VERTICA

Vertica Now

Quarterly Customer Webinar

April 26, 2017

Hewlett Packard
Enterprise

VERTICA

Vertica Now is our customer-only quarterly webinar designed to keep you informed of new releases and updates, helpful technical resources, future roadmap capabilities, and more!

Hewlett Packard Enterprise

Today's Agenda

- **Vertica Updates:** New Websites and Resources **(Joy King)**
  - Vertica.com and my.Vertica.com
  - Blog and Knowledge Base
- **What's New:** Vertica 8.1 Release **(Nga Tran and Waqas Dhillon)**
  - Flattened Tables
  - Kafka Integration
  - Spark Integration
- **Did You Know:** Backup and Restore **(Eugenia Moreno)**
  - How it Works
  - Best Practices
- **Q&A**

VERTICA

Vertica Updates

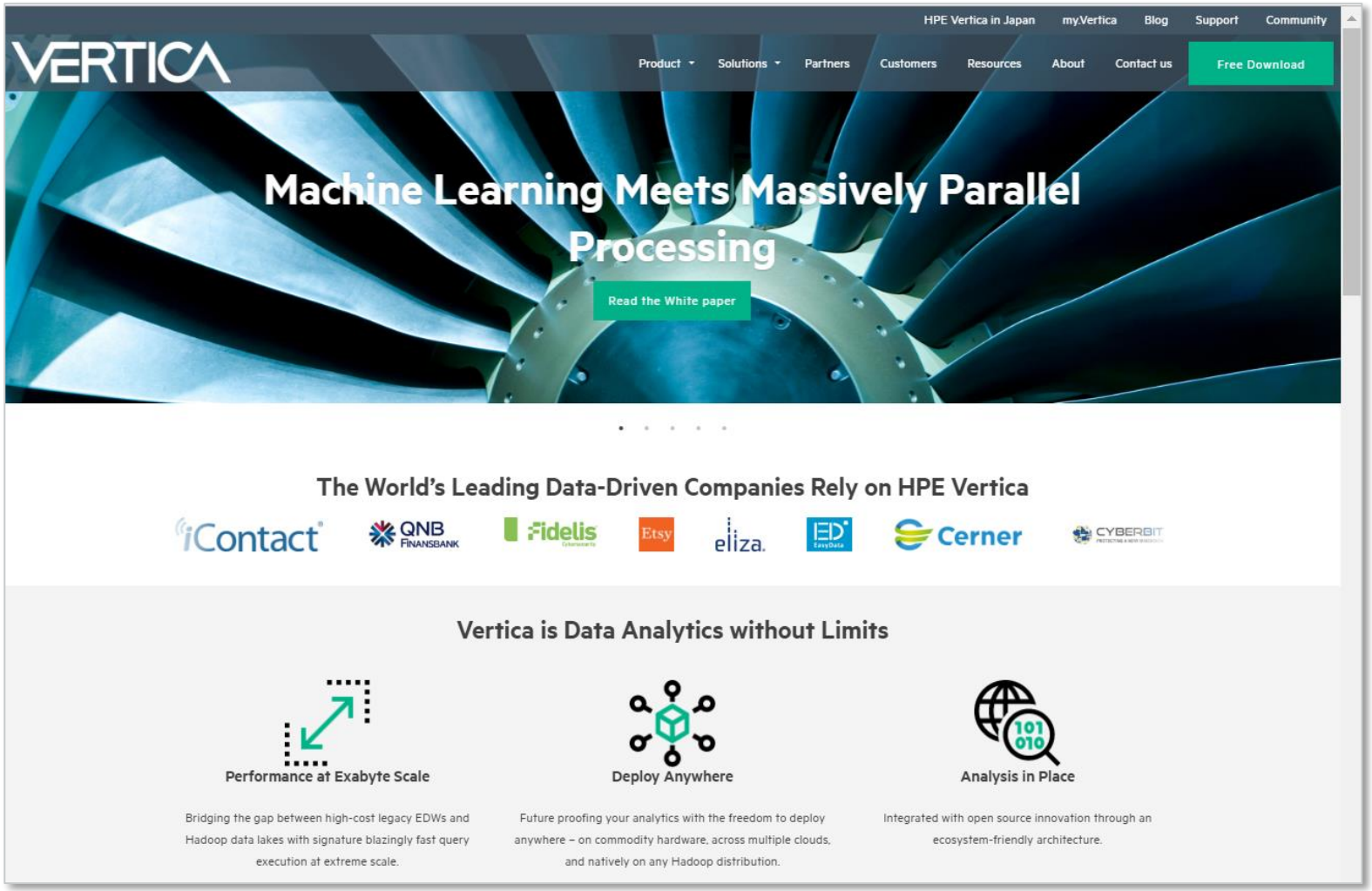
Joy King

Hewlett Packard Enterprise

Launching New Vertica Website

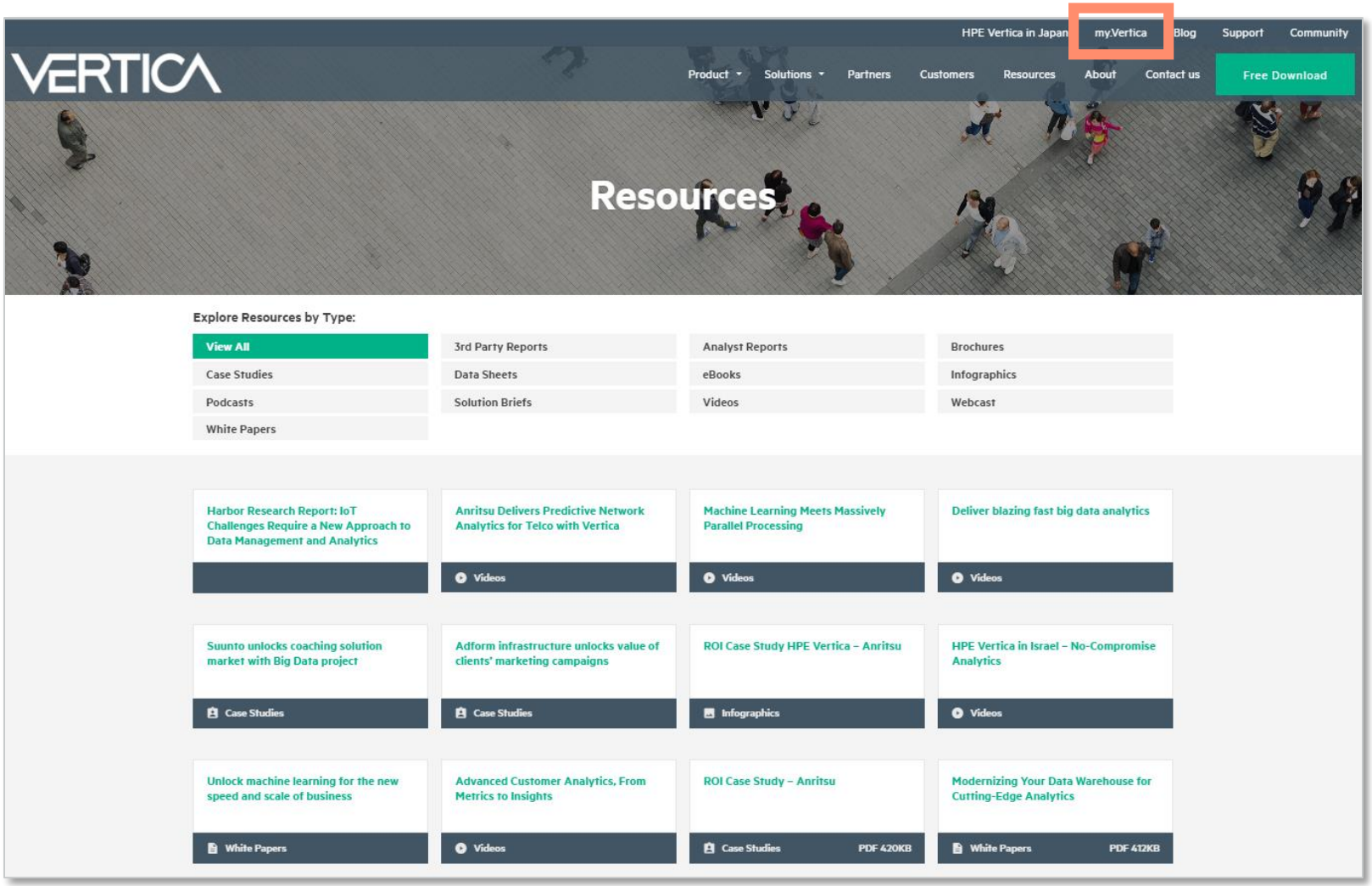
All Vertica content and information now available on one site

