

# Choosing Multiprocessor System Architecture for Parallel Database Systems

---

**Leonid B. Sokolinsky**  
**Chelyabinsk State University**  
**Russian Fed.**  
sokolinsky@acm.org

# Contents

---

- Objectives for Parallel Database Systems
- Stonebraker's classification
- Analysis of Parallel Architectures
- Hierarchical Architectures

# Objectives

---

- Scalability
- Load Balancing
- Reliability
- Data Availability
- Simplicity for programming

# Scalability

---

- assumes the ability of an N-times larger system to perform an N-times larger job in the same elapsed time as the original system

# Load Balancing

---

- **assumes the ability to equally divide the load among the processors while minimizing the coordination and synchronization overhead**

# Reliability

---

- **assumes that any part of the stored data can be still accessed after any hardware component failure**

# Data Availability

---

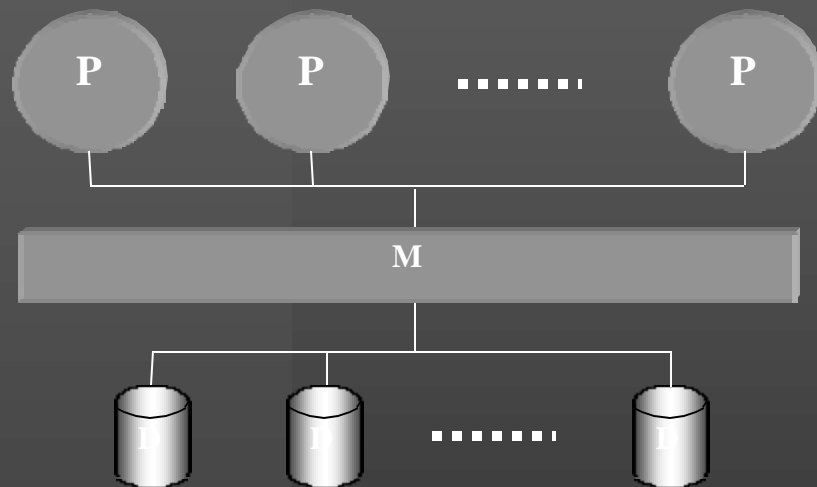
- assumes that in the event of a disk failure, the copy of the data may still be available on one or more disks at ***no additional cost*** (unlike log-based recovery)

# Stonebraker's classification

---

- **Shared-Everything (Shared-Memory)**
- **Shared-Disk**
- **Shared-Nothing**

# Shared-Everything



**A collection of processors is attached to the memory bus and I/O bus, and each can access a common shared memory and a common shared disk system**

# Shared-Everything (SE)

SE architecture has **low scalability** because the memory bus becomes a bottleneck in the system with more than 30 processors.

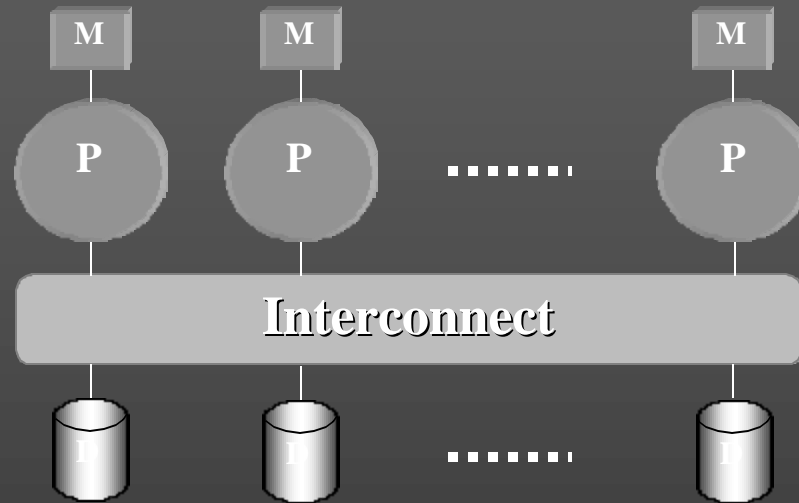
SE architecture allows **good load balancing** because each processor has access to all memory modules and all disks.

SE architecture demonstrates **low reliability** because a failure of any hardware component leads to the whole system failure.

SE architecture demonstrates **low data availability** because it uses a log-based recovery.

SE architecture is relatively **simple for programming**.

