#### SYSTEM J

Hannes Moser

Einführung

Architektur

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

Erfahrungen

# 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

## Einführung

Die Architekur des System J

Die Komponenten des System J

FileManager

BufferManager

RecordManager

IndexManager

LockManager

CatalogManager

Compiler

Optimizer

RunTime

# Was ist System J?

## Einführung

## Architektur

#### Komponente

FileManager
BufferManager
RecordManager
IndexManager
LockManager
CatalogManage
Compiler
Optimizer
RunTime

- ▶ 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?

## Einführung

## A contract to the

#### Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer RunTime

- ▶ 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

## Einführung

## Architaktı

#### Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

- 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

## Einführung

۸ سواد او دار

Komponent

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer RunTime

- ▶ 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

## Einführung

A mala talah da ka

Komponent

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

- ▶ 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

## Einführung

## Architaktuu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer RunTime

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

## Einführung

Δrchitektı

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer RunTime

- ▶ 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

## Einführung

## Δrchitektu

### Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer

- ▶ 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

## Einführung

## Architaktu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer RunTime

- ▶ 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

## Einführung

Architaktu

Komponent

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer RunTime

- ▶ 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

Einführung

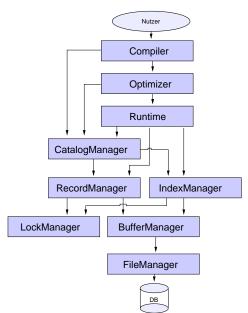
## Architektur

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

Erfahrungen

# Die Architektur des System J



## SYSTEM J

## Hannes Moser

Einführung

Architektur

Komponente

# FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer

Erfahrunger

# FileManager

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

## SYSTEM J

## Hannes Moser

Einführung

Architektu

Komponente

## FileManager

BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

Erfahrungen

# FileManager

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

## SYSTEM I

#### Hannes Moser

BufferManager

# BufferManager

- Verwaltung auf Seitenbasis mittels Hash
- LRU-Algorithmus für Seitenverdrängung
- ► FORCE/NO STEAL (wir haben kein Logging!)

## SYSTEM J

Hannes Moser

Einführung

Architektur

Komponente

BufferManager

BufferManage

IndexManager
LockManager
CatalogManage
Compiler
Optimizer
RunTime

Erfahrungen

# 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

Einführung

Architektu

Komponente

## BufferManager

IndexManager
LockManager
CatalogManag
Compiler
Optimizer
RunTime

- 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!)

## SYSTEM J

#### Hannes Moser

Einführung

Architektu

Komponente

FileManager

BufferManager RecordManager IndexManager

LockManage CatalogMan Compiler Optimizer RunTime

Erfahrungen

# RecordManager

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

# IndexManager

Einführung

Einführung

Komponente

FileManager BufferManager

BufferManager RecordManage

IndexManager

LockManager CatalogManage Compiler

Optimizer RunTime

- ▶ 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

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
- Jeder Index wird auf separatem Segment abgelegt
- ▶ Jeder Knoten des B\*-Baumes wird auf eine Seite
- ▶ B\*-Baum-Verwaltung ist einfach gehalten (keine

#### mannes ivios

Einführung

Komponente

Komponente

FileManager BufferManage

IndexManager

IndexManag

CatalogMan

Compiler Optimizer

Optimizer RunTime

Erfahrungen

# Index Manager

- ► 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)

#### Haimes Wo

Einführung

Komponente

Komponente

BufferManage BosordManage

IndexManager

LockManager CatalogMana

Compiler Optimizer RunTime

Erfahrungen

# Index Manager

- ► 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)

Einführung

⊏imiumrung

Komponente

FileManager BufferManage RecordManage

LockManager CatalogManage Compiler

Optimizer RunTime

- ► 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

Einführung

Lillianiang

Komponente

FileManager BufferManage

BufferManage RecordManage IndexManager

LockManager CatalogManag Compiler

Optimizer RunTime

- ► 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

Einführung

Eimunrung

Komponente

FileManager BufferManager RecordManage IndexManager

LockManager CatalogManager Compiler

Optimizer RunTime

- 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

Einführung

Einführung

Komponente

FileManager BufferManager RecordManage

LockManager CatalogManag Compiler

Optimizer RunTime

- 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
- ► Interpretation der Daten anhand der Metainformationen
- Metainformationen der Systemtabellen sind statisch
- ▶ Beim ersten Systemstart wird automatisch ein Katalog

Einführung

Einführung

Komponent

FileManager BufferManage RecordManager IndexManager LockManager

CatalogManager Compiler

Optimizer RunTime

- 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

Einführung

Einführung

Komponent

FileManager BufferManage RecordManager IndexManager LockManager

CatalogManager Compiler

Optimizer RunTime

- 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



#### Hannes Moser

## CatalogManager

# 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

# 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

Einführung

Architektı

Komponente

FileManager BufferManager RecordManage IndexManager LockManager CatalogManager

Compiler Optimizer

RunTime

- ▶ 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 FROM FROM WHERE WHERE—WHERE
- ▶ Parser mit Bison realisiert

# Compiler

Einführung

Architektu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage

## Compiler

RunTime

- ► 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 FROM

  EBOM EBOM WHERE WHERE WHERE
- ▶ Parser mit Bison realisiert

# Compiler

Einführung

Architektu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage

#### Compiler Optimize

RunTime

- ▶ 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 FROM FROM FROM WHERE WHERE=WHERE
- Parser mit Bison realisiert.

