Hochschule Darmstadt

Benutzerdokumentation zu DBJ

Marc Nesello
14.02.2012
Jens Weibler
09.02.2013
Letzte Änderung: 21.02.2013

Inhaltsverzeichnis

Installation

Systemvoraussetzungen und Installation

Dbj ist momentan nur mit Linux getestet. Die drei Programme dbj, dbjstart und dbjstop können weitergegeben werden und sind in einem Linux nach Installation einiger Bibliotheken lauffähig:

- libboost-program-options
- libluabind
- libpcre3

Allgemeines

Seitengröße

4096 Byte (Standardmäßig – dbj-Varianten können eine andere Seitengröße haben)

Puffergröße

1000 Seiten = 4096000 Byte = 4096 KiB

Datentypen

Integer

Größe (NULLABLE): 5 Byte

Größe (NOT NULLABLE): 4 Byte

Indizierbar: Ja

Varchar

Größe (NULLABLE): Länge der Zeichenkette + 3 Byte (Beschränkt durch maximale Tupelgröße auf 4056 Byte)

Größe (NOT NULLABLE): Länge der Zeichenkette + 2 Byte (Beschränkt durch maximale Tupelgröße auf 4056 Byte)

Indizierbar: Derzeit nicht

Starten und Stoppen des Systems

Starten von DBJ

Mit dem Befehl `dbjstart` wird DBJ gestartet. Dadurch werden die Shared Memory Bereiche für den Datenbankpuffer, die Sperrliste und Wartegraphen, sowie für den Log Manager initialisiert. Falls noch keine Datenbank angelegt wurde, wird hierbei auch der Systemkatalog angelegt.

Benutzung von DBJ

Wurde das System gestartet, kann nun über die Eingabe von `dbj` die DBJ-Befehlszeile gestartet werden. Damit wird eine DBJ-Sitzung gestartet. Mit dem Start einer Sitzung wird automatisch eine Transaktion gestartet. Um die Befehlszeile zu verlassen gibt man `exit;` ein. Dadurch wird die

aktuelle Transaktion automatisch zurückgesetzt und damit beendet. Außerdem wird beim Beenden der Sitzung der Puffer geleert und Seiten, die mit dirty markiert, aber nicht in Benutzung sind, werden auf die Festplatte geschrieben. Es können mehrere DBJ-Sitzungen gleichzeitig gestartet und genutzt werden.

Die DBJ-Befehlszeile kann mit verschiedenen Optionen gestartet werden. Die Usage kann ausgegeben werden indem man als Parameter – angibt, also `dbj -`

Die weiteren Optionen sind:

`-debugkey <Buchstabe>` Alternative Angabe eines Tastenkürzels für Debug-CLI (Standard: Alt+D)

`-scriptkey <Buchstabe>` Alternative Angabe eines Tastenkürzels für Scripting-CLI (Standard: Alt+S)

`-filename <Dateiname>` Liest die angegebene Datei ein und arbeitet die SQL-Befehle in der Datei ab

`-luascript <Dateiname>` Liest die angegebene Datei als Lua-Skript ein

`-verbose` Gibt das jeweils ausgeführte SQL-Statement aus. Diese Option ist hilfreich, wenn eine Datei eingelesen werden soll und in der Ausgabe die SQL-Statements angezeigt werden sollen.

`-plan` Gibt vor jedem SQL-Statement den optimierten Ausführungsplan aus. (Entspricht `SET EXPLAIN MODE ON`)

`-stop` Stoppt die Sitzung bei einem Fehler, anstatt lediglich eine Fehlermeldung auszugeben und die Transaktion zurückzusetzen.

Beenden von DBJ

Mit `dbjstop` wird DBJ gestoppt. Dabei werden die Shared Memory Bereiche zerstört. Ist das System jedoch noch in Benutzung, weil noch Transaktionen offen sind, so wird eine Fehlermeldung ausgegeben und das System nicht beendet. Um das System dennoch zu beenden gibt man `dbjstop force` ein.

Datenbankdateien

In DBJ entspricht eine Tabelle oder ein Index einem Segment. Ein Segment enthält also alle Seiten einer Tabelle oder eines Indexes. Ein Segment wird in einer Datei gespeichert. Dabei haben die Dateien die Form SegXXX.dbj wobei XXX die Nummer des Segments ist. Die Segmente mit Nummern ab 32769 sind Indexe und die Segmente mit einer kleineren Zahl sind Tabellen. Da die Seitengröße fest ist (4096 Byte) lässt sich anhand der Größe der Segmentdatei erkennen wie viele Seiten das Segment enthält.

