SYSTEM J - Konzeption und prototypische Umsetzung eines Praktikums zur Datenbanksystementwicklung

Hannes Moser

Fakultät für Mathematik und Informatik
Universität Jena

Studierendenprogramm der BTW, 01. März 2005

Übersicht

Was ist System J?

- Prototyp eines relationalen Datenbanksystems
- Enthält alle wichtigen Komponenten und Funktionen eines DBMS
- Vereinfachtes System, auf wesentliche Aspekte reduziert
- Implementiert im Rahmen eines Praktikums (1 Semester)
- So konzipiert, daß Aufgabenteilung möglich ist (Komponenten sind weitgehend getrennt implementierbar)

Was ist System J?

- Prototyp eines relationalen Datenbanksystems
- Enthält alle wichtigen Komponenten und Funktionen eines DBMS
- Vereinfachtes System, auf wesentliche Aspekte reduziert
- Implementiert im Rahmen eines Praktikums (1 Semester)
- So konzipiert, daß Aufgabenteilung möglich ist (Komponenten sind weitgehend getrennt implementierbar)

System J in Stichworten

- Unterstützt Untermenge der SQL-Norm
  - u.a. CREATE TABLE, SQL-INSERT, SELECT (mit Join), DELETE, ...
  - Nur Datentypen VARCHAR und INTEGER
- Anfrageoptimierung
- Mehrbenutzerbetrieb
- Transaktionen
- Erfüllung der ACID-Eigenschaften
- B*-Baum- sowie Hash-Indexe
- I/O-Optimierung mittels Systembuffer
- ...

System J in Stichworten

- Unterstützt Untermenge der SQL-Norm
  - u.a. CREATE TABLE, SQL-INSERT, SELECT (mit Join), DELETE, ...
  - Nur Datentypen VARCHAR und INTEGER
- Anfrageoptimierung
- Mehrbenutzerbetrieb
- Transaktionen
- Erfüllung der ACID-Eigenschaften
- B*-Baum- sowie Hash-Indexe
- I/O-Optimierung mittels Systembuffer
- ...

System J in Stichworten

- Unterstützt Untermenge der SQL-Norm
  - u.a. CREATE TABLE, SQL-INSERT, SELECT (mit Join), DELETE, ...
  - Nur Datentypen VARCHAR und INTEGER
- Anfrageoptimierung
- Mehrbenutzerbetrieb
- Transaktionen
- Erfüllung der ACID-Eigenschaften
- B*-Baum- sowie Hash-Indexe
- I/O-Optimierung mittels Systembuffer
- ...

Nicht unterstützte Funktionen

- Authentifizierung
- Logging
- Subselects
- User defined functions, stored procedures
- Sichten
- Sortierung
- umfangreichere Integritätsbedingungen
- weitere Datentypen
- Recovery
- ...

Praktikum

- idealerweise ca. 20-25 Studenten, Gruppen mit jeweils 2-3 Personen
- Eine Komponente pro Gruppe
- Verwendete Programmiersprache C++
- Implementiert auf Linux/AIX/Cygwin
- Vorgegeben sind
  - Schnittstellen der Komponenten
  - Hilfsmethoden (Speicherverwaltung, Fehlerbehandlung, Tracing, Latches, Testumgebung)
- Erstmals implementiert im WS04/05

Praktikum

- idealerweise ca. 20-25 Studenten, Gruppen mit jeweils 2-3 Personen
- Eine Komponente pro Gruppe
- Verwendete Programmiersprache C++
- Implementiert auf Linux/AIX/Cygwin
- Vorgegeben sind
  - Schnittstellen der Komponenten
  - Hilfsmethoden (Speicherverwaltung, Fehlerbehandlung, Tracing, Latches, Testumgebung)
- Erstmals implementiert im WS04/05

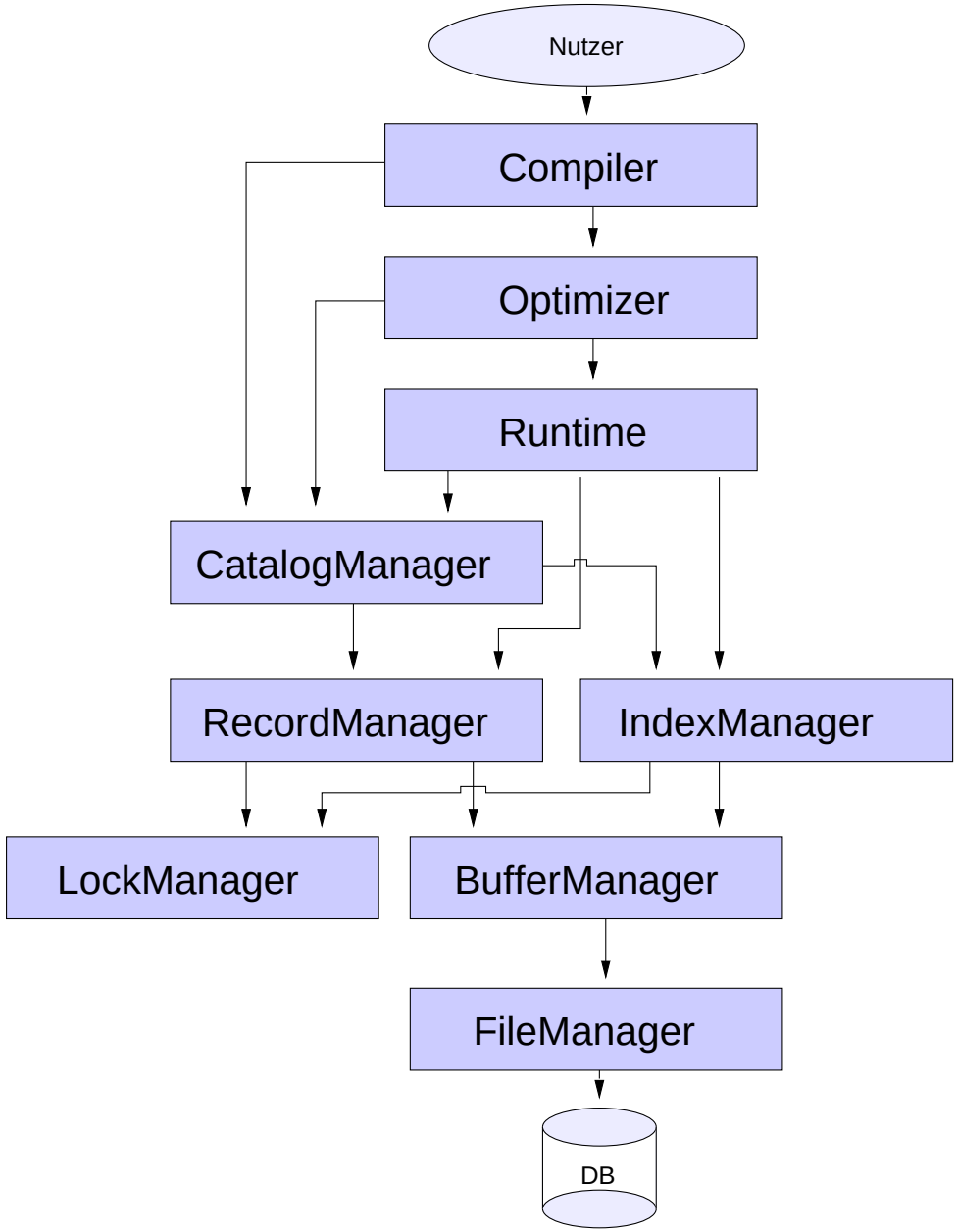
Praktikum

- idealerweise ca. 20-25 Studenten, Gruppen mit jeweils 2-3 Personen
- Eine Komponente pro Gruppe
- Verwendete Programmiersprache C++
- Implementiert auf Linux/AIX/Cygwin
- Vorgegeben sind
  - Schnittstellen der Komponenten
  - Hilfsmethoden (Speicherverwaltung, Fehlerbehandlung, Tracing, Latches, Testumgebung)
- Erstmals implementiert im WS04/05

Praktikum

- idealerweise ca. 20-25 Studenten, Gruppen mit jeweils 2-3 Personen
- Eine Komponente pro Gruppe
- Verwendete Programmiersprache C++
- Implementiert auf Linux/AIX/Cygwin
- Vorgegeben sind
  - Schnittstellen der Komponenten
  - Hilfsmethoden (Speicherverwaltung, Fehlerbehandlung, Tracing, Latches, Testumgebung)
- Erstmals implementiert im WS04/05

Die Architektur des System J



FileManager

- Bildet Datenbanksegmente auf Dateien ab (1:1)
- Verwaltung der geöffneten Dateien/Segmente
- Blockorientierte Schnittstelle
- Blockgröße = Seitengröße (4 kB)

FileManager

- Bildet Datenbanksegmente auf Dateien ab (1:1)
- Verwaltung der geöffneten Dateien/Segmente
- Blockorientierte Schnittstelle
- Blockgröße = Seitengröße (4 kB)

BufferManager

- Verwaltung auf Seitenbasis mittels Hash
- LRU-Algorithmus für Seitenverdrängung
- Systempuffer im Shared Memory
  - BufferManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- FORCE/NO STEAL (wir haben kein Logging!)

