Introduction to Data Management

Lecture #24

SQL  NoSQL

Instructor: Mike Carey
mjcarey@ics.uci.edu

It’s time for the season finale of...

Friday Nights With Databases...

Brought to you by…
Announcements

- Homework info:
  - HW #8 (NoSQL) is the last one!
  - Due Thursday (at 5 PM), with NoLateDay (!)
  - Warning: LOAD can be (path) finicky...
- Endterm exam info:
  - Non-cumulative and in class next Friday (as usual)
- NoSQL lecture plans:
  - Today: NoSQL & Big Data (a la AsterixDB)
    - Refer to the Using SQL++ Primer and other docs on the Apache AsterixDB site
  - Read SQL++ For SQL Users from Couchbase, by Don Chamberlin (the Father of SQL!)
    - Lots of useful info for moving from SQL to SQL++!

What is a NoSQL DB – why “not SQL”?

- Not from the DB world
  - Distributed systems folks
  - Also various startup companies
- From caches → persistent K/V use cases
  - Apps needed massive scale-out
  - OLTP (vs. parallel query DB) apps
  - Simple, low-latency API – get/put by key
  - Need a key K, but want no schema for V
  - Record-level atomicity, replica consistency varies
- In the context of this talk, NoSQL does not mean
  - Hadoop (or SQL on Hadoop)
  - Graph databases or graph analytics platforms
NoSQL Data (JSON-based)

Collection(Orders)
{
  "id": "123",
  "Customer": {
    "custName": "Fred",
    "custCity": "LA"
  }
  "total": 25.97,
  "Items": [
    {
      "product-sku": 401,
      "qty": 2,
      "price": 9.99
    },
    {
      "product-sku": 544,
      "qty": 1,
      "price": 3.99
    }
  ]
}

Collection(Products)
{
  "sku": 401,
  "name": "Garfield T-Shirt",
  "listPrice": 9.99,
  "size": "XL"
},
{
  "sku": 544,
  "name": "USB Charger",
  "listPrice": 5.99,
  "power": "115V"
}

Current NoSQL (document DB) trends

- Popular examples: MongoDB, Couchbase
- Users now coveting the benefits of many DB goodies
  - Secondary indexing and non-key access
  - Declarative queries
  - Aggregates and now (commonly small) joins
- World seems to be heading towards...
  - BDMS (think scalable, OLTP-aimed, parallel/distributed DBMS)
  - Declarative queries and query optimization, applied to schema-less data
  - Return of (some, optional!) schema information
Towards a Big Data Management System (BDMS)

BDMS Desiderata:
- Able to manage data
- Flexible data model
- Full query capability
- Continuous data ingestion
- Efficient and robust parallel runtime
- Cost proportional to task at hand
- Support “Big Data” data types

Apache AsterixDB (from UCI+UCR)

ASTERIX Goal:
To ingest, digest, persist, index, manage, query, analyze, and publish massive quantities of semistructured information...

http://asterixdb.apache.org/
Data Model: JSON (JavaScript Object Notation)

**Customers**

```json
{
  "custid": "C37",
  "name": "T. Hanks",
  "address": {
    "street": "120 Harbor Blvd.",
    "city": "Boston, MA",
    "zipcode": "02115"
  },
  "rating": 750
}

{
  "custid": "C47",
  "name": "S. Lauren",
  "address": {
    "street": "17 Rue d'Antibes",
    "city": "Cannes, France"
  },
  "rating": 625
}
```

**Orders**

```json
{
  "orderno": 1004,
  "custid": "C35",
  "order_date": "2017-07-10",
  "ship_date": "2017-07-15",
  "items": [
    {
      "itemno": 680,
      "qty": 6,
      "price": 9.99
    },
    {
      "itemno": 195,
      "qty": 4,
      "price": 35.00
    }
  ]
}

{
  "orderno": 1008,
  "custid": "C13",
  "order_date": "2017-10-13",
  "items": [
    {
      "itemno": 460,
      "qty": 20,
      "price": 99.99
    }
  ]
}
```

Data from D. Chamberlin, SQL++ for SQL Users: A Tutorial

---

Data from D. Chamberlin, SQL++ for SQL Users: A Tutorial
### Data (Relational version)