Systemkatalog

DBJ erstellt beim erstmaligen Starten einen Systemkatalog. Dieser besteht aus drei Tabellen und vier Indexen.

SYSTABLES

Die Tabelle SYSTABLES speichert die Namen, Table IDs, Column_Counts und Tuple_Counts der angelegten Tabellen, auch von den Tabellen des Systemkatalogs. Der Tuple_Count wird durch den Befehl RUNSTATS aktualisiert.

SYSCOLUMNS

Hier werden die Namen, Datentypen und weitere Attribute der Spalten gespeichert. Jede Spalte ist mit genau einer Table ID verknüpft und damit genau einer Tabelle zugeordnet.

SYSINDEXES

Diese Tabelle speichert die Namen der Indexe und speichert auch welche Spalten auf welchen Tabellen durch diese Indexe indiziert werden.

IDX_SYSTABLES_TABLEID_ID

Dieser Index indiziert die TableId in der Tabelle SYSTABLES. Dadurch kann das System ohne Tablescan die Daten einer TableId, wie Name der Tabelle auslesen.

IDX_SYSCOLUMNS_TABLEID_ID

Mit diesem Index wird die TableId auf SYSCOLUMNS indiziert. Hiermit lassen sich also ohne Tablescan alle Spalten erfassen die zu einer Tabelle gehören.

IDX_SYSINDEXES_TABLEID_ID

Dieser Index indiziert die TableId auf SYSNDEXES. Damit lässt sich feststellen welche Indexe auf eine Tabelle definiert sind, ohne einen Tablescan auf SYSINDEXES zu nutzen.

IDX_SYSINDEXES_INDEXID_ID

Hiermit werden die IndexIds in SYSINDEXES indiziert. Somit lassen sich die Informationen zu einem bestimmten Index auslesen ohne einen Tablescan auszuführen.

Logdateien

Logdateien befinden sich im Arbeitsverzeichnis von DBJ und besitzen die Endung .log. Die Logdateien enthalten verschiedene Logeinträge. Jeder Logeintrag beginnt mit I; oder R;. Überschreitet eine Logdatei 10 MB, wird vom Log Manager eine neue Logdatei angelegt.

Format der Logeinträge

Das Format ist in der unteren Grafik dargestellt. Die einzelnen Informationen werden durch ein Semikolon getrennt.

Logeintrag für Datenoperationen

R	LSN	vorige LSN	Transaktions ID	Primary Tupel ID	Alte Secondary Tupel ID	Neue Secondary Tupel ID	Segment ID	Länge des Before Image	Before Image	Länge des After Image	After Image	Transaktions Typ

Logeintrag für Indexoperationen

I	LSN	vorige LSN	Transaktions ID	Page ID	Segment ID	Länge des Before Image	Before Image	Länge des After Image	After Image	Transaktions Typ

- **LSN** Log Sequence Number setzt sich aus Dateinummer und Offset in der Datei zusammen
- **Vorige LSN** ist die LSN des Vorgänger-Logeintrags
- **Transaktions ID** identifiziert die Transaktion
- **Page ID** Identifiziert die Seite innerhalb des Segments
- **Segment ID**
- **Länge des Before und After Image** in Bytes
- **After und Before Image**
- **Primary Tupel ID** identifiziert das Tupel, besteht aus Page ID und Slot ID
- **Alte und neue Secondary Tupel ID** Ist das Tupel ausgelagert, so hat es eine Secondary Tupel ID. Da sich die Secondary Tupel ID ändern kann, werden alte und neue Secondary Tupel ID gespeichert
- **Transaktionstyp** Identifiziert den Operationstyp des Logeintrags. Mögliche Typen sind
    - 1 = Commit
    - 2 = Rollback
    - 3 = Insert (bei Datenseiten)
    - 4 = Delete (bei Datenseiten)
    - 5 = Update (bei Datenseiten)
    - 8 = Compensate_Insert (Redo Insert oder Undo Delete-Operation auf einer Datenseite)
    - 9 = Compensate_Delete (Redo Delete oder Undo Insert-Operation auf einer Datenseite)
    - 10 = Compensate_Update (Redo Update oder Undo Update-Operation auf einer Datenseite)
    - 16 = Integer_Operation (Operation auf einem Index für Datentyp Integer)
    - 17 = Integer_Redo (Redo-Operation auf einem Index für Datentyp Integer)
    - 18 = Integer_Undo (Undo-Operation auf einem Index für Datentyp Integer)

Storage Manager

Index Manager

Der Index Manager verwaltet die Indizes. Jeder Index wird in einer B^+-Baum-Struktur gespeichert. Der B^+-Baum hat zwei Knotentypen. Innere Knoten, welche nur Verweise auf Kindknoten speichern und Blätter, welche die eigentlichen Daten (Verweis auf das Tupel auf der Datenseite) enthalten. Die Knoten werden solange befüllt bis diese keinen Platz mehr bieten. Ist der Knoten voll, wird dieser gesplittet, sodass zwei Knoten mit der Hälfte an Schlüsseln entstehen.

