Litheraturverzeichnis:

Institut für Parallele und Verteilte Hochleistungsrechner (IPVR), Universität Stuttgart: PARALLEL EXECUTION WITHIN DATABASE SYSTEMS, Videofilm von 1994, VHS, 12 min., zu beziehen durch IPVR Uni-Stuttgart

Kendall Square Research Corporation: KSR QD Product Overview, Kapitel 9 - 11, Handbuch zu Oracle 7.0 und Query Decomposer

Kredel, Heinz und Schumacher, Robert (eds.): Oracle Datenbankkopplung SNI BS2000 - KSR1, Juni 1994, Rechenzentrum Universität Mannheim, Bericht RUM 37/94

Reuter, Andreas: Parallele Datenbanksysteme - Eine Übersicht unter Architektur und Leistungsaspekten, in it+ti 2/95, Seite 18 - 28

Schulze, Hans Herbert: Computer Enzyklopädie - Lexikon und Fachwörterbuch für Datenverarbeitung und Telekommunikation, Rowohlt Taschenbuch Verlag GmbH, Reinbek bei Hamburg, 1993, Seite 793 ff.

Stürner, Günther: Oracle7, Release 7.1 - Die verteilte semantische Datenbank, April 1993, Buch zu Oracle 7.1

Tseng, Emy und Reiner, David: Parallel Database Processing on the KSR1 Computer

→ Memory-Bus-Bandbreite
ein sekundärer Faktor, besonders bei SMP-Systemen, da alle Prozessoren über den Memory-Bus auf den gemeinsamen Hauptspeicher zugreifen

→ DB-Cache
ein sekundärer Faktor, die Größe des DB-Caches bestimmt die Menge der Daten, die Im Arbeitsspeicher gehalten werden, dort bearbeitet und somit nicht ständig von Platte gelesen werden müssen.

→ I/O-Bandbreite
ein sekundärer Faktor, bei hohem Parallelisierungsgrad und intensiven Zugriffen auf die Platten kommt es zu einer starken Belastung des I/O-Systems, so daß dessen Bandbreite Einfluß auf den Parallelisierungsgrad nimmt.

→ Diskkontroller, ein primärer Faktor, da Daten ständig von / auf Platte gelesen / geschrieben werden müssen.

→ Verteilung von temporären Segmenten
ein primärer Faktor, bei bestimmten Operationen, die eine Sortierung nötig machen werden Teilergebnisse in temporären Tabellen abgelegt. Die Verteilung dieser Tabellen auf verschiedene Partitionen spielt beim Weiterverarbeiten (z.B. Zusammenmischen von Teilergebnissen) eine große Rolle (vgl. hierzu Abbildung 1 Folie 15)

→ Verteilung der Daten
ein primärer Faktor, sind die Daten nicht gleichmäßig verteilt, liegt keine Lastenbalance vor, so kann keine effektive Verteilung auf die einzelnen Prozessoren erfolgen oder es muß eine Repartitionierung zur laufzeit erfolgen.

Folie 21: Optimierung eines DBS

Da viele Faktoren Einfluß auf die Parallelisierbarkeit von Queries nehmen, gibt es viele Möglichkeiten eine Datenbank zu optimieren, die Einstellungen des DBS den spezifischen Anforderungen der Datenbank anzupassen. Einige Hinweise und Ansatzpunkte werden hierfür auf Folie 21 gegeben.

Für optimale Ergebnisse bei der Parallelisierung von Queries bedarf es genauer Kenntnisse der Datenbeschaffenheit, des Aufbaus der Datenbank, der Architektur des DB-Rechners und den eingesetzten Komponenten.

Folie 19: Einsatz der Parallelen Query Technologie

Die Steuerung des Parallelisierungsgrades einer Abfrage kann auf drei Arten geschehen. Die Reihenfolge der Aufzählung gibt die Hierarchie der Parallelisierungsangaben an:

1) SQL-Hinweise (Hints):
   SQL-Hinweise sind explizite Angaben des maximal möglichen Parallelisierungsgrades für einen Befehl. Der Befehl PARALLEL <N> bzw. NONPARALLEL steht unmittelbar nach dem eigentlichen SQL-Befehl (vgl. Beispiel). SQL-Hinweise bilden die oberste Hierarchiestufe.

2) Oracle7-Data-Dictionary:
   Im Oracle7-Data-Dictionary werden Informationen über dern maximalen Parallelisierungsgrad einzelner Objekte (Tabellen, Views, etc.) gespeichert. Solche Angaben werden z.b. beim Anlegen einer Tabelle durch die Angabe PARALLEN <N> gespeichert oder durch ALTER TABLE <Tabellenname> PARALLEL <N> geändert. Definitionen oder Änderungen können auch durch PARALLEL(<Objektname>, <N>) erfolgen. Das Oracle7-Data-Dictionary bildet die zweite Hierarchiestufe, d.h. werden keine explizite Angaben durch SQL-Hints gemacht, wird versucht über das Oracle7-Data-Dictionary ein Parallelisierungsgrad für das Objekt zu definieren. Die letzte Hierarchiestufe beinhaltet Default-Werte aus dem init.ora-File.

3) INIT.ORA-File:
   Werden für einen SQL-Befehl weder explizite Angaben durch SQL-Hints gemacht noch sind Angaben über die beteiligten Objekte im Oracle7-Data-Dictionary vorhanden, so wird versucht, den Befehl mit Default-Werten aus dem INIT.ORA-File zu parallelisieren.
   Das INIT.ORA-File enthält Parameter, die die maximale und minimale Anzahl an Prozessen zur Parallelisierung festlegt; Parameter, die die Größe des DB-Caches festlegen für die Sortierung der Daten eines einzelnen Prozesses usw. allgemein Angaben zu einer Standardumgebung für die Parallelisierung von Abfragen. Diese Parameter können vom DB-Administrator geändert und den spezifischen Anforderungen einer DB angepaßt werden.

Folie 20: Einflußfaktoren auf die Parallelisierbarkeit

Um ein Problem, das mit einem Prozessor in einer Zeit t gelöst werden kann, mit n Prozessoren zu lösen, ist theoretisch nur 1/n der Zeit nötig, da jeder Prozessor 1/n des Problemes lösen könnte. Daß diese theoretische Formel in der Praxis nicht korrekt ist, läßt sich schon daran deutlich machen, daß nicht jedes Problem in genau n gleichschwierige Teilprobleme zerlegt werden kann.
Welche weiteren Faktoren Einfluß auf die Parallelisierbarkeit haben und an welcher Stelle sie wirken wird an der Abbildung deutlich.

→ Anzahl der CPUs
   ein primärer Einflußfaktor, er beschränkt den maximalen Parallelisierungsgrad

Im ersten Beispiel wird in der WHERE-Klausel eine Abfrage der rowid eingefügt, so daß auf jeder Partition nur die Daten gesucht werden, die auch in dieser Partition liegen. Die Variable i bezeichnet die Anzahl der Partitionen, auf die die Relation verteilt ist.
Im zweiten Beispiel wird zur Ermittlung des Durchschnitts die Summe und die Anzahl der Elemente parallel für die einzelnen Partitionen ermittelt. Beides wird in einer temporären Relation zwischengespeichert. In einem weiteren Schritt wird aus den Teilergebnissen in der temporären Datei das arithmetische Mittel, der gesuchte Durchschnitt, ermittelt.

Folie 15: Leistungsdaten: Oracle 7.0 & QD

Abbildung 1 zeigt, daß bei der Verwendung von 16 Laufwerken ein höherer Speedup als mit 4 Laufwerken erzielt werden kann. Die Möglichkeiten, durch Parallelisierung mit Hilfe des QD, einen Speedup zu erreichen steigen mit der Anzahl der Laufwerke und damit der Partitionen, auf die die Daten aufgeteilt sind. Außerdem wird klar, daß ein linearer Speedup in der Praxis nicht realisierbar ist.
Abbildung 2 zeigt, daß nicht bei jeder beliebigen Query ein maximaler Speedup erreicht werden kann. Die Parallelisierungsmöglichkeit ist also durchaus von der Art der Abfrage abhängig. Im Schnitt ist jedoch ein Speedup von 8, d.h. 50% möglich.

⇨ vgl. auch Kredel, Schumacher(eds.), 7/94, RUM 37/94

Folie 17: Hardware-Voraussetzungen für Oracle 7.1

Die Nutzung von Oracle 7.1 ist unabhängig von der Architektur des Rechners, auf dem das DBS installiert wird. Die Fähigkeiten, die Resultate und die Möglichkeiten zur Nutzung der Parallelen-Query-Technologie steigen jedoch mit der Parallelität der zugrundeliegenden Architektur.
Dennoch ist schon auf einem Einzelprozessor-System eine Performencesteigerung zu beobachten, wenn die Parallel-Query-Technologie eingesetzt wird.

