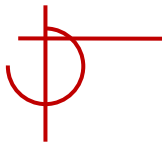
第四章 列族数据库

张元鸣

计算机学院



本章内容

4.1 Cassandra集群机制

4.2 Cassandra数据模型

4.3 Cassandra操作语言

4.4 Cassandra应用实例

4.1 Cassandra集群机制

1、列族数据库简介

列族数据库是一种面向大体量数据，具有高可靠、高性能、可伸缩、结构灵活等特点，能够在大规模集群上存储数据，典型的代表包括Apache Cassandra、 Hadoop Hbase等。

4.1 Cassandra集群机制

Cassandra是Apache开源组织开发的一个开源列族数据库。

- Cassandra是特洛伊王国公主的名字（卡珊德拉），美丽并有预见未来的能力。
- 基于Amazon Dynamo和Google BigTable数据模型。
- 被Facebook，Twitter等公司所采用。

4.1 Cassandra集群机制

Dynamo

+

BigTable

Dynamo：分布式集群管理、复制策略、容错性等

Bigtable：列族数据模型、存储模型等

4.1 Cassandra集群机制

① 弹性可扩展： Cassandra高度可扩展，允许添加更多的硬件以适应更多的客户和更多的数据。

② 始终基于架构：Cassandra没有单点故障，它可以连续用于不能承担故障的关键业务应用程序。

③ 线性可扩展： Cassandra通过增加集群中的节点数量增加吞吐量，保持一个快速的响应时间。

④ 灵活的数据存储：Cassandra适应所有可能的数据格式，包括：结构化，半结构化和非结构化。可以根据需要动态地适应变化的数据结构。

4.1 Cassandra集群机制

⑤ 便捷的数据分发：Cassandra通过在多个数据中心之间复制数据，可以灵活地在需要时分发数据。

⑥ 事务支持：支持最终一致性（AP）。

⑦ 快速写入：被设计为在廉价的商品硬件上运行。它执行快速写入，并可以存储数百TB的数据，而不牺牲读取效率。

4.1 Cassandra集群机制

	Cassandra	关系型数据库
节点水平扩展	Yes	No
高可用性	Yes	No
查询方式	1. CQL（类SQL语言） 2. API，或第三方上层类库	SQL
一致性	可调节一致性	强一致性
事务支持	1. 支持行级别事务 2. 支持轻量级事务处理机制	Yes

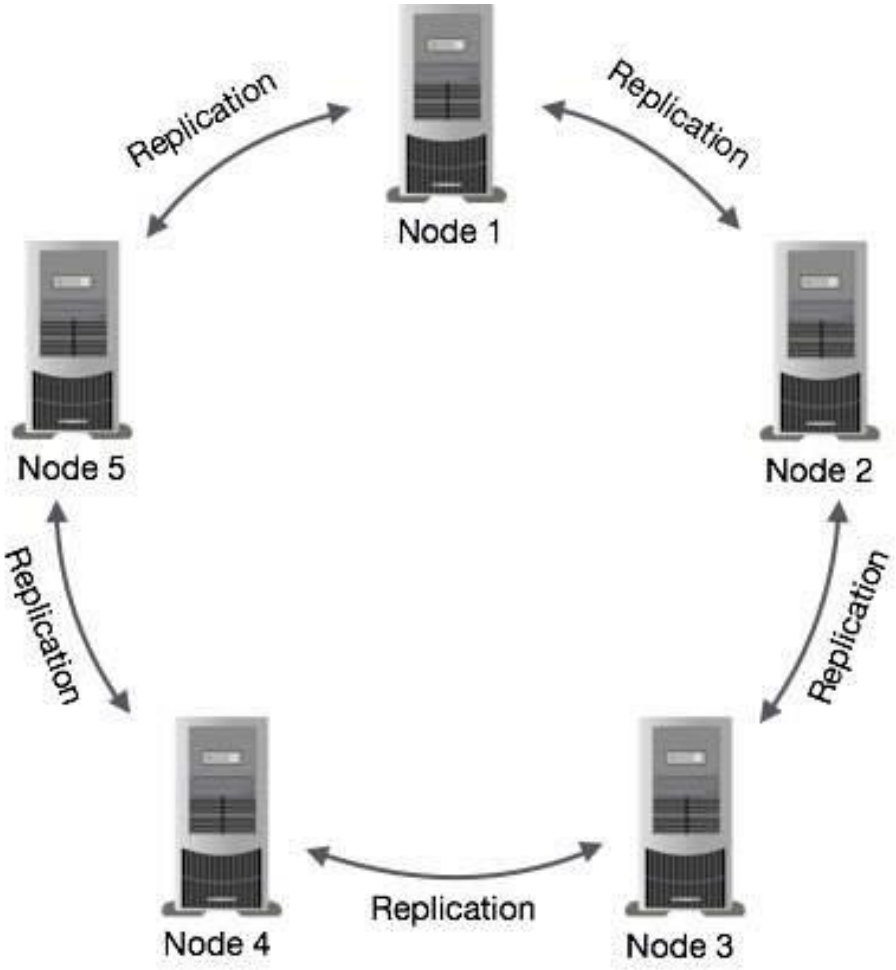
4.1 Cassandra集群机制

2、Cassandra架构

Cassandra的设计目的是处理跨多个节点的大数据工作负载，而没有任何单点故障；节点之间具有对等分布式系统，并且数据分布在集群中的所有节点之间。

- Cassandra集群是一个对等的网络，所有节点都扮演相同的角色， 每个节点是独立的，并且同时互连到其他节点。
- Cassandra集群中的每个节点都可以接受读取和写入请求，无论数据实际位于集群中的何处。
- 当某节点关闭时，可以从网络中的其他节点提供读/写请求。

