DATA DICTIONARY UND DATENORGANISATION

INSY S01

27. FEBRUAR 2015

STEFAN ERCEG
4AHITT

Inhalt

1. Aufgabenstellung

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

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!

2. Frage 1

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

In **MySQL** existiert die Datenbank `INFORMATION_SCHEMA`, welche die Metadaten des DBMS enthält. Somit werden dort z.B. die Namen von Datenbanken bzw. Tabellen, der Datentyp von einer bestimmten Spalte und die Zugriffsrechte für die existierenden User gespeichert.

Der Inhalt dieser Datenbank kann nur mittels `SELECT`-Befehlen gelesen werden, jedoch werden die `INSERT`-, `UPDATE`- und `DELETE`-Befehle nicht unterstützt, somit besteht keine Möglichkeit einer Veränderung. Die Einsicht der Datenbank wird für alle User gewährt. [1]

Ein `SELECT`-Befehl, bei der alle verfügbaren Tabellen der Datenbank gelesen werden, sieht folgendermaßen aus: `SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES;`

In **PostgreSQL** hingegen existieren Tabellen, die `pg_*` genannt werden, in der Datenbank `pg_catalog`, welche von den Usern jederzeit geändert bzw. gelöscht werden können. Es können daher Spalten bzw. Werte mittels `INSERT`-Befehlen hinzugefügt bzw. Werte mittels `UPDATE`-Befehlen geändert werden. [2]

Um eine Liste der Tabellen des Schemas zu erhalten, müsste der `SELECT`-Befehl folgendermaßen aussehen: `SELECT * FROM pg_catalog.pg_tables;`

Die Data Dictionary in **Oracle** kann nicht geändert werden, die Tabellen können daher nur gelesen werden. [3] Die Tabelle `DBA_TABLES` beschreibt alle relationalen Tabellen in der Datenbank. [4]

Um hier eine Liste der Tabellen des Schemas, müsste der `SELECT`-Befehl folgendermaßen aussehen: `SELECT TABLE_NAME FROM DBA_TABLES;`

3. Frage 2

Kann eine Performancesteigerung durch Manipulation am System Catalog erzielt werden?

Es können verschiedene Einstellungen durchgeführt werden, um die Performance drastisch zu erhöhen. Folgende Möglichkeiten werden angeboten:

- `max_connections = <num>`
  Bei dieser Option wird die Anzahl der möglichen gleichzeitigen Verbindungen zu der Datenbank angegeben.

- `shared_buffers = <num>`
  Dies ist die einfachste Methode, um die Performance zu verbessern. Der Shared Buffer sollte auf ca. 25 % von dem verfügbaren RAM gesetzt werden.

- `effective_cache_size = <num>`
  Hier kann angegeben werden, wie viel Speicher für die Cache-Daten verwendet werden soll. Dieser sollte auf ca. 50 % vom Systemspeicher gesetzt werden.

- `work_mem = <num>`
  Diese Option wird verwendet, um die Menge, die für die Sortieroperationen und die Hash-Tabellen genutzt wird, anzugeben. Ursprünglich hieß die Operation `sort_mem` in älteren Versionen von PostgreSQL.

- `max_fsm_pages = <num>`
  Wenn ein bestimmter Inhalt von einer Tabelle gelöscht wird, wird dieser nicht sofort komplett entfernt, sondern als „free“ in der Free Space Map gekennzeichnet. Bei einer hohen Anzahl von `DELETE`- und `INSERT`-Befehlen sollte dieser Wert erhöht werden.

- `commit_delay = <num>` und `commit_siblings = <num>`
  Bei diesen Optionen versucht der Server mehrere Transaktionen gleichzeitig zu committen. Mit `commit_delay` wird die Wartezeit des Servers bis zum nächsten Commit-Versuch in Mikrosekunden angegeben.

Um diese Möglichkeiten effektiv zu nützen, wird es notwendig sein, den gemeinsam genutzten Speicher auf dem System zu erhöhen.

