

TERADATA[®] ASTER

Technical Overview of Aster Jun 26th, 2012

Karthik Guruswamy
Yushu Yao

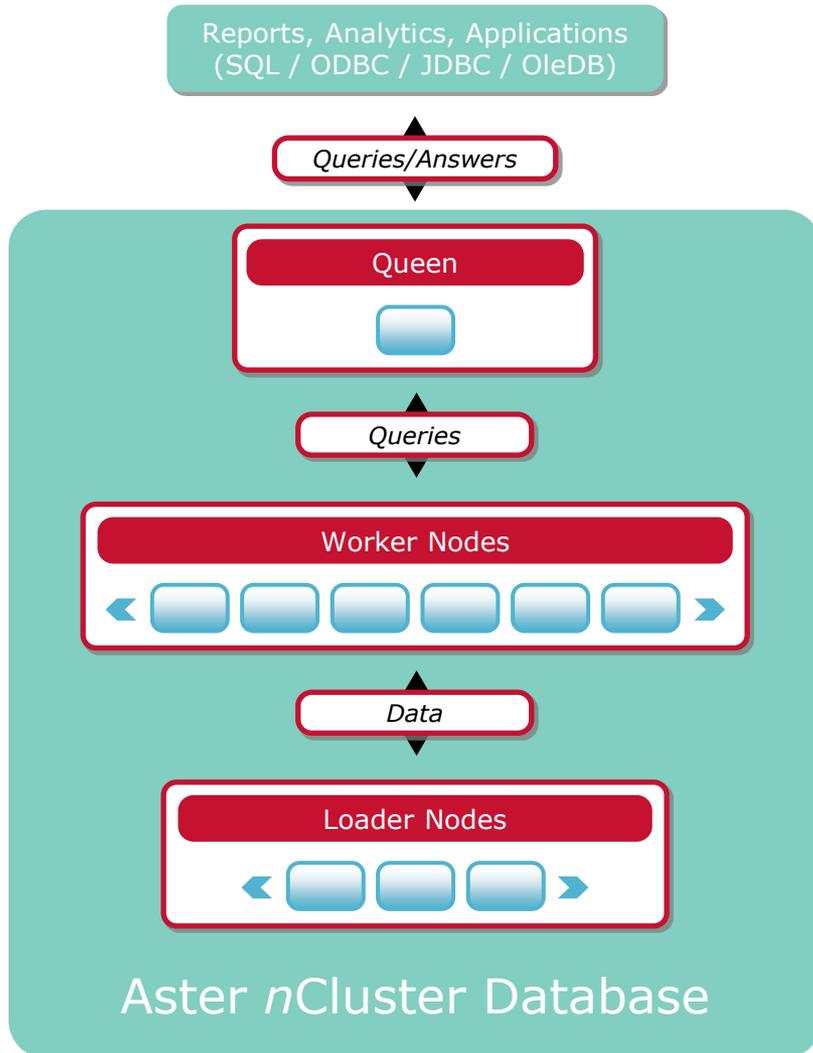
Agenda

- 8:45-9:00 Setup access to NERSC Aster
- 9:00-10:00 Introduction to Aster system structure
- 10:00-11:00 Basic Aster Data Usage
- 11:00-12:00 Best practice to manage your data in Aster
- 12:00-13:00 Working Lunch, Hand-on Session
- 13:00-14:00 Introduction to nCluster
- 14:00-15:00 Introduction to SQL-MapReduce
- 15:00-17:00 Hand on session and Q&A

TERADATA[®] ASTER

Architecture Overview

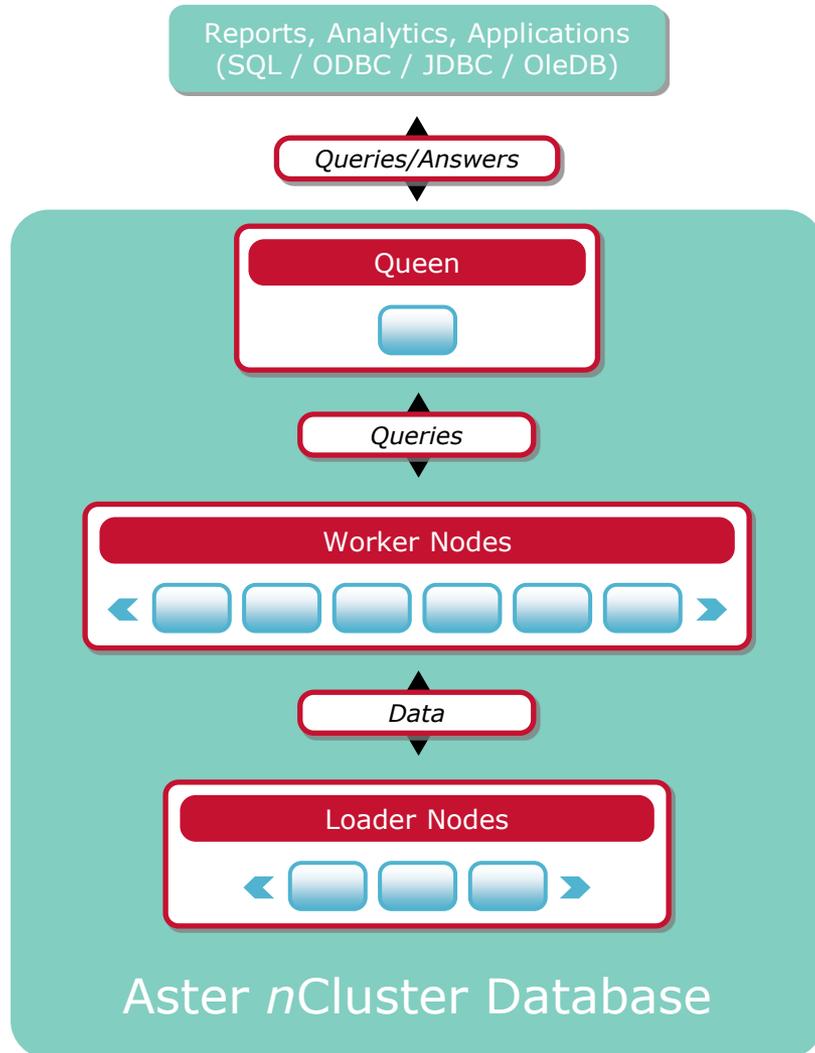
nCluster Architecture



Architectural Highlights:

- Cluster of tens to hundreds of commodity hardware boxes managed as a single database.
- 4 independent (share nothing) tiers for management, query processing, loading, & backup.
- In-database SQL-MapReduce enables parallelization of query and analytics processing.

The nCluster Approach: MPP Analytic Platform

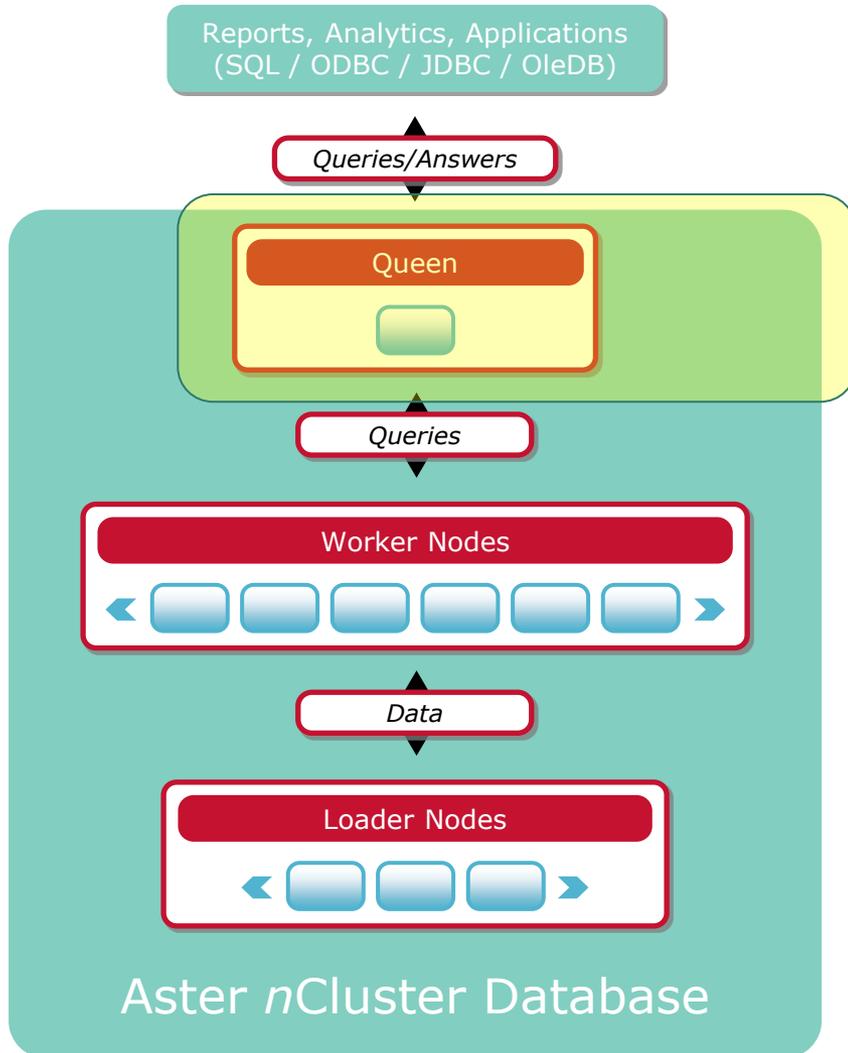


- Built for:
 - Data growth:
 - MPP architecture, simple scale-out
 - Faster loading:
 - Parallelized loader nodes
 - Agile analytics, in-database:
 - SQL/MR framework
 - SQL support:
 - SQL/JDBC/ODBC/OleDB interfaces
 - Availability:
 - Built-in data replication
 - Low cost:
 - Runs on a cluster of x86 servers & GigE/10GigE networks.

The nCluster Approach: Designed for Big Data

- *nCluster* is optimized for *dimensional models (star schemas)*.
 - Native support for distributed and replicated tables enables optimal performance for joins and complex workloads.
- Performance goal is to *maximize local computation*.
 - *nCluster's* push-down optimizations delegate to workers.
- Proper *star schema modeling* enables *nCluster* to optimize performance.
 - There's no need to model specifically for *nCluster*, but physical partitioning and logical partitioning settings enable you to improve performance.

nCluster: The Queen

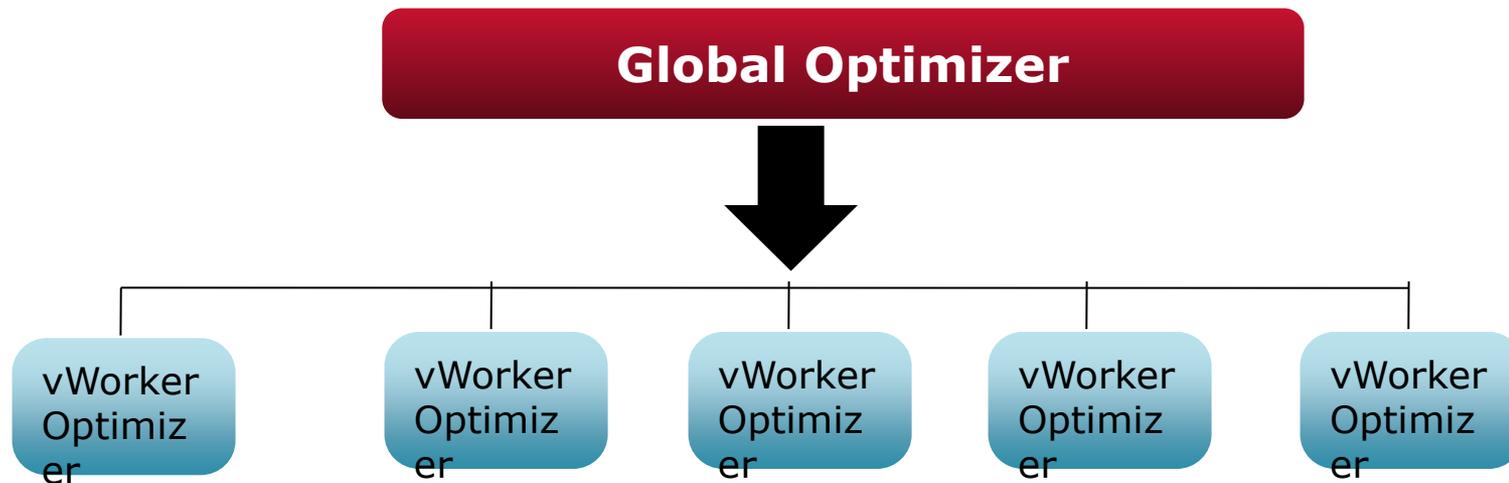


- **Queen**

- Manages system, configurations, schema, and error handling.
- Presents query interface, SQL/ODBC/JDBC/OleDb "The Face of nCluster"
- Global query optimizer coordinates queries in 3 steps:
 - Parses SQL statements and hands off to Planner
 - Planner develops a set of sub-queries to be executed
 - Executor will execute sub-queries and aggregate results

