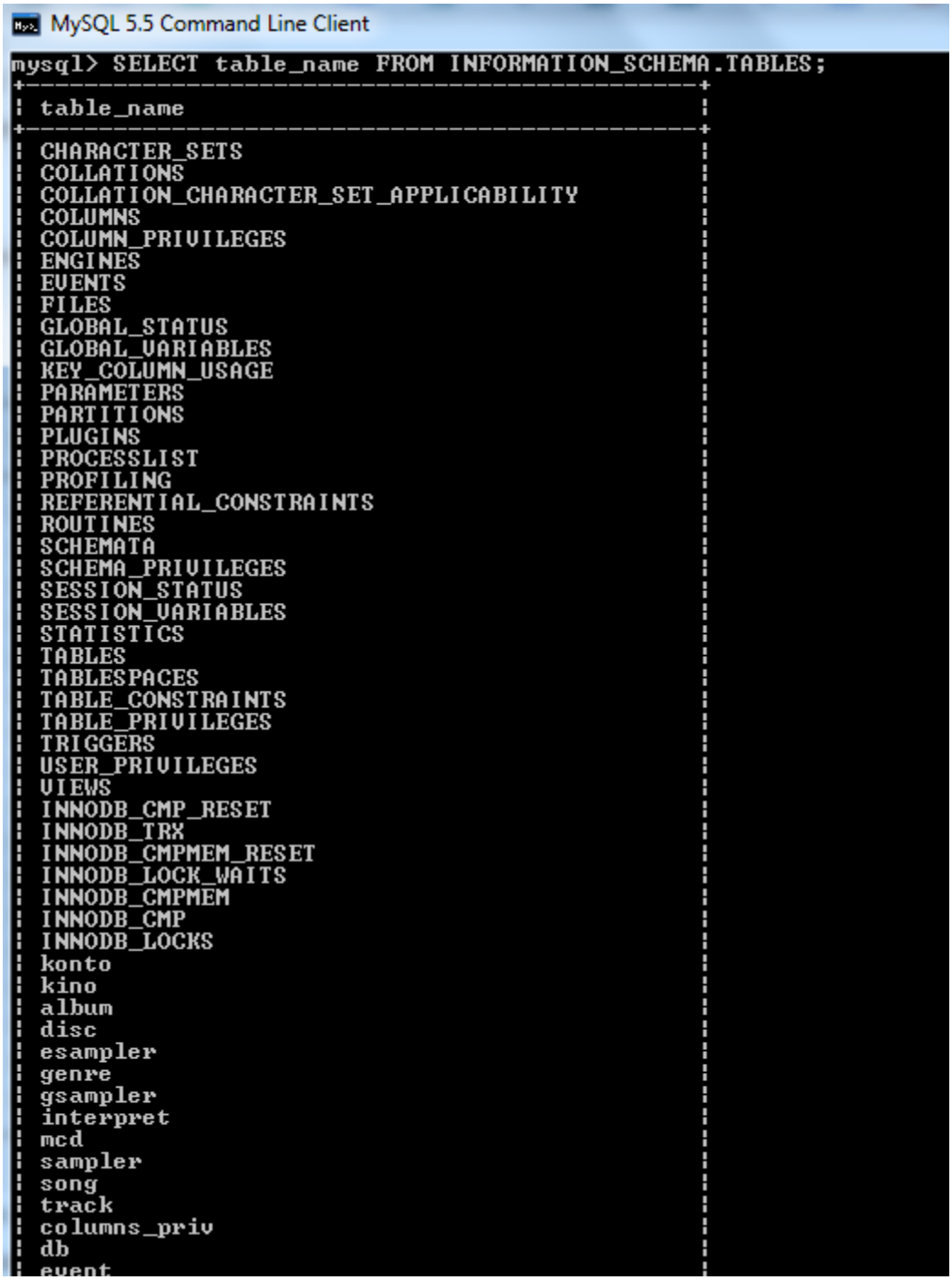
INFORMATION_SCHEMA gibt Zugriff auf Datenbank-Metadaten. Metadaten sind Informationen über Daten, also beispielsweise der Name einer Datenbank oder Tabelle, der Datentyp einer Spalte, oder Zugriffsberechtigungen. Manchmal werden diese Informationen auch als Data Dictionary oder Systemkatalog bezeichnet. INFORMATION_SCHEMA ist die Informationsdatenbank, also der Ort, an dem Informationen über alle anderen auf dem betreffenden MySQL Server gepflegten Datenbanken gespeichert werden.