Folie 18: Nutzung der Parallelen Query Technologie

Zur Nutzung der Parallelen-Query-Technologie muß lediglich beim Anlegen einer Tabelle der maximale Parallelisierungsgrad für die Tabelle angegeben werden. Dies geschieht durch den Zusatz PARALLEL <N> oder NONPARALLEL. Im Vergleich zur seriellen Abarbeitung unter Oracle 7.0 oder bei der nicht-parallelen Ausführung können zur gleichen Zeit mehrere Prozessoren zur Bearbeitung eines Problemes genutzt werden. Die Verteilung der Daten auf die einzelnen Prozessoren, die Partitionierung bedarf es keiner besonderen Programmierung, sie erfolgt durch das System. Alle wesentlichen Funktionen der relationalen Algebra können von Oracle 7.1 parallelisiert werden.

Folie 11: Rechnerarchitekturen für parallele DBS

Zur Realisierung eines parallelen DBSs ist ein Rechner mit paralleler Architektur nötig. Es kann folgende Einteilung vorgenommen werden:

- shared nothing
  → jeder Prozessor besitzt eigenen Hauptspeicher und eigene Peripherie
  → Kommunikationssystem zwischen den Prozessoren
  → Beispiele: Tandem, Workstation Cluster, IBM SP2
  Vorteile:
  → optimale Fehlerisolierung
  → einfache Skalierbarkeit

- shared disk
  → jeder Prozessor verfügt über eigenen Hauptspeicher
  → Kommunikationssystem zwischen den Prozessoren
  → alle Prozessoren können auf alle Platten zugreifen
  → Beispiele: VAX Cluster, nCube, IBM IRLM
  Vorteile:
  → gute Fehlerisolierung
  → einfache Skalierbarkeit
  → bei Einsatz "elektronischer Platten" (RAM-Speicher) hohe Zugriffsraten

- shared everything
  → SMP (SYMETRIC MULTIPROCESSOR)
  → Grupierung aller Prozessoren um gemeinsamen Hauptspeicher
  → Peripherie wird gemeinsam genutzt
  → Beispiele: Convex, Sequent, Sequoia
  Vorteile:
  → einfache parallele Programmierung
  → optimale Möglichkeit zur Lastbalancierung
  Nachteil:
  → schwierige Fehlerisolierung wegen gemeinsamem Hauptspeicher

Folie 14: Beispiele für Funktionsweise des QD

Die Abbildung zeigt wie sich der Query Decomposer zwischen SQL-Abfrage und Datenbanksystem "hineinschaltet" und Abfragen in Unterabfragen, die auf den einzelnen Partitionen in parallelen Prozessen abgearbeitet werden, zerlegt. Die Teilergebnisse nimmt der Query Decomposer von den einzelnen Prozessen des Rechners entgegen und setzt sie zu einem Gesamtergebnis zusammen, das er als Ergebnis zurückgibt, als ob es seriell abgearbeitet worden wäre.

Folie 8/9: Beispiele serieller / paralleler Bearbeitung von DB-Operationen

- Insert (z.B. durch Key-Range-Partitioning)
  Die Daten werden beim Key-Range-Partitioning nach einem bestimmten Schlüsselwert auf die einzelnen Prozessoren bzw. Platten oder Dateien aufgeteilt und dort gespeichert. Das Einfügen der Daten kann im Vergleich zum seriellen System parallel und damit schneller geschehen. Eine mögliche Engstelle des I/O-Systems des seriellen Systems wird so vermieden. → Speedup

- Query
  Die Anfragen mehrerer Nutzer, die gleichzeitig auf einem DBS mit nur einem Prozessor arbeiten, werden seriell abgearbeitet. Während die Anfrage eines Nutzers bearbeitet werden müssen die anderen warten. Datenbanknutzer sind schnell frustriert, da sie die anderen Anfragen nicht sehen bzw. nicht so wichtig wie die eigenen einschätzen.
  Beim Parallelen System hingegen können mehrere Anfragen parallel bearbeitet werden, ev. stehen mehrere Zugänge zum DBS zur Verfügung. Mögliche I/O- Engpässe werden umgangen.
  → Speedup

- Scan
  Vorpartitionierte Daten können parallel gescant werden, d.h. jeder Prozessor such auf seiner Partition "seinen" Teil der Daten nach dem relevanten Wert. Bei optimaler Lastenverteilung kann bei n Prozessoren ein Speedup von 1/n gegenüber der Ausführung auf einem seriellen System erreicht werden. Die Teilergebnisse der einzelnen Prozessoren werden zusammengefügt und dem Anwender als ein Ergebnis, ganz als ob nur ein Prozessor an der Auswetung der Daten beteiligt gewesen wäre, zurückgegeben.

- Join
  Bei einem Join werden Attribute verschiedener Relationen miteinander verglichen. Dies kann auf verschiedene Weisen erfolgen:

  a) Beim Parallel-Nested-Loop Join ist mindestens eine Relation schon nach dem Schlüsselattribut partitioniert. Die zweite Relation wird nach dem gleichen Schlüsselattribut auf die Prozessoren verteilt, so daß die entsprechenden Attribute parallel auf den einzelnen Prozessoren miteinander verglichen werden können. Im seriellen Fall muß jeder Wert des Attributes der einen Relation mit jedem Wert des Attributes der anderen Relation verglichen werden.

  b) Beim parallel hash Join werden beide Relationen zur Laufzeit mit Hilfe einer hash-Funktion bezüglich des relevanten Attributes auf die zur Verfügung stehenden Prozessoren verteilt.So können wieder die beiden Relationen bezüglich des relevanten Attributes auf ihren Partitionen miteinander verglichen werden und die Teilergebnisse zu einem Gesamtergebnis zusammengefügt und als Ergebnis übergeben werden. Auch hier ist trotz der notwendigen Partitionierung ein wesentlicher Speedup gegenüber der seriellen Bearbeitung zu erzielen.

  ⇨ vgl. Folie 10

Folie 7: Ebenen der Parallelisierung

Parallelität läßt sich auf verschiedenen Ebenen eines DBS realisieren. Es kann dabei eine Einteilung in folgende Ebenen vorgenommen werden:

- Transaktionsparallelität (Stufe 0)
  → seit mehreren Jahrzehnten üblich
  → Parallelität auf der Ebene von Transaktionen (Transaktion: Menge an Anforderungen, Anfragen, z.B. Auskunsftsersuche, Buchungsaufträge, Einzahlungen...)
  → Einzelne Transaktionen sind voneinander unabhängig
  → "Parallelität" durch time-sharing auf BS-Ebene
  → Diese "Parallelität" ist durch das DBS nicht bestimmbar, nur verwaltbar
- Parallelität zwischen DML-Statements in derselben Transaktion (Stufe 1)
  → häufig keine Datenflußabhängigkeiten bzgl. DBObjekten, Parallelisierung einfach machbar
  → Schwierigkeiten der Realisierung mit "nicht-parallelen Programmiersprachen" wie z.b. Cobol
  → Parallelisierungsgrad bestimmt durch Datenabhängigkeit.
  → Parallelität von 1-10 SQL-Befehlen pro Transaktion üblich.
- Parallele Implementierung eines DML-Statements (Stufe 2)
  → Kern heutiger DB-Parallelität
  → Stufe 2 und Stufe 3 zusammen sind die Schichten, die Speedup bewirken können
  → SQL-Befehle sind komplexe Befehle, z.B. wegen WHERE-, UNION-Klauseln, das bietet Möglichkeiten zur Parallelisierung.
- Parallelität auf Partitionen (Stufe 3)
  → Datenparallele Ausführung der Basisoperationen
  → Stufe 0-2: Funktionsparallelität, Stufe 3: Datenparallelität
  → bei relationalen DBS theoretisch einn Tupel je Prozessor
  → Faktoren der Parallelität auf dieser Stufe:
    → Verteilung auf Platten: Gleichverteilung der Last?
    → Anzahl der Platten, Prozessoren
  → bei Gleichverteilung über n Platten und n Prozessoren ist Speedup von n möglich
  → weitere Verfeinerungsmöglichkeit: Paralleles Lesen und Schreiben durch Nutzung von RAID-Plattensystemen (Redundant Array of Inexpensive Disks)