4.1 Cassandra集群机制



采用环形网络的集群架构

4.1 Cassandra集群机制

1）Cassandra存储机制：

- 提交日志（CommitLog）： Cassandra在写入数据之前，都要先写日志，用于崩溃后的恢复。
- 存储表（Mem表）：是一个驻留在内存中的数据结构，当超过存储表的块大小时，批量写入到磁盘上，即SSTable上。
- 磁盘文件（SSTable）： Cassandra在磁盘上存储列族数据的文件。
- 布隆（Bloom）过滤器：Cassandra中使用了Bloom过滤器来检测行键所代表的数据是否存在。

4.1 Cassandra集群机制

2）Cassandra通信机制：

Cassandra采用Gossiper协议进行节点间通信，通过该机制可以了解集群中包含哪些节点、每个节点的状态，这使得集群中任何一个节点都可以收到其他服务器的状态信息，完成任意读取和写入操作；

此外，Cassandra采用反熵机制定期检查节点数据，保障不同副本数据之间的一致性。这里采用的检查不一致的方法是Merkle Tree（默克尔树）。

4.1 Cassandra集群机制

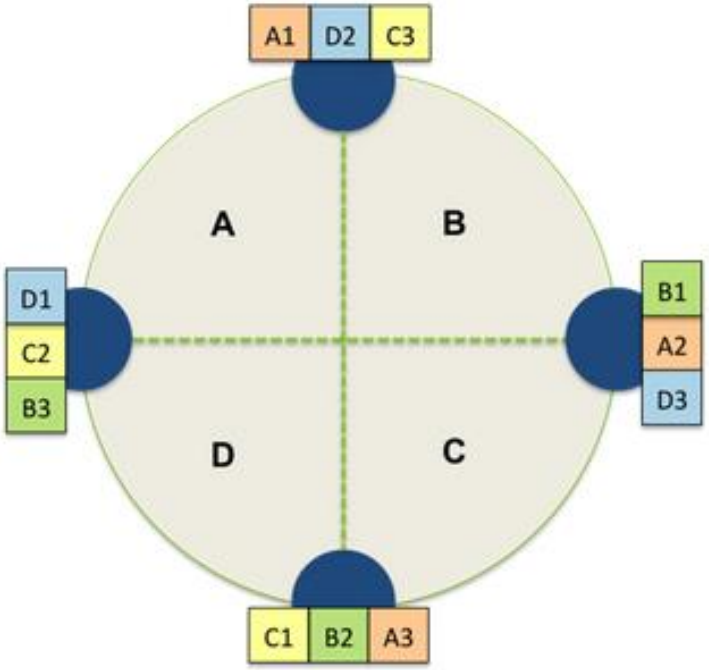
3）Cassandra数据复制

- 数据复制方式：将数据分配到各个节点上，每个节点会存放部分数据的一个副本，允许数据在节点间相互复制，对用户透明。但要求同一行的所有数据必须放在集群的同一节点上。
- 数据副本因子：定义了集群中存储的副本数量，例如副本因子是3，则存储3份数据副本。

4.1 Cassandra集群机制

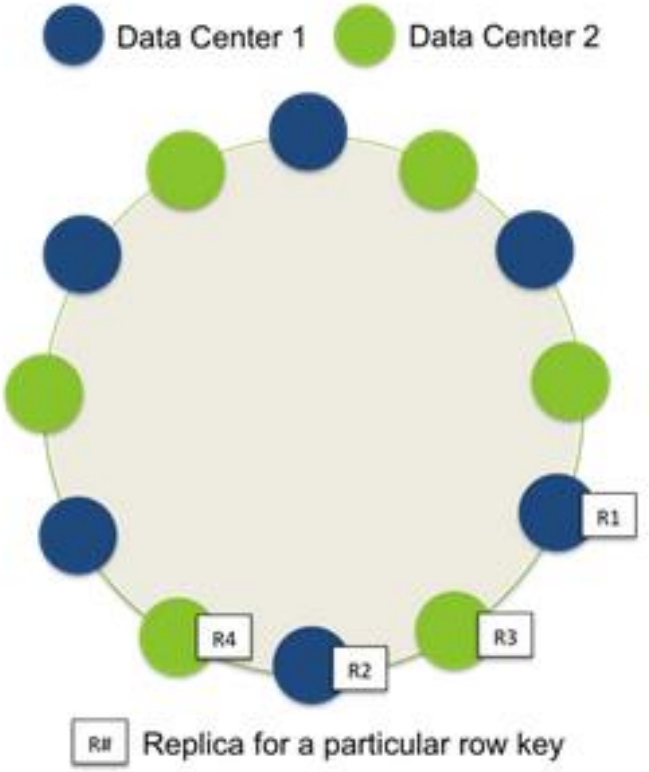
- 数据复制策略：定义数据的副本如何放置到集群环上。
  - SimpleStrategy ： 不考虑机架的因素，存储在连续的几个节点。假如副本数为3，属于A节点的数据存储在相邻的两个节点中。
  - OldNetworkTopologyStrategy :考虑机架的因素，先找一个与第一个数据副本不在同一个数据中心的节点放置第二个副本；然后再继续找与第二个备份节点位于同一个数据中心，但是不同机架的节点进行备份；接下来所有的备份节点寻找策略就按照SimpleStrategy的备份策略继续寻找。
  - NetworkTopologyStrategy ： 将M个副本放置到其他的数据中心，将N-M-1的副本放置在同一数据中心的不同机架中。