# Compiler

Einführung

Architektu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage

### Compiler Optimizer

RunTime

- ▶ 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 FROM FROM FROM WHERE WHERE=WHERE
- Parser mit Bison realisiert

### Hannes Moser

### Einführung

Architektur

### Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManag Compiler

RunTim

Erfahrunger

# Unterstützte SQL-Anweisungen (1)

### Hannes Moser

### Einführung

Architektur

### Komponente

FileManager BufferManager RecordManager IndexManager LockManager

### Compiler

DunTim

Erfahrungen

# Unterstützte SQL-Anweisungen (2)

### Hannes Moser

### Einführung

Architektur

### Komponente

FileManager BufferManager RecordManage IndexManager LockManager CatalogManager

### Compiler

Optimiz

### Erfahrungen

# Unterstützte SQL-Anweisungen (3)

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

# Unterstützte SQL-Anweisungen (4)

Einführung

Komponenter

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager

### Compiler

Optimize

RullTille

### Hannes Moser

### Einführung

Architektur

### Komponenter

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager

### Compiler

Optimiz

Erfahrunger

# Unterstützte SQL-Anweisungen (5)

### Hannes Moser

### Einführung

Architektur

### Komponente

BufferManager RecordManager IndexManager LockManager

### Compiler

DunTim

Erfahrunger

# Unterstützte SQL-Anweisungen (6)

# where-clause:

### predicate:

### Hannes Moser

### Einführung

Architektur

### Komponente

Komponente

BufferManag

RecordMana

IndexManag

CatalogMan

Compiler

O-ti--i--

RunTim

Erfahrungen

## operation:

```
|--+--"="---+
+--"<"---+
+--">="--+
+--">="--+
+--">="--+
```

Unterstützte SQL-Anweisungen (7)

### expression:

Einführung

Architektur

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

- ▶ 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

Einführung

Architektur

Komponente

FileManager
BufferManager
RecordManager
IndexManager
LockManager
CatalogManage
Compiler
Optimizer
RunTime

- ▶ 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

Einführung

Architektu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer RunTime

- ▶ 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

### Hannes Moser

Optimizer

# 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

Einführung

Architektu

Komponente

BufferManager RecordManager IndexManager LockManager CatalogManag Compiler Optimizer

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

Einführung

Δrchitektu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManage Compiler Optimizer

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

### Hannes Moser

### Einführung

### Architektur

### Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer

### Erfahrungen

# Erfahrungen bei der Durchführung des Praktikums

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

Hannes Moser

Einführung

Architektu

Komponente

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer

Erfahrungen

# 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!

FileManager BufferManager RecordManager IndexManager LockManager CatalogManager Compiler Optimizer

### Erfahrungen

# 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

Vorführung Systemstart

Systemtabellen CREATE TABL INSERT/SELE Ausführungspla [dbjtest]\$

/ C++1

Vorführung Systemstart

Systemtabellen CREATE TABL INSERT/SELE Ausführungspla [dbjtest]\$ ls [dbjtest]\$ █ Hannes Moser

Vorführung Systemstart

Systemtabellen CREATE TABL INSERT/SELE Ausführungspla [dbjtest]\$ ls [dbjtest]\$ ipcs█

/orführung

Systemstart
Systemtabellen
CREATE TABL

```
[db.itest]$ ls
[dbjtest]$ ipcs
---- Gemeinsamer Speicher: Segmente ----
Schlüssel shmid
                     Besitzer
                                Rechte
                                           Bytes
                                                      nattch
                                                                 Status
0x00000000 2686976
                                644
                                           151552
                      root
                                                                 dest
0x00000000 2719745
                      root.
                                644
                                           122880
                                                                 dest.
---- Semaphorenfelder -----
Schlüssel SemID
                     Besitzer
                                Rechte
                                           nsems
---- Nachrichtenwarteschlangen -----
Schlüssel msgid
                     Besitzer
                                Rechte
                                           used-bytes
                                                        messages
[db.itest]$
```

### Hannes Moser

Systemstart

```
Systemstart
```

```
[db.itest]$ ls
[dbjtest]$ ipcs
---- Gemeinsamer Speicher: Segmente ----
Schlüssel shmid
                     Besitzer
                                Rechte
                                           Bytes
                                                       nattch
                                                                  Status
0x00000000 2686976
                                           151552
                      root
                                644
                                                                  dest
0x00000000 2719745
                      root.
                                644
                                           122880
                                                                  dest.
---- Semaphorenfelder -----
Schlüssel SemIII.
                     Besitzen
                                Rechte
                                           nsems
---- Nachrichtenwarteschlangen -----
Schlüssel msgid
                     Besitzer
                                Rechte
                                           used-bytes
                                                         messages
[db.itest]$ db.istart
The system was started successfully. SQLSTATE=00000
[db.itest]$
```

### Systemstart

```
[db.itest]$ ls
[dbjtest]$ ipcs
---- Gemeinsamer Speicher: Segmente ----
Schlüssel shmid
                     Besitzer
                                Rechte
                                          Butes
                                                      nattch
                                                                Status
0x00000000 2686976
                                          151552
                   root
                                644
                                                                dest
0x00000000 2719745
                   root.
                                644
                                          122880
                                                                dest.
---- Semaphorenfelder -----
Schlüssel SemIII.
                    Resitzen
                               Rechte
                                          nsems
---- Nachrichtenwarteschlangen -----
Schlüssel msgid
                    Resitzen
                               Rechte
                                          used-butes
                                                       messages
[db.itest]$ db.istart
The system was started successfully. SQLSTATE=00000
[db.itest]$ ls
Seg1.db.j Seg32768.db.j Seg32770.db.j Seg32772.db.j Seg32774.db.j
Seg2.dbj Seg32769.dbj Seg32771.dbj Seg32773.dbj Seg3.dbj
[db.itest]$
```

### Systemstart

```
[db.itest]$ ls
[dbjtest]$ ipcs
---- Gemeinsamer Speicher: Segmente ----
Schlüssel shmid
                     Besitzer
                                Rechte
                                           Butes
                                                      nattch
                                                                 Status
0x00000000 2686976
                                           151552
                   root
                                644
                                                                 dest
0x00000000 2719745
                   root.
                                644
                                           122880
                                                                 dest.
---- Semaphorenfelder -----
Schlüssel SemIII.
                    Resitzen
                               Rechte
                                           nsems
---- Nachrichtenwarteschlangen -----
Schlüssel msgid
                    Resitzen
                               Rechte
                                           used-butes
                                                        messages
[db.itest]$ db.istart
The system was started successfully. SQLSTATE=00000
[db.itest]$ ls
Seg1.db.j Seg32768.db.j Seg32770.db.j Seg32772.db.j Seg32774.db.j
Seg2.dbj Seg32769.dbj Seg32771.dbj Seg32773.dbj Seg3.dbj
[db.itest]$ ipcs
```