- Retrieval Ssystem
  → Daten sind auf einen bestimmten Bereich menschlichen Handelns abgestimmt
  → Daten sind bestimmtem Benutzerkreis oder der Allgemeinheit zugänglich
  → Datenpflege durch zentrale, authorisierte Stelle
  → strukturierte oder unstrukturierte Systeme
  → Auskunfts oder Dokumentationssysteme
- DBS für spezifische Anwendungen
  → Auswertung von Daten mit Hilfe statistischer oder sonstiger Verfahren
  → z.B. Auswertung medizinischer Daten zur Erstellung einer medizinisch Diagnostischen Datenbank

Folie 5: Effekte Paralleler Verarbeitung

Durch parallele Verarbeitung kann eine Leistungssteigerung in zweierlei hinsicht bewirken:

- Scaleup: In gleicher Zeit können durch ein parallele System bedeutend größere Probleme bearbeitet werden als durch das Single-Prozessor System. Gegebene Anwendungen können auf immer größeren Datenbanken mit linearem Durchsatz verarbeitet werden, ohne daß sich die Antwortzeiten interaktiver Abfragen erhöhen.
- Speedup: Für gleichgroße Probleme benötigt das parallele System bedeutend weniger Zeit zur Ausführung als das Single-Prozessor System. So können komplexe Danenbankanfragen in akzeptabler Zeit bearbeitet werden.

Folie 6: Aspekte paralleler Verarbeitung

Um komplexe Aufgaben parallel abarbeiten zu können, muß versucht werden, die komplexe Gesamtaufgabe in kleinere, voneinander unabhängige Teilaufgaben aufzuteilen. Es gibt verschiedene Aspekte, unter denen man eine Verteilung vornehmen kann:

Möglichkeiten der Verteilung:

Wie verteilen?
→ statisch: Die Daten werden **von Anfang an fest** nach bestimmten Schlüsselwerten aufgeteilt.
→ dynamisch: Die Daten werden **zur Laufzeit** aufgeteilt. Sie werden dem Problem entsprechend repartitioniert.

Was verteilen?
→ Daten: Die **Daten** werden den einzelnen Prozessoren statisch oder dynamisch zugeteilt.
→ Services: **Befehle** werden in Unterbefehle oder **Aufgaben** in Teilaufgaben zerlegt und den einzelnen Prozessoren zugeteilt.

Erläuterungen zu den Folien

Notwendigkeit von Datenbanksystemen

Die Information auf Ebene der Unternehmensführung entwickelt sich immer mehr zum "Produkt der Zukunft". Daraus ergibt sich die Notwendikeit eines globalen Informationsmanagements. Ein globales Informationsmanagement ist nötig

→ für dieBeibehaltung der Wettbewerbsfähigkeit
→ für kürzere Modellzykluszeiten
→ für Modularität und Steuerung der Produktion
→ für flexible Fertigungs- und Produktionstechniken
→ für ausgeklügelte Transportlogik

⇨ Notwendigkeit von elektronischen Hilfsmitteln u.a. Datenbank-Systeme zur effizienten Verwaltung und Nutzung von Informationen.

⇨ Komplexe Informationen & Zusammenhänge fordern komplexe Informationssysteme.

Folie 4: Klassifizierung von Datenbanksystemen

Die folgende Klassifizierung versucht, Datenbanksysteme nach dem Schwerpunkt ihrer Nutzung in verschiedene Klassen einzuteilen:

- Storrage and Retrieval Systems
  → verbreiteste System
  → Daten zentral gespeichert
  → Daten dezentral gepflegt
  → strukturierte Datensätze
  → Datenbankmodell ist von spezifischer Aufgabe abhängig
- Storrage System
  → strukturierte Datensätze
  → sehr hohe Datenschutzanforderungen
  → Pflege der Daten unter Umständen nur von zentraler Stelle
  → Abrufbarkeit ggf. nach Benutzerklassen unterteilt
  → Grenze zwischen normalem DBS und Auskunftssystem
  → in sich sehr homogener Datenbestand
  → Datenbestand unterliegt relativ geringen Änderungen

Seminar Parallelrechner

- Parallelität in Datenbank-Systemen -

Vorwort

Die vorliegenden Folien und die Erläuterungen geben einen allgemeinen Überblick über Möglichkeiten zur Parallelisierung in Datanbanksystemen und zeigen die Realisation am Beispiel von Oracle 7.0 zusammen mit Query Decomposer der Firma KSR sowie am Beispiel von Oracle 7.1.
Zuerst werden Grundsätzliche Aspekte und Effekte paralleler Datenbanken dargestellt und verschiedene Ebenen der Parallelisierung bei Datenbanken betrachtet. Im Anschluß daran werden grundlegende Datenbankoperationen im Vergleich serieller und paralleler Abarbeitung vorgestellt und kurz einige parallele Architekturen erläutert. Das Ende des ersten Teiles bildet die Auflistung von Beispielen aktueller Datenbanksysteme mit parallelen Optionen.
Im zweiten Teil werden Funktionsweise und Benutzung von Oracle 7.0 in Verbindung mit Query Decomposer erläutert und einige Leistungsdaten hierzu zitiert. Schließlich werden mögliche Hardwareumgebungen, Funktionsweise, Benutzung, leistungsbeeinflussende Faktoren sowie Optimierungsmöglichkeiten von Oracle 7 Release 7.1 dargestellt.

Technische Hinweise

Die folgenden Folien und Erläuterungen wurden für einen Seminarvortrag zu dem Thema »Parallelität in Datanbank-Systemen« erstellt. Er wurde am 05.02.1996 im Rahmen der Seminarveranstaltung Parallelrechner des Wintersemesters 1995/96 am Rechenzentrum der Universität Mannheim gehalten.
Die beiliegenden Folien dienten als Grundlage des Seminarvortrages. Die folgenden Erläuterungen zu den Folien sollen es dem interessierten Leser ermöglichen - auch ohne den Seminarvortrag gehört zu haben - den Inhalt des Vortrages nachvollziehen zu können.
Die Erläuterungen sind als erklärende und ergänzende Informationen zu den Folien gedacht. Einige Folien sind selbsterklärend oder bedürfen keiner ergänzenden Erläuterung. Weiterführende Informationen kann der interessierte Leser der angegebenen Literatur entnehmen, die auch die Grundlage des Seminarvortrages bildete.

Optimierung eines DBS

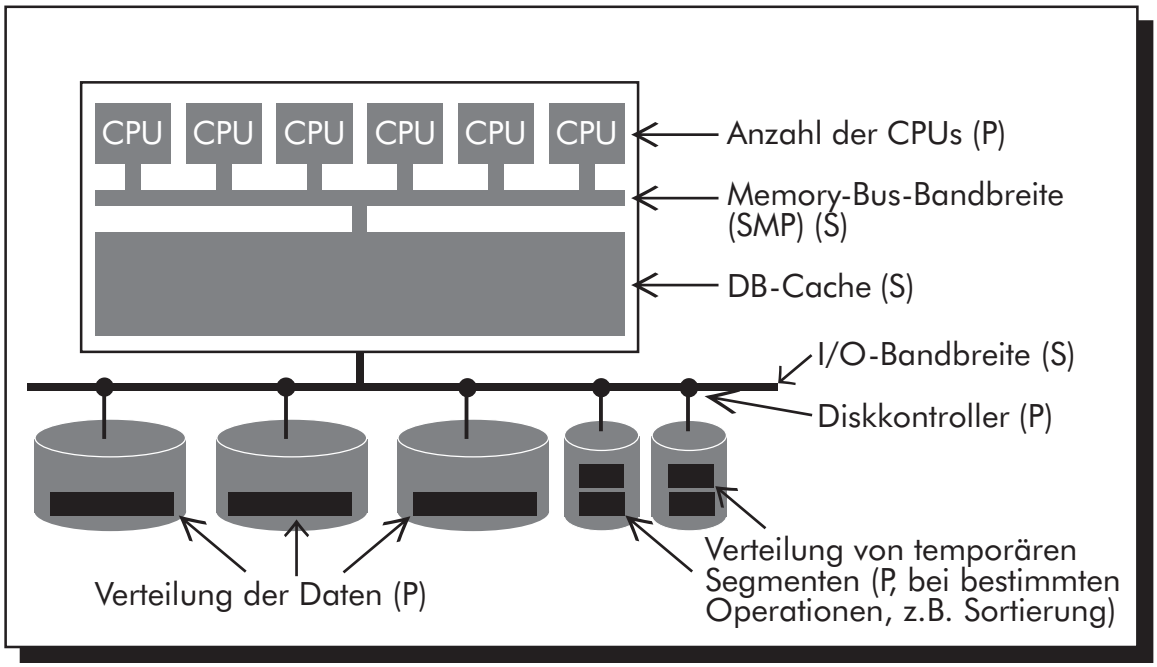
Aspekte, relevante Fragen:

- nicht alleine die Anzahl der CPUs ist relevant
  I/O-Bandbreite, Memorybus-Bandbreite
  Lastenverteilung auf Platten
- alle große Tabellen und Langläufertabellen sollten nicht mit Standardwerten parallelisiert werden
- wie ausgelastet sind die einzelnen CPUs ?
- welches sind die Platten mit den meisten I/Os ?
- sind bereits I/O-Engpässe bekannt ?
- welche DB-Objekte sind auf den stark belasteten Platten abgelegt ?