BufferManager

- Verwaltung auf Seitenbasis mittels Hash
- LRU-Algorithmus für Seitenverdrängung
- Systempuffer im Shared Memory
  - BufferManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- FORCE/NO STEAL (wir haben kein Logging!)

BufferManager

- Verwaltung auf Seitenbasis mittels Hash
- LRU-Algorithmus für Seitenverdrängung
- Systempuffer im Shared Memory
  - BufferManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- FORCE/NO STEAL (wir haben kein Logging!)

RecordManager

- Bildet Records in Datenbankseiten ab
- Einfache Byte-String-Records
- TID-Konzept zur Adressierung
- Update eingeschränkt

IndexManager

- Nur Indexe auf einzelne Spalten!
- UNIQUE sowie NON-UNIQUE möglich
- Unterstützte Indextypen
  - B*-Baum zur Unterstützung von Bereichsanfragen
  - Hash (geplant, aber noch nicht implementiert)
- Unterstützte Indexdatentypen
  - INTEGER
  - VARCHAR mit Begrenzung auf 256 Zeichen
- Jeder Index wird auf separatem Segment abgelegt
- Jeder Knoten des B*-Baumes wird auf eine Seite abgebildet
- B*-Baum-Verwaltung ist einfach gehalten (keine Rotationen beim Löschen)

IndexManager

- Nur Indexe auf einzelne Spalten!
- UNIQUE sowie NON-UNIQUE möglich
- Unterstützte Indextypen
  - B*-Baum zur Unterstützung von Bereichsanfragen
  - Hash (geplant, aber noch nicht implementiert)
- Unterstützte Indexdatentypen
  - INTEGER
  - VARCHAR mit Begrenzung auf 256 Zeichen
- Jeder Index wird auf separatem Segment abgelegt
- Jeder Knoten des B*-Baumes wird auf eine Seite abgebildet
- B*-Baum-Verwaltung ist einfach gehalten (keine Rotationen beim Löschen)

IndexManager

- Nur Indexe auf einzelne Spalten!
- UNIQUE sowie NON-UNIQUE möglich
- Unterstützte Indextypen
  - B*-Baum zur Unterstützung von Bereichsanfragen
  - Hash (geplant, aber noch nicht implementiert)
- Unterstützte Indexdatentypen
  - INTEGER
  - VARCHAR mit Begrenzung auf 256 Zeichen
- Jeder Index wird auf separatem Segment abgelegt
- Jeder Knoten des B*-Baumes wird auf eine Seite abgebildet
- B*-Baum-Verwaltung ist einfach gehalten (keine Rotationen beim Löschen)

IndexManager

- Nur Indexe auf einzelne Spalten!
- UNIQUE sowie NON-UNIQUE möglich
- Unterstützte Indextypen
  - B*-Baum zur Unterstützung von Bereichsanfragen
  - Hash (geplant, aber noch nicht implementiert)
- Unterstützte Indexdatentypen
  - INTEGER
  - VARCHAR mit Begrenzung auf 256 Zeichen
- Jeder Index wird auf separatem Segment abgelegt
- Jeder Knoten des B*-Baumes wird auf eine Seite abgebildet
- B*-Baum-Verwaltung ist einfach gehalten (keine Rotationen beim Löschen)

LockManager

- Sperrgranulat ist Datenbankseite
- Keine Sperrhierarchie
- Transaktionen sind Betriebssystem-Prozesse
- Arten von Sperren
  - Shared Lock
  - Exclusive Lock
- Deadlock-Auflösung mittels Timeout (5 s)
- Sperrtabelle im Shared Memory
  - LockManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- Verwaltung der Sperren per Hash

LockManager

- Sperrgranulat ist Datenbankseite
- Keine Sperrhierarchie
- Transaktionen sind Betriebssystem-Prozesse
- Arten von Sperren
  - Shared Lock
  - Exclusive Lock
- Deadlock-Auflösung mittels Timeout (5 s)
- Sperrtabelle im Shared Memory
  - LockManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- Verwaltung der Sperren per Hash

LockManager

- Sperrgranulat ist Datenbankseite
- Keine Sperrhierarchie
- Transaktionen sind Betriebssystem-Prozesse
- Arten von Sperren
  - Shared Lock
  - Exclusive Lock
- Deadlock-Auflösung mittels Timeout (5 s)
- Sperrtabelle im Shared Memory
  - LockManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- Verwaltung der Sperren per Hash

LockManager

- Sperrgranulat ist Datenbankseite
- Keine Sperrhierarchie
- Transaktionen sind Betriebssystem-Prozesse
- Arten von Sperren
  - Shared Lock
  - Exclusive Lock
- Deadlock-Auflösung mittels Timeout (5 s)
- Sperrtabelle im Shared Memory
  - LockManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- Verwaltung der Sperren per Hash

LockManager

- Sperrgranulat ist Datenbankseite
- Keine Sperrhierarchie
- Transaktionen sind Betriebssystem-Prozesse
- Arten von Sperren
  - Shared Lock
  - Exclusive Lock
- Deadlock-Auflösung mittels Timeout (5 s)
- Sperrtabelle im Shared Memory
  - LockManager läuft in mehreren konkurrierenden Prozessen
  - Prozesse greifen auf gemeinsamen Speicher zu
  - Sicherung mittels Latches
- Verwaltung der Sperren per Hash

CatalogManager

- Vereinfachter Zugriff auf Systemtabellen
- Methoden zur einfachen Verwaltung der Tabellen-Metainformationen
- Verwaltet 3 Systemtabellen
  - SYSTABLES
  - SYSCOLUMNS
  - SYSINDEXES
- Datentypen INT und VARCHAR
- Interpretation der Daten anhand der Metainformationen in Systemtabellen
- Metainformationen der Systemtabellen sind statisch einprogrammiert (aber nach außen transparent)
- Beim ersten Systemstart wird automatisch ein Katalog angelegt

CatalogManager

- Vereinfachter Zugriff auf Systemtabellen
- Methoden zur einfachen Verwaltung der Tabellen-Metainformationen
- Verwaltet 3 Systemtabellen
  - SYSTABLES
  - SYSCOLUMNS
  - SYSINDEXES
- Datentypen INT und VARCHAR
- Interpretation der Daten anhand der Metainformationen in Systemtabellen
- Metainformationen der Systemtabellen sind statisch einprogrammiert (aber nach außen transparent)
- Beim ersten Systemstart wird automatisch ein Katalog angelegt

CatalogManager

- Vereinfachter Zugriff auf Systemtabellen
- Methoden zur einfachen Verwaltung der Tabellen-Metainformationen
- Verwaltet 3 Systemtabellen
  - SYSTABLES
  - SYSCOLUMNS
  - SYSINDEXES
- Datentypen INT und VARCHAR
- Interpretation der Daten anhand der Metainformationen in Systemtabellen
- Metainformationen der Systemtabellen sind statisch einprogrammiert (aber nach außen transparent)
- Beim ersten Systemstart wird automatisch ein Katalog angelegt

CatalogManager

- Vereinfachter Zugriff auf Systemtabellen
- Methoden zur einfachen Verwaltung der Tabellen-Metainformationen
- Verwaltet 3 Systemtabellen
  - SYSTABLES
  - SYSCOLUMNS
  - SYSINDEXES
- Datentypen INT und VARCHAR
- Interpretation der Daten anhand der Metainformationen in Systemtabellen
- Metainformationen der Systemtabellen sind statisch einprogrammiert (aber nach außen transparent)
- Beim ersten Systemstart wird automatisch ein Katalog angelegt