Network-Optimized Query Planning

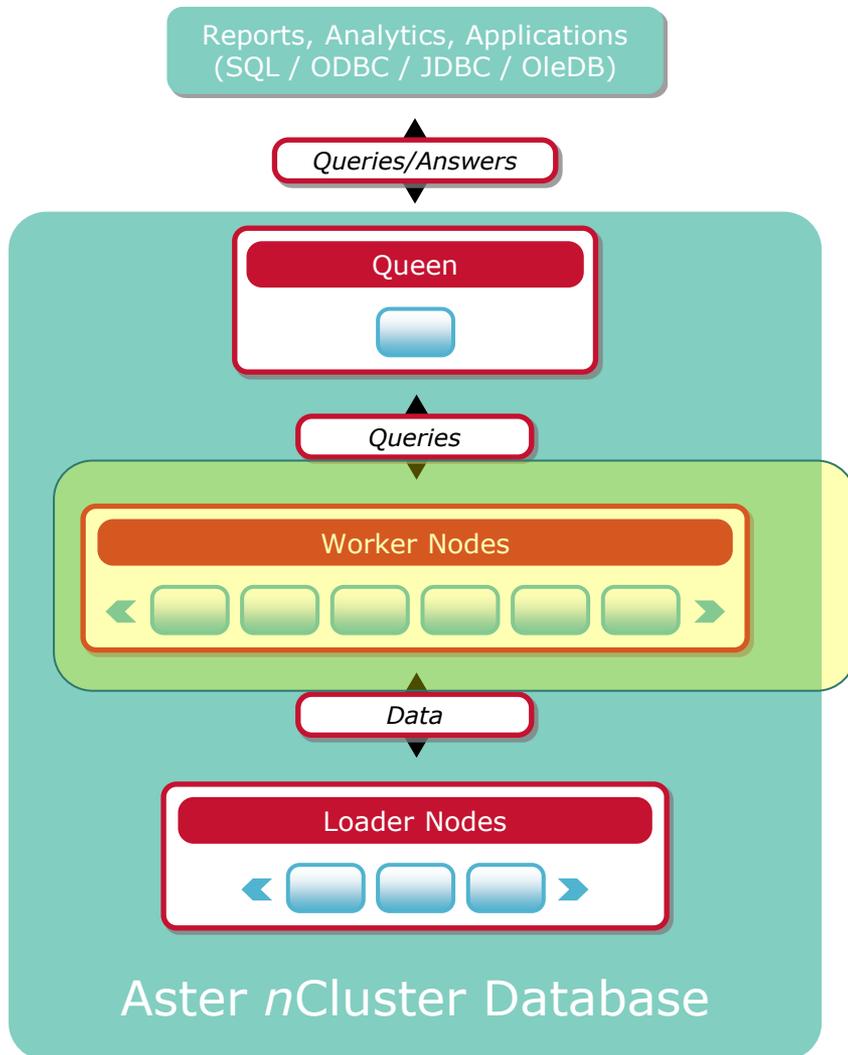
Maximize local computation to minimize network data flow



- Translate query into phases
- Send work to nodes that contain the data
- Reduce data fully before sending across the network

Independent optimization at global and local level maximizes performance

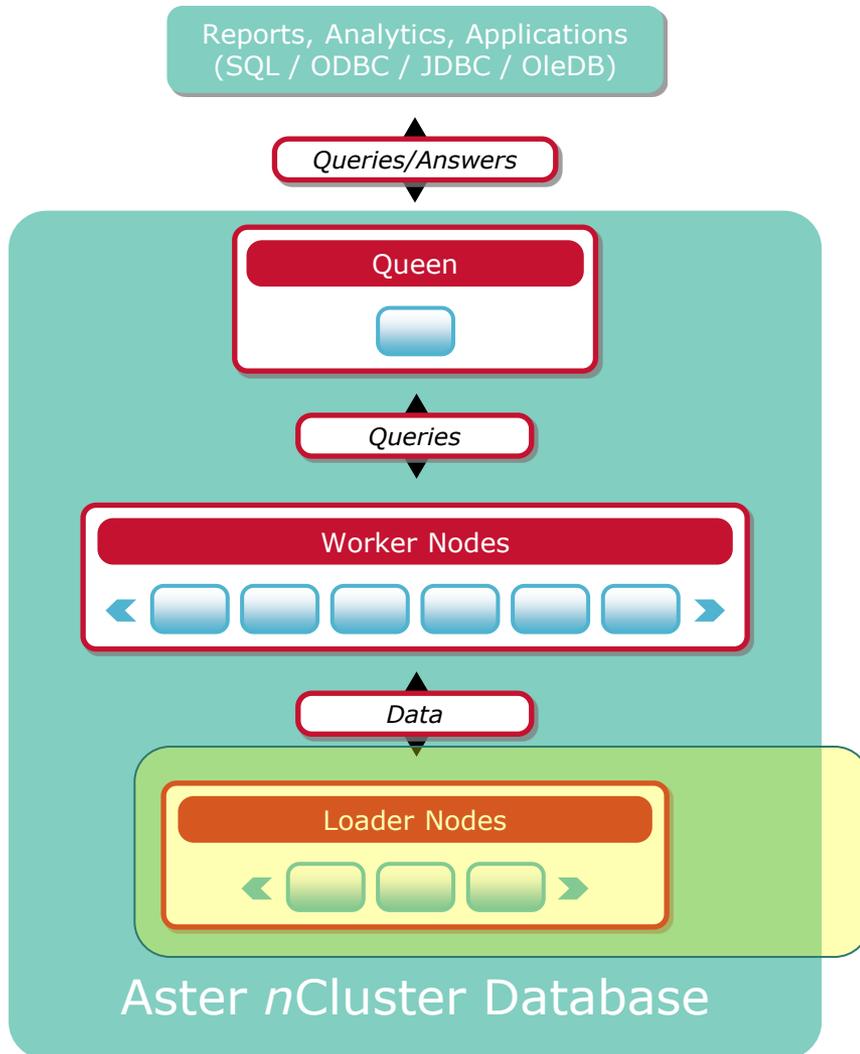
nCluster: Worker Nodes



• Worker Node

- Stores data and interacts with the queen and other workers.
- Executes queen's orders:
 - Run queries
 - Replicate
 - Balance storage
 - Balance processing
- Local query optimizer

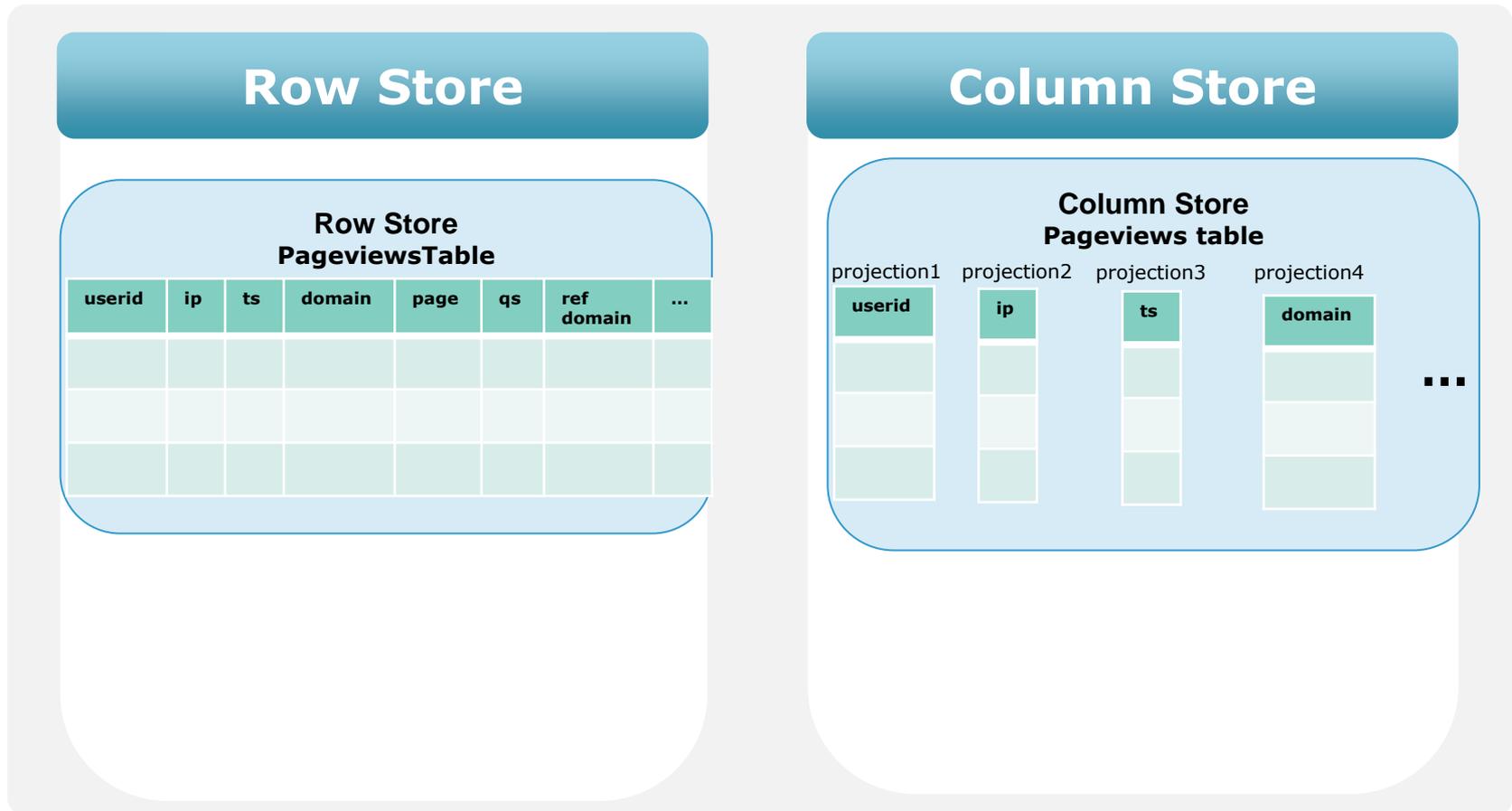
nCluster: Loader Nodes



- **Loader Node**

- Receives new data from bulk loading clients (for example, nCluster_loader, Informatica, similar tools)
- Partitions this data into appropriate segments
- Distributes the segmented data across vworkers

Understanding Row and Column Storage



Hybrid row and column ensures performance with analytic breadth

Where to get a VMWare version of Aster ?

<http://developer.teradata.com/database/articles/introducing-aster-express>

The screenshot shows the Teradata Developer Exchange website. The main navigation bar includes 'General', 'Database', 'Aster', 'Enterprise', 'Extensibility', 'Connectivity', 'Applications', 'Tools', and 'Viewpoint'. A search bar is located on the right. Below the navigation bar is a carousel of featured articles. The main content area displays an article titled 'Introducing Aster Express' by Teradata Mike, dated 22 Mar 2012. The article includes a social media share bar with 'Like', '+1', 'Share', and 'PRINT' buttons. The article text discusses the introduction of Aster Express for VMware Player images. A right-hand sidebar contains sections for 'RELATED TRAINING' and 'FROM THE FORUMS'.

TERADATA Developer Exchange BETA

Search [] Home Downloads Forums Blogs

General Database Aster Enterprise Extensibility Connectivity Applications Tools Viewpoint

Introducing Aster Express
Teradata Mike
ASTER

Teradata 13.10 Statistics Collection Recommendations
carrie
BLOGS

Teradata Alerts / CAM 14.0 released
gryback
VIEWPOINT

Where do the rows go?
PaulSinclair
BLOGS

What's new in Teradata Mapping Manager 2.0.2
ccurry
TOOLS

Stay in sync - Introducing Teradata Unity 13.10
schemanske
ENTERPRISE

TeradataR package for Red Hat Linux now available
Bask
BLOGS

Terat "Live" gryback
VIEWPOINT

Aster

Overview Articles Reference Training

ARTICLE INDEX Like +1 3 Share 4 PRINT

Introducing Aster Express

Article by Teradata Mike on 22 Mar 2012

5 comments

Tags: aster big data express virtual vmware

We are very excited to introduce the Aster Express for VMware Player images. Much like the very popular Teradata Express program, these downloadable Aster virtual images will provide customers with a free evaluation version of the Aster analytic platform that can be run on their PC. While this Express edition is not licensed for production usage, it is a fully functional Aster cluster that is a great tool for developers and testers or anyone else who wants a hands-on introduction to this Big Data analytics platform. Over the coming weeks and months here on Teradata's Developer Exchange, we will be publishing Aster tutorials, along with sample datasets, that will highlight the powers of this exciting platform.

Aster Overview

Aster is the premiere platform for Map Reduce analytics on Big Data. Based on a MPP database platform, Aster's patented SQL-MapReduce architecture combines the power and usability of SQL with Map Reduce components to enable all business users to innovate and discover hidden value in the vast mountains of semi-structured Big Data that is quickly engulfing most companies today.

At a high-level, an Aster cluster is composed of a Queen node and many Worker nodes. The Queen node provides the SQL query access to the Aster cluster, while the Worker nodes provide the distributed storage and computation. This combination of Map Reduce style analytics with traditional SQL allows analytics across all data, including queries

RELATED TRAINING

Introduction to the Aster Data Analytic Platform for Big Data Analytics
The growing importance of leveraging new sources of data and new types of analysis has led to the emergence of new technologies that can proce...

Aster Data Technology Overview: Addressing the Challenges of Big Data Analytics
This course provides an introduction to the technology behind the Aster Data Analytic Platform, a massively parallel software solution that co...

Do More with Your Data: Deep Analytics Using Big Data
Analytics are evolving to a new world from a focus on transactions to a focus on interactions. The analysis of detailed transactional data pro...

FROM THE FORUMS

Left outer join help
Thanks dnoeth. Its pretty much clear now :~)
5 hours ago by Mahs from Database forum

TERADATA[®] ASTER

Database Objects

Creating a Distributed table- Syntax

```
CREATE TABLE AdEventFact (  
  UserID          BIGINT          NOT NULL,  
  AdID            BIGINT          NOT NULL,  
  CampaignID      INTEGER        NOT NULL,  
  PageViewID     BIGINT          NOT NULL,  
  CookieID       VARCHAR(20),  
  IpAddr         CHAR(20)        NOT NULL,  
  EventTimeStamp TIMESTAMP      NOT NULL,  
  EventType      CHAR(10),  
  Page           VARCHAR(100),  
  DomainID      BIGINT          NOT NULL  
)  
DISTRIBUTE BY HASH (UserID)  
STORAGE ROW|COLUMN  
;
```

The column list begins and ends with parenthesis.

FACT tables must declare **DISTRIBUTE BY HASH (col_name)** clause.

Creating a Replicated Table - Syntax

```
CREATE TABLE DimAd(  
  AdID           BIGINT           NOT NULL,  
  AdName        VARCHAR(30)      NOT NULL,  
  AdDescription VARCHAR(100)     NOT NULL,  
  AdType        CHAR(10)         NOT NULL,  
  AdText        TEXT,  
  AdWidth       INTEGER,  
  AdHeight      INTEGER,  
  AdStatus      CHAR(10),  
  CONSTRAINT DimAdPK PRIMARY KEY (AdID)  
)  
DISTRIBUTE BY REPLICATION  
STORAGE ROW|COLUMN  
;
```

Dimension tables often implement PRIMARY KEYs for uniqueness.

Note, there is no Distribution KEY declaration for Replicated tables.