Lösungsmöglichkeiten zur weiteren Optimierung:

- Disk Striping (physisch)
- Logische Striping (Tablespace Striping)
- Striping eines temporären Tabelspaces
- Verringerung des sort_area_size-Parameters

Einflußfaktoren auf die Parallelisierbarkeit



Theoretische Formel:

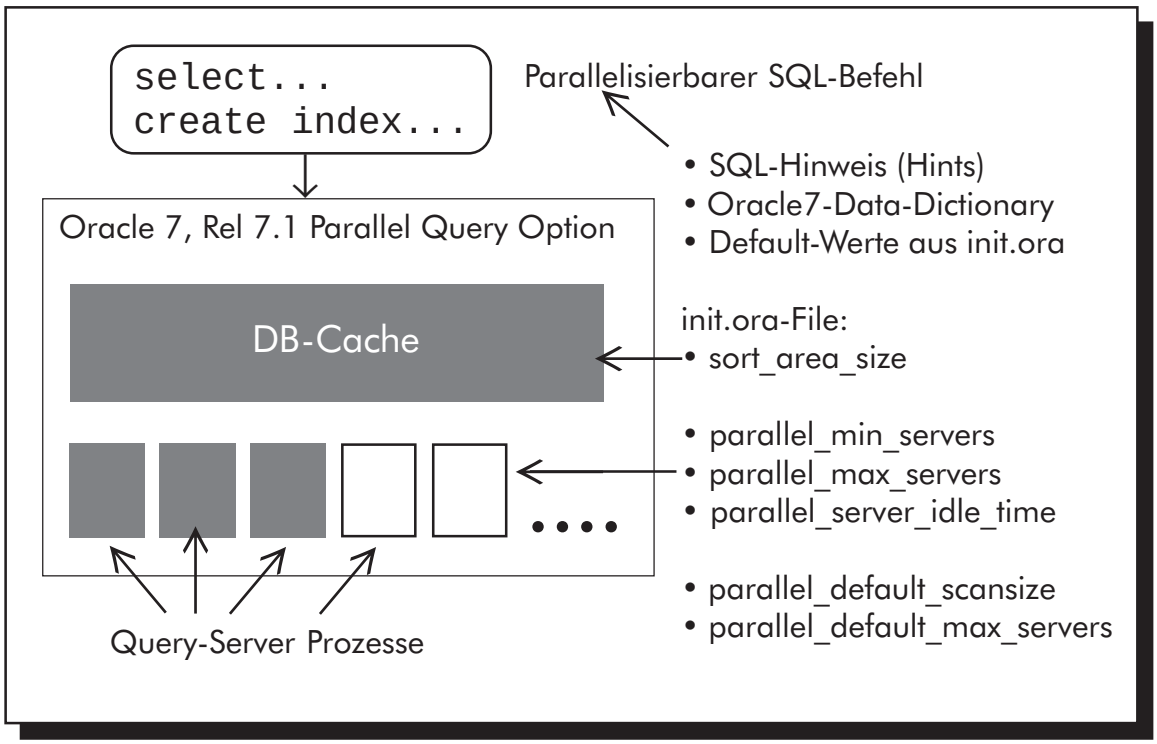
$$t_{parallel} = 1/n * t_{seriell}$$
$$d_{parallel} = n * d_{seriell}$$

Pragmatische Formel:

$$t_{parallel} = t_{seriell} / (n * s)$$
$$d_{parallel} = (n * s) * d_{seriell}$$

n	Anzahl der CPUs	t	Zeit
s	Skalierungsfaktor	d	Durchsatz

Einsatz der Parallelen Query Technologie



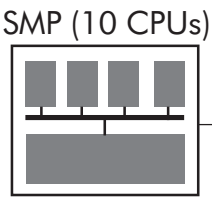
Beispiele Paralleler Anweisungen:

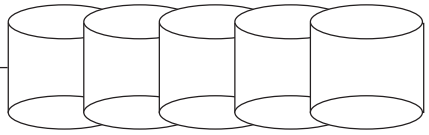
```
create table <table name> (....) parallel <n>;
create table ta (...) parallel 2;
create table tb (...) nonparallel;
select /*+full (ta) parallel (ta, 5)*/ col1, col2
from ta
order by col2;

alter table ta parallel 10;
alter table td nonparallel;
parallel (ta, 8);
nonparallel (ta);

create table tc (...);
```

Nutzung der Parallelen-Query-Technologie





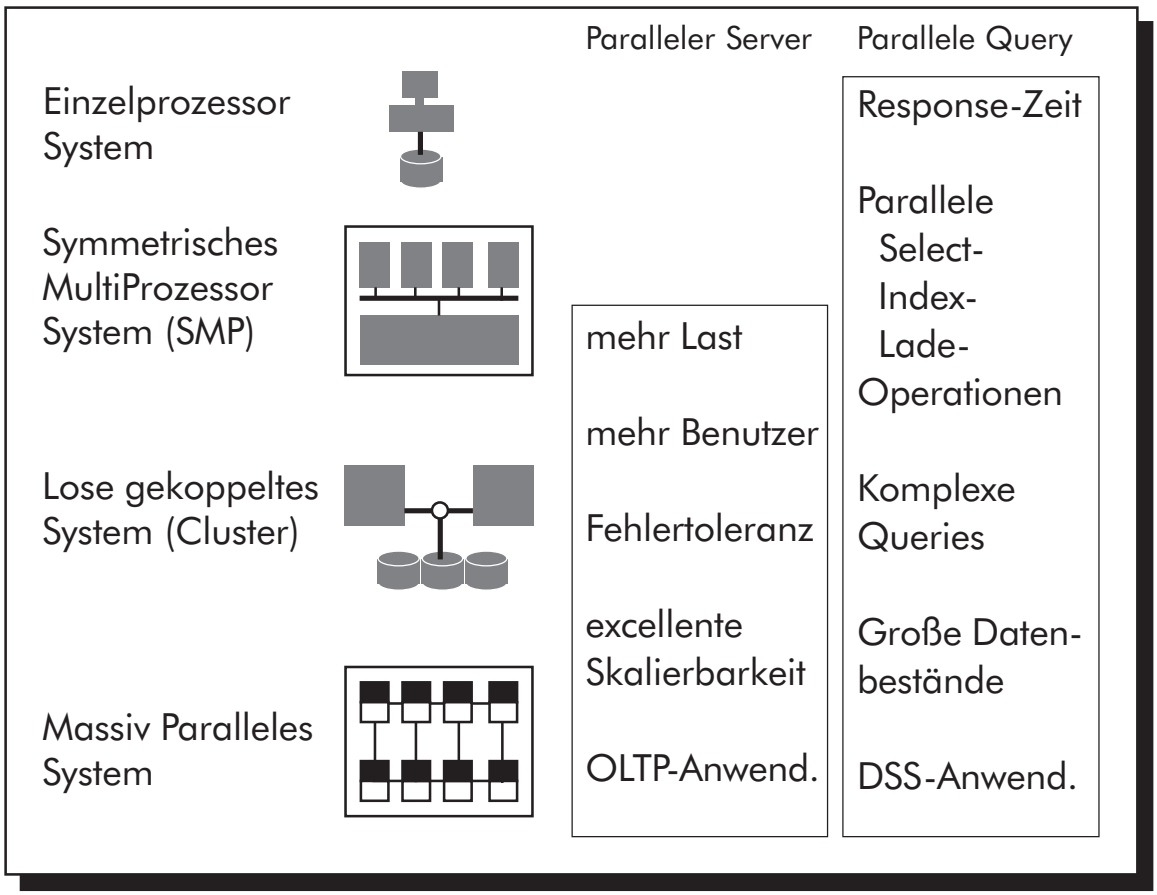
select *
from Mill_10_rows

			relative Antwortzeit
Oracle 7, Release 7.0	full table scan	1 CPU 100% 9 CPUs 0%	1 = 8/8
Oracle 7, Release 7.1 alter table Mill_10_rows nonparallel	full table scan	1 CPU 100% 9 CPUs 0%	1 = 8/8
Oracle 7, Release 7.1 alter table Mill_10_rows parallel 2	full table scan	2 CPU 100% 8 CPUs 0%	0.625 = 5/8
Oracle 7, Release 7.1 alter table Mill_10_rows parallel 10	full table scan	10 CPU 100% 0 CPUs 0%	0.125 = 1/8

