Complex Data Types

Paul O'Fallon

@paulofallon

Summary

Collections

Tuples

User defined types

Leveraging JSON

Collections: Set

More info

Level	Intermediate
Rating	★★★★½
Duration	2h 48m
Released	19 Dec 2012
Features	CC

```
CREATE TABLE courses (
    id varchar,
    name static,
    // ...
    features set<varchar> static,
    module_id int,
    // ...
    PRIMARY KEY (id, module_id)
);
```

Collections: Set

Inserting with a set

```
INSERT INTO courses (id, features)
VALUES ('nodejs-big-picture', {'cc'});
```

Adding to a set

```
UPDATE courses SET features = features + {'cc'}
WHERE course_id = ‘nodejs-big-picture’;
```

Collections: Set

Removing from a set

```
UPDATE courses SET features = features - {'cc'}
WHERE course_id = ‘nodejs-big-picture';
```

Emptying the entire set

```
UPDATE courses SET features = {}
WHERE course_id = 'nodejs-big-picture';
```

Collections: List

Course Overview		1m 10s
Course Overview		1m 10s
Considering Node.js		15m 0s
Course Introduction		3m 4s
Where Is Node.js Commonly Found?		4m 37s
What Makes up Node.js?		2m 53s
A Brief History / When Node May Not Be the Best Fit		4m 25s

Collections: List

```
CREATE TABLE courses (
    id varchar,
    name static,
    // ...
    module_id int,
    clips list<varchar>,
    // ...
    PRIMARY KEY (id, module_id)
);
```

Collections: List

Inserting with a list

```
INSERT INTO courses (id, module_id, clips)
VALUES ('nodejs-big-picture',1,['Course Overview']);
```

Adding to a list

```
UPDATE courses SET clips = ['Course Introduction'] + clips
WHERE course_id = 'nodejs-big-picture' AND module_id = 2;

UPDATE courses SET clips = clips + ['Considering Node.js']
WHERE course_id = 'nodejs-big-picture' AND module_id = 2;
```

Collections: List

Removing from a list

```
UPDATE courses SET clips = clips - [‘Course Overview']
WHERE course_id = ‘nodejs-big-picture‘ and module_id = 1;
```

Manipulating a list by element id

```
UPDATE courses SET clips[2] = ‘What Makes up Node.js?’
WHERE course_id = ‘nodejs-big-picture‘ AND module_id = 2;

DELETE clips[2] FROM courses
WHERE course_id = ‘nodejs-big-picture‘ AND module_id = 2;
```

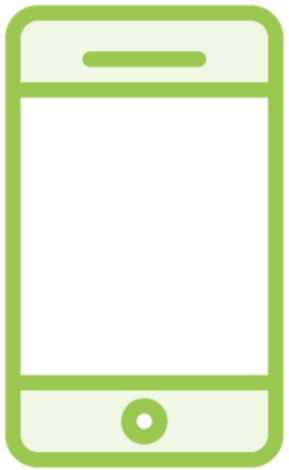
Demo

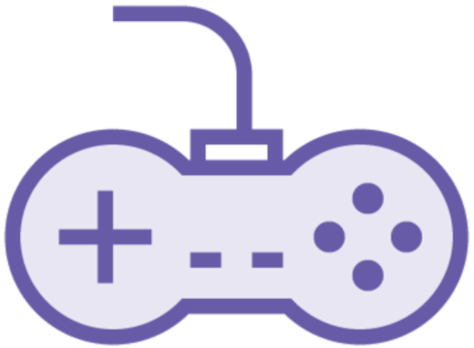
Add course features using a set

Add module clips using a list

Adding and subtracting in collections

Collections: Map





Collections: Map

```
CREATE TABLE users (
    id varchar,
    first_name varchar,
    last_name varchar,
    password varchar,
    reset_token varchar,
    last_login map<varchar,timestamp>,
    PRIMARY KEY (id)
);
```

Collections: Map

Inserting with a map

```
INSERT INTO users (id, first_name, last_name, last_login)
VALUES ('john-doe', 'John', 'Doe',
    {'383cc0867cd2': '2015-06-30 09:02:24'});
```

Updating / adding to a map

```
UPDATE users SET last_login['383cc0867cd2']
  = '2015-07-01 11:17:42' WHERE user_id = 'john-doe';

UPDATE users SET last_login = last_login + {'7eb0a8997f39':
'2015-07-02 07:32:17'} WHERE user_id = 'john-doe';
```

Collections: Map

Removing from a map

```
DELETE last_login['383cc0867cd2'] FROM users
WHERE id = 'john-doe';

UPDATE users SET last_login = last_login - {'7eb0a8997f39'}
WHERE id = 'john-doe';
```

Emptying the entire map

```
UPDATE users SET last_login = {}
WHERE id = 'john-doe';
```

Collections and TTL

```
UPDATE users USING TTL 31536000
SET last_login['383cc0867cd2'] = '2015-07-01 11:17:42'
WHERE user_id = 'john-doe';
```

Demo

Add a map to hold a user's last login

Add a last login with a TTL

Tuples

(varchar, int, int, varchar, timestamp)