Multi-Level Partitioning Syntax Example

```
CREATE TABLE sales
( sales_order_id  int          NOT NULL
, customer_id    int          NOT NULL
, cust_state     char(2)
, region         char(1)
, amount         int
, sales_date     date)
DISTRIBUTE BY HASH (sales_order_id )
PARTITION BY RANGE (sales_date )
(partition sales_old      (END '2010-07-01'::date)
,partition sales_2010_07 (END '2010-08-01' ::date
    PARTITION BY LIST (region)
    (partition east (VALUES ('E')
    ,partition west (VALUES ('W'))))
,partition sales_2010_08 (END '2010-09-01' ::date
    PARTITION BY LIST (region)
    (partition east (VALUES ('E')
    ,partition west (VALUES ('W'))))
,partition sales_2010_09 (END '2010-10-01' ::date)
,partition sales_future  (END '3999-01-01' ::date)
);
```

```
-- Physical Partitioning
-- Logical Partitioning
-- old sales
-- January 2010 sales
-- Logical child partition
-- east division
-- west division
-- February 2010 sales
-- Logical child partition
-- east division
-- west division
-- March 2010 sales
-- future dated sales
```

TERADATA[®] ASTER

SQL/MR Overview

Databases for Big Data?

- Databases have long been data analysis engines
 - For reporting on structured data
- SQL is high-level, easy-to-iterate
 - Works on any schema
- SQL can be scaled to large amounts of data
 - Data warehouses with petabytes well-established
- Databases are well connected to IT infrastructure
 - Standardized Interfaces

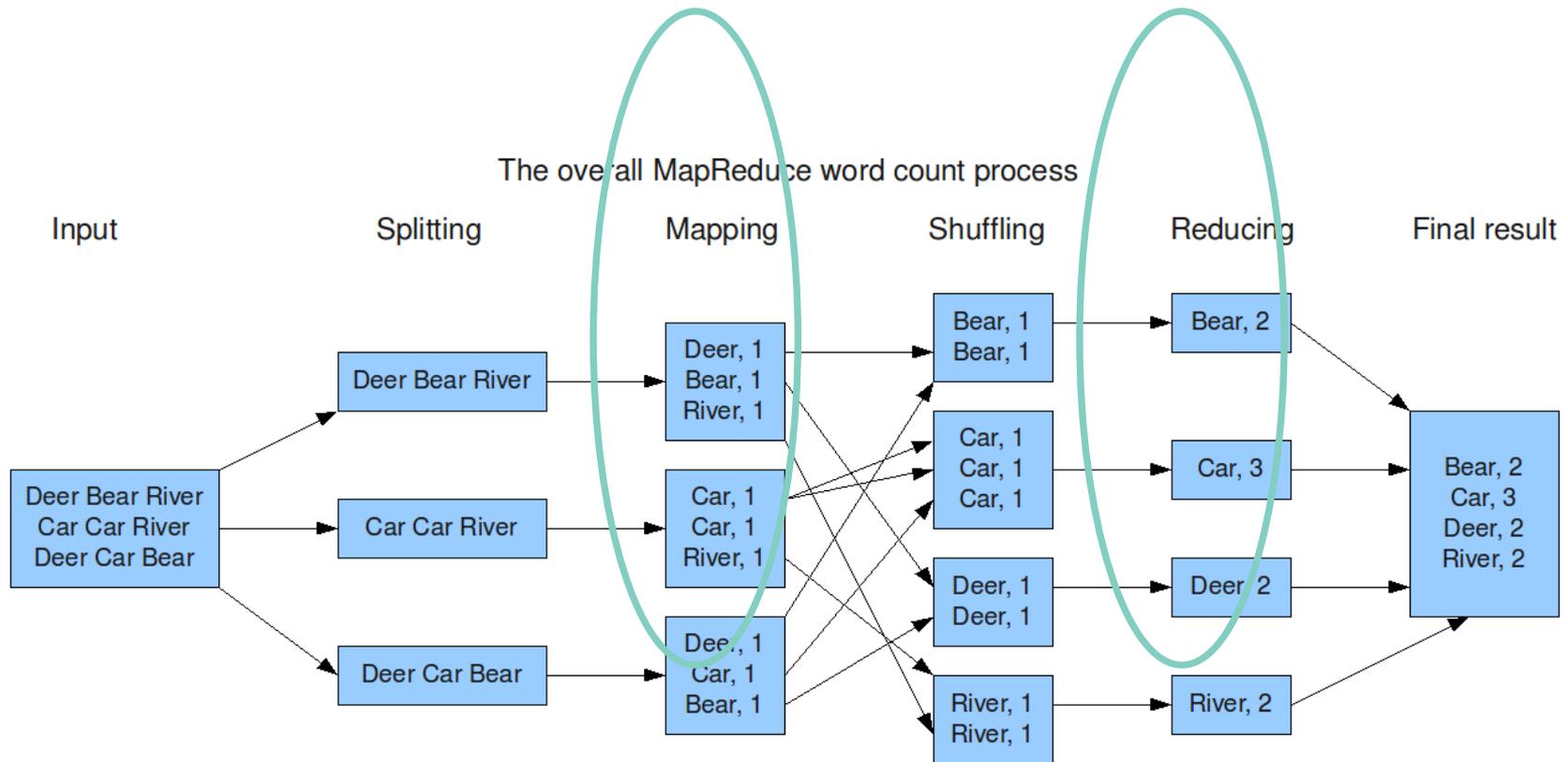
Shortcomings of Databases

- Relational Databases not suited to Multi-structured Data
 - Big Data structure often unknown prior to exploration
 - Don't want to hold up Big Data loading for parsing
- SQL is a poor match for some problems
 - Some queries cumbersome, non-intuitive, or impossible to express
 - Query optimizers can make poor choices leading to poor performance
- Traditional user-defined functions (UDFs) an incomplete fix
 - Scalar, aggregate functions of limited expressiveness
 - Table functions not designed for parallelism
 - Fixed schema limits reusability

The MapReduce Movement

- MapReduce scales processing to huge data volumes
 - Deployed on a large scale at several well-known Internet companies
 - Overcomes scale and expressiveness problems of traditional DBMSs
- Excellent programming model
 - Simple to understand
 - Structured to facilitate parallelization
 - Implemented in many languages

WordCount Application Design with MapReduce



Shortcomings of MapReduce

- But, MapReduce is heavily oriented to coding
 - New question frequently means new code
 - Harder to iterate
- Lost the declarative, reusable functionality of SQL
 - Data model: schema, statistics, locality optimization
 - General-purpose algorithms: joins, grouping, sorting
- Why can't we combine the best of Database and MapReduce to create a declarative, reusable, and scalable analysis tool?

SQL-MR: Design Goals

- **Scalable**
 - It should be easy to leverage hardware resources of 100's of servers
 - Fault-tolerance should be handled by the system
- **Analyst-friendly**
 - Want flexible, declarative language for analysts
 - Enable developers to create widely-reusable tools for analysts
 - Semantics of queries not mixed up with implementation details
- **Developer-friendly**
 - Want straightforward programming model
 - Provide useful platform services to developer to maximize freedom

SQL-MapReduce®: What Is It?

- **Aster Data's patented database implementation of the standard MapReduce framework for advanced analytics**
 - Examples: time-series, path and pattern analysis, graph analysis, market basket, decision trees, cluster analysis, data transformation
- **Architected for Optimal Big Data Analytics Execution**
 - Supports both batch **and** interactive queries executed in parallel
 - Maps sub-logic to multiple nodes and then combines/reduces results
 - Provides single process for executing both SQL and MapReduce logic
- **Easy to Use: Combines Standard SQL and MapReduce**
 - Delivers power of MapReduce through SQL and any SQL-based tool
 - Does not require a Hadoop implementation

SQL-MapReduce®: What Does It Do?

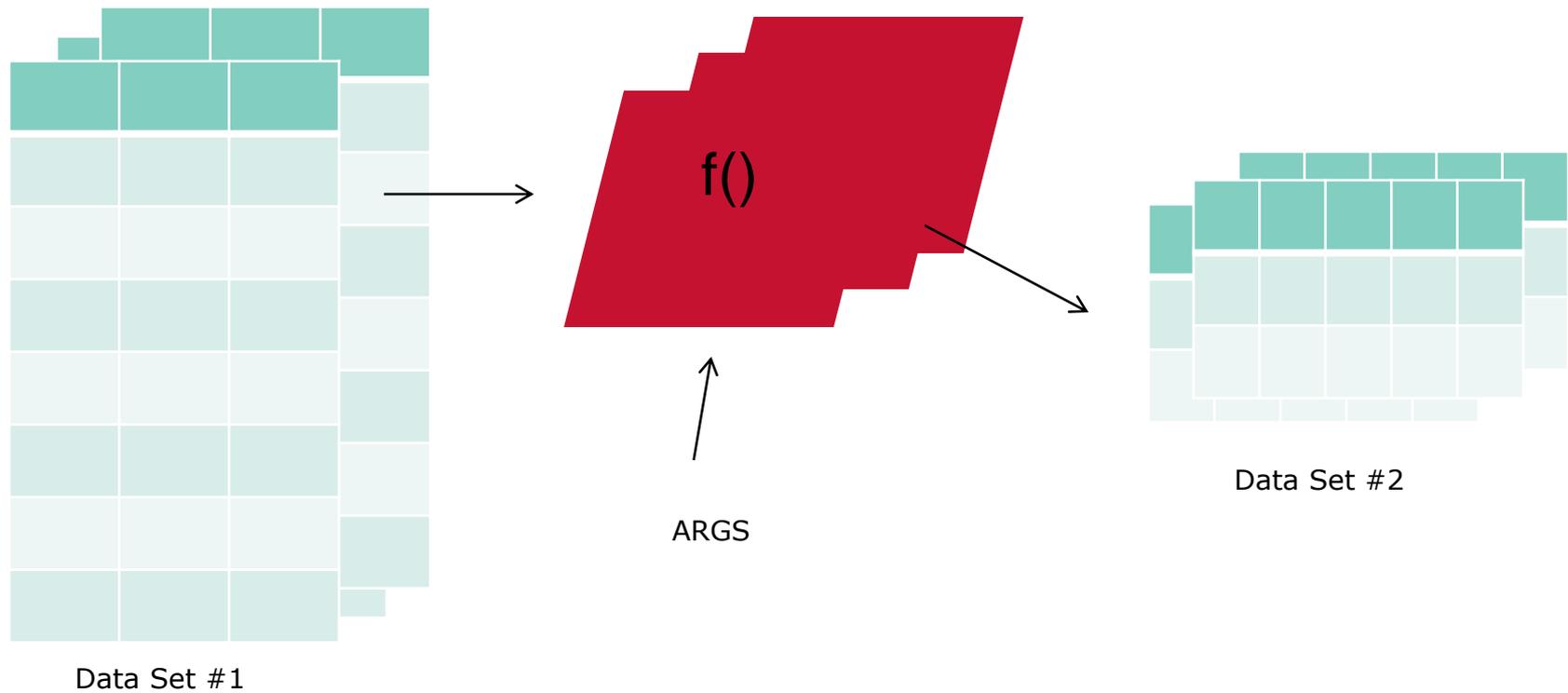
SQL-MR enables the execution of user code within *n*Cluster:

- User code is first installed throughout the cluster, then it's invoked on database data through SQL.
- Execution is automatically parallelized across cluster.
- Supports Java and C as primary languages, but other programming languages (C++, C#, Perl, Python, R) are supported through the Streaming interface.

TERADATA[®] ASTER

SQL-MR Invocation

Basic Primitive: SQL-MR Function



Basic Primitive: SQL-MR Function

- Self-describing, parallelized, relation-to-relation transform
- Operates on an input relation
 - Input relation (table, sub-query, etc.) specified by query
 - By default, allow any input schema; function can reject
 - Input rows can be accessed row-wise or partition-wise
- Emits an output relation
 - Function can be used in query like any relation; joined, aggregated, etc.
- Free-form transformation
 - Output schema chosen by function; based on input schema, argument clauses, external considerations, etc.
 - No enforced relationship between rows in input and output

Types of SQL-MR Functions

- *RowFunction*
 - Corresponds to a *map* function.
 - Must implement the *operateOnSomeRows* method.
 - Must be invoked without a PARTITION BY.
 - “Sees” all the appropriate rows on a particular worker.

- *PartitionFunction*
 - Corresponds to a *reduce* function.
 - Must implement the *operateOnPartition* method.
 - Must be invoked with a PARTITION BY, to specify how rows are reshuffled.
 - “Sees” all the appropriate rows in a partition.

Example RowFunction Invocation (Map)

```
SELECT * FROM
  IPGEO (
    on webclicks
    ip_address_col('ip_address')
  )
;
```

Input (webclicks):

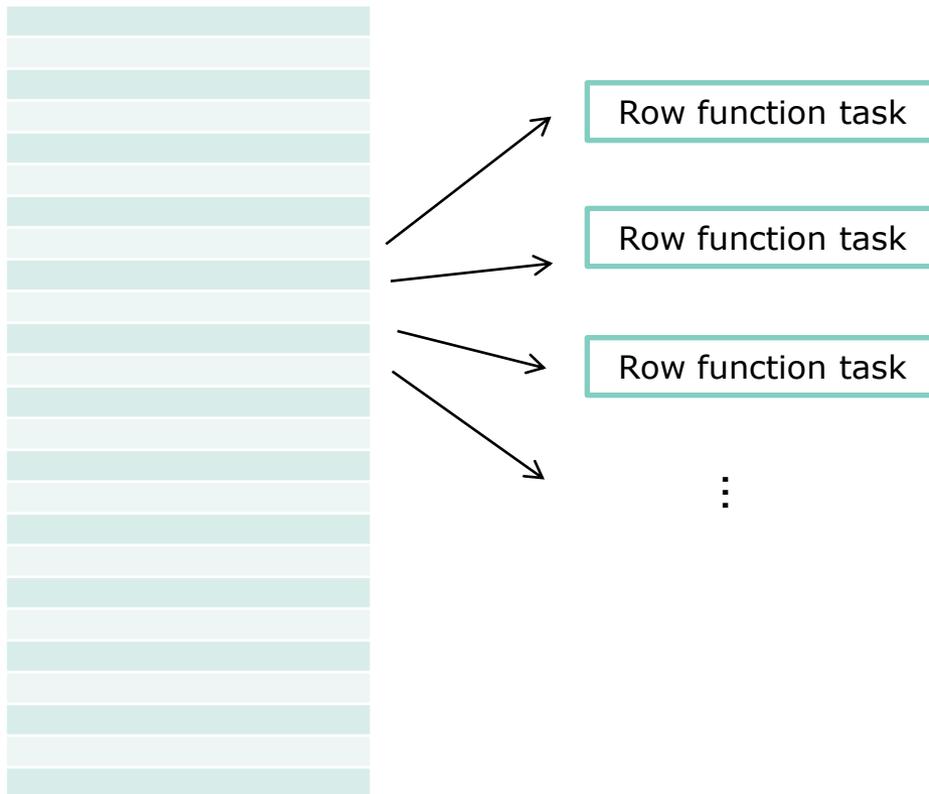
172.18.58.54 | user_id1 | 2012-01-01 23:33:30

Output:

172.18.58.54 | **USA** | **California** | **San Jose** | user_id1 | 2012-..