Parallele Operationen in Oracle 7.1

- Queries
  - » full table scan
  - » join (sort/merge, Nested Loop)
  - » group by
  - » order by
  - » set Operationen (union, intersect, minus)
  - » distinct
  - » where - Klausel in UPDATE, DELETE
  - » sub-queries bei INSERT und CREATE TABLE Befehlen
- Index-Erzeugung
- Lade-Operationen

Hardware-Voraussetzungen für Oracle 7.1



Neuerungen von Oracle 7, Release 7.1

Neuerungen:

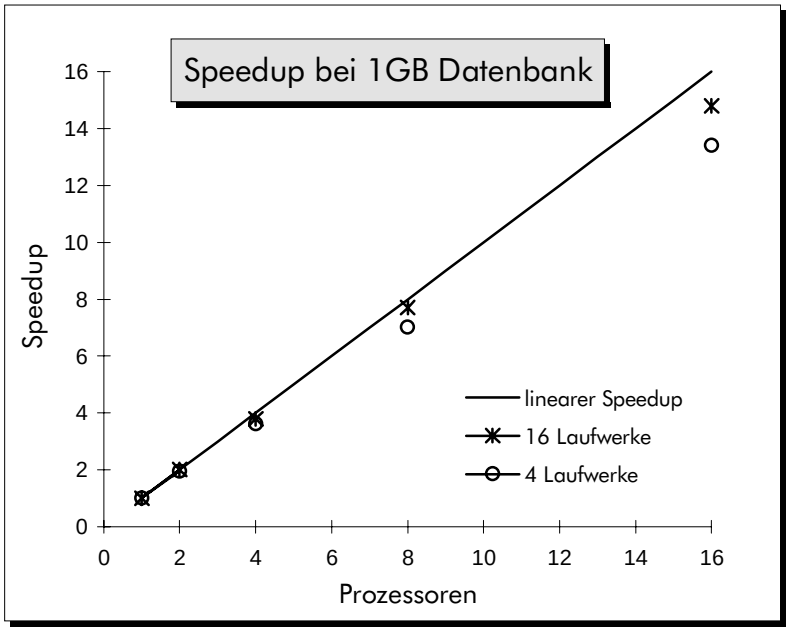
- Einführung der Parallelen Query Technologie
- Einführung des Parallelen Recovery
- Einführung von Read-Only-Tablespaces

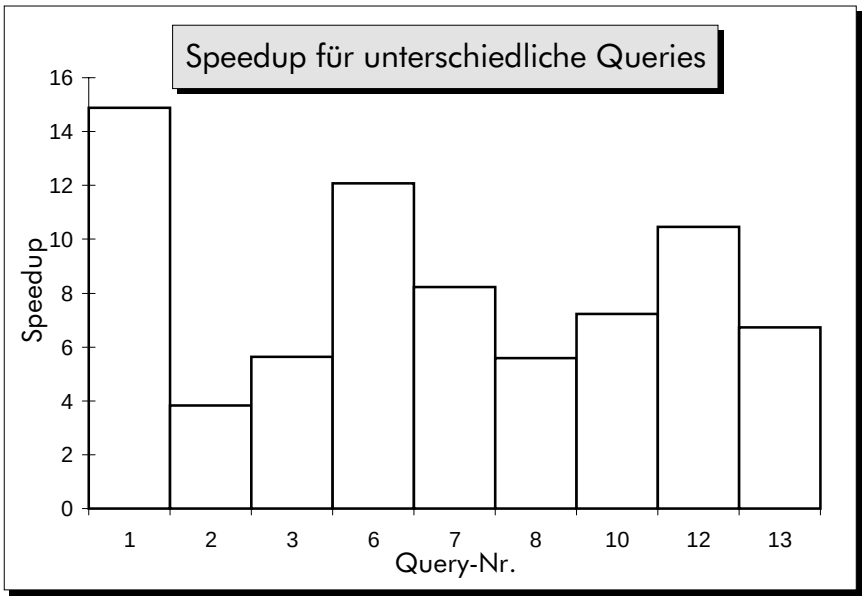
Ziele:

- Erhöhung der OLTP-Performance
- größere Last und größere Benutzerzahlen ohne Abfallen der Performance

Oracle 7.0 & QD - Realisation auf der KSR1

vgl.: Oracle Datenbankkopplung SNI BS2000 - KSR1
Kredel, Schumacher (eds.), 7/94, RUM 37/94



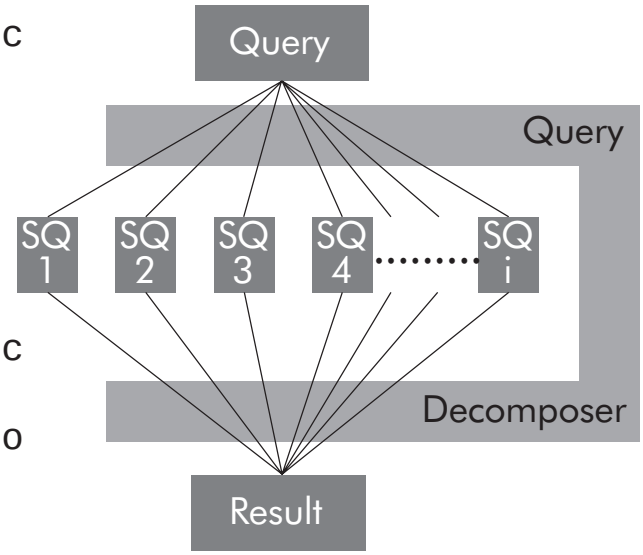


Beispiel für die Zerlegung einer Query in Subqueries

```
SELECT empname, empno, deptloc
FROM emp, dep
WHERE emp.depno = dept.deptno
AND emp.title = 'MTS'
```

Query | Decomposer ↓

```
SELECT empname, empno, deptloc
FROM emp, dept
WHERE emp.deptno = dept.deptno
AND emp.title = 'MTS'
AND emp.rowid >= '0.0.i'
AND enp.rowid < '0.0.i+1'
```



Beispiel einer Anfrage mit Aggregatfunktion

```
SELECT deptno, AVG(salary)
FROM emp GROUP BY deptno
```

Query | Decomposer ↓

```
SELECT deptno, SUM(salary), COUNT(salary)
FROM emp
WHERE emprowid >= '0.0.i'
AND emp.rowid < '0.0.i+1'
GROUP BY deptno
```

Query | Decomposer ↓

```
SELECT (SUM(sum_col)/(SUM(count_col))
FROM temporary_table
```

Oracle 7.0 und Query Decomposer auf der KSR

KSR:

- Skalierbares massiv paralleles System (MPP)
- Skalierbarkeit bis hunderte von Prozessoren
- Virtual shared memory
- Beispiel KSR1/2: 32/64 Prozesoren, 20/40 Mhz, 40/80 Mips bzw. 40/80 Mflops, 1/2 GB Haupt- und 10/20 GB Plattenspeicher

Query Decomposer (QD):

- Voraussetzung:
  - Partitionierung der Tablespaces auf verschiedene Dateien oder Platten
- Funktionsweise:
  - Zerlegung von parallelisierbaren Queries in Subqueries
  - Weitergabe der Subqueries an Oracle
  - je Subquery eigener Prozeß
  - Teilergebnisse werden von QD wieder zusammengesetzt
  - nichtparallelisierbare Queries werden behandelt, als ob QD nicht vorhanden wäre werden

Parallelität in heutigen DBS

Problem:
- kaum Herstellerangaben, was parellel ist und wie dies umgesetzt wird
- schnelle Release-Wechsel

Beispiele (Stand 2/95):
- Teradata von NCR (bzw. AT&T)
  - Platten mit speziellen Controlern
  - Controler sind hardwaremäßig als Binarbaum verknüpft