4.1 Cassandra集群机制



SimpleStrategy：

单数据中心



NetworkTopologyStrategy：

多数据中心

4.1 Cassandra集群机制

4）Cassandra读写过程

- 写操作：
  - 节点的每个写入活动都由提交日志捕获。
  - 稍后数据将被存储在内存表中。
  - 每当内存表满时，数据将写入SStable数据文件。
  - 所有写入都会在整个集群中自动分区和复制。
  - Cassandra会定期整合SSTables，丢弃不必要的数据。
- 读操作：
  - 先从内存表读取所需要的值。
  - 否则从保存所需数据的SSTable中读取。

4.2 Cassandra数据模型

Cassandra的数据模型由Column、SuperColumn、ColumnFamily和Keyspace等组件所构成。

- 列Column：数据库最小的基本单元，包含键、值和时间戳组成。
- 超列SuperColumn：包含一个键，而键的值又是一系列相应的column，无时间戳。
- 列族ColumnFamily:一个列族包括多个列或超列，相当于table表，列族中的数据存储在一起。
- 键空间Keyspace:一个键空间包括多个列族ColumnFamily，相当于Schema，即数据库的结构。

4.2 Cassandra数据模型

1、列Column

列是最小的数据单元，它是一个三元的数据类型，包括name，value，timestamp。

– name称为键，value称为值，即（Key，Value，timestamp）。

列的数据结构

```
{
  "name": "email",
  "value: "me@example.com",
  "timestamp": 127465418310
 }
```

Json格式

4.2 Cassandra数据模型

2、超级列Super Column

超列是列的数组，它包含一个键名以及一组列，这些列就是键的值，超列无时间戳。

{

lisi:{//这是第一个超级列，SuperColumn的列名称
street:“XiTuCheng road”, //子列名称和值
zip:"410083",
city:"BeiJing"
},

wangwu:{//这是第二个超级列，SuperColumn的列名称
street:"XiTuCheng road", //子列名称和值
zip:"410083",
},

zhaoliu:{//这是第三个超级列，SuperColumn的列名称
street:"XiTuCheng road", //子列名称和值
city:"BeiJing"
}

