从零开始Cassandra

应用，原理，选型

李力

2022年11月23日

Code： https://github.com/xiaozhiliaoo/cassandra-practice

Slide： https://github.com/xiaozhiliaoo/my-slides/blob/master/Database-Cassandra.pptx

1 **简介**

Cassandra是什么？

2 **数据建模与开发**

项目使用指南

3 **系统架构**

Cassandra单机，分布式原理

4 **NoSQL存储选型**

我们应该关注什么？

5 **参考资料**

基础与进阶

6 **回顾&QA**

思维导图

1. 简介

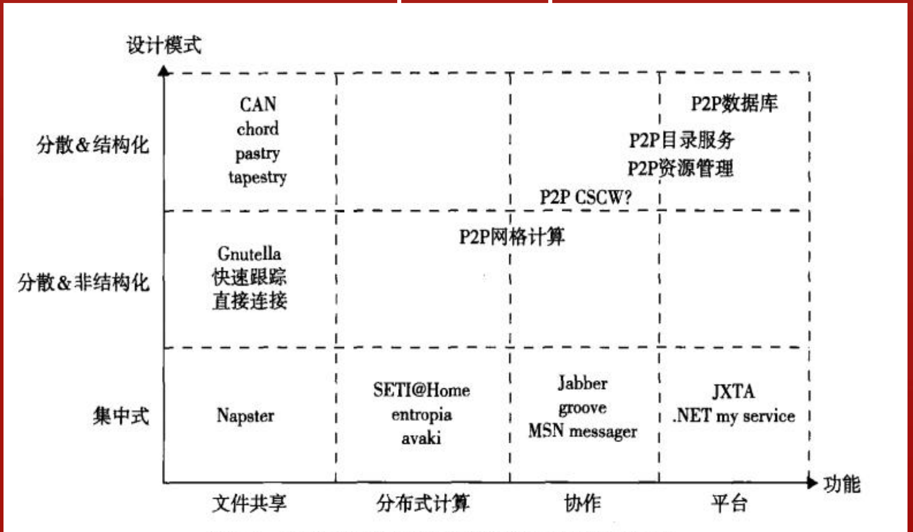
The content of the title

- Apache Cassandra is an open source, distributed,**shared-nothing**, **decentralized（P2P对等结构化覆盖网络）**, elastically scalable,highly available, fault-tolerant, **tuneably consistent（C“A”P，PACELC）**, **row-oriented（宽列）** database that bases its distribution design on Amazon’s **Dynamo** and its data model on Google’s **Bigtable**. Created at Facebook, it is now used at some of the most popular sites on the Web.

- Apache Cassandra is an open source NoSQL **distributed database** trusted by thousands of companies for scalability and high availability without compromising performance. **Linear scalability** and proven fault-tolerance on **commodity hardware** or cloud infrastructure make it the perfect platform for **mission-critical** data.

- Cassandra=Bigtable(数据模型)+Dynamo(分布式特性)

- Cassandra World Party

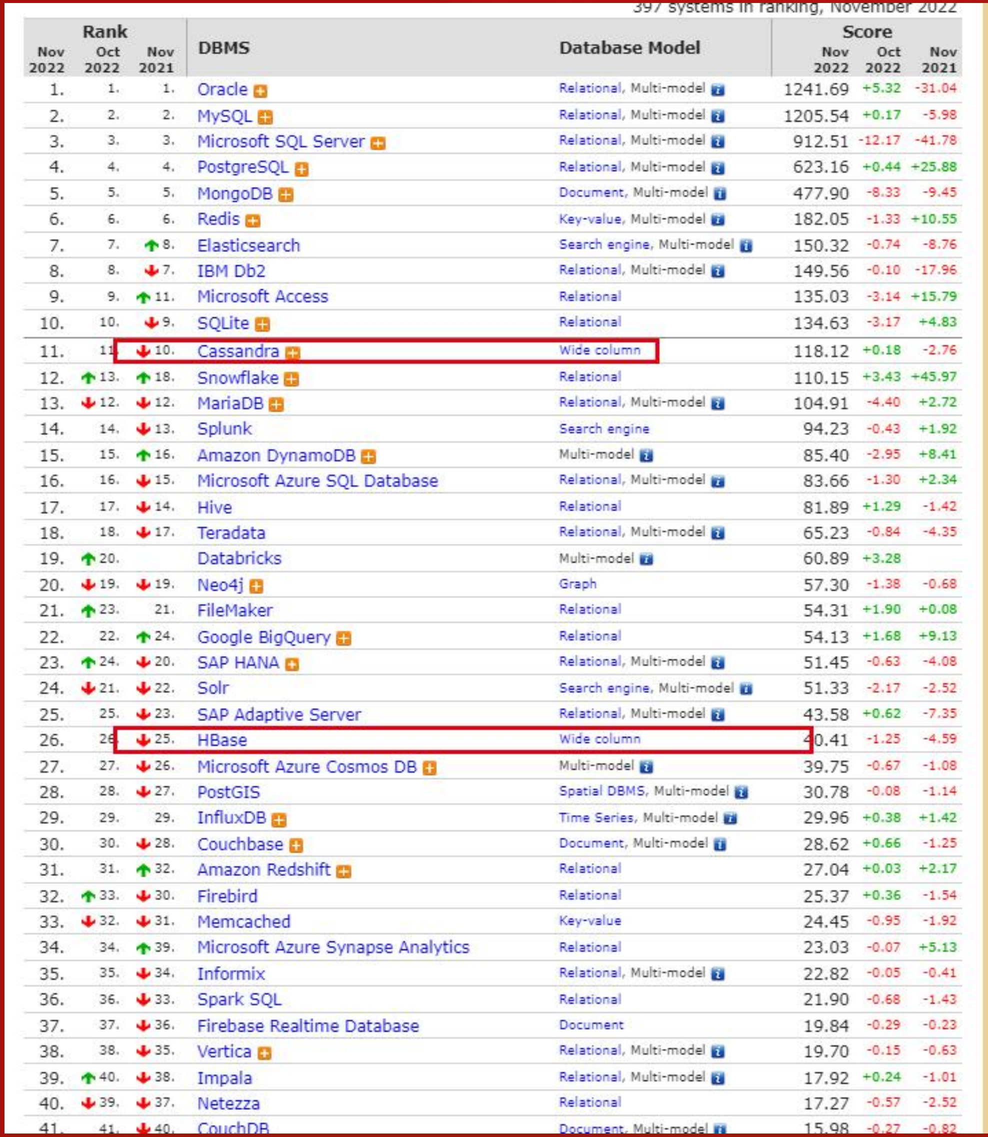


	文件共享	分布式计算	协作	平台
分散&结构化	CAN chord pastry tapestry		P2P CSCW?	P2P数据库 P2P目录服务 P2P资源管理
分散&非结构化	Gnutella 快速跟踪 直接连接	P2P网格计算		
集中式	Napster	SETI@Home entropia avaki	Jabber groove MSN messager	JXTA .NET my service

图 8-9　按照功能和设计模式对 P2P 系统进行分类

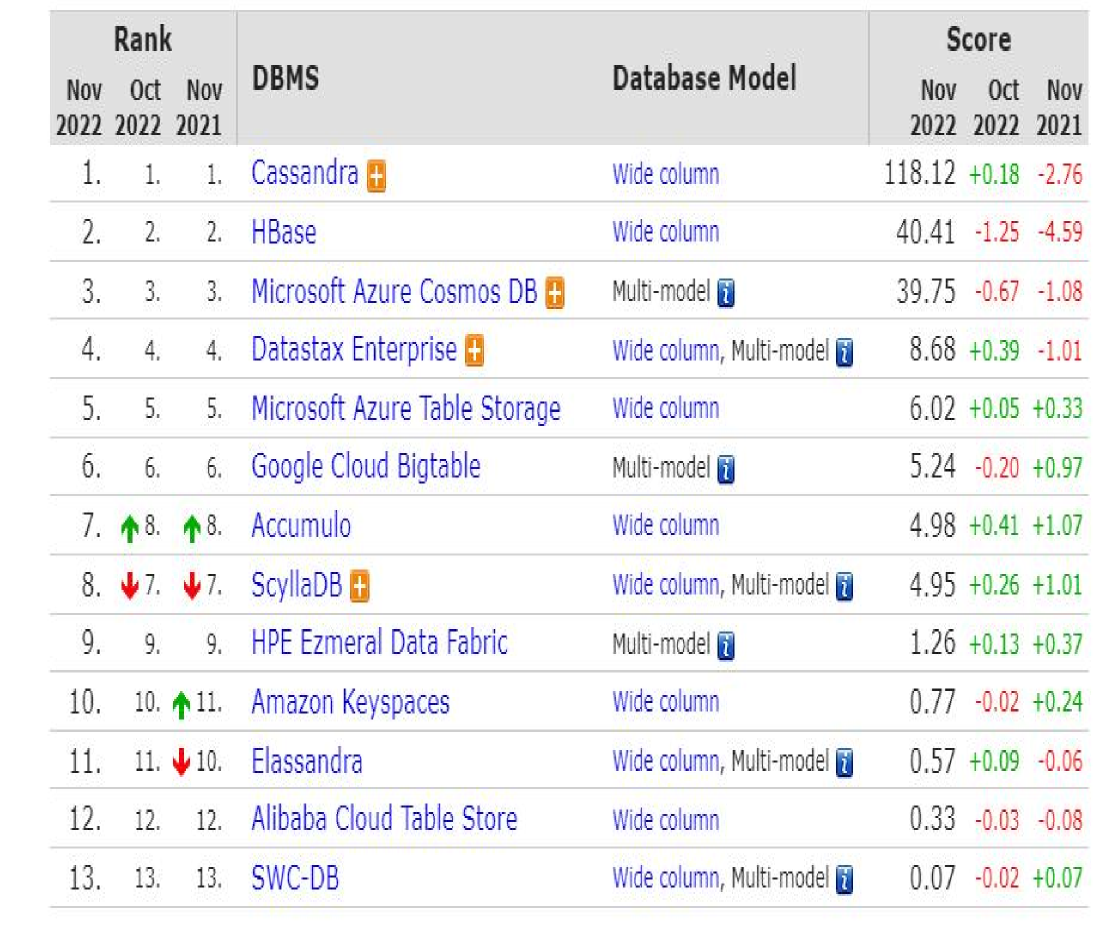
1. 简介-db-engines排名

The content of the title



397 systems in ranking, November 2022

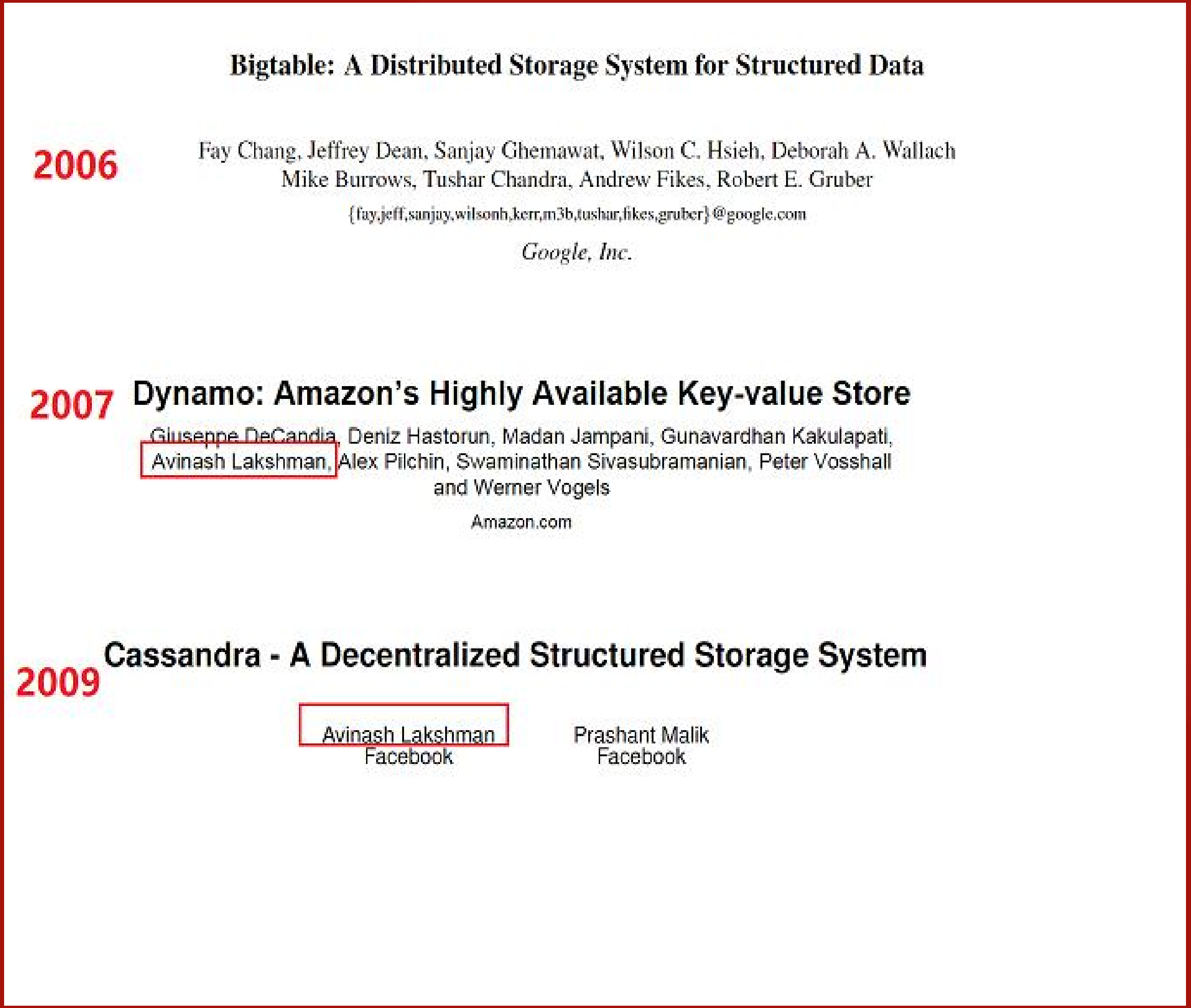
Rank			DBMS	Database Model	Score		
Nov 2022	Oct 2022	Nov 2021			Nov 2022	Oct 2022	Nov 2021
1.	1.	1.	Oracle	Relational, Multi-model	1241.69	+5.32	-31.04
2.	2.	2.	MySQL	Relational, Multi-model	1205.54	+0.17	-5.98
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	912.51	-12.17	-41.78
4.	4.	4.	PostgreSQL	Relational, Multi-model	623.16	+0.44	+25.88
5.	5.	5.	MongoDB	Document, Multi-model	477.90	-8.33	-9.45
6.	6.	6.	Redis	Key-value, Multi-model	182.05	-1.33	+10.55
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	150.32	-0.74	-8.76
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	149.56	-0.10	-17.96
9.	9.	↑ 11.	Microsoft Access	Relational	135.03	-3.14	+15.79
10.	10.	↓ 9.	SQLite	Relational	134.63	-3.17	+4.83
11.	11.	↓ 10.	Cassandra	Wide column	118.12	+0.18	-2.76
12.	↑ 13.	↑ 18.	Snowflake	Relational	110.15	+3.43	+45.97
13.	↓ 12.	↓ 12.	MariaDB	Relational, Multi-model	104.91	-4.40	+2.72
14.	14.	↓ 13.	Splunk	Search engine	94.23	-0.43	+1.92
15.	15.	↑ 16.	Amazon DynamoDB	Multi-model	85.40	-2.95	+8.41
16.	16.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model	83.66	-1.30	+2.34
17.	17.	↓ 14.	Hive	Relational	81.89	+1.29	-1.42
18.	18.	↓ 17.	Teradata	Relational, Multi-model	65.23	-0.84	-4.35
19.	↑ 20.		Databricks	Multi-model	60.89	+3.28	
20.	↓ 19.	↓ 19.	Neo4j	Graph	57.30	-1.38	-0.68
21.	↑ 23.	21.	FileMaker	Relational	54.31	+1.90	+0.08
22.	22.	↑ 24.	Google BigQuery	Relational	54.13	+1.68	+9.13
23.	↑ 24.	↓ 20.	SAP HANA	Relational, Multi-model	51.45	-0.63	-4.08
24.	↓ 21.	↓ 22.	Solr	Search engine, Multi-model	51.33	-2.17	-2.52
25.	25.	↓ 23.	SAP Adaptive Server	Relational, Multi-model	43.58	+0.62	-7.35
26.	26.	↓ 25.	HBase	Wide column	40.41	-1.25	-4.59
27.	27.	↓ 26.	Microsoft Azure Cosmos DB	Multi-model	39.75	-0.67	-1.08
28.	28.	↓ 27.	PostGIS	Spatial DBMS, Multi-model	30.78	-0.08	-1.14
29.	29.	29.	InfluxDB	Time Series, Multi-model	29.96	+0.38	+1.42
30.	30.	↓ 28.	Couchbase	Document, Multi-model	28.62	+0.66	-1.25
31.	31.	↑ 32.	Amazon Redshift	Relational	27.04	+0.03	+2.17
32.	↑ 33.	↓ 30.	Firebird	Relational	25.37	+0.36	-1.54
33.	↓ 32.	↓ 31.	Memcached	Key-value	24.45	-0.95	-1.92
34.	34.	↑ 39.	Microsoft Azure Synapse Analytics	Relational	23.03	-0.07	+5.13
35.	35.	↓ 34.	Informix	Relational, Multi-model	22.82	-0.05	-0.41
36.	36.	↓ 33.	Spark SQL	Relational	21.90	-0.68	-1.43
37.	37.	↓ 36.	Firebase Realtime Database	Document	19.84	-0.29	-0.23
38.	38.	↓ 35.	Vertica	Relational, Multi-model	19.70	-0.15	-0.63
39.	↑ 40.	↓ 38.	Impala	Relational, Multi-model	17.92	+0.24	-1.01
40.	↓ 39.	↓ 37.	Netezza	Relational	17.27	-0.57	-2.52
41.	41.	↓ 40.	CouchDB	Document, Multi-model	15.98	-0.27	-0.82