- NonStopSQL von Tandem Computers
  - alle Operationen können parallel ausgeführt werden
  - beliebig tiefe Pipelines
  - höchste Leistung nach TPC-C-Benchmark
    (110 Prozessoren, 800 GB DB | 16 Prozessoren, 94 MB/s)

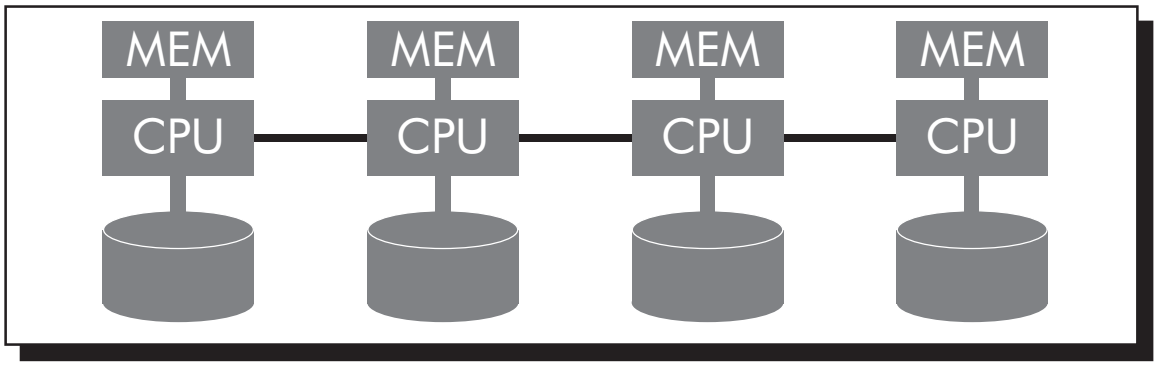
- Oracle
  - ähnlich NonStopSQL
  - beschränkte Pipelines
    (44 MB/s bei 20 Pentium Prozessoren)

- IBM / DB2 für MVS oder RS6000-Systeme
  - nur Scan parallelisiert (2/95)

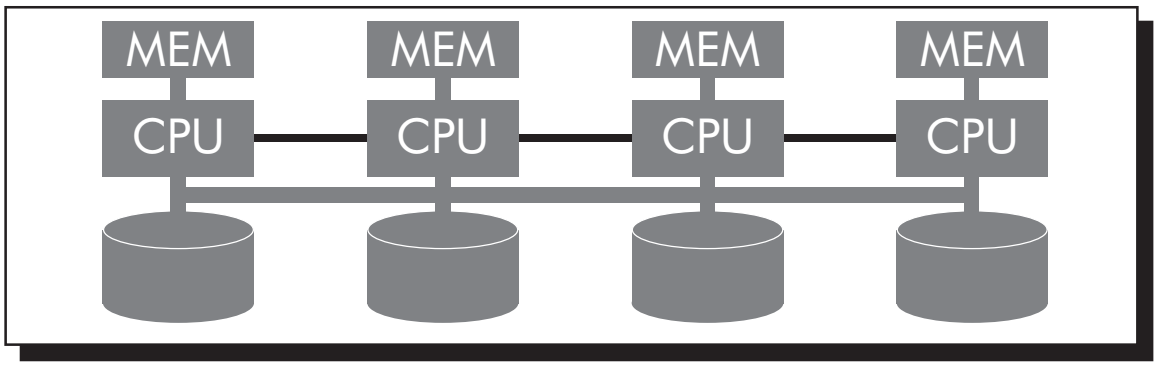
- Informix
  - Leistungen wie NonStopSQL

Rechnerarchitekturen für parallele DBS

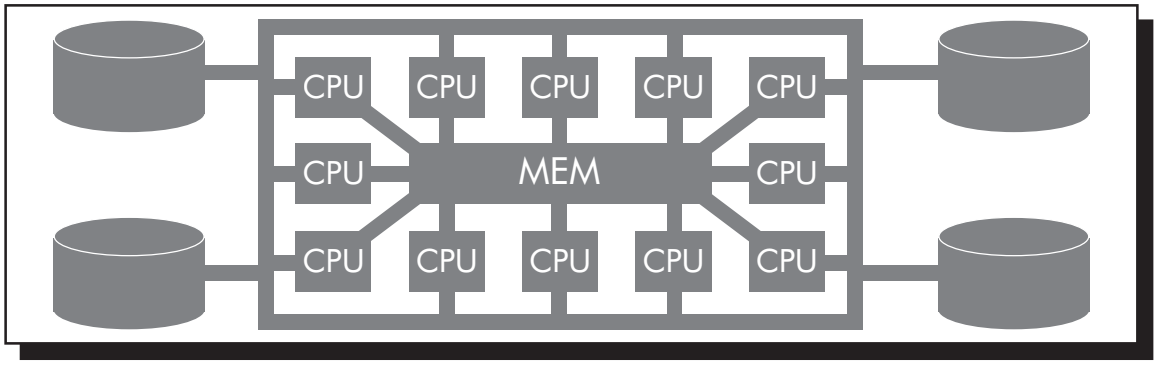
shared nothing



shared disk



shared everything



Parallele Algorithmen für wichtige DB-Operationen

a) Paralleler Verbund

(i) Paralleler Verbund durch Sortieren

- beide Relationen sind vorsortiert
- mindestens eine Relation ist nicht vorsortiert

(ii) Paralleler Verbund durch Hashing

Ziel: Es soll eine Verteilung erreicht werden, so daß
- alle nutzbaren Prozessoren eingesetzt
- jeder Prozessor gleichbelastet (Lastbalancierung)

werden.

Hashing:
- Bildephase (Verteilung der ersten Relation gemäß Hashfunktion h)
- Probephase (Verteilung der zweiten Relation gemäß Hashfunktion h, Vergleich und ggf. Verbund)

nur für Verbund mit "="-Vergleich geeignet

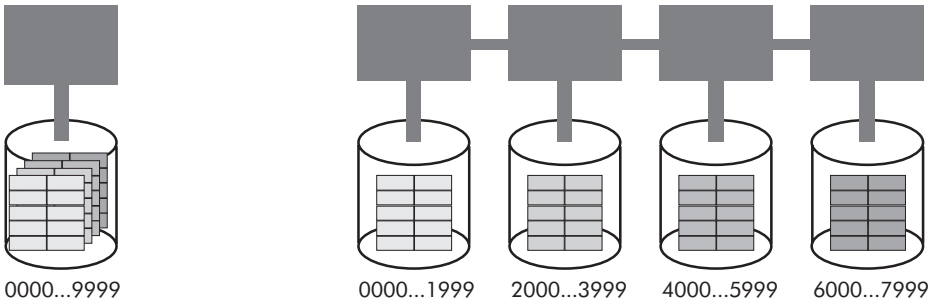
b) Parallele Indexerzeugung

- Vermerk des Anfangszeitpunktes in Protokolldatei
- kein Sperren der Relation während der Indexerstellung
- nach Ende der Indexerstellung: Sperren der Relation und Korrektur der seit Anfangszeitpunkt geänderten, ergänzten Einträge