CatalogManager

- Vereinfachter Zugriff auf Systemtabellen
- Methoden zur einfachen Verwaltung der Tabellen-Metainformationen
- Verwaltet 3 Systemtabellen
  - SYSTABLES
  - SYSCOLUMNS
  - SYSINDEXES
- Datentypen INT und VARCHAR
- Interpretation der Daten anhand der Metainformationen in Systemtabellen
- Metainformationen der Systemtabellen sind statisch einprogrammiert (aber nach außen transparent)
- Beim ersten Systemstart wird automatisch ein Katalog angelegt

Compiler

- Parsen der SQL-Anweisung
- Übersetzung der SQL-Anweisungen in einen Syntaxbaum
- Validierung dieses Syntaxbaumes
- Bestimmung der internen Spalten-, Tabellen- und Index-IDs
- SQL-Schlüsselwörter als Bezeichner verboten
  - Verboten also: SELECT SELECT SELECT FROM FROM FROM WHERE WHERE=WHERE
- Parser mit Bison realisiert

Compiler

- Parsen der SQL-Anweisung
- Übersetzung der SQL-Anweisungen in einen Syntaxbaum
- Validierung dieses Syntaxbaumes
- Bestimmung der internen Spalten-, Tabellen- und Index-IDs
- SQL-Schlüsselwörter als Bezeichner verboten
  - Verboten also: SELECT SELECT SELECT FROM FROM FROM WHERE WHERE=WHERE
- Parser mit Bison realisiert

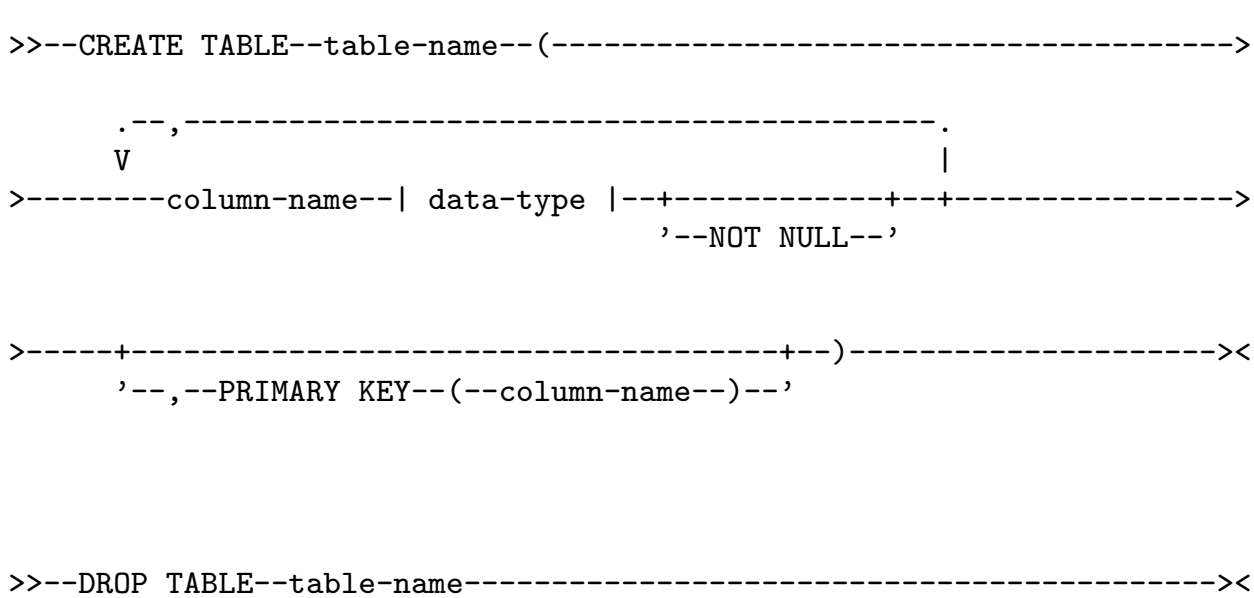
Compiler

- Parsen der SQL-Anweisung
- Übersetzung der SQL-Anweisungen in einen Syntaxbaum
- Validierung dieses Syntaxbaumes
- Bestimmung der internen Spalten-, Tabellen- und Index-IDs
- SQL-Schlüsselwörter als Bezeichner verboten
  - Verboten also: SELECT SELECT SELECT FROM FROM FROM WHERE WHERE=WHERE
- Parser mit Bison realisiert

Compiler

- Parsen der SQL-Anweisung
- Übersetzung der SQL-Anweisungen in einen Syntaxbaum
- Validierung dieses Syntaxbaumes
- Bestimmung der internen Spalten-, Tabellen- und Index-IDs
- SQL-Schlüsselwörter als Bezeichner verboten
  - Verboten also: SELECT SELECT SELECT FROM FROM FROM WHERE WHERE=WHERE
- Parser mit Bison realisiert

Unterstützte SQL-Anweisungen (1)



Unterstützte SQL-Anweisungen (2)

```
data-type:

 |--+--INTEGER----------------+-----------------------------------------|
    +--INT--------------------+
    '--VARCHAR--(--length--)--'
```

```
>>--COMMIT-----------------------------------------------------------><
```

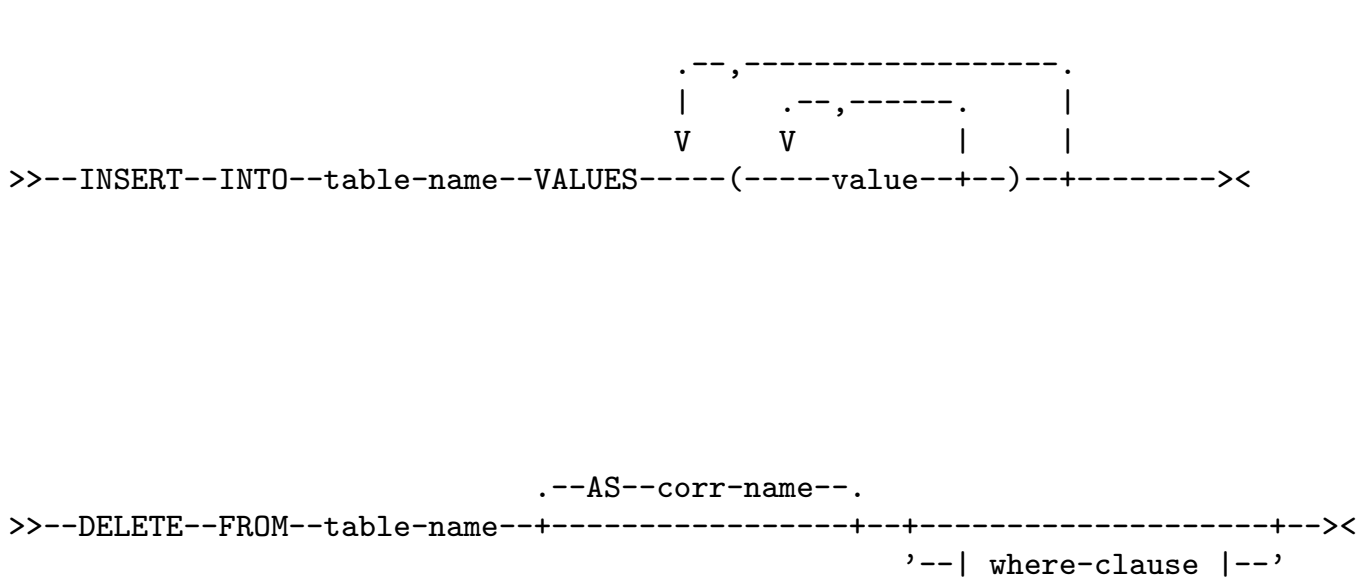
```
>>--ROLLBACK---------------------------------------------------------><
```

Unterstützte SQL-Anweisungen (3)