### Systemstart

```
[db.itest]$ db.istart
The system was started successfully. SQLSTATE=00000
[db,jtest]$ ls
Seg1.dbj Seg32768.dbj Seg32770.dbj Seg32772.dbj Seg32774.dbj
Seg2.db.i Seg32769.db.i Seg32771.db.i Seg32773.db.i Seg3.db.i
[db.itest]$ ipcs
---- Gemeinsamer Speicher: Segmente ----
                     Besitzer
Schlüssel shmid
                                Rechte
                                           Bytes
                                                      nattch
                                                                 Status
0x00000000 2686976
                     root
                                644
                                           151552
                                                                 dest.
0x00000000 2719745
                   root
                                644
                                           122880
                                                                 dest.
0x02010341 3506178
                                660
                                           204834
                      knoppix
0x01010341 3538947
                                660
                      knoppix
                                           4096034
---- Semaphorenfelder -----
Schlüssel SemID
                     Besitzen
                                Rechte
                                           nsems
0×00000000 327680
                                660
                                           3
                      knoppix
0x00000000 360449
                      knoppix
                                660
---- Nachrichtenwarteschlangen
Schlüssel msgid
                     Besitzer
                                Rechte
                                           used-butes
                                                        messages
[db.itest]$ ■
```

### Systemstart

CREATE TABL
INSERT/SELE
Ausführungspla

```
[db.itest]$ db.istart
The system was started successfully. SQLSTATE=00000
[db,jtest]$ ls
Seg1.dbj Seg32768.dbj Seg32770.dbj Seg32772.dbj Seg32774.dbj
Seg2.db.i Seg32769.db.i Seg32771.db.i Seg32773.db.i Seg3.db.i
[db.itest]$ ipcs
---- Gemeinsamer Speicher: Segmente ----
                     Besitzer
                                Rechte
Schlüssel shmid
                                           Butes
                                                      nattch
                                                                 Status
0x00000000 2686976
                     root
                                644
                                           151552
                                                                 dest.
0x00000000 2719745
                   root
                                644
                                           122880
                                                                 dest.
0x02010341 3506178
                                660
                                           204834
                     knoppix
0x01010341 3538947
                                660
                     knoppix
                                           4096034
---- Semaphorenfelder -----
Schlüssel SemID
                     Besitzen
                                Rechte
                                           nsems
0×00000000 327680
                                660
                                           3
                     knoppix
0x00000000 360449
                     knoppix
                                660
---- Nachrichtenwarteschlangen
Schlüssel msgid
                     Besitzer
                                Rechte
                                           used-butes
                                                        messages
[db.itest]$ showPageContent 1 1
```

### SYSTEM J

Hannes Moser

### Vorführung Systemstart

Systemtabellen CREATE TABL INSERT/SELE Ausführungspla

# Systemstart

```
---- Nachrichtenwarteschlangen -----
Schlüssel msgid
                 Besitzer
                          Rechte
                                   used-butes
                                              messages
[db.jtest]$ showPageContent 1 1
    ------ Global Page Header
 I Page number: 1
  Type of Page: 0
  ------ Slot Page Header
 I Number of slots:
   Total free space:
                      3913 (Bytes)
   Largest free block: 3909 (Bytes)
                 ----- Record List -----
   Slot: 0 -- Offset: 4045 -- Record length:
                                          51 (Butes)
   Slot: 1 -- Offset: 3993 -- Record length:
                                          52 (Butes)
         2 -- Offset: 3941 -- Record length:
                                          52 (Bytes)
[dbjtest]$
```

### Systemstart

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

[dbjtest]\$ showPageContent 1 0

### Systemstart

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

/orführung

Systemstart Systemsahel

CREATE TABL INSERT/SELE Ausführungspla

```
Seg1.db.j Seg32768.db.j Seg32770.db.j Seg32772.db.j Seg32774.db.j
Seg2.dbj Seg32769.dbj Seg32771.dbj Seg32773.dbj Seg3.dbj
[db.itest]$ showPageContent 1 0
  -----+
 I Page number: 0
  Type of Page: 3
 +----+
  countEntries: 1
  -----+
       FS LLFR L L Page L FS LLFR L
                               I Page I FS I LFB
             --- no more FSI entries found ---
  FS = free space -- LFB largest free block -- (in 16 Bytes)
```

# Systemtabellen

Vorführung Systemstart

Systemtabellen CREATE TABL INSERT/SELEC [dbjtest]\$

## Systemtabellen

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

[dbjtest]  $$\operatorname{dbjstart}$$  The system was started successfully. SQLSTATE=00000

[dbjtest]\$

# Systemtabellen

Systemtabellen

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

[dbjtest]\$ dbj

### SYSTEM J

### Hannes Moser

Systemstart
Systemtabellen
CREATE TABL
INSERT/SELEC
Ausführungspla

# 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 ⇒ 🛮

#### Hannes Moser

Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELEC

## Systemtabellen

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

[dbjtest]\$ dbj Sustem 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;

1	ABLE_NAME	TABLE_ID	COLUMN_COUNT	CREATE_TIME		TUPLE_COUNT
9	YSTABLES					
	YSCOLUMNS	1	5	2005-02-28	13:45:24.839072	3
	YSINDEXES	2	6	2005-02-28	13:45:24.841806	18
-	ISTANEVES	3	7	2005-02-28	13:45:24.843554	7

3 record(s) returned.

db.i =>

## 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 werdent

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					
SYSCOLUMNS	1	5	2005-02-28 13	3:45:24.839072	3
SYSINDEXES	2	6	2005-02-28 13	3:45:24.841806	18
STSTMDEACS	3	7	2005-02-28 13	3:45:24.843554	7

3 record(s) returned.

dbj => select table\_id, index\_name, index\_type from sysindexes;

# Systemtabellen

7 2005-02-28 13:45:24.843554 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. db.j =>

Hannes Moser

CREATE TABLE

dbj ⇒ ▮

Hannes Moser

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT Ausführungsplan

dbj ⇒ CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

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

dbj ⇒ 🛮

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT

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

dbj => select \* from systables;∎

#### Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

### **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					
SYSCOLUMNS	1	. 5	2005-02-28	14:31:25.743915	4
	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

<sup>4</sup> record(s) returned.

dbj ⇒ ▮

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

db.j => select \* from systables;

TABLE_NAME	TABLE_ID	COLUMN_COUNT	CREATE_TIME	TUPLE_COUNT
SYSTABLES				
SYSCOLUMNS	1	5	2005-02-28 14:31:	25.743915 4
31300001113	2	6	2005-02-28 14:31:3	25,746265 20
SYSINDEXES	3	7	2005-02-28 14:31:3	25.748085 7
TEST	_			
	4	2	2005-02-28 14:39:	51,261357 0

<sup>4</sup> record(s) returned.

