Design of Exercises for the Lecture Distributed Information Systems

Thesis in applied computer science bachelor degree program at department IV – economics and computer science, university of applied sciences and arts Hannover

Issue due date: 25th of August 2015

Author: Alexander Sirotin

This page intentionally left blank

Participants

Author

Alexander Sirotin
Willmerstrasse 20 b
30519 Hannover
Email: alexander.sirotin@stud.hs-hannover.de; sirotin8@web.de

1st examiner

Prof. Dr.-Ing. Arne Koschel
Ricklinger Stadtweg 120
30459 Hannover

Raum: 320
Tel.: +49 511 9296-1880
Fax.: +49 511 9296-1810
E-Mail: arne.koschel@hs-hannover.de

2nd examiner

Prof. Dr. Felix Heine
Ricklinger Stadtweg 120
30459 Hannover

Raum: 302
Tel.: +49 511 9296-1834

E-Mail: felix.heine@hs-hannover.de

Declaration of original authorship

I hereby declare that the presented thesis is my own unaided work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature and acknowledgment of collaborative research and discussions.

Hannover, 09/28/2015

_____

Alexander Sirotin

Credits

I wish to thank, first and foremost, my professors and my parents.

I am indebted to my beloved friends Chris, Patrick, Andreas and Anna for their listening and patience.

This thesis would not have been possible unless by the financial aid of Germany, German government (BAföG) and the organisation of financial support to students in Hannover.

Table of Contents

Table of figures

List of tables

Glossary

"Don't wish it was easier wish you were better. Don't wish for less problems wish for more skills. Don't wish for less challenge wish for more wisdom" Jim Rohn, freelancer in personal development and philosopher

1 Introduction

The main topic of this thesis is the development of exercises about the topic and lecture distributed information systems. Before it will be started with the actual thesis, a short overview as kick-start about the topic will be given.

Like in our daily life, centralizing something like information systems on a dedicated instance for serving only one purpose, comes very handy and is greatly efficient, since of its simplicity. This advantage will enable the user/client world growth and the centralized server world must grow as well, because it must keep up with the needs of the clients. But when an information system becomes very large, new issues need to be managed. The reasons are simply the exceeding of the limited resources and materials, on which the centralized information system is based on to function properly. Instead of providing more resources, like higher CPU power, which will also be more and more difficult to afford and expensive, the information system itself can be split into logical parts and be set up with several physical resources. That means, all distributed parts are still serving together one purpose and this purpose cannot be achieved or obtained completely, if one of these parts are missing. Obviously, the distributed parts need to communicate with each other in order to coordinate their processes, managing their jobs/tasks, planning their resources and much more. Of course these actions are also going to be done in a centralized information system, but not as sophisticated as in the distributed variant. This background processing is not interesting for the clients and should be kept hidden to provide the services as simple and efficient as possible. To keep this complex knowledge out of the end user's world, certain people must obtain this knowledge to administrate, develop and use these systems. In the 70s and 80s trying to implement distributed databases was due to the I/O bottleneck (performance loss using network and HDD VS main memory) not possible [DDBS] (chapter 14.1.1 Objectives).

1.1 Examples for Distributed Services

Following some real life examples of distributed information systems are shown: In healthcare analyzing DNA can improve research work and new knowledge can be gathered about simple protozoon's, more complex bacteria or even humans.

> "The cancer Biomedical Informatics Grid, works to provide a collaborative information network for cancer research. Their mission is to accelerate the discovery of new approaches for the detection, diagnosis, treatment, and prevention of cancer, ultimately improving patient outcomes" [eHealth].

Also in weather forecasting parallel computing platforms are used, amongst others for real-time numerical weather prediction [weatherForecast]. Next example, Microsoft Azure, Google Cloud Computing, Facebook Data Center or Amazon EC2 are powerful clouds which have data centers all over the world and are hosting distributed services like Skype, Google Search, the social network service Facebook and Infrastructure As A Service. With the Hadoop framework, predictive analytics are easier to perform and dozens of companies providing solutions based on this framework as well as hardware companies are providing the needed infrastructure for Hadoop clusters.

1.2 Motivation

The motivation to write about in how exercises over DIS[1] can be created is the launch of a new lecture with this topic. The author had some experiences in the field distribution and saw the offer in writing his thesis about a scenario-development for DIS-exercises as a chance to increase his knowledge and competence in the same field. DISs are for the author very interesting, because of their growth in usage and importance in quite all practical fields like shown in the examples above. This interdisciplinary service of distributed computing is due the facts on the one hand in reaching physical limitations of manufacturing better semiconductors, which centralized information systems heavily depend on for their own growth and on the other hand the explosive increase of data in all areas (medical, business administrations,...), which are also needed to be processed faster. Therefore, many companies have realized the potential of distributed systems and are providing cheap solutions for high performance computing, since the cost of semiconductors for memory and disks are decreasing. Using such products enforces software developers, researchers and analysts, who are dealing with huge amount of data, to have the know-how in distributed computing. Also, companies are starting to write fundamental knowledge about specific distributed databases as a requirement – not anymore as a nice-to-have skill - in their job offers for Associate Engineer. Because of that, gaining the first experiences with distributed information systems during the study, would bring great advantage in getting an interesting offer as a job seeker.

1.3 Aims of the Lecture

A lecture is given about this topic at the University of Applied Sciences and Arts in Hannover. The content is, such as teaching algorithms and management of distributed systems, which are invisible to end users and are the oil between distributed components. Besides, additional advantages of distributed information systems like scalability and replication / fail over tolerance, are also taught there. The aims of the lecture are to share basic knowledge about distributed information systems and to show, how distributed databases are working. One part is to talk about Distributed Relational Database Management Systems and the other is to discuss about the advantages of distributed NoSQL databases. Also, it is preferable to confront students with Big Data-, Cloud Computing-technologies and specific distributed DBS for practical experience.

1.4 Responsibilities and Tasks in this Thesis

The author of this thesis developed an exercise-scenario for the lecture 'Distributed Information Systems'/'Verteilte Informationssysteme' (VIS) of the postgraduate / master study. The scenario was based on the provided learning objectives of the thesis-examiners and should be embedded in all exercises to uniform and integrate them, so that students can concentrate better into the learning matter. The purpose of this work has been first to help reworking and developing the lecture's exercises and secondly to increase the learning effect for the students and decrease their distraction, which lies upon others in having different datasets from different tutors. That means, that in this work exercises have not only to be developed completely new, but also some from the previous semester must be unified and overhauled. In summary, in this thesis the practical reworking, creation of exercises and implementation is partly going to be done by the author of this thesis. In order to do this task efficiently and reproducibly, a concept is going to be developed, which should be the foundation for the creation of exercises.

1 DIS = Distributed Information System

1.5 Structure of this Document

In this chapter, distributed IS and the corresponding lecture has been introduced, as well as the motivation and a short description of the author's work have been given.

The rest of this document is structured as follows. In chapter 2 basic knowledge about distributed databases are going to be described, which is not a subject in the bachelor degree study and is needed to understand the ongoing chapters.

The main part of this thesis is the creation of a concept for exercises of the lecture 'Distributed Information Systems'. First of all, in order to reach the desired goals, the exercises must fulfill certain factors like they must match with the given lesson, be motivating and understandable by the students and of course also be reasonable in difficulty and with regards to the contents. In the exercises these factors are not directly shown, but are important for the learning effect. Therefore, in this concept the requirements, conditions and ideas are covered to provide more transparency for others and which is helping the author to develop more qualitative exercises. The concept will be structured as the didactic design [eTeach_dida] and teaching scenario [eTeach_scen] from www.e-teaching.org. These references will serve as a framework for this concept. This framework will be used in context for exercises only and therefore not every point from the references will be covered. Additionally, other, topic related points will be covered in this work. The following three chapters are covering the concept:

Chapter 3 is about the didactic design of the exercises. Besides the basic conditions and other minor points, it will contain mainly the abstract learning objectives and the exercise design pattern. Chapter 4 contains the concrete learning objectives, which have been swapped out from chapter 3, since of its huge size. Chapter 5 covers the teaching scenario and describes first the choices of the used DBSs and datasets and then goes into detail of several exercises, which takes a main part of that chapter.

Chapter 6 will provide a few implemented exercises. The last chapter gives the final conclusion about the thesis. It will provide the reader a summary of this work, the author's reflection and an estimation about the future work regarding the VIS-exercise development.

This concept will contain the learning aims given by the thesis-examiners in chapter 3.2.3. These aims will be used as a fundament to develop exercises through the conception.

After this introduction, this thesis will go deeper into detail about distributed databases and will describe some of the basics regarding this topic.

2 Basics

The basic chapter contains six points, which are required or are good to know in order to understand the thesis. The points are ordered after the exercises or the lecture's content. First, the partitioning and fragmentation in DRDBS will be explained. Also the students shall understand the partitioning criteria, which will be covered in one of the exercises and therefore it is reasoned to mention this definition in this chapter. Another concept being unknown for most bachelor students is the two-phase commit protocol. This protocol is a simple communication method along more independent DBSs, which is about data on several machines, that need to be synchronized or changed at the same time. After describing 2PC, the deadlock on DRDBS will be handled. Causing or understanding deadlocks in distributed environments does give a better and simpler view on these kind of systems. The last two points are on regard of MapReduce and Hadoop. MapReduce itself is a big topic and just some basics will be mentioned. Same as for Hadoop, only the basic architecture and not the internals will be explained.

These basics have been chosen, because they will give the reader a clearer view of the range and spread this thesis is handling. There are existing of course, many more points to discuss, since it is going about a whole new area. For example also the definition of cloud, a method of comparing different types of DBSs, some teaching methods like the bloom's taxonomy and the architectures of the relational DDBSs like MySQL and MS SQL Server replication could have been introduced. All these things will be skipped to keep a clearer, easier and general view on this work.

2.1 Fragmentation, Partitioning

A partition is a placeholder for fragments of data. A fragment is a part of that data. All fragments in sum are together the data. A duplicated or copied fragment is called a replicate or a replica. Partitioning means to split up a placeholder. Same as for partitioning, fragmentation means, splitting up the data. The partitioning- and fragmentation-method are describing, how to split the data into fragments and the storage into partitions. The partitioning criteria describes, which fragments belong to which partitions. A replica is nothing more than a copy of one of the fragments and will be handled quite same, as the original one except, that the replicas should be saved on different storage containers and therefore on different hosts. The details, for example master-slave approach, are architecture dependent.

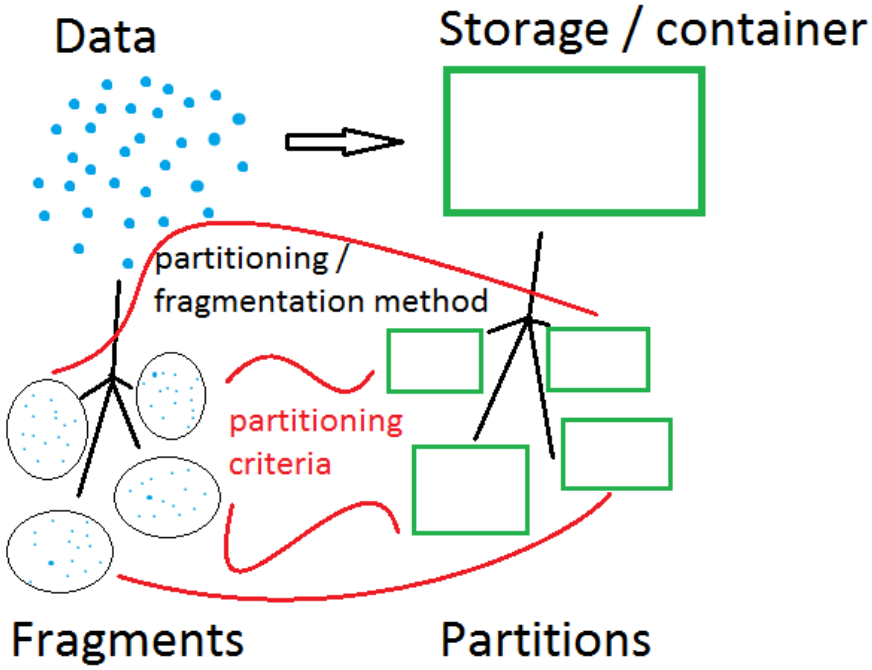


Illustration 1: Relation between fragments and partitions

2.2 Partitioning-Methods and -Criteria

The previous point mentioned the partitioning-methods and -criteria. Now, these terms will be discussed in detail. But before that, one thing should be noted: partitioning does not mean to distribute the data over hosts, but just split them in some way. It can be just different files, different tables, but also different hosts.

2.2.1 Partitioning-, Fragmentation-Methods

Two methods and a mix of both of them will be covered in this sub chapter. The first one is to split data row-oriented, which is called horizontal fragmentation. Each tuple or each group of tuples will be assigned to one partition. Vertical fragmentation does the same with the attributes of a table – not with the tuples of it. The following phrase expresses both definitions in a short and exact way:

> "Vertical partitioning subdivides attributes into groups and assigns each group to a physical object. Horizontal partitioning subdivides object instances (tuples) into groups, all having the same attributes of the original object."[vertPar] (chapter Introduction)

The hybrid fragmentation just splits the data consecutively horizontally and vertically. Means, not only the tuples and attributes of a table will be divided, but the resulting fragments, too.

2.2.2 Partitioning Criteria

Assume the following definitions:

n is the id of a partition starting with 0

N is the total amount of partitions

For horizontal method: i is the id of a tuple starting with 0

For vertical method: i is the id of an attribute starting with 0

Partitioning criteria n=f(i)

According to these definitions, three partitioning criteria will be presented, which are covered in [DDBS]. The first one is Round-robin with its criteria := f(i)=i mod N. The advantage of this partitioning is, when data is accessed or written sequentially, full parallel read and write operations can be enforced through the nodes.

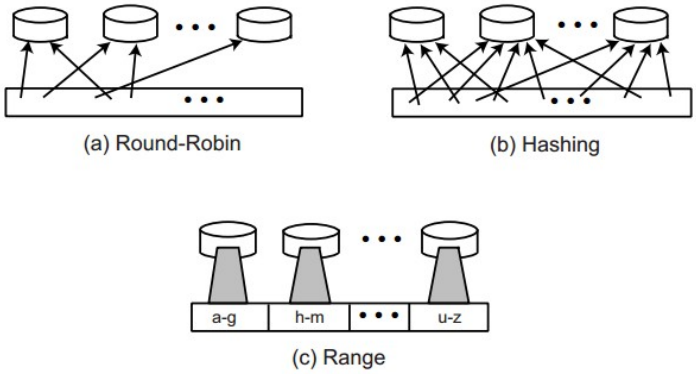


Illustration 2: Partitioning types/criteria [DDBS] (chapter 14.2 Parallel Data Placement)

Next is the hash-based partitioning with the following criteria := f(i | i-th tuple)= (MD5(i) | SHA(i) | more2kSalary(i)|...) mod N. The DBS can either use a technical hash function like MD5 on the id / primary key, which will not be more different than the Round-robin-method, or it can use a business logic function to check for a specific characteristic on tuples (called predicate in [DDBS] in chapter 3.3.1). This makes each partition a special placeholder for data. The reader can imagine the hash-functions as firewall rules:

1. Every employee, who has < 2k salary, goes to partition 1
2. Every employee, who has => 2k salary and is not a boss, goes to partition 2
3. Every employee .....

The advantage is, that the (data-model) developer can decide, which kind of queries should be run on a single node and which all other queries should run in parallel.

The third and last partitioning is the range-based one := f(i | i-th tuple). Following two benefits are existing: If physical storage limitations are critical, the size of a partition can be fixed using the range definitions on the primary key. If tuple attributes are used for ranges (not a primary key, but for example, salary), range-based queries can be efficiently executed on single nodes. These queries would be like full table scans, but on the 'right' nodes.

2.3 Two-Phase Commit Protocol

Distributed transactions are often implemented as the two-phase commit protocol (2PC). To understand, how 2PC works, it is recommended to type and execute all steps manually. Many DBS are already providing XA transactions, which are implemented as 2PC. XA itself is

> "A standard interface for coordinating distributed transactions, allowing multiple databases to participate in a transaction while maintaining ACID compliance" [mysql_ref] (chapter MySQL Glossary)

Before an XA transaction is initiated, a network connection to a database must be established. The next step is to define an XA transaction with "XA start [name]; [SQL statements]; XA end [name]". After that, the defined transaction can be executed in two phases. First phase can be started with

> "XA prepare [name]" and the second with "XA commit [name]" [mysql_ref] (chapter 13.3.7.1 XA Transaction SQL Syntax)

Between the two phases, the client must wait for a positive acknowledge from all servers. If one server couldn't pre-execute successfully, a XA rollback [name] must be sent to all servers to revoke the transaction. The following figure shows the structure of Open XA:

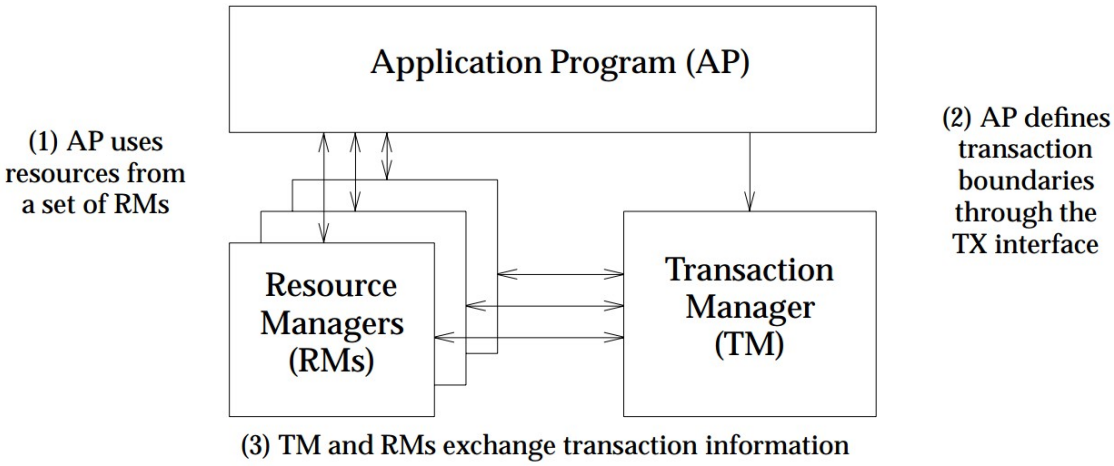


Illustration 3: Simple deployment model of Open XA [oXA]

2.4 Deadlocks on DRDBS

Deadlocks can occur, when transactions are blocking each other by not releasing their write locks until the other transaction releases his lock and the DBS recognize it as a deadlock and rollbacks one of the waiting transactions [mysql_ref] (chapter MySQL Glossary). Since DRDBSs consist of several hosts, a central resource manager is needed to maintain and granting the locks to the database clients. If there is no such a resource manager, the clients are responsible to mark the data they are going to modify and to respect the marks made by other clients. If locking data resources is not provided by a (DR)DBS, data consistency is not guaranteed after every modification / write operation. The resource manager of a DRDBS should handle locks similarly as like as in a non-distributed DBS. Therefore the deadlocks can also be caused in the same way as in single DBS as shown in the illustration below.

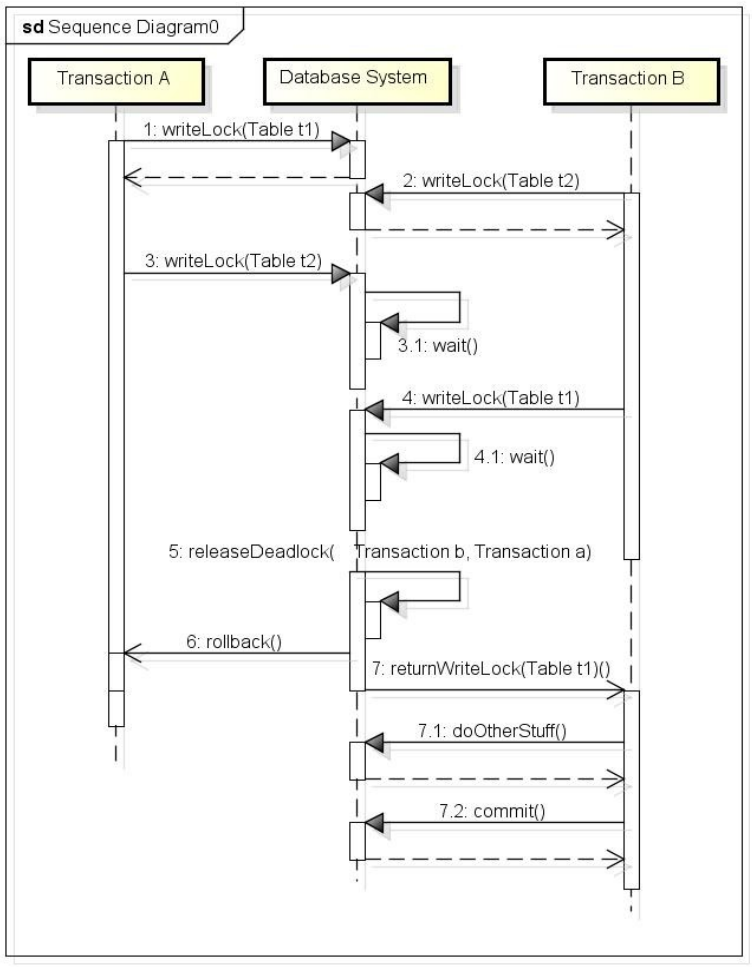


Illustration 4: Provoke Deadlock

2.5 Map-Reduce

Map-Reduce is actually a simple operation. Just when people are trying to do the same things on M/R like on relational DBS, the understanding of M/R will become more difficult. Therefore, it is recommended to ignore the own DBS knowledge in this sub chapter. Also M/R will be described more practical- and cluster-oriented to show the real potential and meaning of this concept.

M/R consists of three main parts: Map, Shuffle and Reduce. The shuffle-process will not be handled in detail. Jobs in M/R are in some kind like the queries in relational DBS. One job is a process of M/R and contains the three above mentioned parts. A job with just map-tasks looks as the following illustration (5). The splits are together one file, which is located mostly on HDFS. When a file is located in HDFS, it might be replicated and distributed (sharded) over the M/R cluster. The splits will be determined by an *InputReader*, which will be executed on the file before the map-tasks.

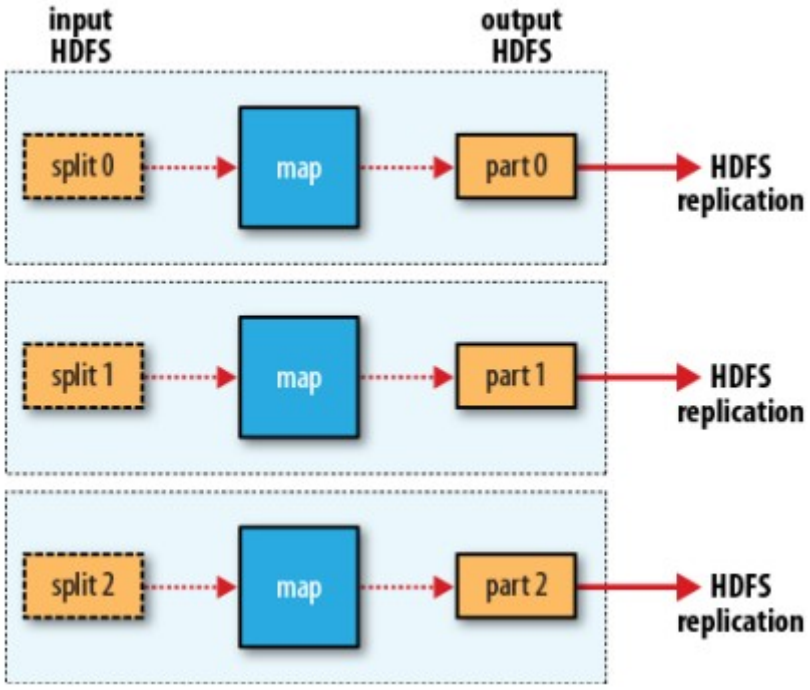


Illustration 5: MapReduce data flow with no reduce tasks.
Source: [HDP_GD] (chapter 2 Data Flow)

When the map-tasks get a hand on the splits, they will be of a predefined type originated from the specific *InputReader* and will be considered as the final input data. Each map-task accesses a part of that data in a key-value oriented way and outputs a list of new key-value pairs. The output (part[number] in illustration 5) can be again of different data structure.

When there are no reduce-tasks selected, also shuffling will not be done, since it is mainly for interconnecting the map- and reduce-tasks with each other to forward and transfer the processed data further. Another difference is the writing the output directly to HDFS, which will cause sharding on the that output, when used mappers only. This is unnecessary, when the final output is made by the reduce-tasks, where in that case the data will be stored in main-memory and/or on local disk as temporary files on the corresponding node.

The illustration 6 shows additionally the reduce-tasks. In between, the white parts are representing the shuffling: it sorts the output of the mappers, copies it to the nodes, where the reduce-tasks are running and merges them together to offer the reduce-tasks a full part of the data.

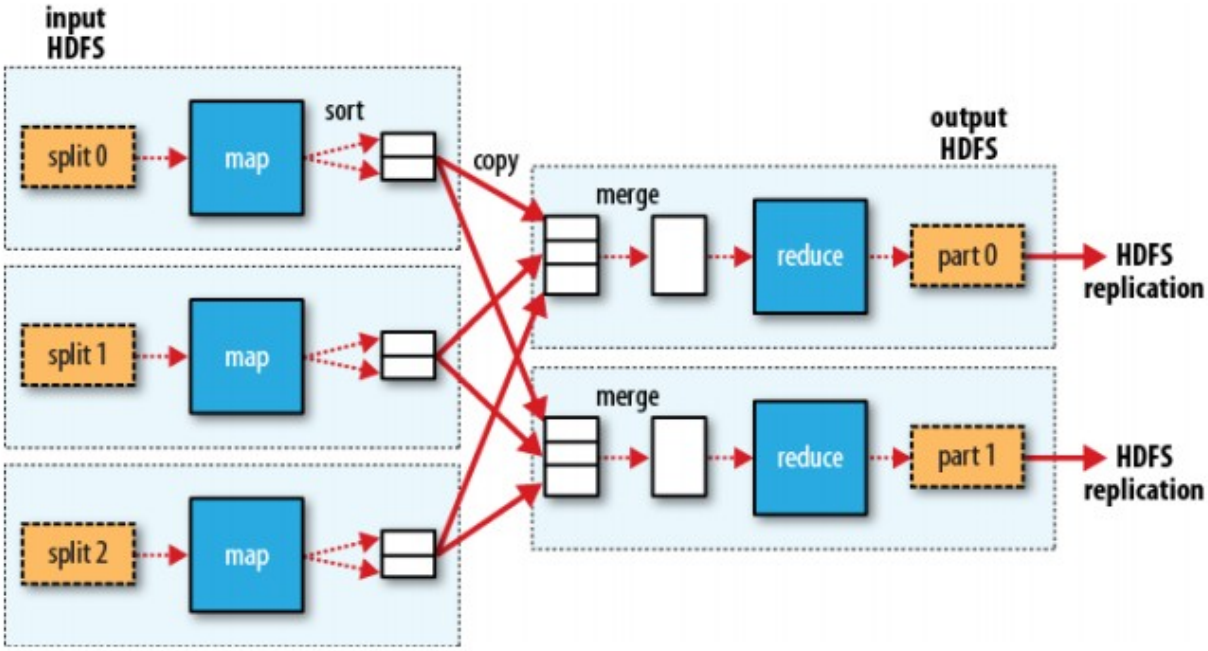


Illustration 6: MapReduce data flow with multiple reduce tasks. Source: [HDP_GD] (chapter 2 Combiner Functions)

2.6 Hadoop Architecture

This architecture consists of two big parts: the HDFS[2] as a lower layer and Yarn as the upper layer. Yarn is also considered as MRv2 replacing and overhauling the older data-computation framework [pche_yarn]. HDFS instead is considered as a storage or 'database' on which Yarn and its jobs can access for writing and reading huge amounts of data. After describing both layers, a short example as deployment will be given to show their interaction.