- Product
- Solutions
- Partners
- Blog
- my.Vertica
- Community

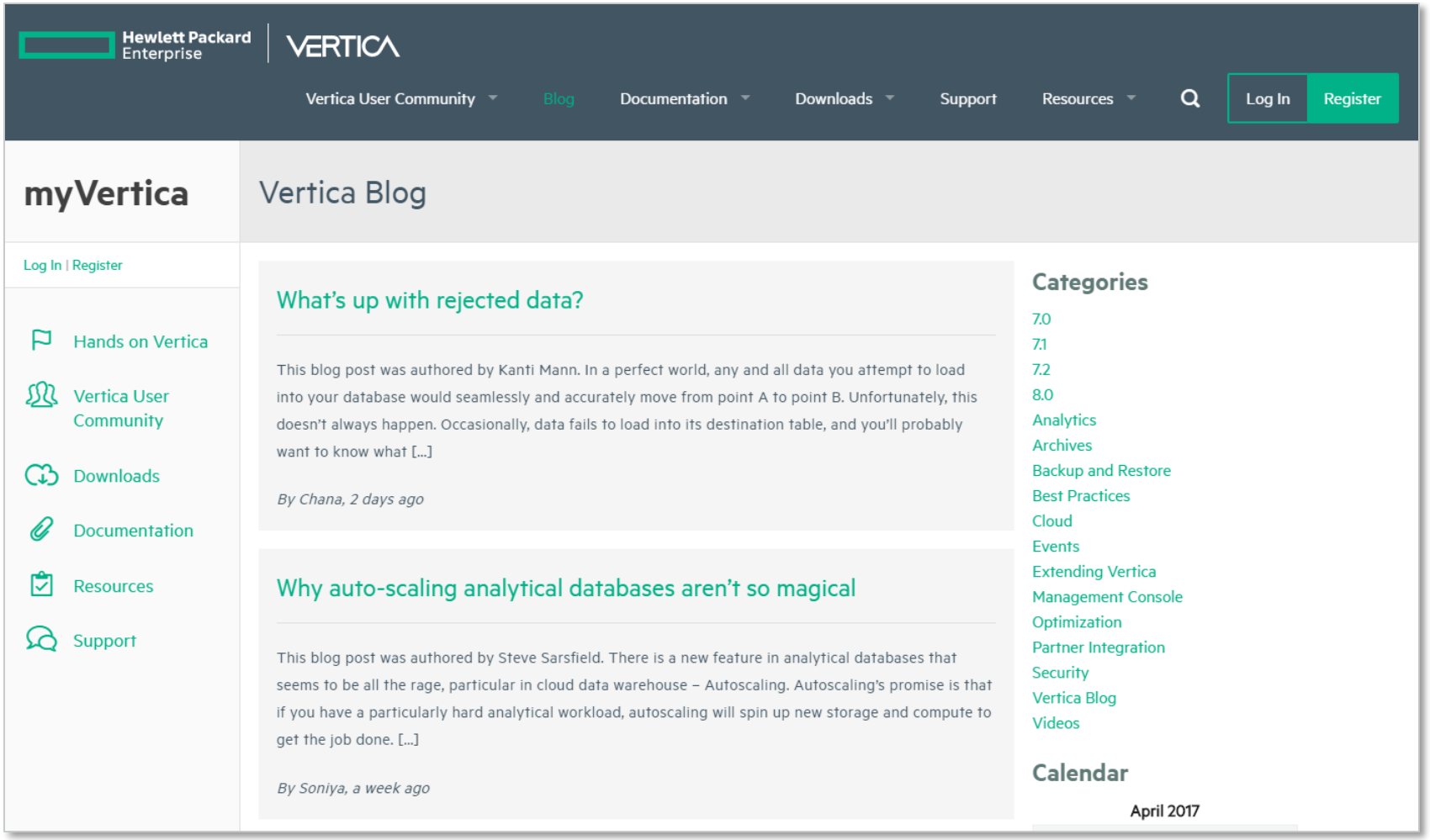


Everything You Need and More

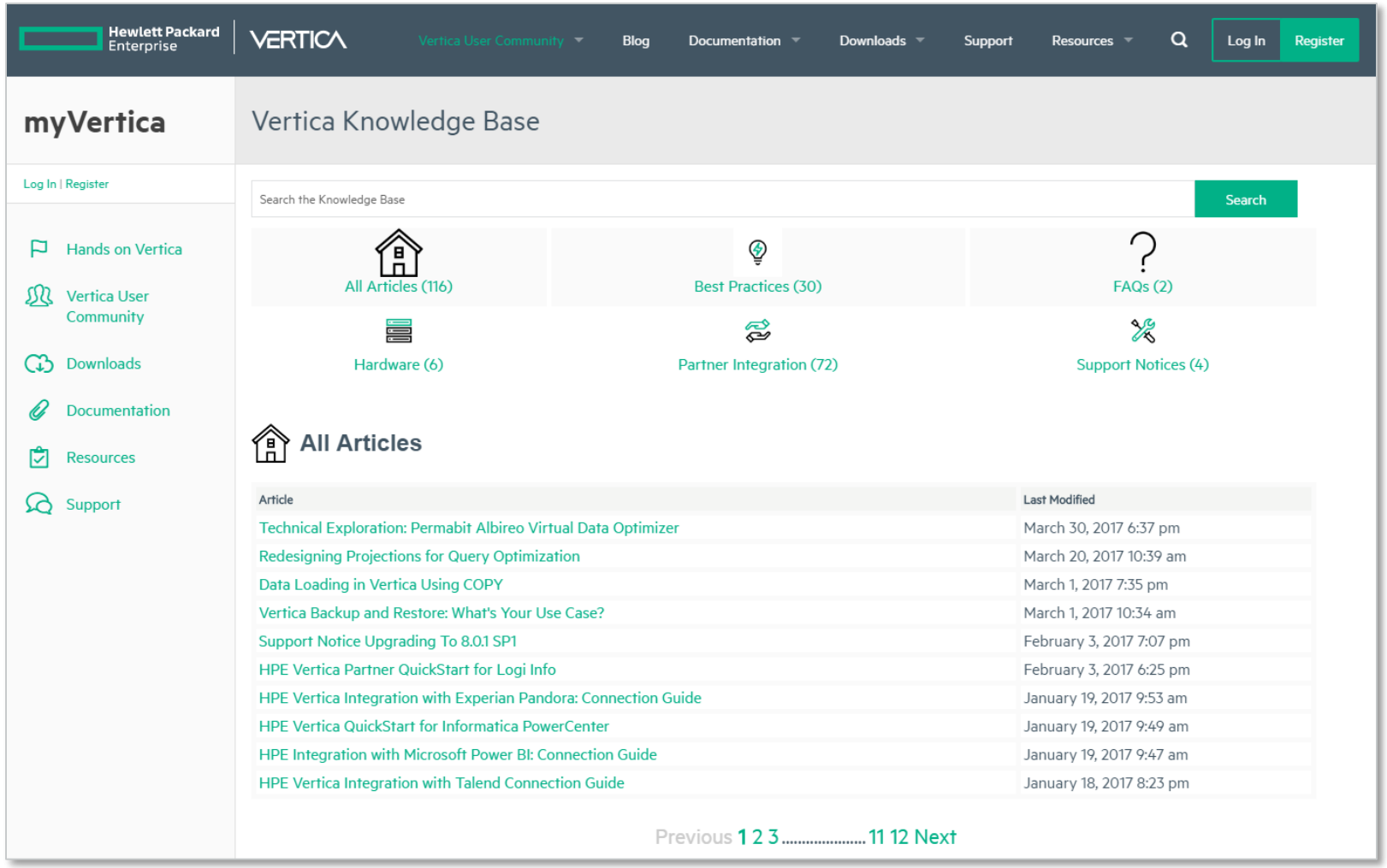
Quick access to my.Vertica resources



The Vertica Blog



The Vertica Knowledge Base



VERTICA

What's New: 8.1 Release

Nga Tran
Waqas Dhillon

Hewlett Packard Enterprise

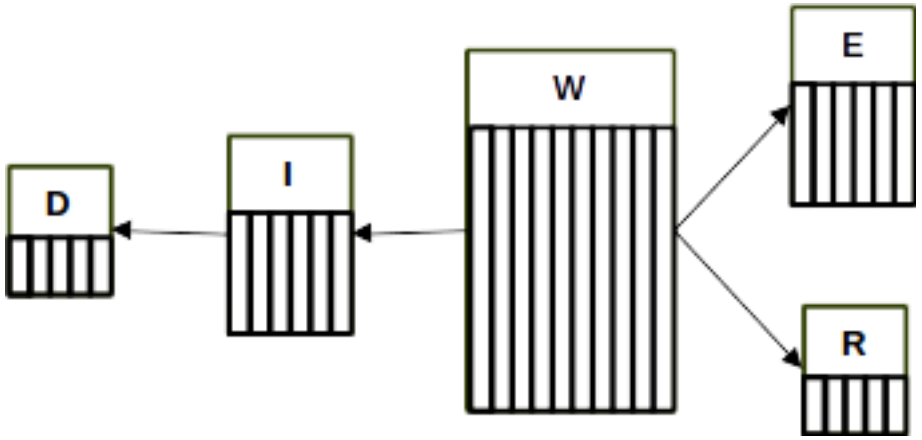
Flattened Tables

What is a Flattened Table?

Let us go through examples first

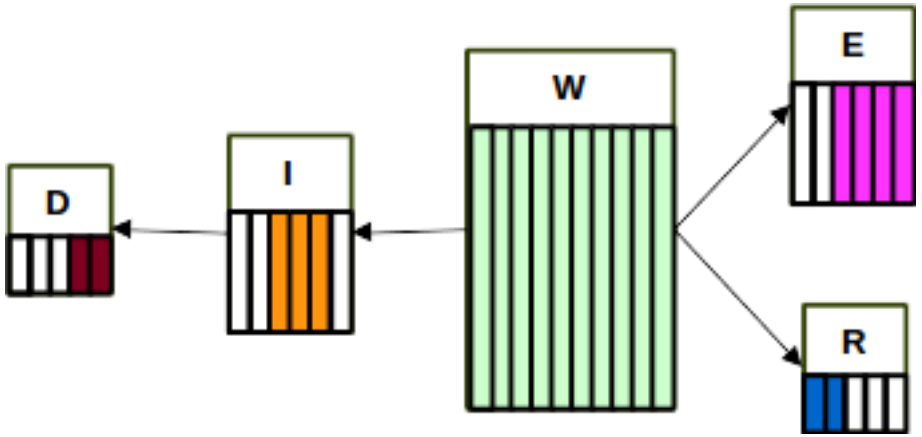
Normalized Schema

No Data Redundancy



Normalized Schema

No Data Redundancy



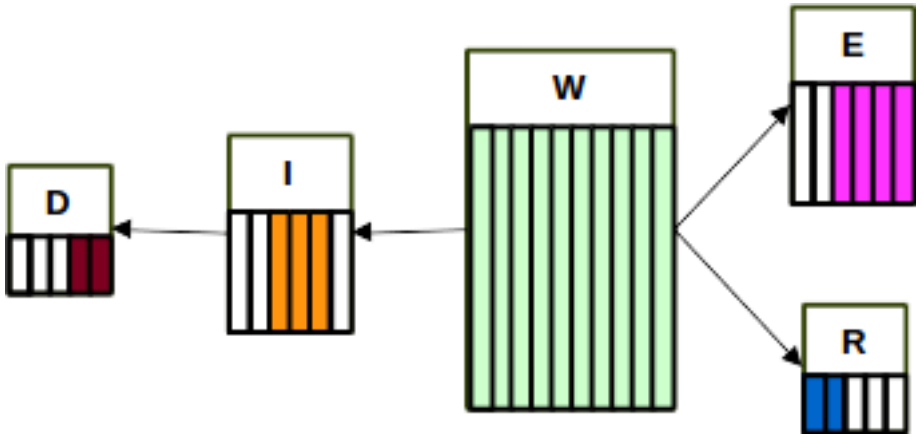
Common queries include

- Subset of **color** columns
- Many **joins**
  - --> Expensive/Slow
  - --> Hard to tune

