Processing Big Data : towards a novel, efficient and smart partitioning approach

Jorge GALICIA
jorge.galicia@ensma.fr

Supervised by : Ladjel BELLATRECHE
Amin MESMOUDI

Sept 13, 2018

Outline

Introduction

1. https://www.winshuttle.com/big-data-timeline/

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

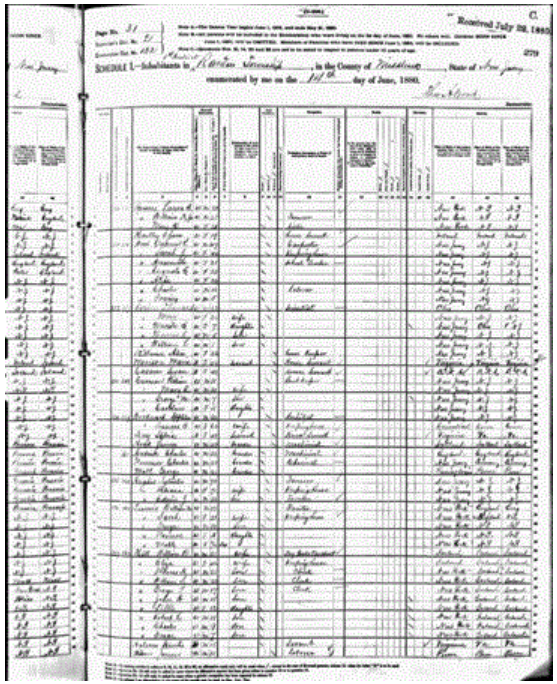
1. https://www.winshuttle.com/big-data-timeline/

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

1880 | U.S. Census



8 years to tabulate !

1. https://www.winshuttle.com/big-data-timeline/

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

- 1880 — U.S. Census
- 1944 — Growth of American university libraries

"Yale Library in 2040 would have approximately 200,000,000 volumes, which will occupy over 6,000 miles of shelves"

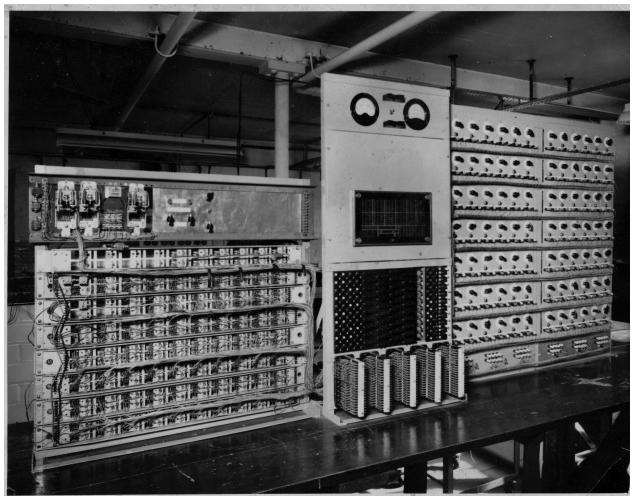
1. https://www.winshuttle.com/big-data-timeline/

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

- 1880 — U.S. Census
- 1944 — Growth of American university libraries
- 1961 — Scientific knowledge expansion
- 1966 — Centralized Computing Systems enter the scene



1. https://www.winshuttle.com/big-data-timeline/

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

- 1880 U.S. Census
- 1944 Growth of American university libraries
- 1961 Scientific knowledge expansion
- 1966 Centralized Computing Systems enter the scene
- 1970 Relational Database by Edgar F. Codd

Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. INTRODUCTION

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

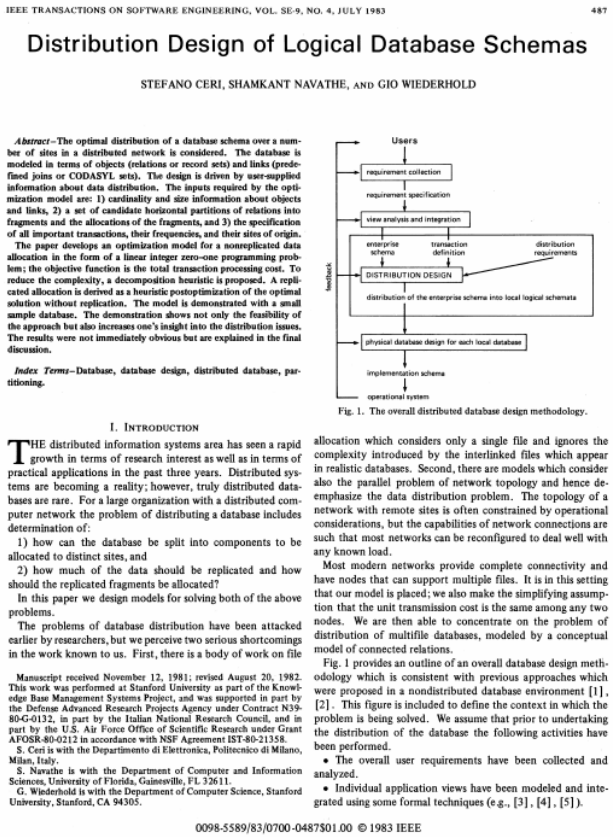
1. https://www.winshuttle.com/big-data-timeline/

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

- 1880 U.S. Census
- 1944 Growth of American university libraries
- 1961 Scientific knowledge expansion
- 1966 Centralized Computing Systems enter the scene
- 1970 Relational Database by Edgar F. Codd
- 1974 IBM System R system built as a research project
- 1978 Oracle launches the first commercially available RDBMS
- 1983 Information Growth



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-9, NO. 4, JULY 1983 487

Distribution Design of Logical Database Schemas

STEFANO CERI, SHAMKANT NAVATHE, AND GIO WIEDERHOLD

Abstract—The optimal distribution of a database schema over a number of sites in a distributed network is considered. The database is modeled in terms of objects (relations or record sets) and links (predefined joins or CODASYL sets). The design is driven by user-supplied information about data distribution. The inputs required by the optimization model are: 1) cardinality and size information about objects and links, 2) a set of candidate horizontal partitions of relations into fragments and the allocations of the fragments, and 3) the specification of all important transactions, their frequencies, and their sites of origin. The paper develops an optimization model for a nonreplicated data allocation in the form of a linear integer zero-one programming problem; the objective function is the total transaction processing cost. To reduce the complexity, a decomposition heuristic is proposed. A replicated allocation is derived as a heuristic postoptimization of the optimal solution without replication. The model is demonstrated with a small sample database. The demonstration shows not only the feasibility of the approach but also increases one's insight into the distribution issues. The results were not immediately obvious but are explained in the final discussion.

Index Terms—Database, database design, distributed database, partitioning.

Fig. 1. The overall distributed database design methodology.

I. Introduction

THE distributed information systems area has seen a rapid growth in terms of research interest as well as in terms of practical applications in the past three years. Distributed systems are becoming a reality; however, truly distributed databases are rare. For a large organization with a distributed computer network the problem of distributing a database includes determination of:

1) how can the database be split into components to be allocated to distinct sites, and

2) how much of the data should be replicated and how should the replicated fragments be allocated?

In this paper we design models for solving both of the above problems.

The problems of database distribution have been attacked earlier by researchers, but we perceive two serious shortcomings in the work known to us. First, there is a body of work on file allocation which considers only a single file and ignores the complexity introduced by the interlinked files which appear in realistic databases. Second, there are models which consider also the parallel problem of network topology and hence deemphasize the data distribution problem. The topology of a network with remote sites is often constrained by operational considerations, but the capabilities of network connections are such that most networks can be reconfigured to deal well with any known load.

Most modern networks provide complete connectivity and have nodes that can support multiple files. It is in this setting that our model is placed; we also make the simplifying assumption that the unit transmission cost is the same among any two nodes. We are then able to concentrate on the problem of distribution of multifile databases, modeled by a conceptual model of connected relations.

Fig. 1 provides an outline of an overall database design methodology which is consistent with previous approaches which were proposed in a nondistributed database environment [1], [2]. This figure is included to define the context in which the problem is being solved. We assume that prior to undertaking the distribution of the database the following activities have been performed.

- The overall user requirements have been collected and analyzed.
- Individual application views have been modeled and integrated using some formal techniques (e.g., [3], [4], [5]).

Manuscript received November 12, 1981; revised August 20, 1982. This work was performed at Stanford University as part of the Knowledge Base Management Systems Project, and was supported in part by the Defense Advanced Research Projects Agency under Contract N39-80-G-0132, in part by the Italian National Research Council, and in part by the U.S. Air Force Office of Scientific Research under Grant AFOSR-80-0212 in accordance with NSF Agreement IST-80-21358.

S. Ceri is with the Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

S. Navathe is with the Department of Computer and Information Sciences, University of Florida, Gainesville, FL 32611.

G. Wiederhold is with the Department of Computer Science, Stanford University, Stanford, CA 94305.

0098-5589/83/0700-0487$01.00 © 1983 IEEE

1. https://www.winshuttle.com/big-data-timeline/

Introduction

1988 Data warehouse foundations

An architecture for a business and information system

by B. A. Devlin
P. T. Murphy

The transaction-processing environment in which companies maintain their operational databases was the original target for computerization and is now well understood. On the other hand, access to company information on a large scale by an end user for reporting and data analysis is relatively new. Within IBM, the computerization of informational systems is progressing, driven by business needs and by the availability of improved tools for accessing the company data. It is now apparent that an architecture is needed to draw together the various strands of informational system activity within the company. IBM Europe, Middle East, and Africa (E/ME/A) has adopted an architecture called the E/ME/A Business Information System (EBIS) architecture as the strategic direction for informational systems. EBIS proposes an integrated warehouse of company data based firmly in the relational database environment. End-user access to this warehouse is simplified by a consistent set of tools provided by an end-user interface and supported by a business data directory that describes the information available in user terms. This paper describes the background and components of the architecture of EBIS.

The environment for information retrieval and reporting within the IBM Europe, Middle East, and Africa (E/ME/A) countries is today characterized by functional systems that deliver information to specific groups. This is often achieved in a redundant and costly way because organizational or functional barriers prevent the free transfer of information. With effort it is possible for any independent and innovative user or group to seek out new data and obtain it. Therefore, a new direction and strategy must be one that promotes and encourages the use of data and makes the required information available to every person in a particular interest group who has need of it.

Today, each function, such as marketing, administration, or finance, tends to have its own method of handling general reporting services and end-user access to data. Most functions are using some form of APL-based system to carry out their reporting and information requirements, but these systems are quite different from one another.

This situation involving eclectic systems, which provided the impetus for the development of a new, more uniform, architecture for information retrieval and reporting, is not unique to IBM. Many companies have developed along similar lines, where discrete needs for particular sets of information have motivated independent and often conflicting procedures for obtaining the necessary data from the established operational systems.

A number of limitations of the current query systems have been identified. Most of us when we are new to an organization experience the fact that data retrieval is not easy for the novice user. Similarly, when it comes to structuring and reporting data, the new user often finds that standard reports are difficult to define and change, requiring detailed knowledge of the data source. Today's informational systems are not directed at the entire potential end-user community. Because of the dependency on detailed

by Barry Devlin and Paul Murphy

Introduction

1988 Data warehouse foundations

1989 Business Intelligence 1.0

1993

Providing OLAP to User-Analysts: An IT Mandate

Introduction

Overview

Recently, there has been a great deal of discussion in the trade press and elsewhere regarding the coexistence of so-called transaction databases with decision support systems. These discussions usually revolve around the argument that the physical design required for acceptable performance of each is incompatible and that therefore, data should be stored redundantly in multiple enterprise databases: one for transaction processing, and the other for decision support type activities. Also, these same arguments usually confuse physical schema with logical and conceptual schema.

These arguments are fuzzy and imprecise. These arguments ignore the fundamental requirements of the types of analytical data models required for efficient information synthesis and also ignore the fact that the majority of enterprises have numerous, diverse data stores from which information needs to be synthesized. This paper defines a category of database processing: Online Analytical Processing (abbreviated OLAP). This paper defines the OLAP category, describes an enabling architecture for OLAP, and identifies the fundamental components and criteria for evaluating a given product's efficacy in its support of the OLAP category. Finally, a commercially available product is evaluated according to the rules for OLAP. In this paper the symbol DBMS denotes a database management system.

E.F. Codd, S.B. Codd and C.T. Salley

E.F. Codd Associates

by Codd, E.F., S.B. Codd and C.T. Salley

Introduction

1988 Data warehouse foundations

1989 Business Intelligence 1.0

1993

1995 World Wide Web Revolution

Introduction

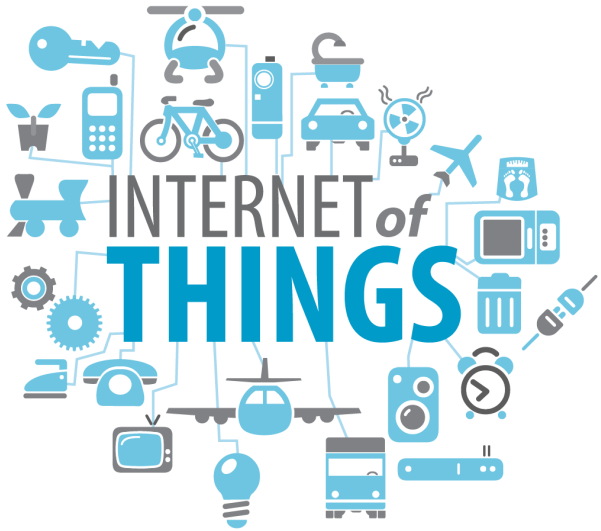
1988 Data warehouse foundations

1989 Business Intelligence 1.0

1993

1995 World Wide Web Revolution

1999 Introduction of Internet of Things



by Kevin Ashton

Web 2.0

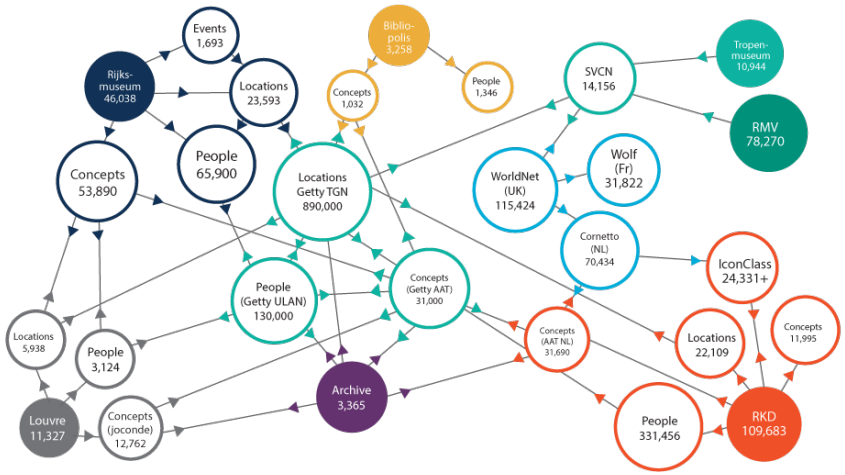
Introduction

- 1988 — Data warehouse foundations
- 1989 — Business Intelligence 1.0
- 1993
- 1995 — World Wide Web Revolution
- 1999 — Introduction of Internet of Things
- 2001 — Volume, Velocity, Variety
- 2006 — Open source solution for big data explosion

Laney Doug, "3D Data Management : Controlling Data Volume, Velocity, and Variety"

Introduction

- 1988 Data warehouse foundations
- 1989 Business Intelligence 1.0
- 1993
- 1995 World Wide Web Revolution
- 1999 Introduction of Internet of Things
- 2001 Volume, Velocity, Variety
- 2006 Open source solution for big data explosion
- 2009 Linked Data

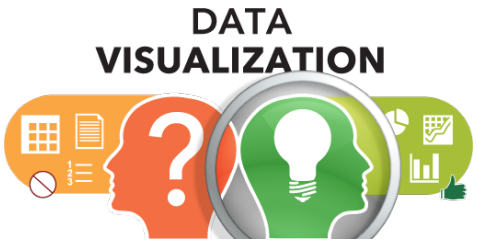


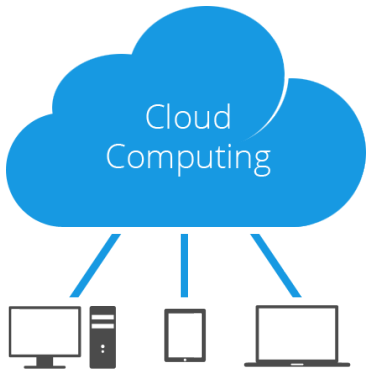
Introduction

- 1988 — Data warehouse foundations
- 1989 — Business Intelligence 1.0
- 1993
- 1995 — World Wide Web Revolution
- 1999 — Introduction of Internet of Things
- 2001 — Volume, Velocity, Variety
- 2006 — Open source solution for big data explosion
- 2009 — Linked Data
- 2010 — Apache Spark

Introduction

- 1988 Data warehouse foundations
- 1989 Business Intelligence 1.0
- 1993
- 1995 World Wide Web Revolution
- 1999 Introduction of Internet of Things
- 2001 Volume, Velocity, Variety
- 2006 Open source solution for big data explosion
- 2009 Linked Data
- 2010 Apache Spark
- 2011 New BI Trends
- 2015 SmartCities
- 2018 ...





