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), no late wiggle room this time.

• Midterm #2 info:
  – Average this time was “a little lower” (oops!)
    • Don’t fret; that’s exactly what curves are for…!

• NoSQL lecture plans:
  – Today: NoSQL & Big Data (a la AsterixDB)
    • Not in book, so: See AsterixDB paper on the wiki!
    • Refer to the primer & docs on the Apache AsterixDB site.
  – Also see SQL++ For SQL Users from Couchbase, written by Don Chamberlin (the Father of SQL!):
    • Be sure to read it (since it’s fair game for the Endterm 😊).
    • Lots of useful info for moving from SQL to SQL++!

AsterixDB: “One Size Fits a Bunch”

BDMS Desiderata:
• Flexible data model
• Efficient runtime
• Full query capability
• Cost proportional to task at hand (!)
• Designed for continuous data ingestion
• Support today’s “Big Data data types”
• • •
Project Goals

• Build a new Big Data Management System (BDMS)
  – Run on large commodity clusters
  – Handle mass quantities of semistructured data
  – Openly *layered*, for selective reuse by others
  – Share with the community via *open source* (ASF)

• Conduct scalable information systems research, e.g.,
  – Large-scale query processing and workload management
  – Highly scalable storage and index management
  – Fuzzy matching, spatial data, date/time data (all in parallel)
  – Novel support for “fast data” (flowing both in and out)

• Train the next generation of “Big Data” graduates

ASTERIX Data Model (ADM)

```java
create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotUserType as {
id: int32,
alias: string,
name: string,
userSince: datetime,
address: {
  street: string,
city: string,
state: string,
zip: string,
country: string
},
friendIds: [{ int32 }],
employment: [EmploymentType]
};
create type EmploymentType as open {
  organizationName: string,
  startDate: date,
  endDate: date?
};
create dataset MugshotUsers(MugshotUserType) primary key id;
```

**Highlights include:**
• JSON++ based data model
• Rich type support (spatial, temporal, …)
• Records, lists, bags
• *Open vs. closed types*
create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotUserType as {
  id: int32
};
create type EmploymentType as open {
  organizationName: string,
  startDate: date,
  endDate: date?
};
create dataset MugshotUsers(MugshotUserType)
  primary key id;

Highlights include:
• JSON++ based data model
• Rich type support (spatial, temporal, …)
• Records, lists, bags
• Open vs. closed types

create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotMessageType as closed {
  messageId: int32,
  authorId: int32,
  timestamp: datetime,
  inResponseTo: int32?,
  senderLocation: point?,
  tags: [{ string }],
  message: string
};
create dataset MugshotMessages(MugshotMessageType)
  primary key messageId;

Highlights include:
• JSON++ based data model
• Rich type support (spatial, temporal, …)
• Records, lists, bags
• Open vs. closed types
Ex: MugshotUsers Data

```json
{ "id":1, "alias":"Margarita", "name":"MargaritaStoddard", "address":{
  "street":"234 Thomas Ave", "city":"San Hugo", "zip":"98765",
  "state":"CA", "country":"USA" },
  "nickname":"Mags", "gender":"F",
  "userSince":datetime("2012-08-20T10:10:00"),
  "friendIds":{[2, 3, 6, 10]}, "employment":{
    "organizationName":"Codetechno", "startDate":date("2006-08-06") }} },

{ "id":2, "alias":"Isbel", "name":"IsbelDull", "address":{
  "street":"345 James Ave", "city":"San Hugo", "zip":"98765",
  "state":"CA", "country":"USA" },
  "userSince":datetime("2011-01-22T10:10:00"),
  "friendIds":{[1, 4]}, "employment":{
    "organizationName":"Hexviafind", "startDate":date("2010-04-27") }} },

{ "id":3, "alias":"Emory", "name":"EmoryUnk", "address":{
  "street":"456 Jose Ave", "city":"San Hugo", "zip":"98765",
  "state":"CA", "country":"USA" },
  "userSince":datetime("2012-07-10T10:00"),
  "friendIds":{[1, 5, 8, 9]}, "employment":{
    "organizationName":"geomeida",
    "startDate":date("2010-06-17"), "endDate":date("2010-01-26") }} }
```