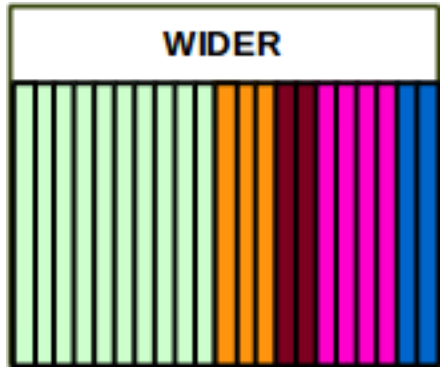
Manually Denormalized Schema – Before 8.1

Normalized Schema + Denormalized Tables

Normalized Schema



Denormalized Table



Common queries include

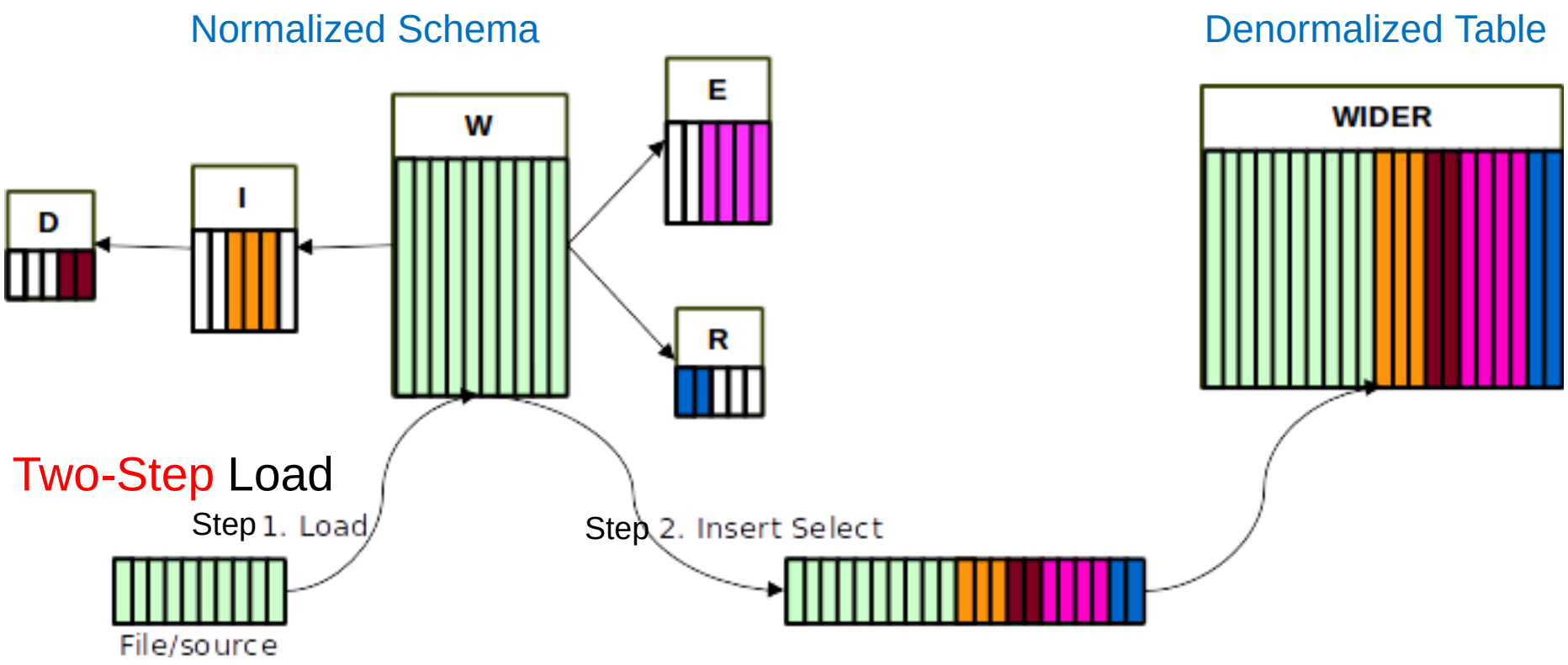
- Subset of **color** columns
- Many **joins**
  - --> Expensive/Slow
  - --> Hard to tune

Common queries includes

- Subset of **color** columns
- **No joins**
  - --> Cheap/Fast
  - --> No extra tuning

Manually Denormalized Schema – Before 8.1

Extra Maintenance Cost

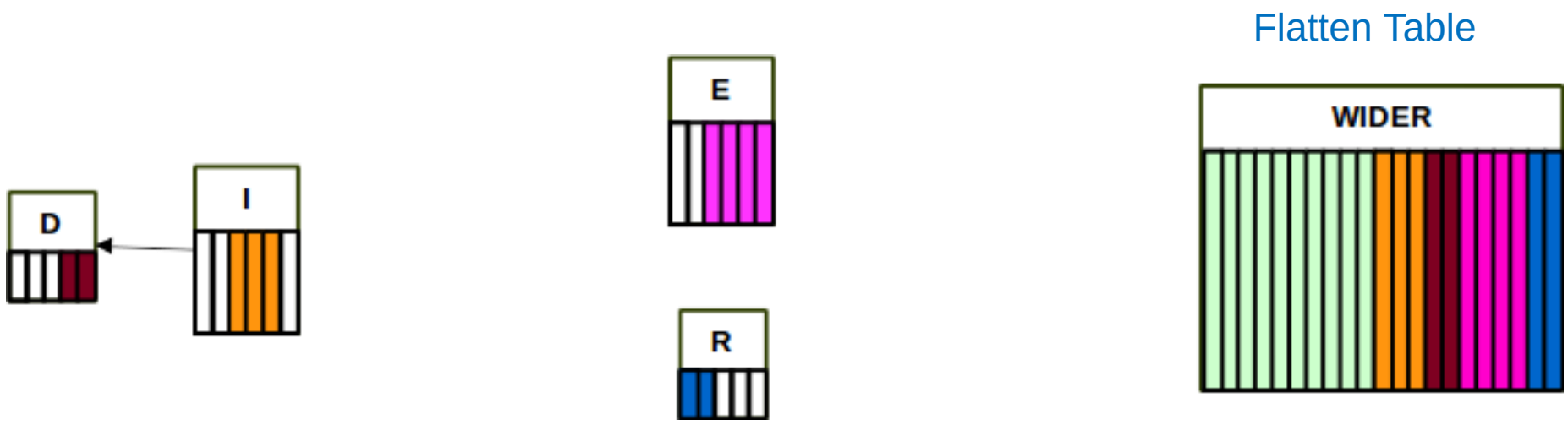


I. Two-Step Load

II. Manually update table **WIDER** if any of tables **D, E, I, R** gets updated
Many of our customers even choose to rebuild table **WIDER** from scratch

Vertica Flattened Tables – 8.1

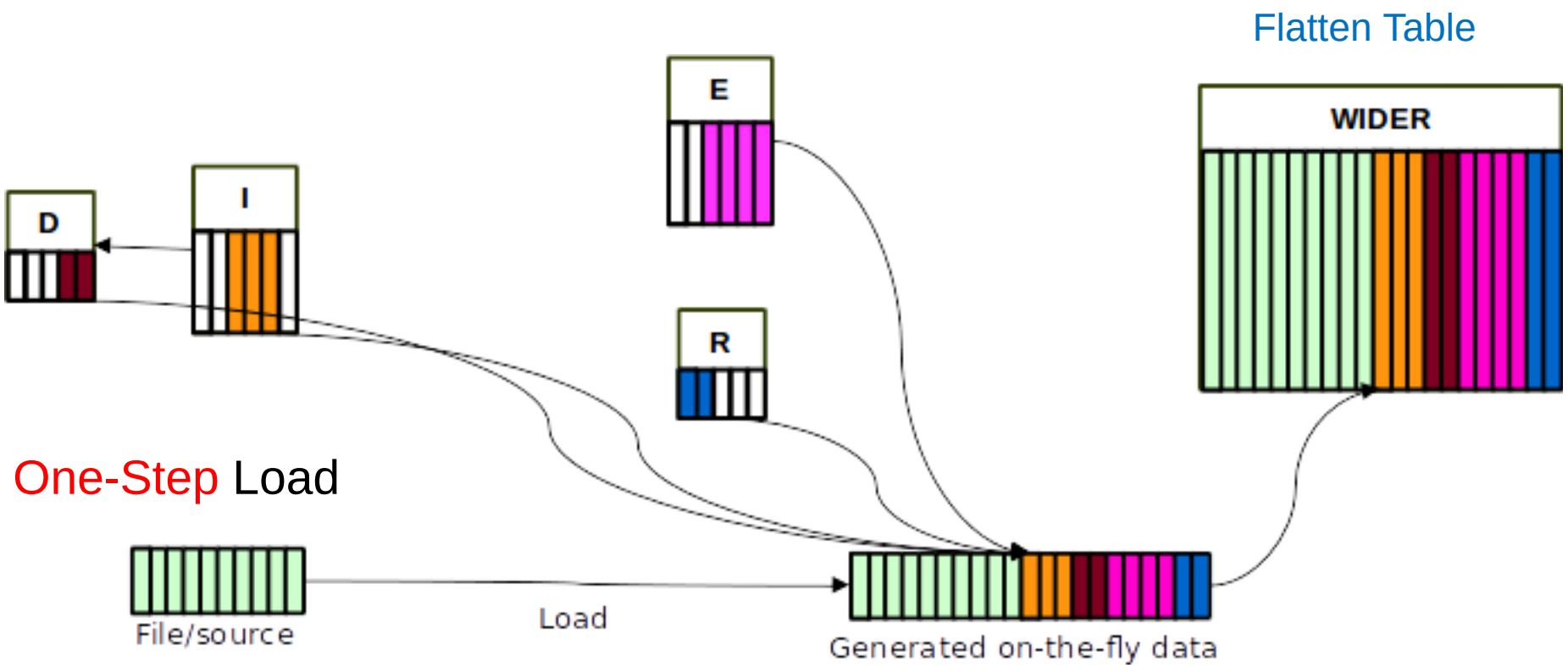
Less Redundancy



- **WIDER** is a Flattened Table which is just a table
- No longer need table **W**
- **I, E, R** have (FK-PK) relationships with table **WIDER** through different definition (see next)

Vertica Flattened Tables – 8.1

Easy Maintenance, Low Cost



I. One-Step Load

II. New syntax refresh_column to update **WIDER** if any of **D, E, I, R** gets updated

III. Easy and effective add and drop any columns

So what is a Vertica Flattened Table?