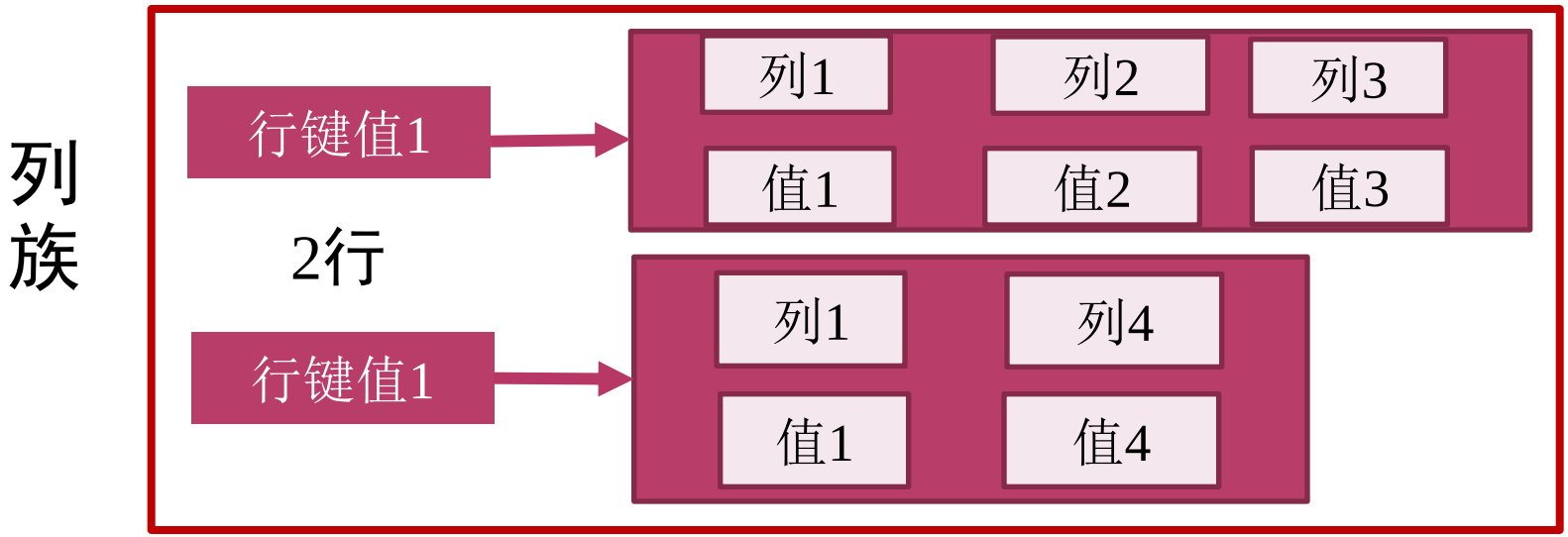
}

4.2 Cassandra数据模型

3、列族Column Family：

列族由许多行组成，每行由一个行键和一组列或超列成，如用户列族、学生列族、地址列族等。

– 把拥有一组列的集合称为行，每行的唯一标识称为行键；其中的列是动态的。



```
{
    zhangsan:{//行键
        lisi:{//这是第一个超级列，SuperColumn的列名称
            street:“XiTuCheng road”, //子列名称和值
            zip:"410083",
            city:"BeiJing"
        },
        wangwu:{//这是第二个超级列，SuperColumn的列名称
            street:"XiTuCheng road", //子列名称和值
            zip:"410083",
        },
        zhaoliu:{//这是第三个超级列，SuperColumn的列名称
            street:"XiTuCheng road", //子列名称和值
            city:"BeiJing"
        }
    }
}
```

Keyspace:Keyspace1

ColumnFamily:Standard2

Key	Columns
studentA	Columns name | value | timestamp "age" | "18" | 1270694041669000 "height" | "172cm" | 1270694041669000
classA	SuperColumns Key | Columns studentX | Columns name | value | timestamp "age" | "20" | 1270694041669000 "height" | "182cm" | 1270694041669000

4.2 Cassandra数据模型

1）双层嵌套：每个ColumnFamily由多个键值对组成，每个键值对都是一个colum.

```
User={//这是一个ColumnFamily，名字是User
        zhangsan:{//这是一个Row，Row的key是zhangsan
        //下面的value可以有无限制的Columns，这里有两个
            username:"zhangsan",
            email:"zhangsan@163.com",
        }，//这个Row结束了
        lisi:{//这是第二个Row，Row的key是lisi
        //value部分，依然是Columns，lisi有三个
            username:"lisi",
            email:"lisi@163.com",
            phone:"123456"
        },//Row结束
}
```

4.2 Cassandra数据模型

2）三层嵌套：每个ColumnFamily的键值对又嵌套了一个键值对，是一个SuperColumn结构。

```
AddressBook={//这是一个SuperColumnFamily，名字是AddressBook
        Computer:{//这是一个Row，行键是zhangsan，下面是任意个SuperColumns
            lisi:{//这是第一个超列，SuperColumn的列名称
                street:“XiTuCheng road”,//子列名称和值
                zip:"410083",
                city:"BeiJing"
            },
            wangwu:{//这是第二个超列，另一个SuperColumn的列名称
                street:"XiTuCheng road", //子列名称和值
                zip:"410083",
                city:"BeiJing"
            },
            zhaoliu:{//这是第三个超列， SuperColumn的列名称
                street:"XiTuCheng road", //子列名称和值
                zip:"410083",
                city:"BeiJing"
            }
        }//end the row of zhangsan
    Maths:{//这是另一个Row，行键是lisi，李四的地址本
        ……
        }
}
```

4.2 Cassandra数据模型

4、键空间Keyspace：

键空间对应RDBMS中的数据库，相当于是列族的容器，包含多个列族，可以进行备份。

- 一般地，一个应用对应一个键空间，一个键空间可以拥有多个列族。
- 列族（或超列族）对应RDBMS中的表，可以存储很多行，每行由行键和一组列组成。

4.2 Cassandra数据模型

Cassandra的数据模型是一个多维的Hashmap：

- 一个Keyspace（库）包含多个列族（表）；
- 一个列族（表）包含多个行；
- 一个行包含一个行键和一族列或一族超列；
- 一个超列包含多个子列。

4.2 Cassandra数据模型

5、Cassandra的新数据模型

由于超级列的灵活性受到限制，在Cassandra1.1版本之后，抛弃了超级列的概念，在列上一级就是表，也就是原先概念上的超级列被抛弃了。

新版本的Cassandra使用了“表”的概念，更接近于传统关系型数据库的相关概念，减少了关系型数据库与非关系型数据库的隔膜。

4.2 Cassandra数据模型

5、Cassandra的新数据模型

- Keyspace：键空间，概念同早期版本，对应一个数据库。
- Table：表，与键族的概念相对应，且与RDBMS中的表很类似，不同的是在 Cassandra 中属于同一张表的数据在物理上是分布在不同节点上存储的，同一张表由多个分区（Partition）组成。

https://www.cnblogs.com/llzx373/p/3324328.html

4.2 Cassandra数据模型

```
CREATE TABLE emp (
  empID int,
  deptID int,
  first_name varchar,
  last_name varchar,
  PRIMARY KEY (empID, deptID)
);
```