2.6.1 Hadoop Distributed File System

HDFS is realized as a master-slave setup. While the *DataNodes* are working as slaves and providing data storage, the *NameNode* operates as a single master on a cluster managing the metadata instead. Data is stored as file blocks on *DataNodes*, which can be reassembled through the *NameNode* and its meta data to common files. As in the previous sub chapter mentioned, these blocks can be replicated and distributed through the cluster. For writing and reading on blocks, the *NameNode* is not necessary and therefore cannot become a bottleneck. The following diagram shows, how the clients can access the blocks directly or get the block-numbers (meta data) by asking the *NameNode*.

2 Hadoop Distributed File System

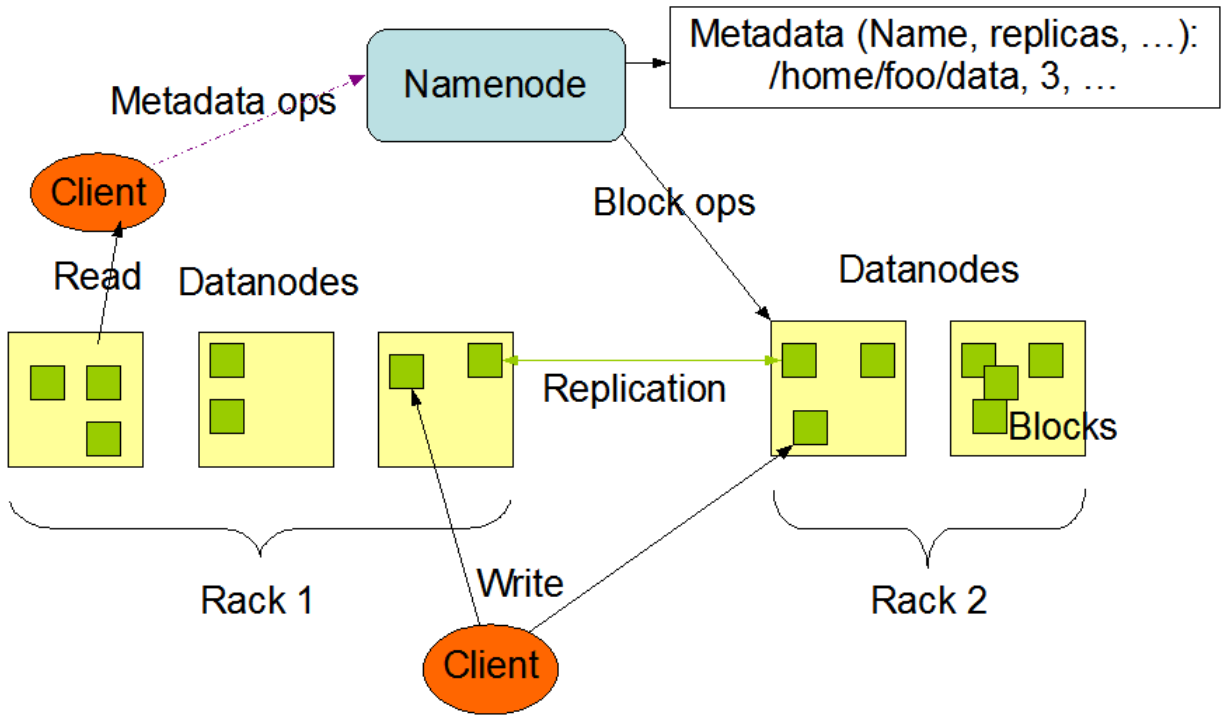


Illustration 7: HDFS Architecture [pche_hdfs]

2.6.2 Yet Another Resource Negotiator

As like as HDFS, Yarn is also realized by a master-slave architecture and is of course quite complicated, why only the *ResourceManager* and the *NodeManager* will be shortly described. First, the *ResourceManager* (single master) accepts jobs (M/R applications) from the clients and launches an *ApplicationMaster* on one of the nodes. The *ApplicationMaster* then negotiates with the *ResourceManager*, how many containers for his mapper- and reducer-tasks can be allocated on the cluster. These containers should not exceed his predefined resource-limits like the memory. The *NodeManager* on each node observes the container's resource usage and forwards the resource usages of the whole node to the *ResourceManager*. The *ApplicationMaster* also sends his reports about the status and progress of his containers to the *ResourceManager*.

2.6.3 Interaction of HDFS and Yarn

The last diagram below (9) presents an already deployed Hadoop cluster. An *ApplicationMaster* can only exist on one of the nodes, where a *NodeManager* is running. This one will retrieve the split information from the client (standard variant) or the cluster and launches his map tasks with these split-information and later the reduce task(s) to summerize the output and write it into HDFS. This means, all running services showed in the diagram do not communicate at all with each other and when they do, then just to arrange an optimal task- and data-flow. The tasks will run quite independently from each other and will only interact with each other during the shuffle phase. Every detail, how M/R is working, is described in [HDP_GD] (chapter 6 YARN (MapReduce 2)).

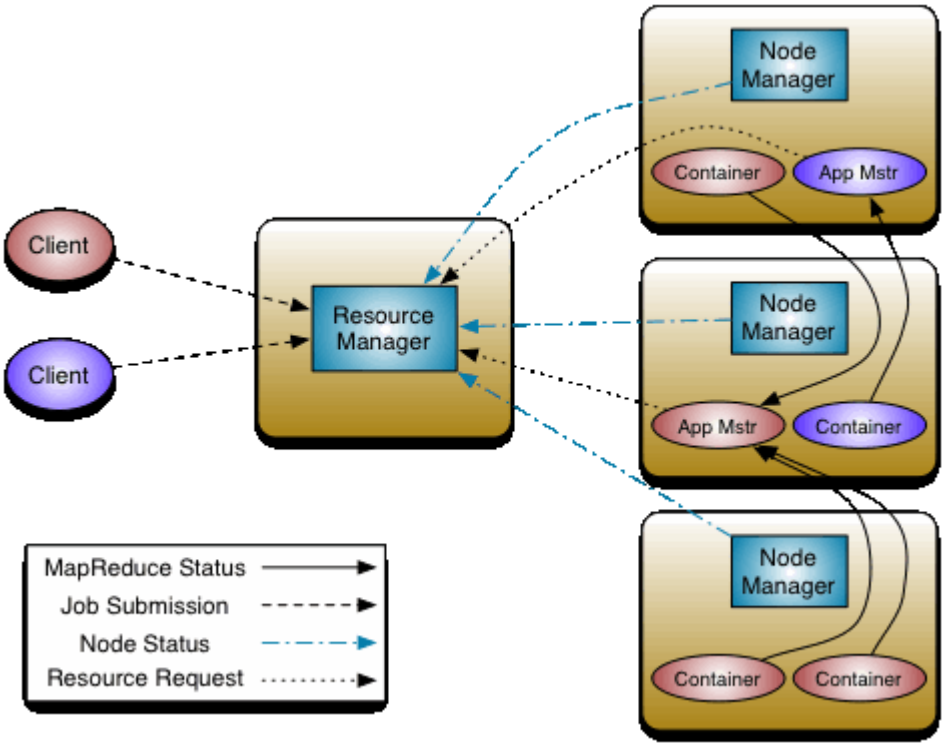


Illustration 8: Yarn Architecture [pche_yarn]

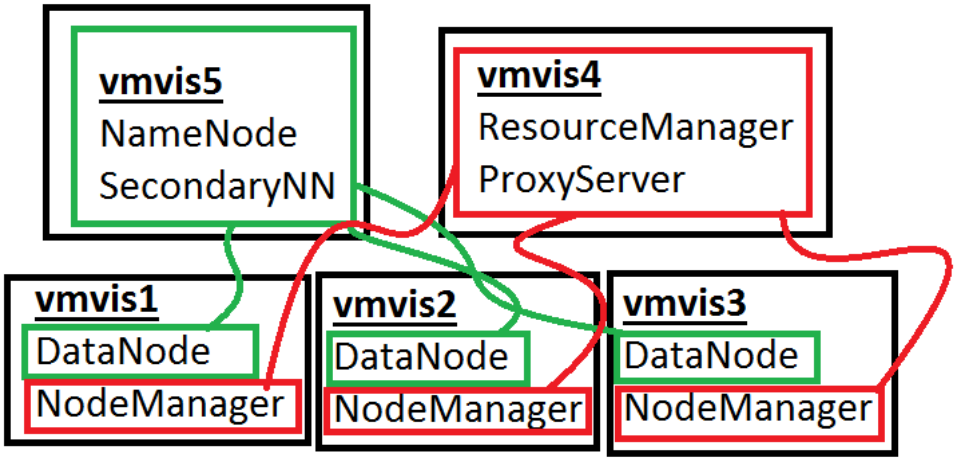


Illustration 9: Concrete example of Yarn and HDFS setup [intro_hadoop]

2.7 Summary

This chapter was going about some of the background information in order to better understand the ongoing work. Sharding, partitioning-methods, -criteria, 2PC, deadlocks on DRDBS, M/R and the Hadoop framework have been introduced. On top of this knowledge a part of the exercises are build on it and therefore this knowledge was required also to develop and create these documents.

The next chapter starts with the didactic design for creating the exercises and discusses some basic points about what should have been taken care in designing the tasks.

3 Didactic Design - Analysis Part I

The didactic design and conception are structured as like as the mind-map from 'Hochschule Bielefeld' [concept_struc]. The map, as like as this design, starts with the branch 'Ziele' / learning objectives and continues in a clockwise direction. The branch 'Medien' is being replaced with 'Design Pattern' in this concept. Besides, the 'Rahmenbedingungen' / basic conditions are taking from the checklist of lehridee.de [concept_checklist] and put on the first place of this design.

3.1 Basic Conditions

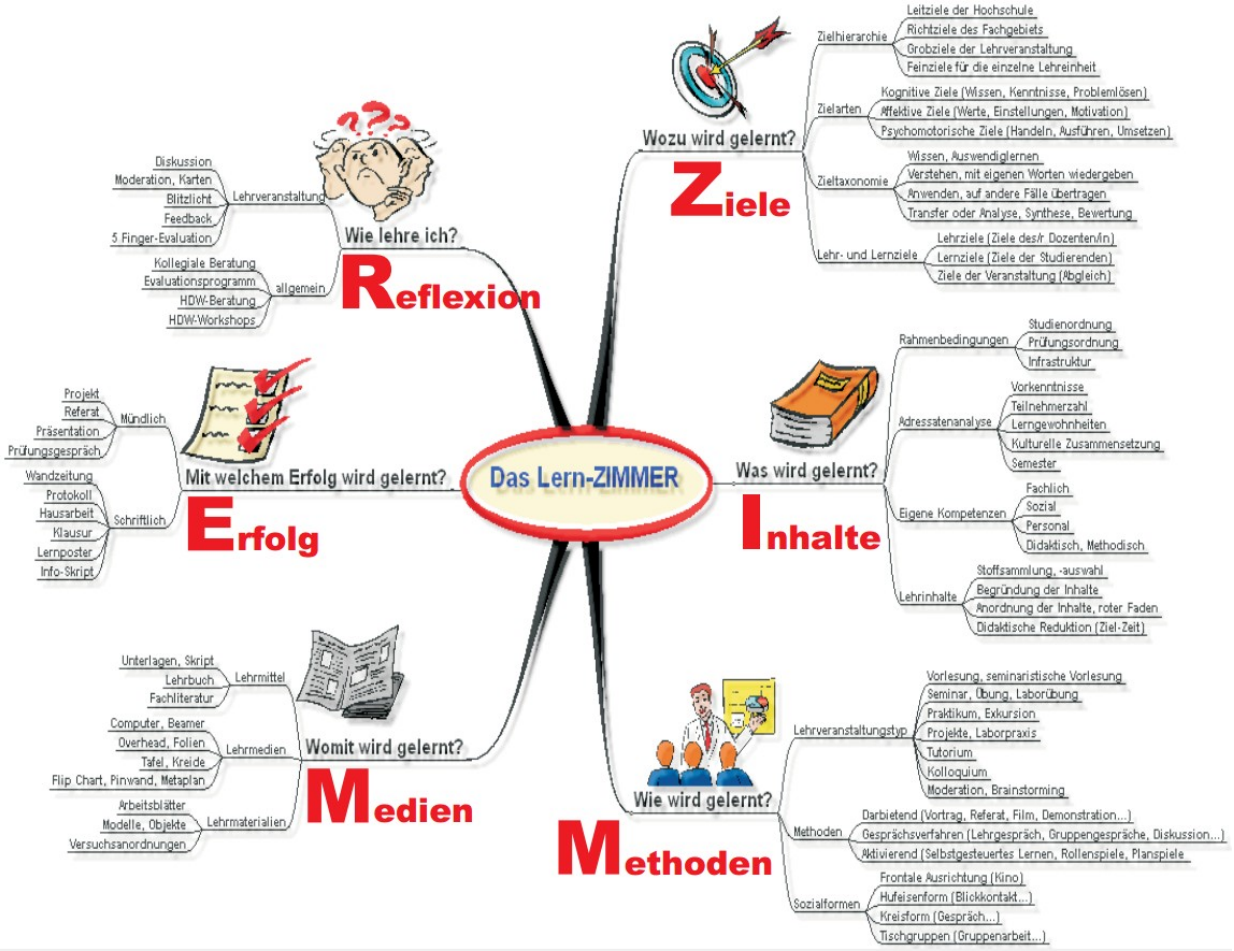


Illustration 10: Structure of original didactic design

First, the basic conditions in creating the exercises are described.

The main reference for the basic conditions is the study guide of computer science from 'Hochschule Hannover' [std_guide]. It specifies the amount of work, learning-objectives, -content and -methods as well as the literature references for the lecture Distributed Information Systems. For this work relevant content in that guideline will be mentioned in the appropriate sub chapters.

The thesis-examiners have further named specific learning objectives to provide a structure and a fundament, which will be presented in chapter 3.2.3 'Learning Aims'. Additional requirements are declaring, that exercises consist of two task-groups: the theoretical part, which should contain technology independent content and the practical part, which challenges in usage of distributed applications like MySQL, Hadoop or MS SQL server.

Next, the number of exercises was stated between 10 and 12. The time of work for each exercise is based on the amount of credit points and the time distribution between lecture and exercise. This lecture has 6 CPs, which corresponds to 180 hours. 34 hours are each for the lecture and exercises. 112 hours are for self study [std_guide]. Half of the self study time is for the practical part to solve the exercises. With 12 exercises, each one will have a volume of work of ((112/2)+34)/12=7.5 hours.

Optionally, the structure of the exercises can be based on the lecture 'Databases' / 'Datenbanken' of the bachelor degree program. This would provide most students a familiar orientation in the consecutive master degree.

Finally, some notices and restrictions will be explained. First, the exercises, which will be created, are on the first run unstable, while the referenced framework for this conception is matured and well-engineered. This will cause easily conflicts and unreachable goals in or after the realization phase. Therefore, some drawbacks in this concept, in context for the first iteration of implementing the exercises, will be made. An example is in chapter 3.4 'Exercise Design Pattern', which takes more focus on the first two levels and less on the last two. The second and last note is to remind, that the enhancements lie on exercises. This concept covers points concerning the whole lecture, because they effect the design of the exercises. However, it is not indented to cover a lecture reading. Since the referenced framework meets the whole lecture-design, many points of it will be faded out.

3.2 Learning Objectives

The learning objectives of this lecture and exercises are structured hierarchically [lernObj_wiki] as different types of aims. The rough and fine aims will be described together and will be swapped out to an own chapter called 4 Exercise Learning Objectives - Analysis Part II. The sub chapter 'Learning Content' will be skipped and partly reflected in chapter 5.3 Creation of Exercises and Tasks.

3.2.1 Guiding Aims

Some aims of the Bologna Process between European countries will be taken here as guiding aims. Upon other terms, this agreement says, that the employability should be national and Europe wide increased for students. Also, they should be motivated to live-learning and to participate more actively in general. To enable researchers their engaging in research project from other countries and working with foreign researchers, is another goal. Lastly mentioned is the integration of modules / lectures with each other in a degree program. These are the basic points taken as a focus for the exercises as guiding aims.

To respect these aims, first technologies should be picked, which are currently often used in the IT and which require skills, that are requested in job offers (employability). Secondly, students can be inspired and asked to do research about technologies and applications beyond this lecture to discuss them in the lessons (live-learning). Third, the previous point supports the self-regulated learning and helps to communicate the student's opinion, which train them to be more active - assumed the topics and tasks are given to them are not too difficult for developing fast an opinion (active participation). Fourth, providing the exercises in English, ensures to use literature from all countries equally, including the ones from Europe. On that foundation, students, who will become later researchers, will have it easier to talk with foreign researchers, since both would have the same literature- and theory-background (European research). The last point is to integrate the exercises into other modules of the same study program and even into some modules from the bachelor program. As the author experienced it by himself, this integration helps and motivates to learn and to practice a lot. However, to do this requires experience and knowledge about both master and bachelor program and in teaching these modules as well as making agreements with other tutors and professors (integration of modules).

These aims will be used in order to better identify the orientation aims, which are on a lower level and will be presented in next sub chapter.

3.2.2 Orientation Aims

The study agreements of the Bologna-Process has been taken over into the European Qualifications Framework [wiki_euFrame] in the levels 6, 7 and 8.

> “The levels of DQR have been assigned to the levels of EQR/EQF in context of so called '1:1' referencing” [drq_eqr].

> “The DQR and the Qualification Framework for University-degrees (HQR) are compatible” [wiki_deFrame].

The level for the master degree of all three frameworks covers the requirements and goals, which will be used as a guideline to formulate the orientation aims at this point. The table above shows the selected aims and how the exercises can cover them.

The selection was based upon the guiding aims and their similarities to the aims of the EU framework. For example the EU aim

> “critical awareness of knowledge issues in a field and at the interface between different fields” [EU_rec]

has not been taken as orientation aim, because it didn't match with one of the five guiding aims described in the previous sub chapter. This EU aim is also very difficult to implement in the exercises in a 5 credit-point module, which teaches a new big topic about information science. Of course, at this point, the reader might think, with 'interface between different fields' could be meant connecting a new topic / module with another module, which would fit to the guiding aim 'integration of the modules'.

Instead the author interprets this aim in a wider aspect like 'What are the issues, using application-level or high level libraries and frameworks for engineering and implementing driver programs used for controlling car engines?' or 'Why do satellites need distributed hardware systems besides of the reasons, which usual clusters are benefiting from, too, like high-availability?' The integration of modules is more or less just helping students to learn better and to get a higher motivation to succeed and is therefore only an aim for this concept – not a student's competence.

Competence	**Possible Coverage**
Knowledge	
Possessing broad, detailed and specialized knowledge on the newest state of awareness in research or in work	Doing research to answer theoretical and practical tasks in context of new software and technologies like BigData
Skills	
Possessing specialized, technical skills for solving strategic problems	Defining a distributed database to make use of the advantages and avoiding the disadvantages of distribution
Evaluating alternatives with incomplete knowledge / information	Explaining a specific detail / behavior of a DBS without doing research about it at all
Developing, using and rate new ideas and methods	Design distributed algorithms or query executions to present and discuss them in class
Social competence	
Supporting the professional development of others	Get help from tutors about the tasks, who supports the professional development. Then, help fellow students in the same way (automatically)
Leading and arguing in area-specific and -comprehensive discussions	Discussing designed algorithms and today's technologies also in context to their advantages in specific use cases
Self-dependence	
Defining goals in application- or research-oriented tasks under social, economic or culture impacts; using appropriate tools and generate independently knowledge for them	Project work about a complex task with social conditions and defined budget, for example creating a Facebook clone

Table 1: Orientation aims - competence <-> possible coverage

The orientation aims will be used next time in chapter 4.5 in order to connect them with the learning objectives, which will be developed in chapter 4. The connection of the aims and the objectives results to an easy formulation of tasks concerning and respecting the requirements from both elements.

3.2.3 Learning Aims

Next, the learning aims are going to be described, which were determined by the thesis-examiners. The aims have been mentioned in chapter 1.4. Now they are going to be explained. These aims are split into four parts: The distributed relational database system (DRDBS), the Map-Reduce pattern (M/R), the comparison of both of them and BigData & cloud architectures.

For the DRDBS exercises, students shall handle and understand distributed partitioning (called sharding), replication, distributed-transactions and -query plans. These exercises will operate on the same dataset like the M/R-part. Preferable, also visualizing query-plans and demonstrating deadlocks are subjects on this topic.

On the Map-Reduce part, the students shall show their understanding about M/R and being able to write Map-Reduce programs / jobs. Also getting to know the limitations of Map-Reduce and alternatives to the same, by showing examples of PageRank- and DataMining-technologies, are important. Lastly, it is wished to emphasize the "Don't ship the data, ship the computation"-approach in the exercises. Among that, it is essential to challenge the students with tasks about, which parts of M/R jobs are locally executed and which parts are globally executed.

Next, in comparing both database types, it is going about to clarify the data distribution in DRDBS and DFS (Distributed File System) and to explain the differences of the relational query plan and the Map-Reduce job as well as the process of modifying data in DRDBS and in DFS.

At last, instructing students doing research about cloud architectures and Big Data technologies should also be part in the exercises.

The learning aims will be the fundament for the creation the concrete learning objectives, which in turn will be connected with the orientation aims of the previous chapter in order to formulate tasks.

3.3 Learning Methods

Exercises can be done best with a partner. Handling the tasks individually is not recommended, but depends in the tasks difficulty, error-quote and the student's self-assessment. A role play game on the first exercise / in the beginning of the lecture can ease the creation of little groups. Another learning method, which will be considered, is the constructive discussion with fellow students or partner. Dealing with a project task of higher complexity in a bigger group, with around four students, will challenge not only the students' technical-, but also their social-skills and self-independence through the group itself. Lastly, the biggest part of the students' work will be executed with the exercise papers, which contain problems and questions covering the learning-objectives, -content and is needed to be solved and answered.

The learning methods are describing, in which way the exercises, developed by this thesis, can be used in the concrete practice lesson. This point has been marginal explained in order to give the tutor much freedom in how he leads the practice lessons.

3.4 Exercise Design Pattern

The design-pattern for the most exercises consists of several parts. These parts are the levels of Bloom's taxonomy [taxonomy], each associated with some use cases about distributed DBS and marked, if they are mainly predominant for theory- or practical-tasks. Using all levels is beneficial due to have variable tasks, types of challenges and difficulties in a constant increase in each exercise. Every exercise should / will begin with a kick start to reason the topic and to motivate the students - even it will be already done in the lectures. The following table shows the design-pattern:

The focus should lie on the first two levels - remembering and understanding, since the other levels are built on these ones. The last two levels, evaluation and creation, should be kept theoretical, because the tasks of the underlying levels need first to be stable in order to correspond better with the practical, upper level tasks. This pattern describes exercises 'logically'. The real, physical structure will consist just of a theoretical and a practical part.

Type	**Use case / Orientation**	**Focus**
Motivate	Quotation, provocative question, conflicting statement, current event,...	:-)
Remember	Describing terms, definitions, ideas, concepts and being able to bring them in relation, explain their similarities and differences	Theory
Understand		
Apply	Developing distributed programs, databases and queries	Practice
Analyze	Inspecting features of DBS; detecting errors in transaction histories	
Evaluate	Scoring transaction costs; testing and judging DBS behaviors	Theory
Create	Design, construct and propose distributed architectures, algorithms and query-executions	

Table 2: Learning Objectives for Distributed Information Systems

This pattern will be used in order to develop a balanced number of different types of learning objectives in chapter 4. This means, that some learning objectives will be created to give students information or some objectives will exist to train their comprehension of the information they learned or to let them apply their knowledge in practice or to analyze already applied concepts and ideas – also in practice or to evaluate / test provided analysis studies on paper or lastly, to create / engineer new concepts and ideas, which are similar to the ones presented in this lecture and exercises.

3.5 Monitoring of the Learning Success

Furthermore, testing students, to validate their understanding about the subject matter and if the learning objectives are reaching their goals, is very important in order to give students an orientation and to keep them on the 'right path' [test_learnObj]. Validations, accompanying the exercise phases, are generally the solutions of the tasks. With the solutions students can test themselves, the tutors can confront and help the students with the right solutions or the filled exercise papers can be collected and evaluated. Important is, to provide in every case solutions, for any exercises.

Besides, one or two exercises can be occupied for doing presentations in front of class about some technologies and / or working in a group for analyzing (distributed) applications. Both methods can be evaluated in a verbal way. Additionally, creating a handout can be part of one of these exercise types, which increases the learning gain. A handout would contain either key data about a wider topic area or details of a single aspect like an algorithm. Besides, exams and exercise tests are another way to do evaluation in a written form.

Testing the students' learning success is very essential for a concept, which helps to develop existing and new learning-objectives. Without tests, it cannot be determined, if the students have learned the content and the learning objectives of a didactic exercise concept, which in turn the reasoning, meaning and sense of such concepts will be questioned.

3.6 Control of Success

Finally, the control of the success of this concept or the exercises will be shortly discussed. It is elementary to get feedback from students, who are learning more research-oriented and independent. They can give their opinion about single tasks, the difficulty and telling, what was best, worst, interesting and motivating or even what can be done better. Eventually, the feedback about the exercises can be told not only at the end of the lecture / semester, but on every second or third practice lesson. This will provide a more detailed feedback and simultaneously it will work like a valve for the students to eventually release easier their frustration and increase their motivation, because they get heard. The reason for doing this is, that the exercises will be new and not optimal for the students on the first run / iteration.

The control of success of the development of exercises is important as every other test phase in a project like for example, implemented code must be tested on its functionalities. The difference between control of success and 'monitoring of learning success' from the last chapter is, that the first one aims to the user of this concept. A user is a person, which develops learning objectives, exercises and implements programs with the help of the concept presented in this thesis. The user has to use the students' feedback in order to create an objective and critical opinion about this concept and can therefore change parts of it. This ensures, that the modifications on this concept will become improvements without initializing new issues.

It is not recommended to use the user's opinion and experiences alone and without the students' voice. This concept is suitable for users, who have none or little experience and knowledge in creating helpful tasks.

3.7 Summary

In this chapter the introduction of the concept has been started by discussing the didactic design. The basic conditions, abstract learning objectives, learning methods, design pattern, monitoring of learning success and control of success – all in regard of developing exercises – has been explained. Only the concrete learning objectives has been skipped, which will be caught up in the next coming chapter. Overall, this chapter kick started this work with some factors and characteristics considering the educational aspect.