Rank			DBMS	Database Model	Score		
Nov 2022	Oct 2022	Nov 2021			Nov 2022	Oct 2022	Nov 2021
1.	1.	1.	Cassandra	Wide column	118.12	+0.18	-2.76
2.	2.	2.	HBase	Wide column	40.41	-1.25	-4.59
3.	3.	3.	Microsoft Azure Cosmos DB	Multi-model	39.75	-0.67	-1.08
4.	4.	4.	Datastax Enterprise	Wide column, Multi-model	8.68	+0.39	-1.01
5.	5.	5.	Microsoft Azure Table Storage	Wide column	6.02	+0.05	+0.33
6.	6.	6.	Google Cloud Bigtable	Multi-model	5.24	-0.20	+0.97
7.	↑ 8.	↑ 8.	Accumulo	Wide column	4.98	+0.41	+1.07
8.	↓ 7.	↓ 7.	ScyllaDB	Wide column, Multi-model	4.95	+0.26	+1.01
9.	9.	9.	HPE Ezmeral Data Fabric	Multi-model	1.26	+0.13	+0.37
10.	10.	↑ 11.	Amazon Keyspaces	Wide column	0.77	-0.02	+0.24
11.	11.	↓ 10.	Elassandra	Wide column, Multi-model	0.57	+0.09	-0.06
12.	12.	12.	Alibaba Cloud Table Store	Wide column	0.33	-0.03	-0.08
13.	13.	13.	SWC-DB	Wide column, Multi-model	0.07	-0.02	+0.07

1. 简介

The content of the title



Bigtable: A Distributed Storage System for Structured Data

2006

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

2007 **Dynamo: Amazon's Highly Available Key-value Store**

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,
Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall
and Werner Vogels
Amazon.com

2009 **Cassandra - A Decentralized Structured Storage System**

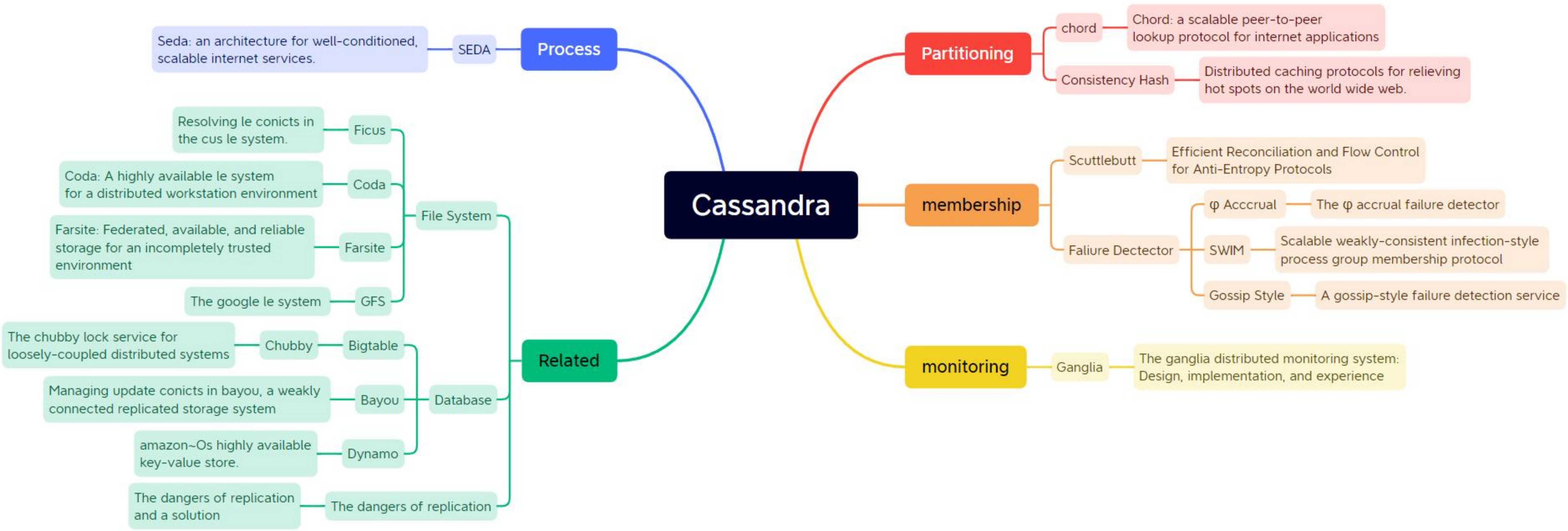
Avinash Lakshman
Facebook

Prashant Malik
Facebook

	Dynamo	Bigtable
单机	插件化：BDB transactional data store，BDB java,MySQL	LSM：LevelDB Data Model：Row，Column Family，TImestamp
分布式	Consistent hash，Vector clock，Sloppy Quorum，Hinted handoff，Anti-entroy using merkle tree，Gossip based membership protocol and failure detection	依赖于GFS的primary-secondary，relaxed consistency，Chubby

1. 简介

The content of the title

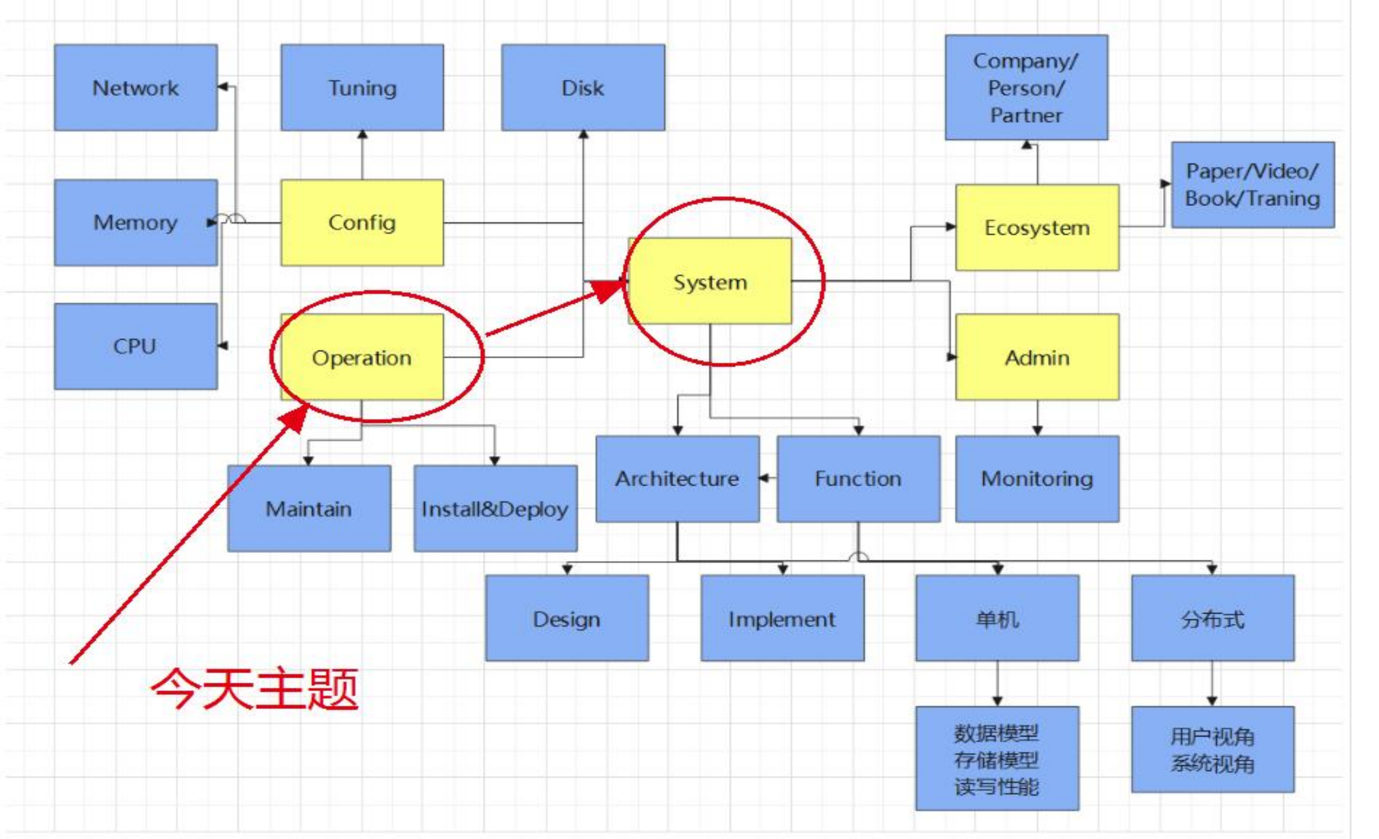


- Cassandra
  - Process
    - SEDA
      - Seda: an architecture for well-conditioned, scalable internet services.
  - Related
    - File System
      - Ficus
        - Resolving le conicts in the cus le system.
      - Coda
        - Coda: A highly available le system for a distributed workstation environment
      - Farsite
        - Farsite: Federated, available, and reliable storage for an incompletely trusted environment
      - GFS
        - The google le system
    - Database
      - Bigtable
        - Chubby
          - The chubby lock service for loosely-coupled distributed systems
      - Bayou
        - Managing update conicts in bayou, a weakly connected replicated storage system
      - Dynamo
        - amazon~Os highly available key-value store.
    - The dangers of replication
      - The dangers of replication and a solution
  - Partitioning
    - chord
      - Chord: a scalable peer-to-peer lookup protocol for internet applications
    - Consistency Hash
      - Distributed caching protocols for relieving hot spots on the world wide web.
  - membership
    - Scuttlebutt
      - Efficient Reconciliation and Flow Control for Anti-Entropy Protocols
    - Faliure Dectector
      - φ Acccrual
        - The φ accrual failure detector
      - SWIM
        - Scalable weakly-consistent infection-style process group membership protocol
      - Gossip Style
        - A gossip-style failure detection service
  - monitoring
    - Ganglia
      - The ganglia distributed monitoring system: Design, implementation, and experience

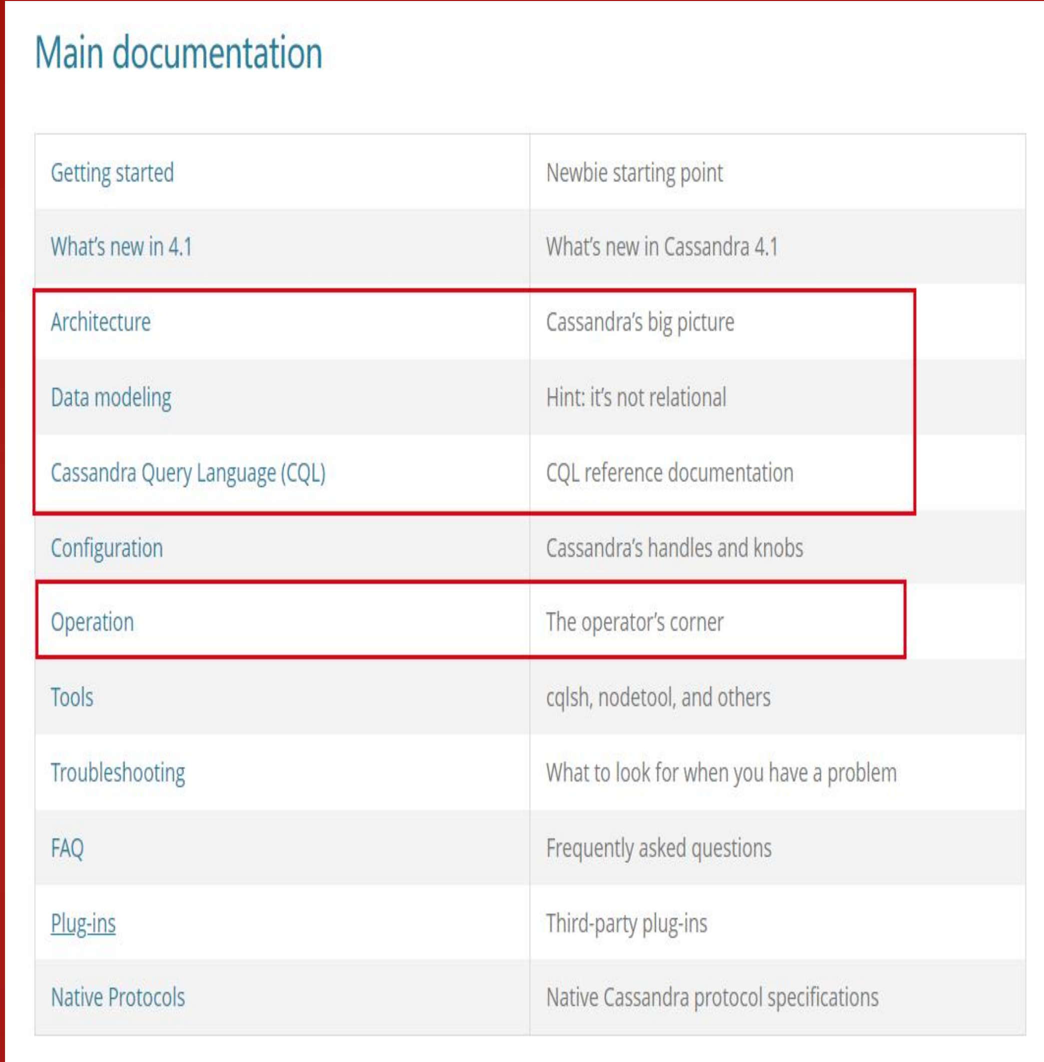
1. 简介-研究模型

The content of the title

- COSEA模型
  - 参考链接



今天主题



Main documentation

Getting started	Newbie starting point
What's new in 4.1	What's new in Cassandra 4.1
Architecture	Cassandra's big picture
Data modeling	Hint: it's not relational
Cassandra Query Language (CQL)	CQL reference documentation
Configuration	Cassandra's handles and knobs
Operation	The operator's corner
Tools	cqlsh, nodetool, and others
Troubleshooting	What to look for when you have a problem
FAQ	Frequently asked questions
Plug-ins	Third-party plug-ins
Native Protocols	Native Cassandra protocol specifications

2. 数据建模-概念

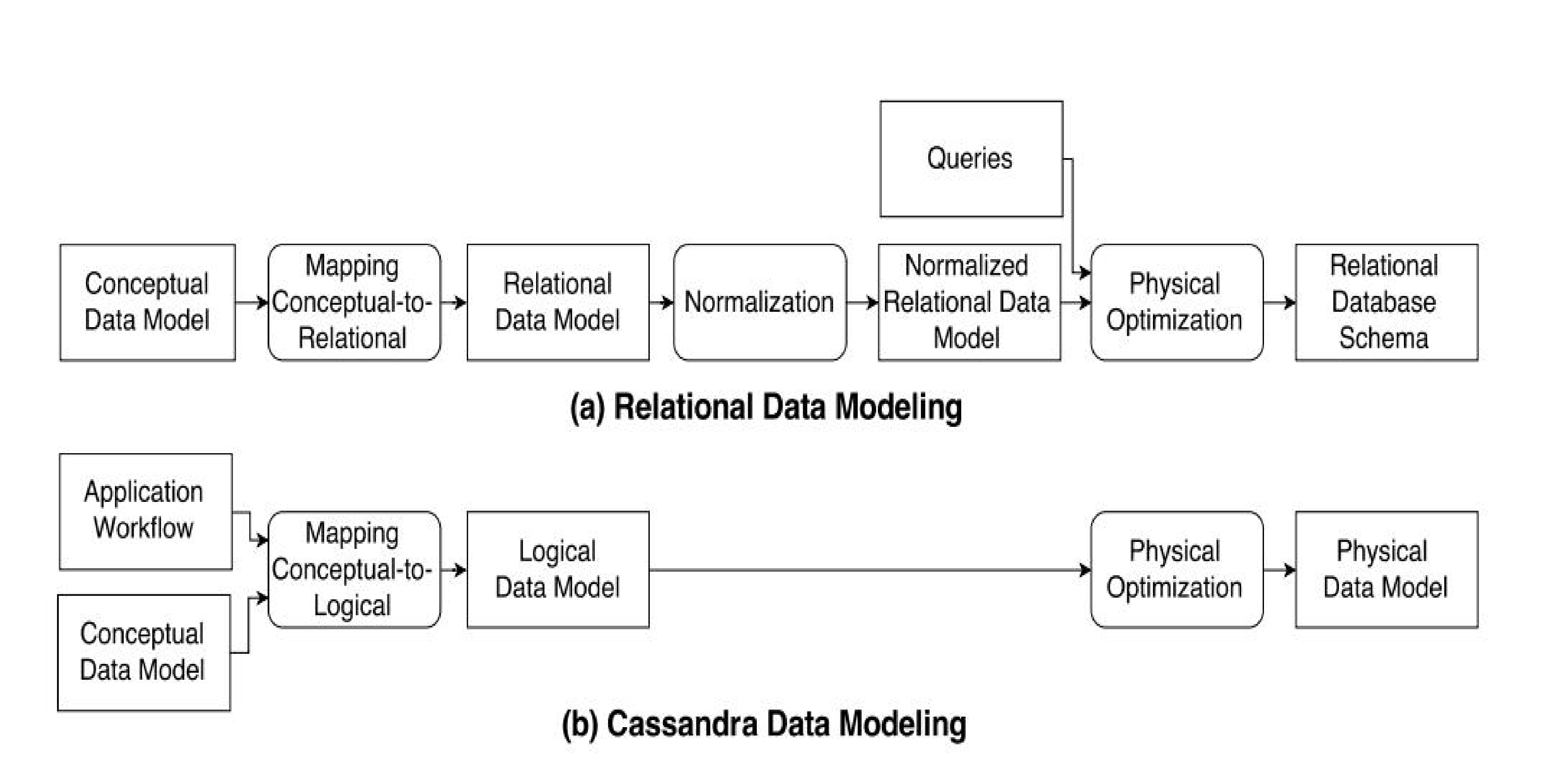
The content of the title

- 理论
  - 范式和反范式，数仓（Star/Snowflake），可变和不可变
- 产品
  - 网状，关系型，NoSQL(KV,文档,宽列,图形)，搜索，数仓列式
- 案例
  - 预定酒店建模
- 重要
  - 模型影响数据存储以及查询。

