Wide Partitions Data Modeling
Hello!

I am Tzach Livyatan

I know something about NoSQL

You can find me at tzach@scylladb.com
About ScyllaDB

+ The Real-Time Big Data Database
+ Drop-in replacement for Apache Cassandra and Amazon DynamoDB
+ 10X the performance & low tail latency
+ Open Source, Enterprise and Cloud options
+ Founded by the creators of KVM hypervisor
+ HQs: Palo Alto, CA, USA; Herzelia, Israel; Warsaw, Poland

We are hiring!

COMCAST
IBM
Grab
Santander
SAMSUNG sucks
Opera
Fanatics
SAMSUNG SDS
OLA
STARBUCKS COFFEE
Lookout
adgear
FIREEYE
AppNexus
Zillow
Zenly
MediaMath
Agenda

NoSQL Intro
Wide Partition Data Modeling
  Choosing the right Partition Key
  Choosing the right Clustering Key
  Using Materialized Views / Secondary Index
Summary
1. NoSQL in 5 mins
NoSQL - By Availability vs Consistency

Pick Two

Availability

Consistency

Partition Tolerance

Or use a more granular model, like PACELC
## NoSQL - By Data Model

<table>
<thead>
<tr>
<th>Data Model</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key / Value</td>
<td>RocksDB, Aerospike, Redis</td>
</tr>
<tr>
<td>Document store</td>
<td>MongoDB, Couchbase</td>
</tr>
<tr>
<td>Wide column store</td>
<td>Cassandra, DynamoDB, HBase, Apache Cassandra</td>
</tr>
<tr>
<td>Graph</td>
<td>JanusGraph, Neo4j</td>
</tr>
</tbody>
</table>
Cluster - Node Ring
Scylla Architecture

Active/active, replicated, auto-sharded
2. Data Modeling
Relational:

NoSQL:
Circle of (NoSQL) Life

Update
Update schema / application

Application
Estimate Read and Write pattern

Measure
Use Metric to evaluate the data model

Data Model
Optimize the model base on Apps SLA
Partition Key

<table>
<thead>
<tr>
<th>pet_chip_id</th>
<th>time</th>
<th>heart_rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>s123d...</td>
<td>2019-03-04</td>
<td>100</td>
</tr>
</tbody>
</table>

Partition Key

ID: s123d...

Partition Hash Function

Hash: X

X

Y

Z
Structure of data in Scylla

- Data is stored in MemTables and SSTables
- The highest level are **partitions**, identified by a partition key
- Partitions contain **rows**, identified by a clustering key
- Data is sorted in the partition by the clustering key, helping the storage engine to store data efficiently for querying
CREATE TABLE pet_owner (  
    pet_chip_id uuid,  
    owner uuid,  
    pet_name text,  
    PRIMARY KEY (pet_chip_id)  
);
Choosing a Partition Key

● High Cardinality
● Even Distribution

Avoid

● Low Cardinality
● Hot Partition
● Large Partition

CREATE TABLE pet_owner (  
  pet_chip_id uuid,  
  owner uuid,  
  pet_name text,  
  PRIMARY KEY (pet_chip_id)  
);
<table>
<thead>
<tr>
<th>Partition Key (pet_chip_id)</th>
<th>Column Name</th>
<th>Column Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>268e074a-a801-476c-8db5-276eb2283b03</td>
<td>owner</td>
<td>642adfee-6ad9-...</td>
</tr>
<tr>
<td></td>
<td>pet_name</td>
<td>Buddy</td>
</tr>
<tr>
<td>feed97e9-4d77-40c9-ba15-c45478542e20</td>
<td>owner</td>
<td>642adfee-6ad9-...</td>
</tr>
<tr>
<td></td>
<td>pet_name</td>
<td>Max</td>
</tr>
<tr>
<td>47045aee-fd11-44c6-9d0f-82428434e887</td>
<td>owner</td>
<td>642adfee-6ad9-...</td>
</tr>
<tr>
<td></td>
<td>pet_name</td>
<td>Rocky</td>
</tr>
</tbody>
</table>
Choosing a Partition Key

- User Name
- User ID
- User ID + Time
- Sensor ID
- Sensor ID + Time
- Customer

- State
- Age
- Favorite NBA Team
- Team Angel or Team Spike
Hot (top) Partition

Nodetool top partitions

Why Amazon DynamoDB isn’t for everyone, Forrest Brazeal
Wide Partition Example

Query:

```
SELECT * from heartrate_v10 WHERE
pet_chip_id = 80d39c78-9dc0-11eb-a8b3-0242ac130003 LIMIT 1;
```

```
SELECT * from heartrate_v10 WHERE
pet_chip_id = 80d39c78-9dc0-11eb-a8b3-0242ac130003 AND
time >= '2021-05-01 01:00+0000' AND
time < '2021-05-01 01:03+0000';
```

https://gist.github.com/tzach/7486f1a0cc904c52f4514f20f14d2a97
CREATE TABLE heartrate_v10 (  
  pet_chip_id uuid,  
  owner uuid,  
  time timestamp,  
  heart_rate int,  
  PRIMARY KEY (pet_chip_id, time)  
);