4 Exercise Learning Objectives - Analysis Part II

Next, the learning objectives for each exercise are going to be described, which are based on the learning aims from chapter 3.2.3 and on the lecture's structure. Since the topic distributed information system is huge, there is a need to reduce the content in order to not overflow the students with information and not to demotivate them. Therefore, a horizontal reduction of the total spectrum will be performed to get single questions and problems to ask and to let be solved by the students. In case that some objectives are not covered in the lecture, an advance organizer [ad_org] can be used right before the respective task.

4.1 Overview

The objectives are similarly structured as the lecture. Thereby, three big areas are existing: DRDBS, Map-Reduce and cloud design patterns. The DRDBS-part consists of four minor parts. In the correct order, these parts are fragmentation, queries, replication and transactions on distributed relational database systems. The Map-Reduce-part consists of four parts, which are processing, limitations, alternatives and DataMining. Only the first minor part of M/R has been realized in this chapter. The last of the major parts is going about cloud and should contain just one exercise. Additionally, a fourth major part is being planned, which is the comparison of both DBS-types. This part will be placed before cloud design patterns and consists of the three minor parts 'data distribution in DRDB and DFS', 'queries versus M/R' and 'data manipulation in DRDBS and HDFS'. Unfortunately, also the comparison part is wholly not implemented in this chapter and can be caught up in the next iteration with the help of this concept.

All described LO-groups can be found in the appendix as exercises. Additionally, the LO-groups replication with DRDBS (no. 3) and processing with M/R (no. 5) are going to be explained in detail in chapter 5.

This concept provides in chapter 3 structural help in order to develop the learning objectives for the limitations, alternatives, DataMining of / with M/R and the comparison of DRDBS and M/R. The requirement in developing the skipped learning objectives is to have detailed knowledge about these specific areas. When the missing learning objectives are implemented, chapter 5 can be used as a guideline in order to create tasks with solutions, which are harmonising and integrated with the exercises being already implemented by this thesis.

The design and structure of a group of learning objectives can be looked up in the coming sub chapters. Following, it will be described shortly, how these groups of learning objectives came off. First, it is important to know and read the lecture's materials, which are going to be presented to the students. With these materials it is possible to construct objectives, which are representing the knowledge contained in the same materials. These objectives are mostly based on the first two levels of the Bloom's taxonomy – remembering and understanding - showed in chapter 3.4. The successful development of objectives for the next two levels – applying and analyzing – can be reached, by being engaged with concrete distributed software systems, reading their source code, books, documentations and opinions of other experienced people, who can be found upon others in mailing lists, and by writing, testing code on the same systems. This ensures, that the user of this concept gets enough experiences in a short time, which in turn are more or less reflecting the current practical situation and technological events. In order to develop the objectives for the last two levels – evaluation and creation – the user would need to develop his experiences through books like [DDBS] and [HDP_GD], which are covering theoretical background; through time for continuing to develop the created exercises and tasks of the first four levels and through the students' feedback, who are using actively the exercises and are engaged in the same field as well. In the end, getting ideas for new tasks depends on the user's creativity and success of earning the needed experience. The mentioned advices are the experience of the author of this thesis.

One referenced group of learning objectives in this chapter will be used as an example to explain to the reader, what the origin of the most learning objectives in this concept is. For that purpose the objectives for processing with M/R (group no. 5 in sub chapter 4.3.1) will be used. First, the definition of the topic will be started, then the reasoning and advantages should be given as a second learning objective. The third one, the topic's definition will be explained in context of DIS and its differences to other DISs. The fourth objective presents the advantages of these differences. The fifth and sixth objectives are starting to investigate features of a concept like in this example the shuffling phase and the combine function. The last two theoretical objectives are going about to use the concept on paper by explaining and noting steps in order to solve a given problem with the help of this concept (M/R). The practical objectives are starting first with the introduction of a software system. Then it is going about, using step by step the software system by solving simple tasks, but learning with every new objective new features like learning how to use a mapper, a reducer, how to sort, to read for example XML files. When the knowledge in using the software system in regard of this lecture is completed, the practical comparison with other systems can be started. In this case, students shall implement M/R jobs, which have the same results as output as the given SQL select-statements.

A counterexample are the objectives for replication (group no. 5 in sub chapter 4.2.3). Even if the LO-group has many parallels with the M/R processing in how it is structured, the ideas didn't come from thoughts, a mind-map or a brainstorm by the author alone, since there were also more lecture material available and about the topic replication more literature have been found, which were useful for exercises. The replication-objectives are based on the presentation slides on this topic and the two books [DDBS] and [MSSQL]. Both books contain a chapter each dedicated for replication.

Altogether, 6 from original 12 minor parts are existing and in each one learning-objectives will be written up. In each piece, theoretical as well as practical objectives will take place. After every formulation, a short table with the objectives will be shown and numbered. In the end, these objectives or minor parts will be assigned to exercise papers with the help of a final table.

4.2 Relational Distributed DBS

As explained in the overview, the topics around relational DDBS are partitioning, queries, replication and transactions. Relational DDBSs are similar the common known relational database systems like MySQL, Oracle Database, MS SQL, MaxDB. Their difference lies in using several machines to increase their performance, capacities and they are networking with within the cluster to accomplish the same goals and services as provided by the single-node DBSs like for example the ACID properties.

4.2.1 Partitioning - DRDBS

The first part is going about fragmentation. The definition of partitions, fragments and shards shall be taught. The next task is to describe the partitioning-methods and -criteria and their advantages. Going into more detail the difference between primary- and derived horizontal fragmentation must be shown. Also to let students describe use cases, which are beneficial for specific partitioning methods, will improve their understanding in this field. To point out the definition for correctness of fragmentation, does students help to simplify and structure their view of DRDBS a bit. Looking out of the scope and doing research about how Data Warehouses are built up and about methods for the bottom-up design, helps to further increase the knowledge about the distribution.

The practical questions are going to be solved with the use of MS SQL Server. Besides getting to know the DBS, it is firstly important to analyze its features like for example the support of different kinds of partitioning-criteria and -methods and how to apply them on a database. Next step is to offer students another partitioning task exclusively with the help of predicates. Like in the theoretical part, it is wise to show a bigger view of MS SQL Server and what it is able and not able to by, for example, comparing it with other DRDBSs like MySQL Cluster.

Theory			
1	Definition of partition, fragment and shard	2	Definition of partitioning-methods and -criteria
3	Advantages of typical partitioning-methods and -criteria	4	Definition of primary- and derived horizontal fragmentation
5	Describe use cases for different partitioning methods	6	Definition of correctness of fragmentation
7	Doing research about different DDBS like Data Warehouse	8	Definition of bottom-up design and global conceptual schema
Practice - MS SQL Server			
9	Analyze DBs according their partitioning-criteria and -methods	10	Apply a partitioning criteria on an own created DB
11	Partition a DB with the help of predicates	12	Experience the differences with MySQL Cluster or other DBSs

Table 3: Partitioning - DRDBS – learning objectives group no. 1

4.2.2 Query - DRDBS

The next group of learning objectives is going mainly about distributed queries. The first things getting to know are the additional difficulties and challenges of distributed query-execution. Next, the cost-factors need to be explained by the students, which will later be used in cost-calculations for queries. The simplification of queries is another task.

As a practical variant, making the cost-evaluation on real queries would manifest the knowledge, skills and experience earned about distributed queries and their challenges.

Theory			
1	Identify the additional difficulties for distributed queries compared to non-distributed queries	2	Explain the cost-factors and their effects of distributed queries
3	Evaluation of costs for distinct queries on distributed DBs	4	Explanation of semi-joins and its advantages
5	Simplification of queries		
Practice			
6	Cost-evaluation on SQL-queries		

Table 4: Query - DRDBS - learning objectives group no. 2

4.2.3 Replication – DRDBS

The third part of the learning objectives about DRDBS is replication. It covers the definition of replication itself and its advantages and disadvantages. Going into more detail, knowing how full-, partial-, or non-replication effect distributed DBSs on certain factors like query processing, reliability or concurrency control is also important. Another objective is to get to know the definition and the difference between local and global transactions. Same as for mutual consistency: being able to point out the advantages of weak and strong mutual consistency including their meaning. Transaction consistency is another form of keeping data synchronized and which is needed to be described by the students. Analyzing distributed transaction histories and detecting errors or determining the support of serializability or mutual consistency will be a task, which gives students a deeper understanding about synchronization of replicas. Updating replicas can be done centralized or distributed. Being able to explain their advantages is another goal in this part. The next point is to understand what eager and lazy update propagation is and to be aware of the definition of limited and full replication transparency.

The practical know-how is going to be taught with the MySQL Cluster, but mainly with the MS SQL Server. For the MS SQL Server, there is the need to know, which distribution methods are existing, how the server does replication, the meaning of roles a server can take, when or for which replication type it is necessary to propagate the primary keys, what increases or decreases the network traffic and database size and when the server uses different replication-agents.

Theory			
1	Advantages and disadvantages of replication	2	Effects of full- and partial replication
3	Definition of local and global transactions	4	Effects of weak and strong mutual consistency
5	Definition of transaction consistency	6	Analyzing distributed transaction history
7	Effects of centralized and distributed updates	8	Definition of eager and lazy update propagation
9	Definition of limited and full replication transparency		
Practice - MS SQL Server			
10	Methods of distributing data	11	Types of data replication
12	Roles like Publisher, Distributor, Subscriber	13	Cause of high network traffic and disk size
14	Tasks of different replication agents		

Table 5: Replication - DRDBS - learning objectives group no. 3

4.2.4 Transaction - DRDBS

Distributed transactions is the topic for another group of learning objectives. First, interpreting a deadlock scenario on DRDBSs shall give students the understanding of similarities to single-node DBSs. The definition of Open XA, 2PC and TPMs will be handled secondly.

In contrast, the practical tasks will aim on the use of 2PC with the help of MySQL-Cluster's XA transactions and locking functionalities of the NDB-engine and MS SQL-engine for distributed tables. Another objective is experiencing different isolation levels for distributed transactions.

Theory			
1	Definition and advantages of TPMs	2	Definition of Open XA and 2PC
3	Sketch deadlock-scenarios for DRDBSs		
Practice - MySQL Cluster			
4	Use of Open XA commands	5	Provoke a deadlock on an NDB-table
6	Identify software components in the MySQL Cluster architecture		
Practice - MS SQL Server			
7	Using different isolation levels for distributed transactions	8	Set up and explain XA transactions with MSDTC

Table 6: Transaction - DRDBS - learning objectives group no. 4

4.3 Map-Reduce

Map-Reduce uses another idea and type of distributed information system and is a concept for data-computational frameworks. Implementations of it like Hadoop are a mix of this concept and a NoSQL DBS. In this thesis, only one group of learning objectives about M/R has been completely developed.

4.3.1 Processing - M/R

The fifth exercise leaves the DRDBS part and starts with the introduction of the Map-Reduce as well as of Hadoop. In the first two objectives, students shall be able to explain the definition and advantages of Map-Reduce. The next two parts combine the last two objectives by discussing the relation between query- and data-distribution in Map-Reduce and its advantages. While the last four parts were concentrating on the mapping- and reducing-process, the fifth starts to go into detail on what is happening between these two processes. The terms like sorting, shuffling and combining are needed to be explained. The last theory-part is to formulate M/R-jobs in pseudo-code and by respecting the intermediate processes sorting, shuffling and combining, the student shall solve query related tasks.

The practical part starts with understanding the elementary commands in using Hadoop and later switches to discussions about selected details of Hadoop like sorting. The third and fourth part covers writing simple M/R-jobs and developing new M/R-jobs with the help of simple SQL statements. The last part discusses higher sophisticated features of Hadoop like implementing joins and secondary sorts.

Theory			
1	Definition of Map-Reduce	2	Advantages of Map-Reduce
3	Explain how the query- and data-distribution are working together in Map-Reduce	4	Name the advantages of Map-Reduce in regarding the mentioned approach
5	Explain when M/R does sorting and shuffling	6	Name the difference between combine and reduce
7	Formulate an M/R-process in pseudo-code	8	Understand how to use keys and values in order to sort
Practice - Hadoop			
9	Getting in touch with Hadoop and earn the first experiences in using it	10	Explain how Hadoop does sorting
11	Write simple M/R-jobs in Java	12	Implement simple SQL-statements as M/R-jobs
13	Implement joins as M/R-jobs		

Table 7: Processing - M/R - learning objectives group no. 5

4.4 Cloud Design Patterns

The last group of learning objectives are about cloud technologies and their service models. First, the cloud definition should be made clear. It is beneficial for the students to know, what cloud actually is not. Next the advantages of clouds should be understandable. Students must know the potential of clouds for bigger groups of users or customers. Another objective is the correct interpretation of cloud characteristics by describing the cloud's requirements. The next task is to show again the benefits of clouds, but this time through use cases. Using the model cloud-service-provider to classify existing cloud solutions does increase the comprehension- and analysis-skills. Also thinking deeper about the model and to test or to compare it with the corresponding needs in the reality will increase the evaluation-skills. The next task is about to let students show through best practices, what is behind the rather abstract term cloud: Best practices have evolved the service models, which contain knowledge, detailed background and descriptions of relations with each other. The task would be, by going into detail of these service models, explaining specific points like their considerations and relations, and give an own conclusion to these model(s). Best way to do so, is to relate this task with real, nowadays challenges (how does it look like in reality). Lastly, the students shall be able to pick up a service model and use it to show their discussion-, analyzing- and demonstration-skills in front of class and / or with his/her neighbor.

There is only one practical task, which aims more to the student's soft-skills than their technical competence. They would need to plan and implement a simpler project within a group over some weeks and discuss also more non-technical needs like social, fair-minded solutions fitting for their target group.

Theory			
1	Definition of Cloud	2	Advantages of Clouds
3	Interpretation of cloud characteristics and stating the basic requirements for a cloud	4	Illustrate the benefits for clouds through use cases
5	Inspecting and ranking cloud solutions based on a model	6	Detecting, analyzing differences of service-provider in a model (SaaS, PaaS, ...)
7	Reciting best practices in the cloud area and recalling, formulating the cloud's definition with the help of researches / references	8	Formulating considerations of service models and relations between them
9	Analyzing, discussing and demonstrating a service model		
Practice - Project Work			
10	Project work about any proposed and reasonable cloud topic or overhauling the Stackexchange usages by integrating them into a (unknown) DDBS. Beside planning and implementation, also social, cultural or financial aspects should be concerned. The result is a handout for every colleague about their DDBS and its features and a presentation of its functionalities with the given dataset(s).		

Table 8: Cloud design patterns - learning objectives group no. 12

4.5 Assignment of Competences

All learning objectives have been described in the previous sub chapters. Now it is going about to reconsider and connect these objectives with the orientation aims of chapter 3.2.2, which will make it possible to formulate concrete tasks easier and in a structured way. The formulation is indead more or less easy, which is why the part will be skipped, which discusses the creation of tasks and questions in a exercise. One example will be given in the end of this introduction. But this is not the end of the development process, since the solutions for each of these tasks are missing.

As discussed in this chapter, only seven of the DQR-aims have been selected, which were considered as appropriately fitting in the exercises. Four of the aims are covering knowledge- and skill-competences. The learning objectives are mainly based on these aims, since the exercises are from a technical and scientific nature. Most objectives will concern the basics about DDBS and will therefore not fulfill every time these competences, because these have a much higher demand, standard and level than just learning the basics.

Another aim is covering self-dependence and is of a great importance, because after the master study, the students shall be able to educate, motivate themselves, having a high self- assurance and be able to take over responsibility – all in order to acquire higher positions or stay in these ones. For example, the next step in the academic career is to receive a doctor's degree, where the examinee wholly depends on him/herself. Self-dependence will be tried to embed or to increase in every task or learning objective. Some of them will be mentioned here explicitly. But the biggest part will be handled in the last group of learning objectives 'Cloud Design Patterns' as a practical project work within groups. As the aim from DRQ says, the students will define their goals in application- or/and research-oriented tasks under social, economic or cultural impacts, using appropriate tools and generate independently knowledge. Details can be reviewed in the appendix where the 12th and last exercise can be found.

The last group of competence is the social one, where two of the DRQ-aims have been selected.

All above mentioned competences will be linked with some learning objectives and listed in table 9. The learning objectives (LO) will be written as a number in the first column of the table. This number consists of the number of LO-group and the LO-ID separated with a dot. The number of every LO-group can be identified either on their corresponding table presented in this chapter or in the table 10 in the ongoing sub chapter 'summary'. The LO-ID can be determined from the LO table of the corresponding LO group in this chapter. An 'X' in the field means, that the competence can or should be covered under the specified LO. Objectives, which are not covering a competence, will not be listed.

Following a short example will be given in order to show the interpretation of the table below: The objective 1.8 has marks at 'Specific details', 'Discussion' and 'Research' and is going about to define the bottom-up design and the GCS. A task, which covers all points could be formulated as follows:

> "The top-down design for the database have been introduced in the lecture. Do research when to use the bottom-up design process and discuss it with your neighbor. Use the book [DDBS] as a source. Describe in your own words how the global conceptual schema (GCS) for DBS can be created."

Discussion and research are obviously covered. The specific know-how are covered by the topic itself and by discussing the GCS and the design-processes, since this knowledge and terms are still on the newest state of awareness.

Learning Objective	**Competence**						
	Knowledge	**Skill**			**Social**		**Self-dependence**
	Specific details	*Technical problem solving*	*Complement incomplete knowledge*	*Method development*	*Discussion*	*Opinion*	*Research*
1.4					**X**	**X**	**X**
1.5		**X**		**X**			
1.6							**X**
1.7	**X**						**X**
1.8	**X**				**X**		**X**
2.2			**X**			**X**	**X**
2.3		**X**					
2.4							**X**
3.2					**X**	**X**	
3.6		**X**					
5.4	**X**						
5.5	**X**						
5.7				**X**			
5.13	**X**	**X**					
12.1						**X**	**X**
12.2					**X**		
12.4		**X**					
12.7						**X**	**X**
12.8	**X**	**X**	**X**				**X**
12.9	**X**	**X**			**X**		**X**
12.10	**X**	**X**		**X**	**X**	**X**	**X**

Table 9: Competence Coverage

4.6 Summary

In this chapter learning objectives has been defined and structured with the help of the learning aims in chapter 3.2.3 and the presentation slides of this lecture. The structure is an ordered list of minor parts, which represents together the topic of this lecture: DRDBS, M/R, Comparison, Cloud -> Distributed Information Systems. The following table summarizes the groups, which have been described individually in this chapter.

No	Origin	Minor group of learning objectives
1	Lecture: 2a. Distributed Relational DBS (DRDBS)	Partitioning - DRDBS
2	Lecture: 2b. Distributed Relational DBS (DRDBS)	Query - DRDBS
3	Lecture: 4. Distributed Relational DBS (DRDBS)	Replication - DRDBS
4	Lecture: 4. Distributed Relational DBS (DRDBS)	Transaction - DRDBS
5	Lecture: 5. Distributed Queries: Hadoop and MapReduce	Processing - M/R
6	Lecture: 6. Queries on Big Data	Limitations - M/R
7	Lecture: 7. Page Rank	Alternatives - M/R
8	Lecture: 8. Data Mining on Large Datasets	DataMining - M/R
9	Learning aim: Comparing data distribution of DRDBS and HDFS	Data distribution - Comparison
10	Learning aim: Differences between queries and M/R-jobs	Query versus M/R - Comparison
11	Learning aim: Comparing data manipulation of DRDBS and DFS	Data manipulation - Comparison
12	Lecture: 9. Cloud Computing and Design Patterns	Cloud design pattern

Table 10: Origin of learning objectives

The table shows, that the groups are covering quite the whole lecture. The LO groups from 7 (Alternatives – M/R) to 11 (Data manipulation – Comparison) has not been covered in this thesis. As mentioned before, implementing these LO groups can be catched up by another thesis work, project group or student assistant. He or she can use this concept not only to consider some aspects in the didactic teaching matter, but also to develop exercises, which will be similar to . For each group at least one exercise will be made and for each LO also at least one task will be made (1-1 translation). Its structure is made up by the corresponding learning objectives. How the learning content would look like and which additional aspects need to be taken care, will be described in the next chapter 5 Teaching Scenario – Planning.

5 Teaching Scenario – Planning

The teaching scenario is the last chapter about the exercise concept in this thesis. First the structure, then the order of this chapter will be presented.

The illustration shown below (11) contains the points, (marked with 'Yes') which has been taken over for the last part of this concept. The complete organization tree has been skipped, since it is the tutors decision how to organize the exercises. From the communication branch has been only taken the discussion over, because the consultation hours is also part of the tutor's teaching style as well as the (tele)-tutoring. 'Semesterapparat' means, providing students a platform for exchanging and receiving lecture materials. This is already provided by the university or the department of information science with software platforms like Moodle, LON-CAPA or just file-storage with SSH/SCP.

Illustration 11: Structure of exercise teaching scenario in [eTeach_exer]

Now the structure will be explained. First, the exercise depended topics (“Uebungsprogramme”) will be discussed, which are the choice of concrete DRDBS for practical tasks and the data model used in the tasks. These should be used as fundament to create a platform, like a DBS-cluster, which students can work and learn with.

Next, two exercises and their tasks and solutions will be formulated to give the understanding, how the concrete exercises should look like (“Aufgaben”). The discussion (“Diskussion”) will not be covered. Some ascpects of the discussion and competences has been described in chapter 4.5 Assignment of Competences to Objectives. Finally, the group work (“Gruppenarbeit”) will be handled to cover also the professional work in a team.

5.1 Choice of Distributed Database Systems

Using DDBSs in practical tasks is obligatory. This sub chapter presents the choice of DDBSs used in the exercises.

The decision, which databases are going to be used in the practical exercises, was actually already made in the previous semester. Currently, three DBS are going to be used: MySQL Cluster, MS SQL Server and Hadoop. There are existing dozens of other distributed DBS or frameworks, which are managing multiple single-node DBSs to offer clustering. All of them are worth to take a look into, since even if they may not offer the best solution or performance, their features can fit to these exercises for educational purposes. Most of them are shortly described in the last exercise paper 'Cloud Design Patterns', which can be reviewed in the appendix.

The table below (11) shows a comparison between the three DBS. Another DBS, called BigCouch, was also added into the table for the demonstration purpose to show, that other DBS are interesting, too, and have as well a lot of features / qualities fitting for the exercises.

The number of features is set under 15 or 20 to avoid high complexity. It also decreases the amount of time for analyzing the DBS and helps to concentrate on a few basic key data of these DBS for having a first impression. The features / key data are mainly aiming to verify their node-clustering – checking if it is really a distributed DBS – and its usability in context of educational matters.

Beside the DBS-type, the information, how the DBS are doing sharding and replication, is significantly for distributed systems. Next the synchronization shows, how well the DBS are managing parallel data reads and writes. Transparency means, how less the clients or the user themselves must know about the specific nodes of the cluster. A data-API makes sense for teaching purposes to show easily the 'data-node-to-data-node'-communication. If such an API does not exist, the more important it is, that the DBS is open source. This allows to analyze or change the code. Else the possibilities in understanding the internals of these systems are restricted in changing their configuration files, interpreting the log-files and reading books or white papers. Testing failover functionality shows, how the DBS is going (to try) to save data in emergency cases. The popularity, OS[3]-support and hardware requirements are tending to motivate students for installing the DBS on their own systems and test them or learn through them. The availability of literature and the experience of the author increases the preparation and creation of new exercises based on these DBSs.

3 Operating System

Feature / **DBS**	**MySQL Cluster**	**MS SQL Server**	**Hadoop**	**BigCouch**
DBS-type	In-memory DRDBS	DRDBS	M/R with DFS	Document-oriented cluster
Sharding	automatically between node-groups	Publication with vert./hori. filter[4]	(auto-) file-block-sharding	auto-sharding
Replication	Node-group replication	DBS/Partition-level-replication	file-individual replication	DBS-level-replication[5]
Synchronization	read-commit [6]	different levels of ACID [ms_acid]	File system read/write-locks[7]	State-less
Transparency	Full	Full	Full	Full
Data-layer API	C NDB-Api	None	None	None
Open-source	Yes	No	Yes	Yes
Fail-over without data loss	Multi-master [8]	Multi-master with Merge-Replication[9]	NameNode as SPOF	Multi-master
Popularity / standard	++	++	++	-
Windows/Linux	Yes/Yes	Yes/No	Yes/Yes	No/Yes
Hardware-requirements	Low	Middle/High	Low	Low
Documentation/ Books	++	++	++	-
Experience	+	-	+	+

Table 11: DBS comparison

5.2 Data Model

To let students use (distributed) databases in an easy way, there is need in providing a dataset. The more complex (having many relations, indexes, functions, ...) the dataset is, the more functionalities of the database system types can be shown. The larger the dataset is, the bigger impact it has on the DBS and its performance and advantages, but also the limitations can be easier shown. Following, several datasets are described, which came into consideration.

4 [MSSQL] (chapter 18 Publications and Articles)
5 [couch_api]
6 [mysql_ref] (chapter 18.1.6.3 Limits Relating to Transaction Handling in MySQL Cluster)
7 [hdp_lock] (method public void write(un)Lock())
8 [mysql_ref] (chapter 18.1 MySQL Cluster Overview)
9 [MSSQL] (chapter 18 Multiple Publishers with Multiple Subscribers)

5.2.1 Mysql Employee Dataset

The MySQL community provides some databases for training and testing purposes. One of them is the employee dataset and is under the Creative Commons license available. It contains over 6 '1-n' relations with several tables like the employee-table with over 300.000 tuples and the salary-table with over 2.8 million tuples. The database is normalized and it is easy to install on MySQL databases, but the SQL-insert scripts can also be altered for other specific relational DBSs. Last but not least, it is well documented and the community is welcoming questions and is offering help.