2. 数据建模-过程

The content of the title

- 建模过程：查询驱动建模



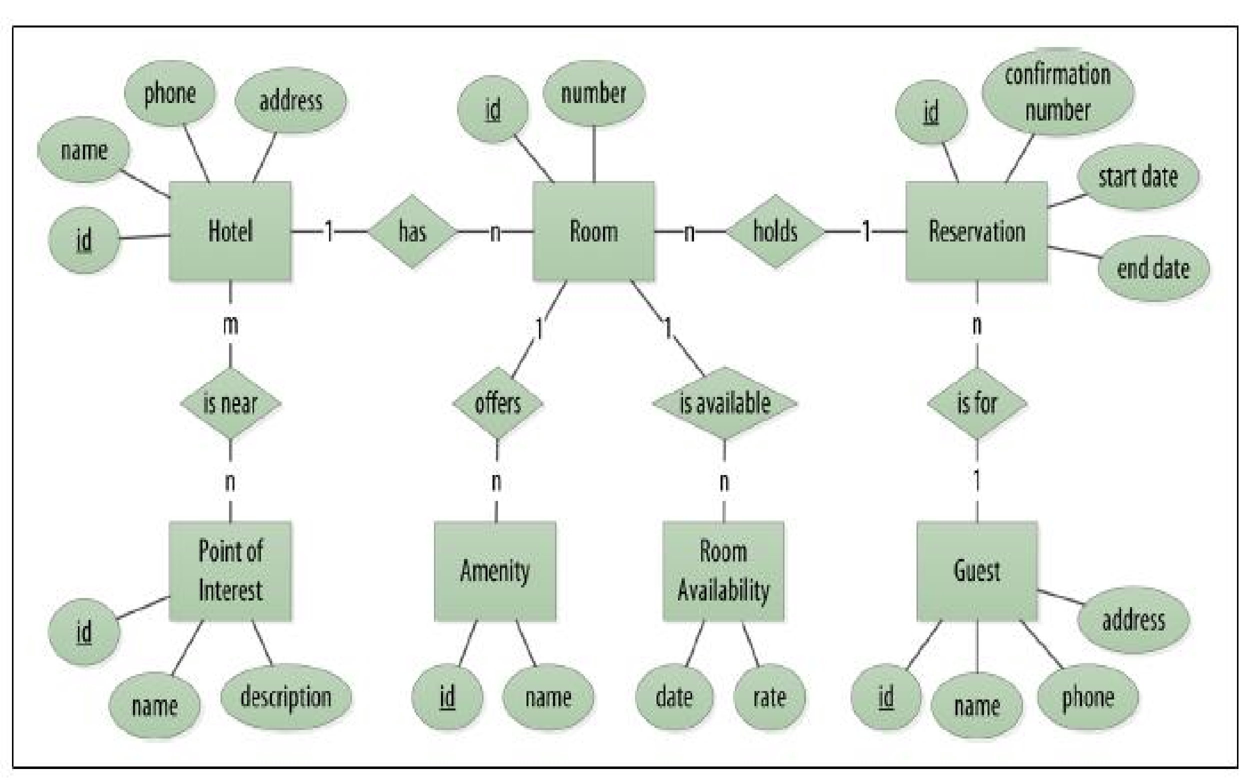
(a) Relational Data Modeling

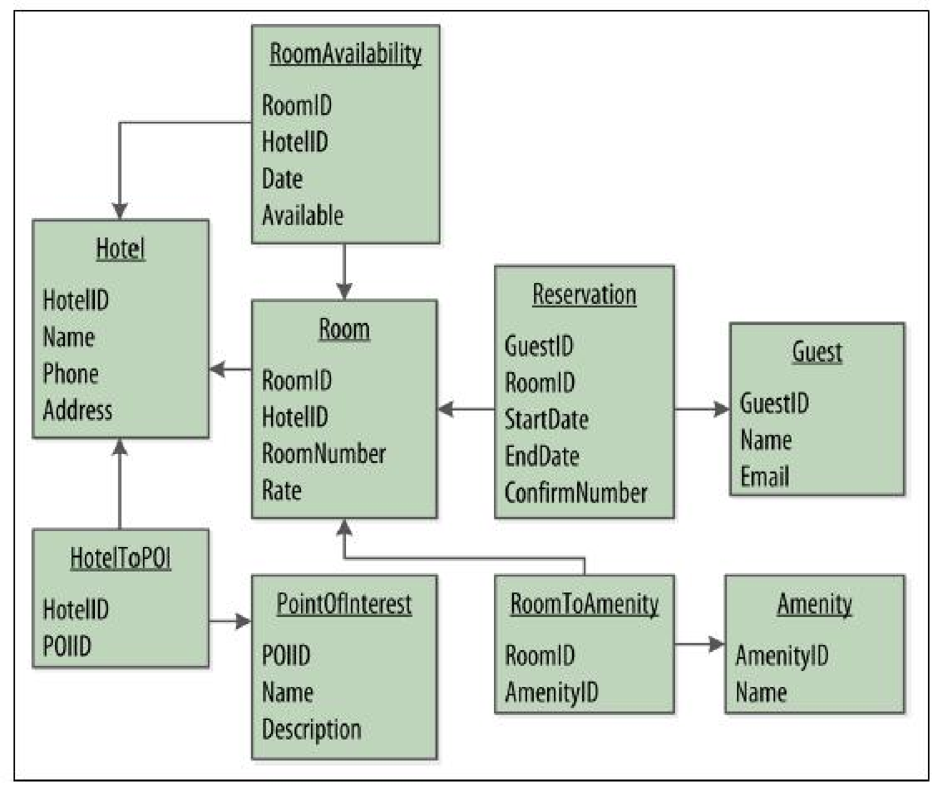
(b) Cassandra Data Modeling

2. 数据建模-案例

The content of the title

- 酒店预定

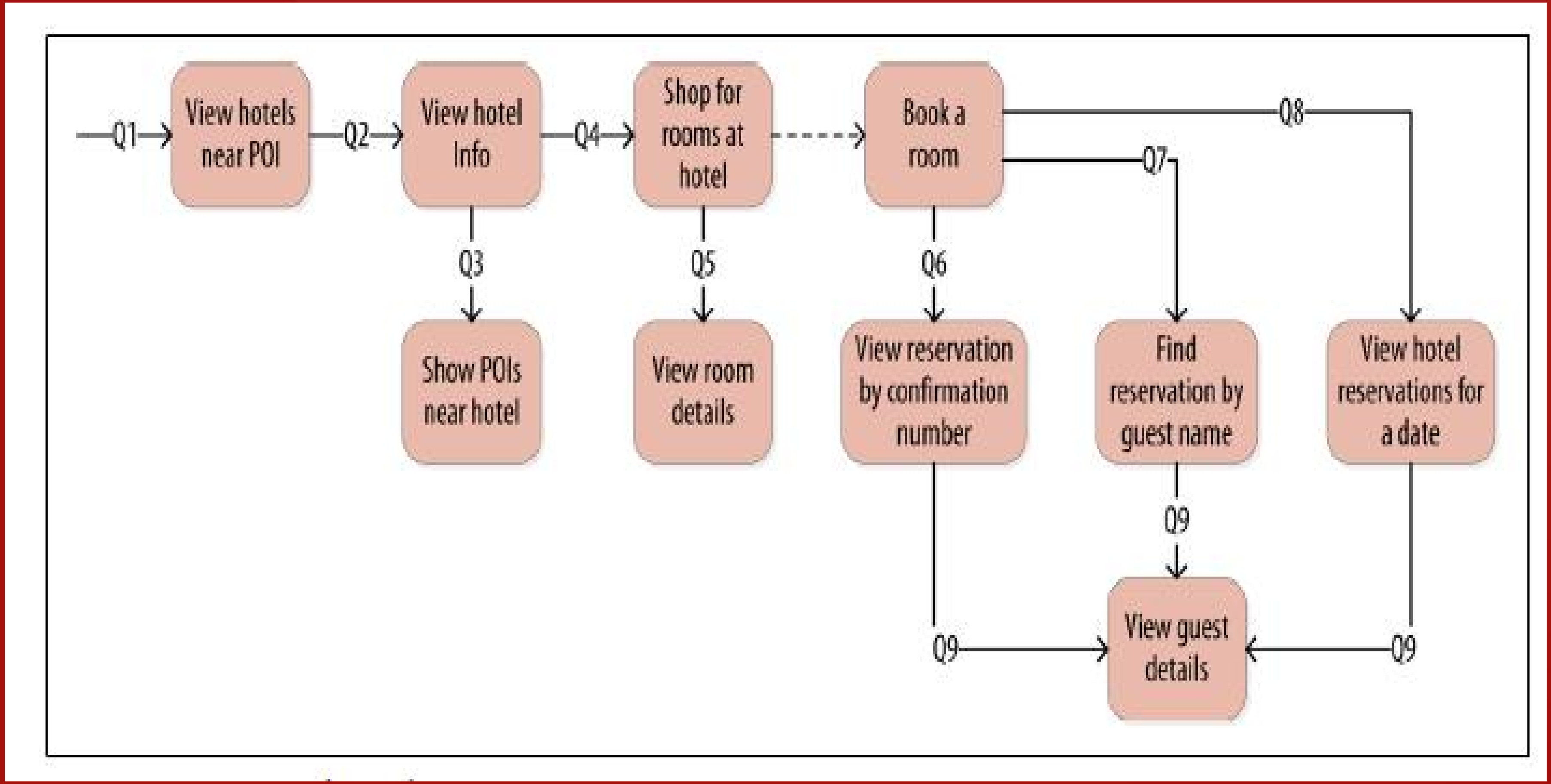




2. 数据建模-案例-工作流

The content of the title

- 定义应用查询



2. 数据建模-案例-逻辑数据建模

The content of the title

- 逻辑数据建模
  - 酒店领域
  - 预定领域

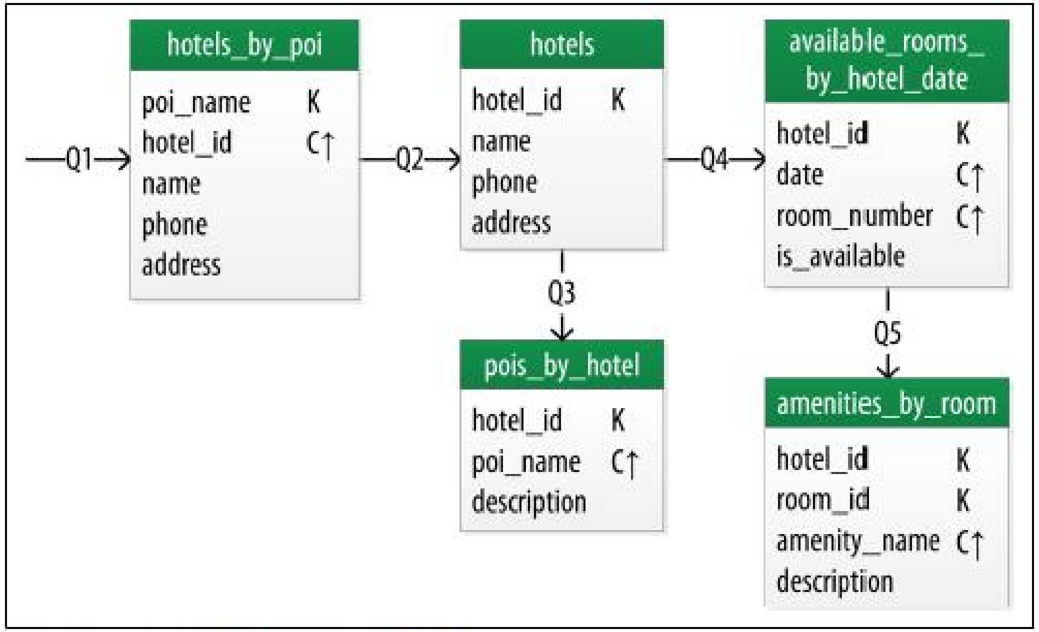


Figure 5-5. Hotel domain logical model

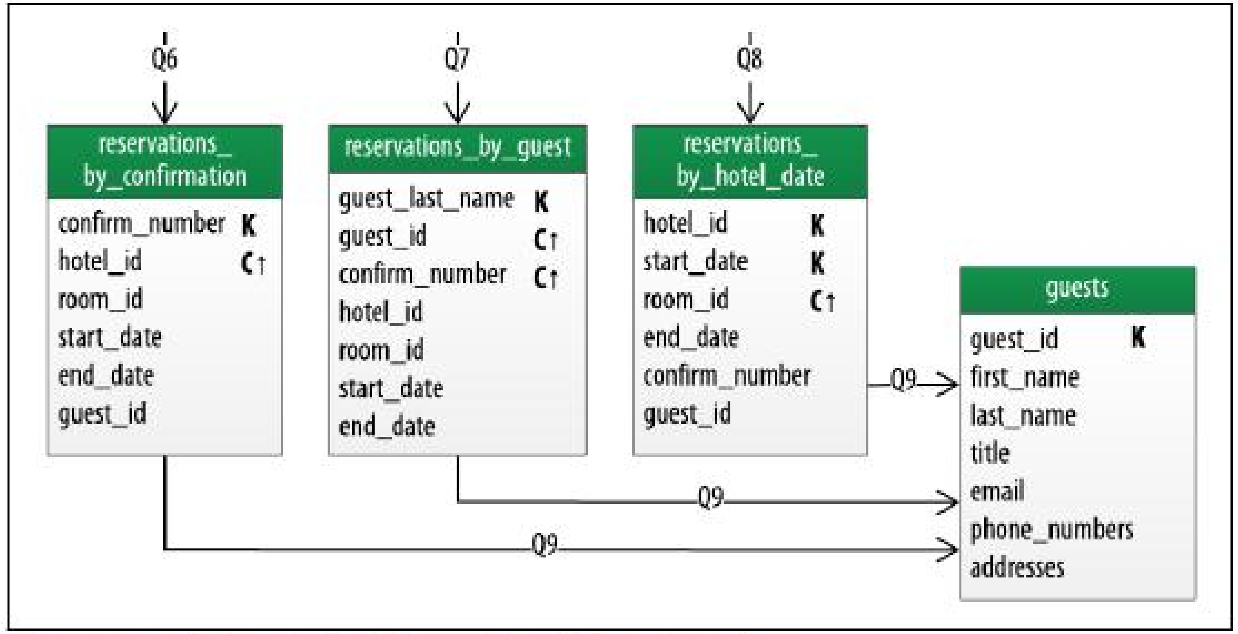
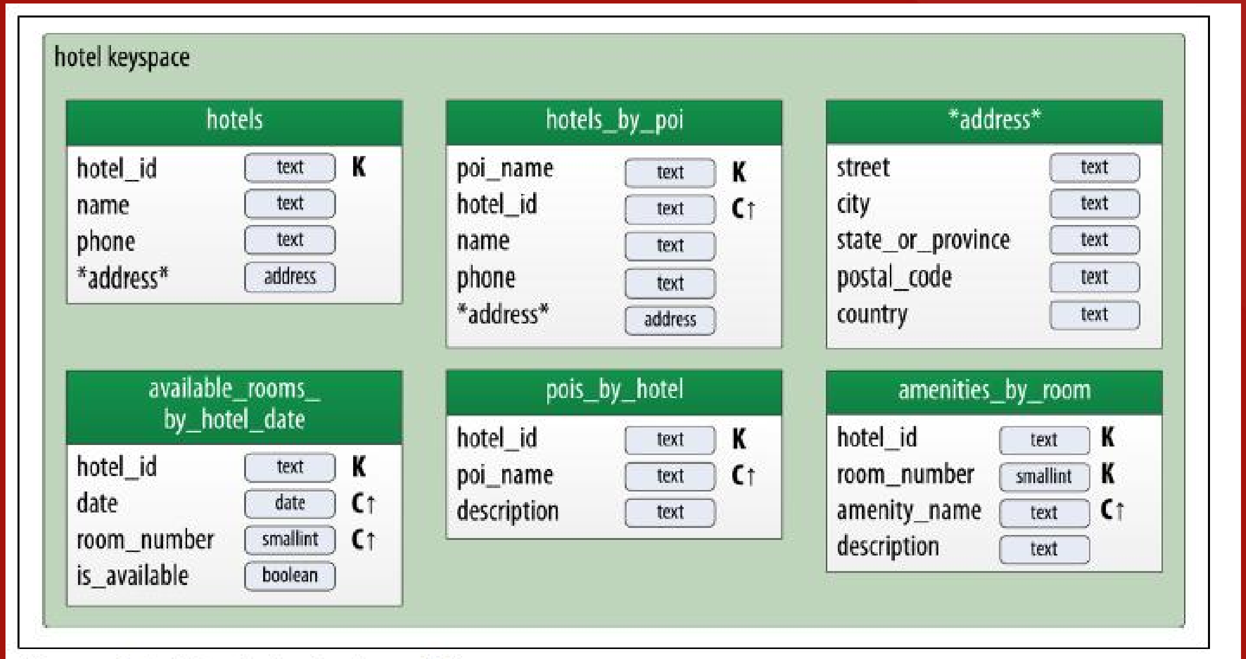


Figure 5-6. A denormalized logical model for reservations

2. 数据建模-案例-物理数据建模

The content of the title

- 物理数据建模（Chebotko图）
  - 酒店keyspace
  - 预定keyspace



hotel keyspace

hotels		
hotel_id	text	K
name	text	
phone	text	
address	address	

hotels_by_poi		
poi_name	text	K
hotel_id	text	C↑
name	text	
phone	text	
address	address	