Im INFORMATION_SCHEMA gibt es eine Reihe von schreibgeschützten Tabellen. Da diese in Wirklichkeit keine Basistabellen, sondern Views sind, werden Sie keine Dateien sehen, die mit ihnen verbunden sind.

Zugriff

```
USE INFORMATION_SCHEMA;

SHOW TABLES;
```


MySQL 5.5 Command Line Client
mysql> SELECT table_name FROM INFORMATION_SCHEMA.TABLES;
table_name
CHARACTER_SETS
COLLATIONS
COLLATION_CHARACTER_SET_APPLICABILITY
COLUMNS
COLUMN_PRIVILEGES
ENGINES
EVENTS
FILES
GLOBAL_STATUS
GLOBAL_VARIABLES
KEY_COLUMN_USAGE
PARAMETERS
PARTITIONS
PLUGINS
PROCESSLIST
PROFILING
REFERENTIAL_CONSTRAINTS
ROUTINES
SCHEMATA
SCHEMA_PRIVILEGES
SESSION_STATUS
SESSION_VARIABLES
STATISTICS
TABLES
TABLESPACES
TABLE_CONSTRAINTS
TABLE_PRIVILEGES
TRIGGERS
USER_PRIVILEGES
VIEWS
INNODB_CMP_RESET
INNODB_TRX
INNODB_CMPMEM_RESET
INNODB_LOCK_WAITS
INNODB_CMPMEM
INNODB_CMP
INNODB_LOCKS
konto
kino
album
disc
esampler
genre
gsampler
interpret
mcd
sampler
song
track
columns_priv
db
event

2.3 Oracle [4]

Erklärung

In Oracle gelten generell die gleichen Sachen wie in MySQL. Es gibt nur ganz wenige Unterschiede.
Data Dictionary: Enthält Metadaten über die Datenbank.
Data Dictionary Besteht aus Tabellen und Views, die Metadaten über die Datenbank enthalten. Mit SELECT * FROM DICTIONARY (kurz SELECT * FROM DICT) erklärt sich das Data Dictionary selber.

ALL_OBJECTS: Enthält alle Objekte, die einem Benutzer zugänglich sind. ALL_CATALOG: Enthält alle Tabellen, Views und Synonyme, die einem Benutzer zugänglich sind. ALL_TABLES: Enthält alle Tabellen, die einem Benutzer zugänglich sind. Analog für diverse andere Dinge (select * from ALL_CATALOG where TABLE_NAME LIKE 'ALL%';). USER_OBJECTS: Enthält alle Objekte, die einem Benutzer gehören. Analog für die anderen, meistens existieren für USER_... auch Abkürzungen, etwa OBJ für USER_OBJECTS. ALL_USERS: Enthält Informationen über alle Benutzer der Datenbank.

Informationen über Zugriffsrechte im Data Dictionary: SELECT * FROM SESSION_PRIVS;

Zugriff

SELECT_CATALOG_ROLE erlaubt den Zugriff auf Data Dictionary Views

DELETE_CATALOG_ROLE ermöglicht das Löschen in den Tabellen des Audit Trails und

EXECUTE_CATALOG_ROLE eröffnet den Zugriff auf Prozeduren und Packages des Data Dictionary.