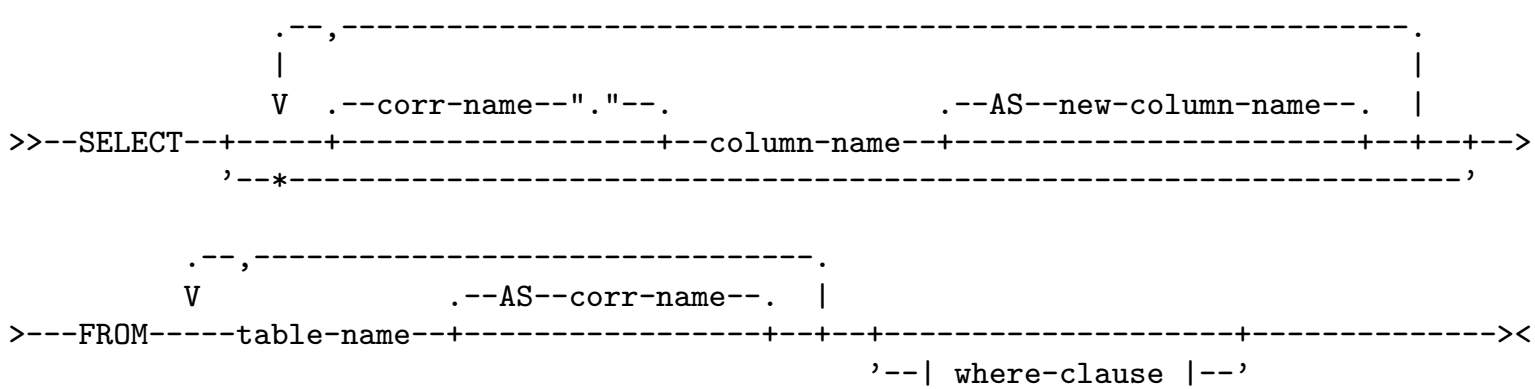
```
>>--CREATE--+----------+--INDEX--index-name--ON--table-name--(--column-name--)-->
            '--UNIQUE--'

      .--OF TYPE BTREE--.
>-----+-----------------+-------------------------------------------------------><
      '--OF TYPE HASH---'
```

```
>>--DROP INDEX--index-name----------------------------------------><
```

Unterstützte SQL-Anweisungen (4)



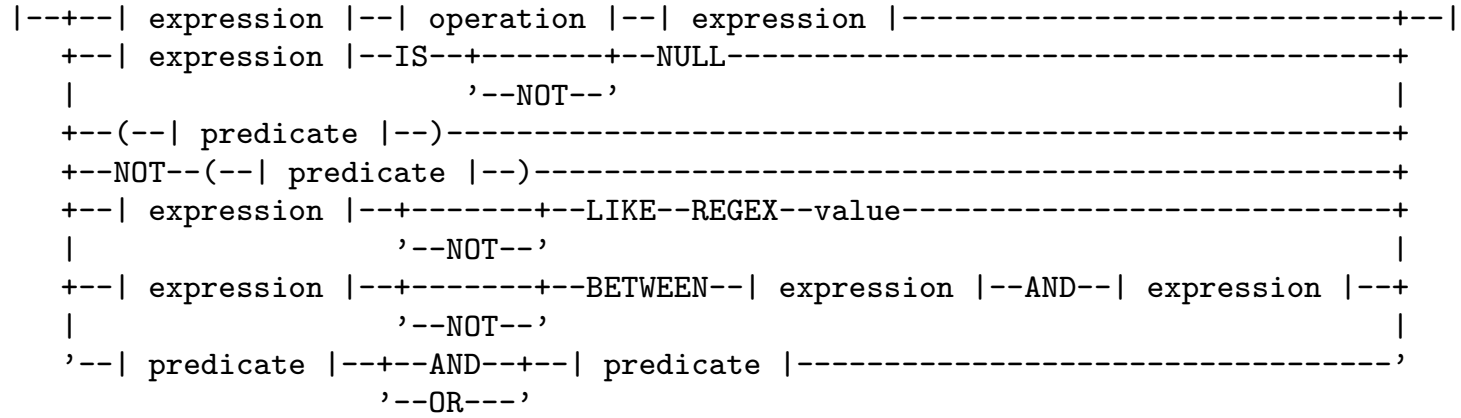
Unterstützte SQL-Anweisungen (5)



Unterstützte SQL-Anweisungen (6)

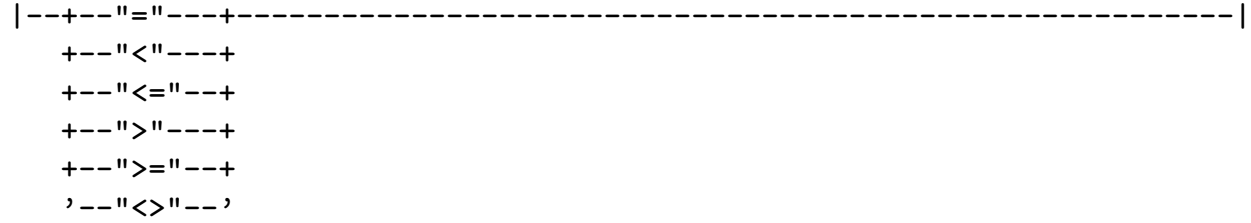
where-clause:

predicate:

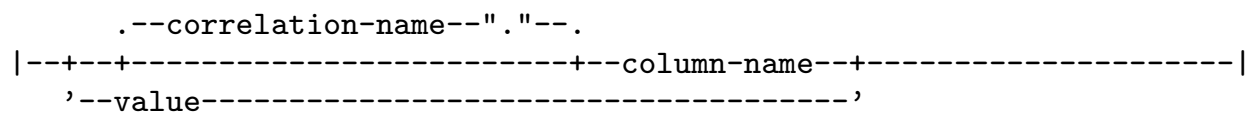


Unterstützte SQL-Anweisungen (7)

operation:



expression:



Optimizer

- Optimiert den Anfragebaum des Compilers
- Momentan nur für SELECT und DELETE
- Einfache regelbasierte Optimierungsstrategien
  - Entfernen von Negationen
  - Normalisierung der WHERE-Klausel
  - Verwenden eines Indexes, wenn möglich und sinnvoll
  - Selektionen wenn möglich vor Joins ausführen
  - Effiziente Anordnung der Tabellen beim Join
  - Unnötige Verschachtelungen eliminieren
- keine kostenbasierte Optimierungen

Optimizer

- Optimiert den Anfragebaum des Compilers
- Momentan nur für SELECT und DELETE
- Einfache regelbasierte Optimierungsstrategien
  - Entfernen von Negationen
  - Normalisierung der WHERE-Klausel
  - Verwenden eines Indexes, wenn möglich und sinnvoll
  - Selektionen wenn möglich vor Joins ausführen
  - Effiziente Anordnung der Tabellen beim Join
  - Unnötige Verschachtelungen eliminieren
- keine kostenbasierte Optimierungen

Optimizer

- Optimiert den Anfragebaum des Compilers
- Momentan nur für SELECT und DELETE
- Einfache regelbasierte Optimierungsstrategien
  - Entfernen von Negationen
  - Normalisierung der WHERE-Klausel
  - Verwenden eines Indexes, wenn möglich und sinnvoll
  - Selektionen wenn möglich vor Joins ausführen
  - Effiziente Anordnung der Tabellen beim Join
  - Unnötige Verschachtelungen eliminieren
- keine kostenbasierte Optimierungen

Optimizer

- Optimiert den Anfragebaum des Compilers
- Momentan nur für SELECT und DELETE
- Einfache regelbasierte Optimierungsstrategien
  - Entfernen von Negationen
  - Normalisierung der WHERE-Klausel
  - Verwenden eines Indexes, wenn möglich und sinnvoll
  - Selektionen wenn möglich vor Joins ausführen
  - Effiziente Anordnung der Tabellen beim Join
  - Unnötige Verschachtelungen eliminieren
- keine kostenbasierte Optimierungen

Optimizer

- Optimiert den Anfragebaum des Compilers
- Momentan nur für SELECT und DELETE
- Einfache regelbasierte Optimierungsstrategien
  - Entfernen von Negationen
  - Normalisierung der WHERE-Klausel
  - Verwenden eines Indexes, wenn möglich und sinnvoll
  - Selektionen wenn möglich vor Joins ausführen
  - Effiziente Anordnung der Tabellen beim Join
  - Unnötige Verschachtelungen eliminieren
- keine kostenbasierte Optimierungen

RunTime

- Abarbeitung des optimierten Syntaxbaumes (Anfrageplan)
- Realisiert die gewünschten Operationen (Insert, Select, ...) mit Hilfe der übrigen Komponenten und liefert Ergebnismenge zurück
- Verwendung eines Iteratorenkonzepts
- Join als Nested-Loop-Join implementiert

RunTime

- Abarbeitung des optimierten Syntaxbaumes (Anfrageplan)
- Realisiert die gewünschten Operationen (Insert, Select, ...) mit Hilfe der übrigen Komponenten und liefert Ergebnismenge zurück
- Verwendung eines Iteratorenkonzepts
- Join als Nested-Loop-Join implementiert

Erfahrungen bei der Durchführung des Praktikums

- Einarbeitungszeit ca. 2 Wochen
- Probleme mit C++ ...
- Coding Standards wurden nicht immer ernst genommen
- Zum Glück wenige Abbrecher
- Testen ist wichtig!
- Gute Versionsverwaltung ist Pflicht!

