JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Schulübung

DATA DICTIONARY UND DATENORGANISATION

INSY/SEW
4AHITM 2015/16

Matthias Mischek
Benedikt Berger

Version 1.0

Note:

Betreuer:

Begonnen am 20. Mai 2016

Beendet am 27. Mai 2016

Inhaltsverzeichnis

1. Einführung

Erarbeiten Sie in 2er-Gruppen folgende Fragestellungen in einem Dokument und geben Sie dieses als PDF ab. Beachten Sie dabei die Zitierregeln!

[BOR16]

1.1. Aufgabenstellung

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

[ELE01]

2. Ergebnisse

2.1. Aufgabe 1 - Unterschiede

Wo liegt der große Unterschied zwischen den Data Dictionaries der einzelnen DBMS und wie erfolgt der Zugriff (MySQL, PostgreSQL, Oracle)?

MySQL

Mithilfe von `SELECT` kann der Inhalt der Datenbank ausgelesen werden, `INSERT`, `DELETE` und `UPDATE` Befehle werden allerdings nicht unterstützt.

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES;
```

[MSQ01]

PostgreSQL

In PostgreSQL gibt es eigene Tabellen (pg_*) welche bearbeitet und gelöscht werden können. Somit kann `INSERT` und `UPDATE` ausgeführt werden.

```
SELECT * FROM pg_catalog.pg_tables;
```

[PSQ02]

Oracle

Das Data Directory kann in Oracle weder ausgelesen noch beschrieben werden. Alle relationalen Tabellen in der Datenbank werden in der Tabelle `DBA_TABLES` beschrieben.

```
SELECT TABLE_NAME FROM DBA_TABLES;
```

[ORA01]

2.2. Aufgabe 2 - Performancesteigerung

Kann eine Performancesteigerung durch Manipulation am System Catalog erzielt werden?

Mithilfe von bestimmten Einstellungen welche angepasst werden können, kann eine Performancesteigerung erzielt werden. Hier ist eine Liste der geläufigsten Einstellungsmöglichkeiten, welche die Performance steigern können:

- **max_connections = <num>**
  Hier wird die maximale Anzahl der gleichzeitigen Verbindungen zu einer Datenbank eingestellt
- **shared_buffers = <num>**
  Mit dieser Einstellung kann eine hohe Performancesteigerung erzieht werden. In der Regel wird hier 25% des verfügbaren Arbeitsspeichers angegeben.
- **effective_cache_size = <num>**
  Arbeitsspeicher, welcher PostgreSQL für „data caching" benutzen darf. In der Regel 50% des Systemspeichers
- **work_mem = <num>**
  Hier kann der Arbeitsspeicher eingestellt werden, welche für Sortieroperationen und Hash-Tables benutzt werden darf
- **max_fsm_pages = <num>**
  Bei Datenlöschung, werden diese nicht gleich komplett gelöscht, sondern in eine sogenannte „free map" gespeichert. Dabei definiert diese Einstellung, wie viel Speicher dieser Funktion zur verfügung steht (bei einer großen Anzahl an Lösch-Operationen, sollte dieser Wert erhöht werden)
- **fsync = <boolean>**
  Diese Option entscheidet, ob alle sogenannten „WAL pages" synchronisiert werden bevor die committed werden. Es ist sicherer, diese Einstellung einzuschalten, allerdings wird dadurch die Schreibgeschwindigkeit verringert
- **commit_delay = <num>**
  Mit commit_delay kann die Zeit in Millisekunden angegeben werden, nach welcher der Server ein erneutes committee bei einem Fehler versucht

[RES01]

2.3. Aufgabe 3 - Attribute ändern

Wie könnte man über den System Catalog den Datentypen eines Attributes einer bestimmten Tabelle ändern? Tun Sie dies und erläutern Sie was dabei nach einem SELECT auf dieses Attribut passiert!

Zuerst wurde eine Datenbank erstellt, inklusive Table und Attributen (Datentyp: VARCHAR). Zusätzlich wird ein Datensatz eingefügt.

