[Cassandra教程]（四）使用Key的正确姿势

时间 2016-09-05 14:36:25 FlyML

原文

NoSQL DB的表与数据模型设计跟传统的RDBMS很不一样。最大的一个不同就是反范式。比如提倡数据冗余，使得不至于写出非常复杂的SQL语句。

就Cassandra而言，最关键的地方在于Key的设计。Cassandra之中一共包含下面4种Key:

1. Primary Key
2. Partition Key
3. Composite Key
4. Compound Key
5. Clustering Key

OMG~~ 是不是太多了？ 让我们一个个的来解释

首先， Primary key 是用来获取某一行的数据， 可以是一列或者多列（复合列 composite）

Primary = Partition Key + [Clustering Key] （Clustering Key 可选）

Clustering keys 包括下面两种情况：

(1) composite key

(2) compound key

```
-- 一列
create table stackoverflow (
            key text PRIMARY KEY,
            data text
);
-- 复合列
create table stackoverflow (
            key_part_one text,
            key_part_two int,
            data text,
            PRIMARY KEY(key_part_one, key_part_two)
    );
```

在上面复合列的table之中，全称： Composite Primary Key

并且：

(1) key_part_one -> partition key

(2) key_part_two -> clustering key

注意： partition key, clustering key 都可以是复合列。 参考下面这个更复杂一些的例子：

```
-- Multiple Partition Keys and Multiple Clustering Keys
create table stackoverflow (
            k_part_one text,
            k_part_two int,
            k_clust_one text,
            k_clust_two int,
            k_clust_three uuid,
            data text,
            PRIMARY KEY((k_part_one,k_part_two), k_clust_one, k_clust_two,
k_clust_three)
    );
```

Partition Key : Cassandra会对partition key 做一个hash计算，并自己决定将这一条记录放在哪个node

比较好的做法就是尽量的将记录区分开来

Goog Sample: UUID 几乎唯一

Bad Sample: ZipCode/TimeZone, 非常有限

Partition Key的设计，可以完全的借用MySQL的主键。MySQL的主键可以是单一主键，也可以是复合主键。但是主键不能重复，这个在笔者见过的数据库产品之中都是一样的。

补充说明：

在Cassandra之中， 如果对相同的primary key 插入多次，实际上第二次开始类似更新

这个过程就像upsert

Cassandra会给每一行数据一个timestamp，如果有多行数据，Cassandra会取时间最新的数据返回

Clustering Key : 主要用于进行Range Query. 并且使用的时候需要按照建表顺序进行提供信息

参考下面代码：

```
-- 创建表
-- 注意state 这个field
CREATE TABLE users (
    mainland text,
```

```
    state text,
    uid int,
    name text,
    zip int,
    PRIMARY KEY ((mainland), state, uid)
)


-- 插入一些值
insert into users (mainland, state, uid, name, zip)
       VALUES ('northamerica', 'washington', 1, 'john', 98100);
insert into users (mainland, state, uid, name, zip)
       VALUES ('northamerica', 'texas', 2, 'lukas', 75000);
insert into users (mainland, state, uid, name, zip)
       VALUES ('northamerica', 'delaware', 3, 'henry', 19904);
insert into users (mainland, state, uid, name, zip)
       VALUES ('northamerica', 'delaware', 4, 'dawson', 19910);
insert into users (mainland, state, uid, name, zip)
       VALUES ('centraleurope', 'italy', 5, 'fabio', 20150);
insert into users (mainland, state, uid, name, zip)
       VALUES ('southamerica', 'argentina', 6, 'alex', 10840);
```

有效的查询：

```
select * from users where mainland = 'northamerica' and state > 'ca' and state < 'ny';
```

查询结果：

```
mainland        | state        | uid | name    | zip
-------------+----------+-----+--------+-------
northamerica | delaware |    3 |    henry | 19904
northamerica | delaware |    4 | dawson | 19910
```

无效的查询：

```
-- 没有提供stat 信息
select * fromuserswheremainland = 'northamerica' and uid < 5;
```

原因很简单，参考上一篇文章提到的数据模型：

```
Map<RowKey, SortedMap<ColumnKey, ColumnValue>>
```

Cassandra 整体数据可以理解成一个巨大的 嵌套的 Map。 只能 按顺序 一层一层的深入，不能跳过中

间某一层~

```
create table stackoverflow (

key_part_one text,

key_part_two int,

data text,

PRIMARY KEY(key_part_one, key_part_two)

);

insert into stackoverflow (key_part_one, key_part_two, data) VALUES (‘ronaldo’, 9, ‘football player’);

insert into stackoverflow (key_part_one, key_part_two, data) VALUES (‘ronaldo’, 10, ‘ex-football player’);

select * from stackoverflow where key_part_one
= ‘ronaldo’

;
```

比如记录PM2.5的历史纪录，就可以设计成(location_id, record_date). 这样进行

对于复合键，在查询的时候并不需要全部提供：

比如下面的例子： partition key 有两个，但是在查询的时候可以只提供其中一个的值

Q: 如果只根据key_part_two 来查询呢？

A: 默认不允许： 会触发数据过滤（需要扫描的数据比较多）

如果一定要执行：在cql 之中增加ALLOW FILTERING

```
select * from stackoverflow where key_part_two = 9 ALLOW FILTERING ;
```

Q: 是不是设置的key越多越好？

A: 需要根据业务来确定：

eg: `PRIMARY KEY((col1, col2), col10, col4))`

正确的where查询条件：

- col1 and col2
- col1 and col2 and col3
- col1 and col2 and col3 and col4

无效的where查询条件：

- col1 (这样Cassandra无法找到在哪一行)
- col1 and col2 and col4
- anything that does not contain both col1 and col2

特别要注意上面红色字体表示的情况

总结一下：

Cassandra之中的存储，是2-level nested Map

Partition Key -> Custering Key -> Data

partition key : eq and in

clustering key: < <= = >= > in

参考网页：

* Difference between partition key, composite key and clustering key in Cassandra?

本文为原创文章，转载请注明 出处：http://www.flyml.net

分享

收藏 纠错