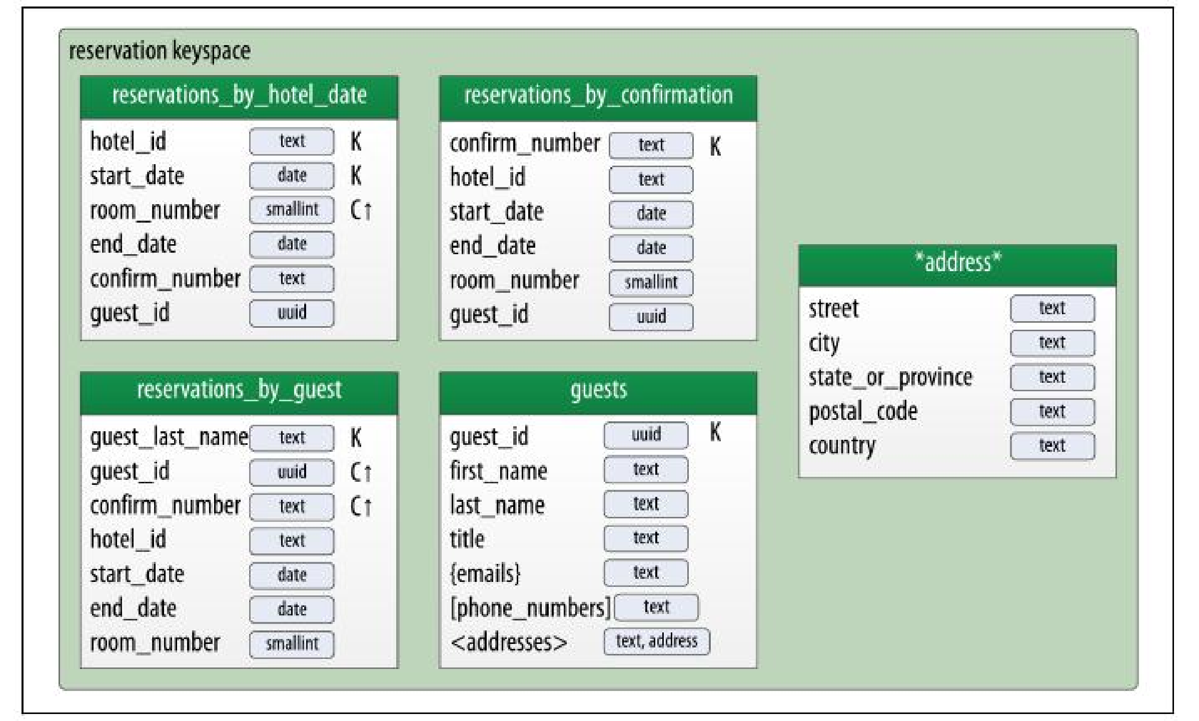
address	
street	text
city	text
state_or_province	text
postal_code	text
country	text

available_rooms_by_hotel_date		
hotel_id	text	K
date	date	C↑
room_number	smallint	C↑
is_available	boolean	

pois_by_hotel		
hotel_id	text	K
poi_name	text	C↑
description	text	

amenities_by_room		
hotel_id	text	K
room_number	smallint	K
amenity_name	text	C↑
description	text	

Figure 5-8. Hotel physical model



reservation keyspace

reservations_by_hotel_date		
hotel_id	text	K
start_date	date	K
room_number	smallint	C↑
end_date	date	
confirm_number	text	
guest_id	uuid	

reservations_by_confirmation		
confirm_number	text	K
hotel_id	text	
start_date	date	
end_date	date	
room_number	smallint	
guest_id	uuid	

reservations_by_guest		
guest_last_name	text	K
guest_id	uuid	C↑
confirm_number	text	C↑
hotel_id	text	
start_date	date	
end_date	date	
room_number	smallint	

guests		
guest_id	uuid	K
first_name	text	
last_name	text	
title	text	
{emails}	text	
[phone_numbers]	text	
<addresses>	text, address	

address	
street	text
city	text
state_or_province	text
postal_code	text
country	text

Figure 5-9. Reservation physical model

2. 数据建模-案例-评估和改进

The content of the title

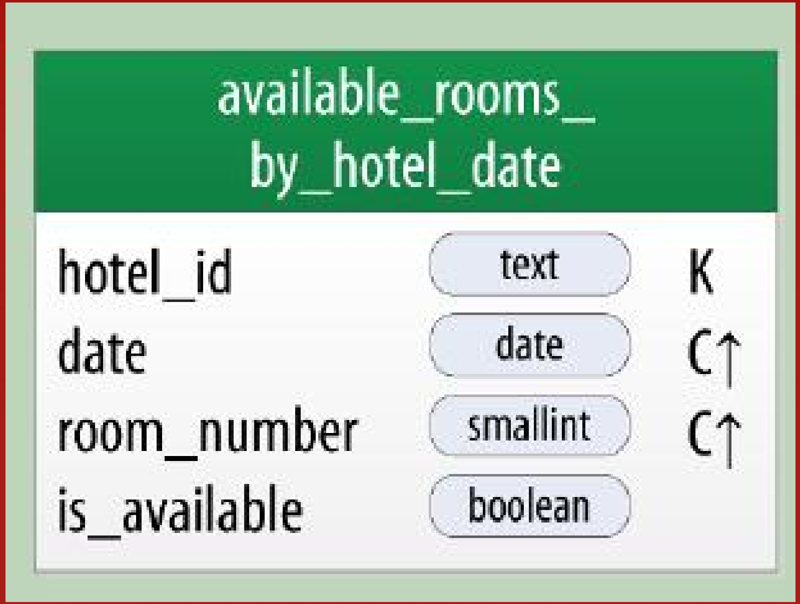
- 计算分区大小

$$N_v = N_r\left(N_c - N_{pk} - N_s\right) + N_s$$

- 计算磁盘大小

$$S_t = \sum_i sizeOf\left(c_{k_i}\right) + \sum_j sizeOf\left(c_{s_j}\right) + N_r \times \left(\sum_k sizeOf\left(c_{r_k}\right) + \sum_l sizeOf\left(c_{c_l}\right)\right) + N_v \times sizeOf\left(t_{avg}\right)$$

- 分解大分区
  - 分区键加列，分桶



available_rooms_by_hotel_date		
hotel_id	text	K
date	date	C↑
room_number	smallint	C↑
is_available	boolean	

2.数据建模-案例-定义数据Schema

The content of the title

- 酒店：参考
- 预定：参考

2. 数据建模-案例-总结

The content of the title

- 反范式化
- 查询驱动建模
- 主键查询优先
- 搜索/聚合转es，分析spark或数仓
- 常见错误整理：参考

2.2.数据建模-客户端

The content of the title

- 线上服务器版本 [cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
- Driver版本：3.7.0

2. 数据建模-客户端

The content of the title

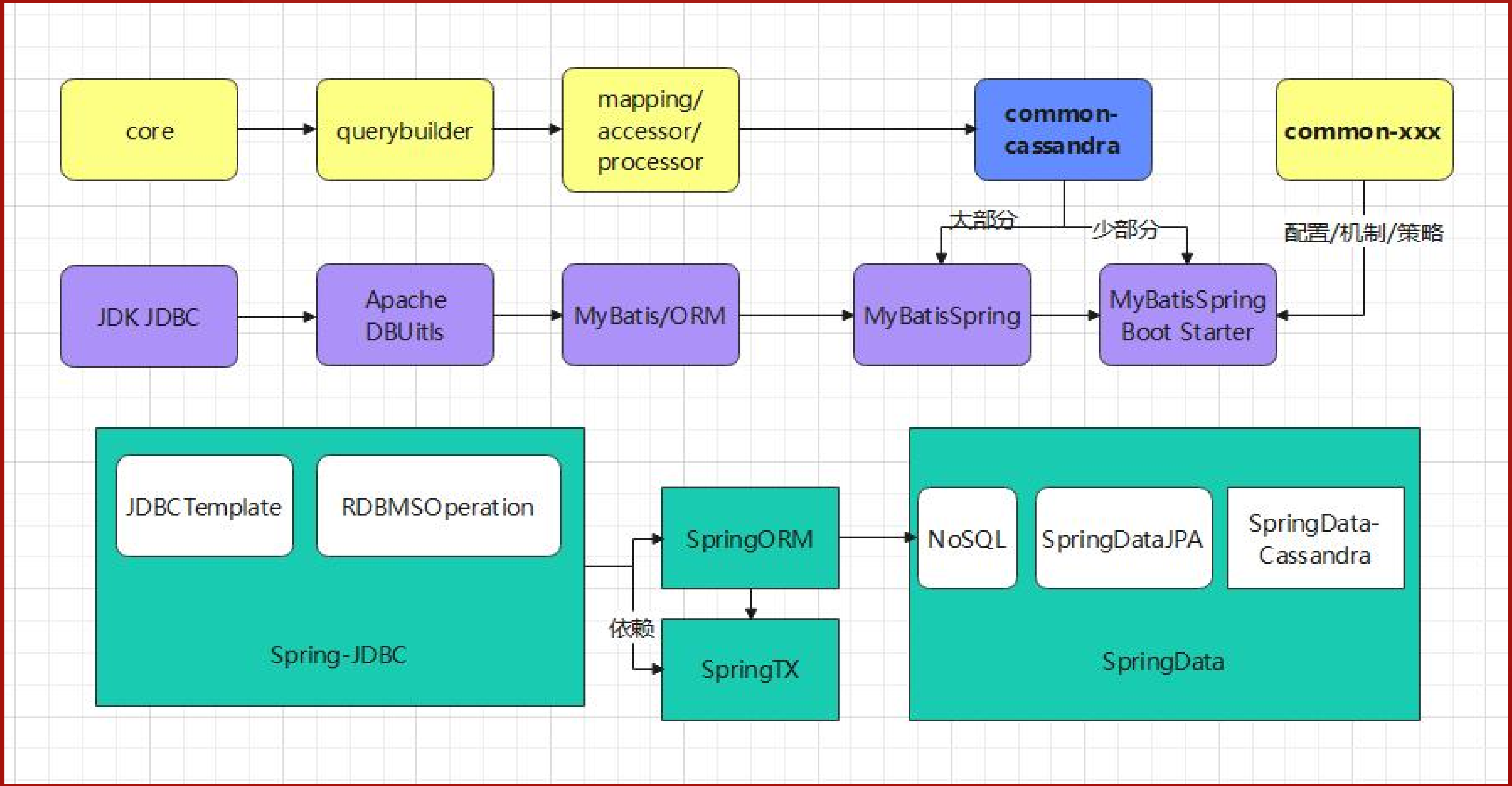
- 存储层的封装非常成熟，模式非常固定，机制却因人而异
  - 同步，异步，反应式
  - 技术演进
    - JDBC->DBUtils->Spring JDBC（JDBCTemplate, RDBMSOperation） -> Spring ORM->Spring data.
    - MyBatis->MyBatis-Spring->MyBatis-Spring-starter.
  - 3.11.0 statement/mapper/accessor 4.0.0 processor

- Cluster和Session
  - 所有keyspace一个session
  - 每个keyspace一个session

2. 数据建模-客户端

The content of the title

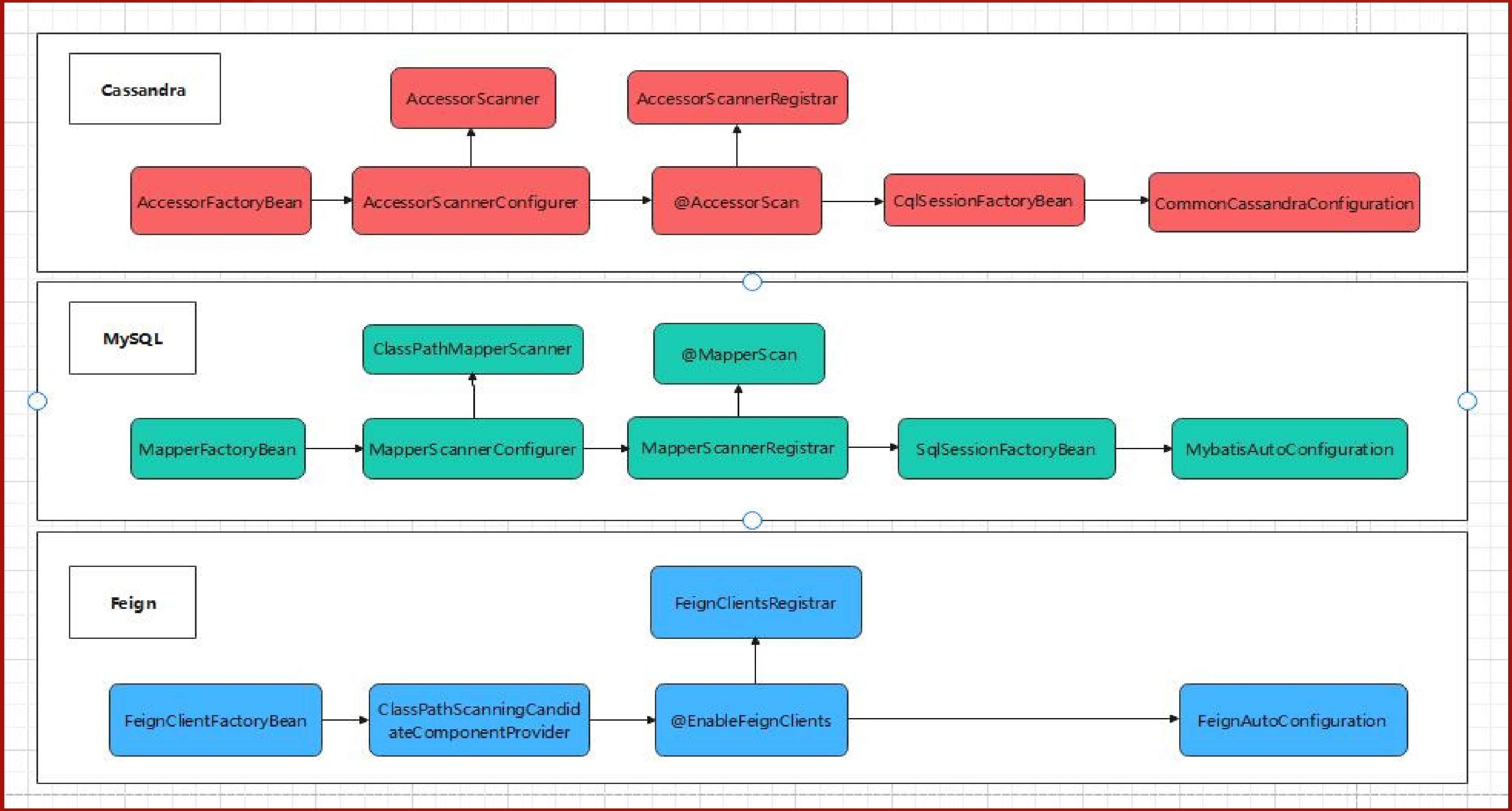
- common-cassandra



2. 数据建模-客户端

The content of the title

- common-cassandra



2. 数据建模-项目使用

The content of the title

- 直播
  - course_live_msg：live_stu_voice live_notice live_all_chat live_stu_praise
- 电商
  - course_process：user_lesson_progress user_watch_process user_point_history user_process user_point_product
  - course_naive：user_profile
- 智学
  - adaplearn_tiku：answers_by_user quiz_user_progress progress_by_user quiz_user_answer exam_user_answer
- 题库
  - course_tiku：user_answer，user_live_answer2，user_paper_answer

2.数据建模-CQRS+Event Source

The content of the title

- Akka Persistent Cassandra

3. 系统架构

The content of the title

- 架构
  - 内部实现，外部交互，系统属性
  - 单机特性-对标数据库
    - 数据模型，存储模型，查询模型（读写性能）
  - 分布式特性-对标分布式系统
    - 用户视角：读写路径
    - 系统视角：复制和一致性，容错，分区，共识
  - 其他模型：https://xiaozhiliaoo.github.io/reading-note/tech-quotes/chapter8.html 820条

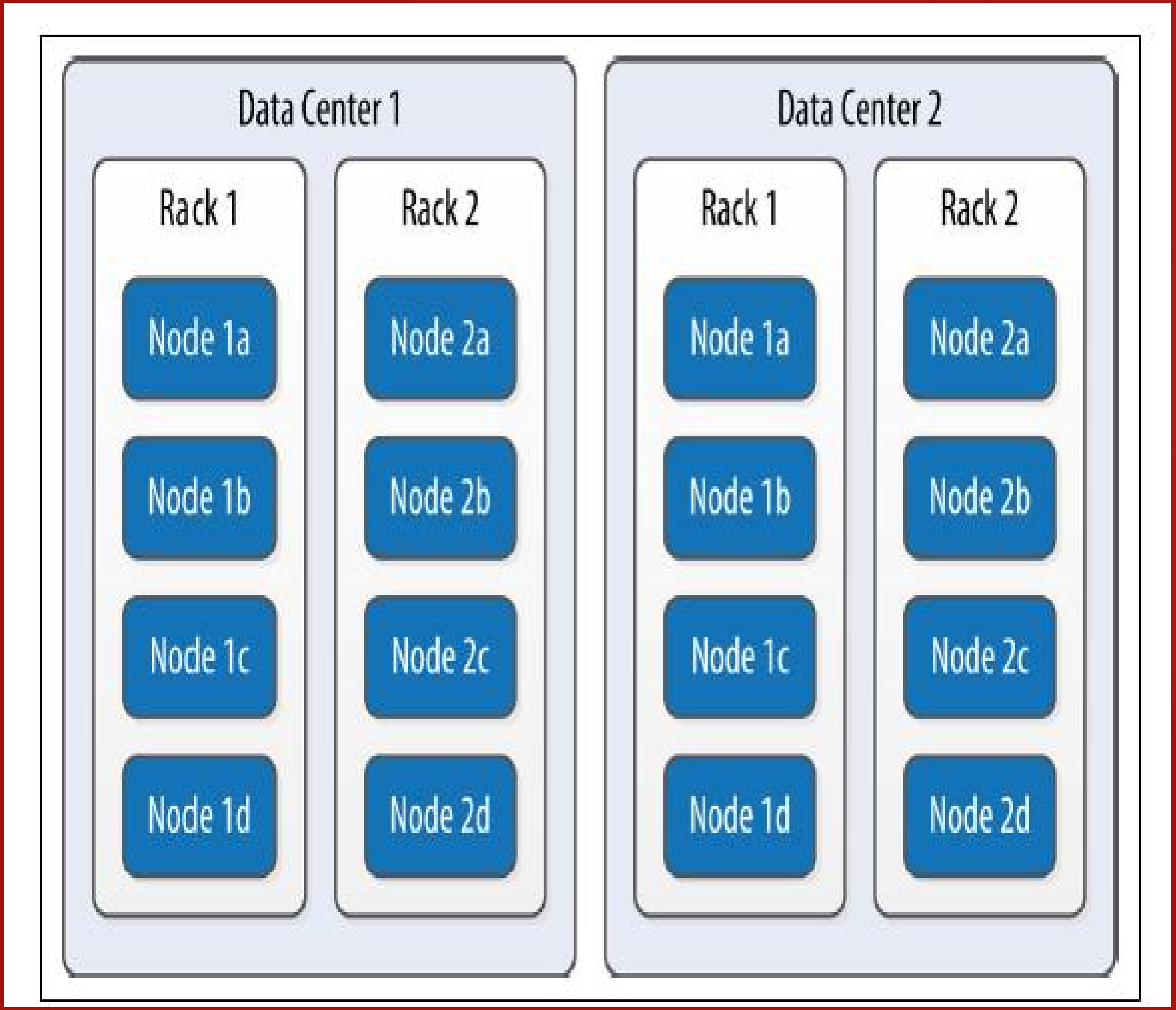
3. 系统架构

The content of the title

- 单机
  - Commit Log，Memmory table，SSTable，布隆过滤器，墓碑，合并，压缩
- 分布式
  - 数据中心和机架
  - Gossip和故障检测，Snitch感应策略
  - DHT和一致性Hash，环和令牌，虚拟节点
  - 分区器
  - 复制策略
  - 一致性级别
  - 查询和协调器节点
  - hint handoff，anti-entory，repair，merkle tree
  - 轻量级事务和Paxos
  - SEDA
- 基本问题
  - 读写路径是什么？（一条SQL执行/打开网页发生了什么）