# Shared-Disk



**The processors with private memory have access to any disk through the specialized interconnect**

# Shared-Disk (SD)

SD architecture has **limited scalability** because the access to shared disks might result in a bottleneck due to limited bus capacity.

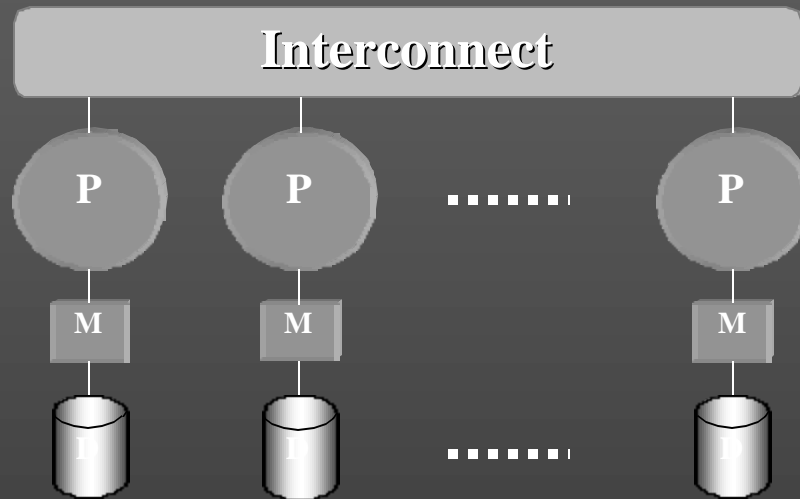
SD architecture allows **fair load balancing** because each processor has access to all disks.

SD architecture demonstrates **fair reliability** because a failure of I/O bus leads to the whole system failure.

SD architecture demonstrates **good data availability** because the copy of the lost data may still be available on other disks at no excessive overhead.

SD architecture is **complex for programming** because we have to maintain a buffer coherence control mechanism and a global locking table.

# Shared-Nothing



**The processors with private memory and disk are connected on a specialized interconnect**

# Shared-Nothing (SN)

SN architecture has **good scalability**.

In SN architecture, **load balancing is difficult** because it relies on the effectiveness of database partitioning for query workloads.

SN architecture demonstrates **good reliability**.

SN architecture demonstrates **fair data availability** because the copy of the lost data may still be available on other disks at some overhead.

SN architecture is **not simple for programming** because we have to implement distributed database functions assuming large numbers of nodes.

# Summary Table

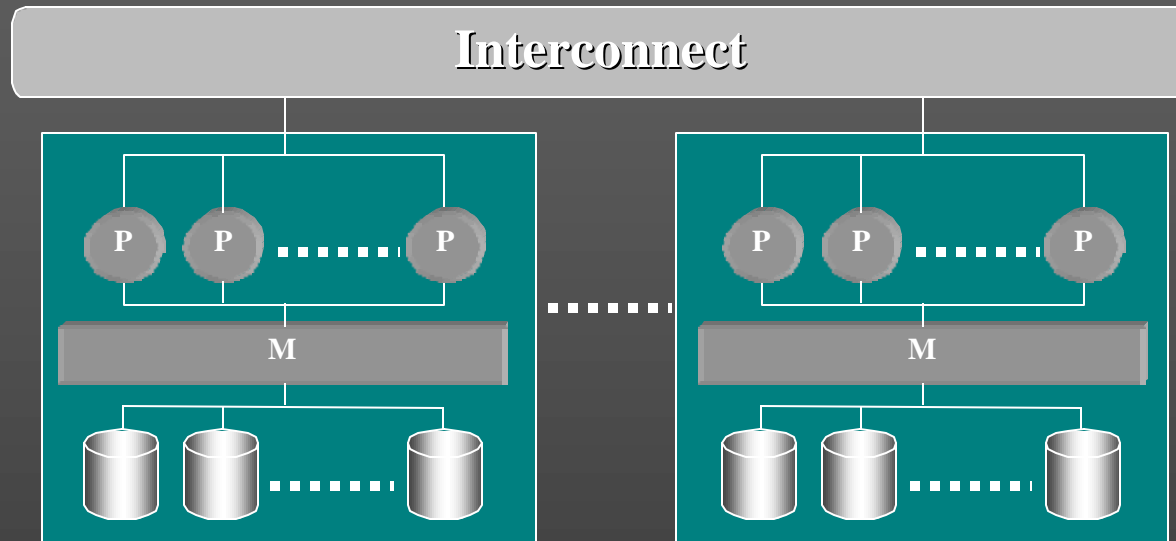
|                                 | SE | SD | SN |
|---------------------------------|----|----|----|
| Scalability                     | -  | ±  | +  |
| Load Balancing                  | +  | ±  | -  |
| Reliability & Data Availability | -  | ±  | +  |
| Simplicity for programming      | +  | -  | ±  |

Shared-disk architecture offers no particularly persuasive arguments for its support.