Scan

seriell parallel

SELECT ArtikelNr FROM ArtikelStamm
WHERE ArtikelNr > 1000

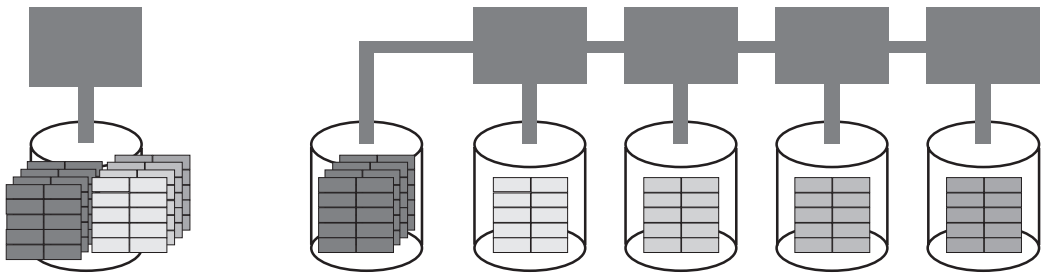


Join

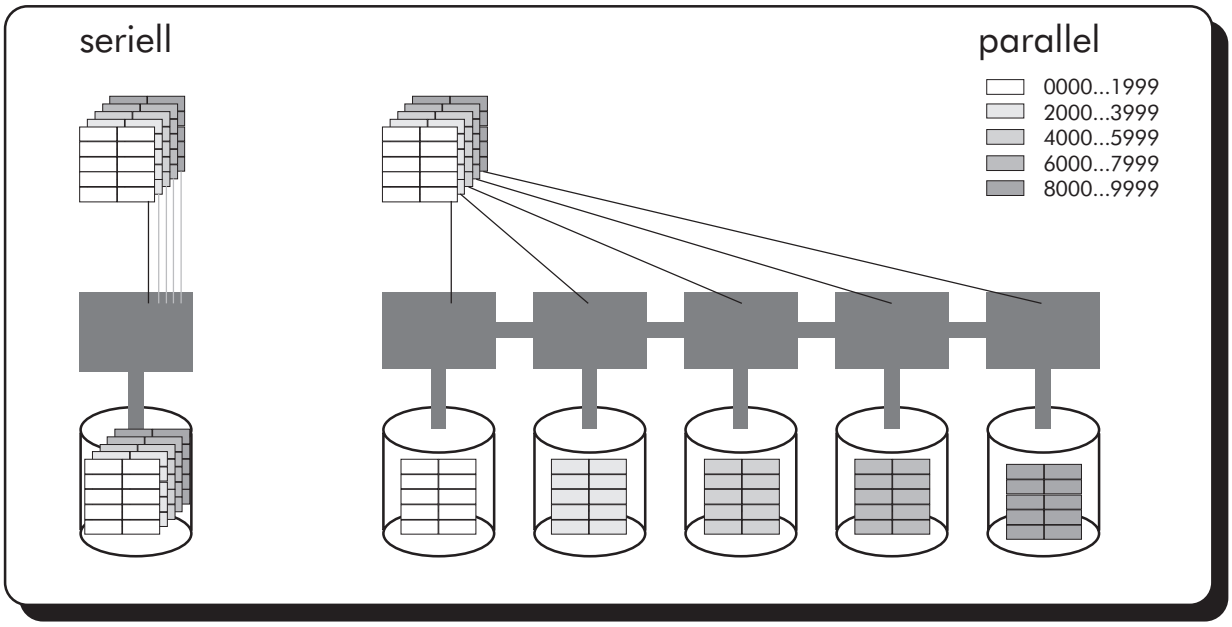
- PARALLEL NESTED LOOP JOIN
- PARALLEL HASH JOIN

seriell parallel

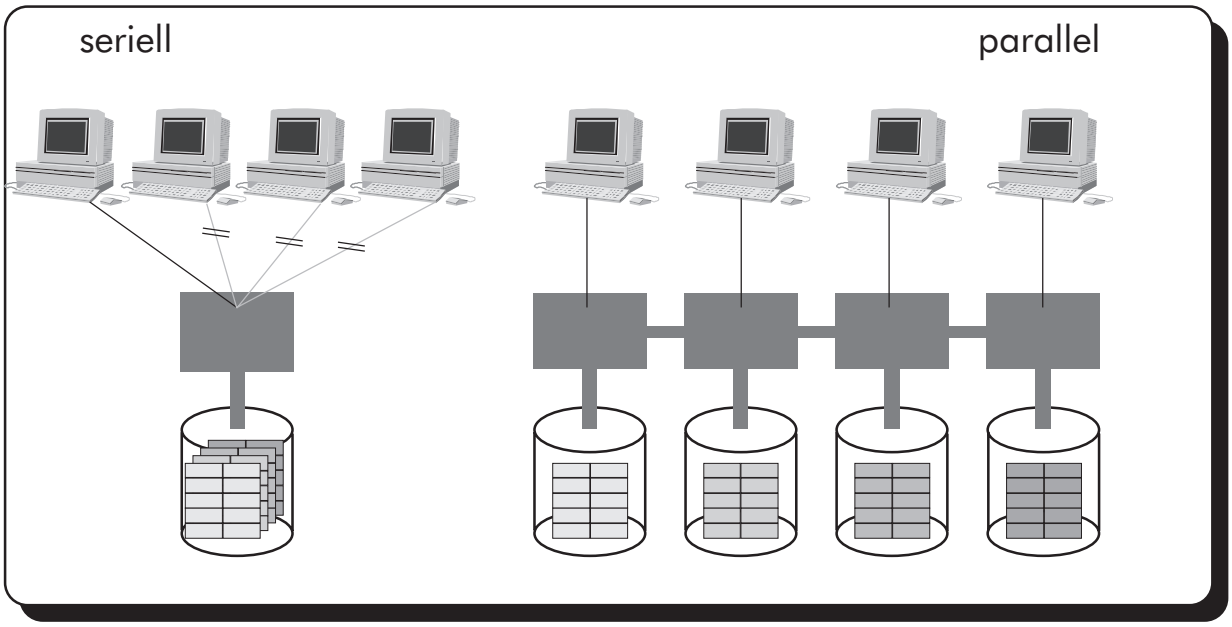
SELECT ArtikelStamm.ArtikelNr, ArtikelStamm.ArtikelBezeichnung
FROM ArtikelStamm, Bestellung
WHERE ArtikelStamm.ArtikelNr = Bestellung.ArtikelNr.



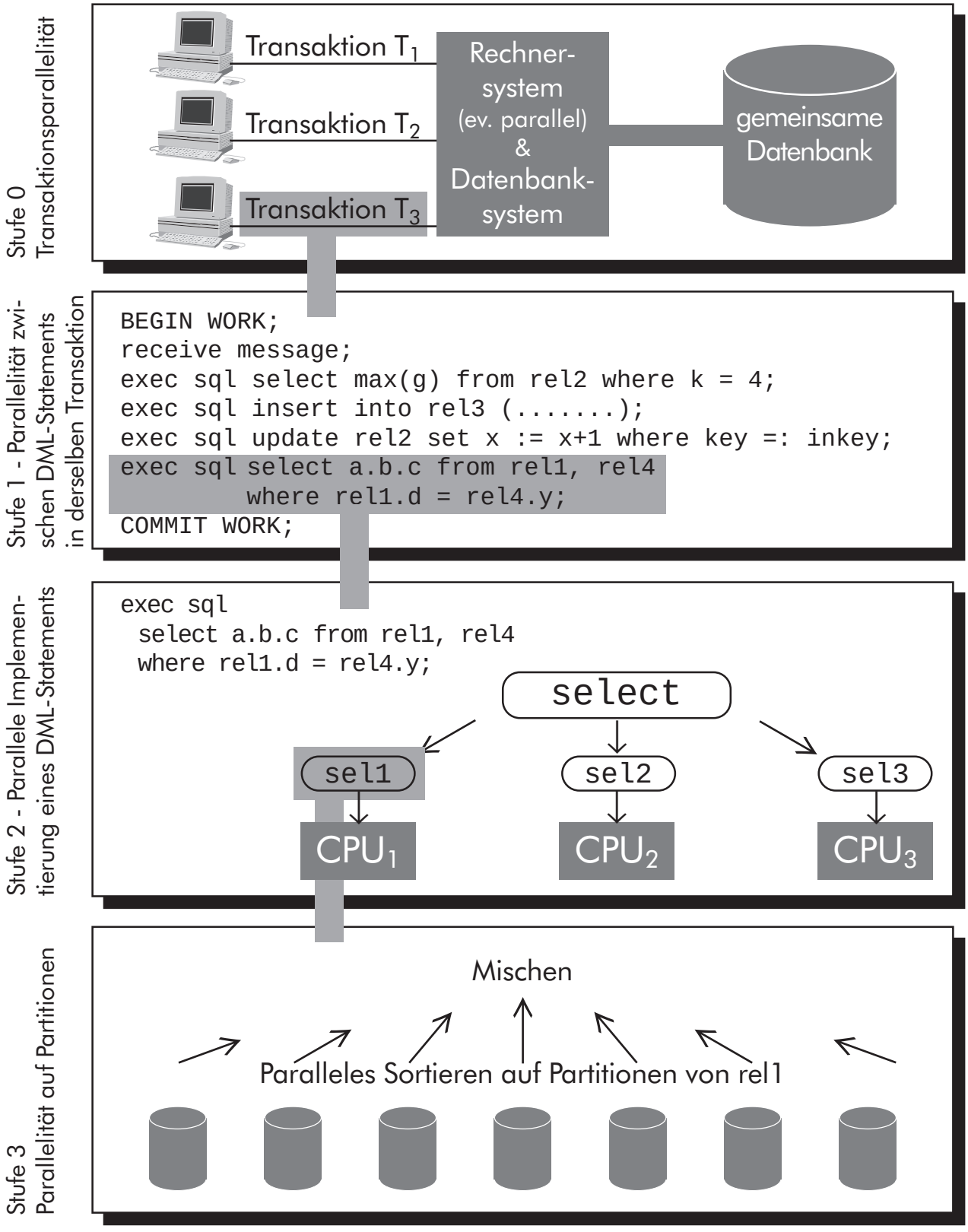
Insert (Key Range Partitioning)



