高级数据库技术

一、传统数据库

1、集中

2、关系（关系代数）主流

关系代数：

A	B	C
a	b	c
d	a	f
c	b	d

(A) 关系R

D	E	F
b	g	a
d	a	f

(B) 关系S

(1.) 并运算

白话: R 和 S 关系合一起, 相同的不写

a	b	c
d	a	f
c	b	d
b	g	a

R∪S

(2.) 差操作

白话: 因为是 R-S, 找 R 在 S 关系中没有的

a	b	c
c	b	d

R−S

(3.) 笛卡尔乘积

设 R 是 n 元关系，S 是 m 元关系，R 和 S 的的笛卡尔积定义为

R×S={(r1,⋯,rn,s1,⋯,sm)|(r1,⋯, rn)∈R∧(s1, ⋯,sm)∈S}。

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

R×S

(4.) 投影操作

白话: 看横行, 如果有两个横行相同, 只写一个. 因此若是 S 关系投影操作的话, 也就是有 b 和 a

A	B	C
a	b	c
d	a	f
c	b	d

(A) 关系R

A	C
a	c
d	f
c	d

$\Pi_{A,C}(R)$

(5.) 选择操作

白话: 把符合条件的拿出来

A	B	C
a	b	c
d	a	f
c	b	d

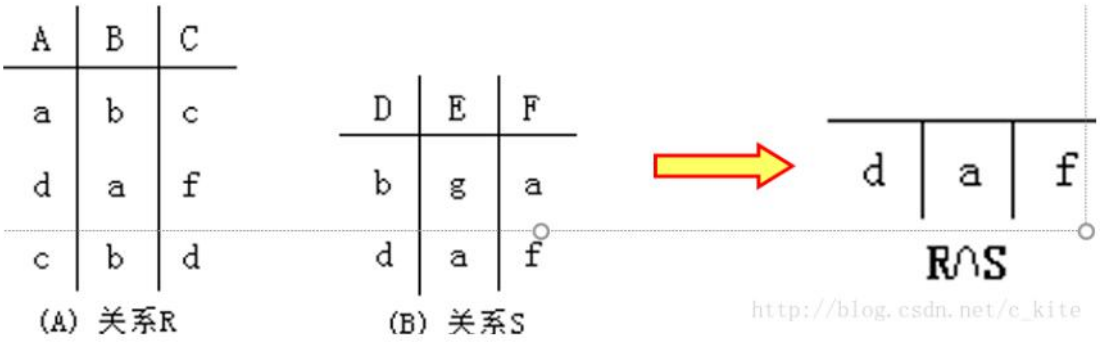
(A) 关系R

A	B	C
a	b	c
c	b	d

$\sigma_{B=b}(R)$

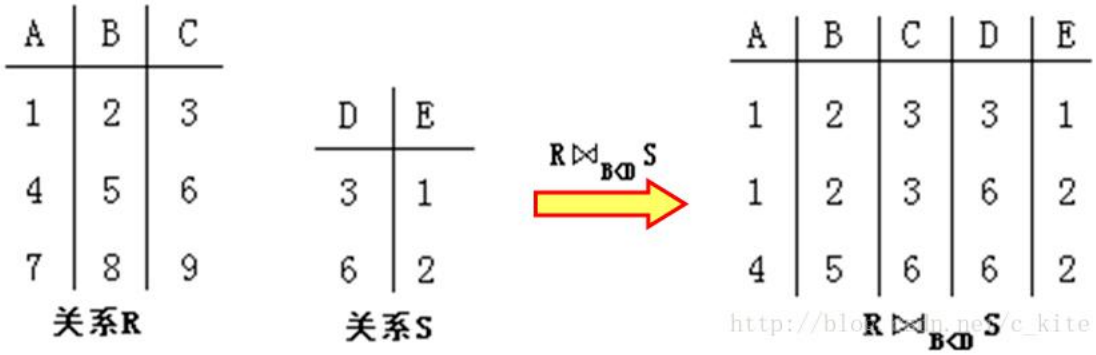
(6.) 交操作

白话: 相同的拿出来



(7.) 连接操作

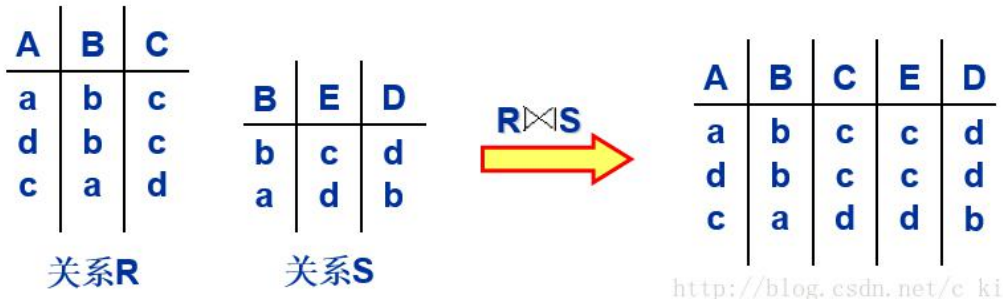