Introduction

What about the hardware ?

Introduction

What about the hardware ?

Ideal database machine [16] :

- Single infinitely fast processor with an infinite memory with infinite bandwidth.
- And would be infinitely **cheap** (free)

Technology provides relatively cheap high performance processors, fast high-capacity disks, and high-capacity RAM memories.

Introduction

What about the hardware ?

Ideal database machine [16] :

- Single infinitely fast processor with an infinite memory with infinite bandwidth.
- And would be infinitely **cheap** (free)

Technology provides relatively cheap high performance processors, fast high-capacity disks, and high-capacity RAM memories.

As stated by DeWitt D. and Jim Gray in [16] the challenge is to **build** an infinitely **fast processor** out of infinitely **many processors** of finite speed, and to build an infinitely **large memory** with infinite memory bandwidth from infinitely **many storage units** of finite speed.

Introduction

What about the hardware ?

Ideal database machine [16] :

- Single infinitely fast processor with an infinite memory with infinite bandwidth.
- And would be infinitely **cheap** (free)

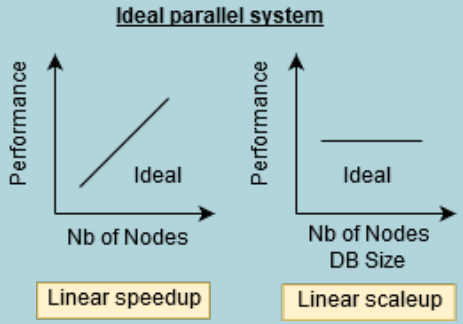
Technology provides relatively cheap high performance processors, fast high-capacity disks, and high-capacity RAM memories.

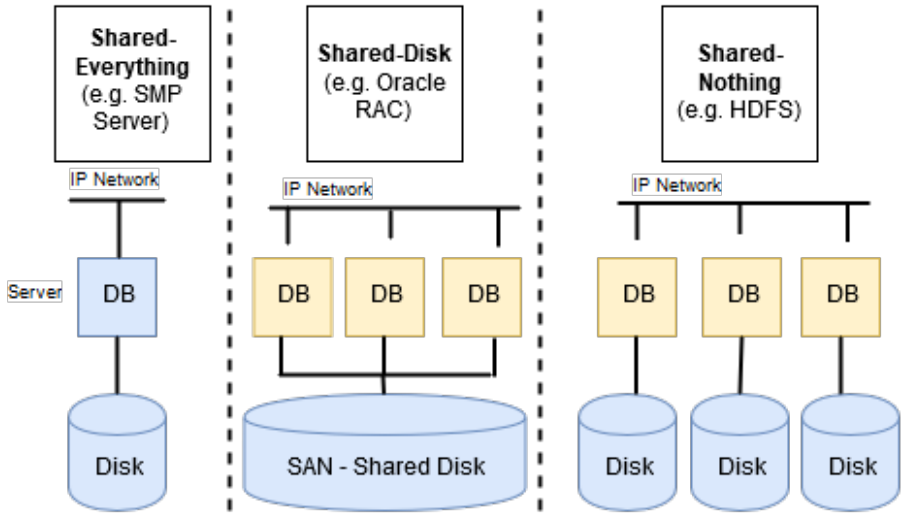
Divide a big problem into many smaller ones to be solved in parallel.

Introduction

Cluster Computing Revolution

- A cluster is a set of independent server nodes intereconnected to share resources and form a single system.
- Machines close to each other and are connected with dedicated high speed LANs and switches.

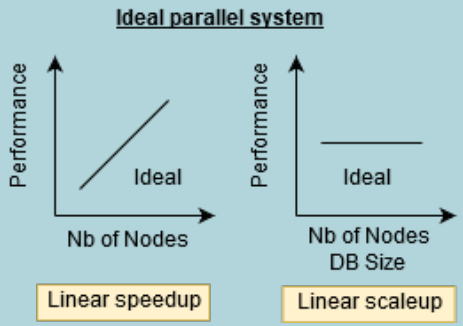


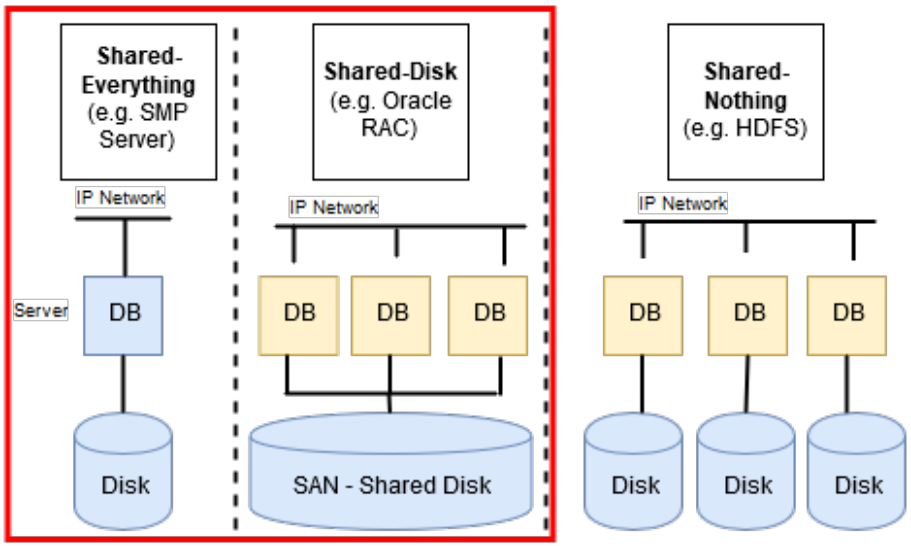


Introduction

Cluster Computing Revolution

- A cluster is a set of independent server nodes intereconnected to share resources and form a single system.
- Machines close to each other and are connected with dedicated high speed LANs and switches.



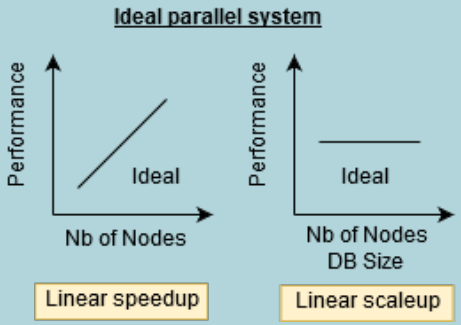


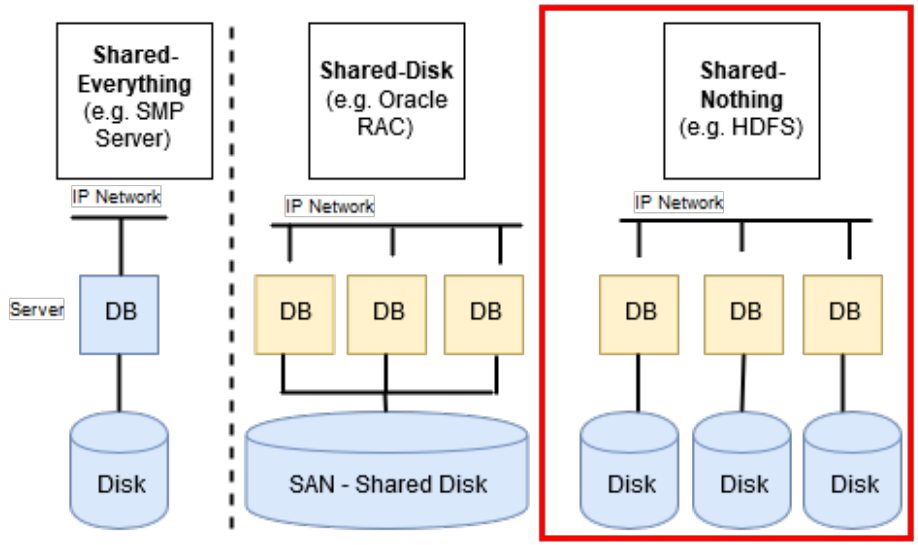
+ Simplicity, Load Balancing
- Limited extensibility, Interference → low scalability

Introduction

Cluster Computing Revolution

- A cluster is a set of independent server nodes intereconnected to share resources and form a single system.
- Machines close to each other and are connected with dedicated high speed LANs and switches.





+ Lower cost, high extensibility, high availability
- More complex, **load balance**→relies on database partitioning, new nodes → data reorganization.

Introduction

Shared-nothing architecture

- It is the most common architecture nowadays.
- In such systems the data are *partitioned* across disk storage units attached directly to each processor allowing to scan/write large amounts of data in parallel.

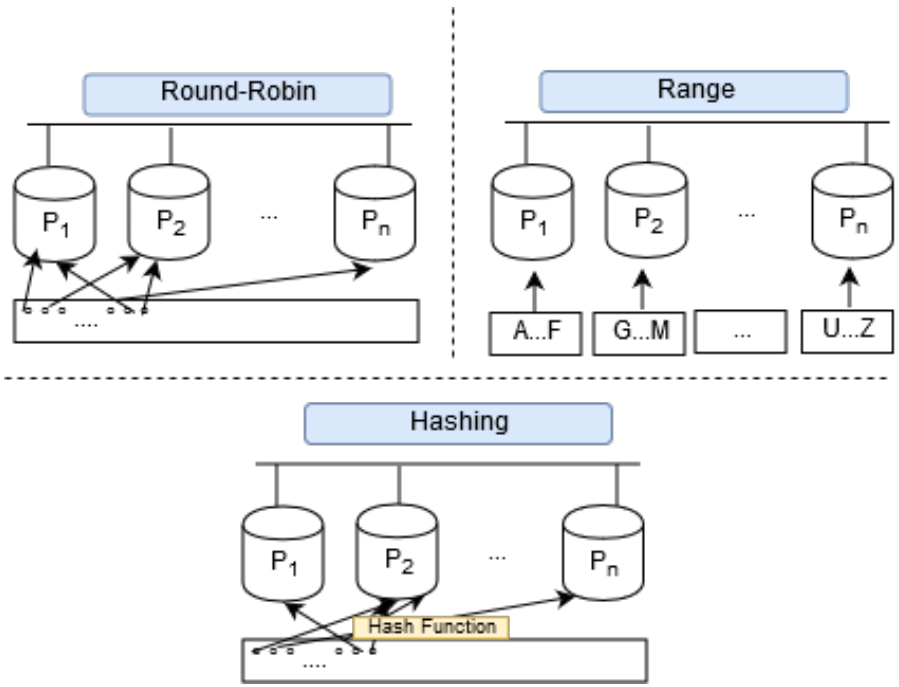
Introduction

Shared-nothing architecture

- It is the most common architecture nowadays.
- In such systems the data are *partitioned* across disk storage units attached directly to each processor allowing to scan/write large amounts of data in parallel.

Data Partitioning

- It has its origins in centralized systems that had to partition large files.
- What distribution unit should be used ? Entire files/relations/graphs ? → **subsets** of them.
- Relational model → distribute tuples.
- RDF model → distribute triples.



Basic partitioning strategies

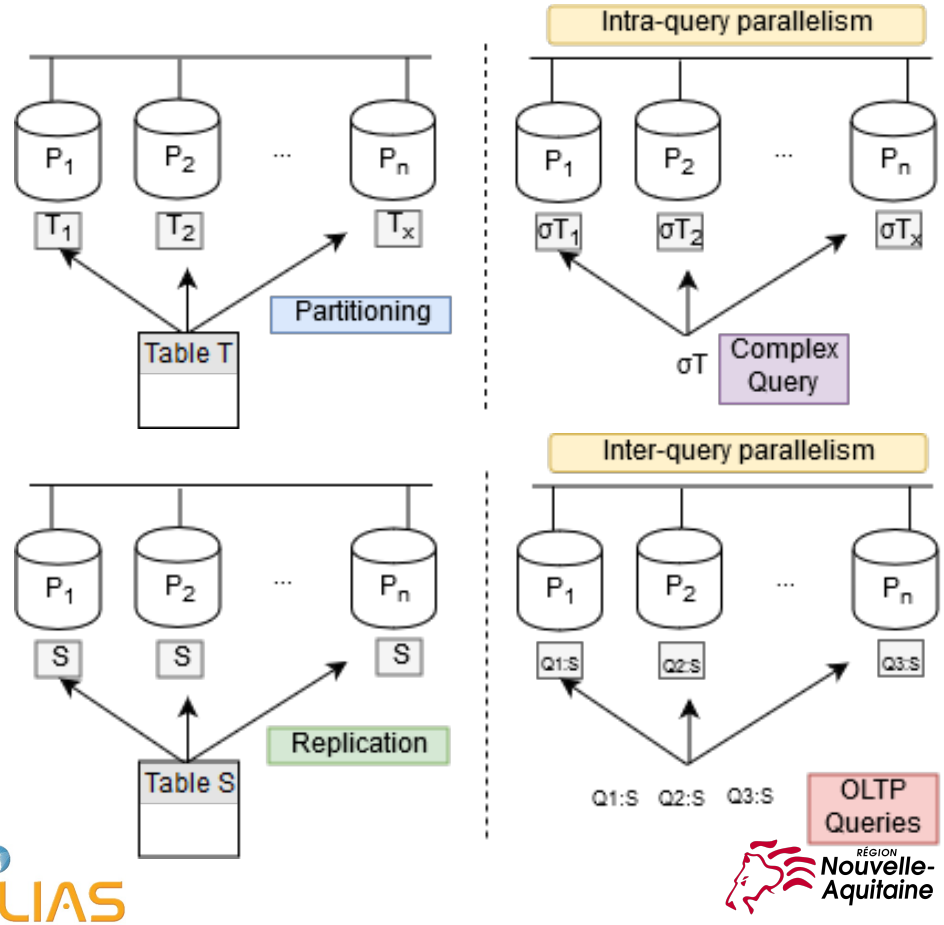
Introduction

Partitioning is a simple concept that is easy to implement, but it raises several new physical database design issues [16] :