3. 系统架构-数据中心和机架

The content of the title



Data Center 1

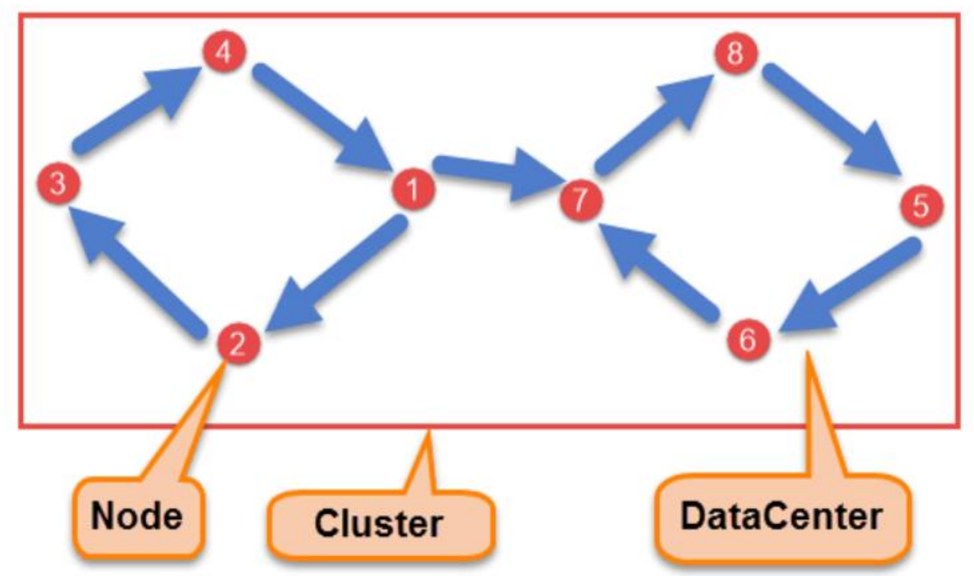
Rack 1: Node 1a, Node 1b, Node 1c, Node 1d

Rack 2: Node 2a, Node 2b, Node 2c, Node 2d

Data Center 2

Rack 1: Node 1a, Node 1b, Node 1c, Node 1d

Rack 2: Node 2a, Node 2b, Node 2c, Node 2d



Cassandra Architecture Diagram

3. 系统架构-通信Gossip，故障检测，和Snitch

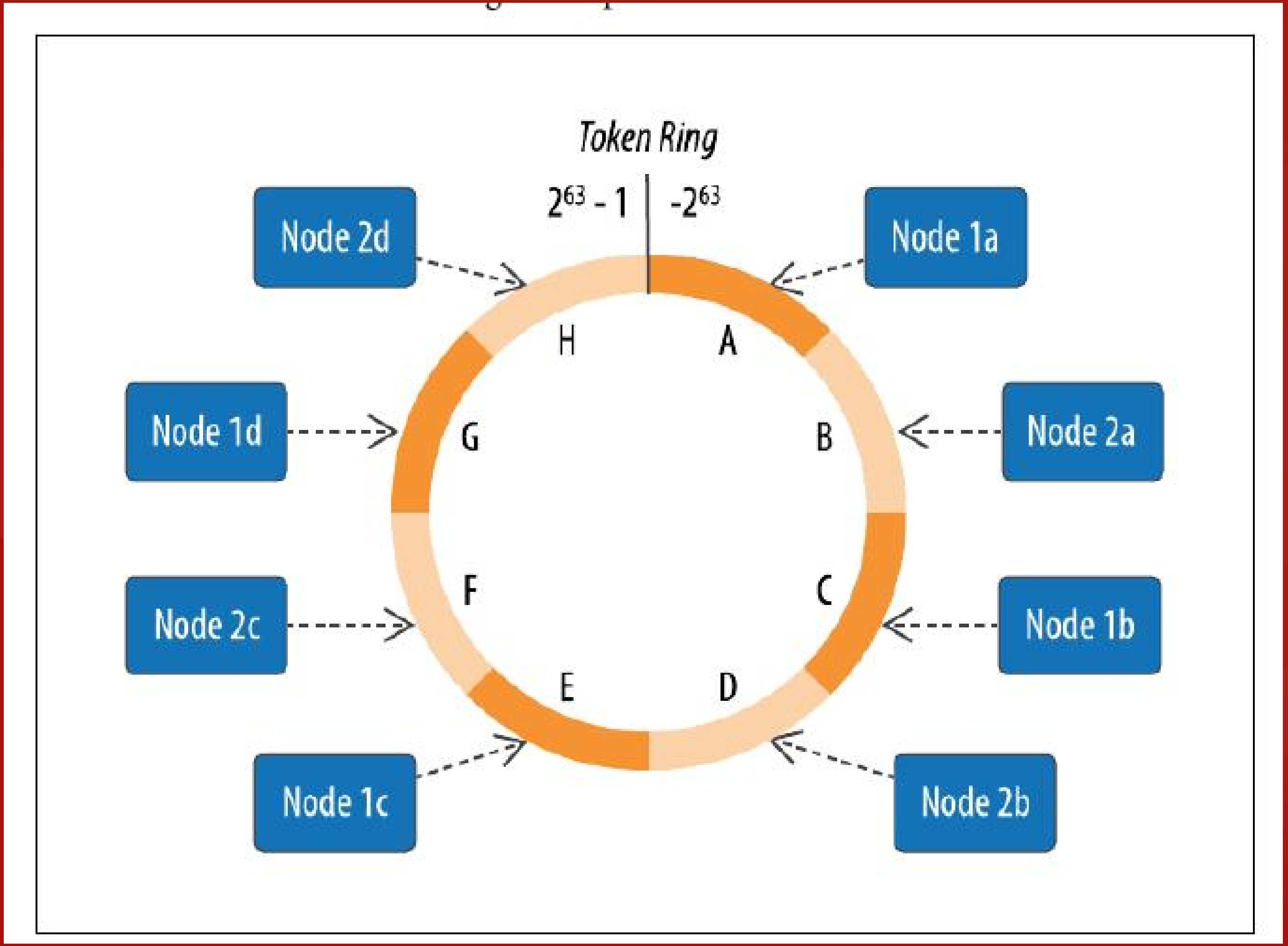
The content of the title

- Gossip
  - Information Dissemination模型的epidemic protocol两大类：anti-entropy和rumor-mongering
  - 设计考虑
    - reconciliation：precise，Scuttlebutt
    - flow control：Spreading Capacity Fairly，Local Adaptation
  - Scuttlebutt：https://awinterman.github.io/simple-scuttle/
- 故障检测
  - Phi Accrual FD
- Snitch
  - 确定相对主机远近程度。收集网络拓扑有关信息，处理路由请求。
  - SimpleSnitch
  - PropertyFileSnitch
  - GossipingPropertyFileSnitch
  - DynamicEndpointSnitch

3. 系统架构-Token Ring

The content of the title

- 一致性Hash和DHT
  - Ring和Range
  - Token
  - 虚拟节点（num_tokens）
- Hash算法评估
  - 单调性：旧数据不变性
  - 平衡性：节点各尽所能
    - 平衡不是平均，马太效应
- 通过环保证单调性
  通过虚拟节点保证平衡性

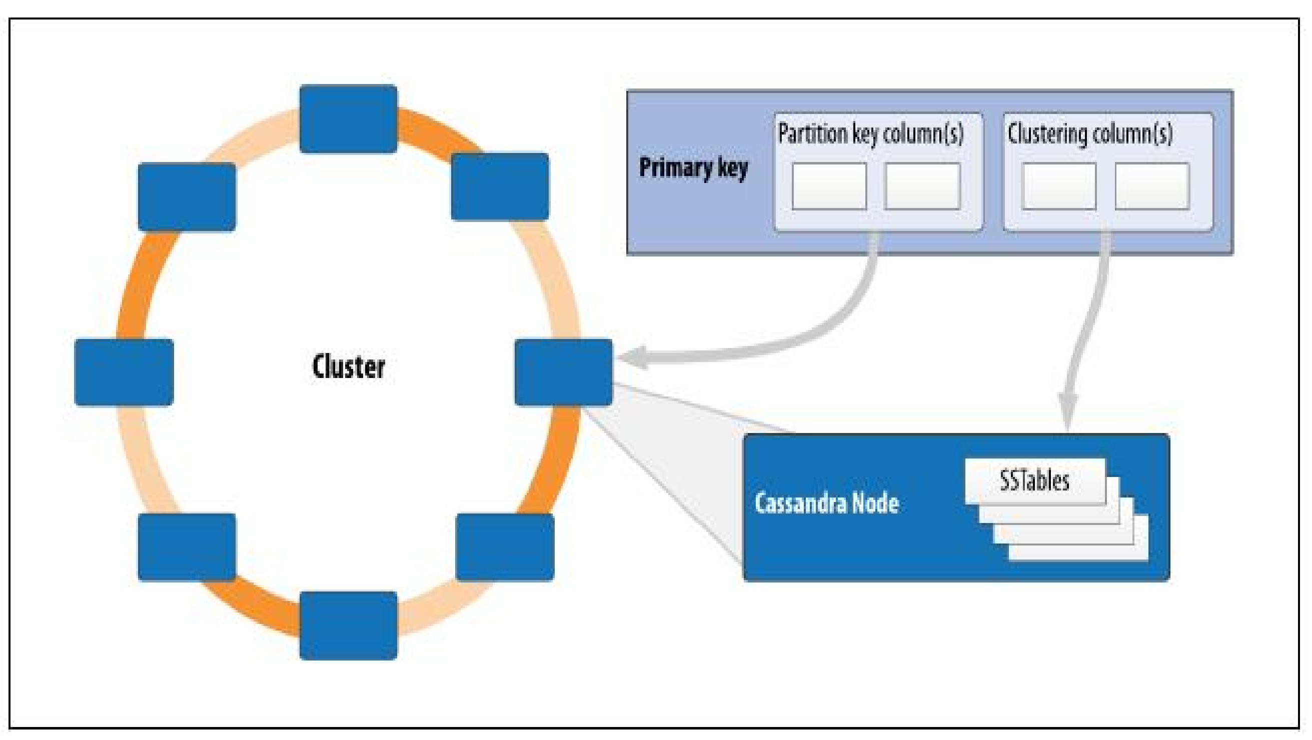


網易 NetEase

3. 系统架构-分区器

The content of the title

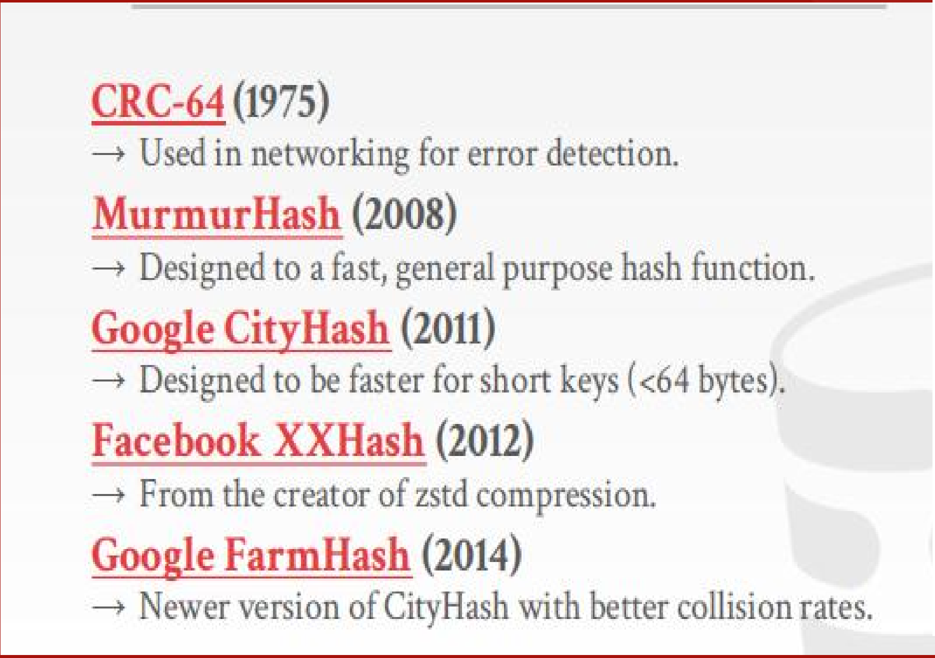
- 找到key在token ring的位置， token = partationFunction(partation key)
  - Murmur3Partitioner(MP-默认)
    - Mu:multiply R:rotate
    - Redis， Nginx,libmemcached,npm, Hadoop,Cassandra,Solr, Elasticsearch,Guava,Kafka， HBase Lucene
  - OrderPreservingPartitioner(OPP)
  - RandomPartitioner(RP)
  - ByteOrderedPartitioner(BOP)
- Murmur3
  - 即使键是有规律的， 算法仍能给出一个很好的随机分布性， 并且算法的计算速度也非常快



3. 系统架构-分区器

The content of the title

- 深入Hash算法



CRC-64 (1975)

→ Used in networking for error detection.

MurmurHash (2008)

→ Designed to a fast, general purpose hash function.

Google CityHash (2011)

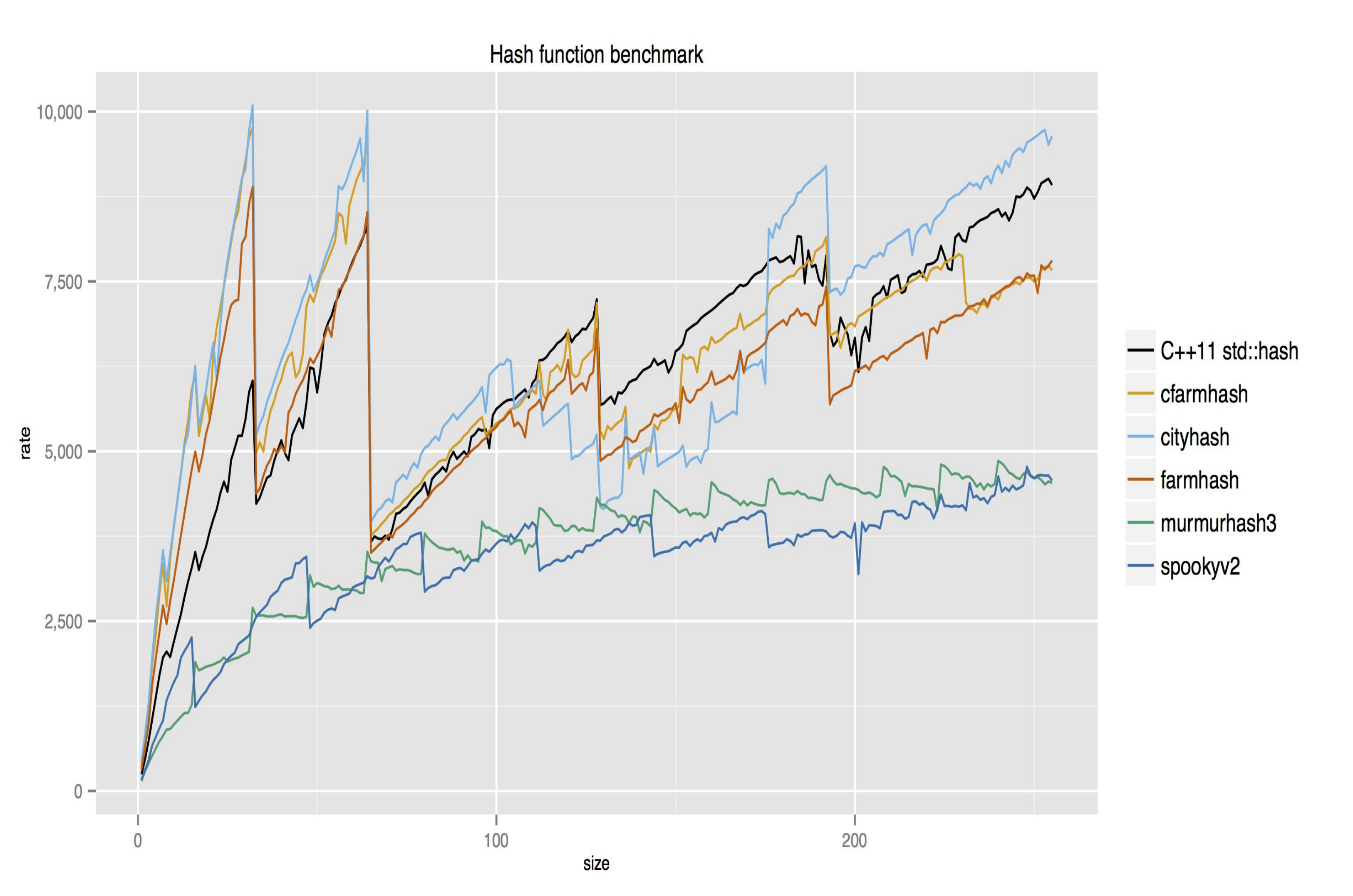
→ Designed to be faster for short keys (<64 bytes).