[5], [6]

4. Frage 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!

Zu Beginn wurde eine Datenbank `testdb` erstellt:

```
postgres=# CREATE DATABASE testdb;
CREATE DATABASE
postgres=# \c testdb;
```

In dieser Datenbank wurde die Tabelle `test` mit 2 Attributen, das eine hat als Datentyp `VARCHAR`, das andere `INTEGER`, erstellt und ein Datensatz hinzugefügt:

```
testdb=# CREATE TABLE test (
testdb(#        attr1 VARCHAR PRIMARY KEY,
testdb(#        attr2 INTEGER
testdb(# );
CREATE TABLE
testdb=# INSERT INTO test VALUES('wert1',1);
INSERT 0 1
```

Nun wird der Datentyp vom Attribut `attr1` von `VARCHAR` auf `NUMERIC` umgeändert.
Dazu schaute ich mir die verfügbaren Spalten in den Tabellen `pg_attribute` [7] und `pg_type` [8] an:

```
testdb=# UPDATE pg_attribute SET atttypid=(SELECT oid FROM pg_type WHERE typname
='numeric') WHERE attname='attr1' AND attrelid='test'::regclass;
UPDATE 1
```

Möchte man nach dem `UPDATE`-Befehl den vorher hinzugefügten Datensatz auslesen, wird angezeigt, dass das Attribut `attr1` kein `VARCHAR` mehr ist:

```
testdb=# SELECT * FROM test WHERE attr1='wert1';
FEHLER:  ungültige Eingabesyntax für Typ numeric: "wert1"
ZEILE 1: SELECT * FROM test WHERE attr1='wert1';
```

Erfolgt ein `SELECT *` - Befehl, wird kein Inhalt angezeigt. Alle Daten wurden daher von der Tabelle gelöscht:

```
testdb=# SELECT * FROM test;
```

5. Frage 4

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

PostgreSQL bietet folgende 5 Index-Typen an: B-Baum, Hash, GiST, SP-GiST und GIN. Standardmäßig erstellt der `CREATE INDEX` – Befehl ein B-Baum Index.

In der folgenden Tabelle wird gezeigt, welche Operationen von den jeweiligen Index-Typen unterstützt werden:

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	

[9]

6. Frage 5

Wie sind B-Bäume grundsätzlich aufgebaut?

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

1.) Jeder Knoten hat mindestens k und maximal 2k Einträge. Ausnahme: die Wurzel kann zwischen 1 und 2k Einträge haben.

2.) Wenn ein Knoten n Einträge hat, hat er n+1 Verweise auf seine Söhne. Ausnahme: Blätter haben keine bzw. undefinierte Verweise.

3.) Ein Blatt muss folgendermaßen organisiert sein (siehe Abbildung 1):

- *Wenn k1 bis 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“* (Moritz Theile, TU-München, B-Bäume S. 5, [10])

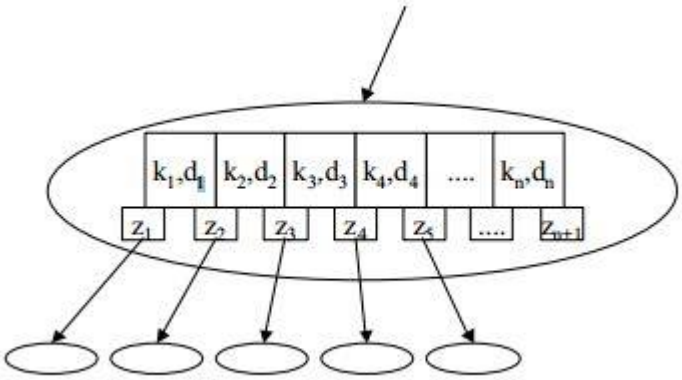

Abbildung 1: B-Baum-Blatt [Abb1_2]