Introduction

Partitioning is a simple concept that is easy to implement, but it raises several new physical database design issues [16] :

- Load balancing
  - Data skew
  - Execution skew
- Replication
  - Fault tolerance
  - High availability



Introduction

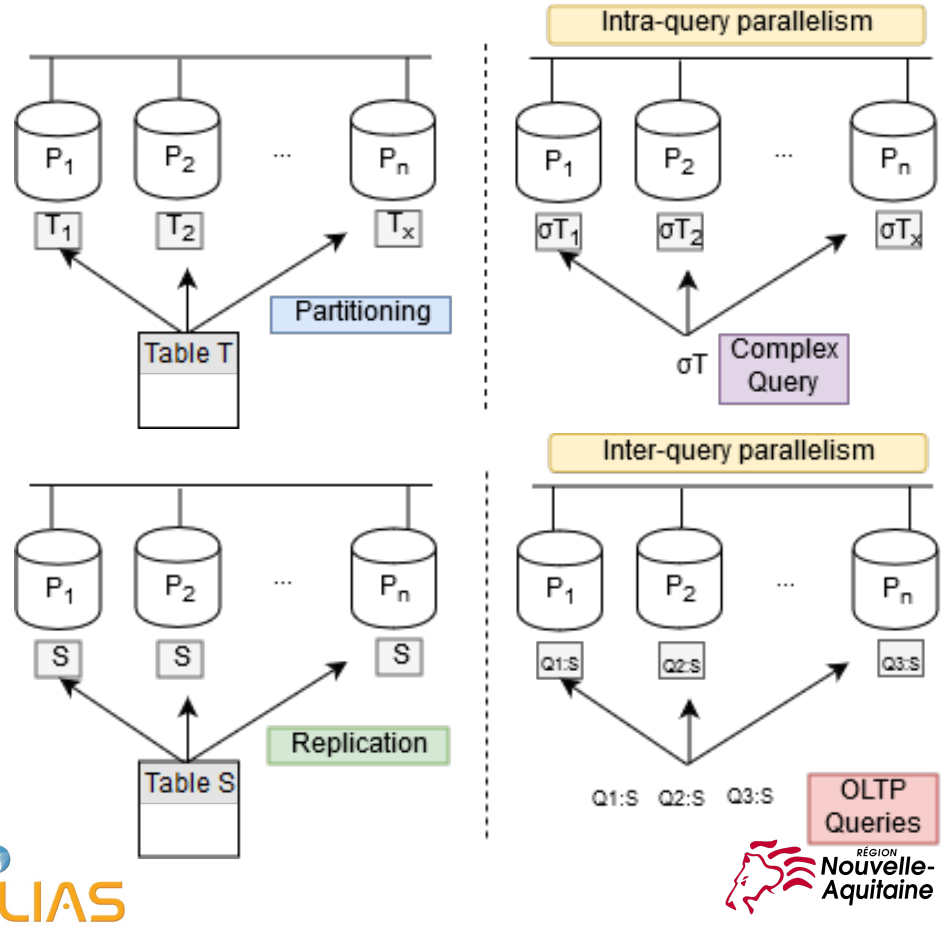
Partitioning is a simple concept that is easy to implement, but it raises several new physical database design issues [16] :

- Load balancing
  - Data skew
  - Execution skew
- Replication
  - Fault tolerance
  - High availability

Trade-offs

Parallelism can **increase throughput** → inter query parallelism, **decrease transaction response time** → intra query parallelism. However :

- In complex queries : ↑ inter-query parallelism → increase total time due to synchronization overhead.
- In a system with high number of concurrent transactions : ↑ intra-query parallelism → decrease in throughput.



Introduction

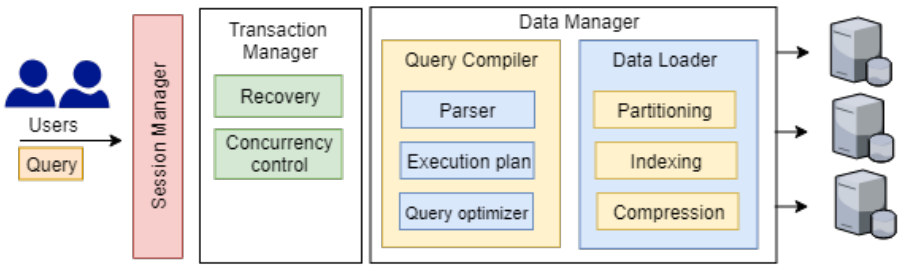
What about the software ?

Introduction

What about the software ?

Parallel database systems in shared-nothing architectures

- Pioneered by Teradata in the late 70s.
- Support standard relational tables and SQL
- The fact that the data is stored on multiple machines is transparent to the end-user.
- Most (or even all) tables are partitioned over the nodes in a cluster.
- The system uses an optimizer that translates SQL commands into a query plan whose execution is divided amongst multiple nodes.

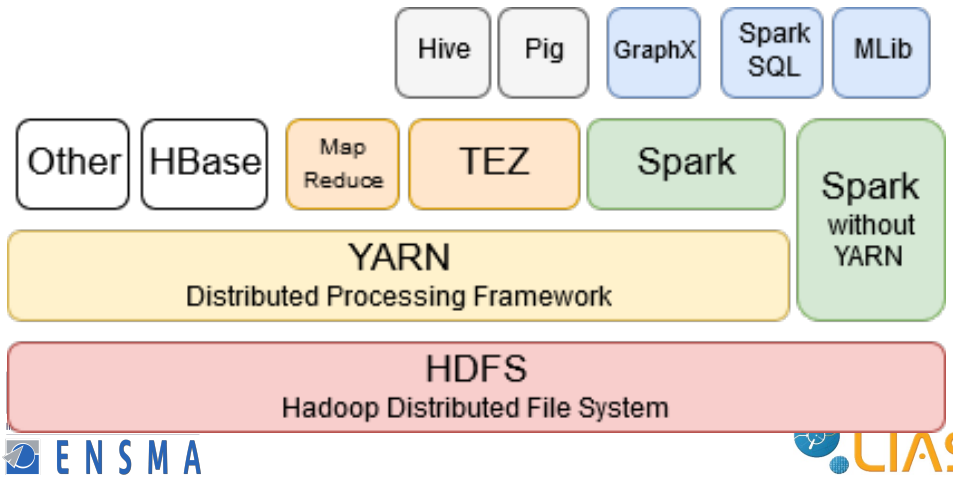


Introduction

Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17],Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects

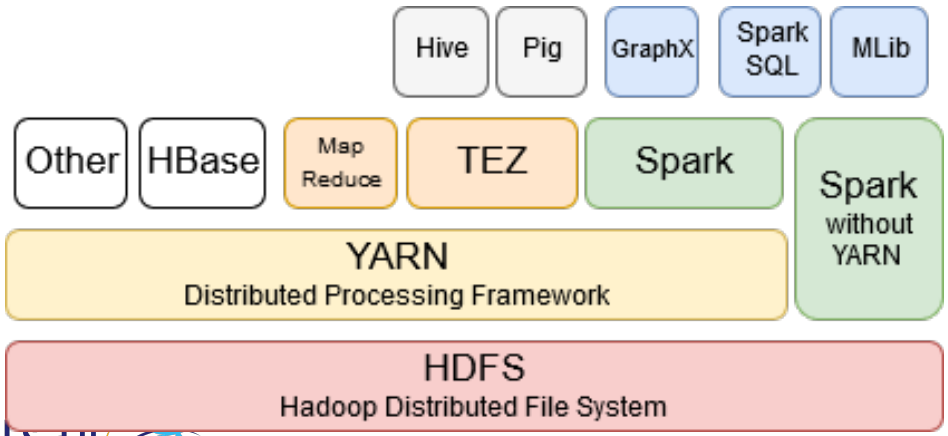


Introduction

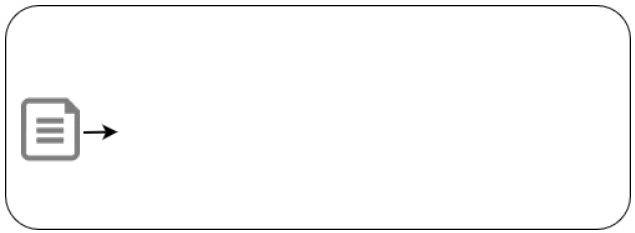
Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17],Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects



Partitioning in HDFS

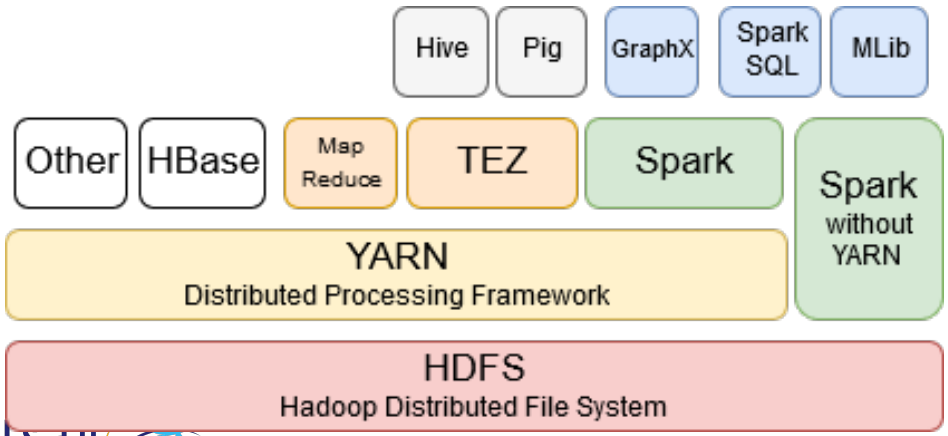


Introduction

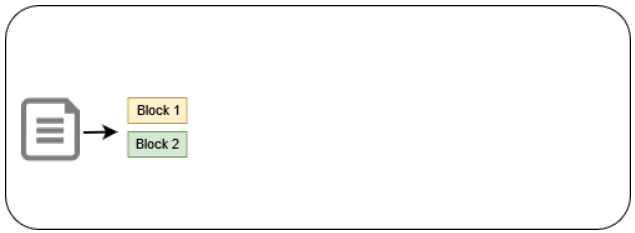
Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17],Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects



Partitioning in HDFS

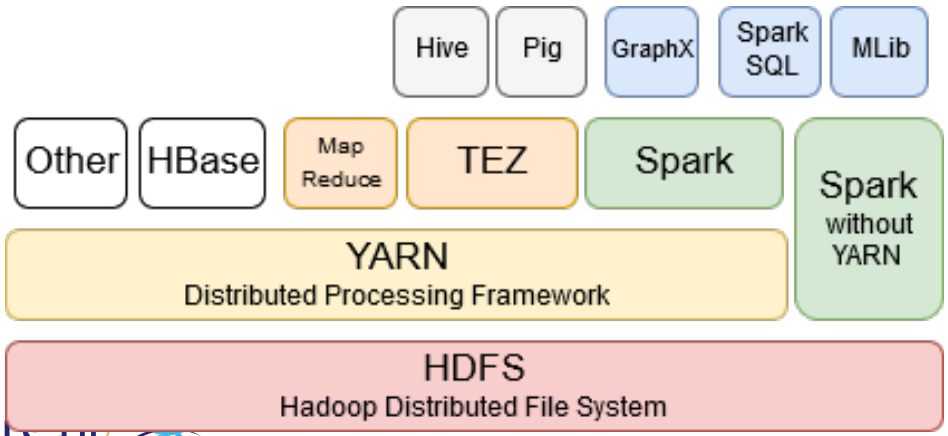


Introduction

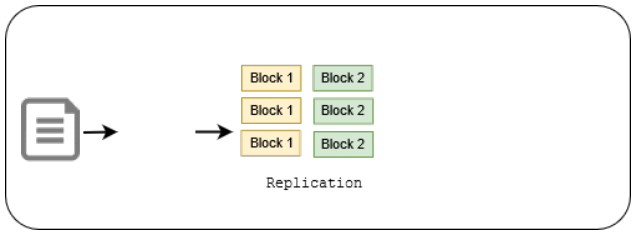
Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17],Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects



Partitioning in HDFS

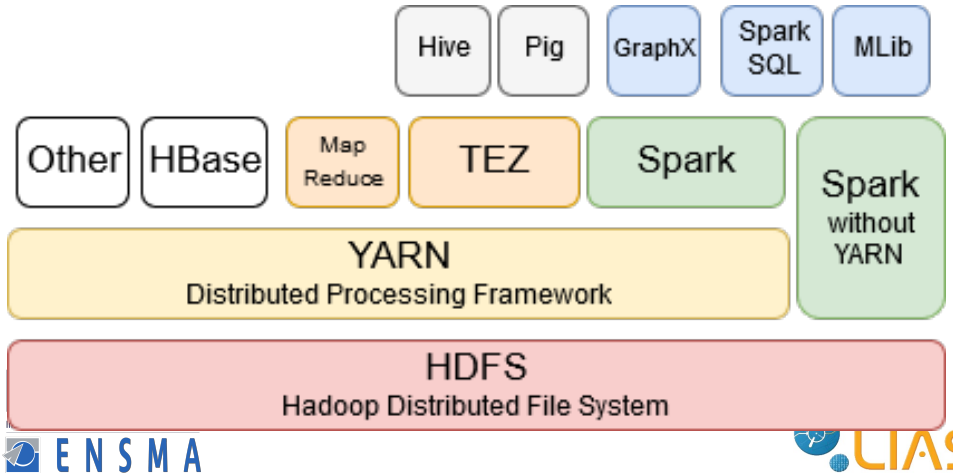


Introduction

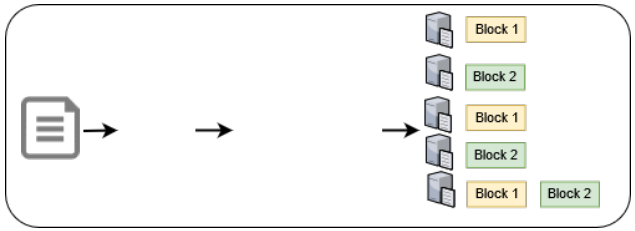
Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17],Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects



Partitioning in HDFS



Introduction

Parallel DBMS	Hadoop and others
Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2)	Hadoop was introduced in 2006
Schema Support : Data should be conform to a well defined relational schema.	**Schema Support** : Absence of a rigid schema defined using the relational model. **Parsers are needed.**

Introduction

Parallel DBMS	Hadoop and others
Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2)	Hadoop was introduced in 2006
Schema Support : Data should be conform to a well defined relational schema.	**Schema Support** : Absence of a rigid schema defined using the relational model. **Parsers are needed.**

Is Hadoop better then ?

Pavlo et al., "A comparison of approaches to large-scale data analysis", SIGMOD, 2009

Introduction

Parallel DBMS	Hadoop and others
Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2)	Hadoop was introduced in 2006
Schema Support : Data should be conform to a well defined relational schema.	**Schema Support** : Absence of a rigid schema defined using the relational model. **Parsers are needed.**

Is Hadoop better then ?