Parallelism – Row Functions

- Each row function task sees some subset of rows
 - The framework is free to schedule tasks to rows in any way
 - Data movement is not required. Think $1 \rightarrow N, N \geq 1$



Example PartitionFunction Invocation (Reduce)

```
SELECT * FROM
  sessionize(
    on webclicks
    partition by user_id
    order by timestamp
    timecolumn('timestamp') session_timeout(30)
  )
;
```

Input (webclicks):

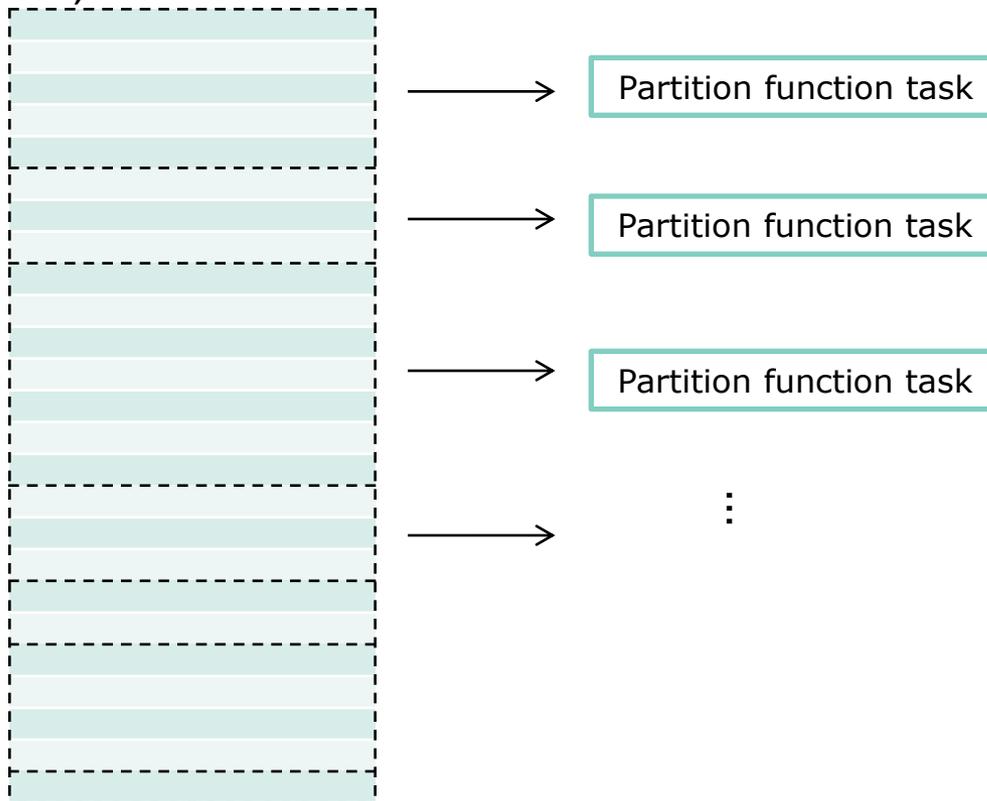
```
172.18.58.54 | user_id1 | 2012-01-01 23:33:30
172.18.58.54 | user_id1 | 2012-01-01 23:45:30
```

Output:

```
172.18.58.54 | user_id1 | 2012-01-01 23:33:30 | 1
172.18.58.54 | user_id1 | 2012-01-01 23:45:30 | 1
```

Parallelism – Partition Functions

- Each partition function task sees a group of rows
 - Rows in group share a common value of the PARTITION BY expression
 - The framework forms groups (shuffling across nodes, only if needed). Think $N \rightarrow M, M \geq 0$



SQL Integration

Functions integrated into SQL queries

```
SELECT ...  
FROM functionname(  
    ON table-or-query  
[PARTITION BY expr... ]  
  [ ORDER BY expr... ]  
  [ clausename ( arg... ) ]...  
)  
  
...
```

SQL-MR Platform Features

- SQL-MR also provides a variety of platform services
 - Services automatically managed by framework
 - Provides maximum freedom to developers
- Out-of-process execution
 - Function inputs and outputs are streamed, enabling high bandwidth
 - Bugs, crashes are isolated from database kernel processes
- Can use arbitrary external libraries, run arbitrary processes
 - Entire process group managed by framework
 - Security, scheduling, etc. managed at process level
 - Does not rely on support from any particular runtime
- Clean exception handling
 - If exception is ClientVisibleException then the user gets message
 - A full stack trace of the exception can be viewed through the AMC

TERADATA[®] ASTER

An Example SQL-MR Function

Word Count in Map/Reduce

- Goal: find frequencies of words in a set of documents
- Input data set:
 - Documents (docid int, body text)
- Output data set:
 - Count_tokens (word, count)

docid	body
1	'Jack and Jill went up the hill to fetch a pail of water. Jack fell down and broke his crown, and Jill came tumbling after.'
2	'Little Miss Muffet sat on a tuffet, eating her curds and whey. Along came a spider, who sat down beside her, and frightened Miss Muffet away!'
3	'The itsy bitsy spider went up the water spout. Down came the ran and washed the spider out. Out came the sun, and dried up all the rain. And the itsy bitsy spider went up the spout again.'



word	count
went	3
hill	1
up	4
down	3
spider	4
tuffet	1
...	

Input: The Documents Table

BEGIN;

```
CREATE FACT TABLE documents (  
    docid int,  
    body text,  
) DISTRIBUTE BY HASH(docid);
```

```
INSERT INTO documents VALUES (0, 'this is a single test  
document. it is simple to count the words in this single  
document by hand. do we need a cluster?');
```

END;

```
SELECT body FROM documents;
```

Invoking the Functions

```
BEGIN;
```

```
\install tokenize.jar
```

```
\install count_tokens.jar
```

```
SELECT word, count
```

```
FROM
```

```
    count_tokens (
```

```
        ON (
```

```
            SELECT word, count
```

```
            FROM tokenize(
```

```
                ON documents)
```

```
        )
```

```
        PARTITION BY word
```

```
    ) ORDER BY word DESC;
```

```
ABORT;
```

Even Better: Forget the Reduce

BEGIN;

\install tokenize.jar

```
SELECT word, sum(count)
  FROM tokenize(ON documents)
GROUP BY word
ORDER BY word;
```

ABORT;

Use SQL, Map and Reduce functions as tools that you can string together in any way to do the job.

TERADATA[®] ASTER

nPath and usecases

Analyzing a Clickstream Logs Table

- Business Question
 - How many distinct users start on a home page, click on an auction, view the seller's profile, and then bid on the item?
- Input Data
 - clicks table (userId, timeOfAction, actionType, ...)
 - actionType is an enum of {visitHome, viewAuction, placeBid, searchItem, viewProfile, ...}

The nPath Query for a Specified Path

```
SELECT count(distinct userId)
FROM nPath(
  ON clicks
  PARTITION BY userId
  ORDER BY timeOfAction
  MODE(NONOVERLAPPING)
  PATTERN('H.A.P.B')
  SYMBOLS(
    actionType = 'visitHome' AS H,
    actionType = 'viewAuction' AS A,
    actionType = 'viewProfile' AS P,
    actionType = 'placeBid' AS B
  )
  RESULT(first(userId of H) as userId)
);
```

The Four nPath Steps

- **Partition** the source data into groups
- **Order** each group to form a sequence
- **Match** subsequences of interest
 - Define a set of symbols via predicates
 - Define the subsequences of interest via a regular expression of symbols
- **Compute aggregates** over each matching subsequence

Path Aggregation Questions and Answers

- Knowing the most common paths that users take through a website helps answer these questions:
 - How are users traversing the site?
 - Where should we focus our site design efforts?
 - Are premature exits occurring? Where?
 - What's the 'Golden Path' (primary path to \$\$)?
 - How many users are following the golden path?
 - Are there equally profitable alternative paths to \$\$?

nPath Query to Find the Top 10 Common Paths

```
SELECT path, count(*) as freq
FROM NPATH (
  ON page_event_fact
  PARTITION BY session_key
  ORDER BY page_event_timestamp
  MODE (OVERLAPPING)
  PATTERN ('^A.B*$')
  SYMBOLS (
    page_key = 1 AS A, TRUE AS B
  )
  RESULT( ACCUMULATE(page_key OF B) AS path )
) T
GROUP BY path
ORDER BY freq DESC LIMIT 10;
```

The Accumulate function will returns a string, containing the sequence of page event keys, like this: "2:5:10:32..."

Page Dwell Analysis

- Page dwell time duration analysis
 - Dwell is the elapsed duration between when the user views a page and when the user moves on to view a subsequent page.
- In an opt-in scenario...
 - A low page dwell time may be desirable. The longer visitors are “distracted” from the event, the more likely they will abandon.
- For pages with educational information...
 - A longer page dwell time is desirable. The longer people view content, the more likely they thoroughly absorb the material.

nPath Query for Page Dwell Time Analysis

Question: Find the Average Page Dwell, in seconds, for all Pages

```
SELECT avg( b_time - a_time) as Avg_Page_Dwell
FROM npath (
  ON (select * from page_event_fact where user_id = 2123)
  PARTITION by session_key
  ORDER by page_event_timestamp
  MODE (OVERLAPPING)
  PATTERN ('A.B')
  SYMBOLS (true as A, true as B )
  RESULT ( LAST(page_event_timestamp of B) as b_time,
  FIRST(page_event_timestamp of A) as a_time)
) T
;
```

Usecase: Credit Line Increase to Default

- Loan transactions table:

customer	eventtimestamp	eventtype	amount
qi	2011-01-01	payment	100
qi	2011-02-10	latepayment	100
qi	2011-03-01	payment	200
qi	2011-03-15	CLI	
qi	2011-04-01	payment	95
qi	2011-05-01	payment	107
qi	2011-06-15	missedpayment	
qi	2011-07-15	missedpayment	

- Find sequence of credit line increase request (CLI) followed eventually by default (2 or more consecutive missedpayment). Find the customer and the latency from CLI to defaults.

Usecase: Credit Line Increase to Default

```
SELECT *, default_ts - cli_ts as cli_to_default_latency
FROM npath ( ON loan_transactions
  partition by customer
  order by eventtimestamp
  mode(nonoverlapping)
  pattern('CLI.X*.DEFT{2}')
  symbols(eventtype='CLI' as CLI, true as X,
    eventtype='missedpayment' as DEFT )
  result( first(customer of CLI) as customer,
    first(eventtimestamp of CLI) as CLI_ts,
    last(eventtimestamp of DEFT) as default_ts )
);
```

customer	cli_ts	default_ts	cli_to_default_latency
qi	2011-03-15	2011-07-15	122

Related Usecase: Last Payment Before Default

- In table `loan_transactions`, for each incidence of default (2 or more consecutive missedpayment), return the customer and the amount of the very last payment made.

```
SELECT *
FROM npath ( ON loan_transactions
partition by customer
order by eventtimestamp
mode(nonoverlapping)
pattern('PAYMENT.DEFT{2}'))
symbols(eventtype='payment' as PAYMENT,
eventtype='missedpayment' as DEFT )
result( first(customer of PAYMENT)
as customer, first(amount of PAYMENT)
as last_payment_amount )
);
```

customer	last_payment_amount
qi	107

Exercise 1: Last 3 Pageviews Before Mortgage App

From the pageviews table, for all incidents of a customer visiting the "mortgage apply form" page, return the customer and the sequence of the last 3 urls visited leading up to the mortgage apply form pageview.

pageviews

customer	eventtimestamp	url
antonio	2011-01-02 02:15:00	page1
antonio	2011-01-02 02:16:00	page1
antonio	2011-01-02 02:25:00	page9
antonio	2011-01-02 05:15:00	mortgage apply form
antonio	2011-01-02 05:16:00	page1
karthik	2011-01-01 12:00:00	seo landing1
karthik	2011-01-01 12:15:00	page1
karthik	2011-01-01 12:16:00	page23
karthik	2011-01-02 09:00:00	page5
karthik	2011-01-02 09:05:00	mortgage apply form
karthik	2011-01-02 10:00:00	page12

Hint: you will want to use *ilike* and *not ilike* in your pattern symbol definitions.

nPath Exercise 1 Answer:

```
SELECT *
FROM npath (
  ON pageviews
  PARTITION BY customer
  ORDER BY eventtimestamp
  MODE ( nonoverlapping )
  PATTERN ( 'PV.PV.PV.MAF' )
  SYMBOLS( url not ilike 'mortgage_apply_form' as PV,
           url ilike 'mortgage_apply_form' as MAF )
  RESULT ( first(customer of PV) as customer,
           ACCUMULATE(url of PV) as leadingPageSequence )
);
```

nPath Cross-Symbol Reference

- Example: find customers who view product P and buy it right away within 5 seconds
- Use nPath LAG function support in SYMBOLS clause
 - LAG must be on left side of comparison and all arithmetic on right side of comparison
 - Can use multiple LAG expressions AND'ed together

```
select * from npath (  
  on pageview  
  partition by userid  
  order by ts  
  mode (nonoverlapping)  
  pattern ('P.BUY')  
  symbols (url='p.html' as P, url='checkout.html' and  
lag(ts,1)>=ts-interval '5 seconds' as BUY)  
  result (first(userid of P) as userid, first(ts of P) as p_ts,  
  first(ts of BUY) as buy_ts)  
);
```

Monthly Balance Declines Followed by a Increase

monthly balances

customer	eventdate	balance
bob	2011-01-01	2500
bob	2011-02-01	2000
bob	2011-03-01	1000
bob	2011-04-01	500
bob	2011-05-01	600
bob	2011-06-01	1700
joe	2011-01-01	1500
joe	2011-02-01	2500
joe	2011-03-01	2000
joe	2011-04-01	1500
joe	2011-05-01	2500
joe	2011-06-01	3500

In table monthly_balances, find instances where customers show at least 2 consecutive month to (prior) month balance decline of at least 10%, immediately followed by a balance increase of at least 30%. Return customer name and sequence of balances covering the declines and increase months.