白话: 两个关系先做笛卡尔积运算, 然后再根据条件进行比对. 留下符合条件的例子:



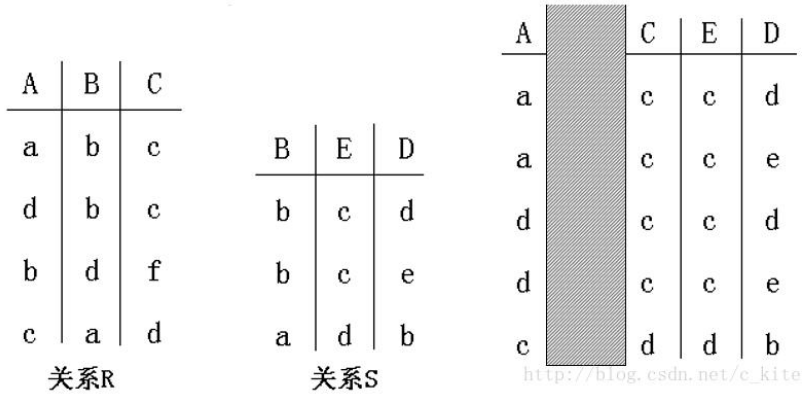
(7. 1) 几个特殊的连接操作

①自然连接 设 Att(R)和 Att(S)分别是 R 和 S 的属性集合。连接条件为 R.B=S.B，连接的结果关系的属性集合为 Att(R)∪(Att(S)-{B})，即 B 在结果关系中只出现一次。称这样的连接操作为自然连接操作，

白话: 找相同的然后拼在一起, 例如 B 属性, 看看下面的例子;



②复合连接 类似于自然连接，只是连接结果不包含连接属性。



A	B	C
a	b	c
d	b	c
b	d	f
c	a	d

关系R

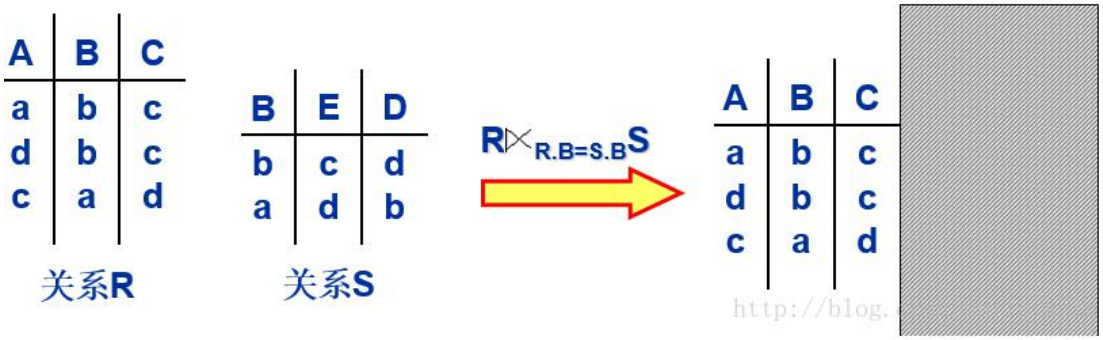
B	E	D
b	c	d
b	c	e
a	d	b

关系S

A	C	E	D
a	c	c	d
a	c	c	e
d	c	c	d
d	c	c	e
c	d	d	b

③半连接

白话: 下面的例子由于是 R 半链接 S, 则因此拿 R 去和 S 所比较



A	B	C
a	b	c
d	b	c
c	a	d

关系R

B	E	D
b	c	d
a	d	b

关系S

$R \ltimes_{R.B=S.B} S$

A	B	C
a	b	c
d	b	c
c	a	d

二、高级数据库技术

1、分布式数据库

分布式数据库系统(DDBS)包含分布式数据库管理系统(DDBMS)和分布式数据库（DDB)。在分布式数据库系统中，一个应用程序可以对数据库进行透明操作，数据库中的数据分别在不同的局部数据库中存储、由不同的 DBMS 进行管理、在不同的机器上运行、由不同的操作系统支持、被不同的通信网络连接在一起。