Shared-Everything architecture suffers from low scalability and reliability.

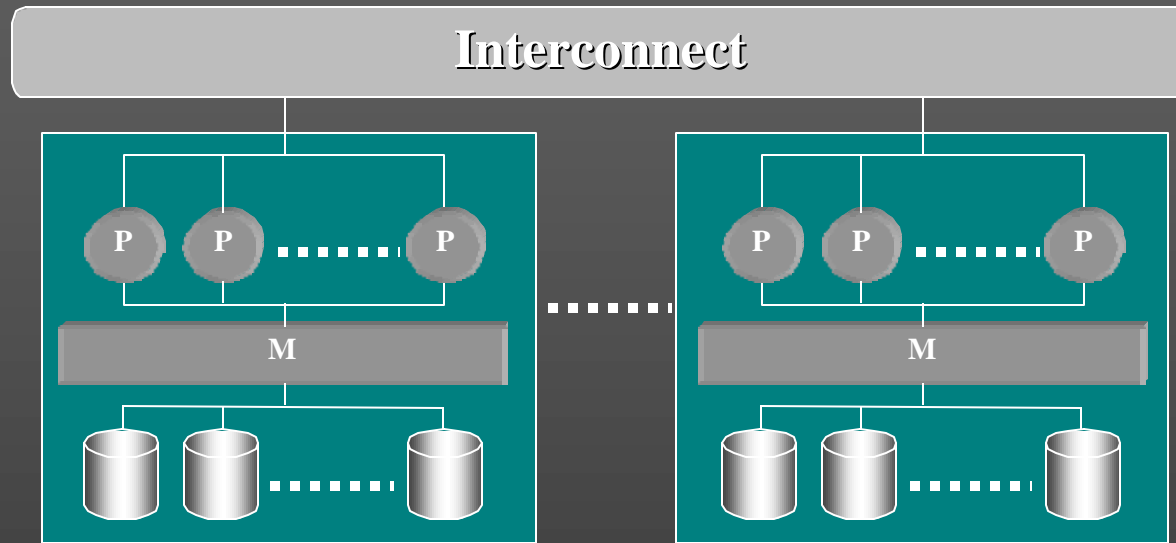
Shared-Nothing architecture suffers from problems with load balancing.

# CE-architecture



To incorporate the best features of shared-nothing and shared-everything architectures Bhide suggested to consider a hierarchical multiprocessor database architecture in which SE-clusters (inner level) are combined in SN-manner (outer level). We denote it as CE (Clustered-Everything) architecture.