3 Performancesteigerung durch Manipulation [5] [6]

3.1 Erklärung

Es kann eine Performancesteigerung durch Manipulation der Datenbanktabellen erzielt werden. Dies wird so gemacht, dass die Informationen des Systemkatalogs Zwischengespeichert werden, denn die Datenbank braucht immer den Zugriff zu diesen Informationen, um den Benutzern den Zugang zu gewähren. Zu diesen Tabellen kann man mittels dem Comment Befehl noch extra Informationen zu diesen Tabellen, Tabellen-Spalten, Views, Operatoren oder Indextypen.

- Um ein Comment zu einer Tabelle hinzuzufügen muss man das Comment System Privilege haben.
- Um ein Comment zu einem Indextype hinzuzufügen muss man das Create Any Indextype System Privilege haben.
- Um ein Comment zu einem Operator hinzuzufügen muss man das Create Any Operator System Privilege haben.

3.2 Beispiel

Create

```
COMMENT ON COLUMN employees.job_id
   IS 'abbreviated job title';
```

Drop

```
COMMENT ON COLUMN employees.job_id IS ' ';
```

4 Datentyp Veränderung [7]

CREATE TABLE konto(

kontostand integer,

saldo integer,

PRIMARY KEY (kontostand)

) ENGINE=INNODB;

Nach der Erstellung einer Tabelle wird diese befüllt.

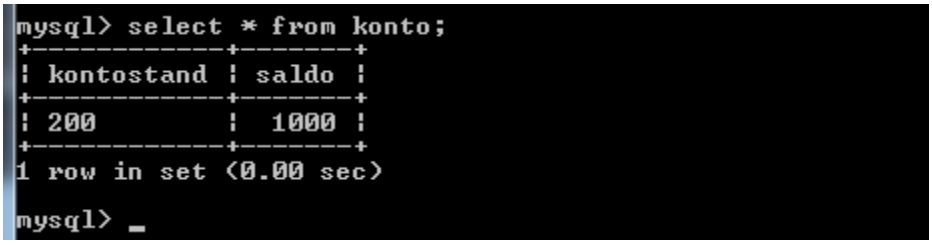
INSERT INTO konto VALUES (100,1000);

Nachdem die Tabelle erstellt und befüllt wurde, wird die Tabelle ausgegeben.

Um herauszufinden was passieren würde, wenn man den Datentyp eines Attributes ändert, verwende ich im Test ALTER TABLE.

Mit ALTER TABLE kann man nach der Erstellung der Tabelle den Datentyp, eines Attributes, gezielt ändern. Jedoch hat die Änderung des Datentyps bestimmte Kriterien. Wenn man z.B. einen INTEGER durch einen VARCHAR(20) ersetzt…

Jedoch hat die Änderung des Datentyps bestimmte Kriterien. Wenn man z.B. einen INTEGER durch einen VARCHAR(20) ersetzt…



…funktioniert glücklicherweise die Änderung. Doch nicht alle Änderungen funktionieren immer wie gewünscht.

5 PostgreSQL Indextypen [8]

5.1 Erklärung

PostgreSQL unterstützt derzeit vier Indextypen_ Jeder davon verwendet intern einen anderen Algorithmus, der ihn für verschiedene Arten von Anfragen geeignet macht. Man kann eine Spalte auch mehrfach mit verschiedenen Typen indizieren, wenn verschiedene Indextypen angebracht sind, In der Praxis kommt das allerdings selten vor.

Der Index-typ kann im Befehl CREATE INDEX angegeben werden:
CREATE INDEX indexname On tabelle indextyp (spalte);
Als Indextyp wird der im Folgenden aufgeführte interne Name verwendet.
Meist muss aber kein Typ angegeben werden, da der Standardtyp verwendet werden soll.