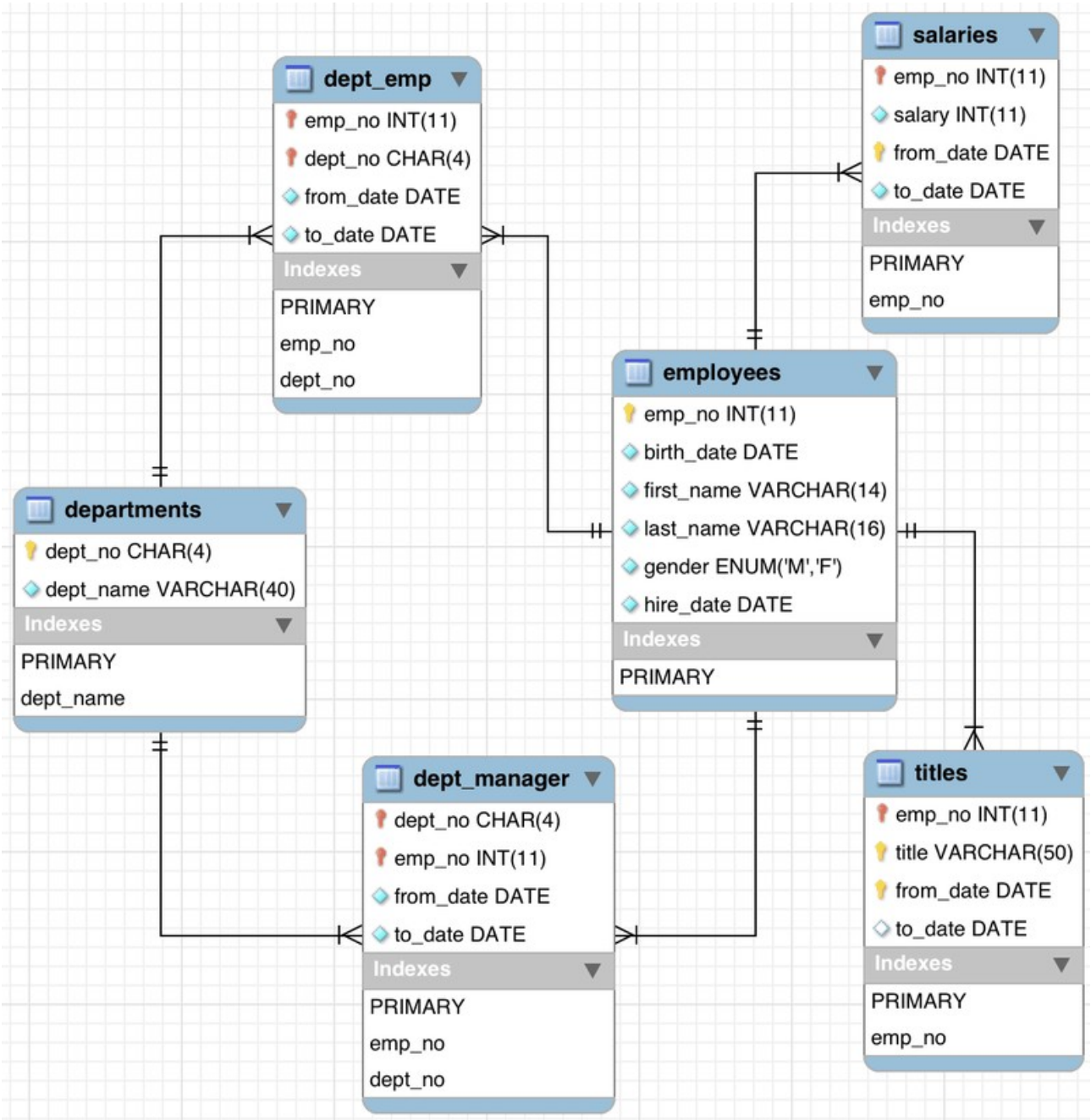


Illustration 12: Mysql employee structure. Source: dev.mysql.com date: 10/25/2015

5.2.2 Stack-Exchange Dataset

Another dataset used in this work is the one from Stack-Exchange. Its data is based from the productive websites of Stack-Overflow, has at least 8 tables, 15 '1-n' relations and is also, like the MySQL employee DB, licensed under Creative Commons [sedump_struct]. The websites offer people with similar interest the possibility to exchange their experience, ask questions, opinions and get answers. Stack-Exchange serves several websites, each for a different category like one for Linux users, another one for coffee drinkers [sedump_data]. These datasets have all the same structure, but contain different content and are of different sizes. For example the dataset 'Stack Overflow' contains over 9.5 million questions and 16 million answers and one of the smallest dataset is about coffee, which has just 293 questions and 468 answers. As expected, the database contains a lot of full text, which for example can be parsed and searched for specific words. This is useful for showing some features of NoSQL or Map-Reduce databases. The data is provided as XML files, which is easier to read by NoSQL databases. To demonstrate how distributed Map-Reduce programs are reading and parsing XML files, it is beneficial to let students know the first steps of processing of distributed files or data. Another advantage is, that there are pre-built MS-SQL-queries for that database already online available [sedump_query].

A big disadvantage using that database is, its XML format and incompatibility to import the data into regular database systems easily. A third party or self developed converter is needed to make this possible. This cost time, but when this is done, this concept contains a free, huge, complex, well tested dataset and the development of the VIS lecture will not be slowed down because of the dataset's limitations. The following illustration shows its data model.

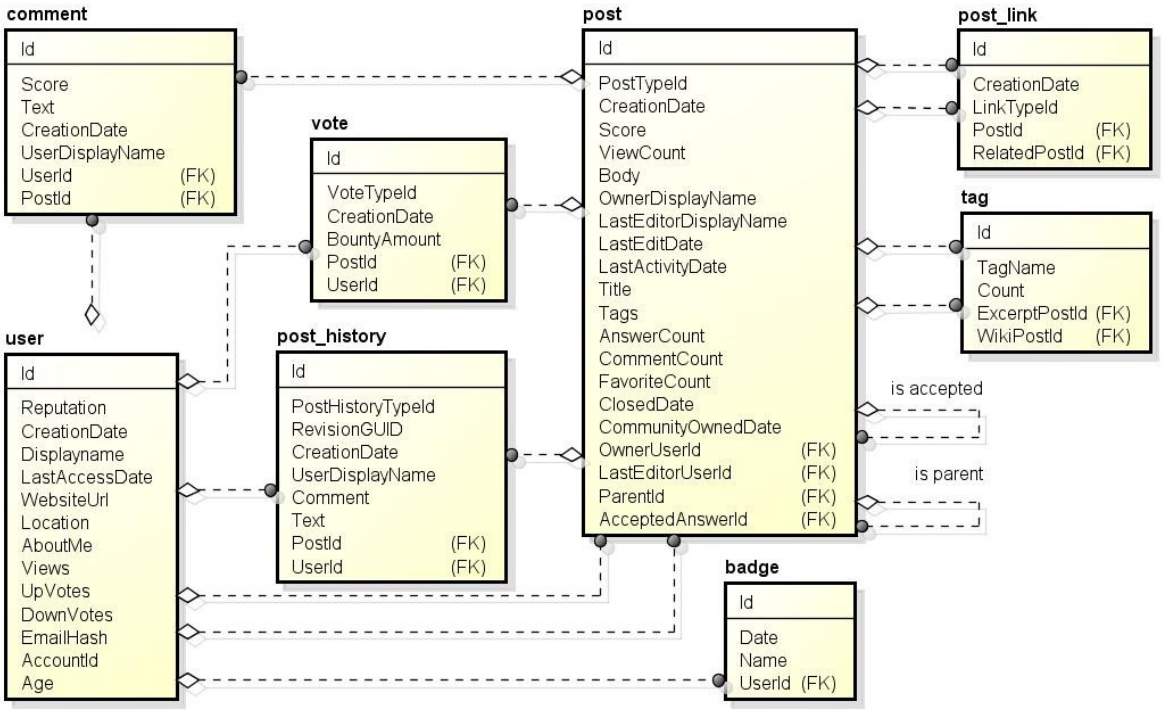


Illustration 13: Stack-Exchange data dump ER model

5.2.3 Self-made Dataset

Creating own datasets for the exercises has the following advantages. First, it gives unlimited space for the fantasy to create a lecture-optimal and easy understandable database, matching to any scenario. Next, the data format can also be chosen freely and because of that, at least one database system will not depend on the dataset's format. Finally, there wouldn't be any licensing problems using a self-made dataset.

One of the disadvantages is the time needed to model the dataset and insert the data. Using a random data generator would save time, but the quality of the dataset would decrease. The other drawback is, that the new data model would need time to be well proved and to become stable.

5.2.4 Dataset Comparison

The datasets, which were taken into consideration were described in the previous sub chapters individually. The following table shows a comparison of some features and facts of these databases or datasets. The character 'V' in the table stands for variable, which can be implemented in different volumes. The more effort will be used, where the features are marked as 'V', the more time is needed in the sum for implementing or creating the dataset. For the Stack Exchange data-dump only the datasets 'coffee' and 'unix' were evaluated. Note that the dataset 'Stack-Overflow(.com)' is over 20 GB huge and exceeds the reasonable amount of tuples.

Dataset Key Data	**Self-made datasets**	**MySQL**	**Stack Exchange**
License	++	+	+
Size on file system	V	~160 MB	2,8-604 MB
Tuples	--	~3,9 million	7.000-1,4 million
Number tables	V	6	>8
Number relations	V	6	>15
Contains full text	V	No	Yes
Format	V	SQL Script	XML
SQL portability	V	++	0
M/R portability	V	+	++
Time to create and install (on 1 DBS)	--	++	+
Stability	--	+	++
Maintenance	--	++	+
Queries available	0	+	++
Documentation	0	++	+

Table 12: Dataset comparison and evaluation

5.2.5 Dataset Selection

Since of its free license, stability, MS SQL queries and huge data as plain text, the Stack-Exchange dataset is being chosen for creating the exercises. The flexibility of this data model in providing different data sizes (unix, coffee, math, ...), scales with the available infrastructure and cluster size of the university. Especially the body text of the posts-table is ideal for using Map-Reduce on them, because the M/R's advantage is to do efficiently full scans on huge amounts of data. The DRDBS part benefits from the already existing MS-SQL queries, which also can be transformed into M/R-queries to have the counterpart in the exercises. When in the tasks M/R-jobs and DRDBS-queries are doing the same data-request or -manipulation of the same dataset, the comparison of these two technologies would be easier seen by the students.

5.3 Creation of Exercises and Tasks

This sub chapter shows two examples, how the exercises would look like and which learning content would reside in them. The first one is about replication with DRDBS and the other one is going about processing data with M/R. For each learning objective at least one task will be created.

The tasks are split into two groups: the theoretical and practical tasks. First, the theoretical ones are based on the materials and references on the lecture and on additional sources, literature, which are covering the concepts of this topic of DDBS. The practical tasks consist of simple hello-world programs to face the technologies and later to understand their fundamental setup. It is ideal to let students see, how the databases are working and serving the client's requests. The two database types, SQL and NoSQL, having different approaches to do the data manipulation and query execution. Their differences, similarities and relations can be difficult to understand. Therefore the use in all exercises and programs of the same datasets is beneficial for students in seeing the parallels and analogies of the database system types.

5.3.1 Replication with DRDBS

The third exercise paper will cover the topic replication on DRDBS. The tasks and solutions will be described in detail. The tasks are based of the objectives described in chapter 4.2.3 (on page 23) and the competences assigned to some to these tasks, which can be reviewed in chapter 4.5 on page 27.

First, the students shall understand the definition of replication and be able to name its advantages and disadvantages. Generally speaking, replication is distributing copies of fragments, called replicas, to other nodes. If one of the replicas will be missed, because of reasons like hardware-, network- or service-failure, a copy of an available replica can be made and be propagated to another node. This ensures reliability. As commonly known, nodes can work in parallel. When nodes contain the same data through replication, they will also be able to query on these data at the same time. Thus, several incoming queries can be distributed to the nodes to increase performance. One of the reasons, why databases become huge and have a high request is due a global demand. In such cases, nodes with their replicated data can be distributed geographically near their clients to decrease latency and increase scalability. Sometimes, also applications or use cases need to afford several copies of its own data for example because of security reasons.

If a database can already handle this requirement for applications, less developer effort would be needed in creating such applications. A disadvantage on the other hand is the additional required disk-space, hardware cost for new nodes and eventually licenses like for operating systems. This overhead can cause a lot of costs. Besides, the replicated data cannot be changed at the same time by several queries. To keep the data in an acceptable state, synchronization processes are needed to encapsulate the database and global locking functionalities for all nodes have to be implemented [DDBS] (chapter 3.2.5 Allocation Alternatives).

After the definition of replication, the next task will be about to discuss how DRDB-systems get effected by replication with certain factors. Although, it will be differentiated between full-, partial- and non-replication.

Next, a simple recognition of a fact can make the understanding about the whole topic easier: Since partial-replicated databases are more realistic, it can happen that some tuples are not replicated. Queries on these tuples alone will be considered as local transactions and will be much easier to execute than on a single-node DBS. Therefore the question shall be asked to students, what is the difference between local and global transaction.

Another term to understand is consistency in replicated databases. In this task students shall define weak and strong mutual consistency. Mutual consistency or constraints are keeping logical data in an acceptable state by providing methods to change their corresponding physical data accordingly, even if they are residing on different nodes. Strong mutual consistency equals with the data consistency on single-node DBSs: After the transaction is finished, physical data is synchronized. This is difficult to implement on distributed systems, while keeping in the same time the performance for these updates high. Therefore, a weak mutual consistency is sufficient, which promises that the DBS must enable the converging process at an unspecified time, even if the physical data after a transaction is diverged from their logical data.

While the past task covers single transactions not executed at the same time, students shall now be confronted with the problem of handling multiple transactions, which need to change the same data. Keeping this concurrency controlled is called transactional consistency. Beside this definition, the interpretation of transaction histories should be taught and the ability to identify the history's serializability and the mutual consistency.

Based on the knowledge of the previous two tasks, the students shall need to understand the two update propagation types. They must recognize that

> "Eager techniques typically enforce strong mutual consistency criteria" ([DDBS] (chapter 13.2.1 Eager Update Propagation)) and
>
> "Lazy propagation is used in those applications for which strong mutual consistency may be unnecessary and too restrictive" ([DDBS] (chapter 13.2.2 Lazy Update Propagation)).

The update management strategies are tools to keep the corresponding consistency types intact.

Beside the propagation methods, the updates can also be differentiated by their location, where they are needed to be performed. Do they always need to be executed on a specific chosen node (single master) or can every node accepts the incoming update transactions (multi master / no master)? Asking for the advantages and disadvantages of both approaches is also fitting in term of this whole topic. Centralized techniques in general speaking are easy to implement and do not need a global synchronization process to handle changes on multiple nodes. This also assures, that there will be always one copy, which is definitely up to date [DDBS]. On the other hand the distributed technique can do a full load balancing for write operations and provides high availability [DDBS]. One useful detail to let students know is, that

> "In some techniques, there is a single master for all replicated data. We refer to these as single master centralized techniques [...]. In other protocols, the master copy for each data item may be different. These are typically known as primary copy centralized techniques." [DDBS] (chapter 13.2.3 Centralized Techniques).

The next task would cover the distinction between limited and full replication transparency. Students should not delve much into detail, but know at least, that clients at full replication transparency must address the host by themselves, which contain the requested data, while at limited replication transparency clients are using the transaction manager to get the address, to where they can send their queries. The transaction managers are distributed programs on client- and server-side. The server-side TM provides information to the client-side TM, about which query can be supplied to which node.

To summarize the last three tasks, the next one will be going about to describe use cases for each of the four update management strategies. Students can discuss scenarios with their neighbor(s) and/or do research by their own. For example lazy centralized databases can be used for hosting wiki-sites, because the writes compared to read-operations are lesser and the updates to all replicas is acceptable to take one or two days. Eager distributed databases are fit for maintaining mailing-boards or forums, since write operations are a frequent task and high response time is also highly demanded there.

The final theory task is about to think how automatic and manual distribution effects a running DBS. Automatic replication is, when the DB-user just sets up a database as he/she would do on a single-node DBS. The replication process would run in the background transparently. This makes the usage easier, more familiar and use-case independent. The manual configuration of the database on the other hand makes a finer granular use of distributed advantages. This can be for example, what the user can decide, which tables need to be provided by all hosts or which ones can just be hosted on a single node or how to handle constraints much more efficient. The cost of this configuration is of course the developer-time, the need of DRDBS know-how and to formulate more complex queries, which in case of automatic distribution or auto-sharding, the DRDBS would do for the user.

The first practical task is to describe the cluster setup on the current MySQL Cluster in this lecture. It would be best to show the setup as a diagram, model or draft. Currently the architecture looks as follows: vmvis[1-4]: DataNodes; vmvis[5]: ManagementNode + SqlNode.

Next is to define the replication type for this DBS by research / investigation. At this point it will be shortly described, that MySQL Cluster does a

> "synchronous replication through a 2PC mechanism" [wiki_mysql].

A good test to examine the student's overall understanding is to let them list differences between a single-node DBS and a DRDBS. These two DBSs should be similar in order to really see more distribution based differences rather than DBS specific ones. Therefore the MySQL server and the MySQL-Cluster are appropriate for this test. The following solutions are developable to stretch this task: First, only read-commit transactions are supported on the cluster and write-commits have not been implemented yet, not because of technical limitations or big performance losses, but because the cluster is still under development and needs more time and effort to be become more stable. Another drawback are many restrictions regarding scheme definitions like a maximum of 32 key-attributes in a table or a maximum row-size of 14,000 bytes. Such limits are typical for memory-DBS. [mysql_ref] (chapter 18.1.6 Known Limitations of MySQL Cluster).

All next tasks are about the MS SQL Server and its replication features. The first one is handling the two provided distribution methods in order to have an overview and later to differentiate easier, which type of replication is not handled in this exercise. The skipped one are distributed transactions, which is realized by a 2PC-protocol and a TM called Microsoft Distribution Transaction Coordinator (DTC). Obviously, this type of replication is an eager distributed replication. The distribution methods, which the students will be confronted with, is the data replication. It is a lazy / asynchronous centralized or distributed replication, which allows nodes to have different states of the same replicas at the same time.

Secondly, the data replication itself can be differentiated in how the changes are handled technically. This helps students to have a deeper understanding of a DRDBS and how details or key data have an effect on a whole system. While with the help of the transaction logs the system can decide, when to start the replication process, the table-triggers on the other hand will start the same process right after a modification by the client.

Like in [MSSQL], after discussing the functionalities, it is appropriate to show next, which nodes are having which functions. In MS SQL Server the architecture is described with the help of roles and the two data units publication and article. Students shall be able to name the publisher as a source database and master for his own publications / replicas. The distributors are nodes, which are receiving the modifications from the publishers and are routing them to their subscribers. The subscribers in turn are providing the publications, mostly as read-only copies to clients. It is important to know, that these roles are not node dependent, but publication dependent. In [DDBS] this method is referred as "primary copy centralized techniques". A publication is a definition of a set of articles and the replication settings. These can be defined on every node, which will then become a publisher. Associated with a list of distributors, a publication will become active. Subscribers can register for publications on the distributors. An article is a table or a part of it. The parts can be defined as the horizontal- or vertical-fragments.

The next two tasks are concerned for specific details. One asks the students, when the MS SQL Server requires a primary key for the replication process. It should be understandable, that transactional replication (don't mix-up with distributed transactions => 2PC) need a key, since this type of data distribution is made due transferring rows only and they need therefore a unique identifier.

The other detail is about how to limit the network traffic or database size. This enforces students to think in another aspect of replication again - this time practically. They need to show that reducing the amount of subscribers, will also reduce the number of replicas (less disk space) of the same data and therefore cause less synchronization over the network. Another possibility to reduce the network load only is to distribute procedures instead of tables as articles. Procedures can contain SQL-queries to modify data on each node, where it was transferred to. This is the so called 'Don't ship the data, ship the computation'-approach and does fit for Big data tasks.

Lastly, a bigger part of the architecture will be aimed for students to do research by their own. They will need to look up in the literature about several available agents, list and describe them and their purpose. Summarizing, the log reader-agent is used for transactional replication (classic replication), which uses the log files to identify the rows needed to be sent. Every publication based on that type must be initiated with a snapshot replication to copy the already existing rows efficiently to the distributors. The snapshot agent can be used for asynchronous, long-term or one-time tasks. The merge agent enables the subscribers to write on the data, they get from the publishers / distributors. The resulting conflicts can be handled by the agent through time-based or custom-based priorities.

5.3.2 Processing with M/R

The fifth exercise paper will cover the topic processing with M/R. The tasks and solutions will be described in detail. The tasks are based of the objectives described in chapter 4.3.1 (on page 23) and the competences assigned to some of these tasks, which can be reviewed in chapter 4.5 on page 27.

The first exercise concerning Map-Reduce is going about its introduction, basics and the use of the programing-framework Hadoop. The first question, the students shall ask themselves is, what Map-Reduce is and how does it work. The technical details and implementation will be skipped at this point and can be discussed in the practical tasks. Students shall learn: The logical view of M/R is the following: “map(k1,v1) -> list(k2,v2) -> reduce(k2,list(v2)) -> list(v3)” [MR_WHITE] (chapter 2.2 Types) and [MR_WIKI]. The idea is to assign input data (v1) a key (k1) like a primary ID, filename, email-address, ... - the key can even be part of the data. These key-value pairs can be processed separately and parallel on different nodes. The result for each pair will be a list of new key-value pairs (k2,v2). The lists can be empty or contain just one or several elements. This output can already be printed out as an end-result if wanted. Optionally, a reduce-process enables to aggregate the key 'k2' and group the lists of values assigned with the same key as one list of values. After that, the user-defined code will be executed to transform the key-list-pairs into another list with different or changed values (list(v3)). The v3-values can be of any type, structure or data. As like as the map-process, the reduce-part can also be executed in parallel for the k2-list(v2)-pairs. The aggregation process, which is happening between mapping and reducing, does sorting after the k2-key and is propagating the intermediate result to the 'right' reducer, which is called shuffling. The details of that process is implementation depended and will be discussed in the practical tasks.

The second task is going about to discuss the advantages of M/R over DRDBS. It is important, that students should clarify, that both concepts have their value to exist and that the most (D)RDBS use cases or usages are still do fit (much) better on relational systems than on M/R. The reason is, that M/R requires a different thinking style in managing data. On relational systems concepts are existing and / or are much higher developed, like consistent transactions and concurrency, static schemes as table-structures and data as their tuples, views, easy query language, query-optimizer, database logs (redo, undo, ...), backup and restore strategies, data migration options, static analysis tools, sophisticated security and privilege options and many more [wiki_DB]. The M/R concept on the other hand has a much different approach. It is from the beginning designed for working with several nodes to achieve a greater processing performance on 'flat' data (data which has no references to itself like the RDBs has them as constraints). For that matter, M/R surrendered in taken care of the RDBS features mentioned above. Also distributed RDBSs have problems to implement the features of their single-node DBS brothers. For this reason, the real advantages of M/R do correspond with the advantages of distributed programs: load-distribution, high availability, better responds time through geographic locations, higher reliability.

The next two tasks cover the M/R-specific approach compared to regular distributed programs and its advantages. A way to realize the differences is to analyze the query- and data-distribution of M/R. Students shall understand, that data will already be prepared for querying on it by distributing it among the cluster as soon the client starts the insertion process. After the client starts a query-job, the only thing, what the cluster need during that process, is to distribute the job over the cluster only – not anymore the data.

The advantage of the method above is to process huge amount of data faster, since the data do lie already distributed among the cluster nodes, while the job-program gets started. Else, also the data would need to be distributed over the network at job start, which will be expensive, especially when data are much bigger than the job-program itself. Another benefit is, that other jobs can be executed on the same data, since this data don't need to be retransmitted and distributed among the nodes. This keeps the data location on the cluster's responsibility, which makes the cluster more to a database-like (NoSQL) environment rather than a framework for distributed computation only.

The next two tasks are going about some details: The sorting-, shuffling-, combine- and reduce-process. First sorting and shuffling will be explained. The sort process is included in the shuffling process and will be done on the results of the map-tasks right before they get written to disk. In the rest of the shuffling, the reducer nodes are downloading their own parts of the map-phase's results from the mapper nodes as shown in the image below. The following quotation explains, that going into more detail, will make the description implementation depended, which is why at this point it is for the students enough to describe shuffling:

> “The shuffle is an area of the codebase where refinements and improvements are continually being made, so the following description necessarily conceals many details (and may change over time; this is for version 0.20). In many ways, the shuffle is the heart of MapReduce and is where the “magic” happens.” [HDP_GD] (chapter 6 Suffle and Sort).

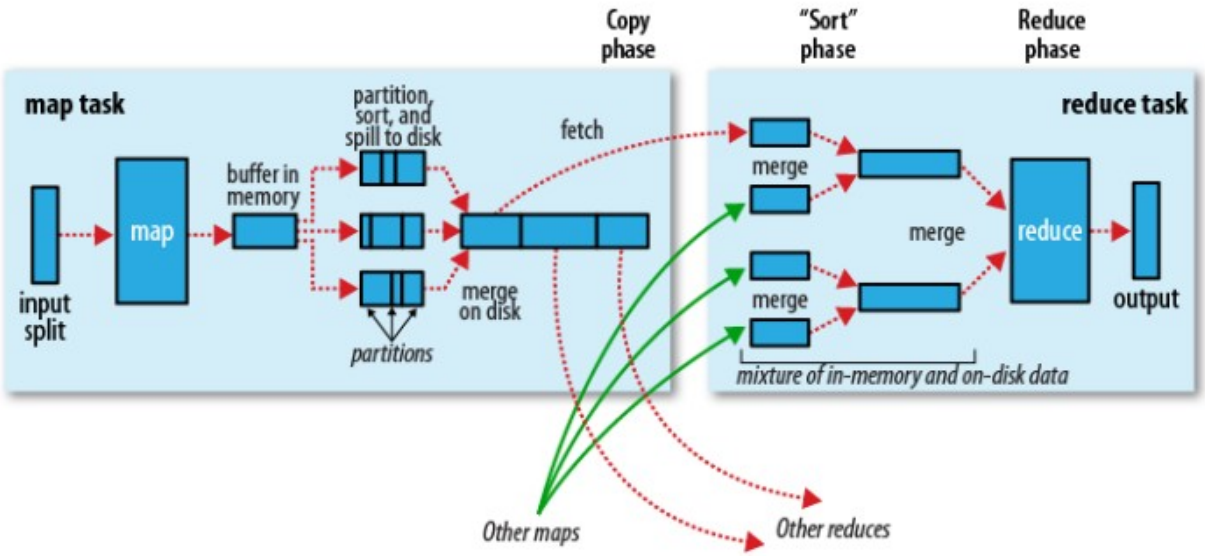


Illustration 14: Shuffle and sort in MapReduce [HDP_GD] (chapter 6 The Map Side)

The other task covers the combine process and its difference to reduce. Combine and reduce are doing the same thing in different time during the whole M/R process. Combine will be done with the sort-process right before the map-results will be written onto disk (spilling partitions to disk). Reduce will be executed after the merge/"sort" process. Both are merging the pairs with non-unique-keys to pairs with unique keys like described in the first task of this exercise.

The last two theoretical tasks are going about to apply the previous tested knowledge by letting students write M/R pseudo-code. The first pseudo-code should just show a simple M/R procedure. The student will get an empty map- and a reduce-methods and have to fill his/her code in. The data can be of any kind - matters is, how the arguments in the map-method look like. In this example an object of class 'Person' will be used as a value-argument in the map-method. A person-object has a getFirstname() method. The task is, to get through the M/R-processing the following result:

```
1  {"[a-h]", 4}: {Donald, Daisy, Dagobert, Goofy}
2  {"[i-q]", 2}: {Mickey, Minnie}
3  {"[r-z]", 3}: {Scrooge, Tick, Trick, Track}
```

The following pseudo code demonstrates a possible solution:

```
map ( Int key, Person value, MResult r)

    String name <= value.getFirstname();
    char firstLetter = name.charAt(0).toUpperCase();
    String newKey;

    if(firstLetter < 'I')
        newKey = "[a-h]";
    else if(firstLetter < 'R')
        newKey = "[i-q]";
    else
        newKey = "[r-z]";

    r.write(newKey, name);

reduce ( String key, String[] value, RResult r)

    String newValue = "";

    for(int i = 0; i<value.length; i++)
        newValue += ", " + value[i];

    newValue = newValue.substring(2);

    r.write(key + ", " + value.length, newValue);
```

Illustration 15: M/R example for using map and reduce in pseudo code

It should be noted, that 'Int key' contains the position in bytes of the value in the 'person-file' and is irrelevant therefore in this example. How the map-job does get the key and the value, is not part of this discussion.

