Literatur – Verwendung

**01: DATABASE SYSTEM CONCEPTS SIXTH EDITION**

* Functions and Procedures (p. 173):
  + **„**Procedures and functions allow “business logic” to be stored in the database, and executed from SQL statements” (p. 173)
  + For example, universities usually have many rules about how many courses a student can take in a given semester
* Triggers (p. 180)
  + „A trigger is a statement that the system executes automatically as a side effect of a modification to the database” (p.180)
  + “Triggers can be used to implement certain integrity constraints that cannot be specified using the constraint mechanism of SQL” (p. 180)
* Materialized Views (p. 607):
  + „When a view is defined, normally the database stores only the query defining the view” (p. 607)
  + “Materialized views constitute redundant data, in that their contents can be inferred from the view definition and the rest of the database contents” (p.607)
  + “Another option for maintaining materialized views is to define triggers on insert, delete, and update of each relation in the view definition” (p. 608)
  + *“A simplistic way of doing so is to completely recompute the materialized view on every update” (p. 608)*
  + „A better option is to modify only the affected parts of the materialized view, which is known as incremental view maintenance” (p. 608)
  + “Indices are just like materialized views, in that they too are derived data, can speed up queries, and may slow down updates. Thus, the problem of index selection is closely related to that of materialized view selection, although it is simpler” (p. 613)
  + “Most database systems provide tools to help the database administrator with index and materialized view selection. These tools examine the history of queries and updates, and suggest indices and views to be materialized. The Microsoft SQL Server Database Tuning Assistant, the IBM DB2 Design Advisor, and the Oracle SQL Tuning Wizard are examples of such tools“ (p. 613)

**02: DATABASE SYSTEMS The Complete Book**

* Indizes => p. 350-353
* Views => p. 341-349, 352-366

**03: SQL Performance Explained**

* Indexing NULL (p. 54)
  + “A missing NOT NULL constraint can prevent index usage in an Oracle database-especially for count(\*) queries” (p. 57)
* Indexing ORDER BY (p. 129)
* Execution Plan with MySQL (p. 188)
  + “Getting an Execution Plan: Put **explain** in front of an SQL statement to retrieve the execution plan” (p. 188)
  + „MySQL’s explain plan tends to give a false sense of safety because it says so much about indexes being used. Although technically correct, it does not mean that it is using the index efficiently” (p.188)
  + “ref, range: Performs a B-tree traversal and walks through the leaf nodes to find all matching entries (similar to INDEX RANGE SCAN). See also “Anatomy of an Index” on page 1.” (p.189)

**04: DATABASE MANAGEMENT SYSTEMS**

* Views (p. 86-90)
  + „A view is a table whose rows are not explicitly stored in the database but are computed as needed from a view definition” (p.86)
  + *“A view can be used just like a base table, or explicitly stored table, in defining new queries or views” (p. 87)*
* Triggers (p. 168-170)
  + “An insert, delete, or update statement could activate a trigger, regardless of which user or application invoked the activating statement” (p. 168)
  + “Users may not even be aware that a trigger wa.'3 executed as a side effect of their program” (p. 168)
  + “A trigger description contains three parts: Event, Condition and Action” (p. 168)
* B-Tree-Index (p. 356-364)
  + „The height of a B+ tree depends on the number of data entries and the size of index entries” (p. 358)
* Hash-Index (p. 370-385)
* Evaluating Relational Operation (p. 441-444)

**05: Designing Data-Intensive Applications**

* Replication (p. 151-193):
  + „Replication means keeping a copy of the same data on multiple machines that are connected via a network” (p. 151)
  + *There are several reasons why you might want to replicate data (p. 151):*
    - *„To keep data geographically close to your users (and thus reduce latency)”*
    - *„To allow the system to continue working even if some of its parts have failed (and thus increase availability)”*
    - *„To scale out the number of machines that can serve read queries (and thus increase read throughput)”*
  + „In the simplest case, the leader logs every write request (statement) that it executes and sends that statement log to its followers” (p. 158)
  + “For a relational database, this means that every INSERT, UPDATE, or DELETE statement is forwarded to followers, and each follower parses and executes that SQL statement as if it had been received from a client” (p.158)
  + “An alternative is to use different log formats for replication and for the storage engine, which allows the replication log to be decoupled from the storage engine internals. This kind of replication log is called a logical log” (p.160)
  + “In the next chapter we will continue looking at data that is distributed across multiple machines, through the counterpart of replication: splitting a large dataset into partitions.” (p.193)
* Partitioning (p. 199- 217):
  + “For very large datasets, or very high query throughput, that is not sufficient: we need to break the data up into partitions, also known as sharding” (p.199)
  + “In effect, each partition is a small database of its own, although the database may support operations that touch multiple partitions at the same time” (p.199)
  + “The main reason for wanting to partition data is scalability. Different partitions can be placed on different nodes in a shared-nothing cluster”. (p.199)
  + *„Partitioned databases were pioneered in the 1980s by products such as Teradata and Tandem NonStop SQL [1], and more recently rediscovered by NoSQL databases and Hadoop-based data warehouses.” (p.200)*
  + **„Partitioning is usually combined with replication so that copies of each partition are stored on multiple nodes. This means that, even though each record belongs to exactly one partition, it may still be stored on several different nodes for fault tolerance” (p.200)**
  + “Because of this risk of skew and hot spots, many distributed datastores use a hash function to determine the partition for a given key” (p.203)
  + „A good hash function takes skewed data and makes it uniformly distributed” (p.203)
  + “The goal of partitioning is to spread the data and query load evenly across multiple machines, avoiding hot spots (nodes with disproportionately high load)” (p.217)

**06: The Data Warehouse Toolkit**

* Aggregate Fact Tables or OLAP Cubes
  + „Aggregate fact tables are simple numeric rollups of atomic fact table data built solely to accelerate query performance” (p. 45)
  + **“A properly designed set of aggregates should behave like database indexes, which accelerate query performance but are not encountered directly by the BI applications or business users” (p. 45)**
  + “Aggregate fact tables contain foreign keys to shrunken conformed dimensions, as well as aggregated facts created by summing measures from more atomic fact tables” (p. 45)

**07: High Performance MySQL, Third Edition**

* Benchmark => p. 35-49
* Grundlagen/Tools => p. 50-66
* Datentypen => p.115-145
* Indizes => p. 147-189
* Views => p. 276-281
* Partition => p. 265-273
* Replikation => p. 447-477

**08: Redis Essentials**

* Replication (p. 146-148):
  + „Replicas are widely used for scalability purposes so that all read operations are handled by replicas and the master handles only write operations” (p.147)
  + “Replicas are very useful in a master failure scenario because they contain all of the most recent data and can be promoted to master” (p.148)
* Range – and List-Partitioning
  + „Partitioning is a general term used to describe the act of breaking up data and distributing it across different hosts” (P.148)
  + “we can use replicas to optimize reads and remove some bottlenecks from the master instance, but in many cases, this is not enough” (p.149)
  + „In the context of Redis, horizontal partitioning means distributing keys across different Redis instances, while vertical partitioning means distributing key values across different Redis instances” (p. 149)
  + “Range partitioning is very simple; data is distributed based on a range of keys” (p.149)