4.2 Cassandra数据模型

- 主键：同之前的行键，是一行数据的唯一标识符；如果主键由多个键组成，则称为复合主键。
- 复合主键：复合主键各部分作用不同：
  - 分区键（partition key）：复合主键的第一个或者第一组是分区键，将根据这个column的值来分区。不同的分区会被存储在不同的节点上，同一行数据分配到相同节点。
  - 聚类键（clustering columns）：复合主键的第二个及之后的键为聚类键，用来给数据排序，或者索引。
- 只有主键中涉及的column，才可以作为查询的条件。

4.2 Cassandra数据模型

- 集合数据类型：Cassandra新版本通过设置集合数据类型来灵活实现超级列：
  - List数据类型：列表，存储类型相同且可以重复的值。
    - [value, value,...]
  - Set数据类型：集合，存储类型相同但不重复的值。
    - {value, value, ...}
  - Map数据类型：提供了键到值的映射，每一个元素在内部作为一列存储，可以修改，替换，删除和查询。
    - {'key1':value1, 'key2':value2}

4.2 Cassandra数据模型

6、Cassandra数据排序

1）行排序规则：

列族中的行是有顺序的，在同一分区中根据聚类键的值在数据写入的时候按照指定的规则进行排序，由CompareWith选项来设置，包括以下排序规则：

- BytesType：按比较字节大小，不检查字节的内容是否符合某种编码。
- LongType：按数值排序，默认从小到大。

  如“3”、“123”、“976”、“832416”。
- UTF8Type：按照第一个字符进行排序。

  如：“123”、“3”、“832416”、“976”。

4.2 Cassandra数据模型

6、Cassandra数据排序

2）列的排序规则：

如果一行中有多个列，那么就要对这多个列进行排序，排序的依据是按照列名称进行，而与列值无关。

- BytesType：默认排序方法，直接比较字节，不检查字节的内容是否符合某种编码。
- LongType：按8字节的长整型数值排序，默认从小到大。
- UTF8Type：按照Column第一个字符进行排序。

4.3 Cassandra操作语言

Cassandra的操作语言是CQL，用于创建、更新、删除Keyspace、列族、索引等，以及查询Cassandra中的数据。

4.3 Cassandra操作语言

1、键空间Keyspace操作

1）创建键空间

```
CREATE KEYSPACE  KeySpaceName
WITH replication = {'class': ‘Strategy name’,
                    'replication_factor' : ‘No.Of  replicas’}
AND durable_writes = ‘Boolean value’;
```

4.3 Cassandra操作语言

- replication：该选项指定数据副本位置策略和副本数量，包括class、 replication_factor两个参数
  - class：设定复制策略，有三种策略，包括SimpleStrategy、NetworkTopologyStrategy、OldNetworkTopologyStrategy。
  - replication_factor：副本个数。
- durable_writes：设置写数据时是否写入commit log，如果设置为false，则写请求不会写commit log，会有丢失数据的风险。默认为true，即要写commit log。

4.3 Cassandra操作语言

```
CREATE KEYSPACE teach
    with replication =
        { ‘class’ : ‘SimpleStrategy’, ‘replication_factor’: 1 }
    and durable_writes = false;
```

4.3 Cassandra操作语言

2）修改键空间：

ALTER KEYSPACE KeySpaceName

WITH replication = {'class': ‘Strategy name’, 'rep

lication_factor' : ‘No.Of replicas’};

3）删除键空间

DROP KEYSPACE “KeySpace name”;

4）查看所有的键空间

DESCRIBE keyspaces;

5）切换键空间

USE KeySpaceName;//切换到指定的键空间

4.3 Cassandra操作语言

2、表（列族）操作

1）创建表：表是一个包括多行的容器，类似于RDBMS的表。

```
CREATE (TABLE | COLUMNFAMILY) <表名>
(
   <列名称> <数据类型> primary key,
   <列名称> <数据类型> [static]
    <列名称> <数据类型> ，……
)
```

– 静态列：列为静态(STATIC)则能够在一个给定分区键里的多行数据之间共享数据。

4.3 Cassandra操作语言

```
CREATE TABLE emp (
  emp_id int PRIMARY KEY,
  emp_name text,
  emp_city text,
  emp_sal varint,
  emp_phone varint
  );

CREATE TABLE places (
   latitude double,
   longitude double,
   name text,
   tags text,
   PRIMARY KEY (latitude, longitude, name)
)
```

4.3 Cassandra操作语言

```
CREATE TABLE users (
    username    text,
    id     timeuuid,
    email   text STATIC,
    encrypted_password   blob STATIC,
    body text,
    PRIMARY KEY (username, id)
);
```