Pavlo et al., "A comparison of approaches to large-scale data analysis", SIGMOD, 2009

- Compared a commercial DBMS and MR across five tasks : Data Loading, Selection, Aggregation, User Defined Aggregation.
- Compared architectural differences :
  - MR : *schema later* paradigm → parsing at runtime.
  - Parallel DBMSs : *strong schema* → parsing at loading.
- The Parallel DBMS was **3.2** times **faster** than MR.
  Why ? Indexes, Storage Mechanisms and **Distribution**, Compression Techniques, Parallel Algorithms.
- Hadoop easier to set up, more flexible and offering a better fault tolerance.

Introduction

Parallel DBMS	Hadoop and others
Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2)	Hadoop was introduced in 2006
Schema Support : Data should be conform to a well defined relational schema.	**Schema Support** : Absence of a rigid schema defined using the relational model. **Parsers are needed.**

Is Hadoop better then ?

Pavlo et al., "A comparison of approaches to large-scale data analysis", SIGMOD, 2009

- Compared a commercial DBMS and MR across five tasks : Data Loading, Selection, Aggregation, User Defined Aggregation.
- Compared architectural differences :
  - MR : *schema later* paradigm → parsing at runtime.
  - Parallel DBMSs : *strong schema* → parsing at loading.
- The Parallel DBMS was **3.2** times **faster** than MR.
  Why ? Indexes, Storage Mechanisms and **Distribution**, Compression Techniques, Parallel Algorithms.
- Hadoop easier to set up, more flexible and offering a better fault tolerance.

We focused our research on Data Partitioning

Related Work

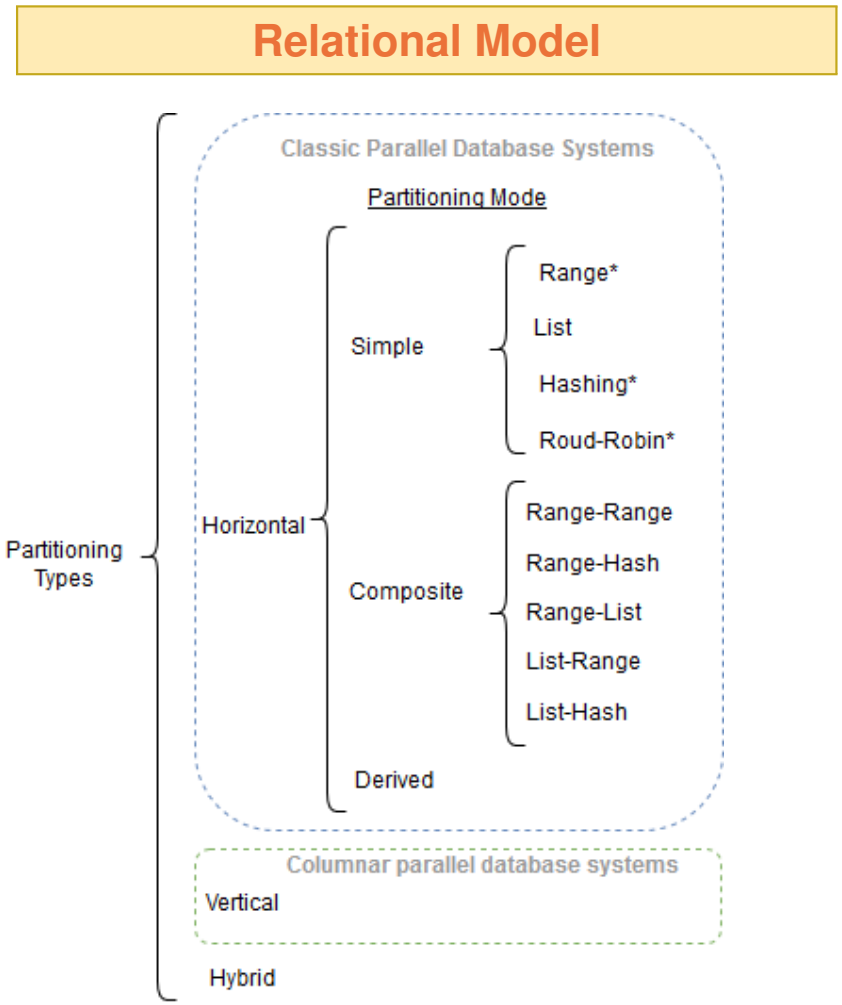
To **enable parallelism** in shared-nothing parallel systems, data is **partitioned** and **distributed** accross the nodes of the system. We explored the partitioning strategies in these systems :

- Parallel Database Systems → *Relational model*
- Distributed Triple Store Systems → *RDF model*
- Hadoop Distributed File System

Related Work

To **enable parallelism** in shared-nothing parallel systems, data is **partitioned** and **distributed** accross the nodes of the system. We explored the partitioning strategies in these systems :

- Parallel Database Systems → *Relational model*
- Distributed Triple Store Systems → *RDF model*
- Hadoop Distributed File System



*basic partitioning strategies

Related Work

Relational Model

Horizontal Fragmentation

Airbus Planes

A_ID	Model	Price M$
1	A320neo	101
2	A330-200	242
3	A350-900	367
4	A380	446

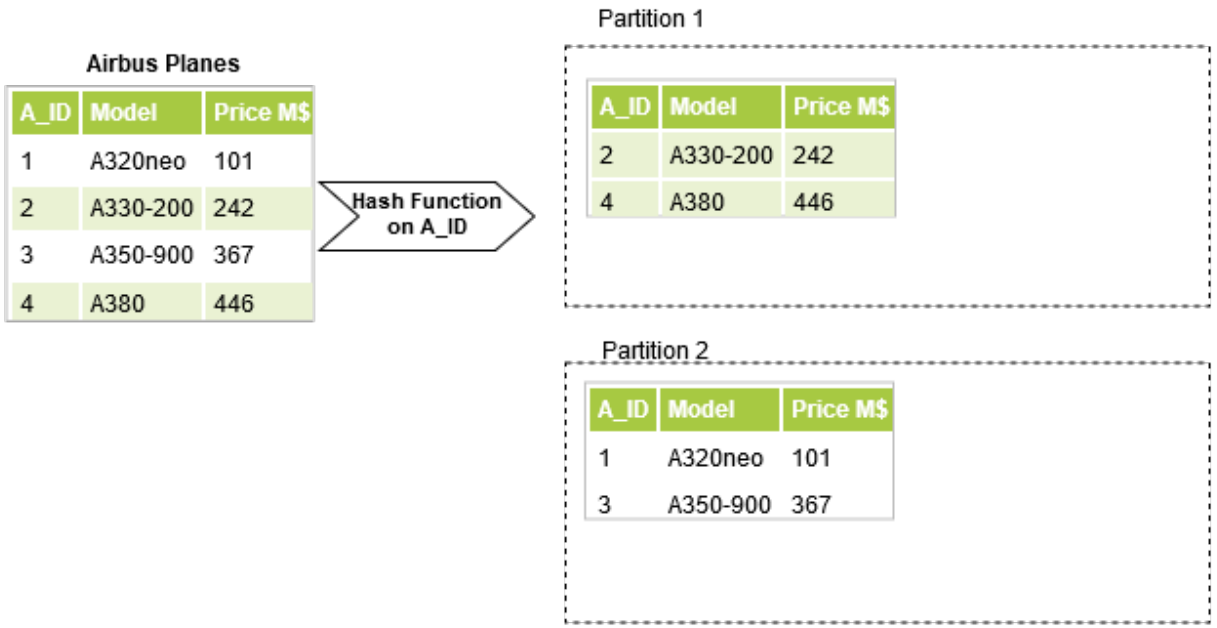
Partition 1

Partition 2

Related Work

Relational Model

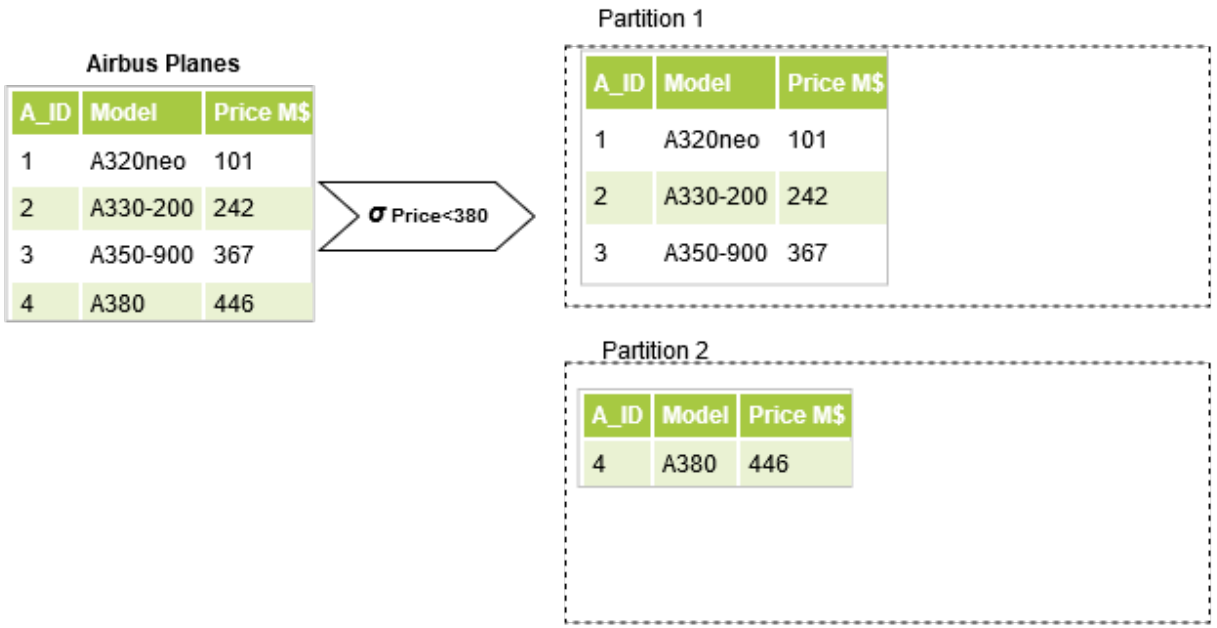
Horizontal Fragmentation : *hashing*



Related Work

Relational Model

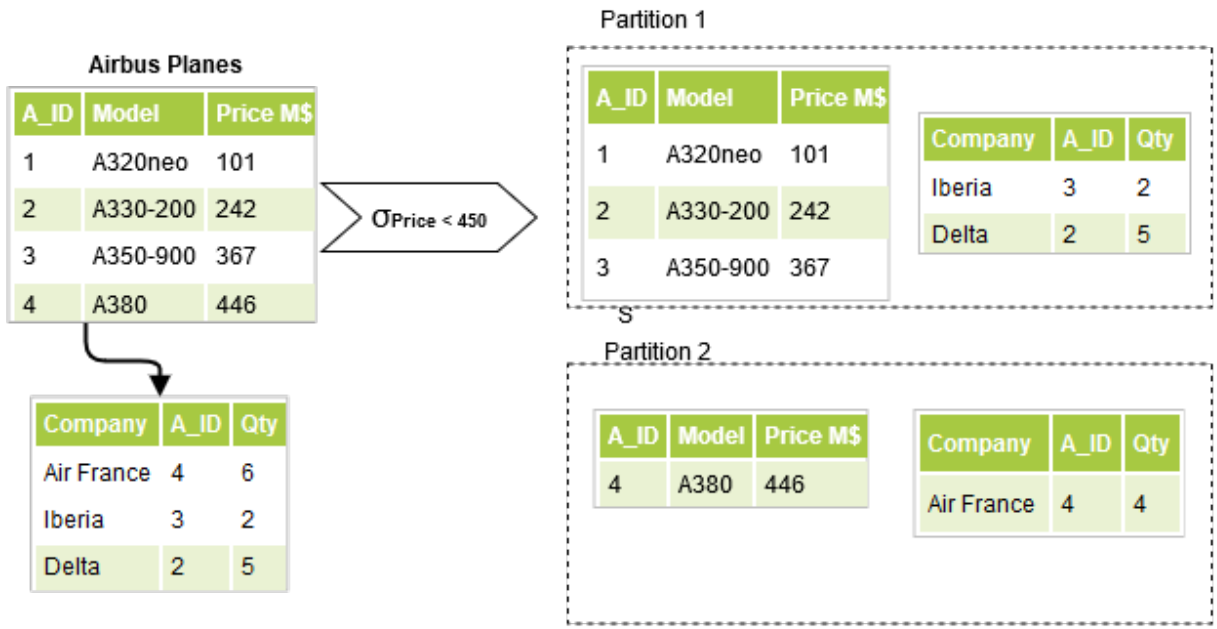
Horizontal Fragmentation : *range*



Related Work

Relational Model

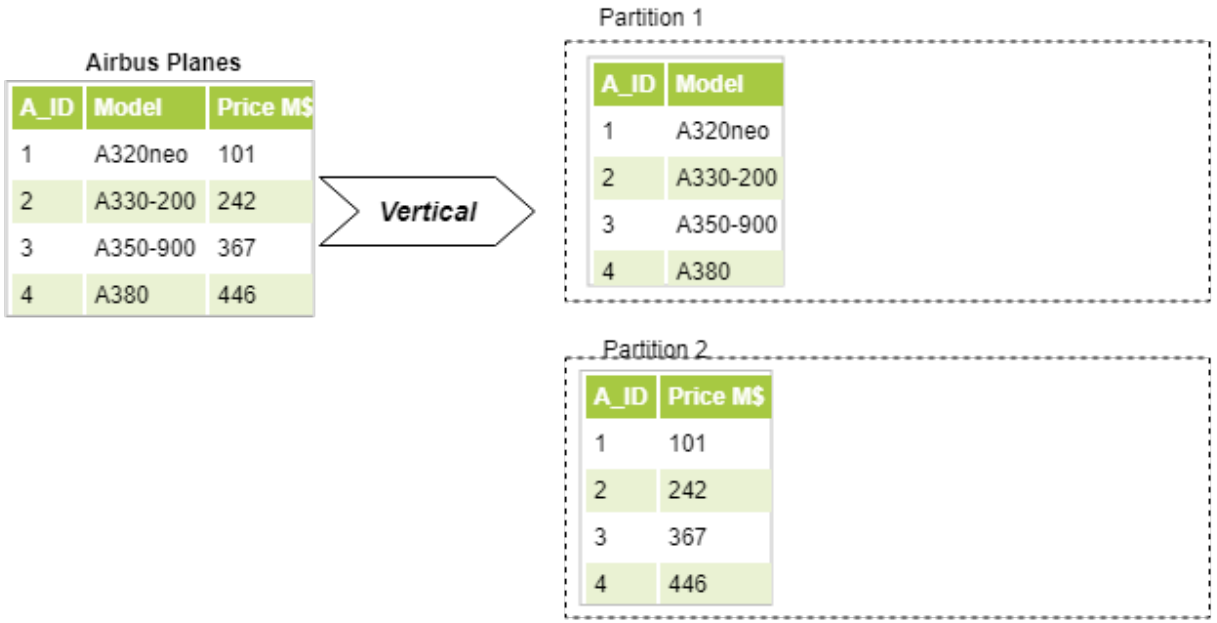
Horizontal Fragmentation : *derived*



Related Work

Relational Model

Vertical Fragmentation



Related Work

	Strategy	Paper	Year	DB Type OLTP	DB Type OLAP	W-A	R	
Initial	Based on tuning advisors using the query optimizer	[35]	2002	✓		✓		IBM - DB2 optimizer
		[4]	2004	✓		✓		Microsoft
	Custom cost function	[32]	2012	✓		✓		Coord+SkF
		[44]	2015	✓			✓	PREF
	Graph-based technique	[14]	2010	✓		✓	✓	**Schism System**
		[11]	2013	✓		✓		HOPE
	Full DB replication	[34]	2002	✓	✓	✓	✓	**Fractured Mirrors**
		[5]	2002	✓	✓	✓	✓	**PAX**
		[27]	2010		✓	✓	✓	Adapt OLAP
		[24]	2011	✓	✓	✓	✓	**PAX+HDFS**
		[13]	2012	✓		✓	✓	Divergent design
	Column Store	[41]	2005	✓	✓	✓		C-Store System
		[8]	2005	✓	✓	✓		MonetDB
Dynamic	Workload	[40]	2016	✓		✓	✓	Clays
	Data	[28]	2012		✓	✓		DynPart

Related Work

RDF Model

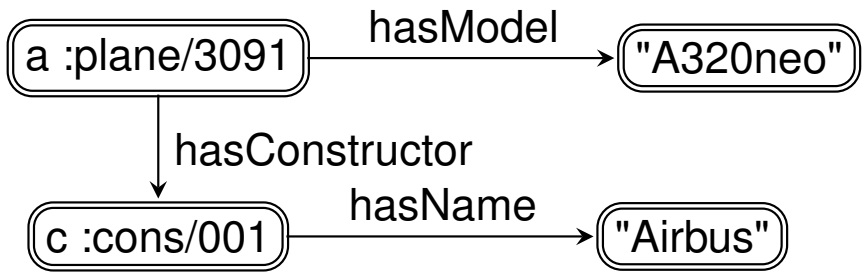
1. https://www.w3.org/TR/rdf11-concepts/

Related Work

RDF Model

Definition

- RDF is a standard model for data interchange in the Web[1].
- It models relational facts as a set of triples <subject, predicate, object>.