Monthly Balance Declines Followed by a Increase

```
select *
from npath ( on monthly_balances
  partition by customer
  order by eventdate
  mode(nonoverlapping)
  pattern('PREV.DOWN+.UP')
  symbols(true as PREV,
    lag(balance,1)>= (balance/0.9)::int as DOWN,
    lag(balance,1)<= (balance/1.3)::int as UP )
  result( first(customer of DOWN) as customer,
    accumulate(balance of ANY(PREV,DOWN,UP)) as balance_sequence )
);
```

customer	balance_sequence
joe	[2500, 2000, 1500, 2500]

Exercise 2: Detect Paycheck Deposits

- From the `savings_transactions` table, identify *any* pairs of adjacent deposits (ignore debits) whose amounts are over \$500 and within 2% of each other. Return the customer, the two deposit amounts, and the time lag between the first and the second of the deposits.

savings_transactions

customer	eventtimestamp	amount
bob	2011-01-01 00:00:00	1050
bob	2011-01-15 00:00:00	1040
bob	2011-01-17 00:00:00	-50
bob	2011-01-19 00:00:00	91
bob	2011-02-01 00:00:00	1051
bob	2011-02-14 00:00:00	1059

Hint: you will want to use *lag* in your symbol definitions.

nPath Exercise 2 Answer:

```
SELECT customer, firstDepositAmount, secondDepositAmount,  
       (secondDepositDate - firstDepositDate) as DepositLag  
FROM npath (  
    ON (select * from savings_transactions where amount > 0)  
    PARTITION BY customer  
    ORDER BY eventtimestamp  
    MODE ( overlapping )  
    PATTERN ( 'D1.D2' )  
    SYMBOLS (amount > 500 as D1,  
            lag(amount, 1) BETWEEN 0.98*amount AND 1.02*amount as D2)  
RESULT (first(customer of D1) as customer,  
       first(eventtimestamp of D1) as firstDepositDate,  
       last(eventTimestamp of D2) as secondDepositDate,  
       first(amount of D1) as firstDepositAmount,  
       last(amount of D2) as secondDepositAmount)  
);
```

Money Laundering: Small Credits then Big Debit

- In table `savings_transactions`, find incidence of at least three consecutive deposits of less than \$30 that are within an hour of the prior transaction, followed by a large debit of at least \$150 that is within 48hrs immediately following last of the string of deposits. Return the customer, the count and sum of the deposits, start/end time of the small deposits, amount and time of the large debit.

customer	eventtimestamp	amount
alice	2011-01-01 00:00:00	150
alice	2011-01-05 00:00:00	300
alice	2011-01-06 00:00:00	-10
alice	2011-01-06 02:00:00	15
alice	2011-01-06 02:03:00	20
alice	2011-01-06 02:04:00	2
alice	2011-01-06 02:50:00	8
alice	2011-01-06 03:30:00	28
alice	2011-01-06 03:31:00	20
alice	2011-01-06 03:35:00	19
alice	2011-01-07 09:00:00	-150
alice	2011-01-07 09:02:00	11

Money Laundering: Small Credits then Big Debit

```
select *
from npath ( on savings_transactions
partition by customer
order by eventtimestamp
mode(nonoverlapping)
pattern('INITIALSMALLDEP.SMALLDEP.SMALLDEP+.BIGDEBIT')
symbols(amount>=0 and amount<30 as INITIALSMALLDEP,
amount>=0 and amount<30 and lag(eventtimestamp,1)>=eventtimestamp-interval '1 hour' as SMALLDEP,
amount<=-150 and lag(eventtimestamp,1)>=eventtimestamp-interval '48 hours' as BIGDEBIT)
result(first(customer of SMALLDEP) as customer,
count(* of ANY(INITIALSMALLDEP,SMALLDEP)) as smalldeposit_count,
sum(amount of ANY(INITIALSMALLDEP,SMALLDEP)) as smalldeposit_total,
first(eventtimestamp of INITIALSMALLDEP) as smalldeposit_starttime,
last(eventtimestamp of SMALLDEP) as smalldeposit_endtime,
first(eventtimestamp of BIGDEBIT) as bigdebit_time,
first(amount of BIGDEBIT) as bigdebit_amount )
);
customer | smalldeposit_count | smalldeposit_total | smalldeposit_starttime | smalldeposit_endtime |
bigdebit_time      | bigdebit_amount
-----+-----+-----+-----+-----+-----
alice     |          7          |          112      | 2011-01-06 02:00:00    | 2011-01-06 03:35:00  |
2011-01-07 09:00:00 |          -150
```

Exercise 3: Auto Repairs Leading to Auto Purchase

From the `credit_transactions` table, identify the incidence of at least 2 autorepair category spend within 2 weeks of a prior autorepair spend, immediately followed by an autodealer spend of at least \$500 that is within 2 weeks of the prior autorepair spend. Return the customer, the number and total autorepair spend, the date of the first and last autorepair spend, and the date and amount of the autodealer spend.

customer	eventdate	merchant	merchantcategory	amount
alice	2011-01-02	m1	cat1	150
alice	2011-01-03	m2	cat2	20
alice	2011-01-10	m2	cat2	-10
alice	2011-01-10	m3	autorepair	150
alice	2011-01-13	m4	autorepair	45
alice	2011-01-14	m3	autorepair	109
alice	2011-01-20	m5	autorepair	230
alice	2011-02-01	m6	autodealer	500

nPath Prototype for Proxy Log Pattern Detection

```
CREATE TABLE npath_results DISTRIBUTE BY HASH(asset_tag) AS
SELECT *, last_ts-first_ts AS pattern_duration
FROM nPath(
  ON ( select p.*, d.mac_address, d.asset_tag
      FROM proxy P
      JOIN
      dhcp d
      ON p.ip = d.ip
      AND p.timestamp BETWEEN d.lease_start AND d.lease_end
      WHERE p.timestamp BETWEEN '2011-10-10' AND '2011-10-17')
PARTITION BY asset_tag
ORDER BY timestamp
MODE( nonoverlapping )
PATTERN( 'SN{1,3}.A*.BT{3,}.A*.BIGMAIL' )
SYMBOLS (
  ( url LIKE '%twitter.com%' OR url LIKE '%facebook.com%' ) AS SN, -- social network
  true AS A,
  (url LIKE '%piratebay%' OR url LIKE '%torrentz%' ) AS BT, -- bit torrent
  (url LIKE '%mail.yahoo.com%' AND bytes_sent > 50000 ) AS BIGMAIL )
RESULT( FIRST (asset_tag OF SN) AS asset_tag,
        ACCUMULATE (p.ip OF ANY(SN,BT,BIGMAIL)) AS ips,
        ACCUMULATE (url OF SN) AS sn_urls,
        ACCUMULATE (url OF BT) AS bt_urls,
        FIRST (timestamp OF SN) AS first_ts,
        LAST (timestamp OF BIGMAIL) AS last_ts )
) T
;
```



Aster Analytical Foundation Overview

Six Categories of Analytic Functions

Path Analysis

Discover patterns in rows of sequential data

Statistical Analysis

High-performance processing of common statistical calculations

Relational Analysis

Discover important relationships among data

Text Analysis

Derive patterns in textual data

Cluster Analysis

Discover natural groupings of data points

Data Transformation

Transform data for more advanced analysis

Sample SQL-MapReduce Packaged Functions

Modules	SQL-MapReduce Analytic Functions
<p>Path Analysis</p> <p>Discover patterns in rows of sequential data</p>	<ul style="list-style-type: none"> • nPath: complex sequential analysis for time series and behavioral patterns • Sessionization: identifies sessions from time series data in single pass • Attribution: operator to help ad networks and websites to distribute “credit”; options such as Uniform, Weighted and Exponential by occurrence or time.
<p>Graph and Relational Analysis</p> <p>Analyze patterns across rows of data</p>	<ul style="list-style-type: none"> • Graph analysis: finds shortest path from distinct node to all other nodes in graph • nTree: new function for performing operations on tree hierarchies. * • Other: triangle finding, square finding, clustering coefficient * 
<p>Text Analysis</p> <p>Derive patterns in textual data</p>	<ul style="list-style-type: none"> • Sentiment Analysis: classify content is positive or negative (for product review, customer feedback) * • Text Categorization: used to label content as spam/not spam * • Entity Extraction/Rules Engine: identify addresses, phone number, names from textual data * • Text Processing: counts occurrences of words, identifies roots, & tracks relative positions of words & multi-word phrases • nGram: split an input stream of text into individual words and phrases • Levenshtein Distance: computes the distance between two words 
<p>Data Transformation</p> <p>Transform data for more advanced analysis</p>	<ul style="list-style-type: none"> • Pivot: convert columns to rows or rows to columns * • Log parser: Generalized tool for parsing Apache logs * • Unpack: extracts nested data for further analysis • Pack: compress multi-column data into a single column • Antiselect: returns all columns except for specified column • Multicase: case statement that supports row match for multiple cases 

Sample SQL-MapReduce Packaged Functions

Modules	SQL-MapReduce Analytic Functions
<p>Statistical Analysis</p> <p>High-performance processing of common statistical calculations</p>	<ul style="list-style-type: none">• GLM: generalized linear model function that supports logistic, linear, log-linear regression models. Returns all parameters similar to R/SAS *• Naïve Bayes Classifier: simple probabilistic classifier; applies Bayes Theorem to data sets. *• Support Vector Machines: a supervised learning method for classification and regression analysis *• PCA: Principal Component Analysis -transforms a set of observations into a set of uncorrelated variables. *• Histogram: function to assign values to bins• Decision Trees: creates model of decisions and their possible implications• Approximate percentiles and distinct counts: calculates within specific variance• Correlation: characterizes the strength of the relation between different columns• Regression: linear/logistic regression btwn output variable & set of input variables• Averages: moving, weighted, exponential or volume-weighted averages
<p>Cluster Analysis</p> <p>Discover natural groupings of data points</p>	<ul style="list-style-type: none">• k-Means: clusters data into a specified number of groupings• Canopy: partitions data into overlapping subsets where k-means is performed• Minhash: buckets highly-dimensional items for cluster analysis• Basket analysis: creates configurable groupings of related items from transaction records in single pass• Collaborative Filter: predicts the interests of a user by collecting interest information from many users



TERADATA[®] ASTER

Path Analysis

Discover patterns in rows of sequential data

- **nPath:** complex sequential analysis for time series analysis and behavioral pattern analysis
- **nPath Extensions:** count entrants, track exit paths, count children, and generate subsequences
- **Sessionization:** identifies sessions from time series data in a single pass over the data
- **Attribution:** operator to help ad networks and websites to distribute "credit"

Path Analysis Functions:

- **nPath:** Complex sequence analysis for pattern matching on time series data.
- **Path Generator:** This function takes as input a set of paths where each path is a route (i.e. a series of pageviews) taken by a user from start to end. Then for each path, it generates the correctly formatted sequence and all possible sub-sequences for analysis by the Path Summarizer function. (See below...)
- **Path Starter:** Generates all the children for a particular parent and sums up their count. Note: the input data has to be partitioned by the parent column.
- **Path Summarizer:** This function takes as input the Path Generator function output and produces sum counts on all nodes. A "node" can be a plain sub-sequence (where the sequence and sub-sequence are different) or an exit sub-sequence (where both

Sessionization

- A session as a sequence of web site clicks by a user where no more than n seconds pass between successive clicks.
- Sessionization is the process of mapping each user click in a clickstream to a session identifier.
- If there is not a click from a user for n seconds then we start a new session.

timestamp	userid	...	timestamp	userid	...	sessionid
10:00:00	238909	...	10:00:00	238909	...	0
00:58:24	7656		10:00:24	238909		0
10:00:24	238909		10:01:23	238909		0
02:30:33	7656		10:02:40	238909		1
10:01:23	238909		00:59:24	7656		0
10:02:40	238909		02:30:33	7656		1

TERADATA[®] ASTER

Statistical Analysis

High-performance processing of common statistical calculations

- **Histogram:** function to provide capability of generating
- **Decision Trees:** Native implementation of parallel random forests.
- **Approximate percentiles and distinct counts:** calculate percentiles and counts within specific variance
- **Correlation:** calculation that characterizes the strength of the relation between different columns
- **Regression:** performs linear or logistic regression between an output variable and a set of input variables
- **Averages:** calculate moving, weighted, exponential or volume-weighted averages over a window of data

Histogram

- The Histogram SQL-MR function is used to understand the probability distribution of a continuous variable.
- It divides the entire data set into bins then identifies the number of points in each bin based on the value of a particular column.
- Bins can be:
 - continuous or discrete
 - equal or un-equal
 - overlapping or non-overlapping
- BI tools do this by issuing multiple queries on the same table.

