Oracle Big Data SQL

Architectural Deep Dive

Dan McClary, Ph.D.
Big Data Product Management
Oracle
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

1. The Data Analytics Challenge
2. Why Unified Query Matters
3. SQL on Hadoop and More: Unifying Metadata
4. Query Franchising: Smart Scan for Hadoop
Data Analytics Challenge (2012)

Separate data access interfaces
Tables on NoSQL

SQL on Hadoop
Data Analytics Challenge: 2013

No comprehensive SQL interface across Oracle, Hadoop and NoSQL
Oracle Big Data Management System

Unified SQL access to all enterprise data

CQL  find()  UnQL  SQL  sql

NoSQL  ORACLE
How to Get the Most From "Polyglot Persistence"
SQL on Hadoop and More: Unifying Metadata
Why Unify Metadata?

Query across sources ➔ Integrate new metadata
• No changes for users and applications
  • Seamlessly handle schema-on-read
  • Exploit remote data distribution
  • Holistically optimize queries

CREATE TABLE customers...
SELECT name FROM customers

CREATE TABLE sales...
SELECT customers.name FROM customers

SQL
How Data is Stored in HDFS

Example: 1TB File

{ "custId":1185972, "movieId":null, "genreId":null, "time":"2012-07-01:00:00:07", "recommended":null, "activity":8 }  
{ "custId":1354924, "movieId":1948, "genreId":9, "time":"2012-07-01:00:00:22", "recommended":"N", "activity":7 }  
{ "custId":1083711, "movieId":null, "genreId":null, "time":"2012-07-01:00:00:26", "recommended":null, "activity":9 }  
{ "custId":1234182, "movieId":11547, "genreId":44, "time":"2012-07-01:00:00:32", "recommended":"Y", "activity":7 }  
{ "custId":1010220, "movieId":11547, "genreId":44, "time":"2012-07-01:00:00:42", "recommended":"Y", "activity":6 }  
{ "custId":1143971, "movieId":null, "genreId":null, "time":"2012-07-01:00:00:43", "recommended":null, "activity":8 }  
{ "custId":1253676, "movieId":null, "genreId":null, "time":"2012-07-01:00:00:50", "recommended":null, "activity":9 }  
{ "custId":1351777, "movieId":608, "genreId":6, "time":"2012-07-01:01:00:03", "recommended":"N", "activity":7 }  
{ "custId":1143971, "movieId":null, "genreId":null, "time":"2012-07-01:00:01:07", "recommended":null, "activity":9 }  
{ "custId":1363545, "movieId":27205, "genreId":9, "time":"2012-07-01:00:01:18", "recommended":"Y", "activity":7 }  
{ "custId":1067283, "movieId":1124, "genreId":9, "time":"2012-07-01:00:01:26", "recommended":"Y", "activity":7 }  
{ "custId":1126174, "movieId":16309, "genreId":9, "time":"2012-07-01:00:01:35", "recommended":"N", "activity":7 }  
{ "custId":1234182, "movieId":11547, "genreId":44, "time":"2012-07-01:00:01:39", "recommended":"Y", "activity":7 } }  
{ "custId":1346299, "movieId":424, "genreId":1, "time":"2012-07-01:00:05:02", "recommended":"Y", "activity":4 }  

Example File = 4096 blocks  
InputSplits = 4096  
Potential scan parallelism
How MapReduce and Hive Read Data

- Scan and row creation needs to be able to work on “any” data format
- Data definitions and column deserializations are needed to provide a table

RecordReader => Scans data (keys and values)
InputFormat => Defines parallelism
SerDe => Makes columns
Metastore => Maps DDL to Java access classes
SQL-on-Hadoop Engines Share Metadata, not MapReduce

Hive Metastore

Oracle Big Data SQL

SparkSQL

Hive

Impala

...