- 将 email 和 encrypted_password 两个字段设置为STATIC列了，则意味着同一个 username 只会有一个 email 和 encrypted_password 。
- 在插入或修改数据时，email 和 encrypted_password 都被设置为相同的值。

CQL类型	对应Java类型	描述
ascii	String	ascii字符串
bigint	long	64位整数
blob	ByteBuffer/byte[]	二进制数组
boolean	boolean	布尔
counter	long	计数器
decimal	BigDecimal	高精度小数
double	double	64位浮点数
float	float	32位浮点数
inet	InetAddress	ipv4或ipv6协议的ip地址
int	int	32位整数
list	List	有序的列表
map	Map	键值对
set	Set	集合
text	String	utf-8编码的字符串
timestamp	Date	日期
uuid	UUID	UUID类型
timeuuid	UUID	时间相关的UUID
varchar	string	text的别名
varint	BigInteger	高精度整型

4.3 Cassandra操作语言

2）查看已创建的表

DESCRIBE tables;//查看所有的表（列族）

3）修改表：使用ALTER命令，可以添加列或者删除列。

ALTER (TABLE | COLUMNFAMILY) <tablename> <instruction>

- 增加一列：
  ALTER TABLE tablename ADD new column datatype;
- 删除一列：
  ALTER TABLE tablename DROP column_name;

4.3 Cassandra操作语言

4）删除表：使用DROP命令，可以删除表。

DROP TABLE <tablename>

5）创建索引:

CREATE INDEX <identifier> ON <tablename>(name)

- 在emp1表上的name属性上创建索引：
  CREATE INDEX name ON emp1 (emp_name);

- 删除索引
  drop index name;

4.3 Cassandra操作语言

3、数据管理

1）插入数据：

INSERT INTO <tablename>
(<列名称1>, <列名称2>，....)
VALUES　(<值1>, <值2>....)
With <option>

- 表名、列名大小写不敏感。
- 列的值大小写敏感。
- 字符串用‘’表示.
- 行键排序是不可控的，总是按照设置的排序规则进行排序。

4.3 Cassandra操作语言

- 在表 emp（emp_id，emp_name，emp_city，emp_phone， emp_sal）中插入数据：
  - INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(1,'ram', 'Hyderabad', 9848022338, 50000);
  - INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(2,'robin', 'Hyderabad', 9848022339, 40000);
  - INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(3,'rahman', 'Chennai', 9848022330, 45000);

4.3 Cassandra操作语言

2）更新数据

UPDATE <tablename>
SET <column name> = <new value>
<column name> = <value>....
WHERE <condition>

- 不可以修改主键的值。

- 更新两列的数据：

```
UPDATE emp SET emp_city='Delhi’, emp_sal=50000
  WHERE emp_id=2;
```

4.3 Cassandra操作语言

3）查询数据

SELECT * FROM <table name>
WHERE <condition>;

- Where 条件中只支持主键列及索引列的查询。
- 分区主键只能用”=”比较运算符进行查询。
- 聚类主键支持=、>、 <、 >=、 <=，且要依次使用。
- 索引列只支持=查询。
- 否则，需要增加allow filtering短语，表示强制查询。

4.3 Cassandra操作语言

- **查询用户ID为“jack”的行信息，其命令为：**

  SELECT * FROM Message WHERE user_id = 'jack‘;

- **查询用户ID为“jack”，消息ID号是“2021”的行信息，其命令为：**

  SELECT * FROM Message WHERE user_id = 'jack’ AND message_id =’2021’;

4.3 Cassandra操作语言

Cassandra 查询限制

```
CREATE TABLE mykeyspace.mytable (
   key1 text,
   key2 text,
   key3 text,
   column1 bigint,
   column2 int,
   column3 timestamp,
   PRIMARY KEY (key1, key2, key3);
)
```

```
select * from mytable where key1=?
select * from mytable where key1=? and key2=?
select * from mytable where key1=? and key2=? and key3=?
select * from mytable where key1=? and key2> ?
select * from mytable where key1=? and key2=? and key3<? and key3>= ?
```

```
select * from mytable  where key1=? and key3=?
select * from mytable where key1=? And column2=?
```

4.3 Cassandra操作语言

4）删除数据

DELETE FROM <identifier> WHERE <condition>;

- 删除emp_id=3的emp_sal列值：
  DELETE emp_sal FROM emp WHERE emp_id=3;

- 删除emp_id=3的一个整个行：
  DELETE FROM emp WHERE emp_id=3;

可以看出删除也只能按照条件进行删除

4.3 Cassandra操作语言

4、集合数据类型

Cassandra还提供了集合数据类型，用于存储多个数据。

1）List数据类型: 列表保持元素的顺序，并且值可以重复。

2）Set数据类型:用于存储一组元素的数据类型，集合的元素将按排序顺序返回，值不重复。

3）Map数据类型: 映射是用于存储键值对的数据类型。

4.3 Cassandra操作语言

1）List数据类型

列表保持元素的顺序，并且值可以被多次存储。

– 创建表：

```
CREATE TABLE data( name text PRIMARY KEY,
                   email list<text> )
```