Erfahrungen bei der Durchführung des Praktikums

- Einarbeitungszeit ca. 2 Wochen
- Probleme mit C++ ...
- Coding Standards wurden nicht immer ernst genommen
- Zum Glück wenige Abbrecher
- Testen ist wichtig!
- Gute Versionsverwaltung ist Pflicht!

Erfahrungen bei der Durchführung des Praktikums

- Einarbeitungszeit ca. 2 Wochen
- Probleme mit C++ ...
- Coding Standards wurden nicht immer ernst genommen
- Zum Glück wenige Abbrecher
- Testen ist wichtig!
- Gute Versionsverwaltung ist Pflicht!

Ende

Gibt es noch Fragen?

Vielen Dank für Ihre Aufmerksamkeit

Systemstart

```
[dbjtest]$
```

Systemstart

```
[dbjtest]$ ls
[dbjtest]$
```

Systemstart

```
[dbjtest]$ ls
[dbjtest]$ ipcs
```

Systemstart

```
[dbjtest]$ ls
[dbjtest]$ ipcs

------ Gemeinsamer Speicher: Segmente -----
Schlüssel shmid      Besitzer   Rechte     Bytes      nattch     Status
0x00000000 2686976    root      644        151552     4          dest
0x00000000 2719745    root      644        122880     4          dest

------ Semaphorenfelder -----
Schlüssel SemID      Besitzer   Rechte     nsems

------ Nachrichtenwarteschlangen -----
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$
```

Systemstart

```
[dbjtest]$ ls
[dbjtest]$ ipcs

------ Gemeinsamer Speicher: Segmente ------
Schlüssel shmid      Besitzer   Rechte     Bytes      nattch     Status
0x00000000 2686976    root      644        151552     4          dest
0x00000000 2719745    root      644        122880     4          dest

------ Semaphorenfelder -----
Schlüssel SemID      Besitzer   Rechte     nsems

------ Nachrichtenwarteschlangen -----
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$
```

Systemstart

```
[dbjtest]$ ls
[dbjtest]$ ipcs

------ Gemeinsamer Speicher: Segmente -----
Schlüssel shmid      Besitzer   Rechte     Bytes      nattch     Status
0x00000000 2686976    root      644        151552     4          dest
0x00000000 2719745    root      644        122880     4          dest

------ Semaphorenfelder -----
Schlüssel SemID      Besitzer   Rechte     nsems

------ Nachrichtenwarteschlangen -----
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ ls
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$
```

Systemstart

```
[dbjtest]$ ls
[dbjtest]$ ipcs

------ Gemeinsamer Speicher: Segmente -----
Schlüssel shmid      Besitzer   Rechte     Bytes      nattch     Status
0x00000000 2686976    root      644        151552     4          dest
0x00000000 2719745    root      644        122880     4          dest

------ Semaphorenfelder -----
Schlüssel SemID      Besitzer   Rechte     nsems

------ Nachrichtenwarteschlangen -----
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ ls
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$ ipcs
```

Systemstart

```
[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ ls
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$ ipcs

------ Gemeinsamer Speicher: Segmente ------
Schlüssel shmid      Besitzer   Rechte     Bytes      nattch     Status
0x00000000 2686976    root      644        151552     4          dest
0x00000000 2719745    root      644        122880     4          dest
0x02010341 3506178    knoppix   660        204834     0
0x01010341 3538947    knoppix   660        4096034    0

------ Semaphorenfelder ------
Schlüssel SemID      Besitzer   Rechte     nsems
0x00000000 327680     knoppix   660        3
0x00000000 360449     knoppix   660        3

------ Nachrichtenwarteschlangen ------
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$
```

Systemstart

```
[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ ls
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$ ipcs

------ Gemeinsamer Speicher: Segmente ------
Schlüssel shmid      Besitzer   Rechte     Bytes      nattch     Status
0x00000000 2686976    root      644        151552     4          dest
0x00000000 2719745    root      644        122880     4          dest
0x02010341 3506178    knoppix   660        204834     0
0x01010341 3538947    knoppix   660        4096034    0

------ Semaphorenfelder -----
Schlüssel SemID      Besitzer   Rechte     nsems
0x00000000 327680     knoppix   660        3
0x00000000 360449     knoppix   660        3

------ Nachrichtenwarteschlangen -----
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$ showPageContent 1 1
```

Systemstart

```
----- Nachrichtenwarteschlangen -----
Schlüssel msqid      Besitzer   Rechte     used-bytes   messages

[dbjtest]$ showPageContent 1 1
+-------------------------------------------------------------------------+
|                                                                         |
| +----------------------- Global Page Header -----------------------+ |
| | Page number:  1                                                   | |
| | Type of Page: 0                                                   | |
| +-------------------------------------------------------------------+ |
| +------------------------ Slot Page Header ------------------------+ |
| | Number of slots:            3                                     | |
| | Total free space:         3913 (Bytes)                            | |
| | Largest free block:       3909 (Bytes)                            | |
| +-------------------------------------------------------------------+ |
| +--------------------------- Record List ---------------------------+ |
| | Slot:   0 -- Offset: 4045 -- Record length:    51 (Bytes)         | |
| | Slot:   1 -- Offset: 3993 -- Record length:    52 (Bytes)         | |
| | Slot:   2 -- Offset: 3941 -- Record length:    52 (Bytes)         | |
| +-------------------------------------------------------------------+ |
|                                                                         |
+-------------------------------------------------------------------------+
[dbjtest]$
```

Systemstart

```
[dbjtest]$ showPageContent 1 1
+-------------------------------------------------------------------------+
|                                                                         |
| +----------------------- Global Page Header -------------------------+ |
| | Page number:  1                                                     | |
| | Type of Page: 0                                                     | |
| +---------------------------------------------------------------------+ |
| +------------------------- Slot Page Header --------------------------+ |
| | Number of slots:            3                                       | |
| | Total free space:         3913 (Bytes)                              | |
| | Largest free block:       3909 (Bytes)                              | |
| +---------------------------------------------------------------------+ |
| +--------------------------- Record List ----------------------------+ |
| | Slot:   0 -- Offset: 4045 -- Record length:    51 (Bytes)           | |
| | Slot:   1 -- Offset: 3993 -- Record length:    52 (Bytes)           | |
| | Slot:   2 -- Offset: 3941 -- Record length:    52 (Bytes)           | |
| +---------------------------------------------------------------------+ |
|                                                                         |
+-------------------------------------------------------------------------+
[dbjtest]$ ls
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$
```

Systemstart

```
[dbjtest]$ showPageContent 1 1
+-------------------------------------------------------------------------+
|                                                                         |
| +----------------------- Global Page Header -------------------------+ |
| | Page number:  1                                                     | |
| | Type of Page: 0                                                     | |
| +---------------------------------------------------------------------+ |
| +------------------------- Slot Page Header --------------------------+ |
| | Number of slots:            3                                       | |
| | Total free space:         3913 (Bytes)                              | |
| | Largest free block:       3909 (Bytes)                              | |
| +---------------------------------------------------------------------+ |
| +--------------------------- Record List ----------------------------+ |
| | Slot:   0 -- Offset: 4045 -- Record length:    51 (Bytes)           | |
| | Slot:   1 -- Offset: 3993 -- Record length:    52 (Bytes)           | |
| | Slot:   2 -- Offset: 3941 -- Record length:    52 (Bytes)           | |
| +---------------------------------------------------------------------+ |
|                                                                         |
+-------------------------------------------------------------------------+
[dbjtest]$ ls
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$ showPageContent 1 0
```