<table>
<thead>
<tr>
<th>pet_chip_id</th>
<th>time</th>
<th>heart_rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:00:00.000000+0000</td>
<td>120</td>
</tr>
<tr>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:01:00.000000+0000</td>
<td>121</td>
</tr>
<tr>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:02:00.000000+0000</td>
<td>120</td>
</tr>
</tbody>
</table>
Wide Partition Example

Partition

Rows sorted by time

Large Partition?
Choosing a Clustering Key

- Allow useful range queries
- Allow useful LIMIT
SELECT * from heartrate_v10 WHERE
pet_chip_id = 80d39c78-9dc0-11eb-a8b3-0242ac130003 LIMIT 1;

CREATE TABLE heartrate_v5 (  
    pet_chip_id uuid,  
    time timestamp,  
    heart_rate int,  
    PRIMARY KEY (pet_chip_id, time)  
) WITH CLUSTERING ORDER BY (time DESC);
CREATE TABLE heartrate_v6 (  
    pet_chip_id uuid,  
    date text,  
    time timestamp,  
    heart_rate int,  
    PRIMARY KEY ((pet_chip_id, date), time));
3. Materialized Views and Secondary Index
Example

CREATE TABLE heartrate_v10 (
    pet_chip_id uuid,
    owner uuid,
    time timestamp,
    heart_rate int,
    PRIMARY KEY (pet_chip_id, time)
);

https://gist.github.com/tzach/7486f1a0cc904c52f4514f20f14d2a97
Example

SELECT * FROM heartrate_v10 WHERE pet_chip_id = a2a60505-3e17-4ad4-8e1a-f11139ca1cc;

SELECT * FROM heartrate_v10 WHERE owner = 642adfee-6ad9-4ca5-aa32-a72e506b8ad8;

SELECT * FROM heartrate_v10 WHERE owner = 642adfee-6ad9-4ca5-aa32-a72e506b8ad8  ALLOW FILTERING;

https://gist.github.com/tzach/4b9dadbc6e8a9c50369da05631c5e13e
CQL syntax

CREATE MATERIALIZED VIEW heartrate_by_owner AS
SELECT * FROM heartrate_v10
WHERE owner IS NOT NULL AND pet_chip_id IS NOT NULL AND time IS NOT NULL
PRIMARY KEY(owner, pet_chip_id, time);

SELECT * FROM heartrate_by_owner WHERE owner = 642adfee-6ad9-4ca5-aa32-a72e506b8ad8;

https://docs.scylladb.com/getting-started/mv/
Example

<table>
<thead>
<tr>
<th>pet_chip_id</th>
<th>time</th>
<th>owner</th>
<th>heart_rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:00:00.000000+0000</td>
<td>642adfee-6ad9-4ca5-a32-a72e506b8ad8</td>
<td>120</td>
</tr>
<tr>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:01:00.000000+0000</td>
<td>642adfee-6ad9-4ca5-a32-a72e506b8ad8</td>
<td>121</td>
</tr>
<tr>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:02:00.000000+0000</td>
<td>642adfee-6ad9-4ca5-a32-a72e506b8ad8</td>
<td>120</td>
</tr>
</tbody>
</table>

CREATE MATERIALIZED VIEW

<table>
<thead>
<tr>
<th>owner</th>
<th>pet_chip_id</th>
<th>time</th>
<th>heart_rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>642adfee-6ad9-4ca5-a32-a72e506b8ad8</td>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:00:00.000000+0000</td>
<td>120</td>
</tr>
<tr>
<td>642adfee-6ad9-4ca5-a32-a72e506b8ad8</td>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:01:00.000000+0000</td>
<td>121</td>
</tr>
<tr>
<td>642adfee-6ad9-4ca5-a32-a72e506b8ad8</td>
<td>80d39c78-9dc0-11eb-a8b3-0242ac130003</td>
<td>2021-05-01 01:02:00.000000+0000</td>
<td>120</td>
</tr>
</tbody>
</table>
1. INSERT INTO heartrate 
   (pet_chip_id, Owner, Time, heart_rate) 
   VALUES (..);

2. INSERT INTO 
   heartrate 
   Base replica 
   View replica 
   Coordinator

3. INSERT INTO 
   heartrate_by_owner

View is another table
View is another table

1. SELECT * FROM heartrate_by_owner WHERE owner = '642a..';

2. SELECT * FROM heartrate_by_owner WHERE owner = '642a..';
Choosing a MV Index Key

- High Cardinality
- Even Distribution

Avoid

- Low Cardinality
- Hot Partition
- Large Partition

Wait, what about JOINs?

- Try to avoid them using denormalization
- For ad-hoc operations - use external engine: Spark, Presto
Circle of (NoSQL) Life

**Update**
Update schema / application

**Application**
Estimate Read and Write pattern

**Measure**
Use Metric to evaluate the data model

**Data Model**
Optimize the model base on Apps SLA
Experience Scylla for Yourself

Download Scylla Open Source: scylladb.com/download

Learn more https://university.scylladb.com/
Contact me tzach@scylladb.com
THANK YOU!

PERcona
LIVEONLINE
MAY 12 - 13th
2021