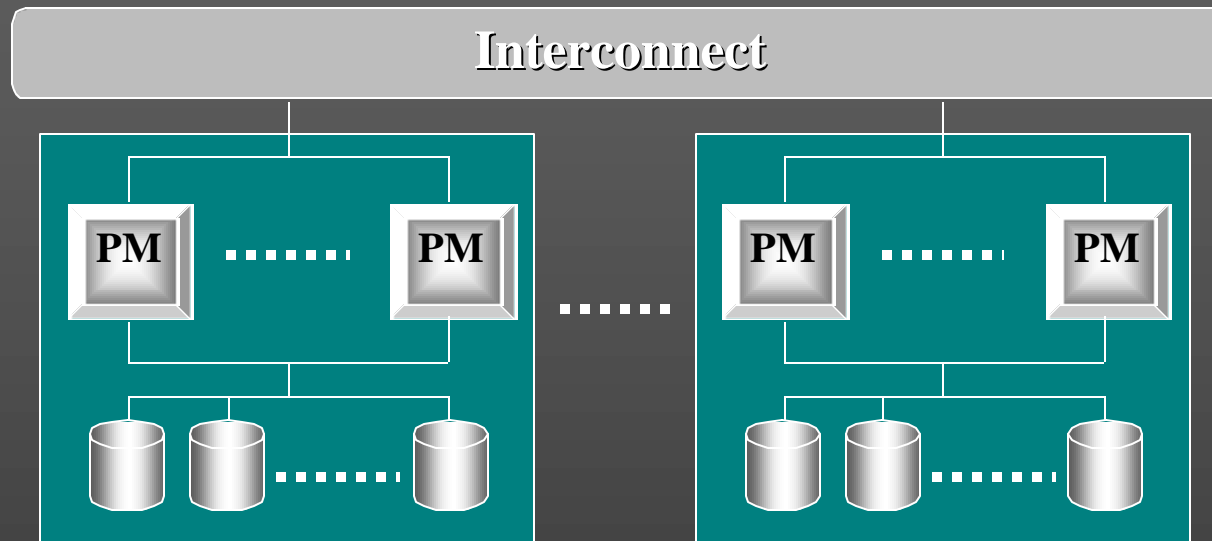
# CE-architecture



CE-architecture demonstrates a good scalability. The load balancing can be arranged more easily than in SN-architecture because it can be addressed at two levels, locally among the processors of each cluster and globally among all clusters.

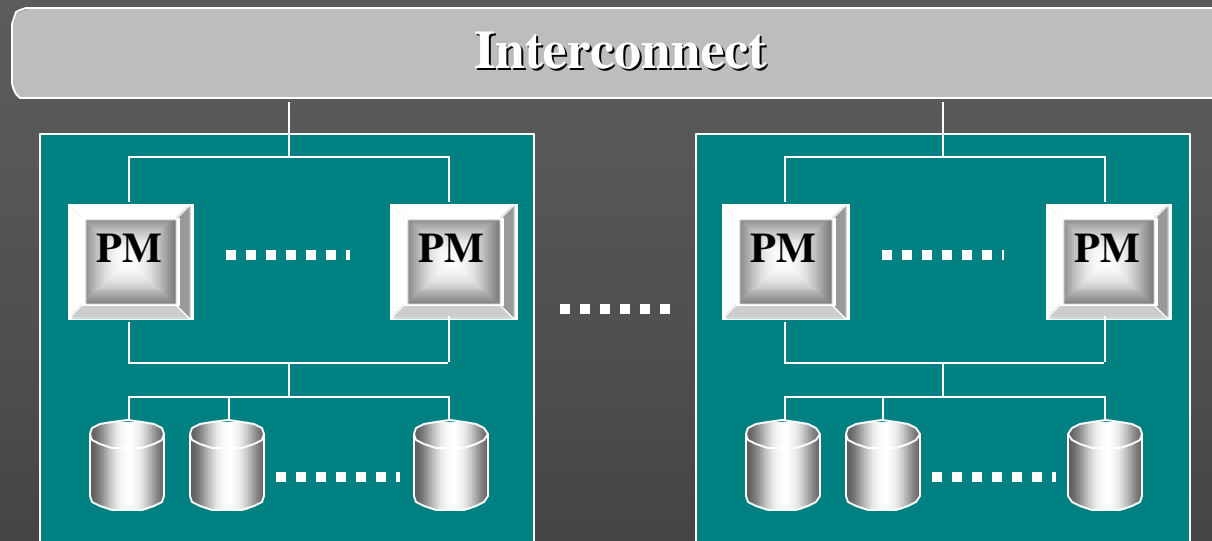
However, CE-architecture can't provide the high data availability because SE-cluster has low reliability.