Derzeit werden nur Indizes für Integer-Werte unterstützt. Ein Blatt kann maximal 582 Schlüssel aufnehmen. Ein innerer Knoten kann 679 Schlüssel plus den Verweis auf den ersten Kindknoten aufnehmen, sodass ein Knoten maximal 680 Verweise auf Kindknoten enthalten kann. Des Weiteren hat jeder Index ein Freispeicherverzeichnis und daher mindestens eine FSV-Seite. Im FSV des Index werden nur ungenutzte Indexseiten gespeichert. Falls also eine Indexseite aus einem Baum ausgehängt wird, weil sie nicht mehr gebraucht wird, wird diese ins FSV des Index eingetragen. Außerdem hat der Wurzelknoten von einem Index immer die PageId 1.

Record Manager

Der Record Manager verwaltet die Datenseiten und die darauf gespeicherten Tupel. Auf einer Datenseite lassen sich maximal 255 Tupel oder 4070 Byte speichern. Die maximale Länge eines Tupels beträgt also 4070 Byte. Der Record Manager versucht möglichst die Seiten vollzuschreiben bevor neue Seiten erstellt werden. Für jedes Tupel muss ein Sloteintrag existieren, welcher 5 Byte belegt. Wird also eine leere Seite beschrieben, müssen 21 Byte für den Header berücksichtigt werden. Es sind also nur 4075 Byte nutzbar. Integer-Werte belegen 4 oder 5 Byte und Varchar-Werte benötigen einen variablen Speicherplatz (Länge der Zeichenkette + 2 oder 3 Byte).

$$Anzahl\ der\ Tupel\ pro\ Seite = \min\{\left(\frac{4075}{Tupelgröße + 5}\right), 255\}$$

$$Anzahl\ der\ Seiten = Anzahl\ der\ Tupel / Anzahl\ der\ Tupel\ pro\ Seite$$

Soll also z.B. die Seite mit INTEGER-Tupeln (5 Byte) vollgeschrieben werden, so sind neben dem Tupel (5 Byte) ein Sloteintrag (5 Byte) nötig. Es sind also pro Tupel 10 Byte Speicherplatz nötig. D.h. es ließen sich 407 Integer-Werte auf eine Seite schreiben, allerdings können nur 255 Slots und damit Werte auf eine Seite geschrieben werden. Daher wird eine zweite Seite angefordert für die restlichen 152 Werte.

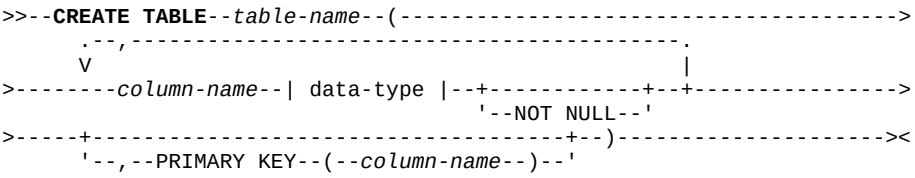
Außerdem besitzt der Record Manager ein Freispeicherverzeichnis, welches pro Datenseite einen Eintrag bereithält. Auf eine FSV-Seite werden 254 Einträge gespeichert, sodass pro 254 Datenseiten eine FSV-Seite benötigt wird.

Unterstützte SQL-Befehle

DDL (Data Definition Language)

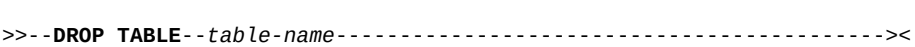
CREATE TABLE

Erstellt eine neue Tabelle mit den angegebenen Spalten und ggf. mit einem Primärschlüssel.