Facebook XXHash (2012)

→ From the creator of zstd compression.

Google FarmHash (2014)

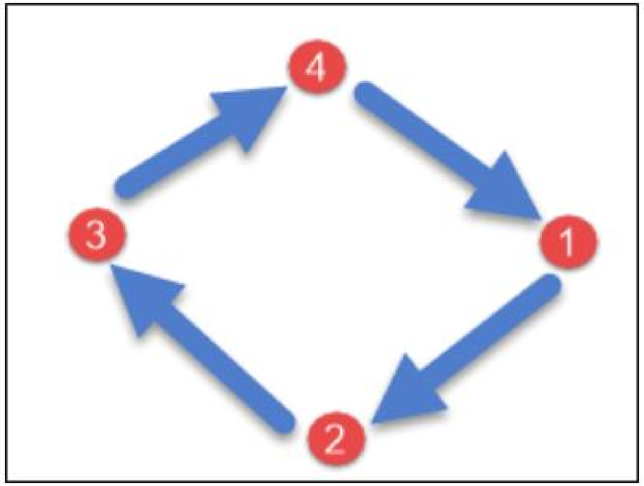
→ Newer version of CityHash with better collision rates.



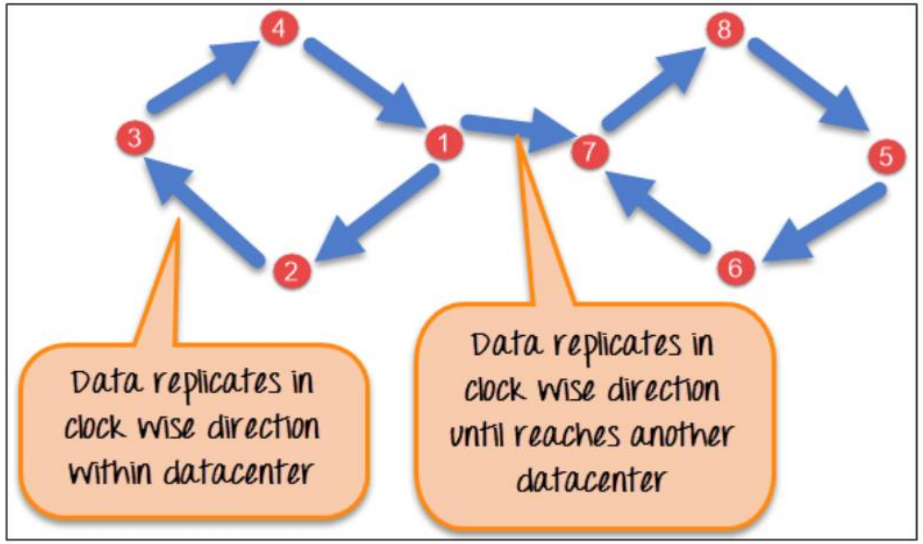
3. 系统架构-复制

The content of the title

- LeaderLess Replication
  - Versioned timestamp data(时钟同步)，LWW-Element-Set CRDT
  - 策略
    - SimpleStrategy
    - NetworkTopologyStrategy
    - LocalStrategy(system keyspace)



SimpleStrategy in Cassandra

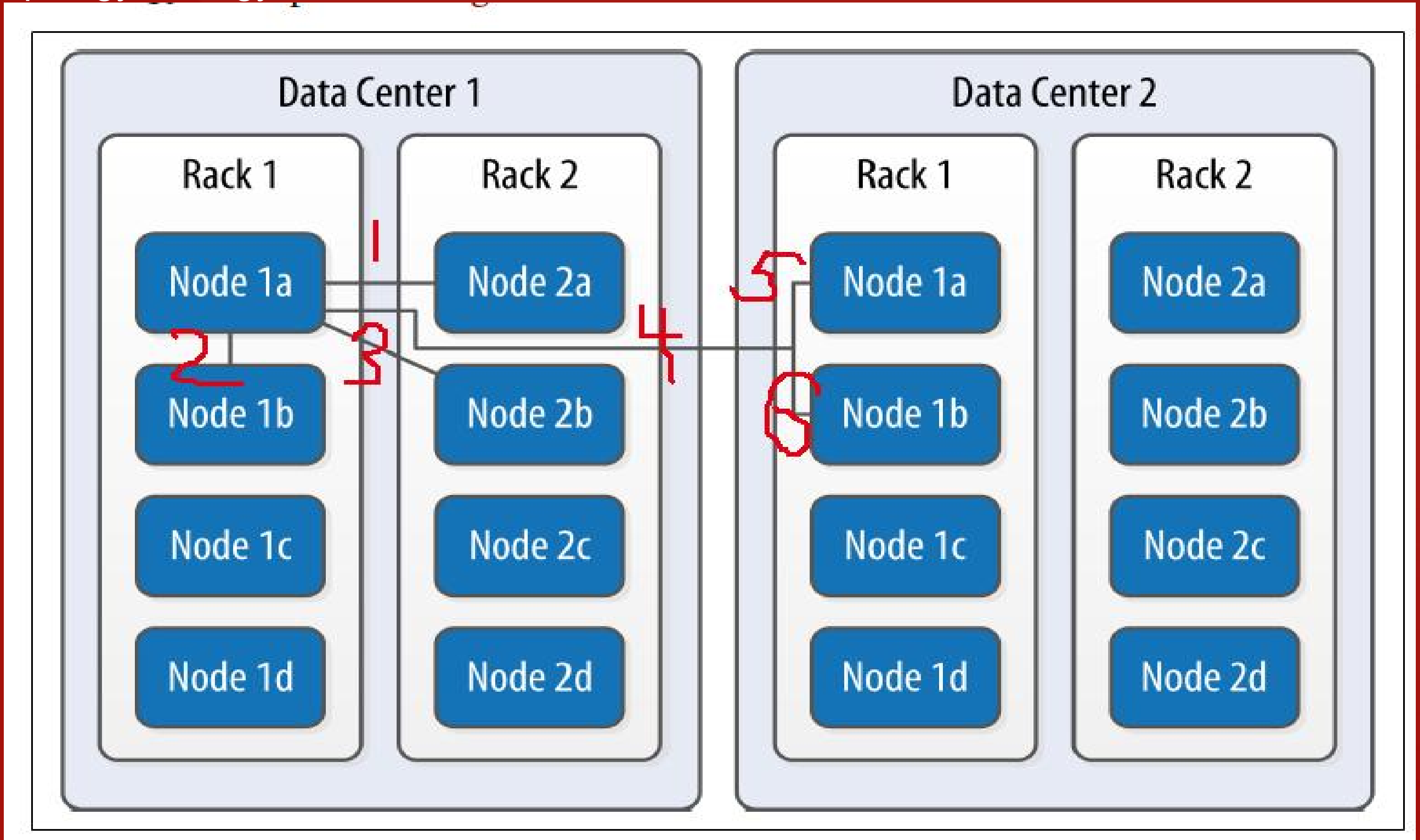

4
3
1
2
7
8
5
6
Data replicates in clock wise direction within datacenter
Data replicates in clock wise direction until reaches another datacenter

NetworkTopologyStrategy in Cassandra

3. 系统架构-复制

The content of the title

- NetworkTopologyStrategy



3. 系统架构--一致性

The content of the title

- 一致性：Tunable Consistency
  - ANY
  - ONE，TWO，THREE
  - LOCAL_ONE，QUORUM，LOCAL_QUORUM，EACH_QUORUM ALL
- Read Path
  - read-repair，Merkle Trees
- Write Path
  - Hinted Handoff(可用性)

3. 系统架构-线程模型SEDA

The content of the title

- SEDA：staged event-driven architecture

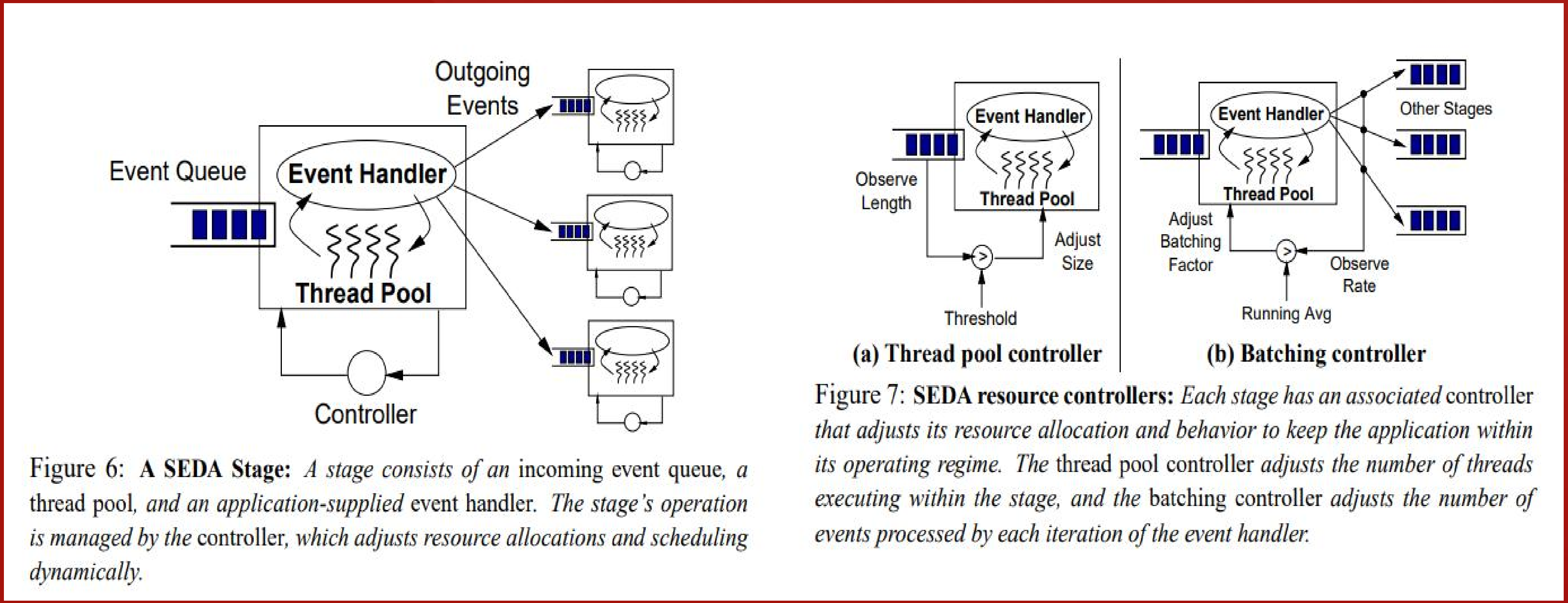


Figure 6: **A SEDA Stage:** *A stage consists of an* incoming event queue, *a* thread pool, *and an application-supplied* event handler. *The stage's operation is managed by the* controller, *which adjusts resource allocations and scheduling dynamically.*

(a) Thread pool controller

(b) Batching controller

Figure 7: **SEDA resource controllers:** *Each stage has an associated* controller *that adjusts its resource allocation and behavior to keep the application within its operating regime. The* thread pool controller *adjusts the number of threads executing within the stage, and the* batching controller *adjusts the number of events processed by each iteration of the event handler.*

3. 系统架构-线程模型SEDA

The content of the title

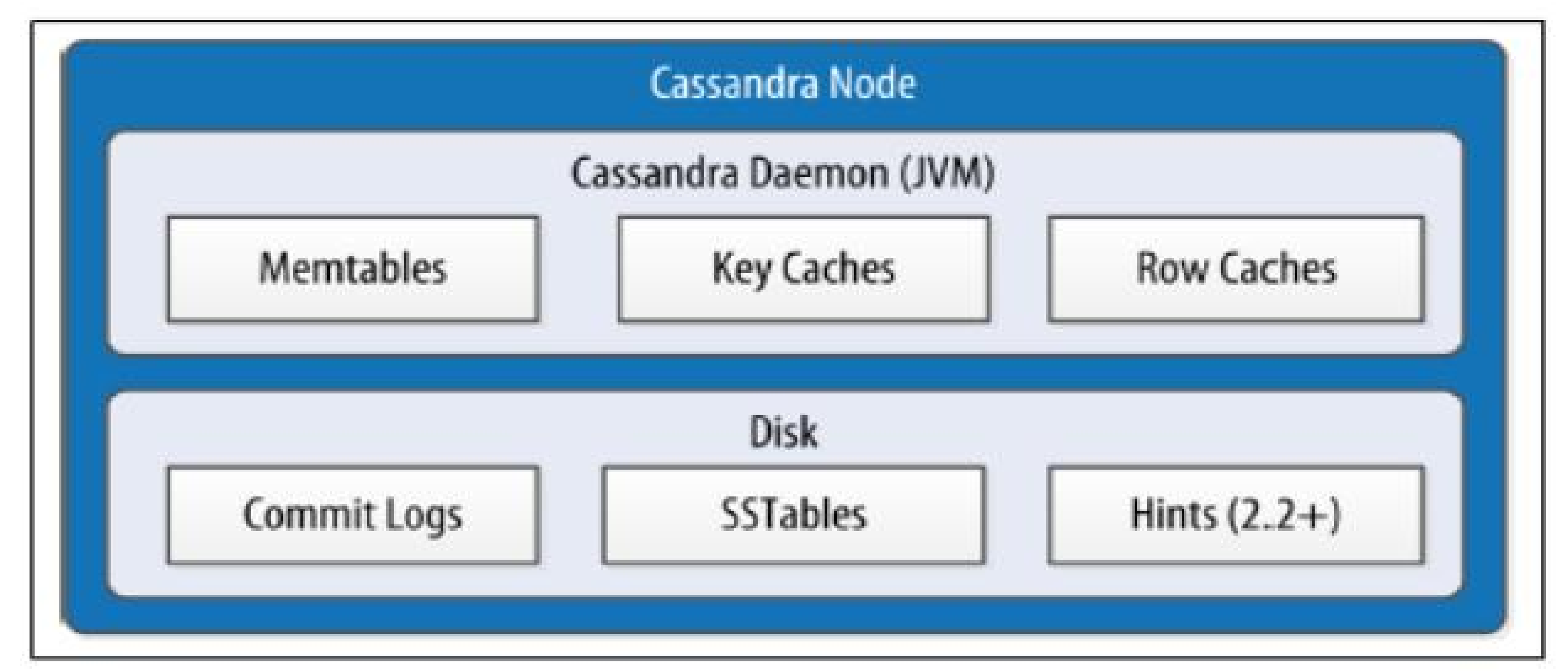
- Stage
  - 读，写
  - Gossip
  - 和其他节点交互
  - anti entropy
  - read repair
  - hinted handoff

- org.apache.cassandra.concurrent.Stage

3. 系统架构-单机存储引擎

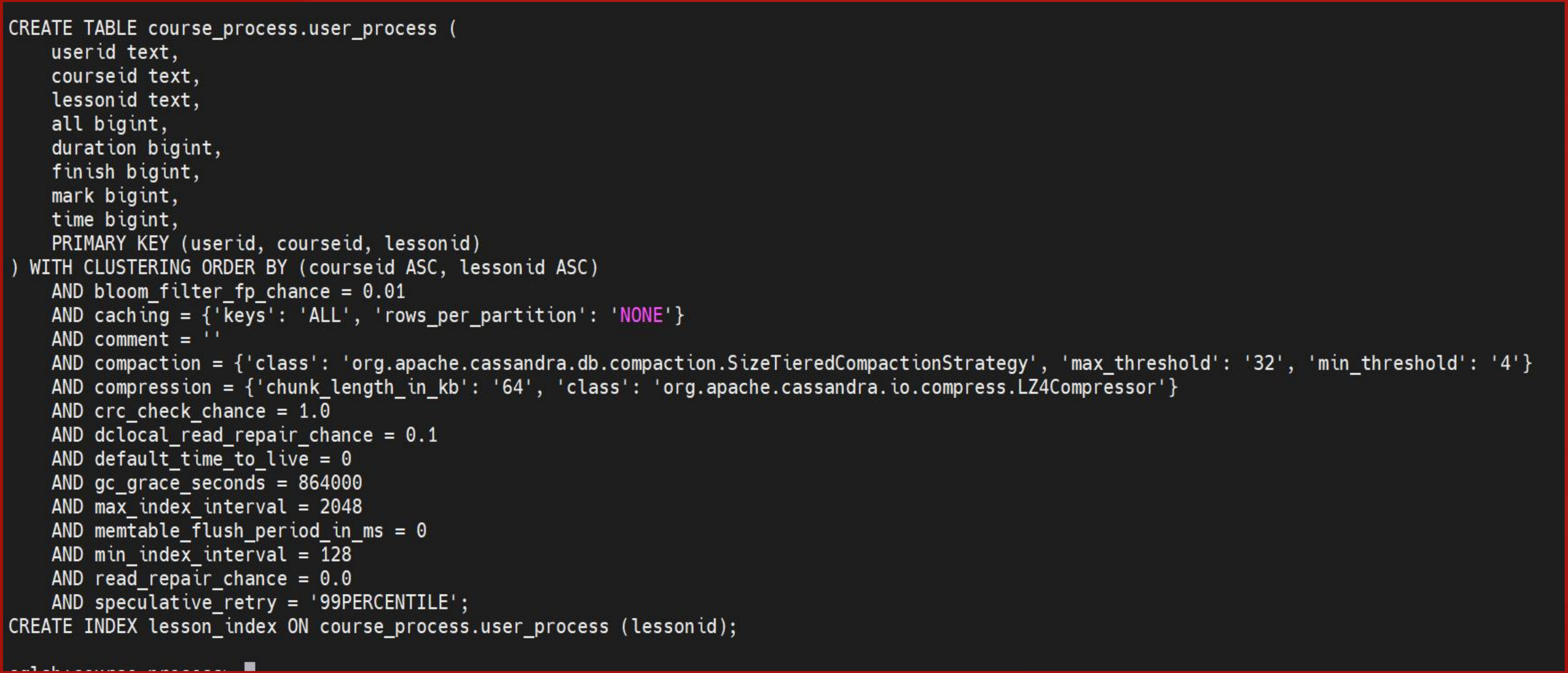
The content of the title

- Memtables, SSTables, and Commit Logs
- Compress
- Compact（读，写，时序）
  - STCS（写多读少），LCS（读多写少），DTCS