5.2 B-Tree

Erklärung

Dieser Indexryp ist eine Implementierung des in der Welt der Datenbanksysteme weit verbreitet ist. Er bedient die meisten Anwendungsfälle für Indexe in postgreSQL. Er unterstützt auch Gleichheitszeichen und Bereichssuchen für Datentypen mit natürlicher Sortierreihenfolge wie Zahlen und Zeichenketten, das heißt er kann alle Anfragen bearbeiten, die die Operatoren <, z, und > sowie darauf aufbauende Konstrukte wie BETREEN und IN verwenden. Außerdem können Anfragen mit IS NULL von B-tree-lndexen bearbeitet werden, B-tree ist der Standardindextyp und wird verwendet, wenn keine andere Angabe vorgenommen wurde. Aus Anwendersicht sollte — vereinfacht gesagt — immer dieser Indextyp verwendet werden, wenn es keinen expliziten Grund gibt, einen anderen Typ zu verwenden.

Nutzen

<,<=,=,>=,>,BETWEEN,IN,IS NULL,IS NOT NULL, LIKE

5.3 Hash

Erklärung

Dieser Indextyp verwendet eine Hash-Tabelle. Er unterstützt nur Gleichheitssuchen, also Anfragen mit dem Operator Er bedient somit nur eine echte Teilmenge der Anwendungsfälle des B-tree. Der Indextyp Hash ist eher als Überbleibsel aus vergangenen Zeiten zu sehen. Er bietet, soweit bekannt, keine bessere Geschwindigkeit als B-tree, hat aber diverse Probleme mit Sperren und bei der Recovery. Da er so gut wie nie verwendet Wird, ist die Implementierung wohl auch nicht so ausgereift wie die des B-tree. Außer bei Experimenten sollte er aktuell nicht verwendet werden.

Nutzen

= kann nur einen einfach Vergleich machen

5.4 GiST

Erklärung

GiST steht für Generalized Search Tree und ist eher als Infrastruktur für die Erstellung verschiedener anwendungsspezifischer Indextypen Zu verstehen. Das eigentliche Verhalten des Index wird bestimmt durch die Datentypen, die Operatoren und die sogenannten Operatorklassen, die sie zusammenfügen, In der Praxis werden CIST Indexe verwendet flir geometrische Datentypen (Bounding-Box-Suchen), PoscGlS, Volltextsuche sowie diverse andere Module, die von der Art her eher nichtskalare Datentypen oder mehr dimensionale Suchoperatlonen anbieten.

Die Dokumentation des entsprechenden Systems Oder des Moduls wird dann darauf hinweisen, dass ein GiST-Index verwendet werden sollte, (Es wird in diesen Fällen meist gar nicht möglich sein, andere Indextypen wie etwa B-tree zu verwenden. Verwechslungsgefahr besteht daher eher nicht.)

Nutzen

Für Zweidimensionale Datentypen

<<,&<,&>,>>,<<|,&<|,|&>,|>>,@>,<@,~=,&&

5.5 GIN

Erklärung

GIN steht für Generalized Inverted Index. Es handelt sich um einen invertierten Index, der Werte mit mehreren Schlüsseln, zum Beispiel Arrays, indizieren kann, Ähnlich wie GiST kann GIN für verschiedenste Zwecke eingesetzt werden, zum Beispiel für die Volltextsuche und verschiedene Module, die Arrays und Listen indizieren. Auch hier gilt, dass die Dokumentation des entsprechenden Systems auf die Möglichkeit einer Indizierung mit GIN hinweisen wird.