– 添加行：

```
INSERT INTO data(name, email)
VALUES (‘Liu’, ['abc@gmail.com’, 'cba@yahoo.com’] )
```

– 更新数据：

```
UPDATE data
SET   email[1] =['xyz@tutorialspoint.com’]
Where name = ‘Liu’;
```

4.3 Cassandra操作语言

2）Set数据类型

集合用于存储一组元素的数据类型，集合的元素将按顺序返回。

- 创建表：

```
CREATE TABLE data2 (name text PRIMARY KEY, phone set<varint>)
```

- 添加行：

```
INSERT INTO data2 (name, phone)
VALUES (‘Liu’, {13522332228,13984802233});
```

- 更新数据：

```
UPDATE data2
SET phone = phone + {9848022330}
Where name = Liu';
```

4.3 Cassandra操作语言

3）Map数据类型

映射Map是用于存储键值对的数据类型。

- 创建表：

```
CREATE TABLE data3 (name text PRIMARY KEY,
 address map<text, text>)
```

- 插入行：

```
INSERT INTO data3 (name, address)
VALUES  ('robin', {'home' : 'hyderabad' , 'office' : 'Delhi' } );
```

- 更新数据：

```
UPDATE data3
SET address[‘offfice’] ='Mumbai’
 WHERE name = 'robin';
```

4.3 Cassandra操作语言

```
create table testset (   //创建表
   id text primary key,
   emails list <text>,       // list类型
   phones set <int>,         //set类型
   courses map<text,text>   //map类型
);
insert into testset (id, emails, phones, courses)
values('1',['1@zjut','11@zjut'] ,{1,2}, {'C01':'92','C02':'75'} );//插入行
insert into testset (id,emails,phones,courses)
values('2',['2@zjut','22@zjut'] ,{2,22}, {'C03':'80','C04':'85'} );//插入行
insert into testset (id,emails,phones,courses)
values('3',['3@zjut','33@zjut'] ,{3,33}, {'C03':'80','C04':'85'} ); //插入行
```

4.3 Cassandra操作语言

- update testset set emails[0]=‘10@zjut’ where id='1' //更新列表类型元素
- update testset set emails=emails-['10@zjut'] where id='1'; //删除list类型元素

- update testset set phones=phones+{10} where id='1'; //增加set类型元素
- update testset set phones=phones-{10} where id='1'; //删除set类型元素

- update testset set courses['C01']='99' where id='1'; //更新map类型元素
- update testset set courses=courses+{'C01':'99'} where id='1'; //更新map类型元素

4.3 Cassandra操作语言

- create index testset_index on testset (courses) ; //在map列上创建索引
- select * from testset where courses contains '80'; //根据map列上的value查找

- create index testset_index1 on testset (keys(courses)) ; //在map列的Key上创建索引
- select * from testset where courses contains key 'C01'; //创建索引之后根据Key查询

4.3 Cassandra操作语言

5、数据排序

Cassandra使用order by语句对查询结果进行排序。但与SQL语句的排序相比，Cassandra对排序语句的使用限制较多，需要遵循以下规则：

（1）查询语句必须有分区键（第一主键）的查询条件（Where语句），分区键决定了数据的存储节点，数据只能在节点内进行排序。

（2）查询语句只能根据聚类键的顺序排序或相同排序。顺序排序是指order by后面只能是先第一聚类键排序，再第二聚类键排序，依次类推；相同排序是指参与排序的聚类键键要么与创建表时指定的顺序一致，要么全部相反。

（3）查询语句的查询条件中不能含有索引列。

4.3 Cassandra操作语言

```
create table User (
  user_id        int,
  area           text,
  name           text,
  age            int ,
  emails         list<text>,
  phones         set<varint>,
  contracts          map<text, text>,
  primary key (user_id, area, name)
) with clustering order by (area desc, name asc);
```

4.3 Cassandra操作语言

- **以下查询语句是错误的：**

① select * from user where user_id=123 order by area desc, name desc; //User表指定的排序规则是area desc, name asc，而查询语句与该排序规则没有完全一致或者完全相反。

②select * from user order by area desc; //没有第一主键（分区键）。

③select * from user where user_id=1 order by name desc; //不能以第二聚类键开始排序。

4.3 Cassandra操作语言

- **以下查询语句是正确的：**

① select * from user where user_id=123 order by area desc;

② select * from user where user_id=123 order by area desc, name asc;

③ select * from user where user_id=123 and area ='hangzhou' order by area desc;

④ select * from user where user_id=123 and area ='hangzhou' order by area desc, name asc;

⑤ select * from user where user_id=123 order by area asc;

⑥ select * from user where user_id=123 order by area asc, name desc;

⑦ select * from user where user_id=123 and area ='hangzhou' order by area asc;

⑧ select * from user where user_id=123 and area ='hangzhou' order by area asc, name desc;