**Customers**

```json
{
  "custid": "C37",
  "name": "T. Hanks",
  "address_street": "120 Harbor Blvd.",
  "address_city": "Boston, MA",
  "address_zipcode": "02115",
  "rating": 750
}
{
  "custid": "C47",
  "name": "S. Lauren",
  "address_street": "17 Rue d'Antibes",
  "address_city": "Cannes, France",
  "address_zipcode": null,
  "rating": 625
}
```

**Orders**

```json
{
  "orderno": 1004,
  "custid": "C35",
  "order_date": "2017-07-10",
  "ship_date": "2017-07-15"
}
{
  "orderno": 1008,
  "custid": "C13",
  "order_date": "2017-10-13",
  "ship_date": null
}
```

**Lineitems**

```json
CREATE TABLE Lineitems (  
orderno INTEGER,  
itemno INTEGER,  
quantity INTEGER NOT NULL,  
price DECIMAL(8,2) NOT NULL,  
PRIMARY KEY (orderno, itemno),  
FOREIGN KEY (orderno) REFERENCES Orders(orderno)
)
```

```json
{
  "orderno": 1004,
  "itemno": 680,
  "qty": 6,
  "price": 9.99,
  "currency": "USD"
}
{
  "orderno": 1004,
  "itemno": 195,
  "qty": 4,
  "price": 35.00,
  "currency": "USD"
}
{
  "orderno": 1008,
  "itemno": 460,
  "qty": 20,
  "price": 99.99,
  "currency": "EUR"
}
```
**Sloppy Data**

**Customers**

```
{  
  "custid": "C37",  
  "name": "T. Hanks",  
  "address": {  
    "street": "120 Harbor Blvd.",  
    "city": "Boston, MA",  
    "zipcode": "02115"  
  },  
  "rating": 750
}
{  
  "custid": "C47",  
  "name": "S. Lauren",  
  "address": {  
    "street": "17 Rue d'Antibes",  
    "city": "Cannes, France"  
  },  
  "rating": 625
}
```

**Orders**

```
{  
  "orderno": 1004,  
  "custid": "C35",  
  "order_date": "2017-07-10",  
  "ship_date": "2017-07-15",  
  "items": [  
    {  
      "itemno": 680,  
      "qty": 6,  
      "price": 9.99
    },  
    {  
      "itemno": 195,  
      "qty": 4,  
      "price": "if you have to ask ..."
    }
  ]
}
```

**SQL++: Just like SQL ...**

```sql
SELECT name  
FROM customers  
WHERE rating > 650;
```

```
[  
  {  
    "name": "M. Streep"
  },  
  {  
    "name": "T. Hanks"
  },  
  {  
    "name": "T. Cruise"
  }
]```
Just like SQL ...

```sql
SELECT name
FROM customers
WHERE rating > 650;
```

```json
[
  {
    "name": "R. Duvall",
    "order_date": "2017-09-02"
  },
  {
    "name": "R. Duvall",
    "order_date": "2017-04-29"
  }
]
```

Just like SQL ...

```sql
SELECT c.name, o.order_date
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
  AND c.custid = "C41";
```

```sql
SELECT c.name, o.order_date
FROM customers AS c
LEFT OUTER JOIN
orders AS o
ON c.custid = o.custid
WHERE c.custid = "C41";
```
Just like SQL ...

```
SELECT name
FROM customers
WHERE rating > 650;

SELECT c.name, o.order_date
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
  AND c.custid = "C41";

SELECT order_date, count(*) AS cnt
FROM orders
GROUP BY order_date
HAVING count(*) > 0
ORDER BY order_date DESC
LIMIT 3;
```

... almost!

```
SELECT name, order_date
FROM customers, orders
WHERE customers.custid = orders.custid
  AND rating > 650;
```

Cannot resolve ambiguous alias reference for identifier rating (in line 4, at column 7)
... almost!

```
SELECT name, order_date
FROM customers, orders
WHERE customers.custid = orders.custid
    AND rating > 650;

SELECT name, c.order_date
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
    AND c.rating > 650;
```

... almost!

