Introduction to Data Management

Lecture #24

SQL NoSQL (cont.)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- Last homework reminder:
  - Due this Thursday (at 5 PM), NoSQL with NoLateDay
  - And remember: LOAD can be path-finicky (see Piazza)
- Endterm exam info:
  - Non-cumulative and in class on Friday!
- Two-part lecture season finale:
  - Today: NoSQL & Big Data (a la AsterixDB), continued
    - See the Using SQL++ Primer and the Don Chamberlin SQL++ book
  - Wednesday: Transactions (a whirlwind tour)
    - See the corresponding textbook sections on the wiki page
**Data Model:** JSON (from last time...)

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": "2017-07-13",
  "items": {
    "itemno": 680,
    "qty": 6,
    "price": 9.99
  },
  "items": {
    "itemno": 195,
    "qty": 4,
    "price": 35.00
  }
}

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

**NESTED DATA:** Nesting

```sql
SELECT VALUE {
  "CustomerName": c.name,
  "Orders": (SELECT VALUE o.orderno FROM orders AS o
              WHERE o.custid = c.custid)
} FROM customers AS c
WHERE c.custid = "C41";
```

```sql
[
{
  "Orders": [
    1006,
    1001
  ],
  "CustomerName": "R. Duvall"
}
]```
Unnesting

SELECT o.orderno,
o.order_date,
i.itemno AS item_number,
i.qty AS quantity
FROM orders AS o UNNEST o.items AS i
WHERE i.qty > 100
ORDER BY o.orderno, item_number;

[{
  "orderno": 1002,
  "order_date": "2017-05-01",
  "item_number": 680,
  "quantity": 150
},
  {
  "orderno": 1005,
  "order_date": "2017-08-30",
  "item_number": 347,
  "quantity": 120
},
  {
  "orderno": 1006,
  "order_date": "2017-09-02",
  "item_number": 460,
  "quantity": 120
}]

Unnesting (cont.)

SELECT o.orderno,
o.order_date,
i.itemno AS item_number,
i.qty AS quantity
FROM orders AS o UNNEST o.items AS i
WHERE i.qty > 100
ORDER BY o.orderno, item_number;

SELECT o.orderno,
o.order_date,
i.itemno AS item_number,
i.qty AS quantity
FROM orders AS o, o.items AS i
WHERE i.qty > 100
ORDER BY o.orderno, item_number;
Quantification

```sql
SELECT DISTINCT VALUE o.custid
FROM orders AS o
WHERE SOME i IN o.items SATISFIES i.price >= 25.00;

[ "C37",
  "C41",
  "C31",
  "C35",
  "C13"
]
```

Quantification

```sql
SELECT DISTINCT VALUE o.custid
FROM orders AS o
WHERE EVERY i IN o.items SATISFIES i.price >= 25.00;
```
Quantification

SELECT DISTINCT VALUE o.custid
FROM orders AS o
WHERE SOME i IN o.items SATISFIES i.price >= 25.00;

SELECT DISTINCT VALUE o.custid
FROM orders AS o
WHERE EVERY i IN o.items SATISFIES i.price >= 25.00;

SELECT DISTINCT VALUE o.custid
FROM orders AS o
WHERE EVERY i IN o.items SATISFIES i.price >= 25.00
AND array_count(o.items) > 0;

SELECT VALUE c
FROM customers AS c
WHERE c.custid IN (
    SELECT DISTINCT VALUE o.custid
    FROM orders AS o
    WHERE SOME i IN o.items SATISFIES i.price >= 25.00
)

GROUPING: SQL Grouping and Aggregation

```sql
SELECT c.address.city, count(*) AS cnt
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
GROUP BY c.address.city
```

```
[{
  "cnt": 2,
  "city": "Boston, MA"
},
{
  "cnt": 6,
  "city": "St. Louis, MO"
}]
```
SELECT c.name, array_count(o.items) AS order_size
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
ORDER BY order_size DESC
LIMIT 3

SELECT VALUE max(rating) FROM customers

[{
  "order_size": 4,
  "name": "T. Hanks"
},
{
  "order_size": 3,
  "name": "R. Duvall"
},
{
  "order_size": 2,
  "name": "R. Duvall"
}]

750
SQL++ Aggregation (only)