分布式数据库系统具有下列优点：

(1)更适合分布式的管理与控制。

(2)具有灵活的体系结构。

(3)系统经济，可靠性高，可用性好。

(4)在一定条件下响应速度加快。如果存取的数据在本地数据库中，那末就可以由用户所在的计算机来执行，速度就快。

(5)可扩展性好，易于集成现有系统，也易于扩充。

分布数数据库系统有如下劣势：
(1) 通信开销较大，故障率高。
(2) 数据的存取结构复杂。
(3) 数据的安全性和保密性较难控制。

（1）关系型数据库
定义：关系数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。简单说，关系型数据库是由多张能互相联接的二维行列表格组成的数据库。关系模型就是指二维表格模型,因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据组织。当前主流的关系型数据库有 Oracle、DB2、PostgreSQL、Microsoft SQL Server、Microsoft Access、MySQL。

类型	特性	优点	缺点
关系型数据库 SQLite、Oracle、mysql	1、关系型数据库，是指采用了关系模型来组织数据的数据库； 2、关系型数据库的最大特点就是事务的一致性； 3、简单来说，关系模型指的就是二维表格模型，而一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。	1、容易理解：二维表结构是非常贴近逻辑世界一个概念，关系模型相对网状、层次等其他模型来说更容易理解； 2、使用方便：通用的SQL语言使得操作关系型数据库非常方便； 3、易于维护：丰富的完整性(实体完整性、参照完整性和用户定义的完整性)大大减低了数据冗余和数据不一致的概率； 4、支持SQL，可用于复杂的查询。	1、为了维护一致性所付出的巨大代价就是其读写性能比较差； 2、固定的表结构； 3、高并发读写需求； 4、海量数据的高效率读写；

非关系型数据库 MongoDb、redis、HBase	1、使用键值对存储数据； 2、分布式； 3、一般不支持ACID特性； 4、非关系型数据库严格上不是一种数据库，应该是一种数据结构化存储方法的集合。	1、无需经过sql层的解析，读写性能很高； 2、基于键值对，数据没有耦合性，容易扩展； 3、存储数据的格式：nosql的存储格式是key,value形式、文档形式、图片形式等等，文档形式、图片形式等等，而关系型数据库则只支持基础类型。	1、不提供sql支持，学习和使用成本较高； 2、无事务处理，附加功能bi和报表等支持也不好；

（2）非关系（Nosql）

分类：

① 键值(Key-Value)存储数据库：

这一类数据库主要会使用到一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。Key/value 模型对于 IT 系统来说的优势在于简单、易部署。但是如果 DBA 只对部分值进行查询或更新的时候，Key/value 就显得效率低下了。举例如：Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB.

② 列存储数据库：

这部分数据库通常是用来应对分布式存储的海量数据。键仍然存在，但是它们的特点是指向了多个列。这些列是由列家族来安排的。如：Cassandra, HBase, Riak.

③ 文档型数据库

文档型数据库的灵感是来自于 Lotus Notes 办公软件的，而且它同第一种键值存储相类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如 JSON。文档型数据库可 以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。如：CouchDB, MongoDb. 国内也有文档型数据库 SequoiaDB，已经开源。

④ 图形(Graph)数据库

图形结构的数据库同其他行列以及刚性结构的 SQL 数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL 数据库没有标准的查询语言(SQL)，因此进行数据库查询需要制定数据模型。许多 NoSQL 数据库都有 REST 式的数据接口或者查询 API。如：Neo4J, InfoGrid, Infinite Graph.