Histogram: Continuous, Non-overlapping, Equal

id	name	age	graduate
100	Henry Cavendish	12	f
200	Sir William	15	f
300	Johann August	19	f
400	Martin Heinrich	20	f
500	Ralph Arthur	25	t
600	Marguerite Catherine	35	t
700	Philip Hauge	40	t
800	Joseph Louis	28	f
900	Marie Curie	12	t



```
SELECT * from histogram_reduce
( ON histogram_map
  ( ON customers
    BIN_SIZE( '10' )
    START_VALUE( '0' )
    VALUE_COLUMN( 'age' )
  )
  PARTITION BY ( bin )
  ACCUMULATE( 'bin', 'start_bin', 'end_bin' )
) ORDER BY bin;
```



bin	frequency	end_bin	start_bin
1	4	20	10
2	3	30	20
3	1	40	30
4	1	50	40

Histogram: Discrete, Overlapping, Un-Equal

id	name	age	graduate
100	Henry Cavendish	12	f
200	Sir William	15	f
300	Johann August	19	f
400	Martin Heinrich	20	f
500	Ralph Arthur	25	t
600	Marguerite Catherine	35	t
700	Philip Hauge	40	t
800	Joseph Louis	28	f
900	Marie Curie	12	t

```
SELECT * from histogram_reduce
( ON histogram_map
  ( ON customers
    INTERVAL('0:30','20:30','40:70','70:100000')
    VALUE_COLUMN('age')
  )
  PARTITION BY ( bin )
  ACCUMULATE ( 'bin', 'end_bin', 'start_bin')
) ORDER BY bin;
```

bin	frequency	end_bin	start_bin
0	7	30	0
1	3	30	20
2	1	70	40

Predictive Analytics

- Predictive Analytics is the process of assigning likelihoods to future actions based upon the occurrences of past actions.
- Pre-packaged analytic operators
 - Machine-learning Classification: Decision Tree; Naïve Bayes
 - Regression Analysis: Linear regression; Logistic regression
- Methodology
 - N-fold cross-validation
- Usecases
 - Ecommerce: propensity to buy big ticket product
 - Insurance: propensity to buy product; propensity to defect
 - Healthcare: propensity for hospital re-admission within N days

Predictive Analytics – Iterative Approach

1. Define prediction problem:

- Input: rows (e.g. set of users) and columns (e.g. # jobs) for past
- Output: boolean/categorical/numeric metric for future

2. Assembly:

- Algorithm
 - Classification: decision tree, Naïve Bayes, SVM, Regression: linear, logistic, ..
- Algorithm parameters (e.g. decision tree # levels)
- Dataset
 - Segment: e.g. US users; high tech users; male users
 - Time snapshot: as of April 1, 2011
- Feature set
- Output (e.g. propensity to buy LinkedIn Premium product)

Predictive Analytics – Iterative Approach (Cont.)

3. Validate:

- Divide labeled dataset into training and test set
- Train on training set, evaluate accuracy metrics on test set
- Repeat on different train/test sets for cross-validation

4. Iterative Refinement:

- Change step#2 specifications
- Perform step #3 validation
- Repeat over and over

Machine Learning Classification

- Data:
 - input features (categorical and numeric)
 - output prediction (categorical)
- Methodology
 - Training: Build model from labeled set of <input features, known output value>
 - Validation: Use model on holdout labeled set, read <input features> and predict output. Evaluate accuracy by comparing predicted and known output.
 - Classification: Use model on unlabeled dataset without known output
 - Cross-validation: repeat train+validate multiple times by dividing one labeled dataset into multiple folds
- Accuracy metrics
 - Accuracy: $\# \text{ predictions match known output} / \# \text{ predictions}$
 - Precision (wrt an output value): $\# \text{ true positives} / \# \text{ predictions of this value}$
 - Recall (wrt an output value): $\# \text{ true positives} / \# \text{ known outputs of this value}$

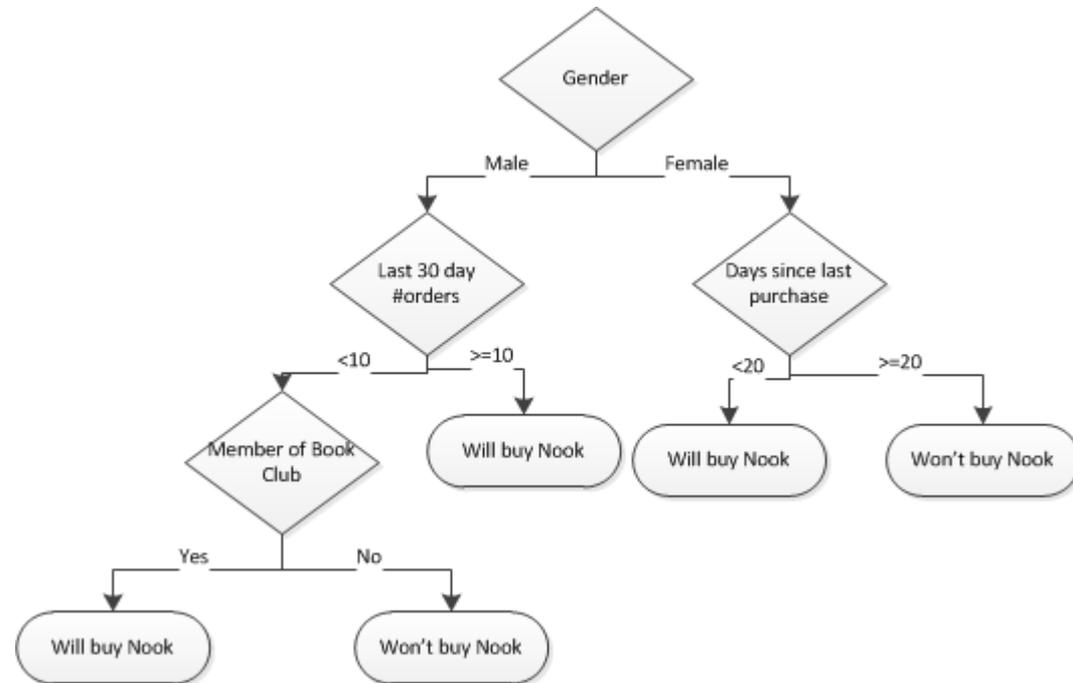
Precision and Recall (From Wikipedia)

- When a program for recognizing dogs in a scene correctly identifies 4 of the 9 dogs mistakes 3 cats for dogs:
 - its precision is $4/7$ (probability that the returned results are relevant)
 - and recall is $4/9$ (probability that relevant results are returned)
- When a search engine returns 30 pages, only 20 of which were relevant, while failing to return 40 relevant pages:
 - its precision is $20/30 = 2/3$
 - while its recall is $20/60 = 1/3$
- Good Prediction Models provide precision and recall close to 1.

Machine Learning Classification - Decision Tree

- Training: use labeled training set to create decision tree model; algo chooses splitting attribute that results in two subtrees each with dominating output class
- Parallelization
 - Training: ensemble method builds a decision tree on each vworker
 - Model: copy all trees to all vworkers
 - Classification: locally classify each row using forest of decision trees – aggregate one output prediction
- Our implementation requires numeric predictor variable
 - Training: categorical values must be converted to integral values
 - Classification: analyst must discretize continuous predicted value into categorical

Hypothetical Nook Purchase Propensity Decision Tree Model



Machine Learning - Decision Tree Function

- Training

```
SELECT * FROM forest_drive (  
  ON (select 1)  
  PARTITION BY 1  
  domain('[queen hostname/ip]')  
  database('[database]')  
  userid('[userid]')  
  password('[password]')  
  inputTable('[input table name]')  
  outputTable('[output model table name]')  
  response('[numeric response column name]')  
  numericInputs('col1','col2'..)  
  categoricalInputs('col3','col4'..)  
  numTrees('[number of trees to grow]')  
);
```

- Labeled dataset Response column – numeric type.
- numTrees: \geq # vworkers in cluster

- Classification

```
SELECT *FROM forest_predict (  
  ON [test_set_table]  
  domain('[domain]')  
  database('[database]')  
  password('[password]')  
  userid('[userid]')  
  forest('[model_table]')  
  numericInputs('[numeric feature column names]')  
  categoricalInputs('[categorical feature column names]')  
  idCol('[id_column to uniquely identify an input row]')  
);
```

- Output of classification: \langle id_column, predicted value(numeric) \rangle
- Join back to input test_set_table to compare input feature values and predicted output
- For validation, join back to input test_set_table to compare true output and predicted output to compute precision/recall

Machine Learning Classification - Naïve Bayes

$$p(C|F_1, \dots, F_n) = \frac{p(C) p(F_1, \dots, F_n|C)}{p(F_1, \dots, F_n)}$$

$$p(C|F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^n p(F_i|C)$$

P(buy nook|male, 20 orders last month, not book club member) =
1/Z * P(buy nook) * P(male|buy nook) * P(20 order last month|buy nook) * P(not book club member|buy nook)

P(not buy nook|male, 20 orders last month, not book club member)
= 1/Z * P(not buy nook) * P(male|not buy nook) * P(20 order last month|not buy nook) * P(not book club member|not buy nook)

- Training: Calculate prior probability distributions on each vWorker. Assume independence of input features. Combine probability distributions globally and copy to all vWorkers.
- Classification: Use prior probability distribution to compute the most likely output class for each local row.

Machine Learning - Naïve Bayes Functions

- Training

```
CREATE DIMENSION TABLE [model_table_name]
AS
SELECT * FROM naiveBayesReduce(
  ON( SELECT * FROM naiveBayesMap(
    ON [input_table]
    response('[response column name]')
    numericInputs('[numeric feature
column names]')
    categoricalInputs('[categorical
feature column names]')
  )
)
PARTITION BY class
);
```

- Response column can be integer or non-numeric categorical values

- Classification

```
SELECT * FROM naiveBayesPredict(
  ON [input_table]
  DOMAIN( [queen_ip:port] )
  DATABASE( [db_name] )
  USERID( [db_userid] )
  PASSWORD( [db_pwd] )
  MODEL( [model_table_name] )
  IDCOL( [id_column to uniquely identify an
input row] )
  numericInputs('[numeric feature column
names]')
  categoricalInputs('[categorical feature
column names]')
);
```

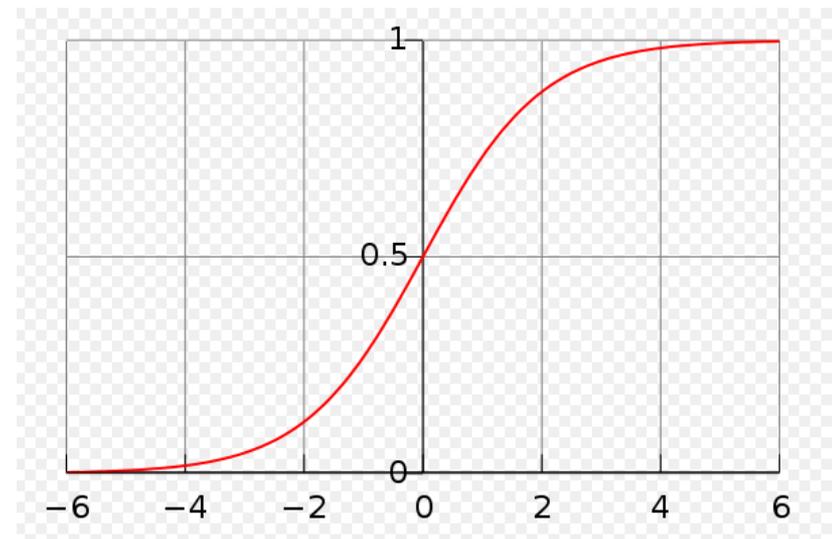
- Output: <id column, predicted categorical value, loglikelihood of each possible categorical value..>

Logistic Regression

$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots + \beta_k x_k,$$

- Logit function takes real input value (function z) and outputs $[0,1]$
- Training: Solve for parameters $B_0..B_k$ for best-fit $f(z(x_1..x_k))$ on training set
- Our implementation requires:
 - First column of input table must be the predictor feature. Must be boolean type.
 - Rest of the columns are input features. Must be real, integer, or boolean type.



Logistic Regression - Training

```
SELECT * FROM log_regression (  
    ON (select 1)  
    PARTITION BY 1  
    domain('[queen hostname/ip]')  
    database('[database]')  
    userid('[userid]')  
    password('[password]')  
    inputTable('[input table name]')  
    outputTable('[output model table name]')  
    weights('weight1','weight2'..)  
    columnnames('col1','col2'..)  
);
```

- Columnnames: optional. Must be a list of <predictor column, input col1, input col2..>
 - Predictor column must be boolean type
 - Input columns must be real/int/boolean type
- Weights: optional. Must have same # of values as Columnnames clause list. Specifies initial weights. Default 0.1 for all features.

Logistic Regression - Prediction

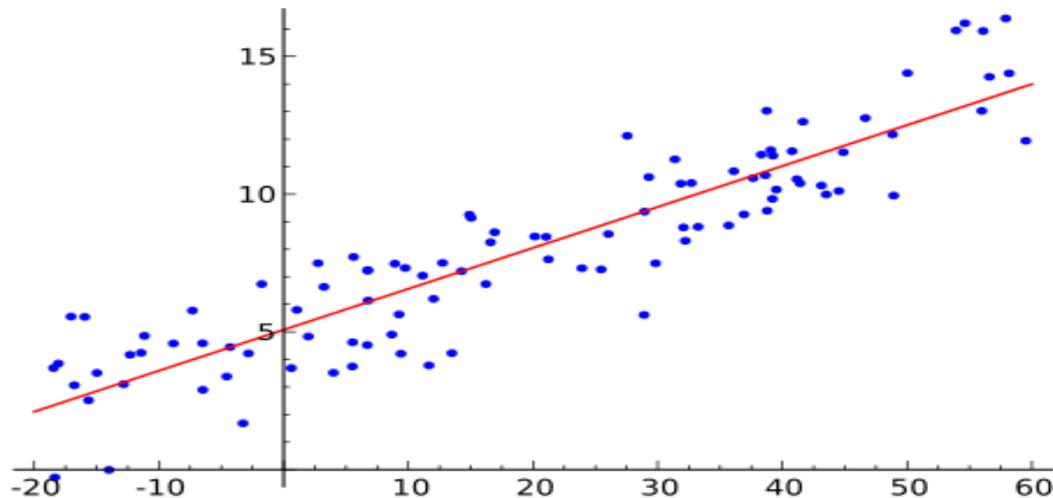
```
SELECT * FROM log_predict (  
    ON [test_set_table/(SELECT QUERY)]  
    domain('[domain]')  
    database('[database]')  
    password('[password]')  
    userid('[userid]')  
    weightstable('[model_table]')  
    thresholds('threshold1', 'threshold2', ..)  
);
```

- ON clause: input relation must be <id column, input col1, input col2...> where the input columns match the input columns fed to log_regression.
- Thresholds: optional. By default we discretize prediction as true/false at 0.5. User can specify multiple custom thresholds.
- Output of classification
 - Default: <id_column, probability between 0 and 1, predicted value true/false >
 - With thresholds clause: <id_column, probability between 0 and 1, predicted value true/false based on threshold1, threshold2, ..>
- User can join back to input test_set_table to compare input values with predicted output.
- For validation, join back to input test_set_table to compare true output and predicted output to compute precision/recall.