```sql
SELECT c.name, array_count(o.items) AS order_size
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
ORDER BY order_size DESC
LIMIT 3

SELECT VALUE max(rating) FROM customers
array_max((SELECT VALUE rating FROM customers))
```

SQL++ Grouping (only)

```sql
SELECT c.address.city, g
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
GROUP BY c.address.city GROUP AS g;
[
 { "city": "Boston, MA", "g": [
  { "c": { "address": { "city": "Boston, MA", ... }, "custid": "C35", "name": "J. Roberts", "rating": 565 }
  , "o": { "custid": "C35", "items": [ { "itemno": 680, "price": 9.99, "qty": 6 } ] , "order_date": "2017-07-10", "orderno": 1004, "ship_date": "2017-07-15" }
  ]
  , "c": { "address": { "city": "Boston, MA", ... }, "custid": "C37", "name": "T. Hanks", "rating": 750 }
  , "o": { "custid": "C37", "items": [ { "itemno": 460, "price": 99.98, "qty": 2 }, { "itemno": 347, "price": 22, "qty": 120 }, { "itemno": 780, "price": 1500, "qty": 1 } , { "itemno": 375, "price": 149.98, "qty": 2 } ] , "order_date": "2017-08-30", "orderno": 1005 }

]
SQL Grouping and Aggregation Explained

```
SELECT c.address.city, count(*) AS cnt
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
GROUP BY c.address.city

SELECT c.address.city, count(*) AS cnt
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
GROUP BY c.address.city

SELECT c.address.city, array_count(g) AS cnt
FROM customers AS c, orders AS o
WHERE c.custid = o.custid
GROUP BY c.address.city, GROUP AS g;
```
MISSING INFORMATION: Remember the data from earlier...

Customers

- custid: "C37"
- name: "T. Hanks"
- address: "120 Harbor Blvd., Boston, MA, 02115"
- rating: 750

- custid: "C47"
- name: "S. Lauren"
- address: "17 Rue d'Antibes, Cannes, France"
- rating: 625

Orders

- orderno: 1008
- custid: "C13"
- order_date: "2017-10-13"
- items:
  - itemno: 460
  - qty: 20
  - price: 99.99

- 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

Have I "missed" anything?

```sql
SELECT o.orderno, o.order_date, o.ship_date, o.custid
FROM orders o
WHERE o.ship_date IS MISSING
```

Data from D. Chamberlin. SQL++ for SQL Users: A Tutorial
SELECT o.orderno, o.order_date, o.ship_date, o.custid FROM orders o WHERE o.ship_date IS MISSING

SELECT VALUE {
  "orderno": o.orderno,
  "order_date": o.order_date,
  "ship_date": o.ship_date,
  "custid": o.custid
}
FROM orders o WHERE o.ship_date IS MISSING

... WHERE o.ship_date IS NOT MISSING
... WHERE o.ship_date IS UNKNOWN
... WHERE o.ship_date IS NULL
...

Have I "missed" anything?

SELECT o.orderno, o.order_date, o.ship_date, o.custid FROM orders o WHERE o.ship_date IS MISSING

SELECT VALUE {
  "orderno": o.orderno,
  "order_date": o.order_date,
  "ship_date": o.ship_date,
  "custid": o.custid
}
FROM orders o WHERE o.ship_date IS MISSING

... WHERE o.ship_date IS NOT MISSING
... WHERE o.ship_date IS UNKNOWN
... WHERE o.ship_date IS NULL
...
Dealing with different "cases"

```sql
SELECT VALUE {
    "orderno": o.orderno,
    "order_date": o.order_date,
    "ship_date":
        CASE
            WHEN o.ship_date IS MISSING THEN "TBD"
            ELSE o.ship_date
        END,
    "custid": o.custid
} FROM orders o
ORDER BY ship_date DESC
```

More information about JSON, SQL++, and AsterixDB

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