Short definition

Flattened Table = Denormalized Table of (a part of) a Normalized Schema (Star Schema)

So what is a Vertica Flattened Table?

Vertica goes beyond that

- ***Flattened Table*** = **Table** with normalized and denormalized columns

- A ***column*** of a Flattened Table is
  - **Defined** as an expression or a query of many tables (q)
  - **Added** and **dropped** as needed, at any point in time
  - **Populated** when
    - Data of other columns loaded, or
    - The column is added, or
    - The column is refreshed
  - **Not automatically refreshed** when data of its defined query(q) get changed
    - Performance is key
    - Operations on flattened tables are decoupled from those on the source and vice versa

  → There is no overhead on performance of the flattened tables when the values in the source are changing

Create and Use a Flattened Table

Create a Flattened Table

Snippet of a CREATE TABLE statement

- ***Denormalized column*** is defined using **DEFAULT** and **SET USING** key words

```
CREATE TABLE flattenedTab
(
    col1      …..
    col2      …..
    colX      int  DEFAULT    ( SELECT val FROM dim WHERE (f_dkey = d_key) )
                   SET USING ( SELECT val FROM dim WHERE (f_dkey = d_key) )
    coln        …..
    ……
);
```

Create a Flattened Table

Let us go through examples

Create a Flattened Table

Normalized Tables: a dimension (custDim) and a fact (orderFact)

custDim		
Cid	**Name**	**Age**
1	Alice	25
2	Bob	30
3	Eve	28

```
CREATE TABLE custDim (
        cid     int PRIMARY KEY,
        name varchar(20),
        age     int
);
```

order Fact		
Order_id	**Cust_id**	**Amount**
100	1	15.00
200	1	1000.00
300	2	-50.00
400	3	100.00
500	2	200.00

```
CREATE TABLE orderFact (
        order_id int PRIMARY KEY,
        cust_id   int
                REFERENCES custDim(cid),
        amount   numeric;
);
```

VERTICA Hewlett Packard Enterprise

Create a Flattened Table

Convert orderFact into a Flattened Table

orderFact			
Order_id	**Cust_id**	***Cust_name***	**Amount**
100	1	*Alice*	15.00
200	1	*Alice*	1000.00
300	2	*Bob*	-50.00
400	3	*Eve*	100.00
500	2	*Bob*	200.00

```
CREATE TABLE orderFact (
        order_id      int PRIMARY KEY,
        cust_id       int
                REFERENCES custDim(cid),
        cust_name varchar(20) DEFAULT (
                SELECT name FROM custDim
                WHERE custDim.cid = cust_id
        ),
        amount        numeric;
);
```

- Starting 8.1
  - DEFAULT column can be a subquery
  - It is a sign of a Flattened Table

Create a Flattened Table

Full Schema

custDim		
Cid	**Name**	**Age**
1	Alice	25
2	Bob	30
3	Eve	28

```
CREATE TABLE custDim (
        cid     int PRIMARY KEY,
        name varchar(20),
        age     int
);
```

orderFact			
Order_id	**Cust_id**	***Cust_name***	**Amount**
100	1	*Alice*	15.00
200	1	*Alice*	1000.00
300	2	*Bob*	-50.00
400	3	*Eve*	100.00
500	2	*Bob*	200.00

```
CREATE TABLE orderFact (
        order_id     int PRIMARY KEY,
        cust_id      int
                REFERENCES custDim(cid),
        cust_name varchar(20) DEFAULT (
                SELECT name FROM custDim
                WHERE custDim.cid = cust_id
        ),
        amount       numeric;
);
```

DML with DEFAULT columns

INSERT data into DEFAULT column

order Fact			
Order_id	**Cust_id**	**Cust_name**	**Amount**
100	1	Alice	15.00
200	1	Alice	1000.00
300	2	Bob	-50.00
400	3	Eve	100.00
500	2	Bob	200.00
600	3	Eve	10.00

De-normalized Value

```
INSERT INTO orderFact VALUES (600, 3, default, 10.00);
INSERT INTO orderFact(order_id, cust_id, amount) VALUES (600, 3, 10.00);
INSERT INTO orderFact(order_id, cust_id, amount) SELECT 600, 3, 10.00;
```

DML with DEFAULT columns

COPY/Load data into DEAFULT column

order Fact			
Order_id	**Cust_id**	**Cust_name**	**Amount**
100	1	Alice	15.00
200	1	Alice	1000.00
300	2	Bob	-50.00
400	3	Eve	100.00
500	2	Bob	200.00
600	3	Eve	10.00

De-normalized Value

```
COPY orderFact(order_id, cust_id, amount) FROM stdin delimiter ',';
600, 3, 10.00
\.
-- Default placeholder is not supported for COPY
```

DML with DEFAULT columns

UPDATE data of DEFAULT column

custDim		
Cid	**Name**	**Age**
1	Alice	25
2	Bob	30
3	**Charlie**	28

orderFact			
Order_id	**Cust_id**	***Cust_name***	**Amount**
100	1	*Alice*	15.00
200	1	*Alice*	1000.00
300	2	*Bob*	-50.00
400	3	***Charlie***	100.00
500	2	*Bob*	200.00

```
UPDATE custDim SET name = 'Charlie' WHERE cid = 3;
UPDATE orderFact SET cust_name = default WHERE cust_id = 3;
```

DDL with DEFAULT columns

Add a DEFAULT column

order Fact				
Order_id	**Cust_id**	**Cust_name**	**Amount**	**Cust_age**
100	1	Alice	15.00	25
200	1	Alice	1000.00	25
300	2	Bob	-50.00	30
400	3	Eve	100.00	28
500	2	Bob	200.00	30

```
ALTER TABLE orderFact ADD COLUMN cust_age int
    DEFAULT (SELECT age FROM custDim WHERE custDim.cid = cust_id);
```

That is not all

Flattened Table can do more

SET USING columns

New keyword “SET USING”

order Fact			
Order_id	**Cust_id**	**Cust_name**	**Amount**
100	1	-	15.00
200	1	-	1000.00
300	2	-	-50.00
400	3	-	100.00
500	2	-	200.00

De-normalized column not populated during loading
(INSERT, INSERT SELECT, COPY)

```
CREATE TABLE orderFact (
        order_id      int PRIMARY KEY,
        cust_id       int
                REFERENCES custDim(cid),
        cust_name varchar(20) SET USING (
                SELECT name FROM custDim
                WHERE custDim.cid = cust_id
        ),
        amount        numeric;
);
```

- **SET USING**
  - Another sign of Flattened Table

SET USING columns

Refresh as needed

order Fact			
Order_id	**Cust_id**	**Cust_name**	**Amount**
100	1	Alice	15.00
200	1	Alice	1000.00
300	2	Bob	-50.00
400	3	Eve	100.00
500	2	Bob	200.00

De-normalized column populated as needed
(SELECT refresh_columns)

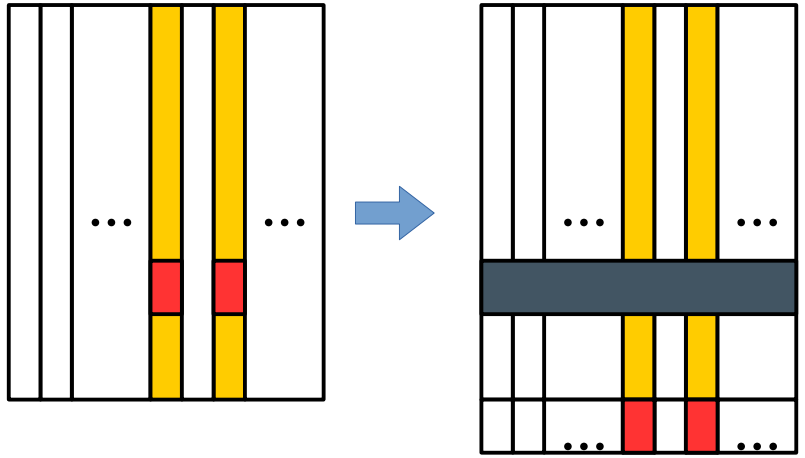
```
-- Populate all SET USING columns
SELECT REFRESH_COLUMNS ('orderFact', '');

-- Populate a specified SET USING column
SELECT REFRESH_COLUMNS ('orderFact', 'cust_name');
```