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

#### Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT

### 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
OUOTARI FO					
SYSTABLES	1	5	2005-02-28	14:31:25.743915	4
SYSCOLUMNS	2		000E 00 00	14.74.0E 74000E	20
SYSINDEXES	2	ь	2005-02-28	14:31:25.746265	20
TEST	3	7	2005-02-28	14:31:25.748085	7
IESI	4	2	2005-02-28	14:39:51.261357	0

<sup>4</sup> record(s) returned.

db.j =>

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

Vorführung Systemstart Systemtabellen CREATE TABLE

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

 $dbj \Rightarrow select * from systables;$ 

TABLE_NAME	TABLE_ID	COLUMN_COUNT	CREATE_TIME		TUPLE_COUNT
SYSTABLES					
	1	5	2005-02-28 1	L4:31:25.743915	4
SYSCOLUMNS	2	6	2005-02-28 1	L4:31:25.746265	20
SYSINDEXES	3	7	2005-02-28 1	L4:31:25.748085	7
TEST	4	2	2005-02-28 1	L4:39:51,261357	0

<sup>4</sup> record(s) returned.

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

dbj => select \* from test;■

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

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

dbj => select \* from test;

1 record(s) returned.

db,j ⇒ ■

<sup>4</sup> record(s) returned.

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

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

dbj => select \* from test;

1 record(s) returned.

db.j => insert into test values (2 | 'abcdefghijk');

<sup>4</sup> record(s) returned.

#### Hannes Moser

Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

### CREATE TABLE

```
SYSTABLES

SYSCOLUMNS

1 5 2005-02-28 14;31;25,743915 4

SYSINDEXES

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');
```

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

dbj => select \* from test;

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=CATO8
Rolling back transaction...

db.j =>

#### Hannes Moser

Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

### CREATE TABLE

```
SYSTABLES
                                                        5 2005-02-28 14:31:25.743915
SYSCOLUMNS
                                                        6 2005-02-28 14:31:25.746265
                                                                                               20
SYSTNDEXES
                                                        7 2005-02-28 14:31:25.748085
TEST
                                                         2 2005-02-28 14:39:51.261357
                                                                                                 Û
  4 record(s) returned.
db.i => insert into test values (1, 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
NHM
            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=CATO8
Rolling back transaction...

dbj => select \* from test;■

#### Hannes Moser

Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

```
SYSINDEXES
                                                        7 2005-02-28 14:31:25.748085
TEST
                                                        2 2005-02-28 14:39:51.261357
                                                                                                Λ
  4 record(s) returned.
db.i => insert into test values (1, 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
            STR
NHM
         1 abod
 1 record(s) returned.
dbj => insert into test values (2, 'abcdefghi.jk');
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 butes, SQLSTATE=CATO8
Rolling back transaction...
db,j => select * from test;
A table named 'TEST' does not exist, SQLSTATE=CP101
Rolling back transaction...
db.j =>
```

```
2 2005-02-28 14:39:51.261357
  4 record(s) returned.
db.j => insert into test values (1, 'abcd');
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
NHM
            STR
          1 abod
 1 record(s) returned.
db.i => insert into test values (2, 'abcdefqhi.ik'):
Cannot set the value 'abcdefohijk' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 butes. SQLSTATE=CATO8
Rolling back transaction...
db.i => select * from test:
A table named 'TEST' does not exist, SQLSTATE=CP101
Rolling back transaction...
db.i => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5)):
The operation was completed successfully, SQLSTATE=00000
db.j =>
```

#### Hannes Moser

Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

```
dbi => insert into test values (1. 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.j => select * from test;
NHM
            STR
          1 abod
 1 record(s) returned.
db.i => insert into test values (2, 'abcdefqhi.ik'):
Cannot set the value 'abcdefohi.ik' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 butes. SQLSTATE=CATO8
Rolling back transaction...
db.i => select * from test:
A table named 'TEST' does not exist, SOLSTATE=CP101
Rolling back transaction...
db.j => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => ■
```

### Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

```
db,j => select * from test;
NUM
            STR
          1 ahod
 1 record(s) returned.
db.i => insert into test values (2, 'abcdefqhi.ik'):
Cannot set the value 'abcdefohi.ik' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 butes. SQLSTATE=CATO8
Rolling back transaction...
db.i => select * from test:
A table named 'TEST' does not exist, SQLSTATE=CP101
Rolling back transaction...
db.i => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5)):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbi => insert into test values (1. 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.j =>
```

```
NUM
            STR
          1 abod
 1 record(s) returned.
db.i => insert into test values (2, 'abcdefghi.ik');
Cannot set the value 'abcdefghi, ik' for column 1 because the value has a length of 11 bytes but t
he column is defined with a maximum length of 5 butes. SQLSTATE=CAT08
Rolling back transaction...
db.i => select * from test:
A table named 'TEST' does not exist, SQLSTATE=CP101
Rolling back transaction...
db.i => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5)):
The operation was completed successfully. SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => insert into test values (1. 'abcd'):
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => ■
```

```
1 record(s) returned.
db.j => insert into test values (2, 'abcdefghi.jk');
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...
db.i => select * from test:
A table named 'TEST' does not exist. SQLSTATE=CP101
Rolling back transaction...
db.i => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5)):
The operation was completed successfully. SQLSTATE=00000
db.i => commit:
The operation was completed successfully. SQLSTATE=00000
db.i => insert into test values (1, 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbi => insert into test values (2, 'efgh'):
The operation was completed successfully, SQLSTATE=00000
db.i => ■
```

db.i =>

