CrateDB

Kryštof Hrubý

Database Model

- SQL relational database
- Search engine
- Document store
- Time Series database

SQL relational database

- Supports most SQL commands
  - Joins, Aggregation, Sort, …
- Built as distributed database from the start
  - Uses Apache Lucene, Elasticsearch, Netty
- Each table split into shards
  - Shards are distributed uniformly across cluster
  - Replication factor can be chosen
- Eventual Consistency
- No support for transactions
  - Any operation on a row is atomic
- Write-ahead logging

Search Engine

- Uses Apache Lucene
  - Storage, indexing, text and geospatial search
- Provides full-text and geospatial search

Document Store

- Each table row is a semi structured document
- Document – nested structure of object and array types
- JSON data can be loaded to either Object or Array type
- Operations on documents are atomic
- Object similar to JSON Object

Query - Create Table

```
INSERT INTO products (
    id,
    name,
    description,
    specification,
    cost,
    inStock
) VALUES (
  1,
  'Fieldmann FZR 70335-A 2x20V',
  'Akumulátorová sekačka na trávu značky FIELDMANN...',
  {
    "length" = 130.4,
    "width" = 50.4
  },
  3399.90,
  10
);
```

```
CREATE TABLE products (
  id INT PRIMARY KEY,
  name TEXT,
  description TEXT,
  specification OBJECT(DYNAMIC) AS (
    length FLOAT,
    width FLOAT
  ),
  cost FLOAT,
  inStock INT,
  reviews ARRAY(TEXT),
  INDEX productNameFt USING FULLTEXT(name)
) CLUSTERED INTO 10 SHARDS;
```

Query - Delete, Update

```sql
DELETE FROM products
WHERE id = 0;
```

```sql
UPDATE products SET
  specification['length'] = 100.0
WHERE id = 1;
```

Query - Text search

```
1   SELECT name, inStock, _score
2   FROM products
3   WHERE MATCH(productNameFt, 'Fieldmann -A')
4   ORDER BY _score DESC;
```

name	inStock	_score
Fieldmann FZR 70335-A 2x20V	10	0.26152915
Fieldmann FZR 70435-0	3	0.13076457

Query - Joins

```sql
CREATE TABLE orders (
  id INT PRIMARY KEY,
  customer TEXT
);

INSERT INTO orders (
  id,
  customer
) VALUES (0, 'Jan Novak'), (1, 'Katerina Novotna');
```

```sql
CREATE TABLE productsOrders (
  productId INT,
  orderId INT,
  count INT,
  PRIMARY KEY (productId, orderId)
);

INSERT INTO productsOrders (
  productId,
  orderId,
  count
) VALUES (0, 0, 1), (1, 0, 2), (2, 1, 3);
```

Query - Joins

```sql
SELECT o.customer, SUM(po.count * p.cost) AS paid
FROM products p
JOIN productsOrders po ON p.id = po.productId
JOIN orders o ON po.orderId = o.id
GROUP BY o.customer;
```

name	paid
Katerina Novotna	16199.699
Jan Novak	11299.699

Advantages, Disadvantages

- ✓ SQL language with objects
  - It might lead user to use slow joins
- ✓ Easy to setup
- ✓ Provides free client with db management, console, …
- ✓ Nice documentation
  - Although sometimes not clear what is supported and what not

- × Full-text indices can be created only when creating table
- × When data are inserted, subsequent select may not find them
  - × Even for small cluster with one node
- × Weird float arithmetics

Thanks for your attention