3. 系统架构-单机存储引擎案例

The content of the title



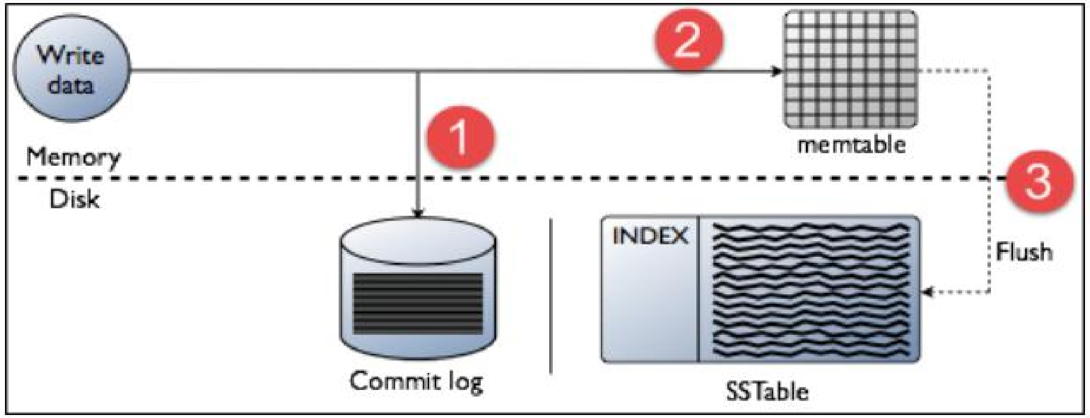
```
CREATE TABLE course_process.user_process (
    userid text,
    courseid text,
    lessonid text,
    all bigint,
    duration bigint,
    finish bigint,
    mark bigint,
    time bigint,
    PRIMARY KEY (userid, courseid, lessonid)
) WITH CLUSTERING ORDER BY (courseid ASC, lessonid ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
CREATE INDEX lesson_index ON course_process.user_process (lessonid);

cqlsh:course_process>
```

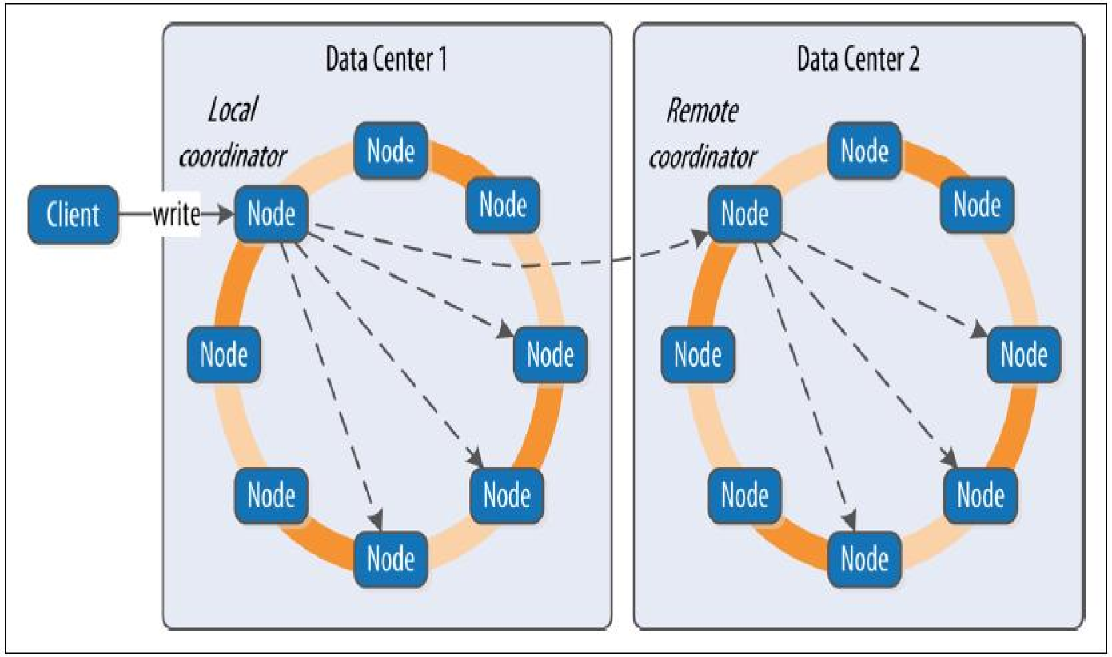
3. 系统架构-写路径

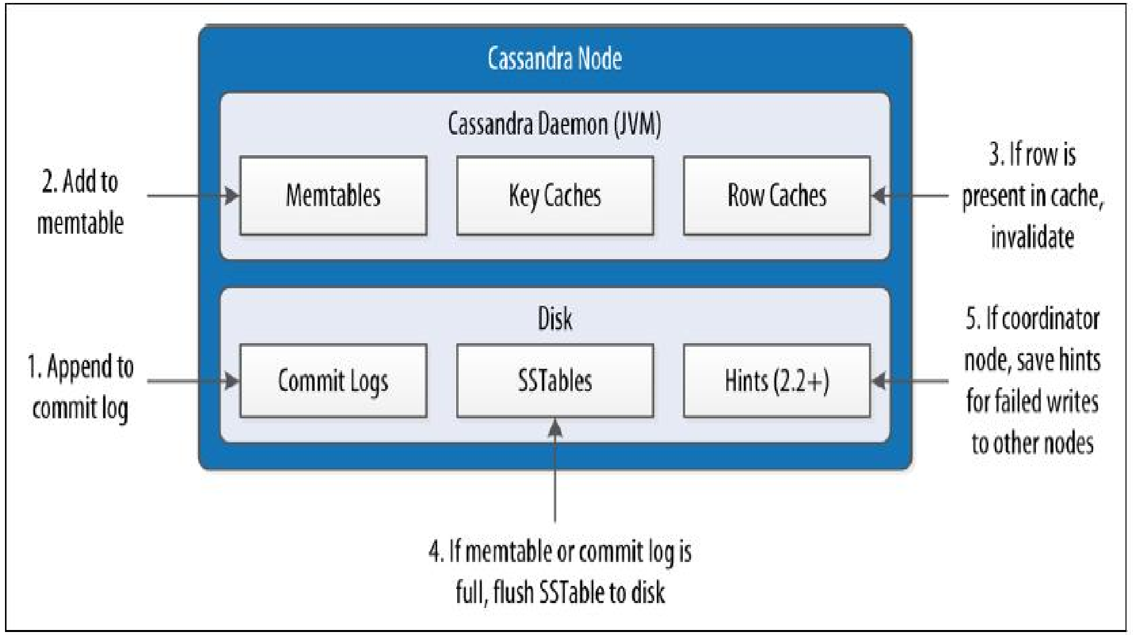
The content of the title

- 正常写
- 轻量级事务(条件插入，更新)
- 批处理
- 删除



Write Operation in Cassandra



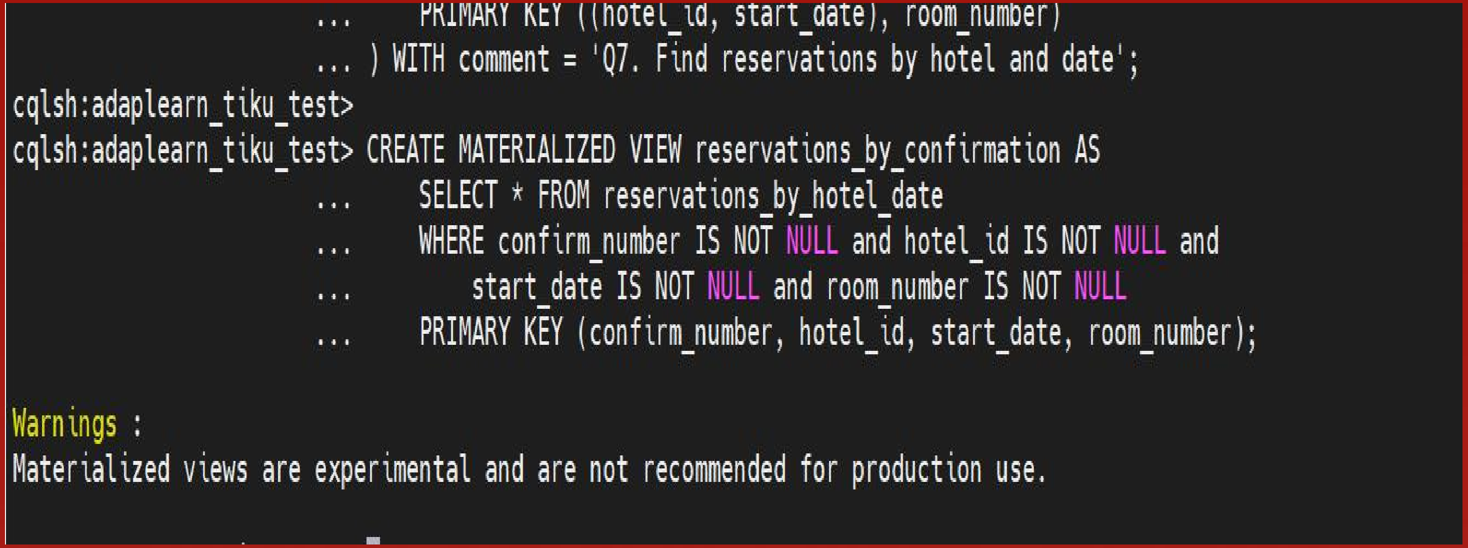


Cassandra Node
Cassandra Daemon (JVM)
Memtables
Key Caches
Row Caches
Disk
Commit Logs
SSTables
Hints (2.2+)
2. Add to memtable
1. Append to commit log
3. If row is present in cache, invalidate
5. If coordinator node, save hints for failed writes to other nodes
4. If memtable or commit log is full, flush SSTable to disk

3. 系统架构-读路径

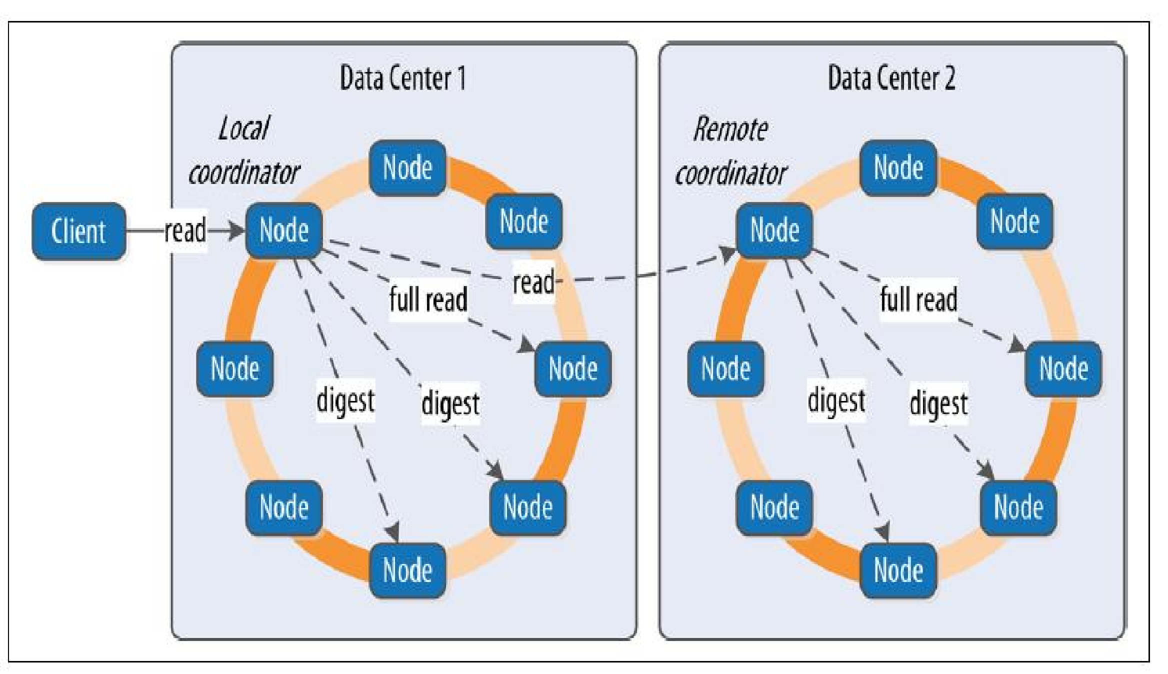
The content of the title

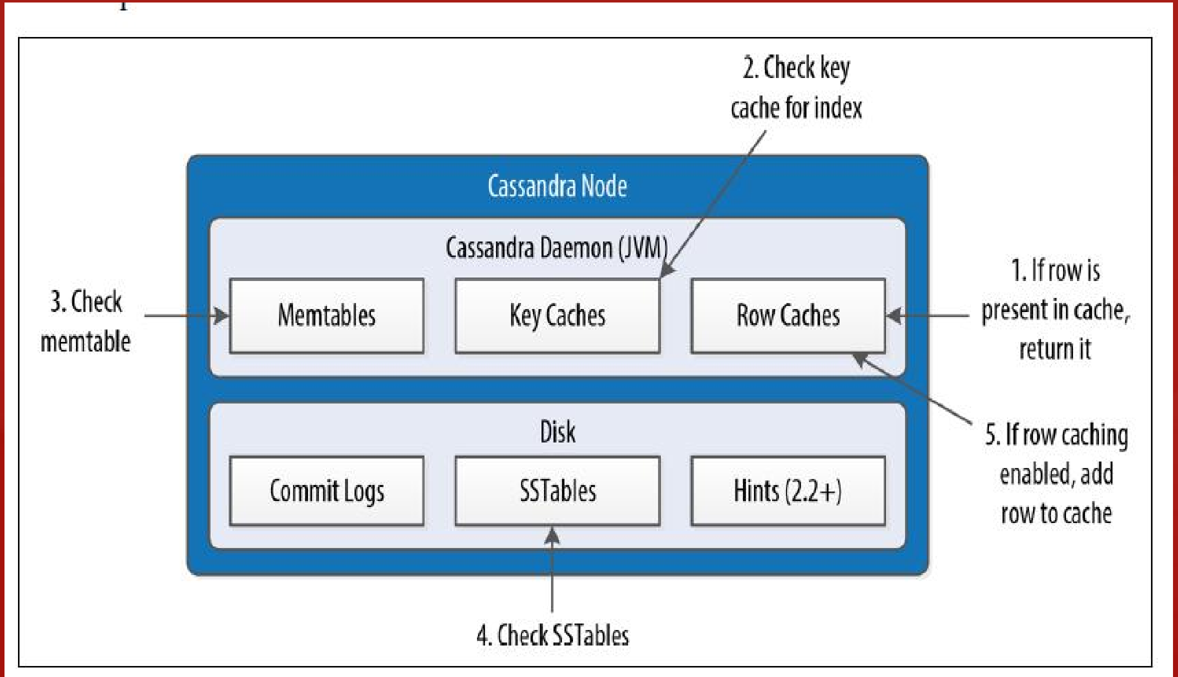
- 正常读/allow filtering查询
- 范围查询
- 排序，过滤，分页，UDF，UDA
- 二级索引，物化视图



```
                     ...     PRIMARY KEY ((hotel_id, start_date), room_number)
                     ... ) WITH comment = 'Q7. Find reservations by hotel and date';
cqlsh:adaplearn_tiku_test>
cqlsh:adaplearn_tiku_test> CREATE MATERIALIZED VIEW reservations_by_confirmation AS
                     ...     SELECT * FROM reservations_by_hotel_date
                     ...     WHERE confirm_number IS NOT NULL and hotel_id IS NOT NULL and
                     ...         start_date IS NOT NULL and room_number IS NOT NULL
                     ...     PRIMARY KEY (confirm_number, hotel_id, start_date, room_number);

Warnings :
Materialized views are experimental and are not recommended for production use.
```