The other task is going about the change of the previous code for sorting the output after the first names as follows:

```
A-H: 1
Daisy: 1
Donald: 1
I-Q: 1
Goofy 1
```

```
R-Z: 1
Scrooge: 1
Tick: 1
Track: 1
Trick: 1
```

The pseudo-code below demonstrates, how students can solve this task:

```
boolean intervalAdded = false;

map ( Int key, Person value, MResult r)

    if(!intervalAdded)
        r.write("A-H",1);
        r.write("I-Q",1);
        r.write("R-Z",1);
        intervalAdded = true;

    r.write(value.getFirstname(), 1);

reduce ( String key, Integer[] value, RResult r)

    int sum = 0;
    for(int i = 0; i < value.length; i++)
        sum += value[i];

    r.write(key, sum);
```

Illustration 16: M/R example for using the sort phase

One or two other tasks can be inherited from the last two ones. The questions would be first, why the reduce-function has to do 'sum += value[i]' instead of putting simply value.length to the 'r.write'-call and second, why in the results the values of the intervals are not always just contain '1', but happens be '2' or '3' and what is the requirement to get different values then '1'.

Next, a few practical tasks will be discussed. First, to warm up, it will be going about understanding the basics, to get in touch with Hadoop and how an M/R job can be started on a cluster. The student will get a script file, which shall be executed. With the help of additional documentation, this script can be understood by the students. Also the input and output data must be inspected in order to know what the program is actually doing. For example the script must show, how to set the job configuration from the outside, without coding it into the Java program. Additionally, it is helpful to demonstrate, how to use the HDFS on a CLI. This task serves also to check, if the student's platform is functioning and if he can develop his own M/R jobs on his/her environment. The following code piece, gives a short demonstration with an explanation:

```
inFormat="-D
mapreduce.job.inputformat.class=org.apache.hadoop.
mapreduce.lib.input.SequenceFileInputFormat"

hadoop jar $example $inFormat words wordcount

hdfs dfs -cat wordcount/\* |less
```

Illustration 17: Piece of bash code piece for starting an M/R job

The first line defines a configuration setting. The second starts the M/R program. It is a Java program saved in a jar file and its path is saved in turn in the variable $example. $inFormat is the configuration setting of the first line. 'words' and 'wordcount' are filenames in HDFS which serve as input and output data. $inFormat specifies, that the input file 'word' must be read as *SequenceFileInputFormat*, a format, which men don't understand. The last line demonstrates, how to investigate the output file on the HDFS.

The first real practical task can pick up some elements from the script on the last task. An example is to ask, when does using the *SequenceFileOutputFormat* make sense. This question is a typical research task. The solution is more or less short: *SequenceFiles* are used to store binary key-value pairs. Its first byte-sequence is 'SEQ' as a magic number to show people, what kind of file it is, when they use programs like 'head', 'less' or 'cat' on it. With 'hadoop fs -text numbers.seq' Hadoop provides an easy way to look in such files and identify the keys and values. This command and the format can be tried out in this task by creating an own sequence file or changing one from an existing M/R job.

The third task would be going about to discuss how different Hadoop with the M/R concept is. Also, the similarities are needed to explain for seeing a better relationship. One good point for making a dedicated task, is the sort process. The sort process of Hadoop is very similar to the M/R sort. In the task, the students could get a list of strings like city names and three or more criteria how these strings should be sorted. Obviously, these criteria should be implemented in the map-method. In doing this, it is common to create a prefix as a sort-key, which can be concatenated with the city-names and put them together as a key of the map's result. Between the map and reduce, the shuffle phase will sort the city names appropriately, after the prefix, which is in turn based on the sorting criteria. When the keys arrived in the reduce-method, their prefix can be removed and the blank city names can be put as keys of the reducer's results. This time, the results will not be sorted and the ordering from the shuffling will prevail.

XML Processing

The next task is more concentrating on understanding, what happens before the map phase. In this case, the dataset from Stack-Exchange can be used, which are formatted in XML. The task is simply to use an already predefined class, which prepares the map's input data into key-value pairs. The student shall revise the source code of the job program and try to understand the process. They must realize, that the pairs are records generated from the *InputSplits*. Each split will be executed by one different map-task. The *InputFormat* is responsible for directly reading the input file with the help of the split object and creating the key-value objects in order to be able to pass these objects to the mapper running on the same node. Another task can be inherited from it by asking the students to create their own formatter, which needs to be able to parse a simpler format than XML. And later, a mapper and a reducer can be implemented in order to fulfill a specified task and of course to test the formatter.

SQL-selects as M/R

Next, it is going about to realize three SQL-statements as M/R jobs. The following examples show, how M/R can be used. In each example, the corresponding SQL statement will be displayed. The coding will be skipped at this place. Some of these tasks will be presented in the next chapter 'Exercises – Implementation'. Following, these statements will be explained. All of them are retrieving data from the dataset of Stack-Exchange. The first statement would be executed on a DRDBMS as a full table scan. The second statement could be also executed as a full table scan, if the 'posttypeid' is not indexed. The third and last one would be executed as an indexed query, since a foreign key is being used in the where-clause.

```
1  select Id as [Post Link], Body, Score from Posts
2   where Len(Body) <= 70 && Len(Body) >= 40
3
4  select ParentId as [Post Link], count(id) from posts
5  where posttypeid = 2 and len(body) <= 200 and
6   (body like '%hank%') group by parentid
7   having count(id) > 1 order by count(id) desc;
8
9  DECLARE @UserId int = ##UserId##
10 SELECT Count(*) AS CommentCount, Score FROM Comments
11 WHERE UserId = @UserId GROUP BY Score ORDER BY Score DESC
```

Illustration 18: Three SQL statements from [sedump_query]

Explained as an SQL-statement, an M/R-job is simply a 'select X from A group by Y sort by Y into table B'. It consists of several parts like this SQL-statement: The tables A and B can be considered as the input and output files for an M/R-job. Y is the key, which the mappers are emitting/propagating to the reducer. X can be any kind of data for M/R. This is important to understand, because with M/R the developer has much more freedom to control the data flow than by SQL-queries. In this current situation and task, it will be simplified, that X is just the aggregated result of the mapper functions. The aggregation will be done by the reduce method. This means, that Y is part of X or is even X. M/R will always do a full table scan and would be therefore not always efficient. But DBSs are also weighing up between a full table scan or a key-select, when their queries are using keys. The reason is more from technical nature, that disks are faster in reading sequentially. Therefore, full table scans are very often more efficient than key-selects, when anyway quite the whole data is being needed to serve the query. And because of this, M/R jobs are not always so inefficient as it seems, especially, when they can redistribute their workload dynamically over many nodes as technically possible.

Join with M/R

The last practical task is going about the implementation of an SQL join as one or several M/R jobs. The SQL statement below is to be considered for this task. It must be noted, that besides the join, also a 'group by' is needed to be done on *OwnerUserId*. Generally, Hadoop provides several ways in implementing joins, which are going to be described shortly and which the students should be aware of. First, the M/R developer can use the Job's configuration to transport one of the both lists / tables / files. The data will be sent serialized to all responsible nodes as part of the configuration and will therefore stay in the memory. This is only useful, when one of the lists is very small and will not affect the Hadoop daemons negatively by consuming too much memory. This data can be received by every node as data objects. The second method for performing joins is to use the distributed cache functionality. It does quite the same as the last option, but stores the data on the node's local file system. The source file will be obtained from the HDFS. This ensures better data locality for the worker-nodes and that different jobs can access to the local data as well. The size of the cache (local file system) is set to 10 GB by default. If the client like to join data, which is always going to be modified right before the job starts on the client's / developer's machine (for example for testing purposes), this data can be told via the '-files' argument of the 'hadoop jar [jar-file]' call. This argument uploads every time a new version into HDFS, which can be downloaded by the job-workers shortly after. These local files will be accessible then through *Path*-objects like every normal file.

```
select
    count(a.Id) as [Accepted Answers],
    sum(case when a.Score = 0 then 0 else 1 end)
        as [Scored Answers],
    sum(case when a.Score = 0 then 1 else 0 end)
        as [Unscored Answers],
    sum(CASE WHEN a.Score = 0 then 1 else 0 end)
        *1000 / count(a.Id) / 10.0
        as [Percentage Unscored]
from Posts q inner join Posts a
        on a.Id = q.AcceptedAnswerId
where and q.OwnerUserId !=  a.OwnerUserId and
    a.postTypeId = 2 group by a.OwnerUserId
```

Illustration 19: SQL Join on StackExchange dataset [sedump_query] (modified)

```
reduce ( TextPair key, Value[] value, RResult r)

    ArrayList fromFirstMapper = new ArrayList();
    ArrayList fromSeconMapper = new ArrayList();

    for(int i = 0; i < value.length; i++)
        if(value[i].tag == "0FromFirstMapper")
            fromFirstMapper.add(value[i]);
        if(value[i].tag == "1FromSeconMapper")
            fromFirstMapper.add(value[i]);

    if(fromFirstMapper.isEmpty()) // right outer join
        fromFirstMapper.add(new Value(1,"8FromNowhere...."));
    if(fromSeconMapper.isEmpty()) // left outer join
        fromSeconMapper.add(new Value(1,"9FromNowhere...."));

    for(int i = 0; i < fromFirstMapper.size(); i++)
        int product = 0;
        for(int j = 0; j < fromSeconMapper.size(); j++)
            product = fromFirstMapper.get(i) *
                fromSeconMapper.get(j);
        r.write(key.first, product);
```

Illustration 20: Determination of join-type in reducer with reduce-side join

The next two methods are using M/R to join the data. The first way is called reduce-side join in [HDP_GD]. In this technique, the shuffle phase is doing the merging process. Both files can be read individually by a different *InputFormat* and a different mapper. But both mappers need to emit their key-value pairs of the same type. Beside that, they also need to tag their pairs as well as their keys with their own footprint. The shuffling will sort and merge the data after the key's footprint. The reduce function will get tuples or elements from both datasets of the same join key and can decide, what kind of join the job should perform. The pseudo code above shows, how to perform a full outer join. If an inner join is needed, then only the lines between 11 and 16 would be commented out. It can be seen, that the reducer can distinguish the value's origin by the value's footprint. More details about the join during the shuffle phase will be described in chapter 6.4 'Joining Two XML Files'.

The other join is called map-side join. It is going to be realized using a special *InputFormat*[10], which is reading two files on HDFS at the same time in order to merge their data into one structure together. It is obvious, that in doing this join successfully, it is required, that the data in both files are already sorted after the join-key. The created key-value records will next be fed to the mapper. Implementing different types of joins is similarly like in the reduce-side join: When for a record no other records in the other file can be found during the merge-sort-read, this record would be joined with a default- or null-record.

All ways of joining have been explained in chapter 8 'MapReduce Features' in [HDP_GD] with more details. Even if it seems that the map-side join equals to an SQL-join by using key-selects, it is still not using indexes. It doesn't matter, if the dataset is already sorted after the index-key, Hadoop still need to read the whole file sequentially in order to identify the Records. Same for the reduce-side joins: They do an SQL-like join with a full table scan. At this point it will be clear, that unlike RDBMSs, Hadoop is not creating automatically an index file and uses it to do seek-operation on the dataset in order to increase the performance. However, Hadoop does offer the creation of indexes and it is possible to implement seek-able, block-retrieval functions, if they are not already implemented by someone. The real difference between a DRDBS and Hadoop is therefore the additional development overhead to realize features as selecting data through indexes. But on the other hand, M/R developers have an immense freedom in controlling their data flow. This freedom comes with great responsibility, because mistakes can be very easily made and therefore it is recommended by the author to test the M/R jobs on every little change, when they become more complex. Another method to enhance the performance is to increase the replication factor of one of the joining files in order to reduce network bandwidth and/or excessive disk-access of individual nodes.

Further, students can implement the SQL join above with more than just one of the discussed methods. They can also name the advantages and disadvantages for each of these methods and implement additional sorts or check for more (where-)conditions. Even including and joining the Users.xml file for printing the user name in the task above would be possible. These challenges can be swapped out into additional tasks. In the end, the above described knowledge shall be shared with the students.

10 InputFormatter calculate the splits for each map-task and dividing each split into key-value pairs as Records, which will be fed to corresponding the mapper.

5.4 Group Work

Group work is always welcome along all exercises with not more than three people. With up to five students a more complex task can be worked on. Group work is part of the last exercise, but this task can be picked up much earlier during the semester – maybe three, four weeks before the 12th exercise paper is being handed over to students for finishing it. In those four weeks, the students can work in parallel on the usual exercise papers and on the given project. They shall use concrete distributed software systems like frameworks or DRDBSs to implement some previous agreed use cases. It can be a website, a media warehouse, a computation of complex math functions or just as a simple database. Important is to show features of the systems, which are more or less unique and are standing out compared to other systems, as well as their disadvantages. As a team, the results of the work must be presented verbally and written, for example as handouts.

5.5 Summary

This chapter has introduced the teaching scenario, in which points are included like the choice of the DBSs and the datasets and the description of the learning content from the two exercises with their possible solutions. Also the group work and how it is a part in this concept has been explained. The description of the tasks has been limited on two of the five exercise papers. All five exercises are being available in the appendix.

In the next chapter, some practical tasks of the M/R exercise will be exemplary solved and discussed.

6 Exercises - Implementation

This chapter is about the demonstration of some of the previous described M/R exercises. All realized tasks and exercises can be reviewed in the appendix. Four M/R programs will be discussed, which are the solutions of their corresponding tasks in the M/R exercise. The first one shows an approach how an M/R job can be run on XML data without Apache Avro. The second is an extension of the first program showing, which similarities an M/R job have with SQL-selects. The next one is covering, how to group, aggregate and sort data after a specific criteria. The last one is built on top of the other three and is covering the joining of two XML files.

6.1 Writing M/R Job on XML Data

The M/R-processing exercise has as the 11th task (chapter 4.3.1 on page 25), to write an M/R job, which picks up one attribute from each element of an XML file and sort these elements. The task was explained in sub chapter XML Processing on page 45. The goal here is to let students understand, what is happening before the map phase, as well as strengthen the understanding about the mapper function. The task is formulated as follows:

> "Write a program which lists all display names from coffee/Users.xml. Please research about how to use the XmlInputFormat from Mahout."

The source file of *XmlInputFormat* is either available on the Hadoop cluster over FTP or it can be obtained under the following link[11].

The following excerpt shows the function '*nextKeyValue*' of the split- and record-creator.

```
65        @Override
66        public boolean nextKeyValue()
67                throws IOException, InterruptedException {
68            if (fsin.getPos() < end) {
69                if (readUntilMatch(startTag, false)) {
70                    try {
71                        buffer.write(startTag);
72                        if (readUntilMatch(endTag, true)) {
73                                                                {
74                            value.set(buffer.getData(), 0,
75                                    buffer.getLength());
76                            key.set(fsin.getPos());
77                            return true;
78                        }
79                    } finally {buffer.reset();}}}
80            return false;
81        }
```

Illustration 21: Class XmlInputFormat for creating records feeding the mapper. Created by the Mahout project.

11 https://github.com/apache/mahout/blob/ad84344e4055b1e6adff5779339a33fa29e1265d/examples/src/main/java/org/apache/mahout/classifier/bayes/XmlInputFormat.java

This algorithm shows, how it sets the key and value each time, when this method get called. It simply cuts out a string inside of a split, which is surrounded by the start- and end-tag. The parent-parent class of this formatter is *FileInputFormat*. It contains the method *getSplits*, which computes the splits in bytes by simply calculating how big the file is and how big a split is allowed to be: "Beginning of split: *length-bytesRemaining*; Length of split: *splitSize*"[12]. '*length*' is the size of the file. '*splitSize*' is the size of the current split and '*bytesRemaining*' is the size of the file subtracted with the sizes of all previous splits.

```
33          Configuration conf = new Configuration();
34
35          conf.set("xmlinput.start", "<row");
36          conf.set("xmlinput.end", "\" />");
37          conf.set("io.serializations",
38                  "org.apache.hadoop.io.serializer."
39                  + "JavaSerialization,org.apache."
40                  + "hadoop.io.serializer."
41                  + "WritableSerialization");
42
43          Job job = Job.getInstance(conf, "CoffeeParsing");
44
45          job.setInputFormatClass(XmlInputFormat.class);
46
47          job.setJarByClass(CoffeeXML.class);
48          job.setMapperClass(CoffeeMapper.class);
49  //      job.setNumReduceTasks(0);
50          job.setOutputKeyClass(Text.class);
51          job.setOutputValueClass(NullWritable.class);
```

Illustration 22: Job class using the XmlInputFormat

The solution of this task is inherited from the work of Kiran Bhakre [mr_xmlPrcssng]. He shows, how to use the above mentioned formatter from Mahout in order to get single elements inside of an XML document as values in the mapper. Then he uses in his mapper a SAX-parser to extract his needed information and write them as the values of the map-results. In concern of this exercise task, his job class and the mapper has been modified to handle the file Users.xml correctly and to provide sort functionality. The excerpt above shows the M/R-job.

On line 35 and 36, the start- and end-tag has been set in a configuration. These tags are being used in the formatter class as described before. Next, the job will get initialized with the modified configuration and the InputFormat, Mapper, OutputKey and OutputValue classes will be set. Line 49 shows a function, preventing Hadoop from executing the reduce phase, since the display names of the users in Users.xml don't need to be aggregated. The problem with this is, that the results of the mapper will be written directly into HDFS without executing shuffling over the same results. As described in the basics, this means, the data will not be sorted, because the sort phase is part of shuffling. Therefore the 'setNumReduceTasks(0)' is commented-out, which will make the job shuffling over the data, but since no reducer class has been set, the data will afterwards be written to HDFS without being aggregated (default reducer does nothing).

12 http://www.grepcode.com/file/repository.cloudera.com/content/repositories/releases/com.cloudera.hadoop/hadoop-core/0.20.2-737/org/apache/hadoop/mapreduce/lib/input/FileInputFormat.java#261

The following code snippet shows the mapper. Important to note is here, that after the display name gets extracted from the XML piece, it will get written as the key, since the shuffle phase sorts only after the keys.

```
20    @Override
21    public void map(LongWritable key, Text val, Context context)
22
23    throws IOException, InterruptedException {
24
25        String xmlString = val.toString();
26
27        SAXBuilder builder = new SAXBuilder();
28        Reader in = new StringReader(xmlString);
29        try {
30
31            Document doc = builder.build(in);
32            Element root = doc.getRootElement();
33            String name = root.getAttribute("DisplayName").getValue();
34            context.write(new Text(name),NullWritable.get() );
```

Illustration 23: Mapper using SAXBuilder

However, when analyzing the code of the getSplits function in the footnote 12 at page 51, it will appear that the split's boundaries will highly possible end up randomly in the document and divide therefore single XML elements. This deviding will make these elements corrupt – not parsable by the formatter and in turn later not recognizable by the mapper. These 'destroyed' elements will be found partly at the end of the first split and at the beginning of the second one, which will be called now prefix and suffix. The prefix- and suffix-elements are clearly separated of the rest of the split thanks to the creation of the records in the formatter (Illustration 21). In order to correct this misbehavior, Hadoop must transfer either of them from one mapper node to the other. Then this other node can combine both parts, run on that the nextKeyValue- and map-function in order to let these elements catch up with the other ones. This explanation is only a conclusion or assumption, since Hadoop does create correct results. It is also possible, that this framework covers this problem with another approach.

6.2 M/R Mapper as SQL Selects

The 12th task (chapter 4.3.1 on page 25) is going about writing M/R jobs, which are retrieving the same data as in the task specified by an SQL query. The task has been described in chapter SQL-selects as M/R on page 45. This sub chapter covers the solution of the first query seen in the illustration 18 on page 46.

To implement this query as a job, only the mapper from the last task needs to be modified. There are of course more changes that need to be done, but they are of a technical nature and will be skipped therefore. The source code below (Illustration 24) shows a big part of the map function. First, it gets the information needed to check the condition from the where-clause. The minimal and maximal length of the body is going to be determined through a configuration setting. If the condition is positive, a customized *writable*-class will be initiated and filled with the data of the XML element and then get written as a value result.

```
42              String body = root.getAttribute("Body").getValue();
43              int min = context.getConfiguration().
44                      getInt("minLen",0);
45              int max = context.getConfiguration().
46                      getInt("maxLen",Integer.MAX_VALUE);
47
48              if (body.length() <= max && body.length() >= min) {
49
50                  PostWritable post = new PostWritable();
51                  post.id = new IntWritable((Integer.parseInt(root.
52                          getAttribute("Id").getValue())));
53                  post.score = new IntWritable(Integer.parseInt(root
54                          .getAttribute("Score").getValue()));
55                  post.body = new Text(body);
56
57                  context.write(NullWritable.get(), post);
58              }
```

Illustration 24: SQL query realization by mapper only

This job will not start a reduce or shuffle phase over the data. The output format is set as *TextOutputFormat* and therefore the results will be written into the HDFS file using the *toString()* function. This process will be detailed explained in [HDP_GD] (chapter 4 Implementing a Custom Writable).

```
38      @Override
39      public String toString(){
40          return id + ", " + score + ", " +
41                  body + "\n\n ------------ \n\n";
42      }
```

Illustration 25: Printing binary data in a human readable form

6.3 Group By and Order By with Shuffle

Also part of the 12th task is the implementation of a customized sort and reducer. The corresponding SQL query can be seen in the illustration 18 on page 46 on line 10. The driver class[13] is very simple: It sets the own created *Mapper*, *Reducer* and additionally a *SortComparator* with the method `setSortComparatorClass.` The mapper extracts the score from each comment and use it as the key. The value is always the number '1'.

```
13      @Override
14      public void map(LongWritable key, Text val, Context context)
15              throws IOException, InterruptedException {
16
17          CommentWritable comment = new CommentWritable(val.toString());
18          context.write(new IntWritable(comment.score.get()),
19                  new IntWritable(1));
20      }
```

Illustration 26: Mapper preparing data to be sorted and grouped

13 Java program with Hadoop, which intializes the Job object among others with the mapper and reducer classes

After the mapping, the shuffling phase starts. The framework will use the modified comparator to sort the integer keys. As seen below, it gets the keys as a byte stream and rebuild it back as integer objects in order to call its '*compareTo*'-method. The result will be negated to enforce a descending order of the keys. Further, Hadoop will group the values together after the key, which happens automatically.

```
13      public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2,
14              int l2) {
15          Integer v1 = ByteBuffer.wrap(b1, s1, l1).getInt();
16          Integer v2 = ByteBuffer.wrap(b2, s2, l2).getInt();
17
18          return (-1) * v1.compareTo(v2);
19      }
```

Illustration 27: Overwriting compare method to sort in descending order [sortComparator]

The only thing, the reducer needs to do is to sum up the values with each other and to forward the sum and the key from the mapper as seen below.

```
13      protected void reduce(IntWritable key,
14              Iterable<IntWritable> values, Context context)
15              throws IOException, InterruptedException {
16
17          Iterator<IntWritable> i = values.iterator();
18          int count = 0;
19          while (i.hasNext()) {
20              count += i.next().get();
21          }
22          context.write(key, new IntWritable(count));
23
24      }
```

Illustration 28: Reducer calculating the sum of the '1'-numbers

It should be noted, that count don't get incremented but summed up. The reason is, that every *Reducer* can be used also as a *Combiner*, if the job is configured respectively. As described on page 42 (under the illustration), combiners can 'steal' the map-results and calculate the sums of the values made on each map-node alone. The reducer can get an already partially calculated and reduced list with values higher than '1'. Therefore, the method should use the values directly without to foreshadow specific values inside. Of course, when no combiner is explicitly set, the input-values of the reducer are always '1', but it is not recommended implementing that way due to the development of a bad habit.

6.4 Joining Two XML Files

The last practical task (13th objective in chapter 4.3.1 on page 25) with Hadoop is about to merge two XML files with M/R in order to join the data of these files. This solution uses the reduce-side join. The corresponding SQL query can be looked up on page 47 (Illustration 19) in chapter Join with M/R (page 46).

Implementing a join isn't easy with Hadoop. Much knowledge in using this framework is required in order to develop efficiently and successfully, since the data structures and functions need to be treated carefully. It is recommended to the reader to make always a deep-copy of the data got by the framework. In the end, Hadoop is a distributed cluster as well. When using additional software modules like JDOM for parsing XML, the chance of mistakes will increase, especially, when the programs are not tested on the local, single-node cluster.

The job is initialized as shown in the code excerpt below. The XML files are being declared through the '*MultipleInputs*'. When a self join is being made (Posts |><| Posts), a copy must still be used, else one mapper will not be able to read from it for some reasons. The *addInputPath* demands beside the path name, an *InputFormat* for splitting files, generating *Records* from the file and a mapper class. The mapper from both input files shall output their key-value pairs of the same type. After that, the pairs from both mappers will be grouped after a customized comparator (*GroupComparator*) and sorted by another one (*SortComparator*). The map-key is from type *TextPair*, which contains just two *Text* attributes. The first comparator compares the pairs with each other only by the first element, while the sort-comparator uses both elements of the key.

This ensures that pairs, which contain an equal *Text*-element in the first position, but have different string sequences on the second element, will still be grouped together. This means, keys, which are only partially equal, will be available in one reduce-call including their values. To distinguish the values from which file or mapper they came from, in order to complete the join in the reduce call, the second comparator helps to sort the values by the second key-element (*SortComparator*). This second element will be used for tagging the keys with the information of origin. The mapper also needs to tag their emitted values. The key-tags are for the *SortComparator* during the shuffle phase, while the value-tags are for the reducer.

```
128         MultipleInputs.addInputPath(job, acceptFile,
129                 XmlInputFormat.class, JoinAccepterMapper.class);
130         MultipleInputs.addInputPath(job, postFile,
131                 XmlInputFormat.class, JoinPostMapper.class);
132         FileOutputFormat.setOutputPath(job, outPath);
133
134         job.setPartitionerClass(KeyPartitioner.class);
135         job.setGroupingComparatorClass(TextPair.GroupComparator.class);
136         job.setSortComparatorClass(TextPair.GroupComparator.class);
137
138         job.setMapOutputKeyClass(TextPair.class);
139         job.setMapOutputValueClass(PostWritable.class);
144             job.setReducerClass(JoinReducer.class);
145             job.setOutputKeyClass(TextPair.class);
146             job.setOutputValueClass(PostWritable.class);
```

Illustration 29: Job initialization for a reduce-side join

To see the data flow, excerpts from both mappers will be given (Illustration 30 and 31). Both mappers generate a *Post* object from the XML string 'val' through the SAX parser in the constructor. These post objects are holding all relevant data from the XML element. Later, also both mappers are tagging the post objects with their own tags as well as the *TextPair*-keys in order to distinguish them later in the reduce phase. The *JoinPostMapper* filters all posts, which are not from type '2' - as the SQL query states. Another difference lies in the assignment of the first key-element 'pair.first'. The *JoinAccepterMapper* uses the '*AcceptedAnswerId*' (post.accept) as the first key-piece. As described above, when the map phase finishes and shuffling starts, the posts of both mappers will be grouped with the help of just the first key-piece through the *GroupComparator* and will be sorted by the *SortComparator* in that way, that the pairs from the *JoinAccepterMapper* will come first.

```
20          PostWritable post = new
21                  PostWritable(val.toString());
22
23          if (post.isOk && post.typeId.
24                  toString().equals("2")) {
25              post.tag = new Text(tag);
26
27              TextPair pair = new TextPair();
28              pair.first = post.id;
29              pair.second = new Text(tag);
30
31              context.write(pair, post);
32          }
```

Illustration 30: JoinPostMapper, which forwards the data

```
post.tag = new Text(tag);

TextPair pair = new TextPair();
pair.first = post.accept;
pair.second = new Text(tag);

context.write(pair, post);
```

Illustration 31: JoinAccepterMapper, which uses the AcceptedAnswerId as key instead

This combination of such comparators is called secondary sort and is explained well in chapter 8 'Secondary Sort' in [HDP_GD]. A reduce-side join is nothing else, then using the join keys, like in this task the *Post.Id* and *Post.AnswerAcceptedId* in combination with the secondary sort.