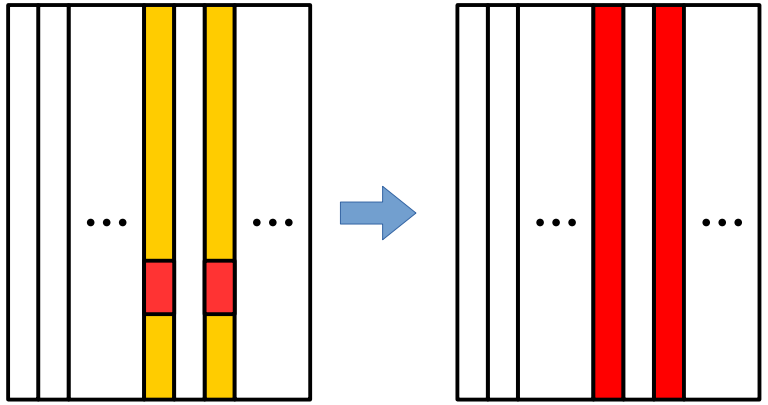
SET USING columns

Two different refresh methods

Via UPDATE



Via REBUILD



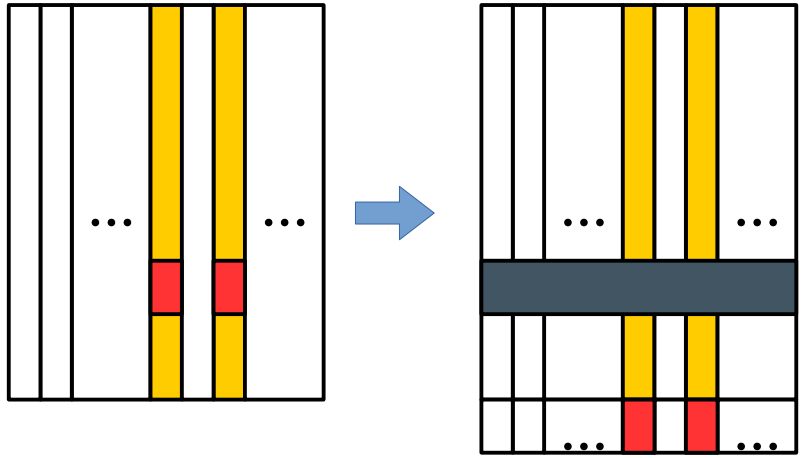
```
--  Via UPDATE
SELECT REFRESH_COLUMNS ('orderFact', ‘cust_name‘, ‘UPDATE’);

-- Via REBUILD
SELECT REFRESH_COLUMNS ('orderFact', 'cust_name‘, ‘REBUILD’);
```

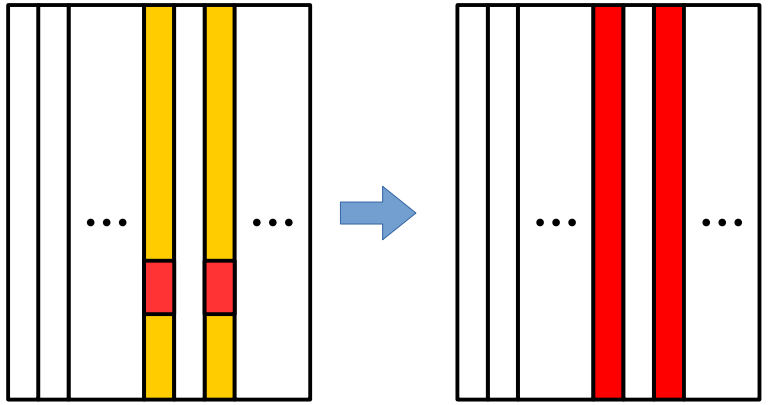
SET USING columns

Two different refresh methods

Via UPDATE



Via REBUILD



- Refresh via UPDATE can be done on any kind of columns
- Refresh via REBUILD can only work on columns that are **not** in
  - Table's partition key
  - Projection's segmentation and sort order

SET USING columns

Refresh many Flattened Tables in the same transaction

```
SELECT REFRESH_COLUMNS ('Table1, Table2, Table3', '', 'UPDATE');

SELECT REFRESH_COLUMNS ('Table1, Table2, Table3', '', 'REBUILD');
```

SET USING columns

DEAULT column vs SET USING column

	DEFAULT	SET USING
New syntax?	DEFAULT is not new, but DEFAULT as subquery is new	SET USING is new
Populate value	Real time (during loading)	When needed (REFRESH_COLUMNS)
Add column	Calculate value	Not calculate value
Sync with modified dimensions	Need to run UPDATE or DROP/ADD column manually	Call REFRESH_COLUMNS
System table	SELECT column_default FROM columns	SELECT column_set_using FROM columns

SET USING columns

DDL

Action	DDL
Create table	CREATE TABLE [tableName] (… [column name] [column type] **SET USING** [expression]);
Add column	ALTER TABLE [table name] ADD COLUMN [column name] [column type] **SET USING** [expression];
Drop attribute	ALTER TABLE [table name] ALTER COLUMN [column name] DROP **SET USING**;
Set attribute	ALTER TABLE [table name] ALTER COLUMN [column name] **SET USING** [expression];

- A column can have **both** DEFAULT and SET USING
- DEFAULT and SET USING of the **same column** can be **different** expressions/queries

More functionalities

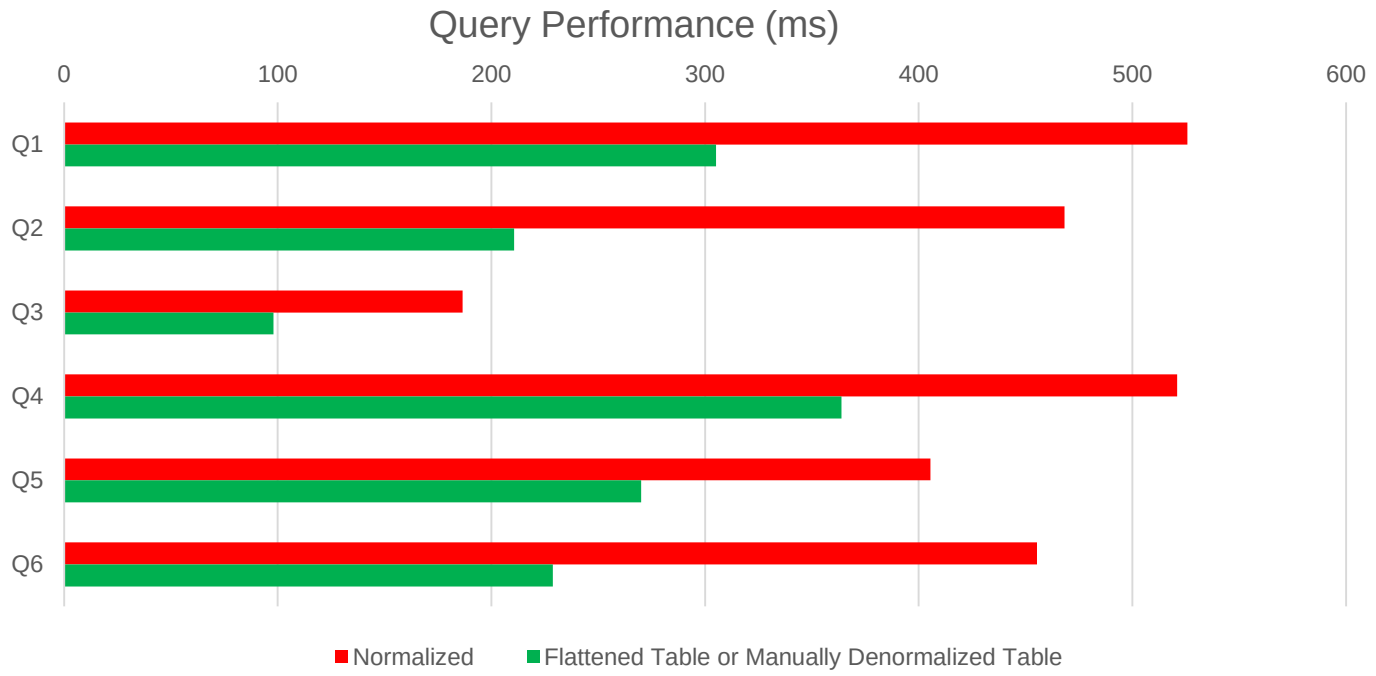
- DEFAULT/ SET USING definition can be any subquery (return single value), not limited to PK-FK
  - Aggregation (count, count distinct, max, min, avg, …)
  - Analytic functions (median, ntile, percentile, …)
  - Top-K (LIMIT 1)
  - Set operations (UNION, INTERSECT, EXCEPT/MINUS)
  - Merge query
- Thus, not only used to de-normalize for star/snowflake schema