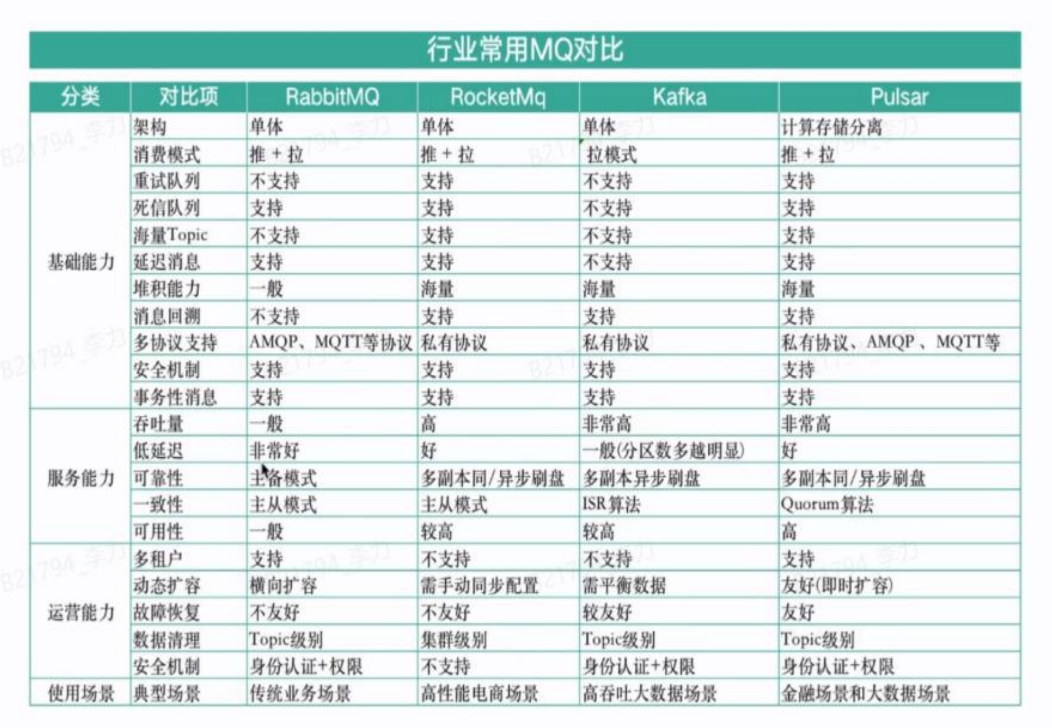
4. NoSQL存储选型

The content of the title

- 模型：属性占比法，决策树，案例分析法，成本收益分析法

- 如何比较存储？
  - 单机　　--->　最终会对标单机数据库
  - 分布式 --->　最终会对标CAP/PACELC

- 同类优先性能，不同优先功能
  - 性能提高不易，功能实现变易。



行业常用MQ对比

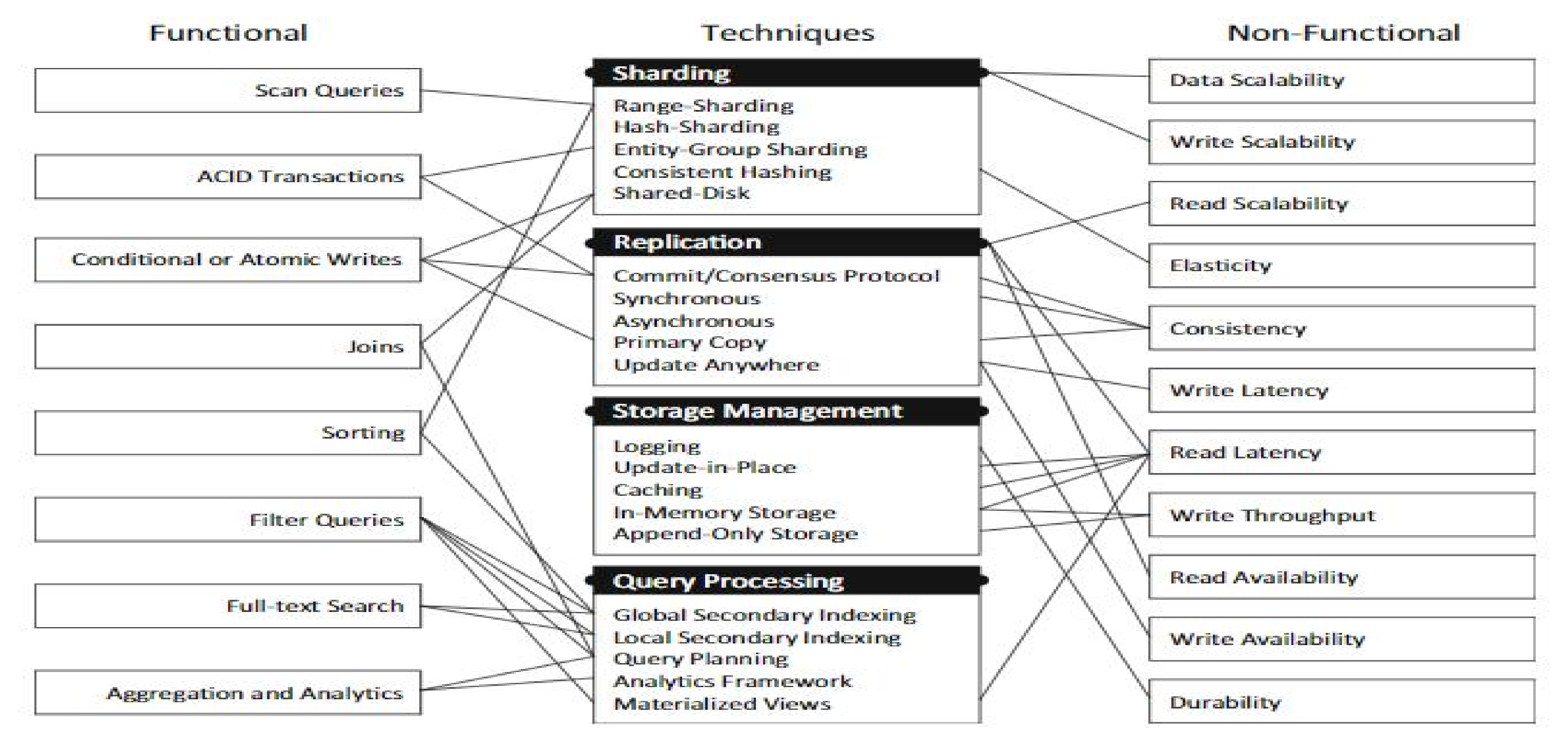
分类	对比项	RabbitMQ	RocketMq	Kafka	Pulsar
基础能力	架构	单体	单体	单体	计算存储分离
	消费模式	推 + 拉	推 + 拉	拉模式	推 + 拉
	重试队列	不支持	支持	不支持	支持
	死信队列	支持	支持	不支持	支持
	海量Topic	不支持	支持	不支持	支持
	延迟消息	支持	支持	不支持	支持
	堆积能力	一般	海量	海量	海量
	消息回溯	不支持	支持	支持	支持
	多协议支持	AMQP、MQTT等协议	私有协议	私有协议	私有协议、AMQP、MQTT等
	安全机制	支持	支持	支持	支持
	事务性消息	支持	支持	支持	支持
服务能力	吞吐量	一般	高	非常高	非常高
	低延迟	非常好	好	一般(分区数多越明显)	好
	可靠性	主备模式	多副本同/异步刷盘	多副本异步刷盘	多副本同/异步刷盘
	一致性	主从模式	主从模式	ISR算法	Quorum算法
	可用性	一般	较高	较高	高
运营能力	多租户	支持	不支持	不支持	支持
	动态扩容	横向扩容	需手动同步配置	需平衡数据	友好(即时扩容)
	故障恢复	不友好	不友好	较友好	友好
	数据清理	Topic级别	集群级别	Topic级别	Topic级别
	安全机制	身份认证+权限	不支持	身份认证+权限	身份认证+权限
使用场景	典型场景	传统业务场景	高性能电商场景	高吞吐大数据场景	金融场景和大数据场景

網易 NetEase

4. NoSQL存储选型

The content of the title

- 属性分析法-NoSQL Toolbox：你觉得哪些因素最重要？



Functional

Scan Queries

ACID Transactions

Conditional or Atomic Writes

Joins

Sorting

Filter Queries

Full-text Search

Aggregation and Analytics

Techniques

Sharding

Range-Sharding
Hash-Sharding
Entity-Group Sharding
Consistent Hashing
Shared-Disk

Replication

Commit/Consensus Protocol
Synchronous
Asynchronous
Primary Copy
Update Anywhere

Storage Management

Logging
Update-in-Place
Caching
In-Memory Storage
Append-Only Storage

Query Processing

Global Secondary Indexing
Local Secondary Indexing
Query Planning
Analytics Framework
Materialized Views

Non-Functional

Data Scalability

Write Scalability

Read Scalability

Elasticity

Consistency

Write Latency

Read Latency

Write Throughput

Read Availability

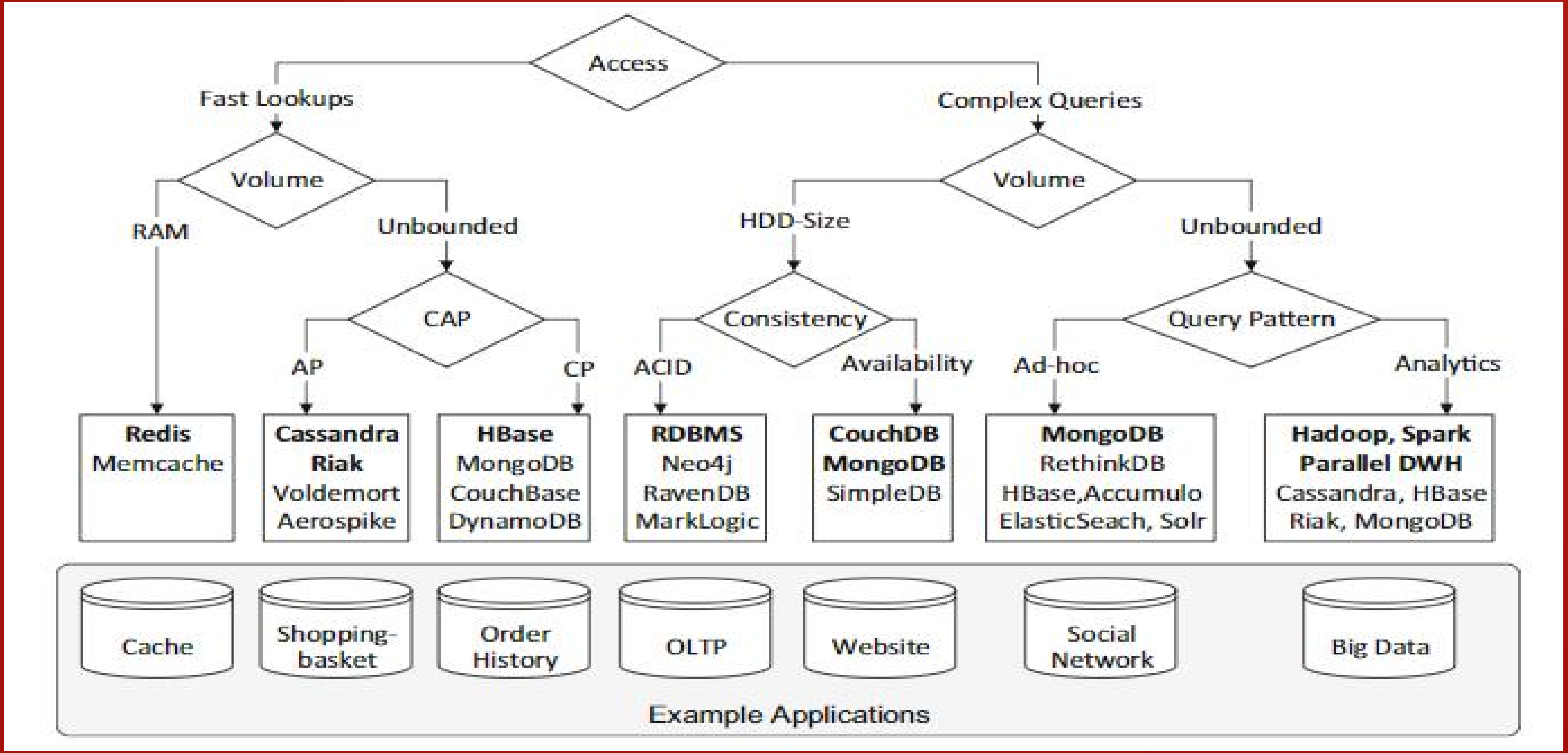
Write Availability

Durability

4. NoSQL存储选型

The content of the title

- 决策树模型：你觉得哪些选择很重要？



4. NoSQL存储选型

The content of the title

- 技术工具：https://dbdb.io/
    - 查找竞品
        - Hazelcast竞品
    - 研究学习
        - 学习Java，MVCC，嵌入式，开源
    - 根据特性查找
        - MapDB

5. 参考资料

The content of the title

- 项目
  - https://github.com/xiaozhiliaoo/cassandra-practice
- 书
  - 《Cassandra权威指南-第二和三版》
  - 《数据库系统内幕》 第7,9,11,12章
  - 《数据密集型应用系统设计》第2,3,5,6,9章
  - 《云计算与分布式系统》第8章
- 论文
  - Dynamo/Bigtable
  - 原始论文：Cassandra - A Decentralized Structured Storage System
  - 线程模型：SEDA: An Architecture for Well-Conditioned,Scalable Internet Services
  - 通信协议：Efficient Reconciliation and Flow Control for Anti-Entropy Protocols
  - 数据建模：A Big Data Modeling Methodology for Apache Cassandra
  - NoSQL选型：NoSQL database systems: a survey and decision guidance

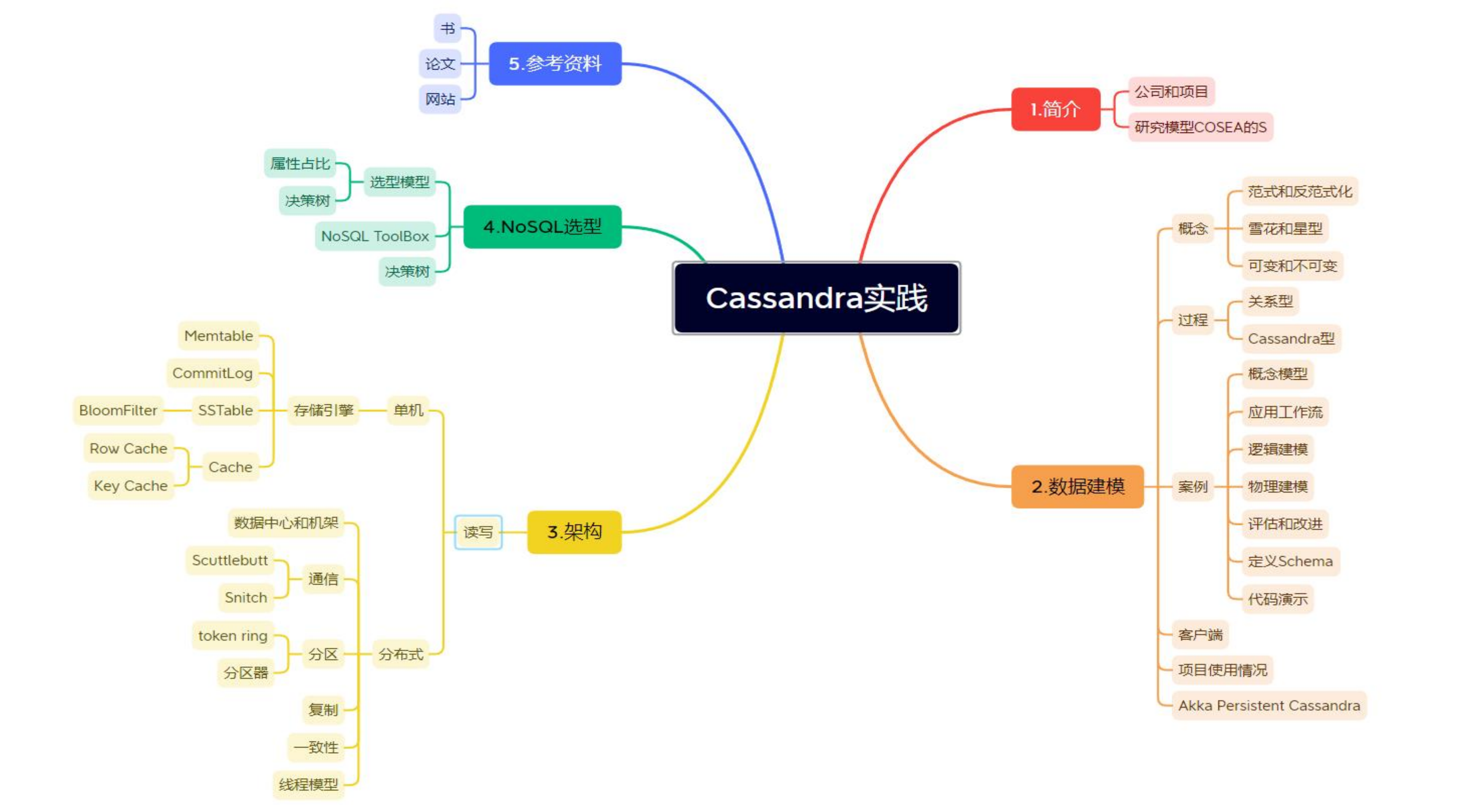
6. 回顾

The content of the title

- 分布式系统泛型
  - 体系结构：非集中式（去中心化）
  - 进程：SEDA
  - 通信：多播-Gossip(逆熵，Scuttlebutt), Snitch，Information【Vector clock(generation(monotonic timestamp), version(logical clock)】
  - 命名：无层次命名-DHT
  - 协调：时钟同步，Last-Write-Wins-Element-Set，Paxos-LWT
  - 一致性和复制：可调一致性，逆熵，修复，一致性Hash复制
  - 容错性：FD故障检测器
  - 安全：认证，授权，加密，SSL，TLS和证书

6. 回顾

The content of the title



- Cassandra实践
  - 1.简介
    - 公司和项目
    - 研究模型COSEA的S
  - 2.数据建模
    - 概念
      - 范式和反范式化
      - 雪花和星型
      - 可变和不可变
    - 过程
      - 关系型
      - Cassandra型
    - 案例
      - 概念模型
      - 应用工作流
      - 逻辑建模
      - 物理建模
      - 评估和改进
      - 定义Schema
      - 代码演示
    - 客户端
    - 项目使用情况
    - Akka Persistent Cassandra
  - 3.架构
    - 读写
      - 单机
        - 存储引擎
          - Memtable
          - CommitLog
          - SSTable
            - BloomFilter
          - Cache
            - Row Cache
            - Key Cache
      - 分布式
        - 数据中心和机架
        - 通信
          - Scuttlebutt
          - Snitch
        - 分区
          - token ring
          - 分区器
        - 复制
        - 一致性
        - 线程模型
  - 4.NoSQL选型
    - 选型模型
      - 属性占比
      - 决策树
    - NoSQL ToolBox
    - 决策树
  - 5.参考资料
    - 书
    - 论文
    - 网站

Q&A

網易 NetEase

THANK YOU

網易 NetEase