```
[  
  {
    "name": "T. Hanks",
    "order_date": "2017-08-30"
  },
  {
    "name": "T. Cruise",
    "order_date": "2017-05-01"
  },
  {
    "name": "T. Cruise",
    "order_date": "2017-10-13"
  },
  {
    "name": "T. Cruise",
    "order_date": "2017-09-13"
  }
]

SELECT c.name, o.order_date
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
    AND c.rating > 650;

SELECT *  
FROM customers AS c, orders AS o  
WHERE c.custid = o.custid  
    AND c.rating > 650;
```

[  
  "c": {  
    "address": {
      "city": "Boston, MA",
      "street": "120 Harbor Blvd.",
      "zipcode": "02115"
    },
    "custid": "C37",
    "name": "T. Hanks",
    "rating": 750
  },  
  "o": {
    "custid": "C37",
    "items": [
      {
        "itemno": 460,
        "price": 99.98,
        "qty": 2
      }
    ]
  }
]...
SELECT VALUE name
FROM customers
WHERE rating > 650;

[  
  "M. Streep",
  "T. Hanks",
  "T. Cruise"
]

SELECT VALUE {
  "CustomerName":c.name,
  "OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

[  
  {  
    "CustomerName": "T. Hanks",
    "OrderDate": "2017-08-30"
  },
  {  
    "CustomerName": "T. Cruise",
    "OrderDate": "2017-09-13"
  },
  {  
    "CustomerName": "T. Cruise",
    "OrderDate": "2017-05-01"
  },
  {  
    "CustomerName": "T. Cruise",
    "OrderDate": "2017-10-13"
  }
]
SELECT VALUE name
FROM customers
WHERE rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT c.name AS CustomerName,
    o.order_date AS OrderDate
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;

SELECT VALUE {
"CustomerName":c.name,
"OrderDate":o.order_date
}
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
AND c.rating > 650;
Quiz

Which query retrieves the names of the customers that have the highest rating?

A

```
SELECT name
FROM customers
WHERE rating =
  (SELECT MAX(rating) FROM customers);
```

B

```
SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2);
```

C

```
SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT MAX(c2.rating) FROM customers AS c2);
```

D

```
SELECT VALUE c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2)[0];
```

SQL Pitfalls and the value of VALUE

```
SELECT name
FROM customers
WHERE rating =
  (SELECT MAX(rating) FROM customers);
```

Type mismatch: expected value of type multiset or array, but got the value of type object (in line 4, at column 28)
SQL Pitfalls and the value of VALUE

SELECT name
FROM customers AS c
WHERE rating =
(SELECT MAX(rating) FROM c);

Type mismatch: expected value of type multiset or array, but got the value of type object (in line 4, at column 28)

SQL Pitfalls and the value of VALUE

SELECT name
FROM customers
WHERE rating =
(SELECT MAX(rating) FROM customers);

SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
(SELECT MAX(c2.rating) FROM customers AS c2);

[ ]
SQL Pitfalls and the value of VALUE

SELECT name
FROM customers
WHERE rating =
  (SELECT MAX(rating) FROM customers);

SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT MAX(c2.rating) FROM customers AS c2);

SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2);

SELECT VALUE c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2)[0];

SELECT name
FROM customers
WHERE rating =
  (SELECT MAX(rating) FROM customers);

SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT MAX(c2.rating) FROM customers AS c2);

SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2);

SELECT VALUE c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2)[0];
Quiz

Which query retrieves the names of the customers that have the highest rating?

A

```sql
SELECT name
FROM customers
WHERE rating =
  (SELECT MAX(rating) FROM customers);
```

B

```sql
SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2);
```

C

```sql
SELECT c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT MAX(c2.rating) FROM customers AS c2);
```

D

```sql
SELECT VALUE c1.name
FROM customers AS c1
WHERE c1.rating =
  (SELECT VALUE MAX(c2.rating) FROM customers AS c2)[0];
```

To be continued....
More information about JSON, SQL++, and AsterixDB

- Asterix project UCI/UCR research home
  - http://asterix.ics.uci.edu/
- Apache AsterixDB home
  - http://asterixdb.apache.org/
- SQL++ Primer
- Navigate from CS122a wiki (HW) to get and install it...!
  - Also, a few other resources and hints in the HW materials