Flattened Tables

Performance Numbers

Benefits of Flattened Table

Queries with Joins are always Faster on Flattened Table

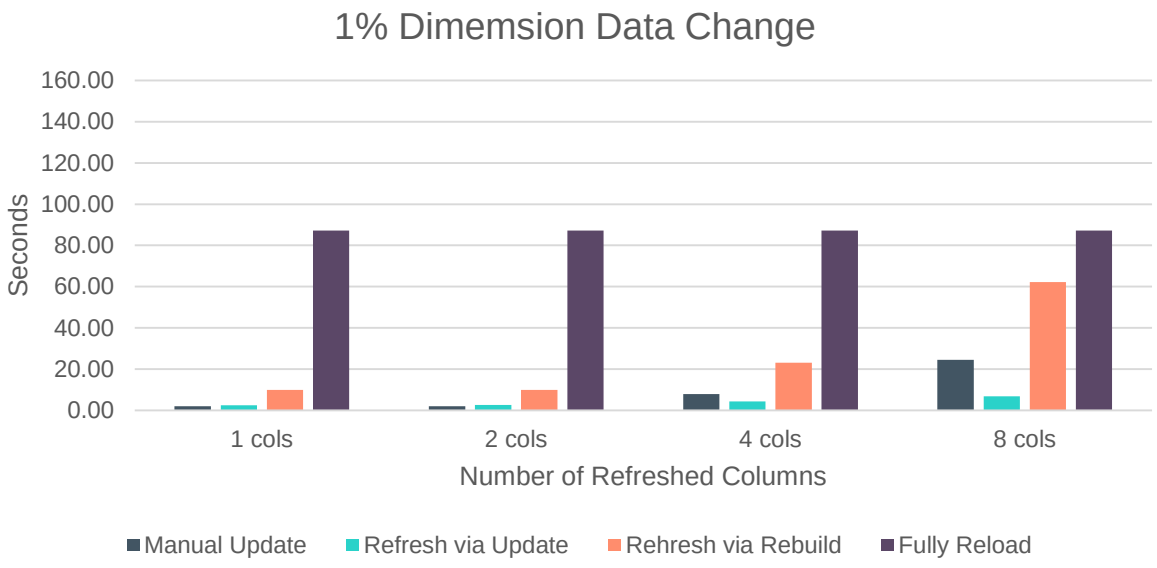


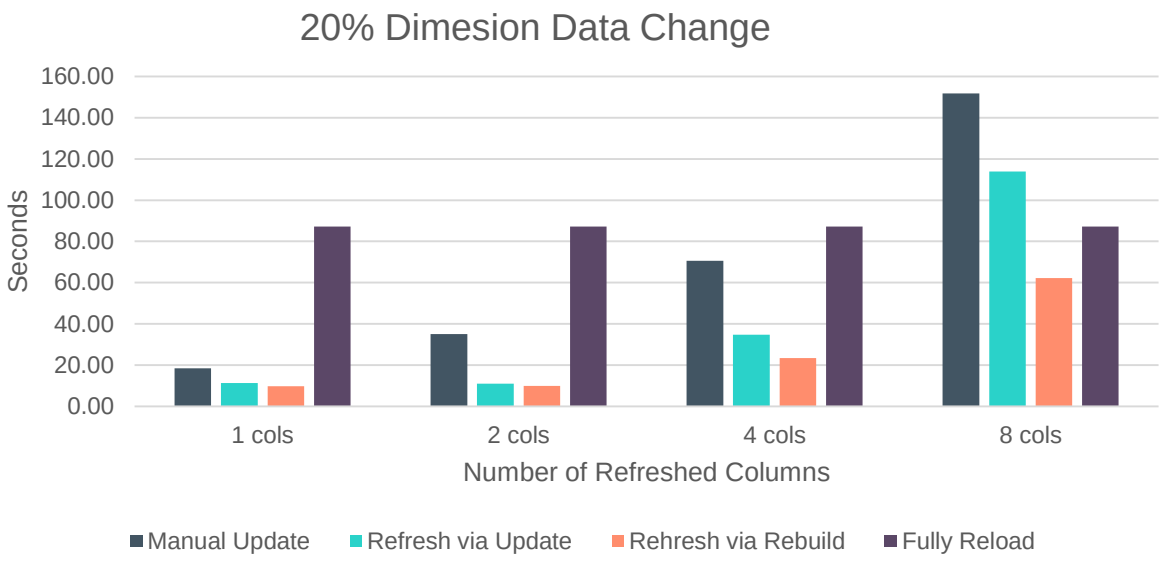
Load Performance

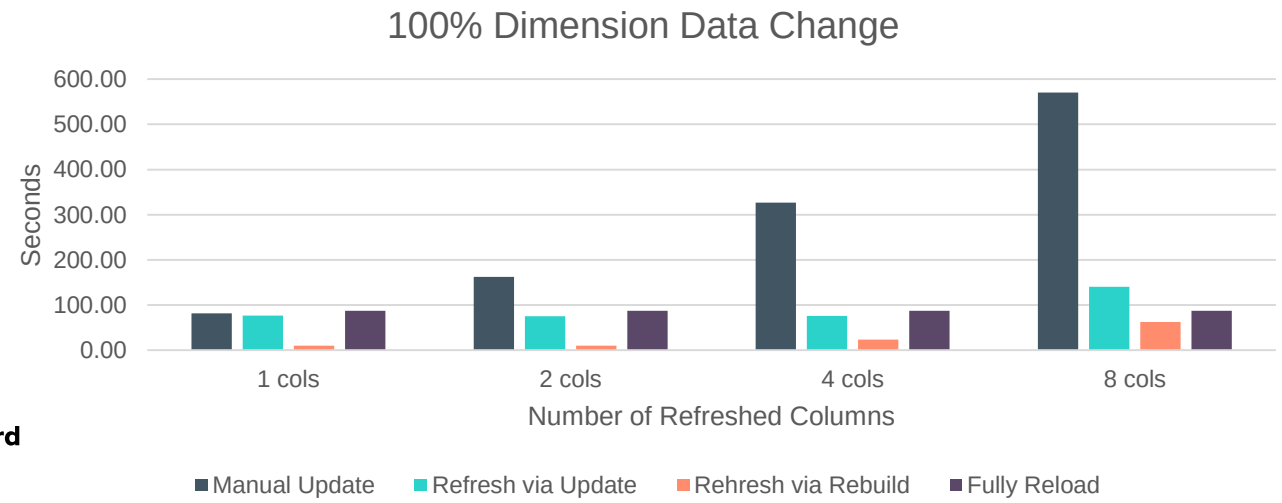
Around Twice Faster the Work Needed without Flattened Table

Refresh Performance

Choose Refresh Method depending on Data Change







VERTICA

Hewlett Packard Enterprise

Kafka Integration

Why Vertica + Kafka is a great choice

Existing users of Kafka

- Vertica is complimentary
- Many large players in the industry have started using Kafka to decrease complexity in their ETL pipelines
- Vertica has connector to move data bi-directional (in/out) with Kafka
- Vertica can do interactive SQL based analytics on very large at rest data sets

Reasons to start using Vertica + Kafka

- Complex ETL pipelines
- Hard to meet throughput and latency requirements
- Need for more real-time information on dashboards
- New projects around IOT, streaming or real-time data analysis
- Analyzing semi-structured data like web logs, clickstream or sensors

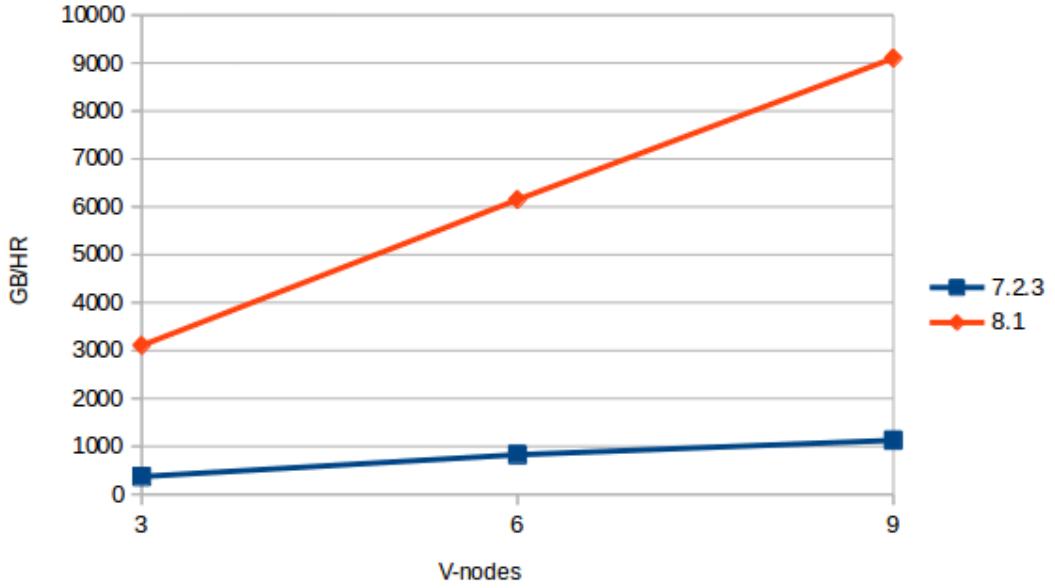
Maximize Kafka load parallelism by using all available threads

User defined source is now aware of how many threads are available to execute the COPY statements. Vertica exposes LoadUnion Parallelism to UDSources.

Benefit: Linear scaling vs. nodes with increased throughput

JSON Load Performance
7.2.3 vs 8.1

- Each partition 20 GB of 5 KB JSON messages.
- At 10 partitions per node we observe ~2.6 gigabits/second/node, or 40k (large, complex) messages/sec/node.
- In 9 node case, with 8.1, at 10 partitions per node we see 360k of 5 KB JSON messages processed per second, or ~23 gigabits/sec total



SSL for enhanced Kafka – Vertica security

– Secure connection between Kafka brokers and Vertica nodes – taking advantage of the SSL support introduced in Kafka 0.9

Kafka Schema Registry flexibility to handle changing IoT data

to reduce downstream maintenance

- Data schema evolves over time creating lot of downstream work for consuming systems
- Kafka platform introduced a centralized schema registry to solve this problem
- Vertica Microbatch supported loading of Avro messages in which schema can be
  - Included in the message
  - Set as a parser parameter
- Adding more manageability and flexibility with a third choice – using schema registry
  - Allows Kafka Avro Parser to parse and decode the message written by Schema registry
  - Also retrieves the Avro schemas stored in SR

Spark Integration

Why Vertica + Spark is a great choice

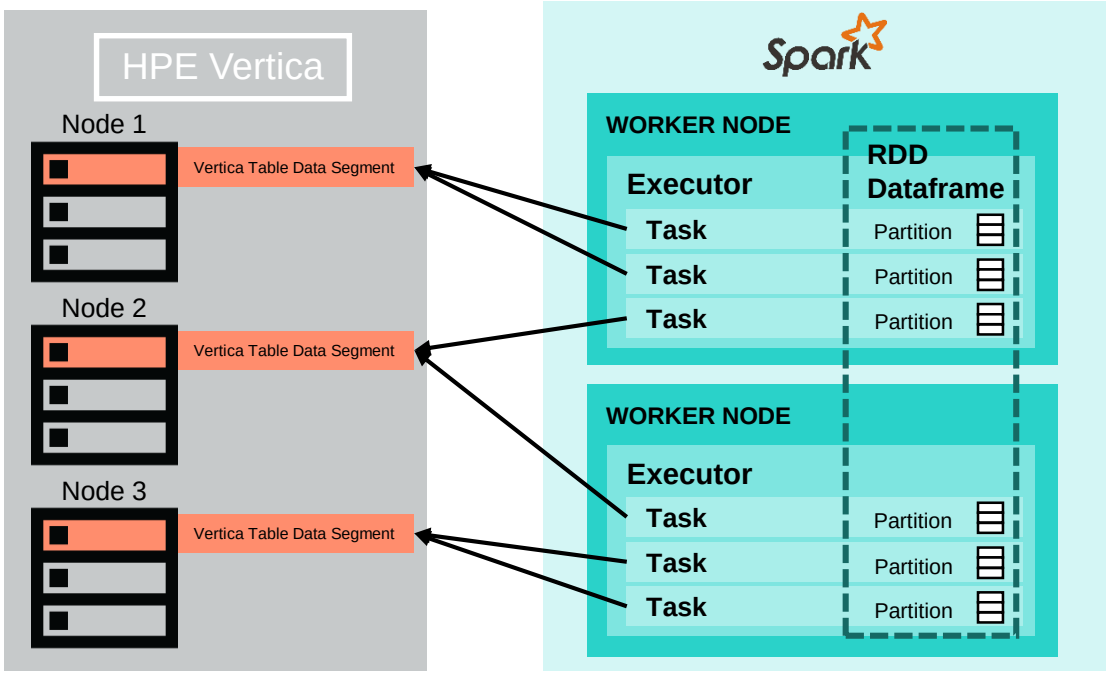
- Vertica is complimentary technology for Spark
- Vertica has connector to move data bi-directional (in/out) with Spark
- Vertica can do interactive SQL based analytics on very large at rest data sets
- Spark is focused on in memory operations
- In some of our largest customers Spark is used in the streaming data pipeline for transformations before data gets to Vertica

Spark 2.0 now integrates with Vertica

Bi-directional data connector optimized for Vertica data layout

- **Spark 2.0 improvements:** API stability, SQL 2003 support, subquery support, Improved ORC and Parquet performance, and unification of data frame with data sets.