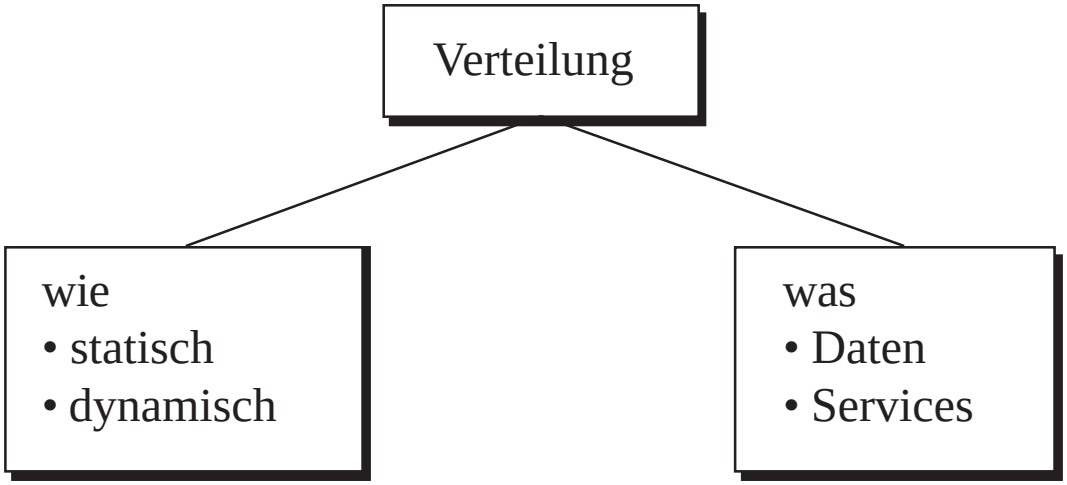
Query



Ebenen der Parallelisierung

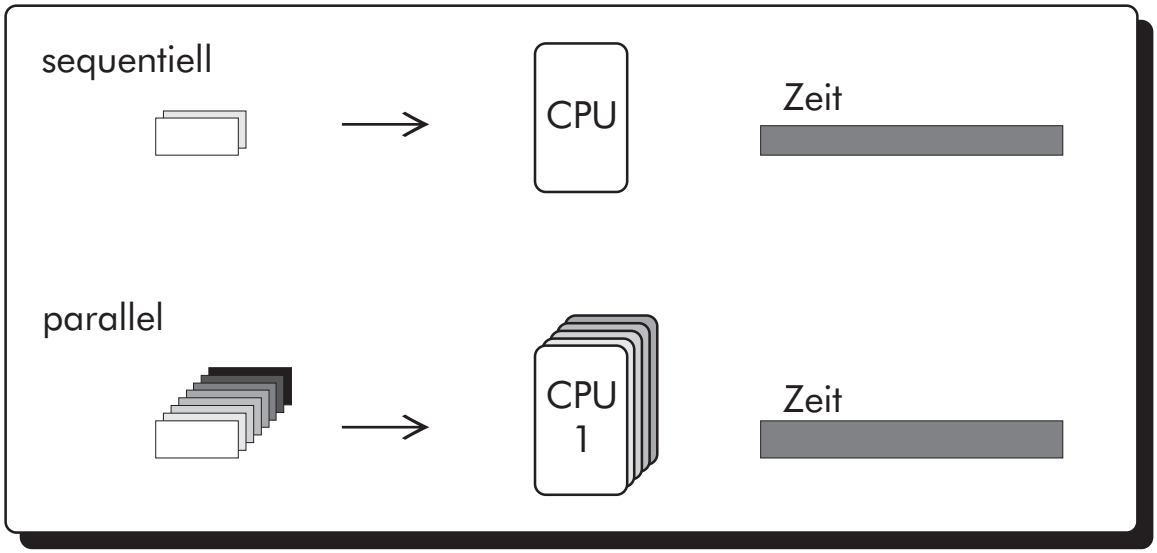


Aspekte paralleler Verarbeitung

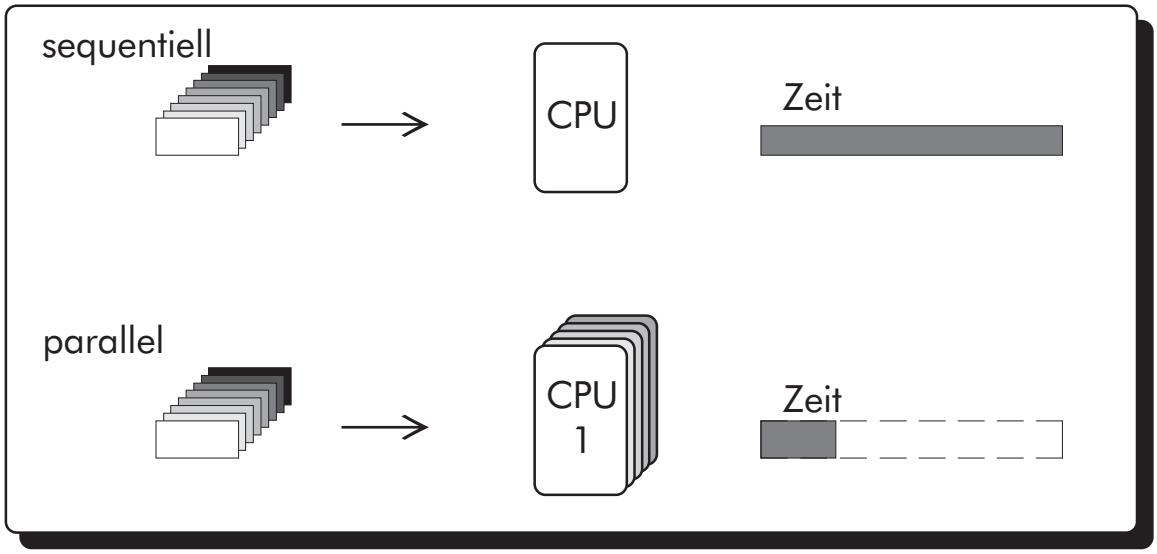


Effekte paralleler Verarbeitung

- Scaleup



- Speedup



Klassifizierung von Datenbanksystemen

- Storrage and Retrieval Systems
  Beispiel:
  Datenbanksystem zur Verwaltung von Anlagen, Werkstoffen, Lagerbeständen, Personal, Kunden, Lieferanten usw.

- Storrage System
  Beispiel:
  Einwohnermeldeamt

- Retrieval System
  Beispiel:
  Verkehrsverbindungen der Bundesbahn, Bibliotheken, JURIS, DATEV, Bildschirmtext

- DBS für spezifische Auswertungen
  Beispiel:
  Medizinisch-diagnostische Datenbank

Überblick:

- Notwendigkeit von Datenbanksystemen
- Klassifizierung von Databanksystemen
- Notwendigkeit und Einsatzgebiete von Parallelisierung in Datenbanksystemen
- Effekte paralleler Verarbeitung
- Aspekte paralleler Verarbeitung
- Ebenen der Parallelisierung
- Beispiele wichtiger DB-Operationen im Vergleich
- Parallele Algorithmen für wichtige DB-Operationen
- Rechnerarchitekturen für parallele DBS
- Parallelität in heutigen DBS
- Oracle 7.0 und Query Decomposer auf der KSR
- Neuerungen in Oracle 7, Release 7.1
- Parallele Query Technologie von Oracle
  - Hardwarevoraussetzungen
  - Nutzung der Parallelen Query Technologie
  - Einsatz der Parallelen Query Technologie
  - Einflußfaktoren auf die Parallelisierbarkeit
  - Optimierung eines DBS

Überblick:

- Notwendigkeit und Klassifizierung Datenbanksystemen
- Parallele Verarbeitung in DBS
- Oracle 7.0 und Query Decomposer auf der KSR
- Parallele Query Technologie von Oracle

Rechenzentrum
der Universität Mannheim

Seminar Parallelrechner

Parallelität in Datenbank-Systemen

Aktuelle Entwicklung auf dem Gebiet paralleler Datenbanken, im speziellen Oracle 7, Release 7.1

Klaus Zimmermann

Wirtschaftsinformatik 7. Semester

Betreuer:
Dr. Heinz Kredel
Rechenzentrum der Universität Mannheim