Nutzen

<@, @>, =, &&

5.6 R-Tree

Erklärung

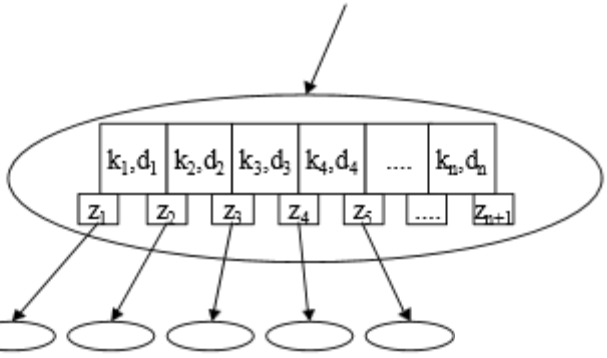
Frühere Versionen von PostgreSQL enthielten einen Indexryp namens R-Tree, der einen R-Baum implementierte, eine bekannte Indizierungsmethode für geometrische Datentypen. Diese Funktionalität wird in aktuellen Versionen von GiST übernommen.

6 Aufbau des B-Trees [9]

Alle Wege von der Wurzel zu den Blättern sind gleich lang.

- Jeder Knoten hat mindestens k und maximal 2k Einträge. Ausnahme: die Wurzel kann zwischen 1 und 2k Einträge haben.
- Wenn ein Knoten n Einträge hat, hat er n+1 Verweise auf seine Söhne. Ausnahme: Blätter haben keine bzw. undefinierte Verweise.
- Ein Blatt muss folgendermaßen organisiert sein:

- Wenn k1bis kn Schlüssel sind, sind z1 bis zn+1 Zeiger auf Söhne.
- z1 zeigt auf Teilbaum mit Schlüsseln kleiner k1
- zi (i=1...n) zeigt auf Teilbaum mit Schlüsseln zwischen ki und ki+1zi+1 zeigt auf Teilbaum mit Schlüsseln grösser



k_{i+1}

Ein B-Baum-Blatt.

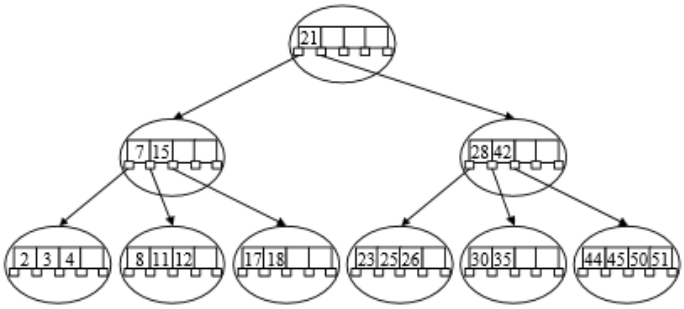
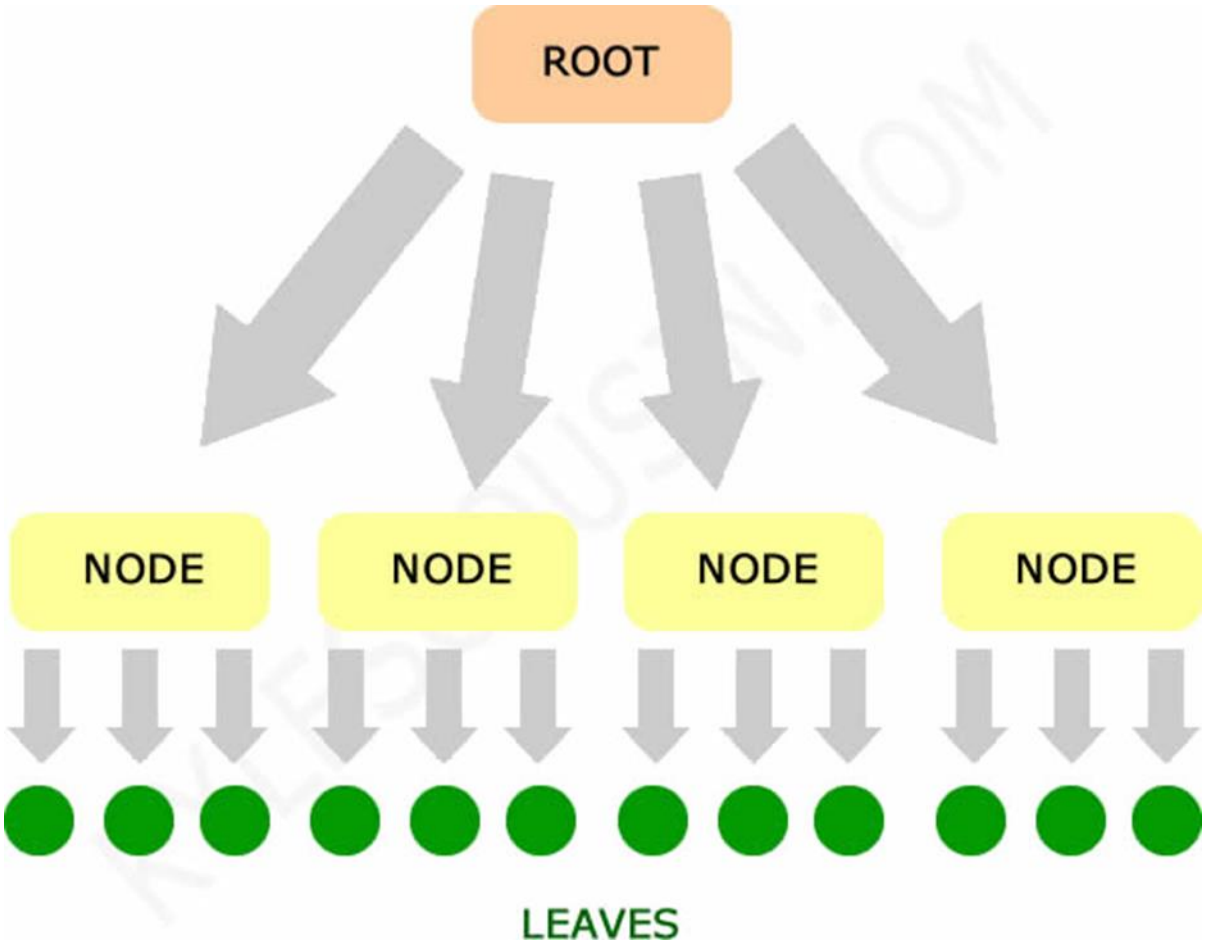