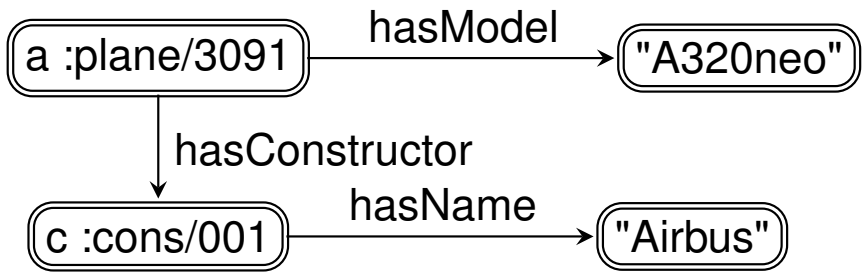
1. https://www.w3.org/TR/rdf11-concepts/

Related Work

RDF Model

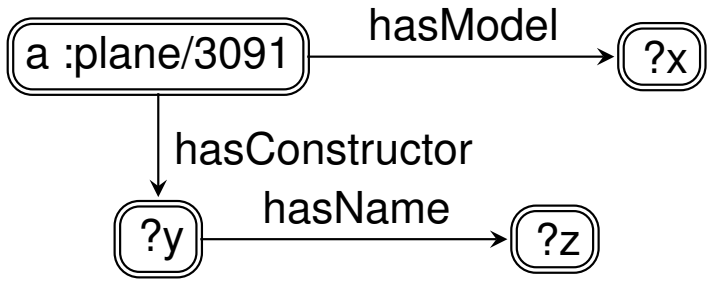
Definition

- RDF is a standard model for data interchange in the Web[1].
- It models relational facts as a set of triples <subject, predicate, object>.



SPARQL

- In a simple definition SPARQL is the query language for RDF data[1].
- The query solution is an ordered set of bindings for $(?x, ?y, ?z)$ → make the SPARQL query graph homomorphic to a subgraph in the data.



1. https://www.w3.org/TR/rdf11-concepts/

Related Work

RDF Processing Approaches

Partitioning is treated individually by each system.

- Centralized Approaches
  1. Relational Mapping
     1. Direct
     2. Property table
     3. Binary table
  2. Graph processing
- Distributed Approaches
  1. On top of distributed systems
  2. Specialized RDF Systems

1. Based on : ÖZSU T., "A Survey of RDF Data Management Systems" [30]

Related Work

Centralized Approaches

Relational Mapping

Direct

- Sesame SQL92SAIL (Broekstra et al, 2002) [10]
- Oracle (Chong et al, VLDB 2005) [12]
- RDF-3X (Neumann T et al, PVLDB 2008) [29]
- Hexastore (Weiss C et al, PVLDB 2008) [42]

s	p	o

Property table

- Jena (Wilkinson K, HP 2006) [43]
- IBM DB2RDF (Bornea M A et al, Sigmod 2013) [9]

s	p_1	p_2 ...
s	o	null

Binary table

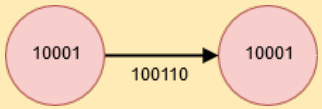
- SW-Store (Abadi et al, VLDB 2009)[2]

predicate

s	o

Graph Processing

- gStore (Özsu et al, PVLDB 2011) [47]
- chameleondb (Gunes A, PhD 2015)[7]



Related Work

Distributed Approaches

↑ **Distributed Systems**

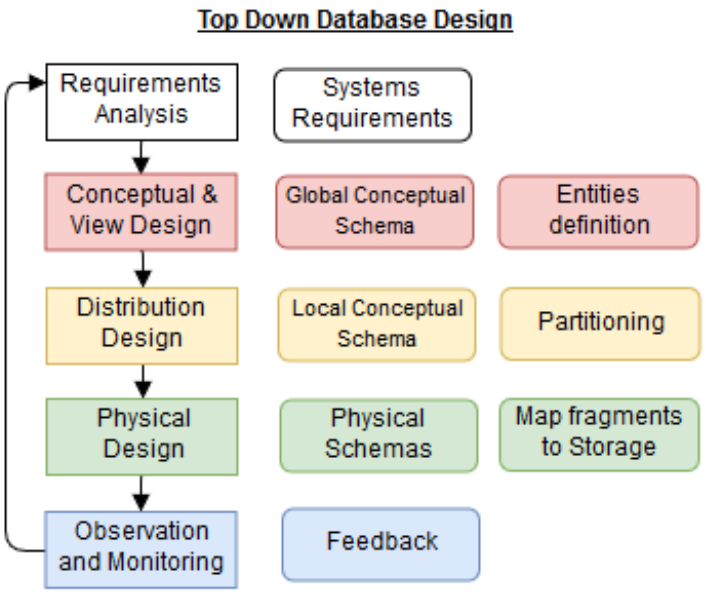
- **MapReduce / HDFS** : SHARD (Direct) [36], HadoopRDF (PropertyTable) [23], CliqueSquare (HashPartition) [18], EAGRE [46], PigSparql [37]
- **Spark** : S2X (GraphX) [39], S2RDF (Core) [38]
- **NoSQL** : H2RDF[31], JenaHBase [25], TrinityRDF [45]

Specialized RDF Systems

- **RDF-3X-Execution** : H-RDF-3X (Metis) [22], SHAPE (Semantic) [26] , WARP (W-A) [21], Dream (Full R)[20]
- **Others** : gStore-D [33], TriAd [19], Ad-Part [6]

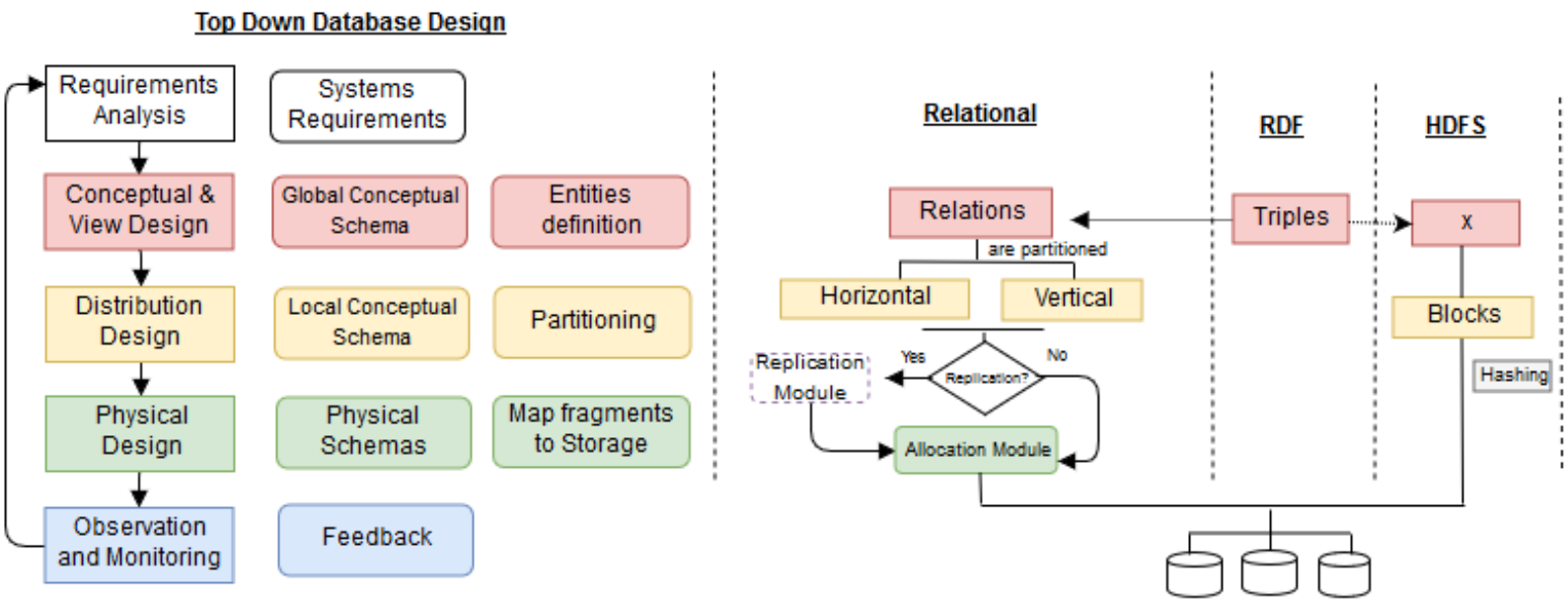
Approach

Top Down Database Design



Approach

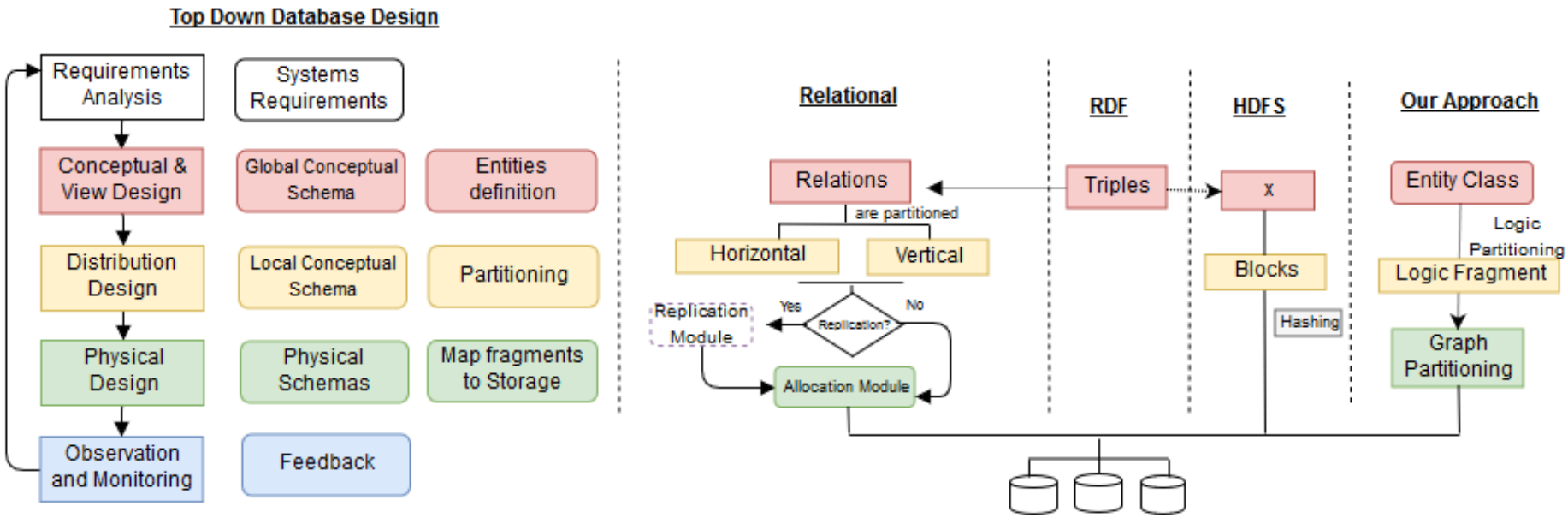
Top Down Database Design



Approach

Top Down Database Design

Our Approach :



Approach

Physical vs Logical Data Model

The data structures used to implement data in the storage module of the system are referred to as **physical data models** (e.g. rows, columns, binary table, property table). Conceptual/**Logical models** are at a somewhat higher level than data structures (e.g. relational, RDF).

Approach

Physical vs Logical Data Model

The data structures used to implement data in the storage module of the system are referred to as **physical data models** (e.g. rows, columns, binary table, property table). Conceptual/**Logical models** are at a somewhat higher level than data structures (e.g. relational, RDF).

- Fragmentation methods for relational and graph (e.g. RDF) models are strongly dependent on the physical data model of the system (e.g. Horizontal fragmentation → physical storage organized as rows, vertical fragmentation → column store).

Approach

Physical vs Logical Data Model

The data structures used to implement data in the storage module of the system are referred to as **physical data models** (e.g. rows, columns, binary table, property table). Conceptual/**Logical models** are at a somewhat higher level than data structures (e.g. relational, RDF).

- Fragmentation methods for relational and graph (e.g. RDF) models are strongly dependent on the physical data model of the system (e.g. Horizontal fragmentation → physical storage organized as rows, vertical fragmentation → column store).

What if we could partition the data at a Logical level **independent** of the **physical storage** module of the system ?

We defined two logical **entities** and fragment the data based on them. The allocation is done based on the connections between these entities.

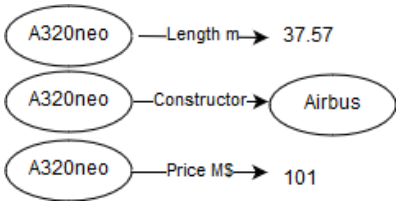
Approach

Logical or Conceptual Model

Relational model : based on tables.

Model	Lenght m	Price M$	Constructor
A320neo	37.57	101	Airbus
A330-200	58.82	242	Airbus
A350-900	66.8	367	Airbus

RDF model : directed graph triple stating that a resource has a property with certain value.



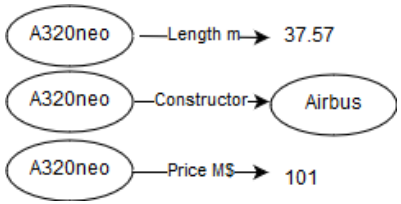
Approach

Logical or Conceptual Model

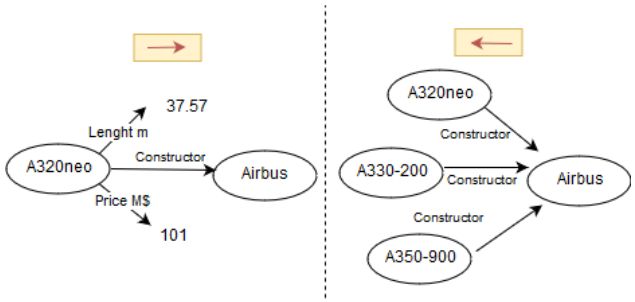
Relational model : based on tables.