```
Vorführung
Systemstart
Systemtabellen
CREATE TABLE
```

```
A table named 'TEST' does not exist, SQLSTATE=CP101
Rolling back transaction...
db.j => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000
db.i => commit;
The operation was completed successfully, SQLSTATE=00000
db.i => insert into test values (1, 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully. SQLSTATE=00000
db.i => insert into test values (2, 'efqh'):
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
NUM
            STR
         1 abcd
          2 efgh
  2 record(s) returned.
```

```
CREATE TABLE
```

```
A table named 'TEST' does not exist, SQLSTATE=CP101
Rolling back transaction...
db.j => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully. SQLSTATE=00000
db.i => commit;
The operation was completed successfully, SQLSTATE=00000
db.i => insert into test values (1, 'abcd'):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully. SQLSTATE=00000
db.i => insert into test values (2, 'efqh'):
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
NUM
            STR
         1 abcd
          2 efgh
  2 record(s) returned.
dbj => rollback;
```

#### Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

```
CREATE TABLE
```

```
db.i => CREATE TABLE test (num INTEGER NOT NULL, str VARCHAR (5));
The operation was completed successfully, SQLSTATE=00000
db.j => commit;
The operation was completed successfully, SQLSTATE=00000
db.j => insert into test values (1, 'abcd');
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => insert into test values (2, 'efqh'):
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
NUM
            STR
         1 abcd
          2 efgh
  2 record(s) returned.
db.i => rollback:
The operation was completed successfully, SQLSTATE=00000
db.i =>
```

```
dbj => commit;
The operation was completed successfully. SQLSTATE=00000
db.i => insert into test values (2, 'efgh');
The operation was completed successfully, SQLSTATE=00000
db.i => select * from test:
NHM
            STR
          1 abcd
          2 efah
  2 record(s) returned.
db.i => rollback:
The operation was completed successfully, SQLSTATE=00000
dbj => select * from test;
NUM
            STR
          1 abcd
 1 record(s) returned.
db.j =>
```

### Hannes Moser

CREATE TABLE

The operation was completed successfully, SQLSTATE=00000

db,j => select \* from test:

NHM STR

1 abod 2 efgh

2 record(s) returned.

db.i => rollback:

The operation was completed successfully, SQLSTATE=00000

db.i => 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=CATO7 Rolling back transaction...

db.j =>

#### Hannes Moser

Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

### CREATE TABLE

CTD

1 abcd 1 record(s) returned.

NON	SIR
	1 abcd 2 efgh
2 r	ord(s) returned.
	rollback; ration was completed successfully. SQLSTATE=00000
dbj =	select * from test;
NUM	STR

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=CATO7 Rolling back transaction...

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

db.j => ■

мим

## INSERT/SELECT

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

dbj ⇒ ▮

#### Hannes Moser

INSERT/SELECT

## INSERT/SELECT

- db.i => CREATE TABLE angest (
- dbj => persNr INTEGER NOT NULL,
- db.i => vorname VARCHAR(10),
- dbj => nachname VARCHAR(20) NOT NULL,
- dbj ⇒ dbj ⇒ gehalt INTEGER NOT NULL,
- adresse VARCHAR(20), dbj =>
- dbj => PRIMARY KEY(persNr));

#### Hannes Moser

Vorführung Systemstart Systemtabellen

#### INSERT/SELECT

Ausführungspla

## 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 ⇒>
```

#### Hannes Moser

#### Vorführung Systemstart Systemtabellen

#### INSERT/SELECT

Austunrungspia

## INSERT/SELECT

db.i => CREATE TABLE angest (

```
db.i =>
        persNr INTEĞER NOT NULL.
db.i =>
        vorname VARCHAR(10).
dbj =>
        nachname VARCHAR(20) NOT NULL.
dbj =>
        gehalt INTEGER NOT NULL.
db.i =>
        adresse VARCHAR(20).
dbj =>
dbj => PRIMARY KEY(persNr));
The operation was completed successfully, SQLSTATE=00000
dbj => commit;
The operation was completed successfully, SQLSTATE=00000
db,j => CREATE INDEX angest_nachname ON angest (nachname);
```

#### Hannes Moser

Vorführung Systemstart Systemtabellen

#### INSERT/SELECT

Austunrungspia

## INSERT/SELECT

db.i =>

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

#### Hannes Moser

Vorführung Systemstart Systemtabellen

#### INSERT/SELECT

Ausführungspla

```
INSERT/SELECT
```

```
db.i => CREATE TABLE angest (
        persNr INTEGER NOT NULL.
db.i =>
db.i =>
        vorname VARCHAR(10).
db.i =>
        nachname VARCHAR(20) NOT NULL,
db.i =>
        gehalt INTEGER NOT NULL.
db.i =>
         adresse VARCHAR(20).
db.i =>
dbj => PRIMARY KEY(persNr));
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db,j => CREATE INDEX angest_nachname ON angest (nachname);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully. SQLSTATE=00000
db.j => select * from sysindexes where index_name = 'angest_nachname':
```

#### Hannes Moser

Vorführung Systemstart Systemtabellen CREATE TABLE

INSERT/SELECT

INSERT/SELECT

```
db.i =>
        PRIMARY KEY(persNr)):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => CREATE INDEX angest_nachname ON angest (nachname);
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db,j => select * from sysindexes where index_name = 'angest_nachname';
TABLE_ID
           INDEX_NAME
                                                      INDEX ID
TIME
         5 ANGEST_NACHNAME
                                                                          2 N
                                                    9 BTREE
                                                                                      2005-0
2-28 14:51:33.141130
 1 record(s) returned.
db.i => ■
```

#### Hannes Moser

#### INSERT/SELECT

## INSERT/SELECT

```
dbj => CREATE TABLE projekt (
```

dbj ⇒ dbj ⇒ projNr INTEGER NOT NULL,

VARCHAR(20) NOT NULL, name dbj => INTEGER, prioritaet

dbj ⇒ dbj ⇒ beschreibung VARCHAR(1000),

PRIMARY KEY (projNr)); dbj =>

### Hannes Moser

# Vorführung

Systemtabellen

# INSERT/SELECT

Austührungspl

```
INSERT/SELECT
```

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

## Hannes Moser

Vorführung

Systemtabellen

# INSERT/SELECT

Austuhrungspl

# INSERT/SELECT

```
dbj => CREATE TABLE projekt (
dbi => projNr INTEGER
```

dbj => projNr INTEGER NOT NULL,

dbj => name VARCHAR(20) NOT NULL, dbj => prioritaet INTEGER,

dbj => beschreibung VARCHAR(1000), dbi =>

dbj =>
dbj => PRIMARY KEY (projNr));