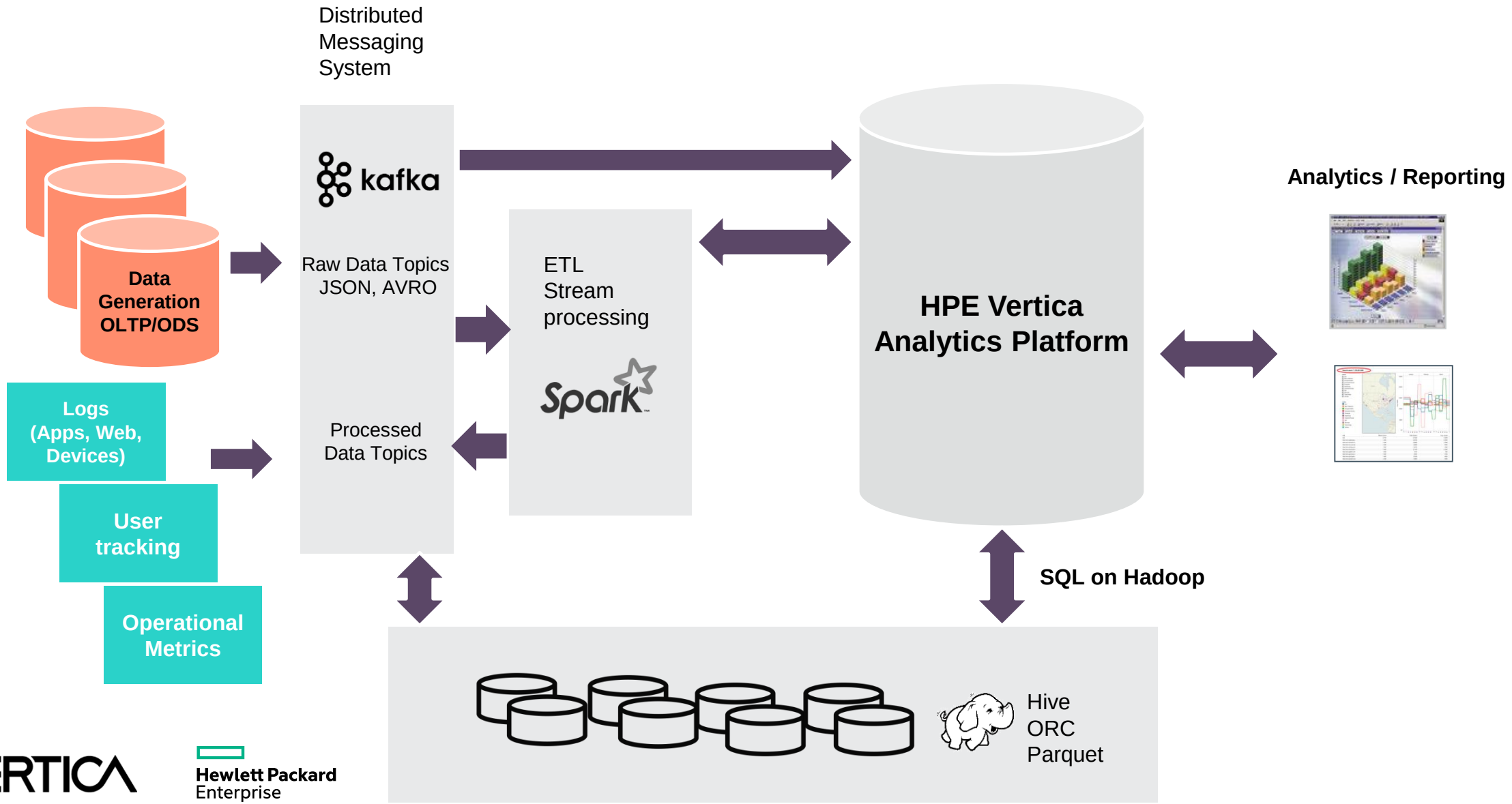
- **Benefits of Vertica <–> Spark integration:**
  - Fast, scalable ETL - loading leverages Vertica's parallelism, HDFS connectivity, and fluency with open source data formats
  - Vertica supplies data to users doing data analytics in Spark. When Spark dataframe needs data from Vertica, there is optimized query processing.
  - Spark users can benefit from Vertica's very advanced SQL analytics
  - Spark users benefit from Vertica's integration with several BI tools for creating dashboards