Model	Lenght m	Price M$	Constructor
A320neo	37.57	101	Airbus
A330-200	58.82	242	Airbus
A350-900	66.8	367	Airbus

RDF model : directed graph triple stating that a resource has a property with certain value.



Our entity model : based on the model of the system SWORD [46]. **Forward Entity** : Given a subject *s*, the set of all triples having *s* as subject. **Backward Entity** :Given an object *o*, the set of all triples having *o* as object.



Approach

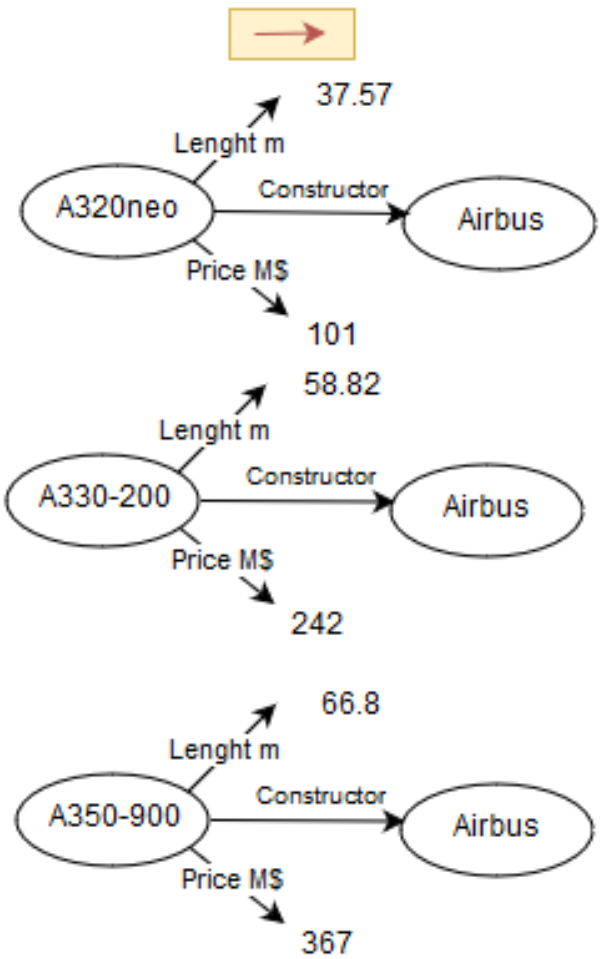
For example,

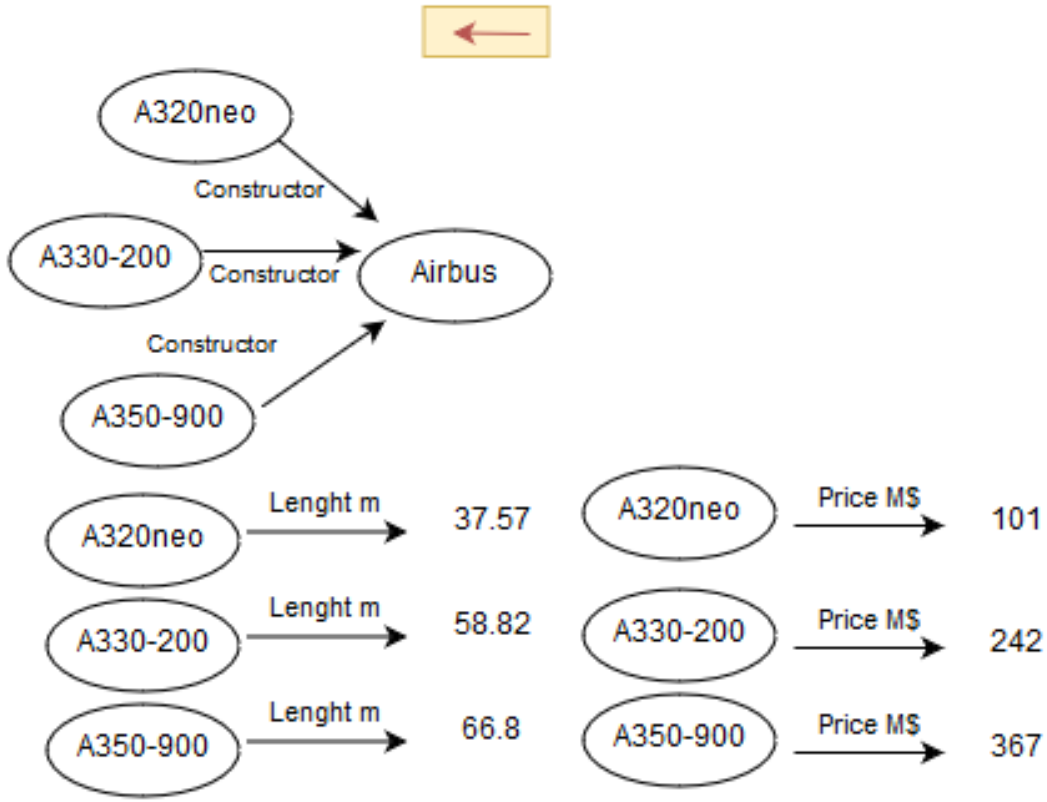
- Let us consider the table :

Model	Lenght m	Price M$	Constructor
A320neo	37.57	101	Airbus
A330-200	58.82	242	Airbus
A350-900	66.8	367	Airbus

Approach

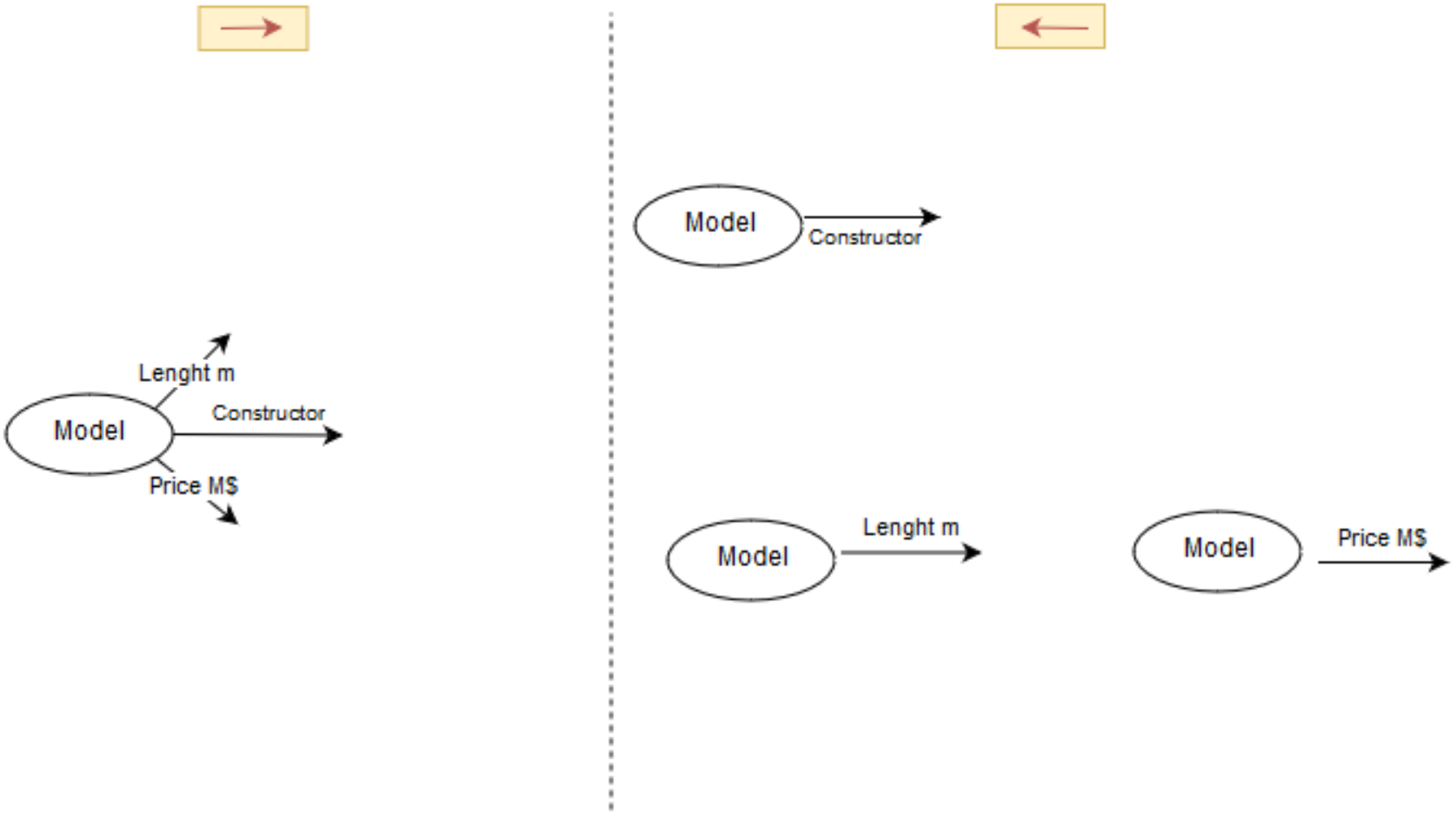
Entity Creation :





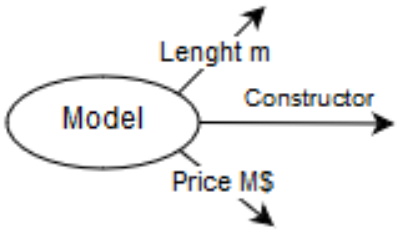
Approach

Fragmentation :

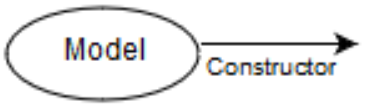


Approach

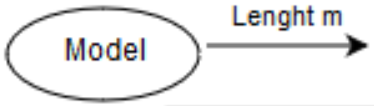
Models mapping :



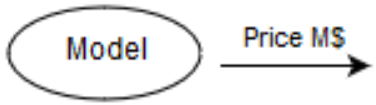
Model	Lenght m	Price M$	Constru
A320neo	37.57	101	Airbus
A330-200	58.82	242	Airbus
A350-900	66.8	367	Airbus



Model	Constructor
A320neo	Airbus
A330-200	Airbus
A350-900	Airbus



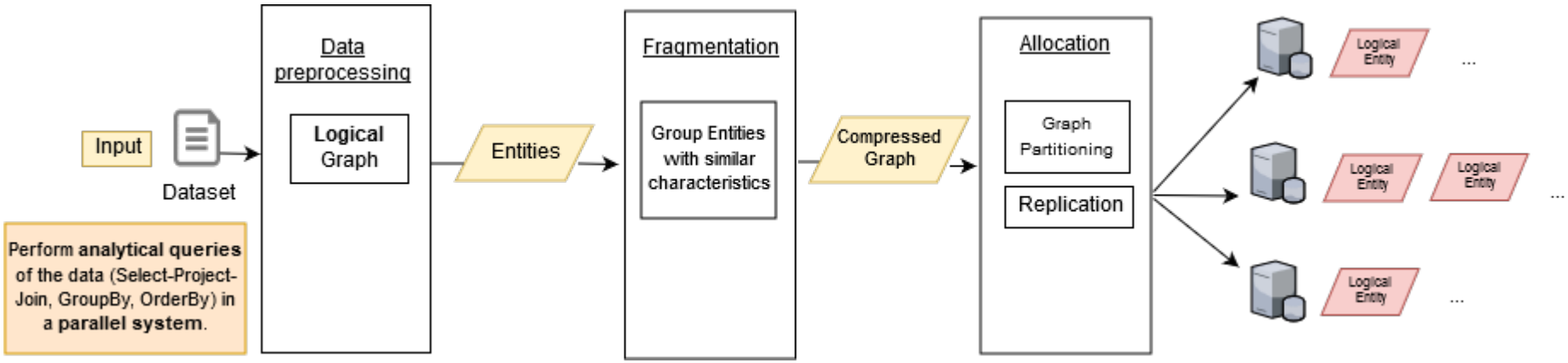
Model	Lenght m
A320neo	37.57
A330-200	58.82
A350-900	66.8