因此，我们总结 NoSQL 数据库在以下的这几种情况下比较适用：1、数据模型比较简单；2、需要灵活性更强的 IT 系统；3、对数据库性能要求较高；4、不需要高度的数据一致性；5、对于给定 key，比较容易映射复杂值的环境。

分类	Examples举例	典型应用场景	数据模型	优点	缺点
键值（key-value）	Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB	内容缓存，主要用于处理大量数据的高访问负载，也用于一些日志系统等等。	Key 指向 Value 的键值对，通常用hash table来实现	查找速度快	数据无结构化，通常只被当作字符串或者二进制数据
列存储数据库	Cassandra, HBase, Riak	分布式的文件系统	以列簇式存储，将同一列数据存在一起	查找速度快，可扩展性强，更容易进行分布式扩展	功能相对局限
文档型数据库	CouchDB, MongoDb	Web应用（与Key-Value类似，Value是结构化的，不同的是数据库能够了解Value的内容）	Key-Value对应的键值对，Value为结构化数据	数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构	查询性能不高，而且缺乏统一的查询语法。
图形(Graph)数据库	Neo4J, InfoGrid, Infinite Graph	社交网络，推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址，N度关系查找等	很多时候需要对整个图做计算才能得出需要的信息，而且这种结构不太好做分布式的集群方案。

关系数据库面临下列挑战：

①并不适用于数据类型多样化（比如图像、视频和文本）的大量（PB 级）数据。

②无法扩展、支持庞大的数据量。

③无法纵向扩展，受制于内存和处理器的功能。

④无法横向扩展，受制于依赖缓存的读取和写入操作。

⑤分片（将数据库分成几个部分，存储在不同的节点）引起操作问题（比如管理共享式故障）。

⑥复杂的 RDBMS 模型

⑦一致性限制了 RDBMS 的可扩展性。

① Newsql

定义：NewSQL 是对各种新的可扩展/高性能数据库的简称，这类数据库不仅具有 NoSQL 对海量数据的存储管理能力，还保持了传统数据库支持 ACID 和 SQL 等特性。NewSQL 是一种相对较新的形式，旨在使用现有的编程语言和以前不可用的技术来结合 SQL 和 NoSQL 中最好的部分。 NewSQL 目标是将 SQL 的 ACID 保证与 NoSQL 的可扩展性和高性能相结合。

② Schemaless

NoSQL 数据库提供了下列解决方案，从而克服了关系模型无法克服的挑战：

1. 横向扩展、什么都不共享的架构，能够在大量节点上运行。
2. 非锁定并发性控制机制，那样实时读取不会与写入产生冲突。
3. 可扩展的复制和分发－－成千上万个机器拥有分布式数据。
4. 每个节点提供的性能高于 RDBMS 的架构。
5. 无模式（schema-less）数据模型。

2、面向分布式的技术：分布式架构、分片与复制、分布式查询优化、分布式事物处理、故障恢复等

3、ACID 到 CPA（属性牺牲）、可扩展性、速度

ACID:

事务有四个特性，也就是经常被提到的 ACID：

原子性(Atomicity):所谓的原子性就是说，在整个事务中的所有操作，要么全部完成，要么全部不做，没有中间状态。对于事务在执行中发生错误，所有的操作都会被回滚，整个事务就像从没被执行过一样。

一致性(Consistency):事务的执行必须保证系统的一致性隔离性(Isolation):所谓的隔离性就是说，事务与事务之间不会互相影响，一个事务的中间状态不会被其他事务感知。

持久性(Durability):所谓的持久性，就是说一单事务完成了，那么事务对数据所做的变更就完全保存在了数据库中，即使发生停电，系统宕机也是如此。

CPA 指的是，在一个分布式系统中，一致性(C)、可用性(A)、分区容错性(P)，三者不可兼得。CPA 是 NoSQL 数据库的基石：

一致性：在分布式系统中的所有数据备份，在同一时刻是否同样的值。（等同于所有节点访问同一份最新的数据副本）

可用性：在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。（对数据更新具备高可用性）

分区容错性：以实际效果而言，分区相当于对通信的时限要求。系统如果不能在时限内达成数据一致性，就意味着发生了分区的情况，必须就当前操作在 C 和 A 之间做出选择。

CAP 理论就是说在分布式存储系统中，最多只能实现上面的两点。而由于当前的网络硬件肯定会出现延迟丢包等问题，所以分区容忍性是我们必须需要实现的。所以我们只能在一致性和可用性之间进行权衡，没有 NoSQL 系统能同时保证这三点。

CAP 理论告诉我们：一个分布式系统不可能同时满足一致性(C:Consistency)、可用性(A:Availability)、分区容错性(P:Partitiontolerance)这三个基本需求，并且最多只能满足其中的两项。

对于一个分布式系统来说，分区容错是基本需求，否则不能称之为分布式系统。因此架构师需要在 C 和 A 之间寻求平衡。

C – Consistency – 一致性（与 ACID 的 C 完全不同）

一致性是指“all nodes see the same data at the same time”，即更新操作成功并返回客户端完成后，所有节点在同一时间的数据完全一致。