Systemstart

```
Seg1.dbj  Seg32768.dbj  Seg32770.dbj  Seg32772.dbj  Seg32774.dbj
Seg2.dbj  Seg32769.dbj  Seg32771.dbj  Seg32773.dbj  Seg3.dbj
[dbjtest]$ showPageContent 1 0
+-------------------------------------------------------------------------+
|                                                                         |
| +----------------------- Global Page Header -------------------------+ |
| | Page number:  0                                                     | |
| | Type of Page: 3                                                     | |
| +---------------------------------------------------------------------+ |
| +------------------------ FSI Page Header --------------------------+ |
| | countEntries: 1                                                     | |
| +---------------------------------------------------------------------+ |
| +---------------------------- FSI Table ------------------------------+ |
| | Page  |  FS  | LFB  |  | Page  |  FS  | LFB  |  | Page  |  FS  | LFB  | |
| |-------|------|------|  |-------|------|------|  |-------|------|------| |
| |     1 |  244 |  244 |  |       |      |      |  |       |      |      | |
| +---------------------------------------------------------------------+ |
| |                  --- no more FSI entries found ---                  | |
| +---------------------------------------------------------------------+ |
| | FS = free space -- LFB largest free block -- (in  16 Bytes)         | |
| +---------------------------------------------------------------------+ |
|                                                                         |
+-------------------------------------------------------------------------+
[dbjtest]$
```

Systemtabellen

```
[dbjtest]$
```

Systemtabellen

```
[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$
```

Systemtabellen

```
[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ dbj
```

Systemtabellen

```
[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ dbj

System J

(c) 2004-2005, Lehrstuhl fuer Datenbanken und Informationssysteme

Command Line
============
- Alle Anweisungen muessen mit einem Semikolon ';' gefolgt vom Zeilenende
  abgeschlossen werden
- Verlassen der Kommandzeile mit 'quit' oder 'exit'


Die Syntax fuer unterstuetzte SQL Anweisungen kann unter folgender Adresse
gefunden werden:

        http://iibm08.inf.uni-jena.de/~mgr/dbj/class_dbj_compiler.html

dbj =>
```

Systemtabellen

```
[dbjtest]$ dbjstart
The system was started successfully. SQLSTATE=00000

[dbjtest]$ dbj

System J

(c) 2004-2005, Lehrstuhl fuer Datenbanken und Informationssysteme

Command Line
============
- Alle Anweisungen muessen mit einem Semikolon ';' gefolgt vom Zeilenende
  abgeschlossen werden
- Verlassen der Kommandzeile mit 'quit' oder 'exit'


Die Syntax fuer unterstuetzte SQL Anweisungen kann unter folgender Adresse
gefunden werden:

        http://iibm08.inf.uni-jena.de/~mgr/dbj/class_dbj_compiler.html

dbj => select * from systables;
```

Systemtabellen

```
Command Line
============
- Alle Anweisungen muessen mit einem Semikolon ';' gefolgt vom Zeilenende
  abgeschlossen werden
- Verlassen der Kommandzeile mit 'quit' oder 'exit'


Die Syntax fuer unterstuetzte SQL Anweisungen kann unter folgender Adresse
gefunden werden:

        http://iibm08.inf.uni-jena.de/~mgr/dbj/class_dbj_compiler.html

dbj => select * from systables;

TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------ ---------- ------------ ------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 13:45:24.839072           3
SYSCOLUMNS
                                       2            6 2005-02-28 13:45:24.841806          18
SYSINDEXES
                                       3            7 2005-02-28 13:45:24.843554           7

  3 record(s) returned.

dbj =>
```

Systemtabellen

```
Command Line
============
- Alle Anweisungen muessen mit einem Semikolon ';' gefolgt vom Zeilenende
  abgeschlossen werden
- Verlassen der Kommandzeile mit 'quit' oder 'exit'


Die Syntax fuer unterstuetzte SQL Anweisungen kann unter folgender Adresse
gefunden werden:

        http://iibm08.inf.uni-jena.de/~mgr/dbj/class_dbj_compiler.html

dbj => select * from systables;

TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
-------------------------------------------------------------------------------------------------
------------------------------ ---------- ------------ ------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 13:45:24.839072           3
SYSCOLUMNS
                                       2            6 2005-02-28 13:45:24.841806          18
SYSINDEXES
                                       3            7 2005-02-28 13:45:24.843554           7

  3 record(s) returned.

dbj => select table_id, index_name, index_type from sysindexes;
```

Systemtabellen

```
                                        3           7 2005-02-28 13:45:24.843554          7

  3 record(s) returned.

dbj => select table_id, index_name, index_type from sysindexes;

TABLE_ID    INDEX_NAME
                                                INDEX_TYPE
----------- ------------------------------------------------------------------------------
------------------------------------------ ----------
          1 IDX_SYSTABLES_TABLEID_ID
                                                BTREE
          1 IDX_SYSTABLES_TABLENAME_ID
                                                BTREE
          2 IDX_SYSCOLUMNS_TABLEID_ID
                                                BTREE
          2 IDX_SYSCOLUMNS_COLUMNNAME_ID
                                                BTREE
          3 IDX_SYSINDEXES_TABLEID_ID
                                                BTREE
          3 IDX_SYSINDEXES_INDEXNAME_ID
                                                BTREE
          3 IDX_SYSINDEXES_INDEXID_ID
                                                BTREE

  7 record(s) returned.

dbj =>
```

CREATE TABLE

```
dbj => 
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => select * from systables;
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => select * from systables;

TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------ ----------- ------------ -------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 14:31:25.743915           4
SYSCOLUMNS
                                       2            6 2005-02-28 14:31:25.746265          20
SYSINDEXES
                                       3            7 2005-02-28 14:31:25.748085           7
TEST
                                       4            2 2005-02-28 14:39:51.261357           0

  4 record(s) returned.

dbj =>
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => select * from systables;

TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------ ---------- ------------ -------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 14:31:25.743915           4
SYSCOLUMNS
                                       2            6 2005-02-28 14:31:25.746265          20
SYSINDEXES
                                       3            7 2005-02-28 14:31:25.748085           7
TEST
                                       4            2 2005-02-28 14:39:51.261357           0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => select * from systables;

TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------ ---------- ------------ -------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 14:31:25.743915           4
SYSCOLUMNS
                                       2            6 2005-02-28 14:31:25.746265          20
SYSINDEXES
                                       3            7 2005-02-28 14:31:25.748085           7
TEST
                                       4            2 2005-02-28 14:39:51.261357           0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => select * from systables;

TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------ ----------- ------------ -------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 14:31:25.743915           4
SYSCOLUMNS
                                       2            6 2005-02-28 14:31:25.746265          20
SYSINDEXES
                                       3            7 2005-02-28 14:31:25.748085           7
TEST
                                       4            2 2005-02-28 14:39:51.261357           0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;
```

CREATE TABLE

```
TABLE_NAME
                                 TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------ ----------- ------------ -------------------------- -----------
SYSTABLES
                                        1            5 2005-02-28 14:31:25.743915           4
SYSCOLUMNS
                                        2            6 2005-02-28 14:31:25.746265          20
SYSINDEXES
                                        3            7 2005-02-28 14:31:25.748085           7
TEST
                                        4            2 2005-02-28 14:39:51.261357           0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj =>
```

CREATE TABLE

```
TABLE_NAME
                                TABLE_ID    COLUMN_COUNT CREATE_TIME                TUPLE_COUNT
------------------------------------------------------------------------------------------------
------------------------------- ----------- ------------ -------------------------- -----------
SYSTABLES
                                       1            5 2005-02-28 14:31:25.743915           4
SYSCOLUMNS
                                       2            6 2005-02-28 14:31:25.746265          20
SYSINDEXES
                                       3            7 2005-02-28 14:31:25.748085           7
TEST
                                       4            2 2005-02-28 14:39:51.261357           0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
```

CREATE TABLE

```
SYSTABLES
                                        1          5 2005-02-28 14:31:25.743915          4
SYSCOLUMNS
                                        2          6 2005-02-28 14:31:25.746265         20
SYSINDEXES
                                        3          7 2005-02-28 14:31:25.748085          7
TEST
                                        4          2 2005-02-28 14:39:51.261357          0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj =>
```