Abbildung 5 Ein Beispiel für einen B-Baum

Wie man an der Definition sieht, kann jedes Blatt eine unterschiedliche Anzahl von Einträgen haben. Dies führt dazu, dass meistens nur ein sehr lokaler Aufwand getrieben werden muss, um den Baum in Balance zu halten.

In einem B-Bum gibt es stets ein Wurzel-Element welches in >=1 Knoten unterteilt wird. Diese Knoten könne sich wiederum unter Knoten unterteilen oder als Blätter enden (Blätter sind die Endpunkte eines Baumes).



Jedes Blatt hat den gleichen Abstand zur Wurzel. Solche Art b-Bäume nennt man Balancierte B- Bäume da die Blätterknoten den gleichen Abstand zueinander haben und die Anzahl der Blätter berechnet sich aus der Tiefe des Baumes aus (3 Ebenen -> drei Blätter zu je Blätterknoten)

7 B-Tree Zugriffszeit [10]

Anhand der Anzahl der Ebenen weiß ich wie hoch die Zugriffzeit ist. Bei einem B-Baum mit 3 Ebenen ist die Zugriffszeit 3. Die Ordnungszahl ist die Tiefe des Baumes und wenn man einen Knoten hinzufügt so steigt die Ordnungszahl immer um 1.

Die Zugriffszeit in einem B-Baum ist umso größer, je mehr Ebenen ein Baum hat. Die gewisse Anzahl an Knoten in einer Ebene darf dabei nicht überschritten werden. Wenn diese Anzahl jedoch erreicht wird, müssen die restlichen Knoten in eine untere Ebene.

Die Zeitkomplexität lässt sich aus dem folgenden Formel berechnen.