Next, the reduce function (seen below) retrieves the values and groups them with the help of the tags seen on line 20. No null-objects are being used to realize an outer join. On line 33, it will be checked, if there is only one or more questions and one accepted answer for these questions available, because the join-condition *Posts.Id==Posts.AcceptedAnswerId* is a unique relation (1-N). Questions, which don't have answers, are not being solved. Answers without questions have not been accepted as an answer (which is the case for the most answers). The case with one question and several accepted answers is not possible, since the attribute *AcceptedAnswerId* lies in the question-post element and it can only contain one key (1 answer → N questions). The other way around is through the data model possible, but shouldn't appear in the use cases of StackExchange, since this would assume cross-site posts and -answers are eligible: Every question should be a thread-starter (*Posts.PostTypeId*="1"), where the answers can only appear in one thread (*Posts.PostTypeId*="2"). Anyway, this case is being included in this job. Next, on line 37 it will be checked, if the answer and the questions are not from the same author. Later, the accepted answer will be written as a value, while his owner will be written as key - together with an unique counter.

```
17          ArrayList<PostWritable> questions = new ArrayList<PostWritable>();
18          ArrayList<PostWritable> answers = new ArrayList<PostWritable>();
19          Iterator<PostWritable> i = values.iterator();
20          while (i.hasNext()) {
21              PostWritable post = new PostWritable(i.next());
22              if (post.tag.toString().equals(JoinPostMapper.ACCE))
23                  questions.add(post);
24              if (post.tag.toString().equals(JoinPostMapper.POST))
25                  answers.add(post);
26          }
33          if (questions.size() >= 1 && answers.size() == 1) {
34
35              for (int j = 0; j < questions.size(); j++) {
36
37                  if (!answers.get(0).ownerId.toString().equals(
38                          questions.get(j).ownerId.toString())) {
39
40                      TextPair pair = new TextPair();
41                      pair.first = new Text(answers.get(0).ownerId);
42                      pair.second = new Text(String.valueOf(j));
43                      context.write(pair, answers.get(0));
```

Illustration 32: JoinReducer for realization the inner join

The reducer's output results into a list of posts as answers. In order to group them by their owner and to create the sums, a second M/R job will be used. The job takes the result from the last job and summarizes the posts after the owner. His reducer does the calculation of the scores as seen below. The job driver (footnote on page 53) of the 2nd job and the mapper are very simple and are just for moving the data – preparing it for the shuffle phase. The complete source code can be reviewed in the appendix after the M/R exercise.

Summarizing, two shuffle phases have been needed to solve this task. The first one was for joining both datasets and provide them to the first reducer, while the second one did the aggregation (group by) process. The 2nd reducer were just needed to iterate over the gotten values and calculate the sums.

```
17          Iterator<PostWritable> i = values.iterator();
18
19          int accepted  = 0;
20          int scored = 0;
21          int unscored = 0;
22          while(i.hasNext()){
23              accepted++;
24              PostWritable val = i.next();
25              if(val.score.get() == 0) unscored ++;
26              else scored++;
27          }
28          ScoreWritable score = new ScoreWritable();
29
30          int percent = scored * 100 / accepted;
31
32          score.accepted = new IntWritable(accepted);
33          score.scored = new IntWritable(scored);
34          score.unscored = new IntWritable(unscored);
35          score.percentUnscored = new IntWritable(percent);
36
37          context.write(new Text(key),score);
```

Illustration 33: SumReducer for aggegating the scores

Lastly, the other methods in how to implement the SQL self-join will be described. First, self-joins can be very easily realized without any join technique of Hadoop, by simply using just one mapper, which is doing the task of both described mappers to emit the same values. This has not been done, because the reason for solving this task was to give the reader an understanding of the reduce-side join. It would also be possible, to use the '*OwnerId*' as map-output-key and to group the tuples, but that can cause high memory usage on the reducer side, since it will get all posts of one user at a time and needs to join the data in memory. Next is to use the map-side join as described shortly on page 47. It should also be noted that this variant would need two M/R jobs, since one shuffle process is needed before the join execution to perform the sort of one of the files after the *AcceptedAnswerId*. The last alternative would be using the configuration parameter or the distributed cache on one of these files. If the chosen file does not need to be processed with M/R, it can be read directly, which results in the creation of just one job / M/R-phase creation instead.

6.5 Summary

This chapter showed with the help of four implementations the functionality of MapReduce and what students should learn from their corresponding tasks. Hadoop is a clean and understandable open-source framework and Java implementation of M/R.

Also the author has experiences of Hadoop add-ons like Mahout, Giraph and Flink, which he has used, tested and developed on them. These add-ons are greatly improving Hadoop's capabilities and tasks could have been developed for the M/R exercises 'Limitations', 'Alternatives' and 'DataMining' (Table 10 on page 29). Unfortunately, none of these programs have become part of this thesis, because they were more or less just 'Hello-World' programs and / or didn't fill in one exercise in order to discuss and teach concepts and knowledge, which are relevant for the lecture. Information on these topics can be found mostly only in books like 'Apache Mahout Cookbook' [mahoutB]. If the reader like to start reading about the add-ons, it is strongly recommended to understand first Hadoop itself and quite everything, what is shared in the 'Hadoop The Definitive Guide' [HDP_GD]. The author developed during his thesis-work with MySQL NDB (C++) and MySQL Fabric (Python). It is also worth taking a look into these libraries as well. The students might develop in that case with other languages, but on the SQL layer, they always can test their programs with the familiar MySQL-server and -client platform.

This was the last chapter explaining the work of this thesis. With the next and final chapter, the thesis will be rounded up and finished.

7 Final Remark

This final chapter will be going about, what have been achieved and not achieved in this thesis as a summary (actual/target status). It will be mentioned, which learning aims and other requirements have been accomplished in the concept and which have been implemented as tasks in the exercises. Also goals from the concept will be checked, if they got implemented.

Next comes the author's reflection of his work and progress and lastly will be described, what could have been done better or in case others will continue this work or work in this area, some suggestions will be made for them.

7.1 Summary

This thesis has described a design of a concept for creating exercises about a specific topic. The topic was Distributed Information Systems, which is being provided as module to master students. Since this topic is covering a huge amount of knowledge, only a few points have been selected for the exercises. The relation between the lecture and this thesis has been made through learning aims in chapter 1.3 Aims of the Lecture. In consideration of these aims, the task in this thesis was to implement programs, creating exercises and designing a concept for this specific task as described in chapter 1.4 Responsibilities and Tasks in this Thesis. In order to achieve these aims, research was needed to be done about this topic and the content of the same aims. Some of the resulting knowledge was presented in chapter 2 Basics.

This thesis covered mainly the conception and started with it in three different parts. The first part in chapter 3 Didactic Design - Analysis Part I, kick started with its basic conditions and moved over to different levels of learning objectives, including the ones from chapter 1.3. Then some words have been left about the learning methods for the exercises and after that, an exercise design pattern was introduced to describe a structure and the focus for exercises with the help of Bloom's taxonomy. Lastly, it was shortly described, how the learning success of the students can be monitored or tested and how some feedback about the concept itself or its exercises can be given / generated.

The second part in chapter 4 Exercise Learning Objectives - Analysis Part II was about the description of the relation between the learning aims and the exercises by the - in this thesis so called - learning objectives. The objectives are grouped after the learning aims and expressing more or less what the tasks of each exercise should be about. Finally, the objectives will be put into correlation with the orientation aims from chapter 3.2.3 to consider the standard and difficulty for master students. This part was not completed. The table 13 shows, how far the definition of the learning objectives has been progressed. Also, none of the very first type of objectives, a motivating kick start for students, which was explained in chapter 3.4, has not been embedded in the learning objectives and the exercises.

No	Minor group of learning objectives	Progress	No	Minor group of learning objectives	Progress
1	Partitioning - DRDBS	100%	7	Alternatives - M/R	5%
2	Query - DRDBS	70%	8	DataMining - M/R	0%
3	Replication - DRDBS	100%	9	Data distribution - Comparison	0%
4	Transaction - DRDBS	70%	10	Query versus M/R - Comparison	0%
5	Processing - M/R	100%	11	Data manipulation - Comparison	0%
6	Limitations - M/R	5%	12	Cloud design pattern	100%
-	Average				~46%

Table 13: Progress of developing the learning objectives

The last part of the concept in chapter 5 Teaching Scenario – Planning were about making decisions, which platform is being used for the exercises, like the choice of database systems and the datasets. Besides, two of the five exercises have been formulated to provide the reader a better understanding.

With chapter 6, this thesis left the conception and presented excerpts of the solutions of four M/R tasks. At this point, also several programs with the NDB-, Fabric-API or some queries on distributed queries could have been presented. Presenting and discussing this knowledge, too, would go beyond the scope of a bachelor thesis.

Following, a reflection and statements about the future work will be given. After that, the appendix will be provided containing background information like the implemented exercises and presented programs. Also two short notebooks can be found there about the installation and configuration of the MySQL Cluster and Hadoop cluster, which were used in this thesis. The system documentation of the Hadoop cluster has been made outside the context of this thesis. Only the exercises will be printed with this thesis. The rest can be found in the CD enclosed to this document.

7.2 Reflection

Obviously, the original work and aims from chapter 1 are not covered or implemented completely by this thesis. This is unfortunate, but on the other side big parts of the conceptional work has been accomplished. Two clusters have been setup up on five virtual machines. Both served as a platform to be able to create practical tasks. A few of these tasks has been implemented. The most theoretical tasks are based on the knowledge from DDBS-books like [DDBS] or [HDP_GD]. Based on the coverage of the lecture, an exercise concept has been described in this thesis, which shows with learning objectives and pedagogical elements, how these exercises have been created. In case the reader like to implement new exercises about this topic, he or she can use this concept as well to increase the quality of his / her work. After all, relying on several books was a good decision in order to put the work in a more stable fundament.

Further, it should be noted, that this thesis used many links and websites as references, where the original author is unknown / anonym / has a nickname or the content is not trustful (like Wikipedia). Behind each of these references, many other websites have been visited and examined in order to earn enough knowledge and to create an opinion. The referenced websites are just equal to the author's opinion of this thesis and it has been decided to credit these sources for their help and their providing of information. After all, these references has not been used to build on profoundly facts on which this thesis lies. This is also one of many reasons, why the comparison of the DBS types is not easy to make and have become a side issue in this work. On the other hand, the reader can see with the help of the reference [mr_xmlPrcssng] of Kiran Bhakre, that not trusted sources are very well helpful to save time. Additionally, when there was a way to validate the sources of their correctness, like in this example by simply implementing and testing the idea, then this possibility was taken. Mr. Bhakre could have been anonym as like as most software developers, helping around the globe with their advises.

The final status of the thesis is, that the usage and embedding of the datasets in the exercises and their tasks, has not been started and done yet. In fact, some programs are working with the datasets of StackExchange like the ones presented in chapter Implementation, but that does not mean they have been integrated together in one scenario. These programs were for testing the suitability of the dataset and have been later further developed. The integration process was planned to be started, when all exercises and definitions of the tasks has been implemented as well as the dummy / hello-world / prototype programs would have been realized.

7.3 Future Works

The last words are spent for giving some suggestions and tips in how further to enhance the exercises, implement programing tasks or develop ideas to teach students helpful knowledge by theoretical tasks. The very first point is to listen to the students, their demands and their suggestions. This does not mean to follow their saying, but to note their critics in order to be able to counter and accompany their interests. Besides, it will calm students by providing them a way expressing their displeasure – doesn't matter if real displeasure does exist.

It is important to further develop the exercises by integrating them with each other. The idea from this thesis is to use the same dataset across the most of the tasks in all exercises. This enables, as mentioned several times, to show students in an easy way the comparison and correlation of concepts and software systems like database system types. But this cannot be done only through a uniformed dataset. A requirement for achieving a development of a scenario, is to get detailed know-how about both information systems – DRDBS and M/R – practically as well as theoretically.

In this thesis and in the exercises, many parallels have been made between DRDBS and M/R. No literature could be found to confirm the thoughts or conclusions, which is why there was no reason to continue making new comparisons, since they would be directly or indirectly used in the exercises for master students. The creation of exercises for providing / teaching students none proofed knowledge, was too much responsibility being acceptable for the author due of his lack of knowledge and in context of a bachelor thesis. Much time has been invested in reading literatures about DRDBS and M/R alone. Developing exercises means, to know so much about a field, that this person is able to explain complex problems and solutions regarding that field in an easier way to people, that they can learn about this topic faster by him/her than of the literature used by that person. At this point, one book should be mentioned, on which a look can be taken into, since it covers MS SQL and Hadoop together. This literature should be helpful in creating more and faster tasks about comparing the DBS types: Microsoft SQL Server 2012 with Hadoop [MS_SQL_HADOOP].

When there is space left in which direction the exercises can be developed, the students' feedback can be considered. For example, they could decide, which dataset, parts of it or DBSs are best to use for specific use cases and for what reasons. By the way this would increase the students' self-dependency. Next, since these exercises can be part of this lecture and therefore in turn can be part of the master consecutive degree program, some content from the bachelor study's lectures like 'Databases', 'Information systems' and 'Parallel programming' can mentioned and used to support some of the teaching materials. One good example is to compare M/R with MPI taught in 'Parallel programming'. Linking some content from the master's lectures as well, would be very beneficial. The author himself got positive experiences in his whole bachelor study, that different lectures had good connections and links with/to each other.

Some programs and queries for the DRDBS exercises shall be developed as well. The MySQL Cluster for example, offers an API for connecting and 'speaking' directly with the data nodes. Unfortunately, only the NDB-API for C/C++ provides the functionalities like full table scans or scheme information lookups [mysql_ndbFunc]. Since all other practical exercises are based on the programing language Java, introducing another language would make the solving of all exercises more difficult for students and disturb the concentration on the actual topic. Also with MS SQL Server more practical ways are existing to show students distribution. These might not be the development of programs, but also the steps of configuration can be seen as procedures. The author does not exclude, that this database can provide very helpful APIs to inspect and track the internals and processes of data- and transactional-replication. It is highly possible, that such APIs also are implemented in C / .NET.

Another aspect is the observation in the development of current DDBS, because they are relatively new – even if the idea is quite old. Many (open-source)-projects and research groups are existing, which are developing and patching the DDBSs with more and more features. The area of cloud and distributed information systems is growing. For example the toolkit Fabric of MySQL is currently very restrictive, but 'soon' it will release a C#-connector [fab_cShrp]. A weakness of this thesis is, that it leans or built on just around three books. To generate an own knowledge and an opinion, more literature is needed or more time must be invested in experimenting and developing programs for such software systems. With this knowledge, new exercises and challenges can be developed to help students in a pedagogical and constructive way.

8 List of Literature

[ad_org] Das Tübinger Portalteam. Advance Organizer. https://www.e-teaching.org/materialien/glossar/advance_organizer, 10/25/2015.

[concept_checklist] Tobina Brinker. Checkliste fuer die Konzeption und Strukturierung von Lehrveranstaltungen. https://ilias-hdw.fh-bielefeld.de/goto.php?target=file_2185_download&client_id=IHDW, 10/25/2015.

[concept_struc] Tobina Brinker & Christian Willems. Gestalten Sie Ihr Lern-ZIMMER. https://ilias-hdw.fh-bielefeld.de/goto.php?target=file_2179_download&client_id=IHDW, 10/25/2015.

[couch_api] IBM Cloudant. BigCouch API. http://bigcouch.cloudant.com/api, 10/25/2015.

[DDBS] M. Tamer Oezsu and Patrick Valduriez. Principles of Distributed Database Systems, 3rd Edition. Springer, Luxenburg, 2011

[drq_eqr] Bundesministerium für Bildung und Forschung. DQR und EQR. http://www.dqr.de/content/2323.php, 10/25/2015.

[eHealth] Grid Talk. Grid computing in five minutes. http://www.gridtalk.org/Documents/ehealth.pdf, 10/25/2015.

[eTeach_dida] Das Tübinger Portalteam. Didaktisches Design. https://www.e-teaching.org/didaktik, 10/25/2015.

[eTeach_exer] Das Tübinger Portalteam. Übung / Tutorium. https://www.e-teaching.org/lehrszenarien/tutorium, 10/25/2015.

[eTeach_scen] Das Tübinger Portalteam. Lehrszenarien. https://www.e-teaching.org/lehrszenarien, 10/25/2015.

[EU_rec] H.-G. Poettering, J. Lanarcic. Recommendations European Parliament Council. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32008H0506%2801%29&from=EN, 10/25/2015.

[fab_cShrp] Roberto Ezequiel Garcia Ballesteros. MySql Fabric for C#. http://forums.mysql.com/read.php?144,618413,618835#msg-618835, 10/25/2015.

[HDP_GD] Tom White. Hadoop: The Definitive Guide, 3rd Edition. O'Reilly Media, Sebastopol, 2012

[hdp_lock] Apache Foundation. NameNode.java. http://grepcode.com/file/repo1.maven.org/maven2/org.apache.hadoop/hadoop-hdfs/2.6.0/org/apache/hadoop/hdfs/server/namenode/NameNode.java#1702, 10/25/2015.

[intro_hadoop] Alexander Sirotin. Einfuehrung Hadoop-System. System Documentation, University of Applied Sciences and Arts Hannover, 2015.

[lernObj_wiki] Wikipedia. Klassifizierung_von_Bildungszielen. https://de.wikipedia.org/wiki/Lernziel#Klassifizierung_von_Bildungszielen, 10/25/2015.

[mahoutB] Piero Giacomelli. Apache Mahout Cookbook. Packt Publishing, Birmingham, 2013

[MR_WHITE] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Whitepaper, Google, Inc., 2004.

[MR_WIKI] Wikipedia Community. MapReduce. https://en.wikipedia.org/wiki/MapReduce#Logical_view, 10/25/2015.

[mr_xmlPrcssng] Kiran Bhakre. XML Processing in Map Reduce . http://handsonhadoop.blogspot.de/2014/04/xml-processing-in-map-reduce.html, 10/25/2015.

[ms_acid] Microsoft. Replication Types. https://technet.microsoft.com/en-us/library/aa198189%28v=sql.80%29.aspx, 10/25/2015.

[MS_SQL_HADOOP] Debarchan Sarkar. Micosoft SQL Server 2012 with Hadoop. Packt Publishing, Birmingham, 2013

[MSSQL] Dušan Petković. Microsoft SQL Server 2012 A Beginner Guide, 5th Edition. McGraw-Hill, New York, 2012

[mysql_ndbFunc] MySQL Community. Core NDB API Classes. https://dev.mysql.com/doc/ndbapi/en/overview-ndb-classes.html, 10/25/2015.

[mysql_ref] MySQL / Oracle. MySQL 5.6 Reference Manual 5.6-en. http://dev.mysql.com/doc/refman/5.6/en/, 10/25/2015.

[oXA] X/Open Company Limited. The XA Specification. http://pubs.opengroup.org/onlinepubs/009680699/toc.pdf, 10/25/2015.

[pche_hdfs] Apache Software Foundation. HDFS Design. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html, 10/25/2015.

[pche_yarn] Apache Software Foundation. Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, 10/25/2015.

[sedump_data] StackExchange. StackExchange sites. http://data.stackexchange.com/, 10/25/2015.

[sedump_query] StackExchange. StackExchange Data Explorer. http://data.stackexchange.com/stackoverflow/queries, 10/25/2015.

[sedump_struct] StackExchange community wiki. Database schema documentation. http://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede, 10/25/2015.

[sortComparator] Nickname Tudor. Sorting by value in Hadoop from a file. http://stackoverflow.com/questions/8289508/sorting-by-value-in-hadoop-from-a-file, 10/25/2015.

[std_guide] Manfred Krause and Volker Ahlers. Study guide - department information science. http://f4.hs-hannover.de/fileadmin/media/doc/f4/Studium/Bachelor_Studiengaenge/BIN/15-05-04-Studienhandbuch-Abteilung-Informatik-F4-Internet-2015.pdf, 10/25/2015.

[taxonomy] Mark Fleshman. Bloom's Taxonomy.

http://www.learninghouse.com/blog/consulting/bloom%E2%80%99s-taxonomy, 10/25/2015.

[test_learnObj] Wikipedia. Lernzielkontrolle. https://de.wikipedia.org/wiki/Lernziel#Lernzielkontrolle, 10/25/2015.

[vertPar] Navathe, Ceri, Wiederhold and Dou. Vertical Partitioning Algorithms for Database Design. Whitepaper, Stanford University, 1984.

[weatherForecast] Davor Davidovic. Feature - Forecasting weather on the grid. http://www.isgtw.org/feature/feature-forecasting-weather-grid, 10/25/2015.

[wiki_DB] Wikipedia Community. Database. https://en.wikipedia.org/wiki/Database#Performance.2C_security.2C_and_availability, 10/25/2015.

[wiki_deFrame] Wikipedia Community. Deutscher Qualifikationsrahmen. https://de.wikipedia.org/wiki/Deutscher_Qualifikationsrahmen#Einordnung, 10/25/2015.

[wiki_euFrame] Wikipedia Community. Europäischer Qualifikationsrahmen. https://de.wikipedia.org/wiki/Europ%C3%A4ischer_Qualifikationsrahmen#Vergleichbarkeit, 10/25/2015.

[wiki_mysql] Wikipedia Community. MySQL Cluster. https://en.wikipedia.org/wiki/MySQL_Cluster#Replication, 10/25/2015.

Distributed Information Systems: MIN-VIS

Exercise 1: Partitioning – DRDBS

Theory questions

1. Speaking in context of DRDBS: What is a partition and what is a fragment? Describe the relation between these two terms with the help of a draft.

Answer:
A partition is a place-holder for fragments of data. A fragment is a part of that data. All fragments in sum are together the data. A duplicated or copied fragment is called replicate. Partitioning means to split up a place-holder. Same as for partitioning, fragmentation means, splitting up the data.

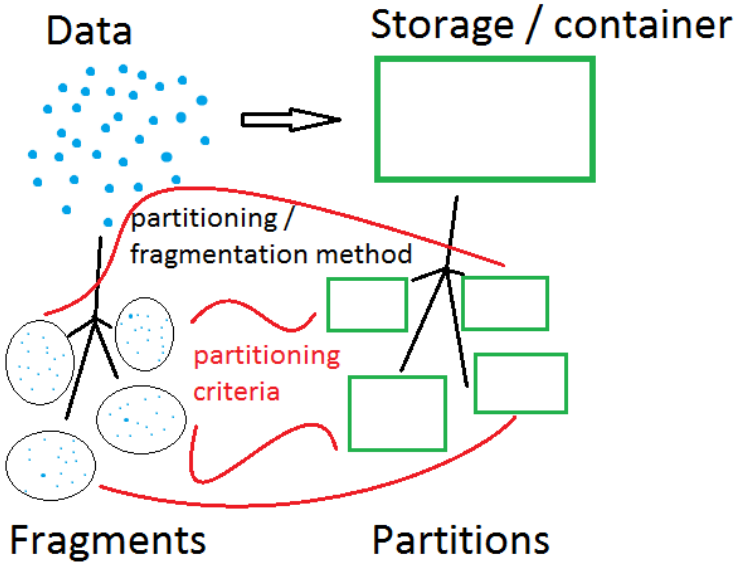


Illustration 1: Relation between fragments and partitions

2. What is a shard and what does sharding mean?

Answer:
"I take sharding to mean the partitioning of a table **over multiple machines** (over multiple database instances in a distributed database system), whereas partitioning may just refer to the splitting up of a table **on the same machine**. So a table that is sharded has been partitioned, but a table that has been partitioned has not necessarily been sharded."[shardDef]

Sometimes sharding is also associated with rebalancing automatically the data afterwards it had been inserted by the DBS on regard of hardware resources like performance and disk-space. Therefore, some frameworks, which just are intercepting SQL-queries and redirecting them to the appropriate SQL-node, do not do sharding relatively to this definition. This behavior can also be labeled as auto-sharding.

3. What are ‘partitioning/fragmentation methods’ and what are ‘partitioning criteria’?

Answer:
Partitioning methods says, how the DRDBS splits its database(-tables) or dataset in order to be able to distribute the data.

The partitioning criteria is a function, which takes an ID of part of data as input and gives the partition ID. With this criteria the DBS can decide, where incoming data should be saved or in which partition the data, which is requested, is residing.

4. Research a few typical ‘partitioning methods’ and ‘partitioning criteria’ and then list them. Discuss the advantage(s) of them with your neighbor.

Answer:

Partitioning methods:
Horizontal: splitting the table by the tuples, rows

Vertical: splitting the table by the attributes, columns

Hybrid: consecutively splitting the table/data horizontally and vertically

Another definition:
“Vertical partitioning subdivides attributes into groups and assigns each group to a physical object. Horizontal partitioning subdivides object instances (tuples) into groups, all having the same attributes of the original object.”[vertPar] (Chapter Introduction)

Assume/definition:
n is the id of a partition starting with 0
N is the total amount of partitions
for horizontal method: i is the id of a tuple starting with 0
for vertical method: i is the id of an attribute starting with 0

Partitioning criteria n=f(i):

1. Round-robin: f(i)=i mod N
+ sequential access enforce full parallel read and write

2. Hash: f(i | i-th tuple)= (MD5(i) | SHA(i) |
more2kSalary(i)|...) mod N
The DBS can either use a technically hash functions like MD5 on the id / primary key, which will not be more different then the Round-robin-method, or a business logic function to check for a specific character on tuples (called predicate in [DDBS] in chapter 3.3.1). This makes each partition a special place holder for data. You can imagine the hash-functions as like as firewall rules:

1. Every employee, who has < 2k salary, goes to partition 1
2. Every employee, who has => 2k salary and is not a boss, goes to partition 2
3. Every employee .....

+ (data-model-) developer can decide, which kind of queries should be run on a single node and which all other queries should run parallel

3. Range: f(i | i-th tuple)
+ If physical storage limitations are critical, the size of an partition can be fixed using the range definitions on the primary key
+ If tuple-attributes are used for ranges (not primary key, but for example salary), range-based queries can be efficient executed on single nodes. These queries would be like full table scans, but on the 'right' nodes.

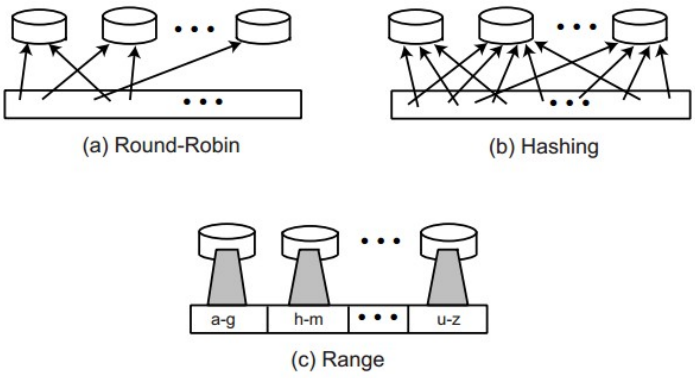


Illustration 2: Partitioning types/criteria

Source: [DDBS] (Chapter 14.2 Parallel Data Placement)

5. What is a primary- and derived-horizontal fragmentation?

Primary horizontal fragmentation is splitting the data directly after the defined (fragmentation) predicates.