Tuples

383cc0867cd2 → 2015-07-01 11:17:42

383cc0867cd2 → 2015-07-01 11:17:42 + 98.203.153.64

Tuples

```
CREATE TABLE users (
    id varchar,
    first_name varchar,
    last_name varchar,
    password varchar,
    reset_token varchar,
    last_login map<varchar,
      frozen<tuple<timestamp,inet>>>,
    PRIMARY KEY (id)
);
```

“Frozen”

- **Nested types are serialized as a single (blob) value**
- **True for nested collections as well (`list<set<varchar>>`)**
- **Nested values must be set or read as a whole**
- `frozen<>` **makes this distinction obvious**
- **Leaves open the possibility of "non-frozen" support in the future**

Demo

Use a tuple to store the last login IP

User Defined Types

Course Overview	1m 10s	
Course Overview	1m 10s	
Considering Node.js	15m 0s	
Course Introduction	3m 4s	
Where Is Node.js Commonly Found?	4m 37s	
What Makes up Node.js?	2m 53s	
A Brief History / When Node May Not Be the Best Fit	4m 25s	

User Defined Types

```
CREATE TYPE clip (name varchar, duration int);

CREATE TABLE courses (
    id varchar,
    author varchar static,
    // ...
    clips list<frozen<clip>>,
    module_id int,
    // ...
    PRIMARY KEY (id, module_id)
);
```

Why Not Just Use a Tuple?

- **Can identify individual components by name (not just order)**
- **More helpful with multiple components of the same type**
- **Start with a Tuple when modeling a User Defined Type**
- **Opportunity to reuse User Defined Type across tables**

User Defined Types

```
CREATE TYPE person (name varchar, id varchar);

CREATE TABLE courses (
    id varchar,
    author frozen<person> static,
    // ...
    clips list<frozen<clip>>,
    module_id int,
    // ...
    PRIMARY KEY (id, module_id)
);
```

Inserting Data with JSON

Inserting data

```
INSERT INTO courses
  (id, module_id, author, clips)
VALUES (‘nodejs-big-picture’, 1,
  {
    name: ‘Paul O''Fallon',
    id: ‘paul-ofallon'
  }, [{
     name: ’Course Overview’,
     duration: 70
  }]
);
```

Inserting data with JSON

```
INSERT INTO courses JSON '{
  "id": "nodejs-big-picture",
  "module_id": 1,
  "author": {
    "name": "Paul O’Fallon",
    "id": "paul-ofallon" },
  "clips": [{
    "name": "Course Overview",
    "duration": 70
  }]
}'
```

Selecting Data with JSON

```
SELECT JSON * FROM courses;
```

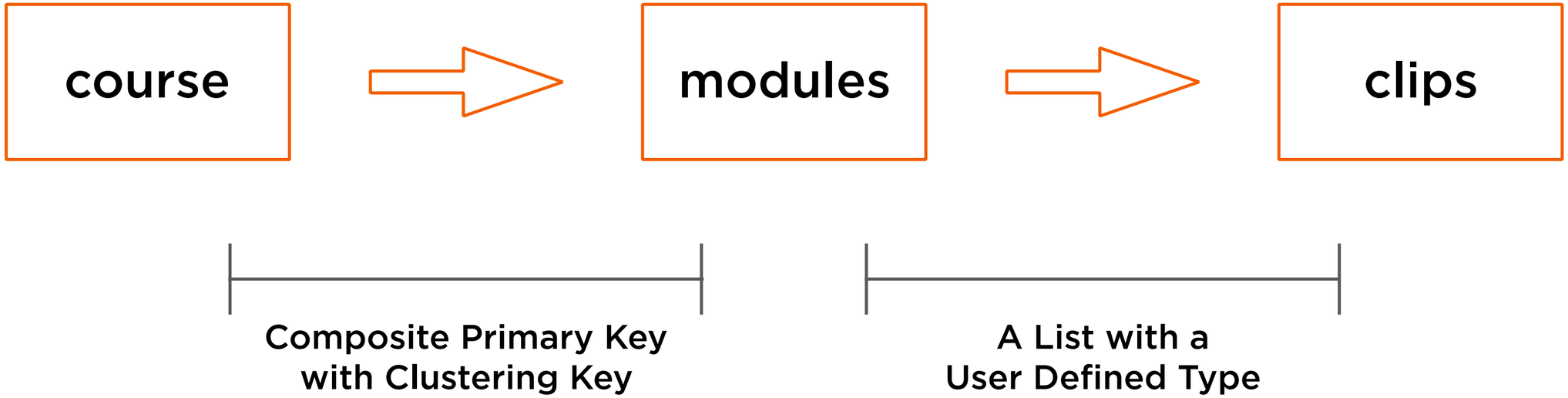
```
SELECT DISTINCT id, name, toJson(released) FROM courses;
```

Demo

Store clips with a user defined type

Select course data as JSON

Courses, Modules and Clips



All in a single partition!

Conclusion

Sets, lists and maps

Collections and TTLs

Tuples

“Frozen” nested complex types

User defined types

Inserting and selecting with JSON