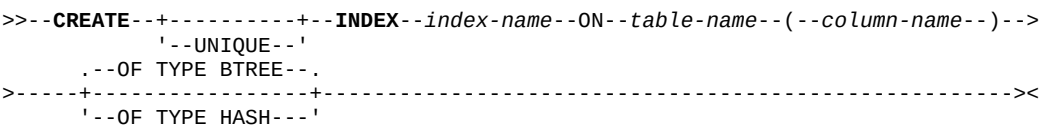
DROP TABLE

Löscht die angegebene Tabelle.



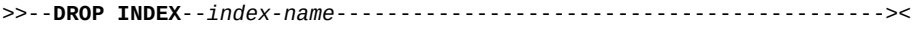
CREATE INDEX

Erstellt einen neuen Index auf die angegebene Spalte der Tabelle. Durch die Nutzung des Schlüsselwerts UNIQUE kann sichergestellt werden, dass in der jeweiligen Spalte jeder Wert nur einmal vorkommt.



DROP INDEX

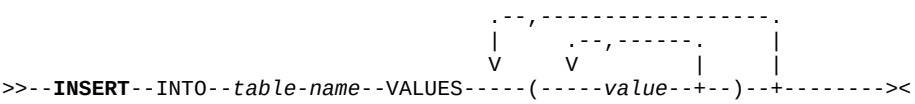
Löscht den angegebenen Index.



DML (Data Manipulation Language)

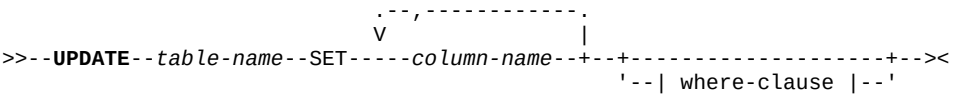
INSERT

Fügt ein Tupel in die angegebene Tabelle an.



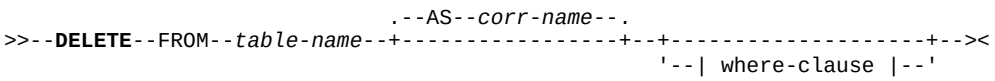
UPDATE

Aktualisiert die durch die Where-Klausel selektierten Tupel einer Tabelle. Wird keine Where-Klausel angegeben, werden alle Tupel der Tabelle mit dem jeweiligen Wert versehen.

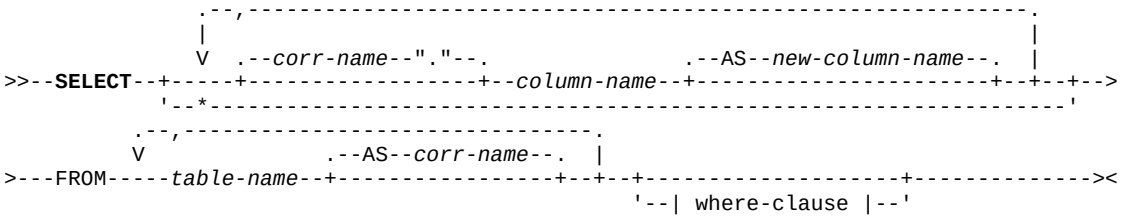


DELETE

Löscht die durch die Where-Klausel selektierten Tupel einer Tabelle. Wird keine Where-Klausel angegeben, werden alle Tupel der Tabelle gelöscht.

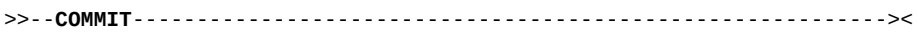


SELECT



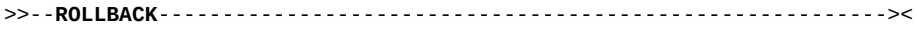
COMMIT

Beendet die aktuelle Transaktion, schreibt deren Änderungen fest und gibt die Sperren dieser Transaktion frei.



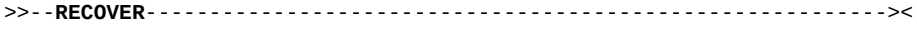
ROLLBACK

Verwirft alle Änderungen der aktuellen Transaktion und beendet diese.



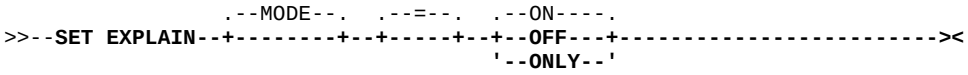
RECOVER

Führt eine Wiederherstellung durch, z.B. nach einem Absturz der Datenbank. Alle Operationen der Transaktionen welche committed wurden, werden nachvollzogen und falls diese nicht in der Datenbank eingebracht wurden, erneut eingebracht. Auswirkungen der Operationen von nicht beendeten Transaktionen werden rückgängig gemacht. Wird Recovery bei einer laufenden Datenbank ausgeführt, werden alle offenen Transaktionen verworfen. In dem Fall hat es also den Effekt, dass für alle offenen Transaktionen ein Rollback ausgeführt wird. Das Recovery kann nur ausgeführt werden, wenn andere Transaktionen keine Sperren mehr halten.