Derived fragmentation is fragmenting data based on, where its connection of already fragmented data. Example: The relations user and post have a 1-n-relationship with the user-id and the post-owner. The relation user will be fragmented after the predicates user1='creationDate < 01/01/2010' and user2='creationDate => 01/01/2010'. The user1-data/-tuples will go to partition 1, the user2-tuples will got to partition 2. All posts, who have a owner, which resides in partition 1, will also be saves in partition 1. The same logic applied to all other posts. The user-relation is primary fragmented, the post-relation is derived fragmented.

More information in [DDBS] chapter '3.3.1.2 Primary Horizontal Fragmentation' and chapter '3.3.1.3 Derived Horizontal Fragmentation'.

6. Explain, when which partitioning method has the best or better usage. You can take the Stackoverflow dataset and show how and why would you split the data.

7. Formally define the three correctness rules of fragmentation.

Completeness: Any data, that can be found in R, can also be found in one or more fragments of R {R1,R2,...,Rn).

Reconstruction: A operator F(Ri,Rj) must exist, that should be able to merge all fragments into the original relation R. This enables to respect constraints of relations.

Disjointness: If a data item di is in Rj, it cannot be found in all other fragments Rx(x<>j).

These three criteria are necessary for ensuring, that an authorized user have a logical view on the **relational** data and perform correct operation on them.

More information can be found at [DDBS] chapter 3.2.4 Correctness Rules of Fragmentation and in chapter 5 Data and Access Control.

8. Explain the following draft. When do you use this approch creating a global database. What is a global conceptual schema (GCS)?

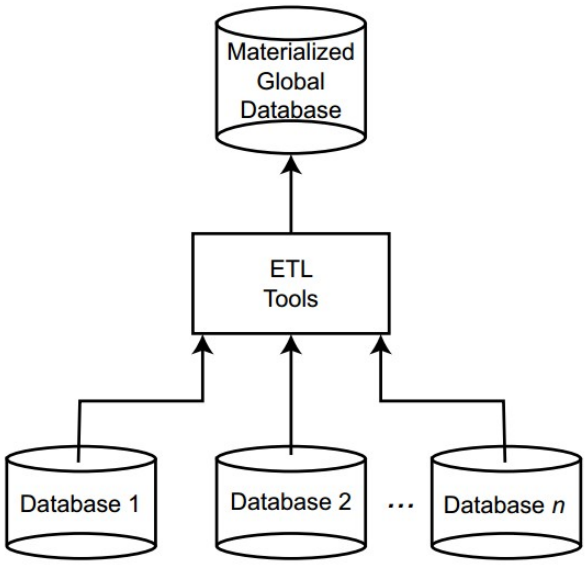


Illustration 3: Data Warehouse Approch. Source: [DDBS] Chapter 4 Database Integration

"Data warehousing [Inmon, 1992; Jarke et al., 2003] supports decision support applications, which are commonly termed On-line Analytical Processing (OLAP) (…) In contrast, OLAP applications, such as trend analysis or forecasting, need to analyze historical, summarized data coming from a number of operational databases. They use complex queries over potentially very large tables. Because of their strategic nature, response time is important. The users are managers or analysts."

Source: [DDBS] Chapter 4 Database Integration

"We first note that the physical data organization on each machine may be, and probably is, different. This means that there needs to be an individual internal schema definition at each site, which we call the local internal schema (LIS). The enterprise view of the data is described by the global conceptual schema (GCS), which is global because it describes the logical structure of the data at all the sites. To handle data fragmentation and replication, the logical organization of data at each site needs to be described. Therefore, there needs to be a third layer in the architecture, the local conceptual schema (LCS). In the architectural model we have chosen, then, the global conceptual schema is the union of the local conceptual schemas. Finally, user applications and user access to the database is supported by external schemas (ESs), defined as being above the global conceptual schema."

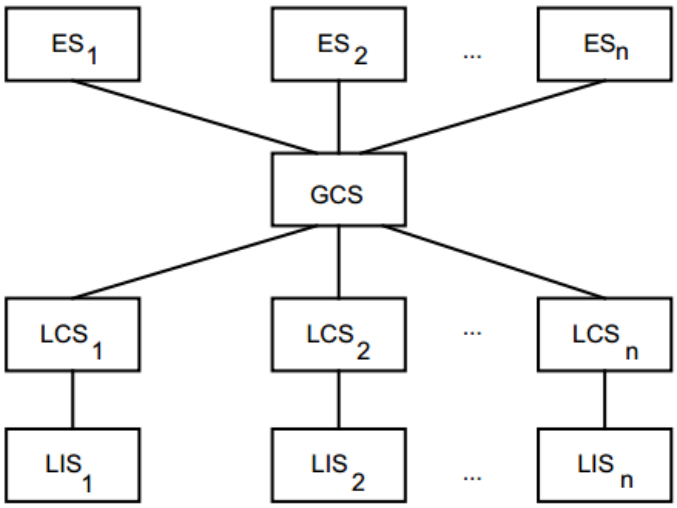


Illustration 4: Distributed Database Reference Architecture. Source [DDBS] chapter 1.7.9 Peer-to-Peer Systems

9. The top-down design for database have been introduced in the lecture and discuss it with your neighbor. Do research when to use the buttom-up design process. Use the book [DDBS] as source. Describe in you own words how the global conceptual schema (GCS) for DBS can be created.

"Bottom-up design involves the process by which information from participating databases can be (physically or logically) integrated to form a single cohesive multidatabase. There are two alternative approaches. In some cases, the global conceptual (or mediated) schema is defined first, in which case the bottom-up design involves mapping LCSs to this schema. This is the case in data warehouses, but the practice is not restricted to these and other data integration methodologies may follow the same strategy. In other cases, the GCS is defined as an integration of parts of LCSs. In this case, the bottom-up design involves both the generation of the GCS and the mapping of individual LCSs to this GCS."

Source: [DDBS] (chapter 4.1 Bottom-Up Design Methodology)

Practical tasks

1. What kind of ‘partitioning criteria’ and ‘partitioning methods’ are available for MS-SQL server and MySQL server?

2. What type of partitioning (method, criteria) was used for the example database ‘unix’/'coffee' on both DBS?

3. Implement a new database on MS SQL Server. (Sample data is available, e.g., ‘coffee.sql’. You can also use other datasets, if you like.

4. Implement a partitioning setup as you like on the database in the task before.

5. Is it possible to implement a partitioned schema on an already existing database? Describe the steps (SQL) which are necessary to implement partitioning.

After you are done, help your fellow students! :-)

Sources

shardDef: Angus Macdonald, What's the difference between sharding and partition?, 2015, https://www.quora.com/Whats-the-difference-between-sharding-and-partition?share=1
vertPar: SHAMKANT NAVATHE, Vertical Partitioning Algorithms for Database Design, 1984
DDBS: M. Tamer Ozsu, Principles of Distributed Database Systems 3rd Edition, 2011

Distributed Information Systems: MIN-VIS

Exercise 2: Distributed Queries – DRDBS

Theory questions

1. What does make distributed queries/SQL statements more difficult to execute by the DBS then the centralized/single-node queries?

Answer:
"In a distributed system, relational algebra is not enough to express execution strategies. It must be supplemented with operators for exchanging data between sites. Besides the choice of ordering relational algebra operators, the distributed query processor must also select the best sites to process data, and possibly the way data should be transformed. This increases the solution space from which to choose the distributed execution strategy, making distributed query processing significantly more difficult."

Source: [DDBS] (chapter 6.1 Query Processing Problem)

2. The query cost-factors for RDBS and DRDBS are diskaccess (I/O) and CPU. Additionally, DRDBS have to consider the communication cost (networking). Explain, why the communication can/is a problem for DRDBS to compete with the RDBS-performance. Estimate, how fast the network access should be and lookup for some real DRDBS and their network-requirements.

Answer:
However, modern distributed processing environments have much faster communication networks, as discussed in Chapter 2, whose bandwidth is comparable to that of disks. Therefore, more recent research efforts consider a weighted combination of these three cost components since they all contribute significantly to the total cost of evaluating a query [Page and Popek, 1985]. Nevertheless, in distributed environments with high bandwidths, the overhead cost incurred for communication between sites (e.g., software protocols) makes communication cost still an important factor.

Source: [DDBS] (chapter 6.2 Objectives of Query Processing)

MySQL cluster:
Network communication and latency. MySQL Cluster requires communication between data nodes and API nodes (including SQL nodes), as well as between data nodes and other data nodes, to execute queries and updates. Communication latency between these processes can directly affect the observed performance and latency of user queries. In addition, to maintain consistency and service despite the silent failure of nodes, MySQL Cluster uses heartbeating and timeout mechanisms which treat an extended loss of communication from a node as node failure.

Source: [mysqlReq]

3. Evaluate the cost of the following query, database and distribution:

4. Is using semijoins always wise? Which disadvantages does this technique have and when it is meaningfuller using just joins?

Answer:
"However, using semijoins may result in an increase in the number of messages and in the local processing time. The early distributed DBMSs, such as SDD-1 [Bernstein et al., 1981], which were designed for slow wide area networks, make extensive use of semijoins. Some later systems, such as R* [Williams et al., 1982], assume faster networks and do not employ semijoins. Rather, they perform joins directly since using joins leads to lower local processing costs. Nevertheless, semijoins are still beneficial in the context of fast networks when they induce a strong reduction of the join operand. Therefore, some query processing algorithms aim at selecting an optimal combination of joins and semijoins [¨Ozsoyoglu and Zhou, 1987; Wah and Lien, 1985]."

Source: [DDBS] (chapter 6.4.8 Use of Semijoins)

5. Simplify the following query:

Practice tasks

1. Evaluate the cost of the SQL-statements using the Query-Designer

a) A list of the top 500 users with the highest average answer score excluding community wiki / closed posts or users with less than 10 answers

```
SELECT
 user.Id as UID,
 Count(post.Id) AS Answers,
 CAST(AVG(CAST(Score AS decimal(6,2))) as decimal(6,2))
 AS Average_Answer_Score
FROM
 post
 INNER JOIN
 user ON user.Id = Owner_User_Id
WHERE
 post_Type_Id = 2
GROUP BY
 user.Id, Display_Name
HAVING
 Count(post.Id) > 10
ORDER BY
 Average_Answer_Score DESC
 limit 500;
```

b) Map all answers containing the word 'thank', reduce them after their referenced question (parent post) and count the occurences of each 'thank'-post per question.

```
 select
 ParentId as [Post Link],
 count(id)
from posts
where posttypeid = 2 and len(body) <= 200
 and (body like '%hank%')
group by parentid
having count(id) > 1
order by count(id) desc;
```

c) Map all comments of one user, reduce them after their score and display the occurrences of each comment-score.

```
DECLARE @UserId int = ##UserId##

SELECT
 Count(*) AS CommentCount,
 Score
FROM
 Comments
WHERE
 UserId = @UserId
GROUP BY
 Score
ORDER BY Score DESC
```

2. Implement a new database. Show example query costs for a selection of all attributes using the 'explain [partitions]'-statements.

3. Change the schema to a 'range columns' partitioned schema. Show the query costs and compare with the query costs you experienced before.

After you are done, help your fellow students! :-)

Sources

mysqlReq: MySQL, MySQL Cluster NDB, 2015, https://dev.mysql.com/doc/mysql-cluster-excerpt/5.1/en/mysql-cluster-overview-requirements.html
DDBS: M. Tamer Ozsu, Principles of Distributed Database Systems 3rd Edition, 2011

Distributed Information Systems: MIN-VIS

Exercise 3: Replication – DRDBS

Theory questions

1. What does replication mean? Which advantages and disadvantages does it have?

Answer:

- + The reasons for replication are reliability
- + Furthermore, read-only queries that access the same data items can be executed in parallel since copies exist on multiple sites (Performance)
- + If there are multiple copies of a data item, there is a good chance that some copy of the data will be accessible somewhere even when system failures occur (System availability)
- + Geographically system distribution / deployment and parallel read increase response time even if the architecture grows (Scalability)
- + Applications might need to maintain multiple copies of data to be in full operation (Application requirements)

- - On the other hand, the execution of update queries cause trouble since the system has to ensure that all the copies of the data are updated properly (Synchronization)
- - Replicated data is redundant and cost additional disk space (Resources)

Source: [DDBS] (chapter 3.2.5 Allocation Alternatives; chapter 13 Data Replication)

2. Please view over again the following table and discuss every field with your neighbor.

	Full replication	Partial replication	Partitioning
QUERY PROCESSING	Easy	Same difficulty ← →	
DIRECTORY MANAGEMENT	Easy or nonexistent	Same difficulty ← →	
CONCURRENCY CONTROL	Moderate	Difficult	Easy
RELIABILITY	Very high	High	Low
REALITY	Possible application	Realistic	Possible application

Illustration 5: Comparison of Replication Alternatives
Source: [DDBS] (chapter 3.2.5 Allocation Alternatives)

3. What is the difference between local and global transactions? Are local transactions possible in a DRDBS?

Answer:
"In the case of a partially replicated database, the number of physical data items for each logical data item may vary, and some data items may even be non-replicated. In this case, transactions that access only non-replicated data items are local transactions (since they can be executed locally at one site) ..."

Source: [DDBS] (chapter 13 Data Replication)

4. What does mutual consistency mean? Which advantages does weak and strong mutual consistency have?

Answer:

"One is mutual consistency, as discussed above, that deals with the convergence of the values of physical data items corresponding to one logical data item"

[DDBS] (chapter 13.1 Consistency of Replicated Databases)

"Strong mutual consistency criteria require that all copies of a data item have the same value at the end of the execution of an update transaction. This is achieved by a variety of means, but the execution of 2PC at the commit point of an update transaction is a common way to achieve strong mutual consistency.

Weak mutual consistency criteria do not require the values of replicas of a data item to be identical when an update transaction terminates. What is required is that, if the update activity ceases for some time, the values eventually become identical. This is commonly referred to as eventual consistency, which refers to the fact that replica values may diverge over time, but will eventually converge."

Source: [DDBS] (chapter 13.1.1 Mutual Consistency)

5. What does Transaction Consistency mean?

Answer:
"Transaction consistency, on the other hand, refers to the actions of concurrent transactions. We would like the database to remain in a consistent state even if there are a number of user requests that are concurrently accessing (reading or updating)"

Source: [DDBS] (chapter 10 Introduction to Transaction Management)

6. View over the following distributed transaction history (DBS-transaction logs are similar). Is the history serialization? Does it have mutual consistency?

7. Explain some advantages of performing the updates in a centralized or distributed way.

Answer:
"The advantages of centralized techniques are two-fold. First, application of the updates is easy since they happen at only the master site, and they do not require synchronization among multiple replica sites. Second, there is the assurance that at least one site – the site that holds the master copy – has up-to-date values for a data item."

Source: [DDBS] (chapter 13.2.3 Centralized Techniques)

"They are appropriate for collaborative applications with distributive decision/operation centers. They can more evenly distribute the load, and may provide the highest system availability if coupled with lazy propagation techniques."

Source: [DDBS] (chapter 13.2.4 Distributed Techniques)

8. Regarding update propagation and in the context of global transactions, what does eager and lazy propagation mean?

Answer:
"Eager techniques perform all of the updates within the context of the global transaction that has initiated the write operations. Thus, when the transaction commits, its updates will have been applied to all of the copies. Lazy techniques, on the other hand, propagate the updates sometime after the initiating transaction has committed."

Source: [DDBS] (chapter 13 Data Replication)

9. Explain in your own words what does limited and what does full replication transparency mean?

Answer:
"Certain replication protocols require each user application to know the master site where the transaction operations are to be submitted. These protocols provide only limited replication transparency to user applications. Other protocols provide full replication transparency by involving the Transaction Manager (TM) at each site. In this case, user applications submit transactions to their local TMs rather than the master site."

Source: [DDBS] (chapter 13 Data Replication)

10. For each of the four replication protocols (eager centralized, eager distributed, lazy centralized, lazy distributed), give a scenario/application where the approach is more suitable than the other approaches. Explain why. [DDBS] (exercise 13.1)

11. Describe automatic distribution. Explain the difference of manual database distribution. Which advantages does distribution have, which is manually and individually configured?

Practical tasks

MySQL Cluster Tasks

1. Look at the MySQL Cluster Setup, which type of nodes are existing in this setup?
   Name them; give a short description of their tasks and features. A physical deployment diagram could be helpful.

2. // How does the MySQL Cluster organize its data if it uses more than two DataNodes?

3. Implement a new table in your assigned group DB on MySQL Cluster. The table should use the features available in MySQL Cluster. Which DB engine is used for this?

Answer:
create table testTable (testColumn varchar(10)) engine ndb; // note: ndb stands for network database

4. Which differences are visible as a client in comparison to a single server system of MySQL? Make a list of pro's and con's.

MS SQL Server Tasks

1. Describe the two general methods for distributing data on multiple database servers.

Answer:
“A distributed transaction is a transaction in which all updates to all locations (where the distributed data is stored) are gathered together and executed synchronously. Distributed database systems use a method called two-phase commit to implement distributed transactions.”
“In contrast to the distributed transaction method, in which all data is distributed on all participating sites at the same time, data replication allows sites to have different data at the same time. Additionally, data replication is an asynchronous process. This means that there is a certain delay during which all copies of data are matched on all participating sites. (This delay can last from a couple of seconds to several days or weeks.)”
“The Microsoft Distributed Transaction Coordinator (DTC) supports distributed transactions using two-phase commit.” (Wich is not going to be used in this exercise)

Source: [MSSQL] (chapter 18 Distributed Data and Methods for Distributing)

2. Explain, how MS SQL Server does replication once with the help of transaction logs and on the other hand with the help of triggers.

Answer:
"If selected rows need to be replicated, the system starts a new process that reads the data from the transaction log and sends it to one or more target databases.

The other method is based on triggers. The modification of a table that contains data to be replicated fires the corresponding trigger, which in turn creates a new table with the data and starts a replication process."

"The Database Engine uses both concepts: the transaction log method for transactional replication and triggers for merge replication. (Transactional and merge replications are described in detail later in this chapter.)"

Source: [MSSQL] (chapter 18 SQL Server Replication: An Overview)

3. Explain the following MS SQL terms:
Roles: Publisher, Distributor, Subscriber
Units: Publication, Article

Answer:
"Publisher (or publishing server): Maintains its source databases, makes data available for replication, and sends the modified data to the distributor
Distributor (or distribution server): Receives all changes to the replicated data from the publisher and stores and forwards them to the appropriate subscribers
Subscriber (or subscription server): Receives and maintains published data
A database server can play many roles in a replication process"

Source: [MSSQL] (chapter 18 Publishers, Distributors, and Subscribers)

"The unit of data to be published is called a publication. An article contains data from a table and/or one or more stored procedures. A table article can be a single table or a subset of data in a table."
"A publication contains one or more articles. Each publication can contain data only from one database."

Source: [MSSQL] (chapter 18 Publications and Articles)

4. Why do you need a primary key for data replication? Which replication type requires a primary key? [MSSQL] (chapter 18 Exercises)

Answer:
Transactional Replication: "All tables published using transactional replication must explicitly contain a primary key. The primary key is required to uniquely identify the rows of the published table, because a row is the transfer unit in transactional replication."
Source: [MSSQL] (chapter 18 Transactional Replication)

5. How can you limit network traffic and/or database size? [MSSQL] (chapter 18 Exercises)

Answer:
"Transactional replication can replicate tables (or parts of tables) and one or more stored procedures. The use of stored procedures by transactional replication increases performance, because the amount of data to be sent over a network is usually significantly smaller. Instead of replicated data, only the stored procedure is sent to the subscribers, where it is executed."
Source: [MSSQL] (chapter 18 Replication Types)
"A stored procedure article can contain one or more stored procedures that exist at the publication time in the database."
Source: [MSSQL] (chapter 18 Publications and Articles)

We can fragment the data or do sharding by applying filter on publications. This will decrease the number of replicas, which are needed to be synchronized with high network-cost. On the other hand the high availability and geographically based response time will be decreased.

6. When does the system use the Log Reader agent, the Merge agent, and the Snapshot agent, respectively? [MSSQL] (chapter 18 Exercises)

Answer:
Transactional Replication: "The Log Reader agent searches for marked transactions and copies them from the transaction log on the publisher to the distribution database."
"Before transactional replication can begin, a copy of the entire database must be transferred to each subscriber; this is performed by executing a snapshot."
Source: [MSSQL] (chapter 18 Transactional Replication)

Snapshot Replication uses the Snapshot agent. This replication is used for asynchronous, long-term copies or for one-time copies.

The Merge agent is used in Merge Replication for enabling the Subscriber to write on their subscribed data. This can cause conflicts and therefore a more complicated synchronizing method (Merge Agent) must be used. In case of conflicts, you can use time-based or custom-based priorities with your own business rules.

After you are done, help your fellow students! :-)

Sources

DDBS: M. Tamer Ozsu, Principles of Distributed Database Systems 3rd Edition, 2011
MSSQL: Dušan Petković, Microsoft SQL Server 2012 A Beginner Guide, 5th Edition, 2012

Distributed Information Systems: MIN-VIS

Exercise 4: Transactions – DRDBS

Advance Organizer:

- Distributed transactions are often implemented as the 2-phase-commit-protocol (2PC). To understand, how 2PC works, it is recommended to type in and execute all steps manually. Many DBS are already providing XA transactions, which is implemented as 2PC. XA itself is "A standard interface for coordinating distributed transactions, allowing multiple databases to participate in a transaction while maintaining ACID compliance" [mysql_ref] (chapter MySQL Glossary). The following figure shows the structure of Open XA:

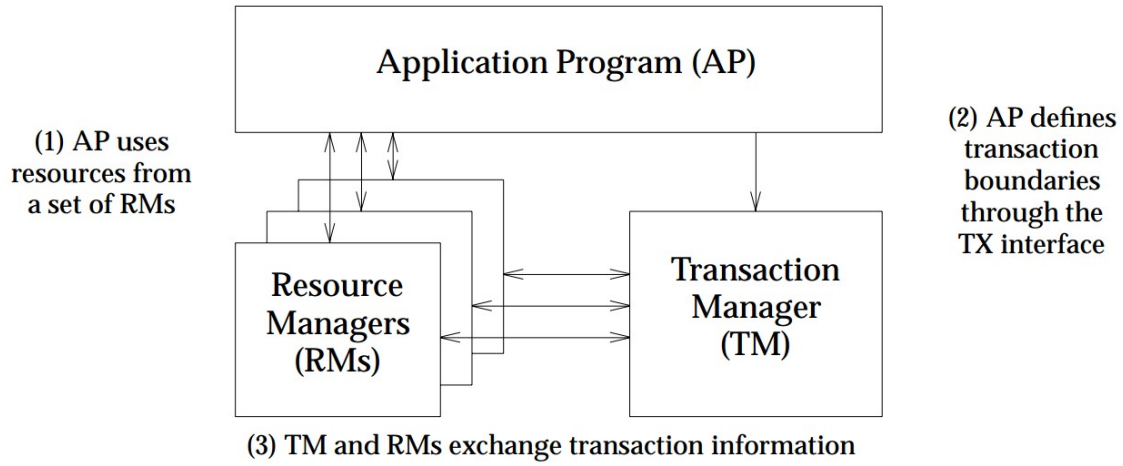


Illustration 6: Simple deployment model of Open XA [oXA]

- Deadlocks can occur, when transactions are blocking each other by not releasing their write blocks until the other transaction releases his lock and the DBS recognize it as a deadlock and rollback one of the waiting transactions [mysql_ref] (chapter MySQL Glossary). Since DRDBS consists of several hosts, a central resource manager is needed to maintain and granting the locks to the database clients. If there is no such a resource manager, the clients are responsible to mark the data they are going to modify and to respect the marks made by other clients. If locking data resources is not provided by a (DR)DBS, data consistency is not guaranteed after every modification / write operation. The resource manager of a DRDBS should handle locks similarly like non-distributed DBS's. Therefore the deadlocks can also be caused in the same way as in single DBS as follows:

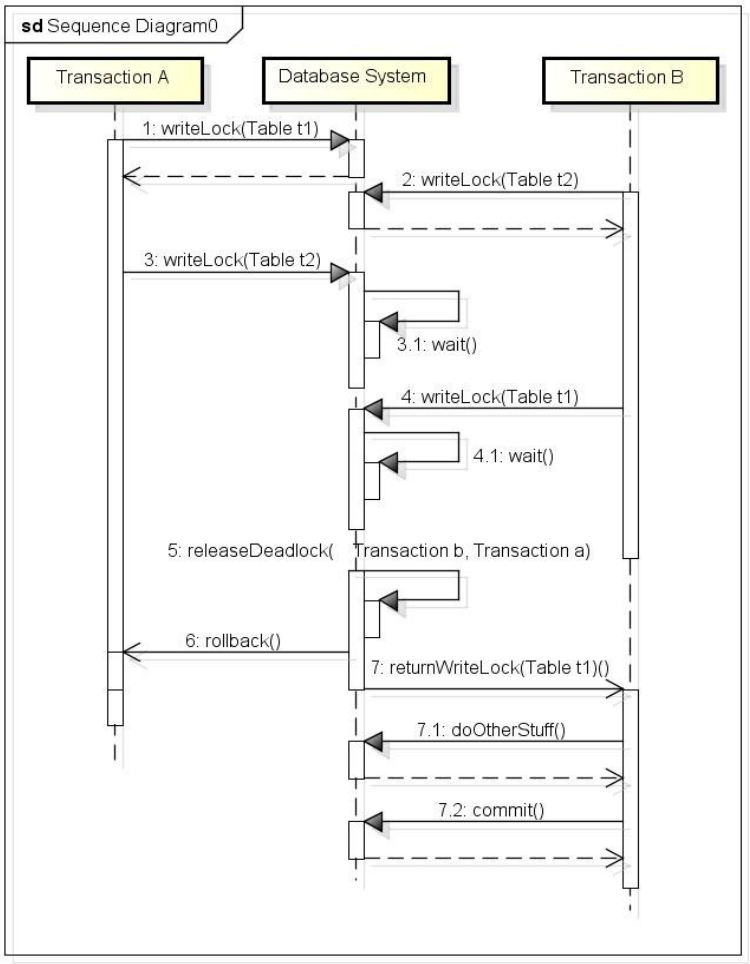


Illustration 7: Provoke Deadlock

Theory questions

1. What are TP monitors and what are there advantages? How do DRDBS's profit of them?

"In general, a TPM provides the following functionality:

- Coordinating resources
- Balancing loads
- Creating new processes as/when needed
- Providing secure access to services
- Routing services
- Wrapping data messages into messages
- Unwrapping messages into data packets/structures
- Monitoring operations/transactions
- Managing queues
- Handling errors through such actions as process restarting
- Hiding inter-process communications details from programmers"

Source: [tmp_func]

"The result is that the database server does not need to do all of the work of managing the consistency/correctness of the database; the **TP Monitor** makes sure that groups of updates take place together or not at all."