```
postgres=# CREATE DATABASE testdb;
CREATE DATABASE
postgres=# CREATE TABLE testdb (
postgres(# attr VARCHAR PRIMARY KEY
postgres(# );
CREATE TABLE
postgres=# INSERT INTO testdb VALUES ('wert');
INSERT 0 1
```

Jetzt wird der Datentyp des Attributes von VARCHAR auf NUMERIC geändert:

```
postgres=# UPDATE pg_attribute SET atttypid=(SELECT oid FROM pg_type WHERE
typname='numeric') WHERE attname='attr' AND attrelid='testdb'::regclass;
UPDATE 1
```

Wie man jetzt sehen kann, hat sich der Datentyp geändert:

```
postgres=# SELECT * FROM testdb WHERE attr='wert';
ERROR:  invalid input syntax for type numeric: "wert"
```

2.4. Aufgabe 4 - Index-Typen

Wo liegt der Unterschied der einzelnen Index-Typen von PostgreSQL?
Listen Sie diese tabellarisch auf!

Unter PostgreSQL gibt es zahlreiche Index-Typen wie beispielsweise B-tree, Hash, GiST and GIN. Jeder einzelne benutzt einen eigenen Algorithmus, welche für die einzelne Abfragen ausgelegt sind. Standardmäßig erstellt der Befehl `CREATE INDEX` einen B-tree Index.

Operation	B-Baum	Hash	GiST	SP-GiST	GIN
<	x				
<=	x				
=	x	x			x
>=	x				
>	x				
BETWEEN	x				
IN	x				
IS (NOT) NULL	x				
<<			x	x	
&<			x		
&>			x		
>>			x	x	
<<|			x		
&<|			x		
|&>			x		
|>>			x		
@>			x		x
<@			x	x	x
~=			x	x	
&&			x		x
<^				x	
>^				x	

[PSQ01]

2.5. Aufgabe 5 - B-Baum

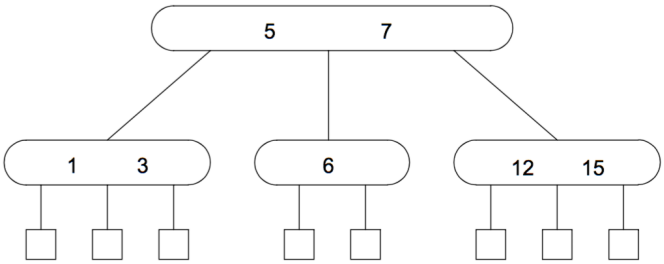
Wie sind B-Bäume grundsätzlich aufgebaut?

1. Jeder Knoten mit Ausnahme der Wurzel enthält mindestens [m/2] -1 Daten. Jeder Knoten enthält höchstens m - 1 Daten. Die Daten sind sortiert.

2. Knoten mit k Daten x1,...,xk haben k+1Referenzen auf Teilbäume mit Schlüsseln aus den Mengen:
   `{ ...,x1 1},{x1 1,...,x2 1},...,{xk 1 1,...xk 1},{xk 1,... }.`

3. Die Referenzen, die einen Knoten verlassen, sind entweder alle null-Referenzen oder alle Referenzen auf Knoten.

4. Alle Blätter haben gleiche Tiefe.



Im Kontext von B-Bäumen sind null-Referenzen durch ein Viereck dargestellt.

Dieser Baum hat 7 Schlüssel und 8 null-Referenzen.

[GOO01]

2.6. Aufgabe 6 - B-Baum Zugriff

Wieso kann bei B-Bäumen stets die maximale Zugriffszeit berechnet werden und in welchen Zusammenhang steht die Ordnungszahl mit den Knoten?

Die maximale Zugriffszeit kann man durch die Anzahl der Ebenen bzw. die Höhe des Grades (Ordnung) erkennen. Der Weg von der Wurzel zu einem Blatt ist stets gleich, wobei der weg der Ordnungszahl entspricht.
Wenn man einen Konten somit hinzufügt, steigt die Ordnungszahl um 1.

[INF01,BAY01]

2.7. Aufgabe 7 - B-Baum Suche

Wie verläuft die Suche bei B-Bäumen?

„Um einen Schlüssel in einem B-Baum zu suchen, verfahren wir wie folgt:

1. Ausgehend von der Wurzel prüfen wir, ob der gesuchte Schlüssel sich in dem gerade betrachteten Knoten befindet.

2. Ist das nicht der Fall, bestimmen wir den kleinsten Schlüssel der größer als der gesuchte ist.
    1. Existiert dieser Schlüssel, gehen wir zum Nachfolger-Knoten links von diesem Schlüssel über.
    2. Existiert der Schlüssel nicht, gehen wir zum letzten Nachfolger-Knoten über.

3. Wenn wir bei einer *null*-Referenz landen, ist die Suche erfolglos. „

[GOO01]

2.8. Aufgabe 8 - weitere Bäume

Welche weitere Bäume werden bei Datenbank-Managementsystemen verwendet? Erläutern Sie kurz die Unterschiede!

B*-Baum

Die wesentliche Veränderung gegenüber dem normalen B-Baum ist, dass die Knoten nunmehr zu 2/3 gefüllt sein müssen; beim B-Baum ist bereits eine Belegung von 50% ausreichend.
Um zu einer 2/3 Belegung der Knoten zu gelangen, bedarf es einiger Änderungen an der Splitting-Regelung beim Einfügen; es soll vermieden werden, dass Knoten allzu oft aufgespaltet werden. Der Grundgedanke ist, dass man ein lokales Umverteilungsschema verwendet, um somit die Spaltung so lange zu vermeiden, bis zwei Nachfolgerknoten voll sind. Man teilt dann diese zwei Knoten in drei auf, wobei dann jeder von ihnen zu 66% voll ist. Die Veränderungen am Algorithmus zur Einhaltung der B-Baum Kriterien halten sich in Grenzen; der Vorteil ist aber, dass die Höhe des Baums geringer wird (häufiges Spalten der Knoten das bis zur Wurzel vordringt vergrößert ja die Höhe des Baums) und somit eine schnellere Suche möglich ist.

B+-Baum

Das entscheidende Merkmal eines B+-Baums (den man auch „hohlen Baum“ nennt) ist, dass alle Datensätze in den Blättern liegen, die ursprünglichen Charakteristika des normalen B-Baums bleiben jedoch erhalten; die Wege von der Wurzel zu den Blättern stets gleich lang. Da die Blattknoten miteinander verknüpft sind und einer verketteten Liste ähneln, eignen sich B+-Bäume gut zur sequentiellen Abarbeitung.
In den inneren Knoten eines B+-Baums liegen nur noch Referenzschlüssel. Um einen gesuchten Wert zu finden, muss man nun durch die Indizes bis zur Blattebene vordringen, wobei die Werte im Indexbereich keine Rolle spielen, da man den vollen

Präfix B+-Baum

Um Speicherplatz zu sparen, führte man Präfix B+-Bäume ein. Die Besonderheit hierbei ist, dass man anstelle von vollständigen (String-)Schlüsseln einfach kürzere Zeichenfolgen oder gar nur ein Zeichen zur Unterscheidung verwendet. Man verwendet hier den kürzesten Wert um Platz zu sparen, also einen Buchstaben.
Somit kann man mehr Schlüssel in einen Knoten unterbringen wodurch die Zahl der Verzweigungen ansteigt und die Höhe des Baums abnimmt. Wie schon erwähnt, spart man dadurch nicht nur Speicherplatz, sondern man erhält auch niedrigere