# CD<sub>2</sub>-architecture



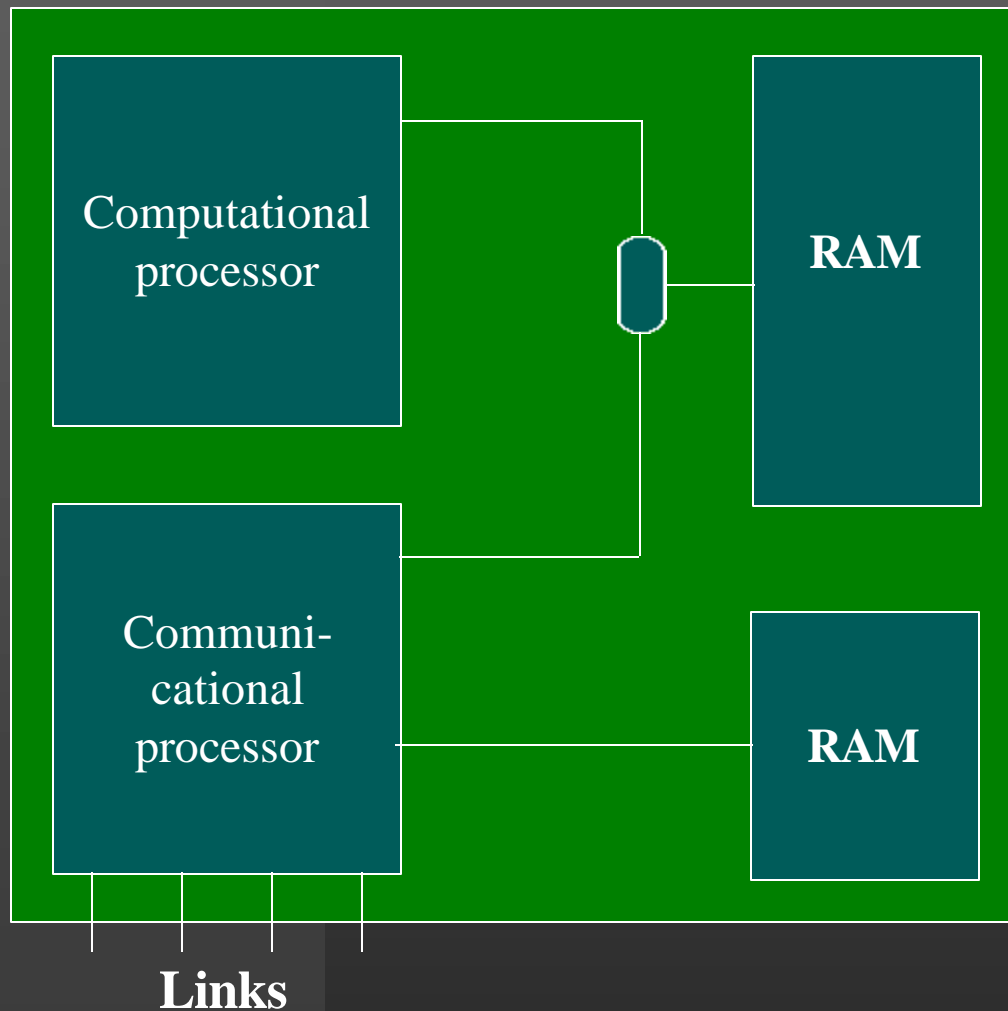
We suggested to consider another kind of hierarchical architecture which we denote as CD<sub>2</sub> (Clustered Disk with 2-processor modules).

# CD<sub>2</sub>-architecture



CD<sub>2</sub>-architecture has 3 levels of hierarchy. The first level is presented by *2-processor units (PM)*. The processor units are composed in *SD-clusters* that form the second level of the hierarchy. On the third level, the SD-clusters are integrated in the whole system in shared-nothing manner.

# Processor Module



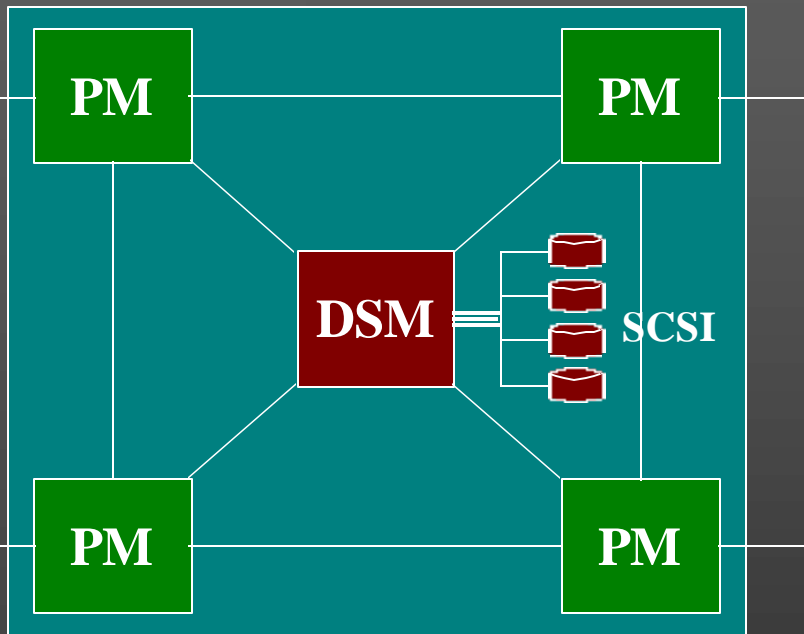
The processor module includes two processor devices: computational processor and communicational processor.