The operation was completed successfully. SQLSTATE=00000

dbj => CREATE INDEX projekt\_prio ON projekt(prioritaet);▮

## Hannes Moser

Vorführung Systemstart Systemtabellen

# CREATE TABLE INSERT/SELECT

Ausführungspla

# INSERT/SELECT

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

## Hannes Moser

Vorführung Systemstart Systemtabellen

# CREATE TABLE INSERT/SELECT

Austührungspl

```
INSERT/SELECT
```

```
db.i => CREATE TABLE projekt (
db.i =>
                        INTEGER NOT NULL.
        pro.iNn
db.i =>
        name
                       VARCHAR(20) NOT NULL.
db.i =>
        prioritaet
                      INTEGER.
        beschreibung VARCHAR(1000),
db.i =>
db.i =>
dbj => PRIMARY KEY (projNr));
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully, SQLSTATE=00000
dbj => commit;
The operation was completed successfully, SQLSTATE=00000
db.i =>
```

## Hannes Moser

Vorführung Systemstart Systemtabellen

# INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => CREATE TABLE projekt (
                        INTEGER NOT NULL.
db.i =>
         pro.iNn
db.i =>
         name
                        VARCHAR(20) NOT NULL.
db.i => prioritaet
                      INTEGER.
         beschreibung VARCHAR(1000),
db.i =>
db.i =>
dbj => PRIMARY KEY (projNr));
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into angest values (1, 'Klaus', 'Kuespert', 10000, NULL),
                                 (2, 'Knut', 'Stolze', 1500, 'Ernst-Abbe-Platz 2'),
db.i =>
                                 (3, Thomas', Mueller', 3000, 'Ernst-Abbe-Platz 2'),
db.i =>
db.i =>
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
```

# Hannes Moser

Vorführung Systemstart Systemtabellen

# CREATE TABLE INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => CREATE TABLE projekt (
db.i =>
         pro.iNn
                        INTEGER NOT NULL.
db.i =>
         name
                        VARCHAR(20) NOT NULL.
db.i => prioritaet
                       INTEGER.
         beschreibung VARCHAR(1000),
db.i =>
db.i =>
dbj => PRIMARY KEY (projNr));
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into angest values (1, 'Klaus', 'Kuespert', 10000, NULL),
                                 (2, 'Knut', 'Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbi =>
                                 (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2'),
db.i =>
dhi =>
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully, SQLSTATE=00000
db.i => ■
```

### Hannes Moser

Vorführung Systemstart Systemtabellen

# INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => CREATE TABLE projekt (
db.i =>
         pro.iNn
                        INTEGER NOT NULL.
db.i =>
         name
                       VARCHAR(20) NOT NULL.
db.i => prioritaet
                      INTEGER.
         beschreibung VARCHAR(1000),
db.i =>
db.i =>
dbj => PRIMARY KEY (projNr));
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into angest values (1, 'Klaus', 'Kuespert', 10000, NULL),
                                 (2, 'Knut', 'Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbi =>
                                 (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2').
db.i =>
dhi =>
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully, SQLSTATE=00000
db.i => commit;
The operation was completed successfully. SQLSTATE=00000
db.i => ■
```

Vorführung Systemstart Systemtabellen

INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => CREATE TABLE projekt (
db.i =>
        pro.iNn
                        INTEGER NOT NULL.
db.i =>
        name
                       VARCHAR(20) NOT NULL.
db.i => prioritaet
                      INTEGER.
db.i =>
        beschreibung VARCHAR(1000).
db.i =>
dbj => PRIMARY KEY (projNr));
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX projekt_prio ON projekt(prioritaet);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into angest values (1, 'Klaus', 'Kuespert', 10000, NULL),
                                 (2, 'Knut', 'Stolze', 1500, 'Ernst-Abbe-Platz 2'),
dbi =>
                                 (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2'),
db.i =>
dhi =>
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully, SQLSTATE=00000
db.i => commit;
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE TABLE mitarbeit (
db.i =>
        persNr INTEGER NOT NULL,
db.i => pro.iNr INTEGER NOT NULL,
db.i => percent INTEGER):■
```

Vorführung
Systemstart
Systemtabellen

# INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => PRIMARY KEY (pro.iNr)):
The operation was completed successfully. SQLSTATE=00000
db.i => CREATE INDEX projekt prio ON projekt(prioritaet):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into angest values (1, 'Klaus', 'Kuespert', 10000, NULL),
                                 (2, 'Knut', 'Stolze', 1500, 'Ernst-Abbe-Platz 2'),
db.i =>
                                 (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2'),
db.i =>
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
db.i =>
The operation was completed successfully. SQLSTATE=00000
db.j => commit;
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE TABLE mitarbeit (
db.i => persNr INTEGER NOT NULL,
db.i => pro.iNr INTEGER NOT NULL,
db.i => percent INTEGER);
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr):
The operation was completed successfully. SQLSTATE=00000
db.i => ■
```

### Hannes Moser

Vorführung Systemstart Systemtabellen

# INSERT/SELECT

Austührungspl

```
INSERT/SELECT
```

```
db.i => CREATE INDEX projekt prio ON projekt(prioritaet):
The operation was completed successfully. SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into angest values (1. 'Klaus'. 'Kuespert'. 10000. NULL).
db.i =>
                                 (2. 'Knut'. 'Stolze'. 1500. 'Ernst-Abbe-Platz 2').
                                 (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2'),
db.i =>
db.i =>
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
db,j => CREATE TABLE mitarbeit (
dbi =>
         persNr INTEGER NOT NULL.
db.i =>
         projNr INTEGER NOT NULL,
db.i => percent INTEGER);
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE INDEX mitarbeit_pro.iNr ON mitarbeit (pro.iNr):
The operation was completed successfully. SQLSTATE=00000
db.i => ■
```

Vorführung Systemstart Systemtabellen

# INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => commit:
The operation was completed successfully. SQLSTATE=00000
dbj => insert into angest values (1. 'Klaus', 'Kuespert', 10000, NULL).
                                 (2. Knut' Stolze' 1500, Ernst-Abbe-Platz 2').
db.i =>
db.i =>
                                 (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2'),
                                 (4, 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1');
db.i =>
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE TABLE mitarbeit (
db.i => persNr INTEGER NOT NULL.
db.j => pro.jNr INTEGER NOT NULL,
db.j => percent INTEGER);
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully. SQLSTATE=00000
db.i => CREATE INDEX mitarbeit_pro.iNr ON mitarbeit (pro.iNr);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully. SQLSTATE=00000
db.i => ■
```