对于一致性，可以分为从客户端和服务端两个不同的视角。

从客户端来看，一致性主要指的是多并发访问时更新过的数据如何获取的问题。从服务端来看，则是更新如何复制分布到整个系统，以保证数据最终一致。一致性是因为有并发读写才有的问题，因此在理解一致性的问题时，一定要注意结合考虑并发读写的场景。

从客户端角度，多进程并发访问时，更新过的数据在不同进程如何获取的不同策略，决定了不同的一致性。对于关系型数据库，要求更新过的数据能被后续的访问都能看到，这是强一致性。如果能容忍后续的部分或者全部访问不到，则是弱一致性。如果经过一段时间后要求能访问到更新后的数据，则是最终一致性。

A – Availability – 可用性

可用性是指“Reads and writes always succeed”，即服务一直可用，而且是正常响应时间。

对于一个可用性的分布式系统，每一个非故障的节点必须对每一个请求作出响应。也就是说，该系统使用的任何算法必须最终终止。当同时要求分区容忍性时，这是一个很强的定义：即使是严重的网络错误，每个请求必须完成。

好的可用性主要是指系统能够很好的为用户服务，不出现用户操作失败或者访问超时等用户体验不好的情况。在通常情况下，可用性与分布式数据冗余、负载均衡等有着很大的关联。

P – Partition tolerance – 分区容错性

分区容错性是指“the system continues to operate despite arbitrary message loss or failureof part of the system”，即分布式系统在遇到某节点或网络分区故障的时候，仍然能够对外提供满足一致性和可用性的服务。

分区容错性和扩展性紧密相关。在分布式应用中，可能因为一些分布式的原因导致系统无法正常运转。好的分区容错性要求能够使应用虽然是一个分布式系统，但看上去却好像是一个可以运转正常的整体。比如现在的分布式系统中有某一个或者几个机器宕掉了，其它剩下的机器还能够正常运转满足系统需求，或者是机器之间有网络异常，将分布式系统分隔成未独立的几个部分，各个部分还能维持分布式系统的运作，这样就具有好的分区容错性。

CA without P

如果不要求 P（不允许分区），则 C（强一致性）和 A（可用性）是可以保证的。但其实分区不是你想不想的问题，而是始终会存在，因此 CA 的系统更多的是允许分区后各子系统依然保持 CA。

CP without A

如果不要求 A（可用），相当于每个请求都需要在 Server 之间强一致，而 P（分区）会导致同步时间无限延长，如此 CP 也是可以保证的。很多传统的数据库分布式事务都属于这种模式。

AP without C

要高可用并允许分区，则需放弃一致性。一旦分区发生，节点之间可能会失去联系，为了高可用，每个节点只能用本地数据提供服务，而这样会导致全局数据的不一致性。现在众多的 NoSQL 都属于此类。

CAP 理论定义了分布式存储的根本问题，但并没有指出一致性和可用性之间到底应该如何权衡。于是出现了 BASE 理论，给出了权衡 A 与 C 的一种可行方案。

具体内容

一、绪论与设计概述【流程，概念设计】（非重点）逻辑方面不考虑事情分布性

二、分布式数据库（概念、特点、架构等，为何分布？）【分析：数据本身、数据处理可扩展性…】

（1）概念：PPT p8

（2）特点：PPT p9

（3）架构：PPT p11

（4）产生原因：课本 p250

三、分布式数据库的分片与复制（分片【目的】、垂直分片【交集为主码、应用高度相关，最好不用再连接】、水平分片【交集为空、记录必须在一个分片里、各分片存取频度相似】、复制【读快、更新麻烦、存储代价、多目标优化】【启发式分配方法】…）

1、分片目的 p274

2、垂直分片 p277

3、水平分片 p275

4、复制（数据分配）：课本 p279 PPT p24

四、分布式数据库的查询优化（SQL 语言、关系代数表示、SQL 与关系代数的等价描述、构建查询树、优化【分片、复制、连接（半连接、直接连接）】）、模式（4 层）、数据分布透明性、站点配置【全局到分布】

