Fachbereich Mathematik und Informatik
Fachgebiet Informatik
Universität Marburg

SoSe 00
4. Mai 2000
Übungsblatt 3

Prof. Dr. Bernhard Seeger
Jochen van den Bercken

Übungen zur Vorlesung

Implementierung von Datenbanksystemen

Sie erhalten in der Übung vom 5.5. nähere Informationen zu den Klassen der XXL-Bibliothek!
Abgabe am 19. 5. in der Übung!

Aufgabe 3.1: Hashing (15)

Implementieren Sie einen ResultSet-Operator Hasher mit folgender Signatur:

Hasher(Function [] hashFunctions, int fanOut, Function newQueue, ResultSet resultSet)

Dieser Operator soll die Tupel des gegebenen *resultSet* mit Hilfe der *hashFunctions* in Partitionen unterteilen. Die einzelnen *hashFunctions* werden mit einem Tupel aufgerufen und liefern ein Integer-Objekt zurück. Dessen Wert modulo *fanOut* bestimmt die Nummer der Partition, der das Tupel zugeordnet werden soll. Die einzelnen Partitionen werden durch jeweils eine Queue implementiert. Wann immer eine neue Partition erzeugt werden soll, ruft man die parameterlose Funktion *newQueue* auf, die eine neue und leere Queue zurückliefert.
Die Partitionierung soll rekursiv mit Hilfe der *hashFunctions* erfolgen, d. h. die Tupel des *resultSet* werden mit Hilfe von *hashFunctions*[0] partitioniert, diese Partitionen wiederum mit Hilfe von *hashFunctions*[1] etc.
Die *next*-Funktionalität des *Hashers* wird so implementiert, daß die sich nach Anwendung der letzten Hashfunktion ergebenden Partitionen in hierarchisch aufsteigender Nummerierung nach und nach tupelweise durchlaufen werden.

Aufgabe 3.2: HashJoin (16)

Implementieren Sie einen ResultSet-Operator HashJoin mit folgender Signatur:

HashJoin(Function [] hashFunctions, int tupelsInMemory, Function newBag, ResultSet resultSet1, ResultSet resultSet2)

Gehen Sie davon aus, daß beide ResultSets *Hasher* sind, die auf demselben Array von Hashfunktionen *hashFunctions* arbeiten, d. h. es werden jeweils die Tupel der korrespondierenden Partitionen geliefert. Mit Hilfe dieses Arrays können Sie die Nummer der Partition eines Tupel berechnen. *tupelsInMemory* gibt Ihnen an, wieviele Tupel Sie maximal im Hauptspeicher des Operators verwalten dürfen. *newBag* gibt Ihnen nach einem parameterlosen Aufruf einen neue und leeren Bag zurück, in der Sie Zwischenergebnisse auslagern können.