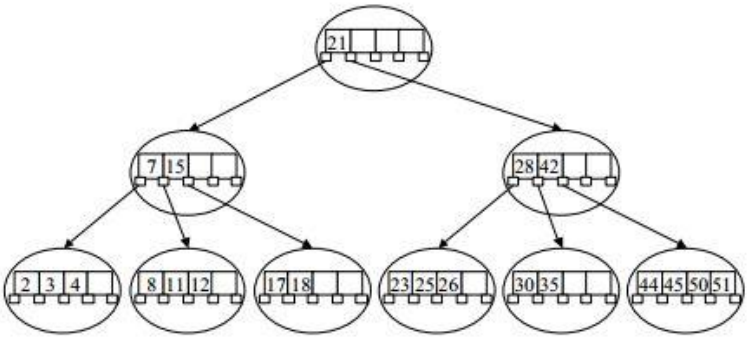

Abbildung 2: Beispiel für einen B-Baum [Abb1_2]

Jedes Blatt kann eine unterschiedliche Anzahl von Einträgen besitzen, dadurch ist nur ein sehr lokaler Aufwand notwendig, um den Baum in Balance zu halten. [10]

7. Frage 6

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 anhand der Höhe und des Grades berechnen, da die Wege von der Wurzel zu einem Blatt stets gleich lang sind. Der Weg entspricht dabei der Höhe bzw. der Ordnungszahl (= n).

Jeder Knoten besitzt ein Key-Value-Paar. Die Keys werden dabei aufsteigend sortiert.
Die inneren Knoten, abgesehen von den Blättern, mit n Einträgen besitzen n+1 Kinder.

[11], ([12], S.5)

8. Frage 7

Wie verläuft die Suche bei B-Bäumen?

Suche in einem B-Baum:

„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.

> *Existiert dieser Schlüssel, gehen wir zum Nachfolger-Knoten links von diesem Schlüssel über.*
>
> *Existiert der Schlüssel nicht, gehen wir zum letzten Nachfolger-Knoten über.*

3. Wenn wir bei einer null-Referenz landen, ist die Suche erfolglos." (Thomas Ottmann, Algorithmen und Datenstrukturen (19 – B-Bäume), Folie 8, [13])

Suche innerhalb eines Knotens:

„Hier sind prinzipiell verschiedene Verfahren denkbar:

- *lineare Suche*
- *binäre Suche*
- *. . .*

Allerdings beeinträchtigt die Suche innerhalb eines Knotens die Rechenzeit eher wenig, weil diese hauptsächlich durch die Anzahl der Zugriffe auf den Hintergrundspeicher (Festplatte) beeinflusst wird." (Thomas Ottmann, Algorithmen und Datenstrukturen (19 – B-Bäume), Folie 9, [13])

9. Frage 8

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

9.1 B+-Baum

B+-Bäume sind im Schlüsselbereich genauso aufgebaut wie B-Bäume. Für die eigentlichen Daten wird ein eigener Speicherbereich verwendet, welche getrennt von den Schlüsseln liegen. [14]

„Knotenstruktur des B+-Baumes:

1. *Die Blattknoten enthalten ausschließlich Nutzdaten und sind in einer linearen Liste geordnet.*
2. *Die inneren Knoten enthalten nur die Schlüsselwerte mit Verweisen auf die Blattknoten. Sie sind wie B-Bäume aufgebaut.*
3. *Ein linker Zeiger eines inneren Knoten verweist auf ein Blatt, in dem nur Nutzdaten mit kleineren Schlüsselwerten enthalten sind.*
4. *Ein rechter Zeiger verweist auf ein Blatt mit größeren Schlüsselwerten.*

Lesen	*Der Suchprozess startet in der Wurzel und sucht im B-Baum-Bereich nach dem Zeiger, der auf den entsprechenden Satz verweist. Die maximale Anzahl der Zugriffe ist die Höhe des Baumes h*
Schreiben	*Wie beim Lesen wird der Zielblock ermittelt. Im Falle des Überlaufs wird der Block gesplittet und der Indexknoten um einen Repräsentanten ergänzt. Die Anzahl der Zugriffe ist minimal 1 und maximal 2h+1.*
Löschen	*Wie bei den B-Bäumen können Blätter mit zu wenigen Sätzen entstehen (Unterlauf), die das Zusammenlegen von Knoten erforderlich macht. Die Anzahl der Zugriffe ist minimal 1 und maximal h + 1 bei Unterlauf, wenn sich die Umverteilung bis zur Wurzel fortpflanzt.*
Speichereffizienz	*Die Schlüsselwerte werden doppelt gespeichert, daher gibt es einen etwas höheren Platzbedarf. Der Füllgrad der Blätter ist zu 50 % durch die Strukturvorschriften gewährleistet.“* (FH Köln, Campus Gummersbach, B+-Baum, [14])