1、SQL 与关系代数的等价描述 p34、p35

2、构建查询树 p36

3、优化 p299~p310

4、四层 p38 p295

5、数据分布透明性 p265

6、站点配置

五、分布式数据库事务管理与恢复（分布式事务提交、日志、检查点、redo&undo）

1、分布式事务提交 p57

2、日志 p322 p55

3、检查点 p56

4、Redo and undo p56

六、数据库并发控制[在并发性能和可串行化之间做的权衡和妥协]
（封锁技术【主站点、主副本、两阶段锁及实现】、时标法、多版本【基于时标、基于锁】）、MVCC【回滚】
1、封锁技术 p65
2、时标法 p72
3、多版本【MVCC】 p75

七、数据库实例研究
Nosql、newsql、特点与趋势
1、nosql 特点
非关系型的、分布式的、开源的、水平可扩展的。最初的目的是为了大规模 web 应用。NoSQL 的拥护者们提倡运用非关系型的数据存储，通常的应用如下特点：模式自由、支持简易复制、简单的 API、最终的一致性（非 ACID）、大容量数据等。
优点缺点

1.易扩展 2.高性能 3.数据类型灵活 4.高可用 1.没有标准 2.没有存储过程 3.不支持 sql4.功能不够完善

（1）优点
易扩展：
NoSQL 数据库种类繁多，但是有一个共同的特点，都是去掉了关系型数据库的关系型特性。数据之间无关系，这样就非常容易扩展。也无形之间，在架构的层面上带来了可扩展的能力。
大数据量，高性能
NoSQL 数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。一般 MySQL 使用 Query Cache，每次表更新 Cache 就失效，是一种大粒度的 Cache，针对 web2.0 的交互频繁的应用，Cache 性能不高。而 NoSQL 的 Cache 是记录级的，是一种细粒度的 Cache，所以 NoSQL 在这个层面上来说性能就要高很多了。

灵活的数据模型：
NoSQL 无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。而在关系型数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段简直就是一个噩梦。这点在大数据量的 web2.0 时代尤其明显。

高可用：
NoSQL 在不太影响性能的情况下，就可以方便地实现高可用的架构。比如 Cassandra、HBase 模型，通过复制模型也能实现高可用。

（2）缺点
① 没有标准：没有对 NoSQL 数据库定义的标准，所以没有两个 NoSQL 数据库是平等的。
② 没有存储过程：NoSQL 数据库中大多没有存储过程。
③ 不支持 SQL：NoSQL 大多不提供对 SQL 的支持：如果不支持 SQL 这样的工业标准，将会对用户产生一定的学习和应用迁移上的成本。
④ 支持的特性不够丰富，产品不够成熟：现有产品所提供的功能都比较有限，不像 MS SQL Server 和 Oracle 那样能提供各种附加功能，比如 BI 和报表等。大多数产品都还处于初创期，和关系型数据库几十年的完善不可同日而语。

2、Newsql 特点
（1）NEWSQL 的定义
针对 OLTP 的读写，提供与 NOSQL 相同的可扩展性和性能，同时能支持满足 ACID 特性的事务。即保持 NOSQL 的高可扩展和高性能，并且保持关系模型。
（2）NEWSQL 的优点
轻松的获得可扩展性
能够使用关系模型和事务，应用逻辑会简化很多
注意，此篇论文中的 NEWSQL 偏向于 OLTP 型数据库，和一些 OLAP 类型的数据库不同，OLAP 数据库更偏向于复杂的只读查询，查询时间往往很长。
（3）NEWSQL 的特性
执行时间短
一般只查询一小部分数据，通过使用索引来达到高效查询的目的
一般执行相同的命令，使用不同的输入参数
采取新架构的全新数据库平台
高度优化的 SQL 引擎
带透明分片的中间层件系统

3、数据库发展趋势

HBase：

宽列式数据库，基于 Apache Hadoop 和 BigTable 的概念。

Apache HBase 是一种 NoSQL 键/值存储系统，它在 Hadoop 分布式文件系统（HDFS）上运行。不像 Hive，HBase 操作在数据库上，而不是 MapReduce 作业上实时运行。HBase 分成表，表又细分成列族（column family）。列族必须在模式中加以声明，它将某一组列（列不需要模式定义）分为小组。比如说，“message”列族可能包括以下这几列：“to”、“from”、“date”、“subject”和“body”。HBase 中的每个键/值对被定义为一个单元（cell），每个键含有行键、列族和时间戳。HBase 中的行是一组键/值映射，由行键来识别。HBase 可以使用 Hadoop 的基础设施，并使用现成服务器实现横向扩展。