Source: [tpm_acid]

“Modern TP monitors provide support for the construction and administration of such large applications, built up from multiple subsystems such as databases, legacy systems, and communication systems. A TP monitor treats each subsystem as a resource manager that provides transactional access to some set of resources. The interface between the TP monitor and the resource manager is defined by a set of transaction primitives, such as begin transaction, commit transaction, abort transaction, and prepare to commit transaction (for two-phase commit).”

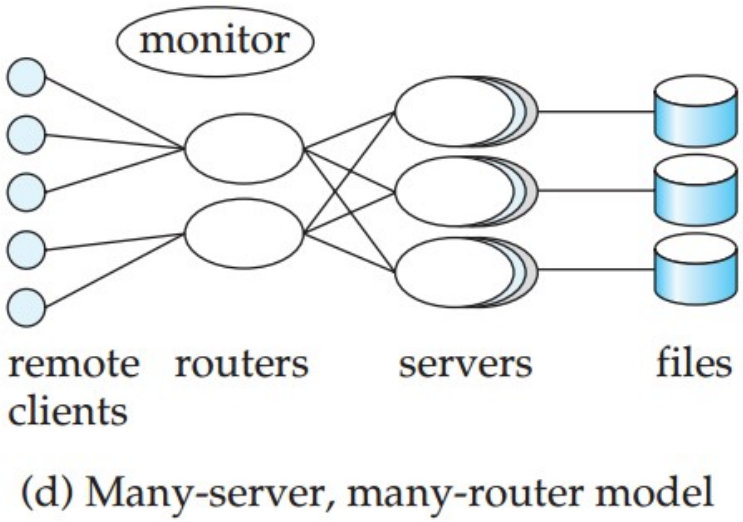


Illustration 8: multiple TPM's for distributed DBS.
Source: [DBSC]

“This model supports independent server processes for multiple applications; further, each application can have a pool of server processes, any one of which can handle a client session. The request can, for example, be routed to the most lightly loaded server in a pool. As before, each server process can itself be multi-threaded, so that it can handle multiple clients concurrently. As a further generalization, the application servers can run on different sites of a parallel or distributed database, and the communication process can handle the coordination among the processes.”

Source: [DBSC] (chapter 26.1 Transaction-Processing Monitors)

In summary, the TPM is an extended application server with higher sophisticated functionalities.

2. How does Open XA work? Explain the two-phase-commit protocol.

Before an XA transaction is initiated, a network-connection to a database must be established. The next step is to define a XA transaction with "XA start [name]; [SQL statements]; XA end [name]". After that, the defined transaction can be executed in two phases. First phase can be started with "XA prepare [name]" and the second with "XA commit [name]" [mysql_ref] (chapter 13.3.7.1 XA Transaction SQL Syntax). Between the two phases, the client must wait for a positive acknowledge from all servers. If one server couldn't pre-execute successfully, a XA rollback [name] must be send to all server to revoke the transaction. One example looks like that:

ssh mysq@vmvia1

mysql -uroot test

XA start 'myXA';

select * from helloworld;

XA end 'myXA';

XA prepare 'myXA'; // DBS will acknowledge, if transaction was preexecuted successfully (not commited yet)

XA commit 'myXA'; // commiting the transaction and make the changes globally available

3. Draft the same activity model for provokating deadlocks on distributed DBS, which does sharding without replication.

Practice tasks

MySQL Cluster

1. Try to provoke a deadlock on the cluster. Explain your steps. How can a deadlock occur or can it even occur on MySQL Clusters?

2. Regarding the MySQL Cluster XA transactions, where is the transaction manager being executed?

“The MySQL implementation of XA MySQL enables a MySQL server to act as a Resource Manager that handles XA transactions within a global transaction. A client program that connects to the MySQL server acts as the Transaction Manager. “

Source: [mysql_ref] (chapter 13.3.7 XA Transactions)

MSSQL Server

1. In case you want to perform distributed transactions, which isolation levels are available on MSSQL 2008 R2?

2. Write two distributed transactions on the “sakila” DB. Both transactions should update the same data. Query them at the same time. What happens?

3. Try to solve the problem using the deadlock_priority setting. The isolation level should be READ COMMITTED. Show and explain your code.

After you are done, help your fellow students! :-)

Bibliography

mysql_ref: MySQL / Oracle, MySQL Reference / Manual 5.6-en, 2015
oXA: The Open Group, The XA Specification, 2015,
http://pubs.opengroup.org/onlinepubs/009680699/toc.pdf
tmp_func: Cory Janssen, Transaction Processing Monitor (TPM), 2015,
https://www.techopedia.com/definition/465/transaction-processing-monitor-tpm
tpm_acid: Christopher Browne, 5. Transaction Processing Monitors, 2015,
http://linuxfinances.info/info/tpmonitor.html
DBSC: Silberschatz, Korth, Sudarshan, Database System Concepts; 6th edition, 2011

Distributed Information Systems: MIN-VIS

Exercise paper 5: Processing – M/R

Theory

1. Explain what Map-Reduce is and how it works with the help of a draft or model.

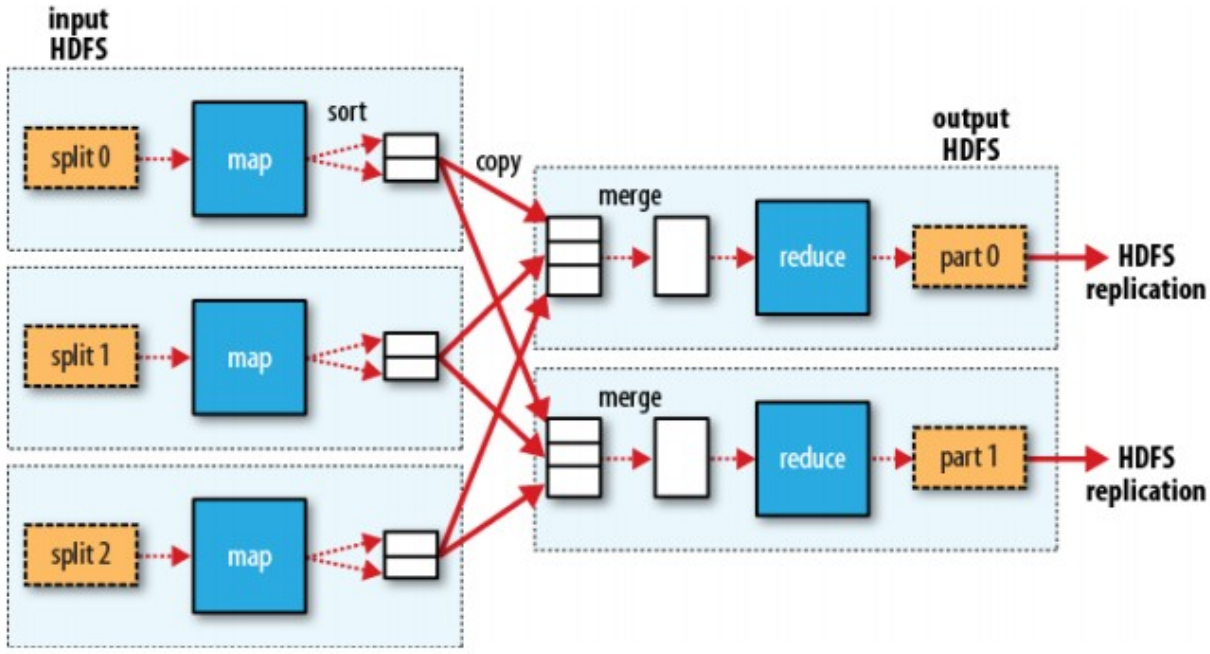


Illustration 9: MapReduce data flow with multiple reduce tasks.

Source: [HDP_GD] (chapter 2 Data Flow)

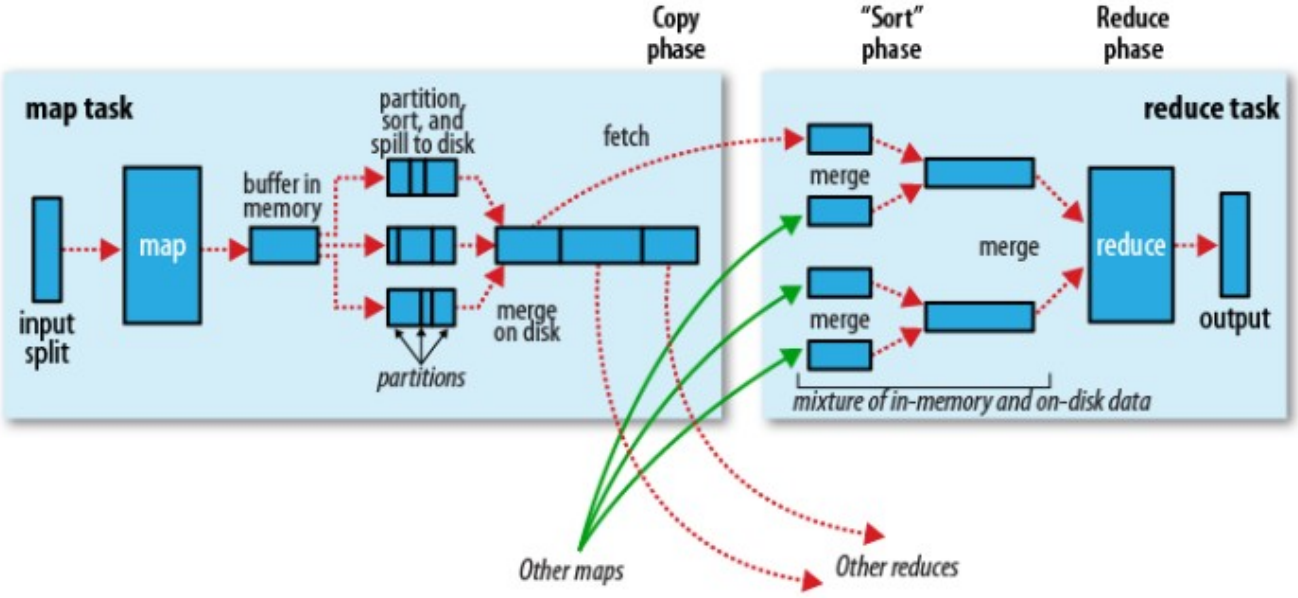
2. Which advantage does Map-Reduce have, working like you described in task 1 compared to relational database systems.

3. Map-Reduce is based on sharding and the distribution of query-programs. Write a pseudo-program, which maps and reduces user-defined data. Show the connection between the sharding and distributed M/R-jobs.

4. Replication does distribute copies of data. How does it help to parallelize programs? What other advantages does replication have for Map-Reduce?

5. Map-Reduce does sorting and shuffling. Explain in which part of the M/R-process this happens and why it is necessary.

"MapReduce makes the guarantee that the input to every reducer is sorted by key. The process by which the system performs the sort — and transfers the map outputs to the reducers as inputs — is known as the shuffle."



"Before it writes to disk, the thread first divides the data into partitions corresponding to the reducers that they will ultimately be sent to. Within each partition, the background thread performs an in-memory sort by key,..."

Source: [HDP_GD] (chapter 6 Shuffle and Sort)

The reduce phase merges all key-value-pairs after the key. Sorting them after the key and before the merge phase will start, will increase the performance.

6. M/R offers to combine the results from the map-phase. What is the difference between combine and reduce?

"Before it writes to disk, the thread first divides the data into partitions corresponding to the reducers that they will ultimately be sent to. Within each partition, the background thread performs an in-memory sort by key, and if there is a combiner function, it is run on the output of the sort. Running the combiner function makes for a more compact map output, so there is less data to write to local disk and to transfer to the reducer."

Source: [HDP_GD] (chapter 6 Shuffle and Sort)

Combine and reduce are doing the same thing in different time during the whole M/R process. Combine will be done, like described above, with sort right before the map-results will get written onto disk. Reduce will be executed after the merge/"sort" process.

7. Write pseudo-code, which gets the se_user.xml as input and returns the following output:

   {"[a-h]", 4}: {Donald, Daisy, Dagobert, Goofy}

   {"[i-q]", 2}: {Mickey, Minnie}

   {"[r-z]", 3}: {Tick, Trick, Track}

   EOF

   Explaination: The key consists of a string, which represents the alphabetic intervals A-H, I-Q and R-Z, and a number, which contains the amount of names this key has. The value is a string, which contains all firstnames of se_user.xml seperated by comma. In order to get the firstnames of se_user.xml, assume that every map-call gets as input-parameter a User-object, which provides the function 'getFirstname():String'.

```
map ( Int key, Person value, MResult r)
     String name <= value.getFirstname();
     char firstLetter = name.charAt(0).toUpperCase();
     String newKey;

     if(firstLetter < 'I')
          newKey = "[a-h]";
     else if(firstLetter < 'R')
          newKey = "[i-q]";
     else
          newKey = "[r-z]";

     r.write(newKey, name);

reduce ( String key, String[] value, RResult r)
     String newValue = "";

     for(int i = 0; i<value.length; i++)
          newValue += ", " + value[i];

     newValue = newValue.substring(2);

     r.write(key + ", " + value.length, newValue);
```

8. Regarding the results of task 9: If the number of firstnames can be removed or ignored, how would you change the pseudo-code to get a sorted list of the firstnames? The output should look like that. To add the intervals into the list, you can use a global flag 'intervalAdded' for help:

A-H: 1
Daisy: 1
Donald: 1
.....
I-Q: 1
...
R-Z: 1
Tick: 1
....
EOF

Practice

1. Warm up: Execute the following commands and try to understand them. If you are interested in how these programs are working, the corresponding source files are in ${HADOOP_HOME}/share/hadoop/mapreduce/sources

```
mkdir ~/hadoop; cd ~/hadoop
example=${HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-
examples-2.4.0.jar
```

wget ftp://anonymous@vmvia1/hadoop/etc/randomTextWriter-site.xml .
hdfs dfs -rm -r -f words

```
hadoop jar $example randomtextwriter -conf randomTextWriter-
site.xml words

hdfs dfs -rm -f -r wordcount

inFormat="-D
mapreduce.job.inputformat.class=org.apache.hadoop.mapreduce.lib.
input.SequenceFileInputFormat"

hadoop jar $example $inFormat words wordcount
hdfs dfs -cat wordcount/\* |less
```

http://stackoverflow.com/questions/18395998/hadoop-map-reduce-secondary-sorting%5D

2. When it is the best using SequenceFileOutputFormat and why?

Sequence files are binary documents and are for processing the output with another M/R-Job, since its compression and parallel readability increases the overall performance.

3. Does Hadoop sort the processed data? If yes, after what criteria and when (after mapping, after reducing)?

Hadoop sorts after the Key right afer the mapping-function.

4. Write a program which lists all Displaynames from coffee/Users.xml. Please research about how to use the XmlInputFormat from Mahout.

```
mkdir ~/hadoop
cd ~/hadoop
rm *.java *.jar
wget -r -nd ftp://anonymous@vmvia1/hadoop/stackexchange/\*
rm -rf classes
mkdir classes
jars=${HADOOP_HOME}/share/hadoop/common/hadoop-common-2.4.0.jar:
${HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-client-
core-2.4.0.jar:${HADOOP_HOME}/share/hadoop/yarn/lib/commons-
logging-1.1.3.jar:${HADOOP_HOME}/share/hadoop/yarn/lib/jdom-
1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/xmlinputformat.jar

javac -cp $jars -d classes/ *.java
jar -cvfe coffeexml.jar hadoop.stack.sortUser.CoffeeXML -C
classes .
hdfs dfs -rm -r -f outcoffee
hadoop jar coffeexml.jar coffee/Users.xml outcoffee
hdfs dfs -cat outcoffee/\* |less
```

5. Implement the following three SQL-statements as M/R-jobs. Alternatively, write pseudo-code, which describes mapper- and reducer-tasks and its input and output files.

a) Print the id's of posts with very short body text

Original SQL statement and link:

select Id as [Post Link], Body, Score from Posts where Len(Body) <= 70 && Len(Body) >= 40

http://data.stackexchange.com/stackoverflow/query/873/posts-containing-a-very-short-body

```
mkdir ~/hadoop
cd ~/hadoop
rm *.java *.jar
wget -r -nd ftp://anonymous@vmvia1/hadoop/stackexchange/shortPost/\*
rm -rf classes
mkdir classes
jars=${HADOOP_HOME}/share/hadoop/common/hadoop-common-2.4.0.jar:$
{HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.4.0.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/commons-logging-1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/jdom-1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/xmlinputformat.jar

javac -cp $jars -d classes/ *.java
jar -cvfe coffeexml.jar hadoop.stack.shortPost.CoffeeXML -C classes .
hdfs dfs -rm -r -f outcoffee
hadoop jar coffeexml.jar coffee/Posts.xml outcoffee 40 70
hdfs dfs -cat outcoffee/\* |less
```

b) Map all answers containing the word 'thank', reduce them after their referenced question (parent post) and count the occurences of each 'thank'-post per question.

Original SQL statement and link:

```
select ParentId as [Post Link], count(id) from posts
where posttypeid = 2 and len(body) <= 200 and (body like '%hank%')
group by parentid having count(id) > 1 order by count(id) desc;
```

http://data.stackexchange.com/stackoverflow/query/886/posts-with-many-thank-you-answers

c) Map all comments of one user, reduce them after their score and display the occurrences of each comment-score. The result is a comment-score distribution.

Original SQL statement and link:

```
DECLARE @UserId int = ##UserId##

SELECT Count(*) AS CommentCount, Score FROM Comments
WHERE UserId = @UserId GROUP BY Score ORDER BY Score DESC
```

http://data.stackexchange.com/stackoverflow/query/947/my-comment-score-distribution

```
mkdir ~/hadoop
cd ~/hadoop
rm *.java *.jar
wget -r -nd ftp://anonymous@vmvia1/hadoop/stackexchange/scoreDistribution/\*
rm -rf classes
mkdir classes
jars=${HADOOP_HOME}/share/hadoop/common/hadoop-common-2.4.0.jar:$
{HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.4.0.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/commons-logging-1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/jdom-1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/xmlinputformat.jar

javac -cp $jars -d classes/ *.java
jar -cvfe coffeexml.jar hadoop.stack.scoreDistribution.CoffeeXML -C classes .
hdfs dfs -rm -r -f outcoffee tmpFile
hadoop jar coffeexml.jar unix/Comments.xml outcoffee
#hadoop jar coffeexml.jar coffee/Comments.xml outcoffee

lookAt=outcoffee/\*
#lookAt=tmpFile/\*

hdfs dfs -cat $lookAt |less
```

6. Implement the following join query as M/R job. Print the number of answers for each person as shown in the projection of the query. Only the posts of an user will be considered, which were marked as an answer, which is why a join is needed to be implemented. Posts, which has been marked as an answer from the same user, must be ignored.

```
select
    count(a.Id) as [Accepted Answers],
    sum(case when a.Score = 0 then 0 else 1 end) as [Scored Answers],
    sum(case when a.Score = 0 then 1 else 0 end) as [Unscored Answers],
    sum(CASE WHEN a.Score = 0 then 1 else 0 end)*1000 / count(a.Id) / 10.0
      as [Percentage Unscored]
from Posts q inner join Posts a on a.Id = q.AcceptedAnswerId
where and q.OwnerUserId !=  a.OwnerUserId and a.postTypeId = 2
group by a.OwnerUserId
```

http://data.stackexchange.com/stackoverflow/query/7521/how-unsung-am-i

```
mkdir ~/hadoop
cd ~/hadoop
rm *.java *.jar
wget -r -nd ftp://anonymous@vmvia1/hadoop/stackexchange/popular/\*
rm -rf classes
mkdir classes
jars=${HADOOP_HOME}/share/hadoop/common/hadoop-common-2.4.0.jar:$
{HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.4.0.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/commons-logging-1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/jdom-1.1.3.jar:$
{HADOOP_HOME}/share/hadoop/yarn/lib/xmlinputformat.jar

javac -cp $jars -d classes/ *.java
jar -cvfe coffeexml.jar hadoop.stack.popular.CoffeeXML -C classes .
hdfs dfs -rm -r -f outcoffee tmpFile
#hadoop jar coffeexml.jar unix/Posts.xml unix/Posts2.xml outcoffee
hadoop jar coffeexml.jar coffee/Posts.xml coffee/Posts2.xml outcoffee

lookAt=outcoffee/\*
#lookAt=tmpFile/\*

hdfs dfs -cat $lookAt |less
```

After you are done, help your fellow students! :-)

Bibliography

HDP_GD: Tom White, Hadoop: The Definitive Guide; 3rd Edition, 2012

Distributed Information Systems: MIN-VIS

Exercise 12: Cloud Design Patterns - Cloud

Theory questions

12. Do research about other definitions for cloud and list them here. Do you agree or disagree with their statements?

13. What is the advantage using a cloud? Does even an advantage exist :-)? Optionally, you can compare the advantages with the disadvantages and discuss, if or how this technology is profitable for huge companies and the society.

14. To fullfill the presented cloud characteristics ([nistDefCloud]) and therefore in some kind the realization of the cloud itself, which conditions or technologies were and are necessary to reach? Besides research, make a mind-map about the fundamentals of the cloud.

15. Name some use-cases, for which clouds can provide a bigger benefit.

16. List around 3 solutions/products for each model: SaaS, PaaS, IaaS. Do all fit in one of these category easily or are there some 'hybrid' services? Should Gaming as a Service (GaaS) exist?

17. Besides the different usage or purpose for each deployment models (private-, community-, .. -cloud), do they require different technologies or interfaces to their corresponding target group? Which other differences or similarities do these cloud-types have with each other?

18. In new areas like cloud, best practices and approaches have been crystallized and envolve these areas. Please give some points, why this is happening. Write down your understanding about research, exhibition platforms, cooperations or prototyping tests and theories generally or in regard of clouds.

19. Which considerations for implementing autoscaling are necessary? Look up in [CLOUD]. Describe in your own words, how the throttling and competing consumer pattern can be related to autoscaling.

20. Lookup for other pattern or guidance / service models. A single one is enough. Discuss it with your partner / neighbor and present your two topics to the rest of the class.

Practical task / project work

1. Create groups with around 3 to 5 people each and work on a provided and chosen project:

Your topic:
If you have something in mind, what you like or always have liked to implement or todo research about it with others, discuss this topic within your group. It is ok, if your group does not come to an agreement. In this case simply attend to a second (global) round and present your topic there and discuss about the topics of your colleques. You must find at least two other people, who like to involve, participate and help out in a corporated topic.

For projects, which have been already united in the first round and have more then three people: The students within there and who are interested in the second round (because to present his/her own project/topic or participate on another ones), are welcome to attend and to help out creating groups. If you don't come to an topic-agreement in the second round, you can still return to your group from the first round. Only [number of member]-3 are allowed to come to the second round from a frist-round-project to keep the first projects 'alive'.

In case you like to attend in a project, where already 5 people are, talk with them, if it is possible to split the project into two groups and if you could participate in one of them (=> two 3-people-groups).

VIS-topic:
For all, who remained without a group can unite for the following topic: The current distributed DBSs in this lecture may currently not be optimal for teaching purposes. Form groups to enhance and rework the learning experience by setting up different distributed databases and installing the Stackexchange dataset there. Then the groups do testing and present the functionalities, which the DBS has or can do better then others.

This kind of projects can influence the learning experience for your successors. Besides the technical planning and realization, you will also be responsible for social effects, the cultural impacts (students from higher semesters are handing their experiences over to students from lower semesters) and the budget/resource limitations (available hardware capacities, ...).

The following list shows several distributed databases, datastores, frameworks or other technologies, that your group can start with:

Iteration	Name	Description	Website
SQL - relational			
1	MS SQL Server	Support for 2PC and lazy update propagation	http://www.microsoft.com/sqlserver/
New	Postgre SQL	Master-Slave replication	http://www.postgresql.org/ [i]
New	CitusDB	Master, which replicates and distributes data accross PostgreSQL-nodes	https://www.citusdata.com/
New	VoltDB	In-Memory with ACID properties	http://voltdb.com/
New	Berkeley DB	Single-master multi-replica database	http://www.oracle.com/us/products/database/berkeley-db/overview/index.html
New	Phoenix	SQL-DBMS over a HBase-store	http://phoenix.apache.org/
New	CUBRID	Object-oriented with static sharding	http://www.cubrid.org/
NoSQL			
1	Hadoop	Map-Reduce with DFS	https://hadoop.apache.org/
New	Hbase	Strong consistent with auto-sharding on files	http://hbase.apache.org/
New	Cassandra	Auto-sharding with SQL-like queries	https://cassandra.apache.org/
New	MongoDB	Document-oriented + M/R	https://www.mongodb.org/
New	BigCouch	Document-oriented + M/R	http://bigcouch.cloudant.com/
New	ClusterPoint	Document-oriented	https://www.clusterpoint.com/
New	Couchbase Server	Document-oriented + M/R	http://www.couchbase.com/
New	Aerospike database	In-Memory	http://www.aerospike.com/
New	Tarantool	In-Memory Lua DB	http://tarantool.org/
New	Redis	In-Memory key-value store	http://redis.io/
New	Scalaris	Transactional, key-value store	http://scalaris.zib.de/
New	FoundationDB	Transactional, key-value store	https://foundationdb.com/
New	Hypertable	Key-value-oriented	http://hypertable.com
New	Accumulo	Key-value store	https://accumulo.apache.org/

New	Oracle NoSQL Database	Key-value-oriented	http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html
New	Voldemort	Key-value store	http://www.project-voldemort.com/voldemort/
New	Riak	Key-value store	http://basho.com/products/#riak
New	Druid	Column-oriented using MySQL	http://druid.io/
New	Neo4j	Graph-oriented	http://neo4j.com/
New	OrientDB	Multi-model: graph-, document-, key/value- and object-oriented	http://orientdb.com/
Frameworks / Tools			
New	Gizzard	(Retired) Sharding framework for RDBs	https://github.com/twitter/gizzard
New	Jetpants	(Quite old,) Range-based sharding toolkit for MySQL	https://github.com/tumblr/jetpants
New	Spock Proxy	(Old) Range-based horizontal partitioning for MySQL nodes	http://spockproxy.sourceforge.net/
New	Scoop	CLI for data-transfer between RDDBS and Hadoop	https://sqoop.apache.org/
Commercial			
New	ClustrixDB	Clustered, shared-nothing, ACID, SQL DBS	http://www.clustrix.com/
New	NuoDB	On-demand, cloud, SQL DBS	http://www.nuodb.com/
New	ScaleDB	OLTP DBMS for data warehousing applications	http://www.scaledb.com/
New	dbShards	Sharding DBMS on top of MySQL	http://codefutures.com/dbshards-features/

After you are done, help your fellow students! :-)

Sources

nistDefCloud: Peter Mell; Timothy Grance, The NIST definition of cloud computing.pdf, 2011
CLOUD: Homer, Sharp, Brader, Cloud Design Patterns, 2014

i Information about PostgreSQL clustering:
http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling
http://www.postgresql.org/docs/current/static/high-availability.html
http://www.repmgr.org/
http://manpages.ubuntu.com/manpages/lucid/man1/pg_wrapper.1.html
http://manpages.ubuntu.com/manpages/hardy/man8/pg_createcluster.8.html

B.1 CD with further Attachments

The CD contains:

- This thesis as PDF
- The exercises as separate PDF files
- The complete source code, which was covered in the chapter 6 'Exercises - Implementation' as an eclipse project
- The system documentation for the MySQL Cluster
- The system documentation for the Hadoop cluster