INSERT/SELECT

```
INSERT/SELECT
```

```
db.i => insert into angest values (1, 'Klaus', 'Kuespert', 10000, NULL),
                                  (2. 'Knut', 'Stolze', 1500, 'Ernst-Abbe-Platz 2'),
db.i =>
                                  (3, 'Thomas', 'Mueller', 3000, 'Ernst-Abbe-Platz 2'),
db.i =>
                                  (4. 'Hannes', 'Moser', 300, 'Ernst-Abbe-Platz 1'):
db.i =>
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE TABLE mitarbeit (
db.i => persNr INTEGER NOT NULL.
db.j =>
         projNr INTEGER NOT NULL.
dbj => percent INTEGER):
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX mitarbeit_persNr ON mitarbeit (persNr);
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX mitarbeit_pro.jNr ON mitarbeit (pro.jNr);
The operation was completed successfully, SQLSTATE=00000
db.i => commit;
The operation was completed successfully. SQLSTATE=00000
db.i => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
db.i =>
                                   (2, 'DBS1', 5, NULL),
                                   (3, 'XML-Seminar', 99, NULL),
db.i =>
                                   (4. 'DBS2', 6, NULL);
db.i =>
```

Vorführung Systemstart Systemtabellen CREATE TABLE

```
INSERT/SELECT
```

```
The operation was completed successfully. SQLSTATE=00000
db.i => commit:
The operation was completed successfully. SQLSTATE=00000
db.i => CREATE TABLE mitarbeit (
db.i => persNr INTEGER NOT NULL.
db.i => pro.iNr INTEGER NOT NULL.
dbj =>
         percent INTEGER):
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE INDEX mitarbeit persNr ON mitarbeit (persNr):
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully, SQLSTATE=00000
db.j => commit;
The operation was completed successfully. SQLSTATE=00000
db.j => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
db.i =>
db.i =>
                                  (3, 'XML-Seminar', 99, NULL),
db.i =>
                                  (4, 'DBS2', 6, NULL):
A table named 'PROJECT' does not exist. SQLSTATE=CP101
Rolling back transaction...
db.i => ■
```

/orführung Systemstart Systemtabellen CREATE TABLE

INSERT/SELECT Ausführungsplan

# INSERT/SELECT

```
The operation was completed successfully. SQLSTATE=00000
db.i => CREATE TABLE mitarbeit (
db.i =>
         persNr INTEGER NOT NULL.
db.i =>
         projing INTEGER NOT NULL.
db.i =>
         percent INTEGER):
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE INDEX mitarbeit persNr ON mitarbeit (persNr):
The operation was completed successfully, SQLSTATE=00000
db.j => CREATE INDEX mitarbeit_projNr ON mitarbeit (projNr);
The operation was completed successfully, SQLSTATE=00000
dbi => commit:
The operation was completed successfully, SQLSTATE=00000
db.j => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
dhi =>
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
db.i =>
dhi =>
                                  (4, 'DBS2', 6, NULL):
A table named 'PROJECT' does not exist. SOLSTATE=CP101
Rolling back transaction...
db.i => insert into projekt values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL):■
```

Vorführung
Systemstart
Systemtabellen
CREATE TABLE

INSERT/SELECT

# INSERT/SELECT

```
db.i =>
         persNr INTEGER NOT NULL.
db.i =>
         projiNr INTEGER NOT NULL.
db.i =>
         percent INTEGER):
The operation was completed successfully. SQLSTATE=00000
db.i => 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
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.j => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
db.i =>
db.i =>
                                  (3, 'XML-Seminar', 99, NULL),
dhi =>
                                  (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist. SOLSTATE=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
db.i => ■
```

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

```
INSERT/SELECT
```

```
The operation was completed successfully. SQLSTATE=00000
db.i => CREATE INDEX mitarbeit persNr ON mitarbeit (persNr):
The operation was completed successfully, SQLSTATE=00000
db.i => CREATE INDEX mitarbeit projNr ON mitarbeit (projNr):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
                                  (2, 'DBS1', 5, NULL),
db.i =>
db.i =>
                                  (3, 'XML-Seminar', 99, NULL),
                                  (4, 'DBS2', 6, NULL);
dbi =>
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
db.i => commit:
The operation was completed successfully. SQLSTATE=00000
db.i => ■
```

/orführung Systemstart Systemtabellen CREATE TABLE

```
INSERT/SELECT
```

```
The operation was completed successfully. SQLSTATE=00000
db.i => CREATE INDEX mitarbeit pro.iNr ON mitarbeit (pro.iNr):
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
db.i =>
                                   (2. 'DBS1'. 5. NULL).
dbj =>
                                   (3, 'XML-Seminar', 99, NULL),
db.i =>
                                   (4, 'DBS2', 6, NULL);
A table named 'PROJECT' does not exist, SOLSTATE=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
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i \Rightarrow insert into mitarbeit values (1.1.10), (1.2.30), (1.3.2),
db.i =>
                                     (2.1.90).
db.i =>
                                     (3.1.0).(3.3.30).
db.i =>
                                     (4.1.100):■
```

/orführung Systemstart Systemtabellen CREATE TABLE

```
INSERT/SELECT
```

```
The operation was completed successfully. SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
db.i =>
                                   (2, 'DBS1', 5, NULL),
dbj =>
                                   (3, 'XML-Seminar', 99, NULL).
db.i =>
                                   (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
db.j => commit;
The operation was completed successfully. SQLSTATE=00000
db.i \Rightarrow insert into mitarbeit values (1.1.10), (1.2.30), (1.3.2),
db.i =>
                                     (2.1.90),
db.i =>
                                      (3.1.0).(3.3.30).
db.i =>
                                     (4.1.100):
The operation was completed successfully. SQLSTATE=00000
db.i => ■
```