Table Definitions:
movieapp_log_json
Tweets
avro_log

Hive Metastore

Metastore maps DDL to Java access classes
CREATE TABLE movielog (  
  click VARCHAR2(4000))  
ORGANIZATION EXTERNAL (  
  TYPE ORACLE_HIVE  
  DEFAULT DIRECTORY DEFAULT_DIR  
  ACCESS PARAMETERS  
  (  
    com.oracle.bigdata.tablename logs  
    com.oracle.bigdata.cluster mycluster  
  ))  
REJECT LIMIT UNLIMITED;

• New types of external tables
  – ORACLE_HIVE (inherit metadata)
  – ORACLE_HDFS (specify metadata)

• Access parameters for Big Data
  – Hadoop cluster
  – Remote Hive database/table
    • DBMS_HADOOP Package for automatic import
-- these are just tweets from dataset
-- oracle can use the fastest possible processing on this data.

CREATE TABLE TWEETS
(
    TWEET VARCHAR2(4000)
)

ORGANIZATION EXTERNAL
(
    TYPE ORACLE_HIVE
    DEFAULT DIRECTORY DEFAULT_DIR
    ACCESS PARAMETERS
    (   com.oracle.bigdata.datamode=c
    )
    REJECT LIMIT UNLIMITED;

select tweet from tweets where rownum < 50;

select t.tweet.interaction.author.username,
    t.tweet.twitter."user".followers_count as followers,
    t.tweet.saliency.content.sentiment as sentiment,
    t.tweet.twitter."user"."time_zone" as tz
from tweets t
where t.tweet.twitter."user"."time_zone" is not null
and t.tweet.saliency.content.sentiment is not null;
A Smarter External Table

Oracle Table

You define:
• Table name
• Oracle types
• Any Degree of Parallelism

HDFS Data

You get:
• Automatic discovery of Hive table metadata
• Automatic translation from Hadoop types
• Automatic conversion from any InputFormat
• Fan-out Parallelism across the Hadoop cluster
StorageHandlers: Extensibility Beyond HDFS

Oracle Big Data SQL

StorageHandlers are a *metadata bridge*.
CREATE TABLE FLUSHOTS
    ( "MID" VARCHAR2(100 BYTE),
      "PERSON_COUNT" NUMBER,
      "WEEK" NUMBER,
      "name" VARCHAR2(100 BYTE),
      "SHORT_NAME" VARCHAR2(2 BYTE),
      "FIPS_ID" NUMBER,
      "DISPARITY" VARCHAR2(100 BYTE),
      "WEEK_START" VARCHAR2(15 BYTE),
      "MEDICARE_STATUS" VARCHAR2(10 BYTE),
      "year" NUMBER,
      "PERCENTAGE" NUMBER,
      "ETHNICITY" VARCHAR2(1 BYTE) )
    ORGANIZATION EXTERNAL
    ( TYPE ORACLE_HIVE
      DEFAULT DIRECTORY "DEFAULT_DIR"
      ACCESS PARAMETERS ()
    )
    REJECT LIMIT UNLIMITED ;

SELECT * FROM FLUSHOTS WHERE "year" > 2013;

SELECT all_eligible - lag(all_eligible) over (partition
    may redeemed - lag(max redeemed) over(partition by t
    to

Query Franchising: Smart Scan for Hadoop
Language-level Federation Fails
Been there, done that.

Hadoop Part

```sql
with sites as
(
  select s.custid as cust_id,
       listagg(s.site, ',')
     within group (order by s.custid) site_list
  from shortcodes s
  group by custid
),

select c.first_name,
       c.last_name, c.AGE,
       c.state_province, s.site_list
from customers c, sites s;
```

Database Part
Language-level Federation Fails

Been there, done that.

```sql
select s.custid as cust_id,
    listagg(s.site, ',')
    within group (order by s.custid) site_list
from shortcodes s
    group by custid
```

• Operators exist in both places?
• Is their behavior consistent?
• How do you negotiate resources?
Query Franchising – dispatch of query processing to self-similar compute agents on disparate systems without loss of operational fidelity
Query Franchising: Uniform Behavior, Disparate Location

1. Top-level plan created
   - Holistic plan plan for all work
   - Distribute to franchises based by location

2. Franchisees carry out local work
   - Franchises secure and utilize resources
   - All franchises speak the internal language

3. Global operations optimized
   - Adapts to local variation
   - Nothing “lost in translation”
SELECT name, SUM(purchase) 
FROM customers 
GROUP BY name;