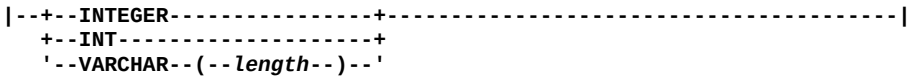


SET EXPLAIN

Stellt den Explain Modus ein oder aus. Wenn auf ON gesetzt, wird bei jeder SQL-Query ein Ausführungsplan angezeigt. Wird der Explain Modus auf ONLY gesetzt, wird ein Ausführungsplan angezeigt, das SQL-Statement wird jedoch nicht ausgeführt. Ist der Explain Modus auf ON entspricht, dass dem Aufruf von DBJ mit der Option `-plan`.

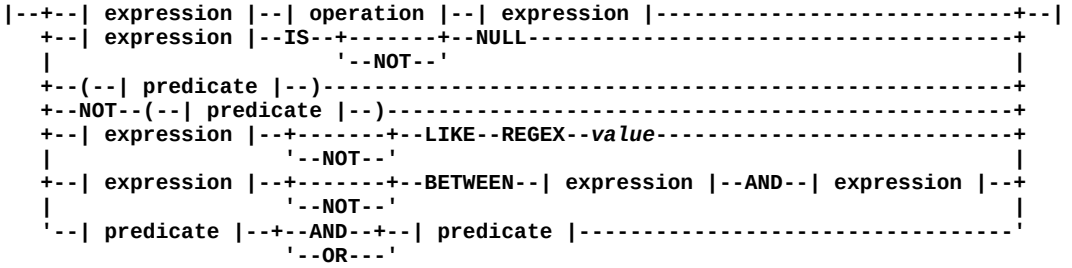


Data Type



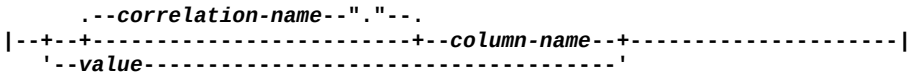
Where-Clause

Predicate



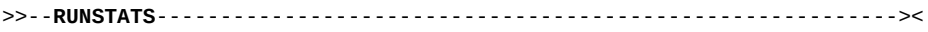
Operation

Expression



RUNSTATS

Aktualisiert die Spalte Tuple_Count in der Tabelle SYSTABLES. Tuple_Count gibt an, wie viele Tupel eine Relation enthält und wird benötigt um bei Joins die optimale Reihenfolge der Joins zu ermitteln.



Debug-Interface

Das Debug-Interface ist über die Tastenkombination Alt+D erreichbar (wenn nicht per Parameter geändert – siehe Benutzung von DBJ). Zum Verlassen wird die Tastenkombination Strg+D genutzt. Das Debug-Interface ermöglicht den Zugriff auf verschiedene Statistiken und Dump-Funktionen des DBMS.

Scripting-Interface

Das Scripting-Interface ist über die Tastenkombination Alt+S erreichbar (wenn nicht per Parameter geändert – siehe Benutzung von DBJ). Zum Verlassen wird die Tastenkombination Strg+D genutzt. Das Scripting-Interface ermöglicht über folgende Befehle die Interaktion mit der Skripting-Umgebung:

- Reload – lädt die per Parameter angegebene Lua-Skriptdatei neu
- Call <function>[parameter1, param2, param3] – ermöglicht eine in Lua definierte Methode direkt aufzurufen

SHOW TABLE_ALL

Diese Funktion zeigt die Informationen zu einer bestimmten Relation an.

SHOW TABLE_ALL INFO <tableName> zeigt zu jeder Seite der Relation die PageId, den Seitentyp (DataPage oder FSVPage), die Anzahl der Slots/Einträge und den belegten Speicherplatz auf der Seite an.

SHOW TABLE_ALL DUMP <tableName> zeigt zu jeder Seite der Relation einen ausführlichen Dump an (siehe SHOW TABLE_PAGE)

SHOW TABLE_PAGES

Diese Funktion zeigt die Informationen zu einem Intervall von Seiten einer Relation an.