/orführung Systemstart Systemtabellen CREATE TABLE

```
INSERT/SELECT
```

```
The operation was completed successfully. SQLSTATE=00000
dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
db.i =>
                                   (2. 'DBS1', 5. NULL).
                                   (3, 'XML-Seminar', 99, NULL),
db.i =>
db.i =>
                                   (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, SOLSTATE=00000
db.j => commit;
The operation was completed successfully, SQLSTATE=00000
db.j =  insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
db.i =>
                                     (2,1,90),
                                      (3,1,0),(3,3,30),
db.i =>
db.i =>
                                     (4,1,100);
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => ■
```

Vorführung
Systemstart
Systemtabellen
CREATE TABLE

CREATE TABLE
INSERT/SELECT

```
INSERT/SELECT
```

```
The operation was completed successfully. SQLSTATE=00000
dbj => insert into project values (1, 'DBS-Entwicklung', 1, NULL),
db.i =>
                                   (2. 'DBS1', 5. NULL).
                                   (3, 'XML-Seminar', 99, NULL),
db.i =>
db.i =>
                                   (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, SOLSTATE=00000
db.j => commit;
The operation was completed successfully, SQLSTATE=00000
db.j =  insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
db.i =>
                                     (2,1,90),
db.i =>
                                     (3,1,0),(3,3,30),
db.i =>
                                     (4,1,100);
The operation was completed successfully, SQLSTATE=00000
db.i => commit:
The operation was completed successfully, SQLSTATE=00000
db.i => select * from angest:
```

INSERT/SELECT

```
(3, 'XML-Seminar', 99, NULL),
(4, 'DBS2', 6, NULL):
```

The operation was completed successfully. SQLSTATE=00000

db.i => commit:

The operation was completed successfully, SQLSTATE=00000

- dbj = insert into mitarbeit values (1,1,10), (1,2,30), (1,3,2),
- db.i => (2,1,90),
- (3.1.0).(3.3.30). dbj => (4.1.100):
- db.i =>

The operation was completed successfully, SQLSTATE=00000

dbi => commit:

The operation was completed successfully, SQLSTATE=00000

db.j => 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.

db.i => ■

# INSERT/SELECT

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

dbj ⇒ ▮

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

dbj ⇒ select projNr, name, prioritaet from projekt∰

# INSERT/SELECT

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

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

4 record(s) returned.

dbj ⇒

Vorführung
Systemstart
Systemtabellen
CREATE TABLE

INSERT/SELECT Ausführungsplan

# INSERT/SELECT

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

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

4 record(s) returned.

dbj ⇒ select \* from mitarbeit;

INSERT/SELECT

# INSERT/SELECT

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

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

4 record(s) returned.

db,j => select \* from mitarbeit;

PERSNR	PROJNR	PERCENT
1	. 1	10
1	. 2	30 2
1	. 3	
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.

db.j => select \* from mitarbeit;

PERSNR	PROJNR	PERCENT
1	1	10
1	. 2	30
1	. 3	2
2 3 3	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,
- db.i => 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
- db.i => p.pro.inr = m.pro.inr:

# Hannes Moser

Vorführung Systemstart Systemstabellen CREATE TABLE INSERT/SELECT

# INSERT/SELECT

1	2	30 2
1	3	2
1 2 3 3	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 1 2 3 3	Klaus Klaus Klaus Knut Thomas Thomas Hannes	Kuespert Kuespert Kuespert Stolze Mueller Mueller Moser	2 3 1 1 3	DBS-Entwicklung DBS1 XML-Seminar DBS-Entwicklung DBS-Entwicklung XML-Seminar DBS-Entwicklung	10 30 2 90 0 30 100

7 record(s) returned.

dbj ⇒

# Ausführungsplan

Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

dbj ⇒ 🛮

# Ausführungsplan

dbj => select \* from angest;■

### Hannes Moser

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELEC

Ausführungsplan

# 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) [Correlation name: ANGEST] - Column ('AURESSE'/4) [Correlation name: ANGEST]

Sources - Table ('ANGEST'/5)

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

4 record(s) returned.

dbj ⇒ ▮

### Hannes Moser

Vorführung Systemstart Systemtabellen CREATE TABLE INSERT/SELECT

Ausführungsplan

# Ausführungsplan

# SelectStmt

Projections - Column ('PERSMR'/0) [Correlation name: ANGEST] - Column ('VORNAME'/1) [Correlation name: ANGEST] - Column ('NACHNAME'/2) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correlation name: ANGEST]

Sources - Table ('ANGEST'/5)

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

<sup>4</sup> record(s) returned.

dbj => select \* from angest where nachname = 'Moser'

### Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

# 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.

# SelectStmt

Projections - Column ('PERSNR'/0) [Correlation name: ANGEST] - Column ('VORNAME'/1) [Correlation name: ANGEST] - Column ('NACHNAME'/2) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correlation name: ANGEST] - Column ('GEHALT'/3) [Correlation name: ANGEST] - Column ('GEHALT'/3) [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.

db.j =>

### Hannes Moser

Vorführung
Systemstart
Systemtabellen
CREATE TABLE
INSERT/SELECT
Ausführungsplan

```
4 record(s) returned.
db.i => select * from angest where nachname = 'Moser':
Optimized access plan
SelectStmt
Projections - Column ('PERSNR'/O) [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
                                                300 Ernst-Abbe-Platz 1
          4 Hannes
                      Moser
 1 record(s) returned.
dbj => SELECT a.persnr AS angest_nr, a.vorname, a.nachname, p.projnr AS proj_nr,
db.j => p.name AS pro.j_name, m.percent
db.j => FROM angest AS a, mitarbeit AS m, projekt AS p
db.j => WHERE a.persnr = m.persnr AND
db.i => p.pro.jnr = m.pro.jnr;
```

## SYSTEM I

Ausführungsplan

Ausführungsplan Hannes Moser

```
R'/0) [Correlation name: p] [New column name: pro.i 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 ('PERSNR'/O) [Correlation name: a] - Comparison ('=') - Column ('PERSNR'/O) [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.

db.i =>