1. Oracle SQL query issued
   - Plan constructed
   - Query executed

2. Smart Scan Works on Storage
   - Filter out unneeded rows
   - Project only queried columns
   - Score data models
   - Bloom filters to speed up joins
Big Data SQL: A New Hadoop Processing Engine

Processing Layer
- MapReduce and Hive
- Spark
- Impala
- Se... (Big Data SQL)

Resource Management (YARN, cgroups)

Storage Layer
- Filesystem (HDFS)
- NoSQL Databases (Oracle NoSQL DB, Hbase)
Smart Scan for Hadoop: Optimizing Performance

"Oracle on top"
- Apply filter predicates
- Project columns
- Parse semi-structured data

"Hadoop on the bottom"
- Work close to the data
- Schema-on-read with Hadoop classes
- Transformation into Oracle data stream
Mapping Hadoop to Oracle
Parallel Query and Hadoop

1. Determine Hadoop Parallelism
   - Determine schema-for-read
   - Determine InputSplits
   - Arrange splits for best performance

2. Map to Oracle Parallelism
   - Map splits to granules
   - Assign granules to PX Servers

3. PX Servers Route Work
   - Offload work to Big Data SQL Servers
   - Aggregate
   - Join
   - Apply PL/SQL
Big Data SQL 1.1 Introduces Enhanced Parallelism

Big Data SQL 1.0

• Hadoop DoP linked to RDBMS DoP
  – Lead to many idle PQ processes
  – Required explicit declaration

Big Data SQL 1.1

• Unlink Hadoop and RDBMS DoP
• Automatic max Hadoop parallelism
  – Even on serial tables
• An average of 40% faster
  – Even at equivalent DoP
Big Data SQL Dataflow

1. Read data from HDFS Data Node
   - Direct-path reads
   - C-based readers when possible
   - Use native Hadoop classes otherwise

2. Translate bytes to Oracle

3. Apply Smart Scan to Oracle bytes
   - Apply filters
   - Project Columns
   - Parse JSON/XML
   - Score models
from tweets t
where t.tweet.twitter."user"."time_zone" is not null
and t.tweet.saliency.content.sentiment is not null;

-- how many tweets do we have? how many tweets have anything to do with the flu?
select count(*) as total_tweets from tweets union all
select count(*) as flu_tweets from tweets t where t.tweet.interaction."content" like 'flu%';

-- let's figure out where people were tweeting about the flu
select /*+ parallel(64) */
  count(*) as total_tweets,
  sum(t.tweet.twitter."user".followers_count) as total_followers,
  t.tweet.twitter."user"."time_zone" as tz
from tweets t
where t.tweet.twitter."user"."time_zone" is not null
and t.tweet.interaction."content" LIKE 'flu%'
group by t.tweet.twitter."user"."time_zone"
order by total_tweets desc;

-- let's dig in: while there were a lot of flu-related tweets in the US, how does that compare with the --number of cases of flu in the US that same year? -- to figure this out, we'll need to join data
But How Does Security Work?

```
DBMS_REDACT.ADD_POLICY(
    object_schema => 'MCLICK',
    object_name => 'TWEET_V',
    column_name => 'USERNAME',
    policy_name => 'tweet_redaction',
    function_type => DBMS_REDACT.PARTIAL,
    function_parameters => 'VVVVVVVVVVVVVVVVVVVVVVVVV,*,3,25',
    expression => '1=1'
);
```

1. **Database security for query access**
   - Virtual Private Databases
   - Redaction
   - Audit Vault and Database Firewall

2. **Hadoop security for Hadoop jobs**
   - Kerberos Authentication
   - Apache Sentry (RBAC)
   - Audit Vault

3. **System-specific encryption**
   - Database tablespace encryption
   - BDA On-disk Encryption
SQL, Everywhere

Futures
More Lessons from Exadata

Move less data → Go faster

1. Storage Indexes
   - Skip reads on irrelevant data
   - Big Hadoop Blocks ~ Big Speed Up

2. Caching
   - Cache frequently accessed columns
   - HDFS Caching
Try it out: Oracle Big Data Lite

Hardware and Software
Engineered to Work Together
Fan-Out Parallelism Approach #1
Schema-on-Write, Small blocks
Fan-Out Parallelism Approach #2
Schema-on-Read, Large Blocks