SHOW TABLE_PAGES INFO <tableName> <pageId1> <pageId2> zeigt das gleiche wie SHOW TABLE_PAGE INFO für alle Seiten im Bereich von pageId1 bis pageId2 an.

SHOW TABLE_PAGES DUMP <tableName> <pageId1> <pageId2> zeigt das gleiche wie SHOW TABLE_PAGE DUMP für alle Seiten im Bereich von pageId1 bis pageId2 an.

SHOW TABLE_PAGE

Diese Funktion zeigt die Informationen zu einer bestimmten Seite einer Relation an.

SHOW TABLE_PAGE INFO <tableName> <pageId> zeigt zu der angegebenen PageId in der angegebenen Relation den Seitentyp, PageId, Anzahl der Slots/Einträge und den belegten Speicherplatz an.

SHOW TABLE_PAGE DUMP <tableName> <pageId> zeigt zu der angegebenen Seite in der angegebenen Relation einen kompletten Dump an. Der Dump beinhaltet die Informationen im Seitenheader (PageId, SegmentId, LSN, Freispeicherinfos, Anzahl der Sloteinträge), die Rohdaten der Seite (im Hex- und ASCII-Format), und die aus den Rohdaten aufbereitete Slotliste an.

Bei der Wahl von INFO wird folgendes ausgegeben:

- **PageType** Gibt an, ob es sich um eine FSV- oder Datenseite handelt
- **PageId**
- **Number of Entries/Slots** Anzahl der FSV-Einträge (auf FSV-Seiten) oder der Sloteinträge (auf Datenseiten)
- **Space used** Prozentuale Belegung des Speicherplatzes der Datenseite

Bei der Wahl von DUMP wird folgendes ausgegeben:

- **SegmentId**
- **PageId**
- **PageType**
- **isDirty** gibt an, ob die Seite modifiziert ist und die Änderungen noch nicht rausgeschrieben sind
- **isFix** gibt an, ob die Seite gefixt, also in Benutzung ist
- **EntriesCount** Anzahl der Einträge (bei FSV-Seiten)
- **LSN** Aktuelle LSN der Seite (bei Datenseiten)
- **FreeSpace** Freispeicher auf der Seite insgesamt (bei Datenseiten)
- **Length of biggest Consecutive Free Space Block** gibt den Speicher des größten zusammenhängenden Speicherbereichs auf der Seite an (bei Datenseiten)
- **Offset of biggest Consecutive Free Space Block** Adresse des größten zusammenhängenden Speicherbereichs auf der Seite (bei Datenseiten)
- **Space used** Prozentuale Belegung des Speicherplatzes der Datenseite
- **Anzahl der Slots** Gibt die Anzahl der Sloteinträge auf der Seite an. Diese Zahl ist nicht identisch mit der Anzahl der Tupel auf der Seite, da es auch leere Sloteinträge geben kann
- **SlotID** Slotid des jeweiligen Sloteintrags
- **Record Offset** Adresse auf der Seite vom zugehörigen Tupel
- **Record Length** Länge des zum Sloteintrag gehörenden Tupels
- **Record Type** gibt an, ob es sich um einen Primary, Secondary Record oder ein Placeholder TID handelt
- **FSI for page n: y** FSV-Eintrag für Seite n. Gibt an wie viel Freispeicher y (in 16 Byte-Blöcken) auf der Seite vorhanden ist
- **Freespace Block: z** Gibt an wie viel freier Speicher z (in 16 Byte-Blöcken) sich im größten freien Speicherblock auf der Seite befindet. Ist y = 0 und z=254, so existiert die jeweilige Seite nicht.

SHOW TRANSACTIONID

SHOW TRANSACTIONID zeigt die aktuelle TransaktionsId der aktuellen Sitzung an.

SHOW LM_STATS

Mit der Funktion LM_STATS können die Datenstrukturen des Lock Managers ausgegeben werden.

SHOW LM_STATS gibt die Sperrliste aus und beinhaltet dabei die Sperren aller Transaktionen.

SHOW LM_STATS current transaction gibt nur die Sperrliste mit den Sperren der aktuellen Transaktion aus

Die Sperrliste besteht aus Einträgen, mit ID, Transaktions ID, Sperrtyp, SegmentId, Page ID (65535 = Segmentsperre), SlotID (255 = Segment- oder Seitensperre), sowie den IDs der Vorgänger und Nachfolgereinträge. Mit <end> (der maximalen Anzahl an Locklisteneinträgen) wird angezeigt, dass der Eintrag keinen Nachfolger oder Vorgänger hat.