9.2 B*-Bäume

B*-Bäume verzichten im Gegensatz zu B-Bäumen auf die Schlüsselinformationen in inneren Knoten und die Schlüssel liegen zusammen mit den assoziierten Informationen auf der Blattebene. [15]

*„**Eigenschaften und Vorteile von B*-Baumen:***

- *Die Paare (x, a) stehen nur auf den Blättern des B*-Baumes.*
- *Die redundant gespeicherten Schlüssel in den inneren Knoten erfordern nur einen geringfügigen zusätzlichen Speicherbedarf.*
- *Bei vorgegebener Seitengröße können auf den inneren Knoten erheblich mehr Paare der Form (x, p) als Tripel der Form (x, a, p) gespeichert werden.*
    - *Dies bewirkt einen höheren Verzweigungsgrad und somit eine niedrigere Höhe des Baumes.*
    - *Schnelleres Suchen und Manipulieren*
- *Löschen geschieht immer auf einem Blatt.*
    - *Der Löschalgorithmus wird einfacher.*
- *Die Folge der Blätter eines B*-Baumes kann als sequentielle Datei aufgefasst werden.*
- *Ein B*-Baum kann auch aufgefasst werden als eine sequentielle Datei von Blattern zusammen mit einem Indexteil, der ein B-Baum ist. Der B-Baum enthält dabei nur die Schlüssel ohne assoziierte Information.“* (Peter Becker, Varianten von B-Bäumen, [15])

9.3 Präfix B+-Bäume

Das Ziel bei dieser Baumart ist es, Speicherplatz zu sparen. Hier kommen keine Schlüssel zum Einsatz. Stattdessen verwendet man zur Unterscheidung lediglich eine kürzere Zeichenfolge bzw. nur ein Zeichen. Dadurch ist es möglich, mehrere Schlüssel in einem Knoten zu unterbringen. Die Folge davon ist, dass die Anzahl der Verzweigungen ansteigt und die Höhe des Baums abnimmt, was auch zu niedrigeren Zugriffszeiten führt. ([12], S. 15)

9.4 Binärbäume

Bei den Binärbäumen besitzt jeder Knoten maximal 2 Nachfolger, dadurch wird die Struktur in einen linken und einen rechten Teilbaum aufgeteilt. Alle Knoten im linken Teilbaum sind niedrigwertiger als der betrachtete Knoten, alle Knoten im rechten Teilbaum hingegen höherwertiger. Die Knoten, die keinen Nachfolger haben, werden als „Blattknoten“ bezeichnet. ([16], S. 166)

Die exakte Definition von Binärbäumen lautet folgendermaßen:
„Eine Binärbaum B ist eine endliche Menge B von Elementen (Knoten) für die gilt: B ist entweder leer (leerer Baum, Nullbaum) oder es existiert ein ausgezeichneter Knoten W (die Wurzel), so dass alle übrigen Knoten ein geordnetes Paar zweier diskunkter Bäume BL und BR den linken Teilbaum und den rechten Teilbaum bilden.“ (Harmut Ernst, Grundlagen und Konzepte der Informatik, S. 586, [17])

Vorteile:

- schnelle Suche im Baum aufgrund des Binärverfahrens möglich
- Datenstruktur ist für rekursive Bearbeitung geeignet
  ➔ jeder Knoten kann so behandelt werden als wäre er die Wurzel eines ganzen Binärbaumes

Nachteile:

- Baum kann bei häufiger Hinzufügung und Entfernung von Datenelementen degenerieren

([16], S. 166)

10. Quellenangaben

[1] Oracle Corporation (2015). Chapter 21 INFORMATION_SCHEMA Tables [Online]. Available at: http://dev.mysql.com/doc/refman/5.6/en/information-schema.html [zuletzt abgerufen am 19.02.2015]