CREATE TABLE

```
SYSTABLES
                                1          5 2005-02-28 14:31:25.743915          4
SYSCOLUMNS
                                2          6 2005-02-28 14:31:25.746265         20
SYSINDEXES
                                3          7 2005-02-28 14:31:25.748085          7
TEST
                                4          2 2005-02-28 14:39:51.261357          0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
```

CREATE TABLE

```
SYSINDEXES
                                        3           7 2005-02-28 14:31:25.748085          7
TEST
                                        4           2 2005-02-28 14:39:51.261357          0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj =>
```

CREATE TABLE

```
                                        4           2 2005-02-28 14:39:51.261357          0

  4 record(s) returned.

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
  1 record(s) returned.

dbj => insert into test values (2, 'abcdefghijk');
Cannot set the value 'abcdefghijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 bytes. SQLSTATE=CAT08
Rolling back transaction...

dbj => select * from test;
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (2, 'efgh');
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (2, 'efgh');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd
          2 efgh

  2 record(s) returned.

dbj =>
```

CREATE TABLE

```
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (2, 'efgh');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd
          2 efgh

  2 record(s) returned.

dbj => rollback;
```

CREATE TABLE

```
dbj => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (1, 'abcd');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (2, 'efgh');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd
          2 efgh

  2 record(s) returned.

dbj => rollback;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

CREATE TABLE

```
dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into test values (2, 'efgh');
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd
          2 efgh

  2 record(s) returned.

dbj => rollback;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj =>
```

CREATE TABLE

```
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd
          2 efgh

  2 record(s) returned.

dbj => rollback;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (NULL, 'xyz');
Cannot set the attribute 0 in a record to NULL because the definition of the table the record be
longs to forbids NULLs in that column. SQLSTATE=CAT07
Rolling back transaction...

dbj =>
```

CREATE TABLE

```
NUM         STR
----------- -----
          1 abcd
          2 efgh

  2 record(s) returned.

dbj => rollback;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from test;

NUM         STR
----------- -----
          1 abcd

  1 record(s) returned.

dbj => insert into test values (NULL, 'xyz');
Cannot set the attribute 0 in a record to NULL because the definition of the table the record be
longs to forbids NULLs in that column. SQLSTATE=CAT07
Rolling back transaction...

dbj => insert into test values (2, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => 
```

INSERT/SELECT

```
dbj => CREATE TABLE angest (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   vorname  VARCHAR(10),
dbj =>   nachname VARCHAR(20) NOT NULL,
dbj =>   gehalt   INTEGER NOT NULL,
dbj =>   adresse  VARCHAR(20),
dbj =>
dbj =>   PRIMARY KEY(persNr));
```

INSERT/SELECT

```
dbj => CREATE TABLE angest (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   vorname  VARCHAR(10),
dbj =>   nachname VARCHAR(20) NOT NULL,
dbj =>   gehalt   INTEGER NOT NULL,
dbj =>   adresse  VARCHAR(20),
dbj =>
dbj =>   PRIMARY KEY(persNr));
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE angest (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   vorname  VARCHAR(10),
dbj =>   nachname VARCHAR(20) NOT NULL,
dbj =>   gehalt   INTEGER NOT NULL,
dbj =>   adresse  VARCHAR(20),
dbj =>
dbj =>   PRIMARY KEY(persNr));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX angest_nachname ON angest (nachname);
```

INSERT/SELECT

```
dbj => CREATE TABLE angest (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   vorname  VARCHAR(10),
dbj =>   nachname VARCHAR(20) NOT NULL,
dbj =>   gehalt   INTEGER NOT NULL,
dbj =>   adresse  VARCHAR(20),
dbj =>
dbj =>   PRIMARY KEY(persNr));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX angest_nachname ON angest (nachname);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE angest (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   vorname  VARCHAR(10),
dbj =>   nachname VARCHAR(20) NOT NULL,
dbj =>   gehalt   INTEGER NOT NULL,
dbj =>   adresse  VARCHAR(20),
dbj =>
dbj =>   PRIMARY KEY(persNr));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX angest_nachname ON angest (nachname);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from sysindexes where index_name = 'angest_nachname';
```

INSERT/SELECT

```
dbj =>
dbj =>   PRIMARY KEY(persNr));
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX angest_nachname ON angest (nachname);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from sysindexes where index_name = 'angest_nachname';

TABLE_ID    INDEX_NAME
                                                INDEX_ID    INDEX_TYPE COLUMN_ID   IS_UNIQUE CREATE
_TIME
----------- ------------------------------------------------------------------------------------
---------------------------------------- ----------- ---------- ----------- --------- ------
--------------------
          5 ANGEST_NACHNAME
                                                       9 BTREE                2 N         2005-0
2-28 14:51:33.141130

  1 record(s) returned.

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbj =>   projNr          INTEGER NOT NULL,
dbj =>   name            VARCHAR(20) NOT NULL,
dbj =>   prioritaet      INTEGER,
dbj =>   beschreibung    VARCHAR(1000),
dbj =>
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
```

INSERT/SELECT

```
dbj =>   PRIMARY KEY (projNr));
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
dbj => insert into angest values (1,'Klaus','Kuespert', 10000, NULL),
dbj =>                           (2,'Knut','Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbj =>                           (3,'Thomas','Mueller',3000, 'Ernst-Abbe-Platz 2'),
dbj =>                           (4,'Hannes','Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj =>
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE TABLE mitarbeit (
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
```

INSERT/SELECT

```
dbj =>   persNr   INTEGER NOT NULL,
dbj =>   projNr   INTEGER NOT NULL,
dbj =>   percent  INTEGER);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
dbj =>                              (2,1,90),
dbj =>                              (3,1,0),(3,3,30),
dbj =>                              (4,1,100);
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
dbj =>                               (2,1,90),
dbj =>                               (3,1,0),(3,3,30),
dbj =>                               (4,1,100);
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
dbj =>                              (2,1,90),
dbj =>                              (3,1,0),(3,3,30),
dbj =>                              (4,1,100);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj =>
```

INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000

dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dbj =>                            (2, 'DBS1', 5, NULL),
dbj =>                            (3, 'XML-Seminar', 99, NULL),
dbj =>                            (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...

dbj => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
dbj =>                              (2,1,90),
dbj =>                              (3,1,0),(3,3,30),
dbj =>                              (4,1,100);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from angest;
```

INSERT/SELECT

```
                                    (3, 'XML-Seminar', 99, NULL),
                                    (4, 'DBS2', 6, NULL);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
dbj =>                              (2,1,90),
dbj =>                              (3,1,0),(3,3,30),
dbj =>                              (4,1,100);
The operation was completed successfully. SQLSTATE=00000

dbj => commit;
The operation was completed successfully. SQLSTATE=00000

dbj => select * from angest;

PERSNR      VORNAME    NACHNAME             GEHALT      ADRESSE
----------- ---------- -------------------- ----------- --------------------
          1 Klaus      Kuespert                   10000 -
          2 Knut       Stolze                      1500 Ernst-Abbe-Platz 2
          3 Thomas     Mueller                     3000 Ernst-Abbe-Platz 2
          4 Hannes     Moser                        300 Ernst-Abbe-Platz 1

  4 record(s) returned.

dbj =>
```

INSERT/SELECT

```
dbj => 
```

INSERT/SELECT

```
dbj => select projNr, name, prioritaet from projekt;
```

INSERT/SELECT

```
dbj => select projNr, name, prioritaet from projekt;

PROJNR      NAME                 PRIORITAET
----------- -------------------- -----------
          1 DBS-Entwicklung                1
          2 DBS1                           5
          3 XML-Seminar                   99
          4 DBS2                           6

  4 record(s) returned.

dbj =>
```

INSERT/SELECT

```
dbj => select projNr, name, prioritaet from projekt;

PROJNR      NAME                 PRIORITAET
----------- -------------------- -----------
          1 DBS-Entwicklung                1
          2 DBS1                           5
          3 XML-Seminar                   99
          4 DBS2                           6

  4 record(s) returned.

dbj => select * from mitarbeit;
```

INSERT/SELECT

```
dbj => select projNr, name, prioritaet from projekt;

PROJNR      NAME                 PRIORITAET
----------- -------------------- -----------
          1 DBS-Entwicklung                1
          2 DBS1                           5
          3 XML-Seminar                   99
          4 DBS2                           6

  4 record(s) returned.

dbj => select * from mitarbeit;

PERSNR      PROJNR      PERCENT
----------- ----------- -----------
          1           1          10
          1           2          30
          1           3           2
          2           1          90
          3           1           0
          3           3          30
          4           1         100

  7 record(s) returned.

dbj =>
```