SHOW LM_STATS waitinglist gibt die Warteliste aus. In der Warteliste hat jeder Eintrag eine ID, die Transaktions ID der wartenden Transaktion und die ID des Locklisteneintrags auf den gewartet wird.

Auch hier werden ID des Vorgängers und Nachfolgers gespeichert. Der Eintrag <end> gibt an, dass es keinen Nachfolger oder Vorgänger gibt.

SHOW LM_STATS hashlist wird die Hashliste ausgegeben. Die Hashliste besteht aus 100 Einträgen, welche die Buckets repräsentieren und besitzt für jeden Eintrag eine ID und die ID des ersten Sperrlisteneintrags. Ist kein Sperrlisteneintrag in diesem Bucket, so wird <end> angezeigt. Weitere Einträge des Buckets lassen sich über die Nachfolger des ersten Sperrlisteneintrags ermitteln.

SHOW BM_STATS

Mit dieser Funktion lassen sich Statistiken und die Datenstrukturen des Buffer Managers anzeigen.

SHOW BM_STATS oder *SHOW BM_STATS hitratio* geben Statistiken für die Anzahl der physischen und logischen Zugriffe aus sowie die daraus berechnete Buffer Hit Ratio. Dabei werden die Statistiken getrennt für die aktuelle Transaktion und für alle Transaktionen ausgegeben.

Mit *reset BM_STATS* lassen sich die Statistiken über physische und logische Zugriffe und damit die Buffer Hit Ratio für die aktuelle Transaktion zurücksetzen.

SHOW BM_STATS pages zeigt die Seiten an, die sich derzeit im Puffer befinden.

- **Slot-of-Page** Slot, welchen die Seite im Puffer belegt
- **SegmentId**
- **PageId**
- **FixCount** gibt an wie oft die Seite derzeit genutzt wird
- **Dirty** Gibt an, ob die Seite modifiziert wurde, und die Modifizierungen noch nicht auf die Festplatte geschrieben wurden
- **LSN** Log Sequence Number gibt die LSN des Log Records an, zu der letzten Modifikation dieser Seite

SHOW BM_STATS lru gibt die LRU-Liste aus. Die LRU-Liste wird im Puffer verwaltet und speichert die Reihenfolge der Seiten nach der LRU-Strategie. Die zuletzt genutzte Seite steht ganz oben in der Liste. Die Seite die am längsten nicht verwendet wurde, steht ganz unten in der Liste.

- **Slot-in-LRU** Slot, welchen die Seite im Puffer belegt
- **Next** Slot des Nachfolgers in der LRU-Liste

SHOW BM_STATS hashmap zeigt die Hashliste an, welche die Seiten im Puffer verwaltet. Bei dieser Funktion werden die Buckets der Hashliste angezeigt und die jeweiligen Hasheinträge in den Buckets. Im Puffer stehen Slots für die Seiten zur Verfügung.

- **Slot-in-Hash** Slot, welchen die Seite im Puffer belegt
- **Next** Slot des Nachfolgers

SHOW LOG_PRINT

Die Funktion LOG_PRINT zeigt verschiedene Ausgaben des Log Managers an, welcher für das Schreiben der Logeinträge verantwortlich ist.

SHOW LOG_PRINT SHOWLOG <transactionId> lassen sich die Logeinträge zu einer bestimmten Transaktion anzeigen. Um beispielsweise Die Logeinträge der aktuellen Transaktion anzeigen zu lassen, kann zuerst mit SHOW TRANSACTIONID die Transaktions Id ermittelt werden um anschließend die Logeinträge dieser Transaktion auszugeben.

SHOW LOG_PRINT SHOWLOG <LSN1> <LSN2> werden alle Logeinträge zwischen den angegebenen LSN ausgegeben.

Die Ausgabe der Logeinträge entspricht der Darstellung im Abschnitt *Format der Logeinträge*.

Die LSN einer Logdatei kann man sich mit der Funktion *SHOW LOG_PRINT LISTLSN <FileNr>* anzeigen lassen.

SHOW INDEX_ALL

Diese Funktion zeigt alle Seiten eines Indexes (B+Baum-Seiten und FSV-Seiten) an.

SHOW INDEX_ALL INFO|DUMP <IndexId> zeigt alle Seiten des gewählten Index an. Die IndexId lässt sich über die Systemtabelle SYSINDEXES ermitteln. Dabei zeigt die Option *INFO* nur Kurzinformationen zu den Indexseiten angezeigt. Mit der Option *DUMP* werden zusätzliche alle Schlüssel der jeweiligen Seite ausgegeben. Die Ausgabe der einzelnen Seiten entspricht der Ausgabe von *SHOW INDEX_PAGE*.