HBase 的工作方式是，将数据存储为键/值。它支持四种主要的操作：添加或更新行的 put，检索一组单元的 scan，返回某个指定行的单元的 get，以及从表上删除行、列或列版本的 delete。拥有版本控制功能，那样可以获取数据的之前值（历史记录可以通过 HBase 压缩时不时删除，以释放空间）。虽然 HBase 包括表，但只有表和列族才需要模式，列不需要模式，它还包括增量/计数器功能。

HBase 查询用一种需要学习的自定义语言来编写。可以通过 Apache Phoenix，获得类似 SQL 的功能，不过其代价是需要维护模式。此外，HBase 并不完全符合 ACID，不过它确实支持某些属性。最后但并非最不重要的是，为了运行 HBase，就需要 ZooKeeper——这是面向分布式协调的服务器，比如配置、维护和命名。

HBase 最适合大数据的实时查询。Facebook 将它用于消息传递和实时分析。Facebook 甚至将它用于计数 Facebook 点赞。

Hbase 有集中式架构， Master 服务器负责监控集群中的所有 RegionServer（负责服务和管理区域）实例，它也是查看所有元数据变化的界面。它提供了 CAP 原理中的 CP（一致性和可用性）。

HBase 针对读取操作进行了优化，得到单次写入 master 的支持，支持因而获得的严格一致性模型，以及使用支持行扫描的顺序分区（Ordered Partitioning）。HBase 很适合执行基于范围的扫描。

线性可扩展性，支持大表和范围扫描－－由于顺序分区，HBase 很容易横向扩展，同时仍支持行键范围扫描。

辅助索引－－Hbase 并不直接支持辅助索引，但触发器的一个使用场合是，“put”方面的触发器会自动确保辅助索引是最新版本，因而并不给应用程序（客户端）添加负担。

简单聚合－－Hbase Co Processors 支持 HBase 中的即开即用的简单聚合。SUM、MIN、MAX、AVG 和 STD。如果定义 java 类，就可以构建其他聚合，从而执行聚合操作。

实际应用：Facebook Messanger

Cassandra:

宽列式数据库，基于 BigTable 和 DynamoDB 的概念。

Apache Cassandra 是一种主要的 NoSQL 分布式数据库管理系统，它支撑着如今的许多现代商务应用系统，它提供了持续可用性、高扩展性和高性能、强安全性和操作简单性，同时降低了总体拥有成本。

Cassandra 拥有分散式架构。任何节点都能执行任何操作。它提供了 CAP 原理中的 AP（可用性和分区可容忍性）。

Cassandra 拥有出色的单行读取性能，只要最终的一致性语义足以满足使用场合的需要。Cassandra quorum 读取是严格一致性所需要的，它自然不如 Hbase 读取来得快。Cassandra 不支持基于范围的行扫描，这在某些使用场合可能具有局限性。Cassandra 很适合支持单行查询，或者基于列值索引选择多行。

如果数据存储在 Cassandra 中的列里面以支持范围扫描，Cassandra 中行大小的实际限制是 10MB。大于这个数的行会在压缩开销和时间方面引起问题。

Cassandra 支持列族辅助索引，其中列的名称已知（但不支持动态列）。

Cassandra 中的聚合并不受到 Cassandra 节点的支持－－客户端必须提供聚合机

制。聚合需求横跨多个行时，随机分区（Random Partitioning）使得聚合对客户端来说很难。建议使用 Storm 或 Hadoop 用于聚合。

实际应用：Twitter

MongoDB：

最流行的文档数据库之一。

它是一种面向文档的数据库。Mongodb 中的所有数据以 JSON/BSON 格式来处理。它是一种无模式数据库，数据库中的数据量超过 TB 级。它还支持主从复制方法，以便在服务器上复制数据的多个副本，从而使得某些应用系统中的数据整合来得更容易、更快速。

MongoDB 结合了关系数据库的优点和 NoSQL 技术的创新，让工程师能够构建现代应用。

MongoDB 保留了关系数据库最宝贵的功能特性：强一致性、表达式查询语言和辅助索引。因而，开发人员能够以比 NoSQL 数据库更快的速度来构建高度实用的应用程序。

MongoDB 提供了 NoSQL 数据库的数据模型灵活性、弹性可扩展性以及高性能。因而，工程师可以不断改进应用，并且可以在商用硬件上实现几乎无限制的可扩展性。

支持全索引，以实现高性能

实际应用：FourSquare