Computational and communicational processors share common memory.

The communicational processor has its own private memory. These two processors exchange data in shared memory.

The communicational processor has four serial links for board-to-board communications.

# SD-Cluster Structure



Processor  
Module



Disk Subsystem  
Module

The second level of system hierarchy is presented by the *SD-cluster*. The *SD-cluster* is a shared-disk system whose nodes are 2-processor modules.

Disk Subsystem Module has its own communicational processor, which is connected to disk devices by SCSI bus.

One link of each processor module remains free to connect the given *SD-cluster* to others.

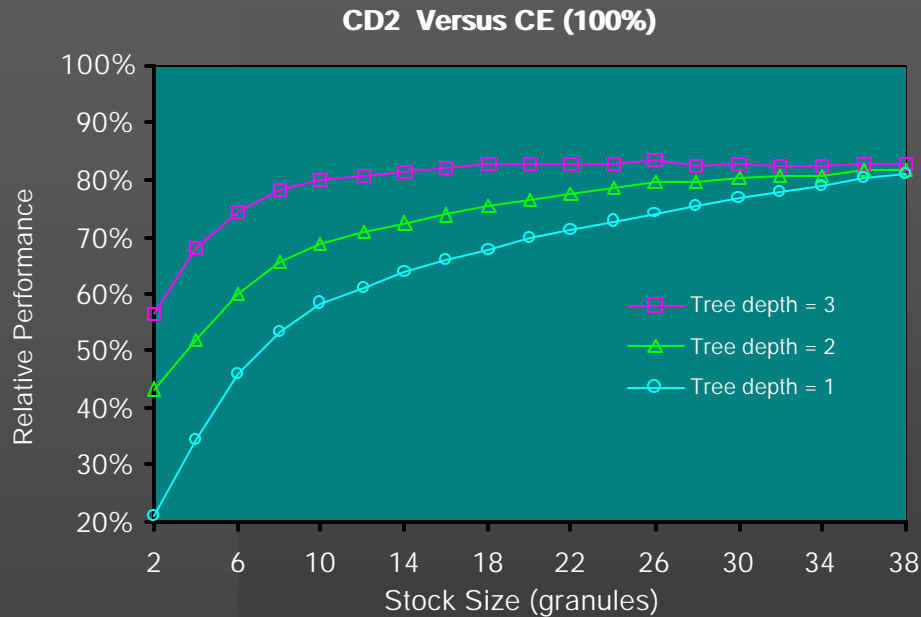
# Implementation

We have implemented a prototype of the CD<sub>2</sub> parallel database system on a 8-processor MBC-100 computer.

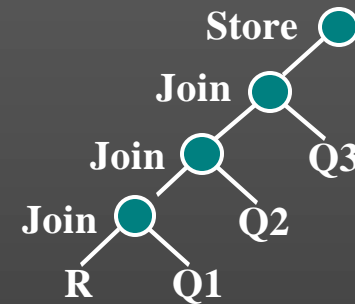
## System parameters

| Parameter                    | Default Value        |
|------------------------------|----------------------|
| <b>Cluster Parameters</b>    |                      |
| <i>cpu_speed</i>             | 20 MIPS              |
| <i>link_speed</i>            | 20 Mbit/sec          |
| <i>number_of_cpus</i>        | 4                    |
| <i>number_of_discs</i>       | 4                    |
| <i>rout_length</i>           | 1.3 (mean)           |
| <b>Workload parameters</b>   |                      |
| <i>schedule_cost</i>         | 500 instructions     |
| <i>granule_size</i>          | 1000 bytes           |
| <i>inst_per_granule</i>      | 8000 instructions    |
| <b>Database Parameters</b>   |                      |
| <i>R_cardinality</i>         | 1600 granules        |
| <i>Ri_value_count_factor</i> | 0.1                  |
| <i>R_distribution</i>        | Zipfian (80-20 rule) |
| <i>Qi_distribution</i>       | Zipfian (80-20 rule) |
| <b>Query Parameters</b>      |                      |
| <i>selectivity_factor</i>    | 1                    |
| <i>tree_depth</i>            | 3                    |

# Performance Evaluation



## Query template



The experimental results show that CD<sub>2</sub>- system can provide performance very close to that of the CE-system while providing higher data availability then the last one.