Anmerkung: Um diese Funktion zu nutzen muss der Index geöffnet sein. Dies kann beispielsweise durch ein geeignetes select-Statement auf einer indizierten Spalte sichergestellt werden.

SHOW INDEX_PAGES

Diese Funktion zeigt ein Intervall von Seiten eines Index an.

SHOW INDEX_ALL INFO|DUMP <IndexId> <PageId1> <PageId2> zeigt alle Seiten im gewählten Intervall an. Die Ausgabe der einzelnen Seiten entspricht der Ausgabe von *SHOW INDEX_PAGE*.

Anmerkung: Um diese Funktion zu nutzen muss der Index geöffnet sein. Dies kann beispielsweise durch ein geeignetes select-Statement auf einer indizierten Spalte sichergestellt werden.

SHOW INDEX_LEAFS

Diese Funktion zeigt nur die Blattseiten eines Indexbaums an.

SHOW INDEX_LEAFS INFO|DUMP <IndexId> zeigt nur die Blattseiten des B+Baum –Indexes an. Die Ausgabe der einzelnen Seiten entspricht der Ausgabe von *SHOW INDEX_PAGE.*

Anmerkung: Um diese Funktion zu nutzen muss der Index geöffnet sein. Dies kann beispielsweise durch ein geeignetes select-Statement auf einer indizierten Spalte sichergestellt werden.

SHOW INDEX_FSI

Diese Funktion zeigt nur die FSV-Seiten des Index an.

SHOW INDEX_FSI INFO|DUMP <IndexId> zeigt nur die FSV-Seiten zum jeweiligen Index an. Die Ausgabe der einzelnen Seiten entspricht der Ausgabe von *SHOW INDEX_PAGE.*

Anmerkung: Um diese Funktion zu nutzen muss der Index geöffnet sein. Dies kann beispielsweise durch ein geeignetes select-Statement auf einer indizierten Spalte sichergestellt werden.

SHOW INDEX_PAGE

Diese Funktion zeigt eine bestimmte Seite des gewählten Index an.

SHOW INDEX_LEAFS INFO|DUMP <IndexId> <PageId> zeigt die Seite *PageId* des jeweiligen Indexes an. Bei Wahl der Option *INFO* werden folgende Informationen ausgegeben:

- **SegmentId**
- **PageId**
- **PrevId** PageId des Vorgängerblatts (nur bei Blattseiten)
- **NextId** PageId des Nachfolgerblatts (bei Blattseiten) oder die PageId der nächsten FSV-Seite (bei FSV-Seiten)
- **Number of Elements** Anzahl der Elemente auf der Seite
- **LastId** Zuletzt vergebene PageId (Bei FSV-Seiten)
- **SpaceLeft** Freier Speicherplätze für Verweise auf Kindknoten (bei Inneren Knoten)
- **FirstChild** PageId des ersten Kindknotens (bei Inneren Knoten)

Bei Wahl der Option *DUMP* wird zusätzlich folgendes ausgegeben:

- **[i] -> v (child: p)** bezeichnet den i-ten Eintrag auf der Seite. Dieser repräsentiert den Wert v und verweist auf den Kindknoten mit PageId p (bei inneren Knoten)
- **[v;p,s]** bezeichnet einen Schlüssel für Wert v welcher auf Slot s auf Datenseite p verweist (auf Blattseiten)
- **PageId # i : p** bezeichnet den i-ten Eintrag für eine freie Seite mit PageId p (auf FSV-Seiten)

Anmerkung: Um diese Funktion zu nutzen muss der Index geöffnet sein. Dies kann beispielsweise durch ein geeignetes select-Statement auf einer indizierten Spalte sichergestellt werden.

Trace-Funktion

DBJ hat eine Trace-Funktionalität und kann Tracefiles erstellen, die in einer Tracedatei vermerken, welche Funktionen aufgerufen wurden.

Um ein Trace zu starten, muss einfach für die Umgebungsvariable DBJ_TRACE_FILE ein Dateiname gesetzt sein. Dann wird der Trace in ebendieser Datei gespeichert.

Mit der Umgebungvariable DBJ_PERF_FILE wird die Laufzeit von Methoden in dbj gemessen und am Ende in der angegebenen Datei ausgegeben.

Beide Umgebungsvariablen können auch auf stdout bzw. stderr gesetzt werden. Die Ausgabe erfolgt dann auf der Konsole.