[2] The PostgreSQL Global Development Group (1996, 2015). Chapter 44. System Catalogs [Online]. Available at: http://www.postgresql.org/docs/9.4/static/indexes-types.html [zuletzt abgerufen am 19.02.2015]

[3] Oracle Corporation (1996, 2002). The Data Dictionary [Online]. Available at: https://docs.oracle.com/html/A96524_01/c05dicti.htm [zuletzt abgerufen am 19.02.2015]

[4] Oracle Corporation (2014). DBA_TABLES [Online]. Available at: http://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_4155.htm#REFRN23286 [zuletzt abgerufen am 19.02.2015]

[5] Frank Wiles (2002). Performance Tuning PostgreSQL [Online]. Available at: http://www.revsys.com/writings/postgresql-performance.html [zuletzt abgerufen am 19.02.2015]

[6] Shridhar Daithankar, Josh Berkus (2003). Tuning PostgreSQL for performance [Online]. Available at: http://www.varlena.com/GeneralBits/Tidbits/perf.html [zuletzt abgerufen am 19.02.2015]

[7] The PostgreSQL Global Development Group (1996, 2015). Chapter 48.7. pg_attribute [Online]. Available at: http://www.postgresql.org/docs/9.4/static/catalog-pg-attribute.html [zuletzt abgerufen am 22.02.2015]

[8] The PostgreSQL Global Development Group (1996, 2015). Chapter 48.52. pg_type [Online]. Available at: http://www.postgresql.org/docs/9.4/static/catalog-pg-type.html [zuletzt abgerufen am 22.02.2015]

[9] The PostgreSQL Global Development Group (1996, 2015). Chapter 11.2. Index Types [Online]. Available at: http://www.postgresql.org/docs/9.4/static/indexes-types.html [zuletzt abgerufen am 22.02.2015]

[10] Moritz Theile, TU-München (2001). B-Bäume [Online]. Available at: http://wwwbayer.in.tum.de/lehre/WS2001/HSEM-bayer/BTreesAusarbeitung.pdf [zuletzt abgerufen am 22.02.2015]

[11] Ulf Peter Schroeder, Universität Paderborn (2002). B-Bäume [Online]. Available at: http://www2.cs.uni-paderborn.de/cs/ag-monien/LEHRE/SS03/DuA/B-baeume.pdf [zuletzt abgerufen am 22.02.2015]

[12] Ludwig Bachmaier (2003). B-Bäume [Online]. Available at: http://www.informatik-forum.at/attachment.php?attachmentid=948 [zuletzt abgerufen am 22.02.2015]

[13] Thomas Ottmann (2003). Algorithmen und Datenstrukturen (19 – B-Bäume) [Online]. Available at: https://electures.informatik.uni-freiburg.de/portal/download/59/6149/INFO2-19-B_Baeume_4up.pdf [zuletzt abgerufen am 25.02.2015]

[14] FH Köln, Campus Gummersbach (2010). B+-Baum [Online]. Available at: http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/B-Plus-Baum [zuletzt abgerufen am 26.02.2015]

[15] Peter Becker, Hochschule Bonn-Rhein-Sieg (2012). Varianten von B-Bäumen [Online]. Available at: http://www2.inf.fh-rhein-sieg.de/~pbecke2m/ordbs/b-baum2.pdf [zuletzt abgerufen am 27.02.2015]

[16] Wolfgang H. Janko (1998, 2. Auflage). Informationswirtschaft 1: Grundlagen der Informatik für die Informationswirtschaft. ISBN: 9783540648123 [zuletzt abgerufen am 27.02.2015]

[17] Harmut Ernst (2000, 2. Auflage). Grundlagen und Konzepte der Informatik. ISBN: 3528157178 [zuletzt abgerufen am 27.02.2015]

11. Abbildungsverzeichnis

[Abb1_2] Moritz Theile, TU-München (2003). B-Bäume [Online]. Available at: http://wwwbayer.in.tum.de/lehre/WS2001/HSEM-bayer/BTreesAusarbeitung.pdf [zuletzt abgerufen am 19.02.2015]