INSERT/SELECT

```
PROJNR      NAME                 PRIORITAET
----------- -------------------- ----------
          1 DBS-Entwicklung               1
          2 DBS1                          5
          3 XML-Seminar                  99
          4 DBS2                          6

  4 record(s) returned.

dbj => select * from mitarbeit;

PERSNR      PROJNR      PERCENT
----------- ----------- -----------
          1           1          10
          1           2          30
          1           3           2
          2           1          90
          3           1           0
          3           3          30
          4           1         100

  7 record(s) returned.

dbj => select a.persnr AS angest_nr, a.vorname, a.nachname, p.projnr as proj_nr,
dbj =>        p.name as proj_name, m.percent
dbj => from angest AS a, mitarbeit as m, projekt as p
dbj => where a.persnr = m.persnr AND
dbj =>       p.projnr = m.projnr;
```

INSERT/SELECT

```
          1          2         30
          1          3          2
          2          1         90
          3          1          0
          3          3         30
          4          1        100

  7 record(s) returned.

dbj => select a.persnr AS angest_nr, a.vorname, a.nachname, p.projnr as proj_nr,
dbj =>        p.name as proj_name, m.percent
dbj => from angest AS a, mitarbeit as m, projekt as p
dbj => where a.persnr = m.persnr AND
dbj =>       p.projnr = m.projnr;

PERSNR      VORNAME    NACHNAME             PROJNR      NAME                 PERCENT
---------- ---------- -------------------- ----------- -------------------- -----------
          1 Klaus      Kuespert                     1 DBS-Entwicklung              10
          1 Klaus      Kuespert                     2 DBS1                         30
          1 Klaus      Kuespert                     3 XML-Seminar                   2
          2 Knut       Stolze                       1 DBS-Entwicklung              90
          3 Thomas     Mueller                      1 DBS-Entwicklung               0
          3 Thomas     Mueller                      3 XML-Seminar                  30
          4 Hannes     Moser                        1 DBS-Entwicklung             100

  7 record(s) returned.

dbj =>
```

Ausführungsplan

```
dbj => 
```

Ausführungsplan

```
dbj => select * from angest;
```

Ausführungsplan

```
dbj => select * from angest;
==========================================================================
Optimized access plan
--------------------------------------------------------------------------
SelectStmt
  |
Projections - Column ('PERSNR'/0) [Correlation name: ANGEST] - Column ('VORNAME'/1) [Correlation
 name: ANGEST] - Column ('NACHNAME'/2) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correla
tion name: ANGEST] - Column ('ADRESSE'/4) [Correlation name: ANGEST]
  |
Sources - Table ('ANGEST'/5)
==========================================================================

PERSNR      VORNAME    NACHNAME             GEHALT      ADRESSE
----------- ---------- -------------------- ----------- --------------------
          1 Klaus      Kuespert                   10000 -
          2 Knut       Stolze                      1500 Ernst-Abbe-Platz 2
          3 Thomas     Mueller                     3000 Ernst-Abbe-Platz 2
          4 Hannes     Moser                        300 Ernst-Abbe-Platz 1

  4 record(s) returned.

dbj =>
```

Ausführungsplan

```
dbj => select * from angest;
=============================================================================
Optimized access plan
-----------------------------------------------------------------------------
SelectStmt
  |
Projections - Column ('PERSNR'/0) [Correlation name: ANGEST] - Column ('VORNAME'/1) [Correlation
 name: ANGEST] - Column ('NACHNAME'/2) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correla
tion name: ANGEST] - Column ('ADRESSE'/4) [Correlation name: ANGEST]
  |
Sources - Table ('ANGEST'/5)
=============================================================================

PERSNR      VORNAME    NACHNAME             GEHALT      ADRESSE
---------- ---------- -------------------- ---------- --------------------
         1 Klaus      Kuespert                  10000 -
         2 Knut       Stolze                     1500 Ernst-Abbe-Platz 2
         3 Thomas     Mueller                    3000 Ernst-Abbe-Platz 2
         4 Hannes     Moser                       300 Ernst-Abbe-Platz 1

  4 record(s) returned.

dbj => select * from angest where nachname = 'Moser';
```

Ausführungsplan

```
          2 Knut       Stolze                    1500 Ernst-Abbe-Platz 2
          3 Thomas     Mueller                   3000 Ernst-Abbe-Platz 2
          4 Hannes     Moser                      300 Ernst-Abbe-Platz 1

  4 record(s) returned.

dbj => select * from angest where nachname = 'Moser';
========================================================================
Optimized access plan
------------------------------------------------------------------------
SelectStmt
  |
Projections - Column ('PERSNR'/0) [Correlation name: ANGEST] - Column ('VORNAME'/1) [Correlation
 name: ANGEST] - Column ('NACHNAME'/2) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correla
tion name: ANGEST] - Column ('ADRESSE'/4) [Correlation name: ANGEST]
  |
Sources - Table ('ANGEST'/5)
            |
          Index ('ANGEST_NACHNAME'/9) [(VARCHAR) 'Moser'..'Moser']
========================================================================

PERSNR      VORNAME    NACHNAME             GEHALT      ADRESSE
----------- ---------- -------------------- ----------- --------------------
          4 Hannes     Moser                        300 Ernst-Abbe-Platz 1

  1 record(s) returned.

dbj =>
```

Ausführungsplan

```
  4 record(s) returned.

dbj => select * from angest where nachname = 'Moser';
=============================================================================
Optimized access plan
-----------------------------------------------------------------------------
SelectStmt
  |
Projections - Column ('PERSNR'/0) [Correlation name: ANGEST] - Column ('VORNAME'/1) [Correlation
 name: ANGEST] - Column ('NACHNAME'/2) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correla
tion name: ANGEST] - Column ('ADRESSE'/4) [Correlation name: ANGEST]
  |
Sources - Table ('ANGEST'/5)
            |
          Index ('ANGEST_NACHNAME'/9) [(VARCHAR) 'Moser'..'Moser']
=============================================================================

PERSNR      VORNAME    NACHNAME             GEHALT      ADRESSE
----------- ---------- -------------------- ----------- --------------------
          4 Hannes     Moser                        300 Ernst-Abbe-Platz 1

  1 record(s) returned.

dbj => SELECT a.persnr AS angest_nr, a.vorname, a.nachname, p.projnr AS proj_nr,
dbj =>        p.name AS proj_name, m.percent
dbj => FROM angest AS a, mitarbeit AS m, projekt AS p
dbj => WHERE a.persnr = m.persnr AND
dbj =>       p.projnr = m.projnr;
```

Ausführungsplan

```
R'/0) [Correlation name: p] [New column name: proj_nr] - Column ('NAME'/1) [Correlation name: p]
 [New column name: proj_name] - Column ('PERCENT'/2) [Correlation name: m]
  |
Sources - Table ('ANGEST'/5) [Correlation name: a] - Table ('PROJEKT'/6) [Correlation name: p] -
 Table ('MITARBEIT'/7) [Correlation name: m]
  |
WhereClause - Predicate - LogicalOp ('AND') - Predicate
                |                               |
                |                              Column ('PROJNR'/0) [Correlation name: p] - Compar
ison ('=') - Column ('PROJNR'/1) [Correlation name: m]
                |
              Column ('PERSNR'/0) [Correlation name: a] - Comparison ('=') - Column ('PERSNR'/0)
 [Correlation name: m]
==========================================================================

PERSNR      VORNAME    NACHNAME             PROJNR      NAME                 PERCENT
---------- ---------- -------------------- ---------- -------------------- ----------
         1 Klaus      Kuespert                      1 DBS-Entwicklung              10
         1 Klaus      Kuespert                      2 DBS1                         30
         1 Klaus      Kuespert                      3 XML-Seminar                   2
         2 Knut       Stolze                        1 DBS-Entwicklung              90
         3 Thomas     Mueller                       1 DBS-Entwicklung               0
         3 Thomas     Mueller                       3 XML-Seminar                  30
         4 Hannes     Moser                         1 DBS-Entwicklung             100

  7 record(s) returned.

dbj => 
```