Model	Price M$
A320neo	101
A330-200	242
A350-900	367

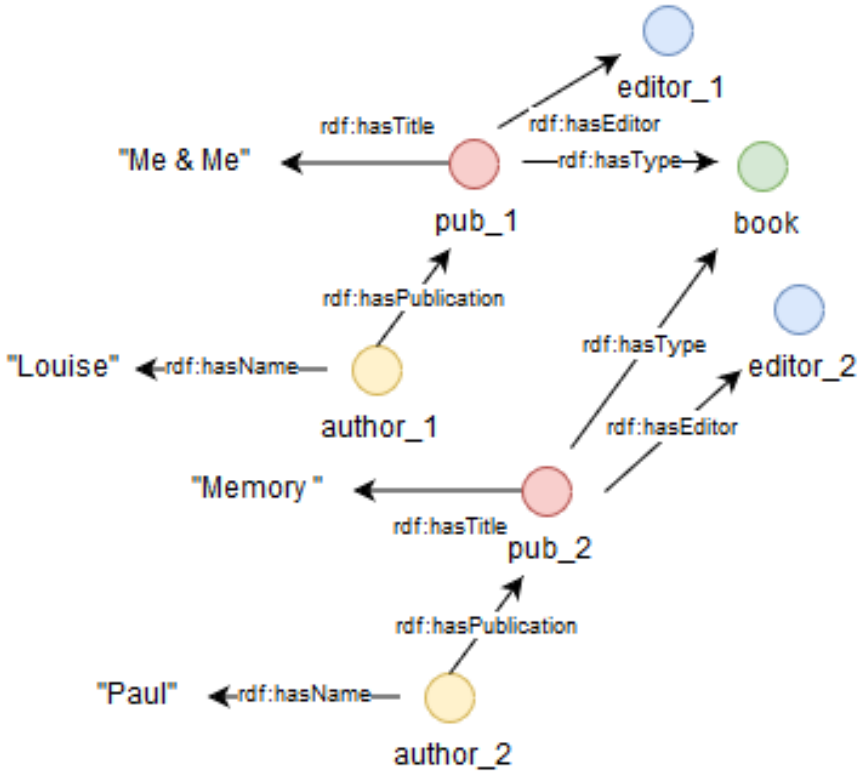
Approach

System overview



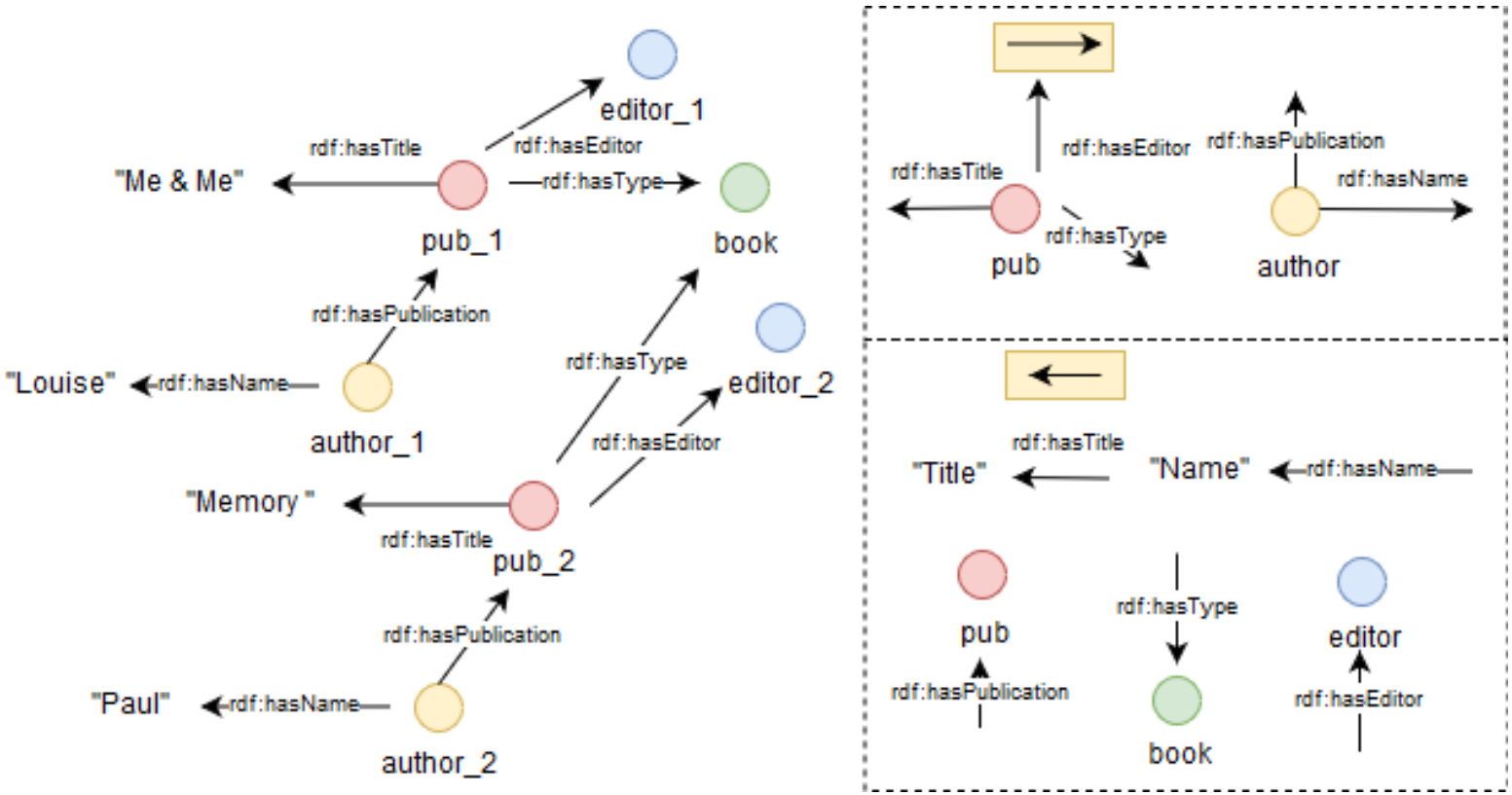
Implementation

Example :



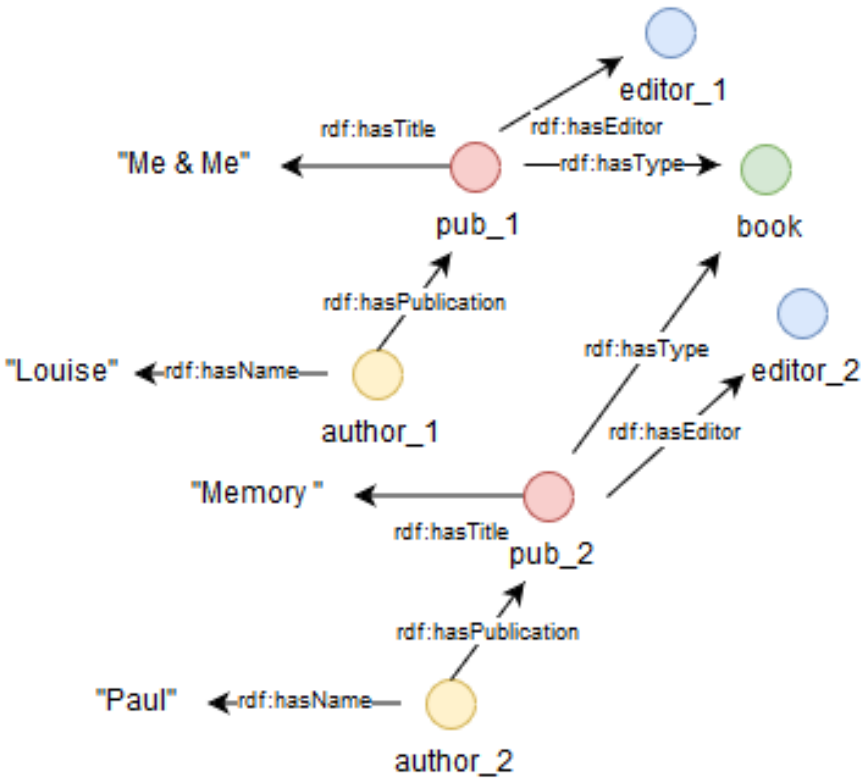
Implementation

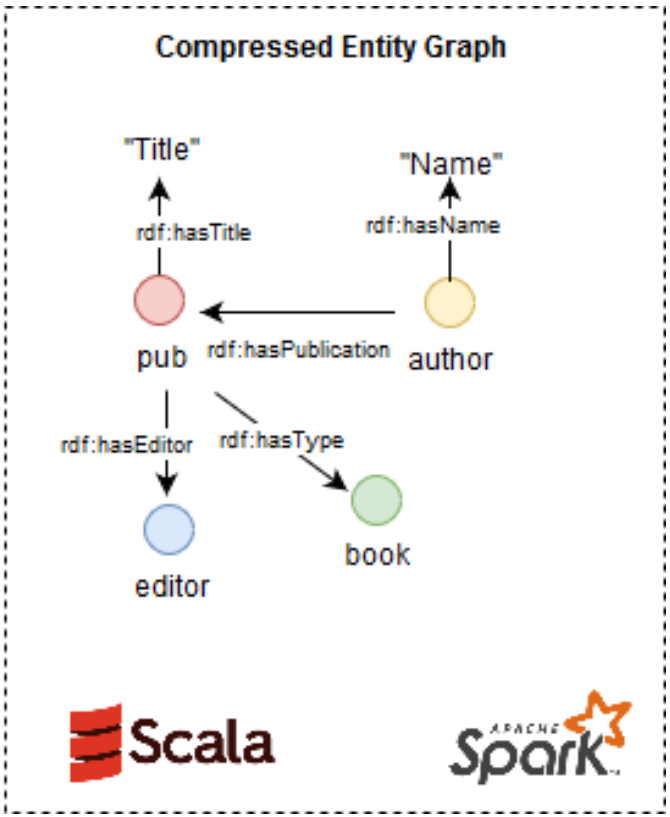
Pre-processing :



Implementation

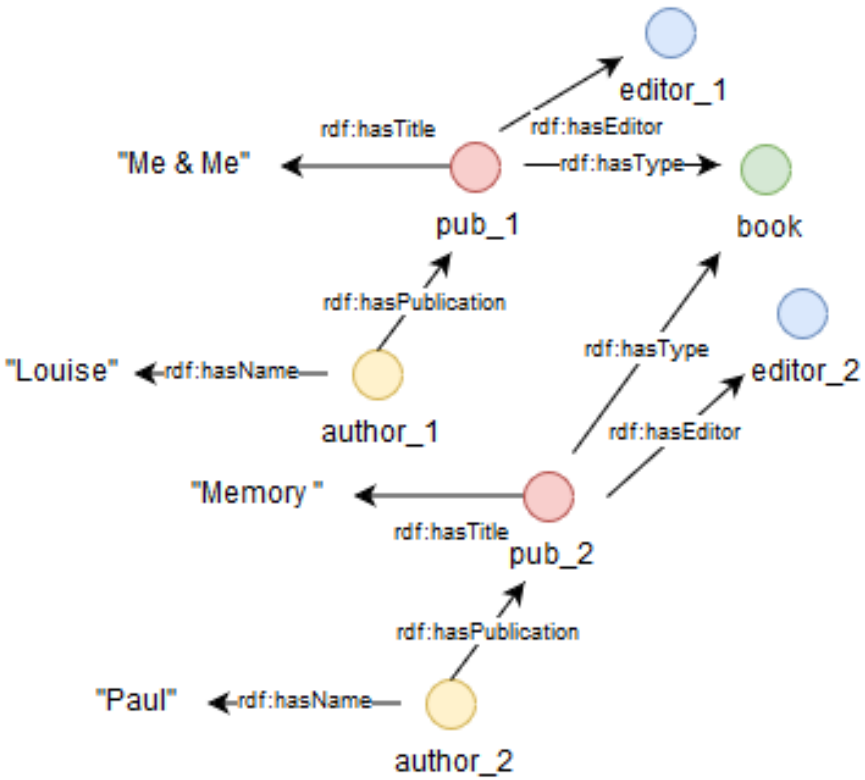
Fragmentation :

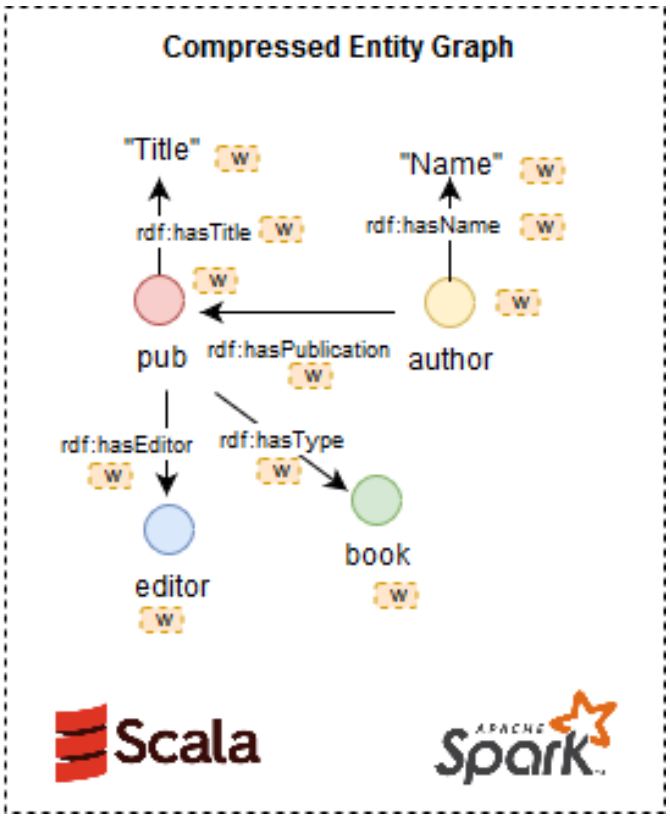




Implementation

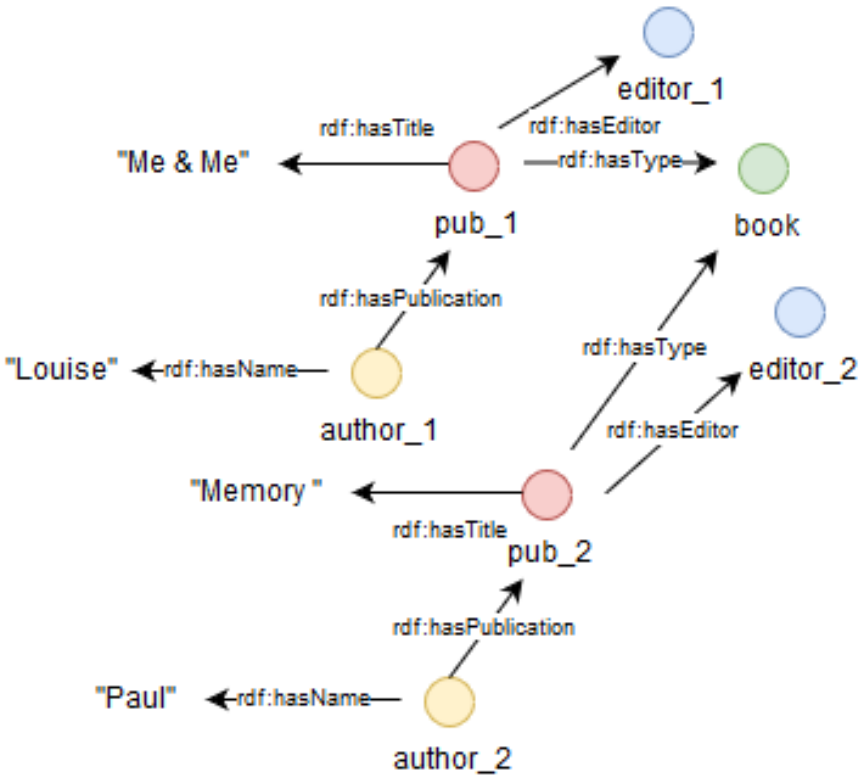
Fragmentation :

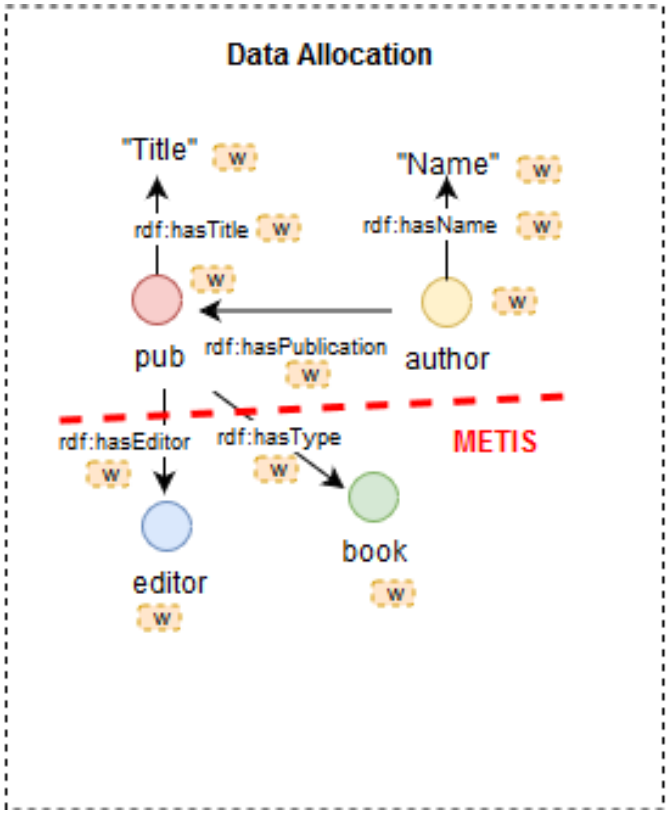




Implementation

Allocation :





Approach

Need for re-allocation

- The preprocessing and fragmentation tasks involve a huge volume of data to process.
- Accordingly, a scalable computing paradigm e.g. MapReduce, SparkRDD is needed to process.
- Therefore, the initial data is first randomly partitioned to the computing nodes (since it is stored to the HDFS). A re-allocation of the data is performed later on.

Experiments

We created a partitioning system :

- *Data pre-processing and Fragmentation* : Scala-Spark
- *Allocation* : Scala-Spark, Metis Latest stable release (5.1.0) :

YARN Cluster infrastructure (1 MN + 4DN)

Component	Description
Processor	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
RAM	8GB
HD	100 GB + 1(400 GB)

RDF datasets

M : millions. # S,# P,# O denote number of distinct subjects, objects and predicates.