Other DDL Features

```sql
create index msUserSinceIdx on MugshotUsers(userSince);
create index msTimestampIdx on MugshotMessages(timestamp);
create index msAuthorIdx on MugshotMessages(authorId) type btree;
create index msSenderLocIndex on MugshotMessages(senderLocation) type rtree;
create index msMessageIdx on MugshotMessages(message) type keyword;

create type AccessLogType as closed
  { ip: string, time: string, user: string, string, `path`: string, stat: int32, size: int32 };
create external dataset AccessLog(AccessLogType) using localfs
  ("path"="[hostname]://[path]", ("format"="delimited-text", ("delimiter"="|"));
create feed mySocketFeed using socket_adaptor
  ("sockets"="[address]:[port]", ("addressType"="IP"),
  ("type-name"="MugshotMessageType"), ("format"="adm"));
connect feed mySocketFeed to dataset MugshotMessages;
```

External data highlights:
- Equal opportunity access
- "Keep everything!"
- Data ingestion, not streams
ASTERIX Queries (SQL++ or AQL)

- **Q1:** List the user name and messages sent by those users who joined the Mugshot social network in a certain time window:

```sql
select user.name as uname,
       (select value msg.message
        from MugshotMessages msg
        where msg.authorId = user.id) as messages
from MugshotUsers user
where user.userSince >= datetime('2010-07-22T00:00:00')
    and user.userSince <= datetime('2012-07-29T23:59:59');
```

```json
{ "uname": "IsbelDull", "messages": [ "like samsung the plan is amazing", "like t-mobile its platform is mind-blowing" ] }
{ "uname": "EmoryUnk", "messages": [ "love sprint its shortcut-menu is awesome:)", ... ] }
```

SQL++ (cont.)

- **Q2:** Identify active users and group/count them by country:

```sql
with endTime as current_datetime(),
     startTime as endTime - duration("P30D")
select user.address.country as country, count(users) as activeUsers
from MugshotUsers user
where some logrec in AccessLog satisfies
    user.alias = logrec.user
    and datetime(logrec.time) >= startTime
    and datetime(logrec.time) <= endTime
group by user.address.country;
```

**SQL++ highlights:**
- Lots of other features (see website!)
- Spatial predicates and aggregation
- Set-similarity (“fuzzy”) matching
Updates and Transactions

- **Q3:** Add a new user to Mugshot.com:

```sql
insert into MugshotUsers
(
    "id" : 11, "alias" : "John",
    "name" : "JohnDoe",
    "userSince" : datetime("2012-08-20T10:00:00Z"),
    "address" : {
        "street" : "789 Jane St",
        "city" : "San Harry", "state" : "CA",
        "zip" : "98767", "country" : "USA"
    },
    "friendIds" : { 5, 9, 11 },
    "employment" : [ { "organizationName" : "Kongreen", "startDate" : date("2009-08-11") } ]
);
```

- Key-value store-like transactions (w/record level atomicity)
- Insert, delete, and upsert ops; index-consistent
- 2PL concurrency
- WAL no-steal, no-force with LSM shadowing

---

AsterixDB Cluster Overview

- **Data Loads and Feeds**
- **AQL queries and results**
- **Data publishing**

---

AsterixDB SQL++
Example AsterixDB Use Cases

- Potential use case areas include
  - Social data analytics
  - Cell phone event analytics
  - Behavioral science
  - Education
  - Public health
  - Power usage monitoring
  - Cluster management log analytics
  - ....
Current Status

• 4 year initial NSF project (250+ KLOC), started 2009
• Now officially Apache AsterixDB!
  – Semistructured “NoSQL” style data model
  – Declarative parallel queries, inserts, deletes, ...
  – Data storage/indexing (primary & secondary, LSM-based)
  – Internal and external datasets both supported
  – Rich set of data types (including text, time, location)
  – Fuzzy and spatial query processing
  – NoSQL-like transactions (for inserts/deletes)
  – Data feeds and indexes for external datasets
  – ....

For More Information

• Asterix project UCI/UCR research home
  – http://asterix.ics.uci.edu/
• Apache AsterixDB home
  – http://asterixdb.apache.org/
• SQL++ Primer
  – http://asterixdb.apache.org/docs/0.9.3/index.html
• Navigate from CS122a wiki (HW) to get and install it!
  – A few other resources and hints in the HW materials.

QUESTIONS...?