Linear Regression

$$Y = B_0 + B_1 x_1 + \dots + B_k x_k$$

- Linear regression function is a linear combination of the input variables.
- Training: Solve for parameters $B_0..B_k$ for a best-fit $Y=f(x_1..x_k)$ on the training set.



Linear Regression - Training

```
SELECT *  
FROM LINREG  
  ( ON LINREGMATRIX  
    ( ON tablename/(select query)  
      )  
    PARTITION BY 1  
  );
```

- LINREGMATRIX function executes on all vworkers in parallel, output small amount of summary data
 - Input schema: <input col1, input col2, ... input colK, output predictor column>
 - All input columns and output column must be numerical (integral/real) type
- LINREG function executes on only 1 vworker by combining output of LINREGMATRIX
- Output coefficients $B_0..B_k$.

Linear Regression - Prediction

- Unlike logistic regression, there is not a function for running prediction.
- User must write SQL to apply the coefficients to compute the prediction.

```
SELECT *
FROM LINREG
  ( ON LINREGMATRIX
    ( ON (select input1,input2,output from dataset)
      )
    PARTITION BY 1
  );
```

coefficient_index	value
0	2.85714285714288
1	-2.0571428571429
2	0.342857142857149

```
select input1,input2,output, 2.8571 - 2.0571*input1 +0.3429*input2 as output_predict
from dataset limit 5;
```

input1	input2	output	output_predict
7	37	1	1.1447
7	38	1	1.4876
7	39	2	1.8305
7	40	3	2.1734
7	41	3	2.5163

Simple Moving Average

```
SELECT *  
FROM SMAVG  
(  
  ON input_table  
  COLUMNS('column_names')  
  RETURN_ALL('true|false')  
  WINDOW_SIZE('window_size')  
)
```

- **COLUMNS:** Optional clause which specifies the column name for which simple moving average is required. If this clause is omitted, all the input rows are output as is.
- **RETURN_ALL:** Optional clause which specifies if the first WINDOW_SIZE rows should be output or not. Since simple moving average for the first WINDOW_SIZE is not defined, null's will be returned for those columns.
- **WINDOW_SIZE:** Optional clause which specifies the number of old values to be used for calculating the new weighted moving average. The default window size is 10.

Other “Average” SQL-MR Functions:

- **Exponential moving average:** The weighted moving average function computes the average over a number of points in a time series but applies a damping (weighting) factor to older values. The weighting for the older values decreases exponentially without discarding older values.
- **Weighted moving average:** The weighted moving average computes the average over a number of points in a time series but applies a weighting to older values. The weighting for the older values decreases arithmetically.
- **Volume weighted average price:** The volume weighted average price computes the average trade price of a stock over a specified time interval.

TERADATA[®] ASTER

Relational Analysis

Discover important relationships among data

- **Graph analysis:** finds shortest path from a distinct node to all other nodes in a graph
- **Tokenization:** splits strings into individual words to assist text processing

Graph Analysis - Single Source Shortest Path

- Input
 - Relational representation of edges in a graph
<source vertex id, destination vertex id>
 - Specify starting vertex id
- Output: list of all vertices
 - Vertex id
 - Whether this vertex is reachable from starting vertex
 - All vertices this vertex is connected to via an edge
 - Least # hops from starting vertex
 - Path (sequence of vertex ids) of shortest path from starting vertex

Graph Analysis – nTree

What is nTree?

- SQL-MapReduce function
- Build order tree hierarchies
- Link order to root
- Propagate/aggregate information across the tree

Benefits

- **Multi-level trees built with single pass** over the data
- **Variety of aggregates** built-in : propagate, sum, average, path.
- Detect and **break cycles** in bad data
- Variety of **tree-traversal** options : push down, push up.

SQL-based approaches would require significant SQL code changes to modify analysis (traversal pattern, aggregates)

CAVEAT: problem must be partition able

Hierarchy analysis

```
SELECT *
FROM nTree(
  ON (orders)
  PARTITION BY symbol
  KEY('ord_uuid')
  PARENT('parent_ord_uuid')
  ISROOT('is_root')
  NOCYCLE('1')
);
```

Ord_id	Parent Ord_id	Symbol	Root Order_id	Level
9Y091		AAPL	9Y091	1
5X452	9Y091	AAPL	9Y091	2
5Z347	5X452	AAPL	9Y091	3
7U198	5Z347	AAPL	9Y091	4

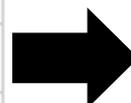
nTree On Order Data

```
SELECT *
FROM nTree(
  ON (orders) -----> Parallelize across workers
  PARTITION BY symbol ----->
  ROOT_NODE('parent_ord_id=NULL')
  PARENTS('ord_id')
  CHILDS ('parent_ord_id') } -----> Build along ord_uuid
  STARTS_WITH(ROOT) and parent_ord_uuid
  MODE (PUSH_DOWN) -----> Traversal options
RESULT(
  PROPOGATE(ord_id) as grand_parent_id,
  LEVEL() as level
)
OUTPUT('ALL')
ALLOW_CYCLES('true') -----> Detect cycles
ID('ord_id')
);
```

nGram

Parse text into groupings of words.

id	src	txt
1	wikipedia	the Quick brown fox jumps over the lazy dog
2	sampledoc	hello world. again, I say hello world



id	src	ngram
1	wikipedia	the quick
1	wikipedia	quick brown
1	wikipedia	brown fox
1	wikipedia	fox jumps
1	wikipedia	jumps over
1	wikipedia	over the
1	wikipedia	the lazy
1	wikipedia	lazy dog
2	sampledoc	hello world
2	sampledoc	I say
2	sampledoc	say hello
2	sampledoc	hello world

```
SELECT * FROM nGram
(
  ON my_docs
  TEXT_COLUMN('txt')
  DELIMITER(' ')
  GRAMS(2)
  OVERLAPPING('true')
  CASE_INSENSITIVE('true')
  PUNCTUATION('[.,?!]')
  RESET('[.,?!]')
  ACCUMULATE('id','src')
);
```

TERADATA[®] ASTER

Text Analysis

Derive patterns in textual data

- **Text Processing:** counts occurrences of words, identifies roots, & tracks relative positions of words & multi-word phrases
- **Text Partition:** analyzes text data over multiple rows
- **Levenshtein Distance:** computes the distance between two words

Text/Document Processing

- Extracting text from documents/files:
 - PDF
 - Office
 - HTML
 - HDFS
- Text analysis
 - Preprocessing
 - Keyword/phrase discovery
 - Text classification
 - Sentiment Analysis
- Extracting structure from unstructured content:
 - Apache Weblog Parser
 - JSON Parser
 - XML Parser

Sentiment Analysis

- Sentiment Analysis
 - Dictionary approach: Positive/negative word lists.
 - Scan a document and count # of +/- words
 - Score count normalized by document length
- Text classification
 - Naïve Bayes classification
 - Input documents processed as sparse occurrence vector of words.
 - Usecase
 - Spam detection
 - Sentiment analysis
 - Topic classification. E.g. is the customer service chat log trending about mortgage/savings/investing/insurance/loan products?

Text Parser

The `text_parser` function has the following specifications:

- Input: a column containing text data
- Output: one row for each unique token in each document
 - Token, count, position(s), other columns specified
- Variety of control parameters
 - Case sensitivity
 - Porter stemming (e.g., "containing" -> "contain")
 - Removal of stop words (e.g., "of", "the")
- Useful for processing large databases of text

Text Parser Sample Invocation - Input

- Input table: war_diary

reportkey	date	category	region	text
FE21A53...	2004-05-18	Non-Combat Event	RC EAST	At the Village of Dara there seems to be another...
FE21A53...	2004-05-18	Enemy Action	RC EAST	At 0200X, The WAZA KHW police...
...

Text Parser Sample Invocation - Process

```
SELECT * FROM text_parser
(
    ON war_diary
    TEXT_COLUMN ('summary')
    ACCUMULATE('reportkey')
    REMOVE_STOP_WORDS('true')
    LIST_POSITIONS('true')
);
```

Text Parser Sample Invocation - Output

reportkey	token	frequency	position
BFBE783...	delhi	1	21
080e000...	Picked	1	13
60C3b2...	locals	5	85,104,...
811A82...	madrassa	2	17,39
8CB6E0...	frontiers	1	50
199711...	afghani	3	2,29,31
...

TERADATA[®] ASTER

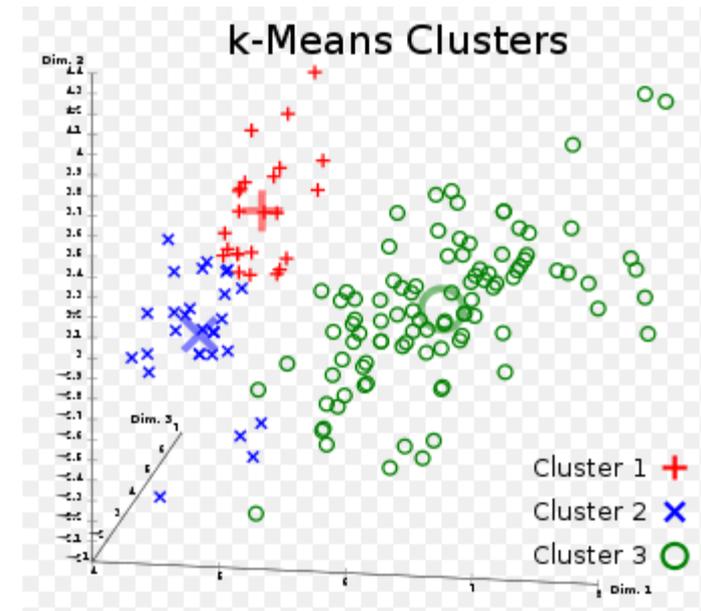
Cluster Analysis

Discover natural groupings of data points

- **k-Means:** clusters data into a specified number of groupings
- **Canopy:** partitions data into overlapping subsets within which k-means is performed
- **Minhash:** buckets highly-dimensional items for cluster analysis
- **Basket analysis:** creates configurable groupings of related items from transaction records in single pass
- **Collaborative Filter:** predicts the interests of a user by collecting interest information from many users

K-means Clustering

- Find hidden patterns/clusters in data by Dividing a set of data points into K disjoint subsets of similar/nearby points
- Algorithm:
 1. Starting with K clustroids - User specified/pick random points in space/pick random points
 2. Calculate cluster membership for each data points – find closest clustroid
 3. Recompute clustroid as avg of cluster member points
 4. Repeat steps 2 and 3 until converge or reach limit in # iterations
- Data points are set of numeric feature values
- Usecases: customer segmentation, anomaly detection, recommendation systems...



K-means Clustering Function

```
SELECT *
FROM kmeans
(
  ON (SELECT 1)
  PARTITION BY 1
  [ DOMAIN('<host_ip>') ]
  [ DATABASE('<database_name>') ]
  [ USERID('<db_user>') ]
  [ PASSWORD('<password>') ]
  INPUTTABLE('<input_table_name>')
  OUTPUTTABLE('<output_table_name>')
  NUMBERK(<number_of_means>)
  [ MEANS(<starting_clusters>) ]
  THRESHOLD(<threshold>)
  MAXITERNUM(<max_iterations>)
);
```

- Input table schema: <id column, numeric feature1, numeric feature2 ...>
- Output table schema: <id column, cluster id (0 to k-1), numerical feature1, numerical feature2 ...>
- Means clause: optional. User specified list of starting clustroids. E.g. for k=2, "means('15_70', '22_150')" maps to starting clustroid coordinates (15,70) and (22,150).
- Threshold: optional. Convergence criteria for distances in centroid between iterations.
- Maxiternum: optional. Convergence criteria in terms of # iterations.

Canopy Clustering

- Quickly find a set of clustroids subject to constraints:
 - Max distance from clustroid to cluster member point (t1)
 - Min distance between any two clustroids (t2)
- Can be used as preprocessing step to K-means
- Quickly partition data points into overlapping canopies (clusters), then apply more expensive clustering technique within each canopy.
- Input table schema: <id column, numerical feature1, numerical feature2 ...>
- Output table schema: <canopy id, numerical feature1, numerical feature2 ...>
- Input parameters: t1, t2 where t1>t2.

```
java -classpath canopyDriver.jar:<class path to file> -database=beehive -  
inputtable=canopyinput -outputtable=canopyoutput -t1=2 -t2=1 -userid=beehive -  
password=beehive -domain=192.168.75.100
```

Minhash/Locality Sensitive Hashing

- Generate probabilistic and overlapping clusters of users who are “similar” because they have bought similar items
 - Can generate similar items that have been bought by similar users
- Assign a pair of users to the same cluster with probability proportional to the overlap between the set of items that these users have bought
 - For a given user, calculate multiple cluster ids using several hash functions applied to a randomly chosen item this user has bought.

Market Basket Generator

- Generate all itemsets of size K
 - E.g. K=2: <milk, bread> are bought in the same transaction 1000 times; <milk, beer> are bought together 500 times.
- Usecases
 - Item to item affinity: What products are most commonly bought with milk?
 - Recommendation:
 - 1000 users watched both video1 and video2 (output of market basket generator). Video1 was watched by 1200 users, video2 was watched by 1100 users.
 - $\text{Probability}(\text{watch video2} \mid \text{watch video1}) = 1000/1200=0.83$. This probability is high, if a user watches video1 and has not yet seen video2, we should recommend video2.
 - $\text{Probability}(\text{watch video1} \mid \text{watch video2}) = 1000/1100=0.91$. If a user watches video2 and has not yet seen video1, we should recommend video1.

Market Basket Generator Function

```
SELECT *
FROM mbg
(
ON table_name|(query)
PARTITION BY <partition_column_1> [,
... ]
[BASKET_SIZE(<basket_size_value>)]
BASKET_ITEM('<basket_item_column>')
ACCUMULATE('column1 [, column2, ...]')
[ITEM_SET_MAX(<item_set_max_value>)]
[COMBINATIONS('true|false')]
);
```

- PARTITION BY: grain of baskets. E.g. Partition by customerid versus partition by customerid, transactionid
- BASKET_ITEM: single column that defines an item, e.g. productid
- BASKET_SIZE: default 2. Generate all itemsets of <basket_size> # of items
- ACCUMULATE: input columns to output with the itemset, e.g. customerid or customerid,transactionid
- COMBINATIONS: optional. false: return <item1,item2> as well as <item2,item1>. Default True: return only 1, the one in lexicographical order.
- ITEM_SET_MAX: optional. Default 100. If a partition has more than this # items, no output will be emitted.