Zugriffszeiten.

[INF01]

R-Baum

Der R-Baum ist für mehr- dimensionale Räume und Phänomene räumlicher Ausdehnung geeignet. Es ist ein höhenbalanzierter Baum ähnlich einem Binärbaum, dessen Indexeinträge in den Blättern Zeiger zu den realen Objekten beinhalten. Die Knoten korrespondieren zu den physikalischen Plattenpages. Er erfordert keine periodische Reorganisation. Zur Suche nach räumlichen Objekten ist nur eine kleine Anzahl von Knoten zu analysieren - im Gegensatz zu hierarchischen Baumstrukturen kann allerdings die parallele Suche in verschiedenen Ästen notwendig sein.

R*-Baum

R*-Baum ist eine kombinierte Optimierungsmethode von Bereich, Breite und Überlappung von jedem MBB in Knoten. Er ist eine Spezialisierung vom R- Baum bzw. ein erweiterter R-Baum.
Die einzigen Unterschiede zwischen dem R-Baum und dem R*-Baum sind nur bei Einfügen- und Splittenalgorithmen. Der Struktur von den beiden ist nicht zu unterscheiden.

R+-Baum

R+-Baum vermeidet überlappende Suchraum-Regionen in den Nicht-Blatt- knoten, wodurch der Suchraum disjunkt aufgeteilt wird. R+-Baum ist ein Vertreter des Clipping-Ansatzes (allerdings kein reines Clipping).

[DBS01]

3. Quellen

[GOO01] VORLESUNG INFORMATIK 2 ALGORITHMEN UND DATENSTRUKTUREN [ONLINE] AVAILABLE AT:
HTTPS://ELECTURES.INFORMATIK.UNI-FREIBURG.DE/PORTAL/DOWNLOAD/59/6149/INFO2-19-B_BAEUME_4UP.PDF
[ABGERUFEN AM 24. MAI 2016]

[PSQ01] DOCUMENTATION: 9.0: INDEX TYPES [ONLINE] AVAILABLE AT:
HTTP://WWW.POSTGRESQL.ORG/DOCS/9.0/STATIC/INDEXES-TYPES.HTML
[ABGERUFEN AM 24. MAI 2016]

[RES01] PERFORMANCE TUNING POSTGRESQL [ONLINE] AVAILABLE AT:
HTTP://WWW.REVSYS.COM/WRITINGS/POSTGRESQL-PERFORMANCE.HTML
[ABGERUFEN AM 24. MAI 2016]

[INF01] LUDWIG BACHMAIER, B-BÄUME [ONLINE] AVAILABLE AT:
HTTP://WWW.INFORMATIK-FORUM.AT/ATTACHMENT.PHP?ATTACHMENTID=948
[ABGERUFEN AM 24. MAI 2016]

[BAY01] MORITZ THEILE, TU-MÜNCHEN, B-BÄUME [ONLINE] AVAILABLE AT:
HTTP://WWWBAYER.IN.TUM.DE/LEHRE/WS2001/HSEM-BAYER/BTREESAUSARBEITUNG.PDF
[ABGERUFEN AM 24. MAI 2016]

[DBS01] JOANA BENDORAITYTE, R-BAUM UND SEINE SPEZIALISIERUNGEN [ONLINE] AVAILABLE AT:
HTTP://DBS.UNI-LEIPZIG.DE/FILE/RBAUM_0.PDF
[ABGERUFEN AM 24. MAI 2016]

[MSQ01] INFORMATION_SCHEMA TABLES ORACLE CORPORATION [ONLINE] AVAILABLE AT:
HTTP://DEV.MYSQL.COM/DOC/REFMAN/5.6/EN/INFORMATION-SCHEMA.HTML
[ABGERUFEN AM 24. MAI 2016]

[PSQ02] SYSTEM CATALOGS - POSTGRESQL GLOBAL DEVELOPMENT GROUP [ONLINE] AVAILABLE AT:
HTTP://WWW.POSTGRESQL.ORG/DOCS/9.4/STATIC/INDEXES-TYPES.HTML
[ABGERUFEN AM 24. MAI 2016]

[ORA01] THE DATA DICTIONARY - ORACLE CORPORATION [ONLINE] AVAILABLE AT:
HTTPS://DOCS.ORACLE.COM/HTML/A96524_01/C05DICTI.HTM
[ABGERUFEN AM 24. MAI 2016]