H= ist die Höhe des Baumes

N = unterknoten im Index Knoten

H + N = 4 + (4000/20) = 204 I/Os

Daraus Lässt sich die benötigte CPU Leistung berechnen

CPU = 204*0.04 = 8.16 secs

Je mehr ebenen ein Baum hat desto größer ist die Zugriffszeit in einem B-Baum. Es darf dabei eine gewisse Anzahl an Knoten in einer Ebene nicht überschritten werden. Falls diese Anzahl erreicht wird müssen die restlichen Knoten in eine untere Ebene. Die Ordnungszahl ist sozusagen die Anzahl der Ebenen im Baum. [Nach Prof. Schabel]

8 B-Tree Suche [11]

8.1 Erklärung

Der Suchalgorithmus auf einem B-Baum ist nicht sehr kompliziert. Er fängt an der Wurzel an zu suchen. Wenn man das gewünschte Element nicht findet, durchsucht man den in Frage kommenden Kind-Knoten usw. Die Suche nach einem Schlüssel liefert denjenigen Knoten, der diesen Schlüssel speichert, und die Position innerhalb dieses Knotens, für die gilt, dass Enthält der Baum den Schlüssel nicht, liefert die Suche das Ergebnis nicht enthalten.

Die Suche läuft in folgenden Schritten ab:

1.)Die Suche beginnt mit dem Wurzelknoten als aktuellem Knoten.

2.)Ist ein innerer Knoten, wird die Position des kleinsten Schlüssels

bestimmt, der größer oder gleich ist. Existiert eine solche Position, aber ist, kann der gesuchte Schlüssel nur in dem
Unterbaum mit Wurzel enthalten sein. Die Suche wird daher mit Schritt 2 und dem Knoten als aktuellem Knoten fortgesetzt. ansonsten wurde der Schlüssel gefunden und wird als Ergebnis zurückgeliefert.

3.)Existiert keine solche Position, ist der Schlüssel größer als alle im

aktuellen Knoten gespeicherten Schlüssel. In diesem Fall kann der gesuchte Schlüssel nur noch in dem Unterbaum enthalten sein, auf den der letzte Kind verweis zeigt. In diesem Fall wird die Suche mit Schritt 2 und dem Knoten als aktuellem Knoten fortgesetzt.

4.)Ist ein Blattknoten,

Wird in den Schlüsseln von gesucht.
Wenn der Schlüssel an Position gefunden wird, ist das Ergebnis, ansonsten nicht enthalten.

8.2 Zusatzinfo

Das Suchen im b-Baum ähnelt der suche im binär Baum nur die Auswahl ob man rechts oder links gehen muss wurde einem weggenommen und durch die Auswahl child1, child2, child3,…. Ersetzt.

Die richtigen Kind Elemente werden durch eine lineare Suche ausgewählt. Dabei werden die Inhalte der Knoten anhand der Vergleiches Operatoren verglichen. Nachdem der richtige Inhalt (<,>,=) gefunden wurde wird der Pointer zum dazugehörigen Knoten verfolgt. Die suche wird abgebrochen sobald der gewünschte Inhalt gefunden wurde.

Die Wurzel der Bäume wird stets im Speicher gehalten.

Man folgt einem Pfad von der Wurzel hinunter bis zu einem Blatt. Dabei wird jedes Mal eine Disk gelesen. Die Formel für die Nummer der Disks die lesen lautet O(h)=O(log n) h→ höhe des B-Baumes und n ist die Nummer der Schlüssel.