Dataset	Triples (M)	# S(M)	# P(M)	# O(M)	Size (GB)
Yago2	284.30	10.12	98	52.34	42
WatDiv	1,092.16	52.12	86	179.09	149

Experiments

Results : Number of entity classes

Dataset	$\#\overrightarrow{\mathcal{E}_C}$	$\#\overleftarrow{\mathcal{E}_C}$
Yago2	25,511	1,359
WatDiv	96,344	4,718

Data skew problem

- We found entity classes that were bigger than the maximum size allowed for a partition.
- We adopted two strategies to re partition :
  1. Hashing on subject.
  2. Based on the class neighbors.

Dataset	#80%	%80%	$\%\overrightarrow{\mathcal{E}_C}$	$\%\overleftarrow{\mathcal{E}_C}$
Yago2	51	0.19	45	35
WatDiv	381	0.38	31	49

Experiments

We have evaluated the performance at the moment based on two metrics :

Execution time

Time taken in minutes for the data loading+pre-processing, fragmentation and allocation tasks.

Dataset	Time (hours)
Yago2	14.2
WatDiv	>20

The loading times reported in [3] for both datasets are much more inferior, but the tests in this paper were perfrmed on a 12 machine cluster, each machine has a 148GB RAM and two 2.1GHz AMD Opteron 6172 CPUs (12 cores each).

Quality of partitioning

We count how many edges were cut between two entity classes that are assigned to two different partitions.

%											
5		10		20		40		60		80	
LP	M	LP	M	LP	M	LP	M	LP	M	LP	M
62	65	48	53	47	60	10	36	14	26	6	16

Future perspectives

- Perform exhaustive experiments in a much more complex cluster of computers.
- We need to test if the partitioning algorithm helps to improve the performance to solve a workload. We have thought of testing in the gStoreD system [33] and the execution engine developped by a LIAS PhD student.
- Test our approach on the relational model mapping relations to triples.
- Explore dynamic partitioning.

References I

McKinsey Global Institute (MGI). “Big Data : The next frontier for innovation, competition, and productivity,” in : (2012).

Daniel J. ABADI et al. “SW-Store : a vertically partitioned DBMS for Semantic Web data management”. In : **VLDB J.** 18.2 (2009), p. 385-406. DOI : 10.1007/s00778-008-0125-y. URL : https://doi.org/10.1007/s00778-008-0125-y.

Ibrahim ABDELAZIZ et al. “A Survey and Experimental Comparison of Distributed SPARQL Engines for Very Large RDF Data”. In : **PVLDB** 10.13 (2017), p. 2049-2060. DOI : 10.14778/3151106.3151109. URL : http://www.vldb.org/pvldb/vol10/p2049-abdelaziz.pdf.

Sanjay AGRAWAL, Vivek R. NARASAYYA et Beverly YANG. “Integrating Vertical and Horizontal Partitioning Into Automated Physical Database Design”. In : **Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004**. 2004, p. 359-370. DOI : 10.1145/1007568.1007609. URL : http://doi.acm.org/10.1145/1007568.1007609.

Anastassia AILAMAKI, David J. DEWITT et Mark D. HILL. “Data page layouts for relational databases on deep memory hierarchies”. In : **VLDB J.** 11.3 (2002), p. 198-215. DOI : 10.1007/s00778-002-0074-9. URL : https://doi.org/10.1007/s00778-002-0074-9.

References II

Razen AL-HARBI et al. “Accelerating SPARQL queries by exploiting hash-based locality and adaptive partitioning”. In : **VLDB J.** 25.3 (2016), p. 355-380. DOI : 10.1007/s00778-016-0420-y. URL : https://doi.org/10.1007/s00778-016-0420-y.

Gunes ALUC. “Workload Matters : A Robust Approach to Physical RDF Database Design”. Thèse de doct. University of Waterloo, Ontario, Canada, 2015. URL : http://hdl.handle.net/10012/9774.

Peter A. BONCZ, Marcin ZUKOWSKI et Niels NES. “MonetDB/X100 : Hyper-Pipelining Query Execution”. In : **CIDR 2005, Second Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings**. 2005, p. 225-237. URL : http://cidrdb.org/cidr2005/papers/P19.pdf.

Mihaela A. BORNEA et al. “Building an efficient RDF store over a relational database”. In : **Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013**. 2013, p. 121-132. DOI : 10.1145/2463676.2463718. URL : http://doi.acm.org/10.1145/2463676.2463718.

References III

Jeen BROEKSTRA, Arjohn KAMPMAN et Frank van HARMELEN. “Sesame : A Generic Architecture for Storing and Querying RDF and RDF Schema”. In : **The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings**. 2002, p. 54-68. DOI : 10.1007/3-540-48005-6_7. URL : https://doi.org/10.1007/3-540-48005-6%5C_7.

Yu CAO et al. “HOPE : Iterative and interactive database partitioning for OLTP workloads”. In : **IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014**. 2014, p. 1274-1277. DOI : 10.1109/ICDE.2014.6816759. URL : https://doi.org/10.1109/ICDE.2014.6816759.

Eugene Inseok CHONG et al. “An Efficient SQL-based RDF Querying Scheme”. In : **Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005**. 2005, p. 1216-1227. URL : http://www.vldb.org/archives/website/2005/program/paper/fri/p1216-chong.pdf.

Mariano P. CONSENS et al. “Divergent physical design tuning for replicated databases”. In : **Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012**. 2012, p. 49-60. DOI : 10.1145/2213836.2213843. URL : http://doi.acm.org/10.1145/2213836.2213843.

References IV

Carlo CURINO et al. “Schism : a Workload-Driven Approach to Database Replication and Partitioning”. In : **PVLDB** 3.1 (2010), p. 48-57. URL : http://www.comp.nus.edu.sg/~vldb2010/proceedings/files/papers/R04.pdf.

Jeffrey DEAN et Sanjay GHEMAWAT. “MapReduce : Simplified Data Processing on Large Clusters”. In : **6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004**. 2004, p. 137-150. URL : http://www.usenix.org/events/osdi04/tech/dean.html.

David J. DEWITT et Jim GRAY. “Parallel Database Systems : The Future of High Performance Database Systems”. In : **Commun. ACM** 35.6 (1992), p. 85-98. DOI : 10.1145/129888.129894. URL : http://doi.acm.org/10.1145/129888.129894.

Sanjay GHEMAWAT, Howard GOBIOFF et Shun-Tak LEUNG. “The Google file system”. In : **Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, SOSP 2003, Bolton Landing, NY, USA, October 19-22, 2003**. 2003, p. 29-43. DOI : 10.1145/945445.945450. URL : http://doi.acm.org/10.1145/945445.945450.

References V

François GOASDOUÉ et al. “CliqueSquare : Flat plans for massively parallel RDF queries”. In : **31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015**. 2015, p. 771-782. DOI : 10.1109/ICDE.2015.7113332. URL : https://doi.org/10.1109/ICDE.2015.7113332.

Sairam GURAJADA et al. “TriAD : a distributed shared-nothing RDF engine based on asynchronous message passing”. In : **International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014**. 2014, p. 289-300. DOI : 10.1145/2588555.2610511. URL : http://doi.acm.org/10.1145/2588555.2610511.

Mohammad HAMMOUD et al. “DREAM : Distributed RDF Engine with Adaptive Query Planner and Minimal Communication”. In : **PVLDB** 8.6 (2015), p. 654-665. DOI : 10.14778/2735703.2735705. URL : http://www.vldb.org/pvldb/vol8/p654-Hammoud.pdf.

Katja HOSE et Ralf SCHENKEL. “WARP : Workload-aware replication and partitioning for RDF”. In : **Workshops Proceedings of the 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013**. 2013, p. 1-6. DOI : 10.1109/ICDEW.2013.6547414. URL : https://doi.org/10.1109/ICDEW.2013.6547414.

References VI

Jiewen HUANG, Daniel J. ABADI et Kun REN. “Scalable SPARQL Querying of Large RDF Graphs”. In : **PVLDB** 4.11 (2011), p. 1123-1134. URL : http://www.vldb.org/pvldb/vol4/p1123-huang.pdf.

Mohammad Farhan HUSAIN et al. “Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing”. In : **IEEE Trans. Knowl. Data Eng.** 23.9 (2011), p. 1312-1327. DOI : 10.1109/TKDE.2011.103. URL : https://doi.org/10.1109/TKDE.2011.103.

Alekh JINDAL et Jens DITTRICH. “Relax and Let the Database Do the Partitioning Online”. In : **Enabling Real-Time Business Intelligence - 5th International Workshop, BIRTE 2011, Held at the 37th International Conference on Very Large Databases, VLDB 2011, Seattle, WA, USA, September 2, 2011, Revised Selected Papers**. 2011, p. 65-80. DOI : 10.1007/978-3-642-33500-6_5. URL : https://doi.org/10.1007/978-3-642-33500-6_5.

Vaibhav KHADILKAR et al. “Jena-HBase : A Distributed, Scalable and Effcient RDF Triple Store”. In : **Proceedings of the ISWC 2012 Posters & Demonstrations Track, Boston, USA, November 11-15, 2012**. 2012. URL : http://ceur-ws.org/Vol-914/paper%5C_14.pdf.

References VII

Kisung LEE et Ling LIU. “Scaling Queries over Big RDF Graphs with Semantic Hash Partitioning”. In : **PVLDB** 6.14 (2013), p. 1894-1905. DOI : 10.14778/2556549.2556571. URL : http://www.vldb.org/pvldb/vol6/p1894-lee.pdf.

Alexandre A. B. LIMA, Marta MATTOSO et Patrick VALDURIEZ. “Adaptive Virtual Partitioning for OLAP Query Processing in a Database Cluster”. In : **JIDM** 1.1 (2010), p. 75-88. URL : http://seer.lcc.ufmg.br/index.php/jidm/article/view/25.

Miguel LIROZ-GISTAU et al. “Dynamic Workload-Based Partitioning for Large-Scale Databases”. In : **Database and Expert Systems Applications - 23rd International Conference, DEXA 2012, Vienna, Austria, September 3-6, 2012. Proceedings, Part II**. 2012, p. 183-190. DOI : 10.1007/978-3-642-32597-7_16. URL : https://doi.org/10.1007/978-3-642-32597-7_16.

Thomas NEUMANN et Gerhard WEIKUM. “RDF-3X : a RISC-style engine for RDF”. In : **PVLDB** 1.1 (2008), p. 647-659. DOI : 10.14778/1453856.1453927. URL : http://www.vldb.org/pvldb/1/1453927.pdf.

M. Tamer ÖZSU. “A survey of RDF data management systems”. In : **Frontiers Comput. Sci.** 10.3 (2016), p. 418-432. DOI : 10.1007/s11704-016-5554-y. URL : https://doi.org/10.1007/s11704-016-5554-y.

References VIII

Nikolaos PAPAILIOU et al. “H2RDF+ : High-performance distributed joins over large-scale RDF graphs”. In : **Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA**. 2013, p. 255-263. DOI : 10.1109/BigData.2013.6691582. URL : https://doi.org/10.1109/BigData.2013.6691582.

Andrew PAVLO, Carlo CURINO et Stanley B. ZDONIK. “Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems”. In : **Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012**. 2012, p. 61-72. DOI : 10.1145/2213836.2213844. URL : http://doi.acm.org/10.1145/2213836.2213844.

Peng PENG et al. “Processing SPARQL queries over distributed RDF graphs”. In : **VLDB J.** 25.2 (2016), p. 243-268. DOI : 10.1007/s00778-015-0415-0. URL : https://doi.org/10.1007/s00778-015-0415-0.

Ravishankar RAMAMURTHY, David J. DEWITT et Qi SU. “A Case for Fractured Mirrors”. In : **VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China**. 2002, p. 430-441. URL : http://www.vldb.org/conf/2002/S12P03.pdf.

References IX

Jun RAO et al. “Automating physical database design in a parallel database”. In : **Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002**. 2002, p. 558-569. DOI : 10.1145/564691.564757. URL : http://doi.acm.org/10.1145/564691.564757.

Kurt ROHLOFF et Richard E. SCHANTZ. “Clause-iteration with MapReduce to scalably query datagraphs in the SHARD graph-store”. In : **DIDC’11, Proceedings of the Fourth International Workshop on Data-intensive Distributed Computing, San Jose, CA, USA, June 8, 2011**. 2011, p. 35-44. DOI : 10.1145/1996014.1996021. URL : http://doi.acm.org/10.1145/1996014.1996021.

Alexander SCHÄTZLE, Martin PRZYJACIEL-ZABLOCKI et Georg LAUSEN. “PigSPARQL : mapping SPARQL to Pig Latin”. In : **Proceedings of the International Workshop on Semantic Web Information Management, SWIM 2011, Athens, Greece, June 12, 2011**. 2011, p. 4. DOI : 10.1145/1999299.1999303. URL : http://doi.acm.org/10.1145/1999299.1999303.

Alexander SCHÄTZLE et al. “S2RDF : RDF Querying with SPARQL on Spark”. In : **PVLDB** 9.10 (2016), p. 804-815. DOI : 10.14778/2977797.2977806. URL : http://www.vldb.org/pvldb/vol9/p804-schaetzle.pdf.

References X

Alexander SCHÄTZLE et al. “S2X : Graph-Parallel Querying of RDF with GraphX”. In : **Biomedical Data Management and Graph Online Querying - VLDB 2015 Workshops, Big-O(Q) and DMAH, Waikoloa, HI, USA, August 31 - September 4, 2015, Revised Selected Papers**. 2015, p. 155-168. DOI : 10.1007/978-3-319-41576-5_12. URL : https://doi.org/10.1007/978-3-319-41576-5%5C_12.

Marco SERAFINI et al. “Clay : Fine-Grained Adaptive Partitioning for General Database Schemas”. In : **PVLDB** 10.4 (2016), p. 445-456. URL : http://www.vldb.org/pvldb/vol10/p445-serafini.pdf.

Michael STONEBRAKER et al. “C-Store : A Column-oriented DBMS”. In : **Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005**. 2005, p. 553-564. URL : http://www.vldb.org/archives/website/2005/program/paper/thu/p553-stonebraker.pdf.

Cathrin WEISS, Panagiotis KARRAS et Abraham BERNSTEIN. “Hexastore : sextuple indexing for semantic web data management”. In : **PVLDB** 1.1 (2008), p. 1008-1019. DOI : 10.14778/1453856.1453965. URL : http://www.vldb.org/pvldb/1/1453965.pdf.

Kevin WILKINSON. **Jena property table implementation**. 2006.

References XI

Erfan ZAMANIAN, Carsten BINNIG et Abdallah SALAMA. “Locality-aware Partitioning in Parallel Database Systems”. In : **Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015**. 2015, p. 17-30. DOI : 10.1145/2723372.2723718. URL : http://doi.acm.org/10.1145/2723372.2723718.

Kai ZENG et al. “A Distributed Graph Engine for Web Scale RDF Data”. In : **PVLDB** 6.4 (2013), p. 265-276. DOI : 10.14778/2535570.2488333. URL : http://www.vldb.org/pvldb/vol6/p265-zeng.pdf.

Xiaofei ZHANG et al. “EAGRE : Towards scalable I/O efficient SPARQL query evaluation on the cloud”. In : **29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013**. 2013, p. 565-576. DOI : 10.1109/ICDE.2013.6544856. URL : https://doi.org/10.1109/ICDE.2013.6544856.

Lei ZOU et al. “gStore : Answering SPARQL Queries via Subgraph Matching”. In : **PVLDB** 4.8 (2011), p. 482-493. DOI : 10.14778/2002974.2002976. URL : http://www.vldb.org/pvldb/vol4/p482-zou.pdf.