Market Basket Generator Output

- Output: <accumulate columns, item1, item2, .., item_basketsize, count>
 - Count: # baskets that contain this item set
- From itemsets to recommendations using SQL
 - Generate basket count for each unique item
 - Join back to market basket output, e.g.
<item1, item2, copurchase_basketcount, item1_basketcount, item2_basketcount>
 - Compute recommendation metric
 - $\text{Copurchase_basketcount} / (\max(\text{item1_basketcount}, \text{item2_basketcount}))$
 - $\text{Copurchase_basketcount}^2 / (\text{item1_basketcount} * \text{item2_basketcount})$
 - Metric range [0-1]. 1 means 100% affinity, $P(\text{item1}|\text{item2})=P(\text{item2}|\text{item1})=1.0$.
- Forward in time basket generation
 - Milk and bread copurchase affinity there is no time dependency
 - Algebra book and calculus book copurchase has a strong time order; auto insurance and homeowners insurance probably has a fairly strong time order
 - Adding a flag to market basket generator SQLMR

Market Basket Analysis Syntax

```
SELECT *  
FROM BASKET_GENERATOR  
(  
  ON input_table  
  PARTITION BY (col, [, ...])  
  BASKET_ITEM('basket_item_column')  
  BASKET_SIZE('basket_size_column')  
  [ACCUMULATE('col' [, '...'])]  
  [COMBINATIONS('true|false')]  
);
```

Market Basket Function Parameters

- `input_table`: Table that contains the items to be collected into baskets.
- `BASKET_ITEM`: Required. Name of the column (in the input table) that contains the items to be collected into baskets. Each row in the `BASKET_ITEM` column is considered to be one item.
- `BASKET_SIZE`: Required. Number of items to be included in a basket.
- `ACCUMULATE`: Optional. Names of input columns to be returned as-is in the output. All input columns not named here are left out of output.
- `COMBINATIONS`: Optional. Specifies if the output should include all permutations of the items (each unique ordering of the items is considered a unique basket) or only all combinations of the items ("tomatoes and basil" is considered the same as "basil and tomatoes"). By default, the function returns only the unique combinations.

Market Basket Requirements

- User is expected to partition the input data in such a way that each partition represents a collection of items. All the columns specified in the ACCUMULATE clause will be emitted as is.
- It is assumed that each row of the BASKET_ITEM column specifies one single item.
- Columns specified in the ACCUMULATE clause should be a subset of columns specified in the PARTITION BY clause.

Market Basket Example

Input

Userid	Sku	Trans #
123	111	555
123	222	556
123	333	557
123	444	558

Output

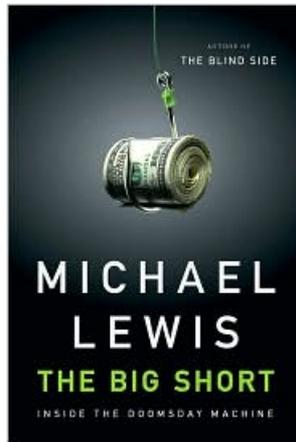
Userid	Sku1	Sku2	Sku3
123	111	222	333
123	111	222	444
123	222	333	444
123	111	333	444

```
SELECT *
FROM BASKET_GENERATOR
(
  ON transactions_table
  PARTITION BY (userid)
  BASKET_ITEM('sku')
  BASKET_SIZE('3')
  ACCUMULATE('userid')
  COMBINATIONS('true')
);
```

Collaborative Filtering

- Very common use case for retailers, on-line retailers, internet and consumer-focused financial institutions
- Source data could be:
 - retail purchase data
 - on-line purchase data
 - activity data
 - credit card purchase data
- Output could fuel “analytics products” like:
 - *“people you bought this also bought ...”*
 - *“people who viewed this profile also viewed...”*
 - *“people who liked this job also liked ...”*

B&N Recommendations with Collaborative Filtering



The Big Short: Inside the Doomsday Machine

by Michael Lewis

DIGITAL (eBook) > Learn more

Reader Rating: ★★★★★ (247 ratings)

> See All Reviews > Write a Review

- Pub. Date: May 2010
- Available for download via Wi-Fi and 3G
- 320pp
- Sales Rank: 203

OTHER FORMATS

- > Hardcover \$15.65
- > Paperback \$10.76
- > Compact Disc \$28.79
- > MP3 Book - Unabridged \$19.87

 **LendMe™ beta**
This eBook is Lendable!
> How it Works

BUY THIS EBOOK

\$27.95 List price

\$9.99 eBook Price
(You Save 64%)

BUY NOW
Read in Seconds

[About buying eBooks](#)

Available
These iter



GET FREE SAMPLE

Start reading a sample of this eBook for free! [Learn More](#)

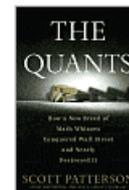
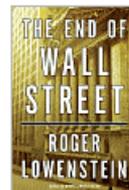
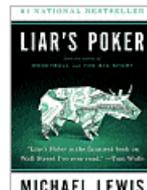
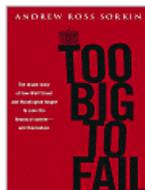
[Get Free Sample](#)

Also v

Welcome most a reader. books, magazi exclusi Barnes deliver free wi

[Discov](#)

Customers who bought this also bought



TERADATA[®] ASTER

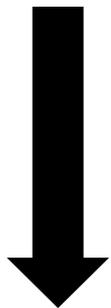
Data Transformation

Transform data for more advanced analysis

- **Unpack:** extracts nested data for further analysis
- **Pack:** compress multi-column data into a single column
- **Antiselect:** returns all columns except for specified column
- **Multicase:** case statement that supports row match for multiple cases
- **JSON Reader:** extracts elements from JSON data structure
- **Apache Reader:** extracts elements from Apache web log

Data Transformation - Pack

id	src	age	gender	race	numBuys	numSells
1	ebay	62	male	white	30	44
2	paypal	29	female	asian	33	23

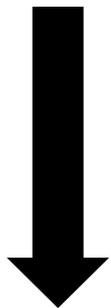


```
SELECT * FROM pack (  
  on sample_table  
  COLUMN_NAMES ('age', 'gender', 'race', 'numBuys', 'numSells')  
  COLUMN_DELIMITER (',')  
  INCLUDE_COLUMN_NAME ('false')  
  PACKED_COLUMN_NAME('packed_data')  
);
```

id	src	packed_data
1	ebay	62,male,white,30,44
2	paypal	29,female,asian,33,23
3	Bad_data	THISISINVALIDDATA

Data Transformation - Unpack

id	src	packed_data
1	ebay	62,male,white,30,44
2	paypal	29,female,asian,33,23
3	Bad_data	THISISINVALIDDATA



```
SELECT * FROM unpack
(
  on sample_table
  DATA_COLUMN('packed_data')
  COLUMN_NAMES('age', 'gender', 'race', 'numBuys', 'numSells')
  COLUMN_TYPES('integer', 'varchar', 'varchar', 'integer', 'integer')
  COLUMN_DELIMITER(',')
  IGNORE_BAD_ROWS('true')
);
```

id	src	age	gender	race	numBuys	numSells
1	ebay	62	male	white	30	44
2	paypal	29	female	asian	33	23

Data Transformation – Multicase (Input & Output)

<u>UserID</u>	Name	Age
100	Henry Cavendish	12
200	Sir William	15
300	Johann August	19
400	Martin Heinrich	20
500	Ralph Arthur	25
600	Marguerite Catherine	35
700	Philip Hauge	40
800	Joseph Louis	28
900	Marie Curie	12



<u>UserID</u>	Name	Age	Category
100	Henry Cavendish	12	kid
200	Sir William	15	teenager
300	Johann August	19	teenager
300	Johann August	19	young adult
400	Martin Heinrich	20	young adult
500	Ralph Arthur	25	young adult
500	Ralph Arthur	25	adult
600	Marguerite Catherine	35	adult
600	Marguerite Catherine	35	middle aged
700	Philip Hauge	40	adult
700	Philip Hauge	40	middle aged
800	Joseph Louis	28	Adult
900	Marie Curie	12	kid

Data Transformation – Multicase (Processing)

```
select * from multi_case
(ON
    (
        SELECT *,
        age < 1 as case1,
        (age >= 1 && age <=2 ) as case2,
        (age >= 2 && age <=12) as case3,
        (age >=13 && age <=19) as case4,
        (age >=16 && age <=25) as case5,
        (age >=21 && age <=40) as case6,
        (age >=35 && age <=60) as case7,
        (age >=60) as case8
        FROM mydata
    )
    LABELS(
        'case1 as "infant"',
        'case2 as "toddler"',
        'case3 as "kid"',
        'case4 as "teenager"',
        'case5 as "young adult"',
        'case6 as "adult"',
        'case7 as "middle aged person"',
        'case8 as "senior citizens"'
    )
);
```

JSON Parsing

- `SELECT * FROM logs`

```
row_id | json_string
```

```
-----+-----
```

```
1 | [1024,"Karthik",42]
```

```
2 | [923,"George",25]
```

```
3 | [48,"Frank",45]
```

```
4 | [148,"Joe",49]
```

SOURCE: logs Table

- `SELECT * FROM jsonparse(ON logs ...)`

- ```
row_id | id | name | age
```

- ```
-----+-----+-----+-----
```

- ```
1 | 1024 | Karthik | 42
```

- ```
2 | 923 | George | 25
```

- ```
3 | 48 | Frank | 45
```

- ```
4 | 148 | Joe | 49
```

Using the jsonparse function

SQL-MR Apache Log Format Parsing

```
96.255.99.50 - - [01/Jun/2010:05:28:07 +0000] "GET /origin-  
log.enquisite.com/d.js?id=a1a3af-  
ly6l645&referrer=http://www.google.com/search?hl=en&q=budget+planner&aq=5&aqi=g  
10&aql=&oq=budget+&gs_rfai=&location=https://money.strands.com/content/simple-  
and-free-monthly-budget-planner&ua=Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0;  
SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30618; .NET CLR 3.5.30729;  
InfoPath.2)&pc=pgys63w0xgn102in8ms37wka8quxe74e&sc=cr1kto0wmxqik1wlr9p9weh  
6yxy8q8sa&r=0.07550191624904945 HTTP/1.1" 200 380 "-" "Mozilla/4.0 (compatible;  
MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30618; .NET CLR  
3.5.30729; InfoPath.2)" "ac=bd76aad174480000679a044cfda00e005b130000"
```

Apache Log Parsing Example

Dynamically interpret web log data via MapReduce

Aster Data MPP Analytic Platform

SQL-MapReduce Staging for nPath

Timestamp	Referral	Customer	Session

SQL-MapReduce Staging for nPath

Timestamp	Referral	Customer	Session

Raw Log Input

Clickstream_Log

**SQL-MapReduce
program runs
in-platform**

Raw Log Input

Clickstream_Log

- High-speed, parallel loading

Click-stream Log Data

```
96.255.99.50 - - [01/Jun/2010:05:28:07 +0000] "GET /origin-  
log.enquisite.com/d.js?id=a1a3af-  
ly6l645&referrer=http://www.google.com/search?hl=en&q=budget+planner&aq=5&aqi=g  
10&aql=&oq=budget+&gs_rfai=&location=https://money.strands.com/content/simple-  
and-free-monthly-budget-planner&ua=Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0;  
SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30618; .NET CLR 3.5.30729;  
InfoPath.2)&pc=pgys63w0xgn102in8ms37wka8quxe74e&sc=cr1kto0wmxqik1wlr9p9weh  
6yxy8q8sa&r=0.07550191624904945 HTTP/1.1" 200 380 "-" "Mozilla/4.0 (compatible;  
MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30618; .NET CLR  
3.5.30729; InfoPath.2)" "ac=bd76aad174480000679a044cfda00e005b130000"
```

- Raw Apache weblogs

Encrypted Data Queries

```
beehive=> select * from myspacedecrypt_keyfile( on testdecrypt inputcolumn('data'));  
a | data | decrypted_output | decryption_error
```

```
-----+-----  
-----+-----  
-----+-----  
-----
```

```
3 |  
n%2b0prKAebczfG13CmB%2fL170xxzjSK8k%2f19igSi%2f1c3uQLTXYANJMky0qJd%2b9I2k8XWd1RMDad  
U%2bPXFFnu2dIGF%2bP5In0m0qtpKOiBgJNRjwzbOwD5WLjzRf3Zx8VEZYqclvNCxxMKgNzI5Oed0U9n6h%2  
bWWvO4BDPQ2TCD7dH6i6msdHAKAqgoqMfpUcNYZUwyM4J0gp%2bBliskuemgFoqCOD%2fL5ovOI6kGCFmj  
4NZLqWgfoZAqHjG5LsAW9ouWp |  
track_type=ActivityClick&display_context=UserHome2&rcpt_uid=350157709&parent_oid=750580&init_uid  
=528039026&raised_activity=PhotoAdd&oid=6978324&raise_ts=2010-07-01 19:10:39.000 |
```

```
4 |  
n%2b0prKAebczfG13CmB%2fL170xxzjSK8k%2f19igSi%2f1c3uQLTXYANJMky0qJd%2b9I2k0CR4XbEEO8H  
Km91kINM7PE3niq88y1dnxFEQhXZ7MzGRP27THsk7jlQLncOv7ZLHIK%2bhoRwFs3eeBoKpTk5VzrBKuu4p3I7  
9X8R0QRb4g8yQ8vvnvHCdXNT2M3SzLlcco40EG5uZ5AdKCCDW7nE5pXpSgspPBVj37LCMY6lZOFhl%2bNIUUa  
QsFvt4EhstbNOoI |  
track_type=ActivityClick&display_context=UserHome2&rcpt_uid=356999567&parent_oid=-  
1916564383&init_uid=32050898&raised_activity=BulletinAdd&oid=1&raise_ts=2010-06-27 22:23:00.000  
|
```

Interpretation of Multi-Structured Data

- JSON (JavaScript Object Notation)
- Apache Logs
- Encrypted Data
- Word doc
- PDF
- HTML
- Excel
- PPT
- XML
- Sets of elements (Pack/Unpack)
- Lucene indexes