8.3 Algorithmus:

```
i <- 1
while i <= n[x] and k > key_i[x]
     do i <- i + 1
if i <= n[x] and k = key_i[x]
     then return (x, i)
if leaf[x]
     then return NIL
     else Disk-Read(c_i[x])
          return B-Tree-Search(c_i[x], k)
```

9 Trees in Datenbank-Managementsystemen [12]

Storage Engine	Permitted Indexes
Aria	BTREE, RTREE
MyISAM	BTREE, RTREE
InnoDB	BTREE
MEMORY/HEAP	HASH, BTREE
NDB	BTREE, HASH

9.1 B-TREE

B-Tree ist in der Informatik eine Daten- oder Indexstruktur, die häufig in Datenbanken und Dateisystemen eingesetzt wird. Ein B-Baum ist ein immer vollständig balancierter Baum, der Daten nach Schlüsseln sortiert speichert. Er kann binär sein, ist aber im Allgemeinen kein Binärbaum. Das Einfügen, Suchen und Löschen von Daten in B-Bäumen ist in amortisiert logarithmischer Zeit möglich. B-Bäume wachsen – und schrumpfen – anders als viele Suchbäume von den Blättern hin zur Wurzel.

9.2 R-TREE

Ein R-Baum ist eine in Datenbanksystemen verwendete mehrdimensionale dynamische Indexstruktur. Ähnlich zu einem B-Baum handelt es sich hier um eine balancierte Indexstruktur.

9.3 HASH-TREE

In der Kryptographie und Informatik ist ein Hash-Baum eine Datenstruktur, die einen Baum aus Hashwerten von Datenblöcken bildet, beispielsweise von einer Datei. Hash-Bäume sind eine Erweiterung von Hash-Listen und dienen gleichermaßen dazu, die Integrität von Daten sicherzustellen. Wenn sie die Tiger-Hashfunktion als Grundlage verwenden, so werden sie als Tiger Trees oder Tiger Tree Hashes bezeichnet.

10 Aufwandsabschätzung

Arbeit	Stundenabschätzung
Aufgaben 1-4	4:00 – 5:00 h
Aufgaben 5-8	3:00 – 4:00 h

11 Problem

Im Großen und Ganzen gab es keine Probleme, es war nur etwas anstrengend die Reference Manuels zu Übersetzten und die Information heraus zu kriegen.

12 Quellen

INDEX	Quelltitel + Information
[1]	Wikipedia: https://de.wikipedia.org/wiki/Data-Dictionary Kapitel: 2 zuletzt abgerufen am [10.05.2016]
[2]	Manual: http://www.postgresql.org/docs/8.3/static/textsearch-dictionaries.html Kapitel: 2.1 zuletzt abgerufen am [10.05.2016]
[3]	Manual: https://dev.mysql.com/doc/refman/5.5/en/information-schema.html Kapitel: 2.2 zuletzt abgerufen am [10.05.2016]
[4]	Manual: https://docs.oracle.com/cd/E11882_01/server.112/e40540/datadict.htm#CNCPT002 Kapitel: 2.3 zuletzt abgerufen am [10.05.2016]
[5]	Online: https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_4009.htm Kapitel: 3 zuletzt abgerufen am [10.05.2016]
[6]	Online: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch8.htm Kapitel: 3 zuletzt abgerufen am [10.05.2016]

[7]	Online: http://dev.mysql.com/doc/refman/5.6/en/alter-table.html Kapitel: 4 zuletzt abgerufen am [10.05.2016]
[8]	Manual: http://www.postgresql.org/docs/9.2/static/indexes-types.html Kapitel: 5 zuletzt abgerufen am [10.05.2016]
[9]	Manual: https://dev.mysql.com/doc/refman/5.5/en/index-btree-hash.html Kapitel: 6 zuletzt abgerufen am [10.05.2016]
[10]	Manual: https://de.wikipedia.org/wiki/B%2B-Baum Kapitel: 7 zuletzt abgerufen am [10.05.2016]
[11]	Manual: http://use-the-index-luke.com/sql/anatomy/the-tree Kapitel: 8 zuletzt abgerufen am [10.05.2016]
[12]	Manual: https://www.tutorialcup.com/dbms/b-tree.htm Kapitel: 9 zuletzt abgerufen am [10.05.2016]