4.3 Cassandra操作语言

6、聚合函数

Cassandra列族数据库提供了一些聚合函数对表中的数据进行统计。

（1）count()函数：返回表中行的数量。

- 返回user表中的所有行数，其命令为：

SELECT COUNT (*) FROM user;

- 返回user表中年龄非空的行数，其命令为：

SELECT COUNT (age) FROM user;

年龄为空的列不计算在内。

4.3 Cassandra操作语言

（2）min()函数：用于返回某一列的最小值。

- 返回user表中年龄的最小值，其命令为：

  SELECT min(age) FROM user;

（3）max()函数：用于返回某一列的最大值。

- 返回user表中年龄的最大值，其命令为：

  SELECT max(age) FROM user;

（4）avg()函数：用于返回某一列的平均值。

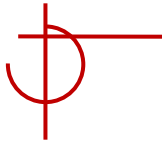
- 计算user表中的平均年龄，其命令为：

  SELECT avg(age) FROM user;

（5）sum()函数：用于返回某一列的总和。

- 计算user表中的年龄总和，其命令为：

  SELECT sum(age) FROM user;



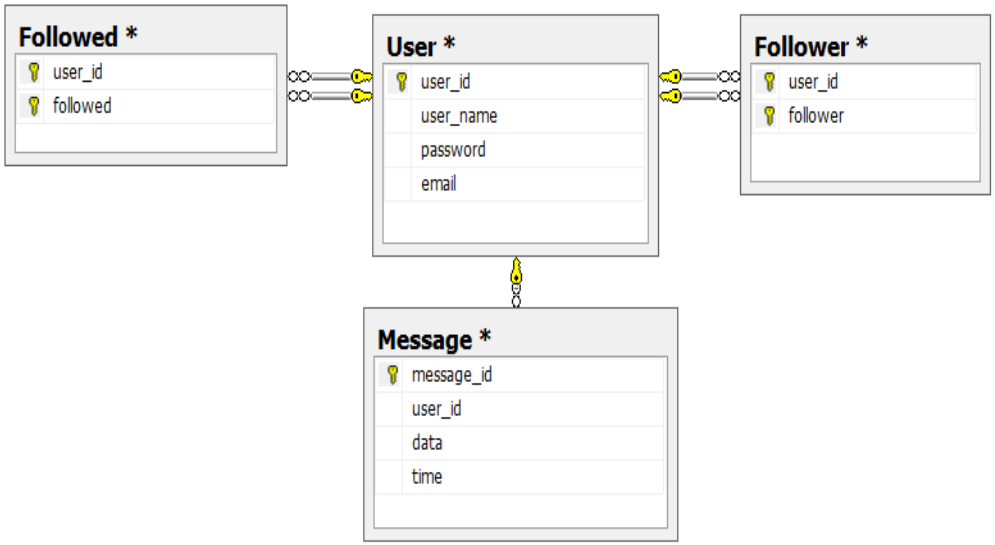
4.3 Cassandra操作语言

总结

与SQL相比，CQL并不支持连接（join）操作和分组（group by）操作，也不支持复杂的排序操作，为了实现数据均衡分区，需要精心选择分区键，并保证分区键唯一；此外，根据业务特征，将偏向读操作的表和写操作的表实现分离，有助于提高查询性能。

4.4 应用案例

以微博为例给出一个应用实例。假设每个注册用户都有一些个人信息，每个用户都有自己关注的朋友，并且也有关注该用户的用户，每个用户可以发长度不超过140个字符的信息。如果是用关系型数据库存储这些数据，可以按照右边的方式设计数据库结构，如图所示。



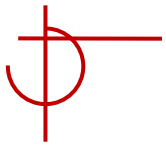
字段名	说明	允许空	字段类型	描述
user_id	分区键	N	INT	用户 ID
message_id	聚类键	N	INT	消息 ID
user_name	静态列	N	VARCHAR (64)	用户名
password	静态列	N	VARCHAR (8)	密码
email	静态列		VARCHAR (64)	邮箱
followed			SET	关注的用户
follower			SET	被关注的用户
message			VARCHAR (64)	消息
datatime			TIMESTAMP	消息时间

4.4 应用案例

利用Cassandra数据存储以上数据，设计user_message表用来存储用户及其发布的消息：

- user_id：分区键，列族数据库按照该列进行分区，不同的用户的数据被映射到相应的节点上，相同用户的数据存储在同一个节点；
- Message_id：聚类键，并按照升序排序；
- user_name、password、email：静态列，一个用户无论有多少行，实际只存储一份数据；
- follower和followed：数据类型是集合类型，分别存储多个该用户关注的其他用户和被其他用户关注的用户；
- Message：用户发表的消息，最长140个字节；
- Timestamp：发表消息的时间戳。

```
CREATE TABLE user_message(
  user_id int,
  message_id int,
  user_name varchar(64) static,
  password varchar(8) static,
  email varchar(64) static,
  follower set< int >,
  followed set< int >,
  message varchar(140),
  timestamp timestamp,
  PRIMARY KEY(User_id, Message_id)
)with clustering order by (Message_id asc);
```



Chapter over