VERTICA

Hewlett Packard Enterprise

Apache Kafka + Spark + HPE Vertica for both Batch and Streaming Analytics



VERTICA

Hewlett Packard Enterprise

VERTICA

Did You Know: Backup and Restore

Eugenia Moreno

Hewlett Packard Enterprise

Did You Know: Backup and Restore

Vertica backup utility /opt/vertica/bin/vbr does several tasks

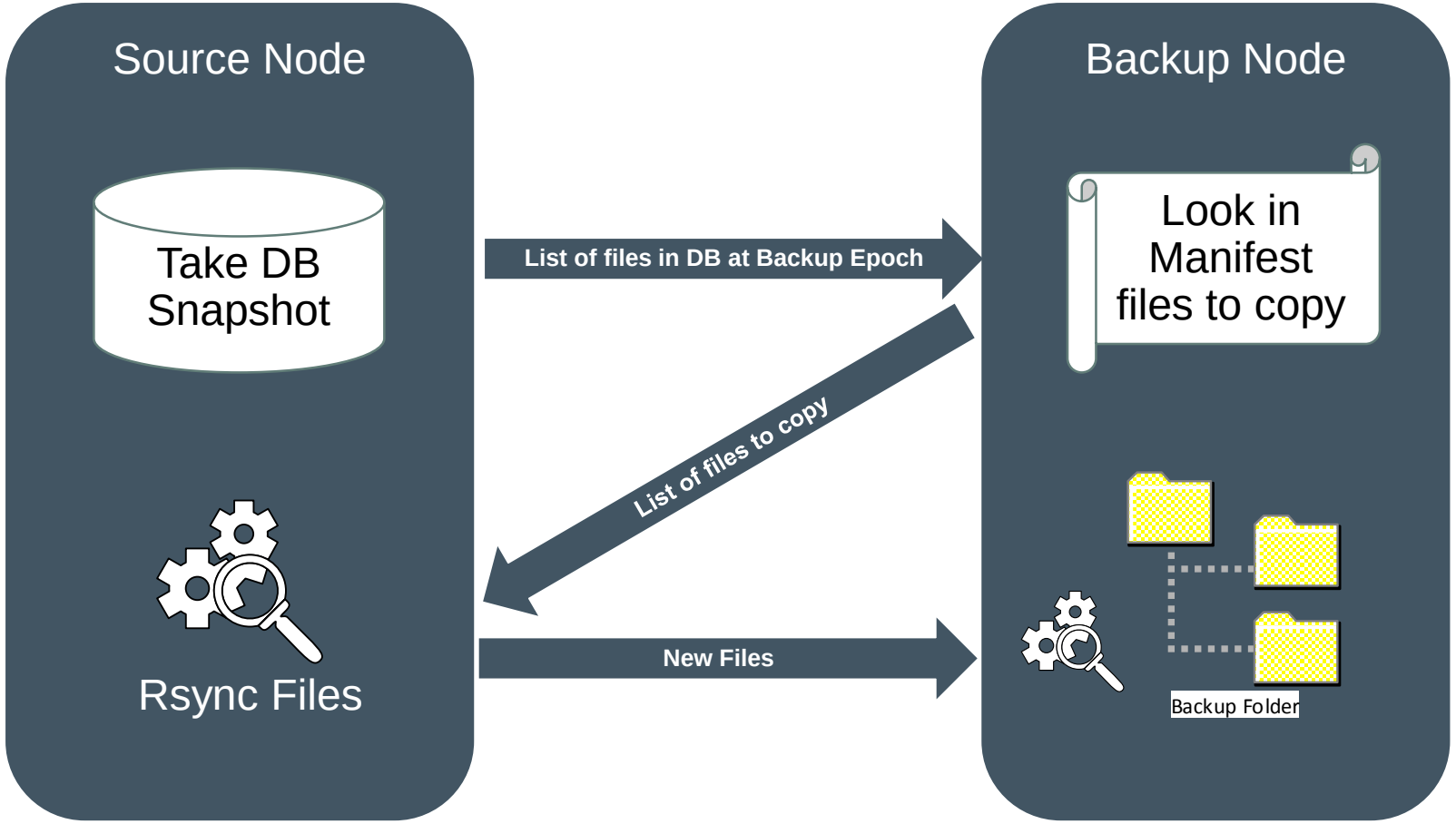
Backup/Restore tasks

- Backup
- Restore
- Copycluster
- Replicate

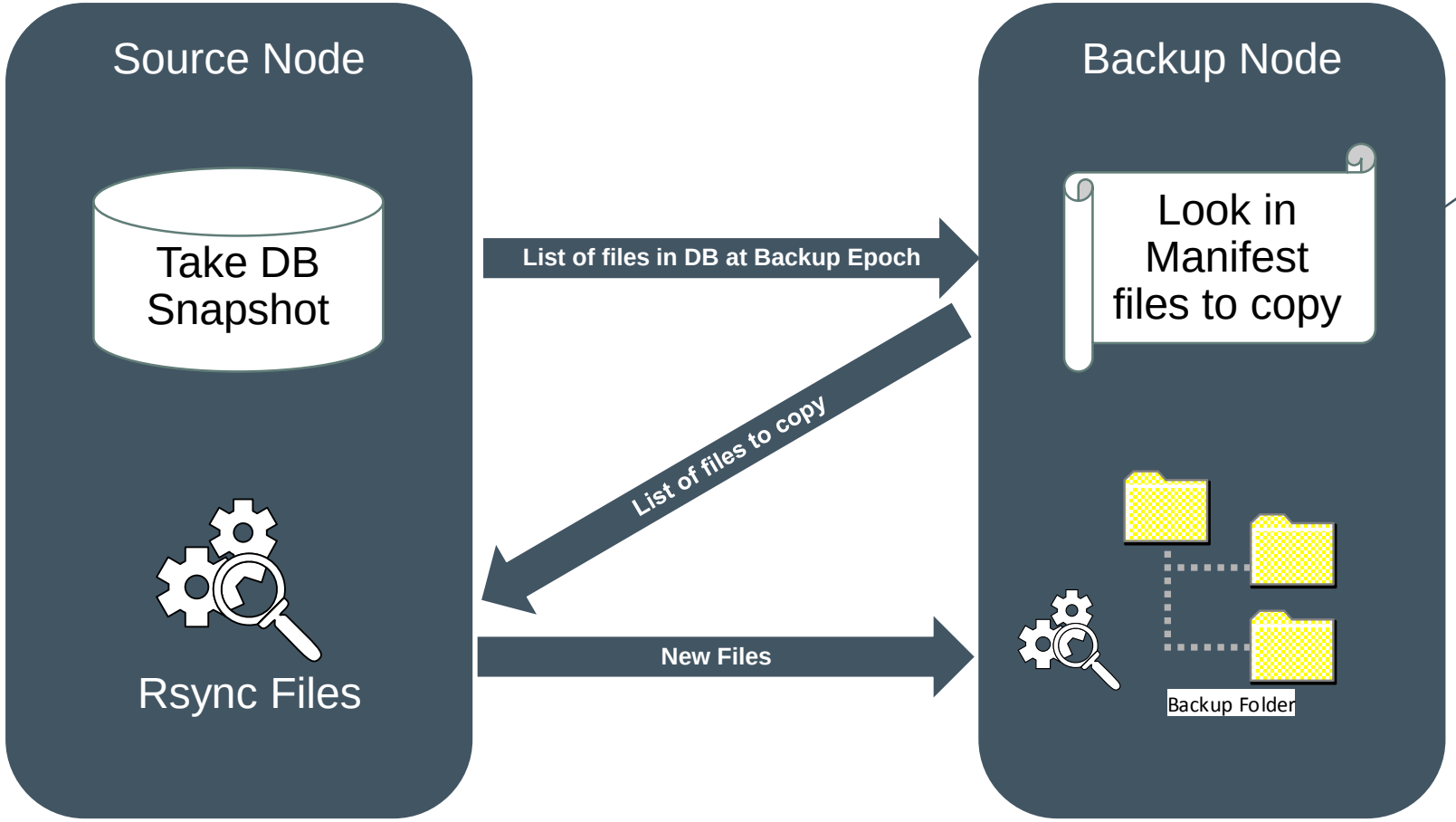
Maintenance Tasks

- Init
- 7.2_ugrade
- Listbackup
- Quick-check
- Full-Check
- Quick-repair
- Collect-garbage
- Remove

Backup: how it works



Backup: how it works



Backup Node Can be:
- Local Node
- Remote NFS
- Hard Link

Restoring Backup

– Full Restore
  – On Original Cluster
  – New Cluster
  – Cluster is down.

```
[dbadmin@emoreno1 ~]$ /opt/vertica/bin/vbr --task restore --config backup_test.ini
Starting full restore of database vmart.
Participating nodes: v_vmart_node0001, v_vmart_node0002, v_vmart_node0003.
Restoring from restore point: backup_snapshot_20170410_194328
Determining what data to restore from backup.
[==================================================] 100%
Approximate bytes to copy: 731492352 of 1349946254 total.
Syncing data from backup to cluster nodes.
[==================================================] 100%
Restoring catalog.
Restore complete!
```

Restoring Backup

– Restore a single object of full backup
  – Cluster UP

```
[dbadmin@emoreno1 ~]$ /opt/vertica/bin/vbr --task restore --restore-objects store.store_orders_fact --config backup_test.ini
Starting object restore of database vmart.
Participating nodes: v_vmart_node0001,v_vmart_node0002,v_vmart_node0003.
Objects to restore: store.store_orders_fact.
Enter vertica password:
Restoring from restore point: backup_snapshot_20170412_205927
Loading snapshot catalog from backup.
Extracting objects from catalog.
Approximate bytes to copy: 18487828.
Syncing data from backup to cluster nodes.
Progress: 18487828 out of 18487828 bytes (100%)
Finalizing restore.
Restore complete!
```

Copy cluster

– You can copy a full database to other location
  – It has to have the same number of nodes.
  – Database copy is full, data copy is incremental.
  – Source Cluster UP – Target Cluster DOWN.

```
[dbadmin@emorenol ~]$ vbr --task copycluster --config-file copy_test.ini --debug 3
Configured backup directories in mapping are ignored for copy cluster.
Starting copy of database vmart.
Participating nodes: v_vmart_node0001, v_vmart_node0002, v_vmart_node0003.
Enter vertica password:
Snapshotting database.
Snapshot complete.
Determining what data to copy.
Progress: 589 out of 589 objects (100%)
Approximate bytes to copy: 1939944020 of 1939944020 total.
Syncing data to destination cluster.
Progress: 1939944020 out of 1939944020 bytes (100%)
Reinitializing destination catalog.
Copycluster complete!
[dbadmin@emorenol ~]$
```

Replicate Objects

- You can replicate objects from one cluster to other
  - It has to have the same number of nodes
  - For replication Target Cluster is UP

```
[dbadmin@emoreno1 ~]$ vbr --task replicate  --config-file copy_test.ini --debug 3
Configured backup directories in mapping are ignored for object replication.
Starting replication of objects ['store.store_orders_fact'] from vmart.
Participating nodes: v_vmart_node0001, v_vmart_node0002, v_vmart_node0003.
Enter vertica password:
Snapshotting the source database.
Snapshot complete.
Copying catalog snapshot from source to destination.
Preparing destination database for replicating objects.
Prep complete, start syncing files.
Approximate bytes to copy: 18487828 of 36975656 total.
Progress: 18487828 out of 18487828 bytes (100%)
Complete syncing files, removing snapshot from the source database.
Finalizing object replication.
Object replication complete!
```

Backup Maintenance tasks

Task	Description
7.2_ugrade	Upgrades exiting 7.1 or earlier backup to new Manifest based version.
Init	Creates the intial manifest. Needs to be run before start using vbr
Listbackup	Displays the existing backups associated with the configuration file you supply.
Quick-check	Confirms that all backed-up objects appear in the backup manifest. Outputs discrepancies.
Full-Check	Verifies all objects listed in the backup manifest against file system metadata, outputting missing and unreferenced objects.
Quick-repair	Builds a replacement backup manifest, based on storage locations and objects.
Collect-garbage	Rebuilds the backup manifest and deletes any unreferenced objects in the backup location.
Remove	Removes the specified backup or restore point.

Backup and Recovery : Best Practices

- Backup Database
  - Before you upgrade Vertica to another release.
  - Before and after you add, remove, or replace nodes in your database cluster.
  - At regular intervals like daily, weekly.
- Performance: Dedicated Backup host.
- Backup Hosts Accessible from source cluster.
  - Firewall OK, Rsync Port Open
  - Passwordless SSH
  - identical versions of rsync & python
  - Write permission to backup dir and tempDir
- Vertica supports restore actions only to the same exact version of Vertica that created the backup

Backup and Recovery – Best Practices

Hard link Backup

- Use case :
  - During upgrade vertica versions
  - As a staging backup to write to slower backup target location
  - Fast disaster recovery to recover accidentally deleted data
- Advantages :
  - Speed
  - Reduced network activities
  - Less disk space
- Disadvantages
  - If the local disk becomes corrupt, so does the hard link local backup.
  - Hold locks on physical file, needed to free space after ROS are mergeout or removed.
  - Should be removed when no needed or do not keep old backups

Backup and Recovery – Best Practices

- **Interrupting** Backup
  - Backup operations are isolated. Interrupting a backup operation does not affect the previous backup.
  - The previous backup is replaced only as the very last step of backing up your database.
  - Interrupting may leave can leave untracked files in backup location. Use option collect-garbage to clean up.

- **Interrupting Restore**
  - The full restore or copy-cluster operations overwrite the database catalog directory, so interrupting either of these processes leaves the database unusable until you restart the process and allow it to finish.
  - Restore of single object or Replication of objects are transactional and safe to interrupt.

Backup and Recovery

Useful community Docs

- **Vertica Backup and Restore: What is Your Use Case?**

https://my.vertica.com/kb/Vertica-Backup-and-Restore-Whats-Your-Use-Case/Content/BestPractices/Vertica-Backup-and-Restore-Whats-Your-Use-Case.htm

- **Copy and Restore from a Vertica Cluster to a Backup**

https://my.vertica.com/kb/Copy-and-Restore-Data-from-a-Vertica-Cluster-to-a-Backup/Content/BestPractices/Copy-and-Restore-Data-from-a-Vertica-Cluster-to-a-Backup.htm

- **Copying Data Between Dissimilar Clusters**

https://my.vertica.com/kb/Copying-Data-Between-Dissimilar-Vertica-Clusters/Content/BestPractices/Copying-Data-Between-Dissimilar-Vertica-Clusters.htm

- **Copying Data Between Similar Clusters**

https://my.vertica.com/kb/Copying-Data-Between-Similar-Vertica-Clusters/Content/BestPractices/Copying-Data-Between-Similar-Vertica-Clusters.htm

- **Vertica on Amazon Web Services Backup and Restore Guide**

https://my.vertica.com/kb/Vertica-on-Amazon-Web-Services-Backup-and-Restore-Guide/Content/BestPractices/Vertica-on-Amazon-Web-Services-Backup-and-Restore-Guide.htm

VERTICA

Q&A

Do you have additional feedback for our team?

BigDataPlatformCustomerCareTeam@HPE.com

Hewlett Packard
Enterprise