SMART DATA FAST.™

VOLTDB

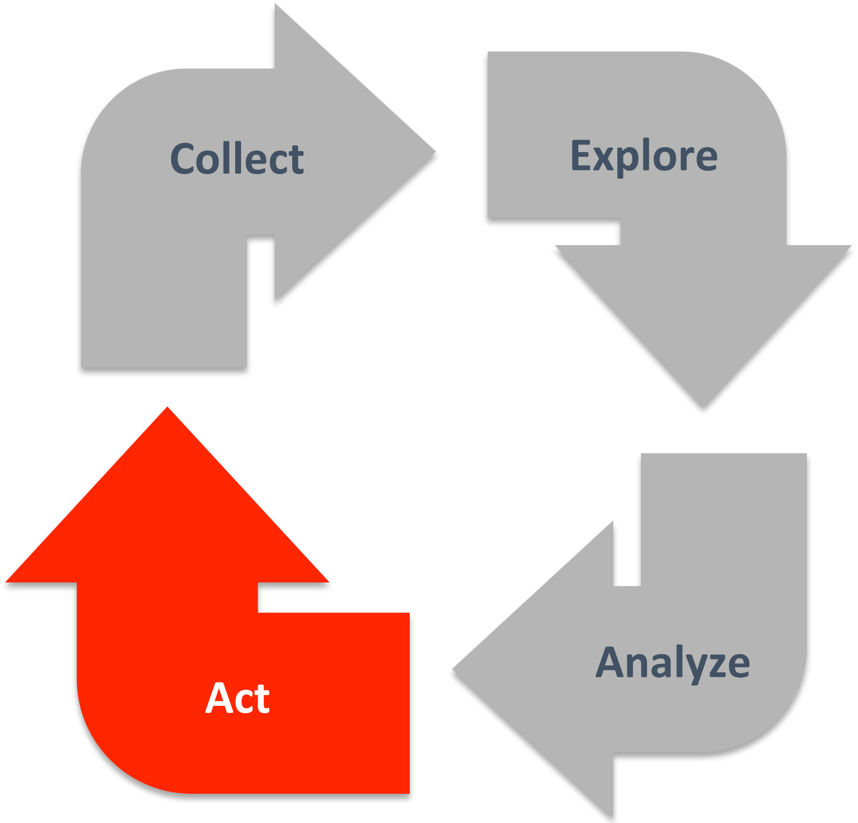
BUILDING FAST DATA APPLICATIONS WITH STREAMING DATA

Ryan Betts, CTO

VoltDB

AGENDA

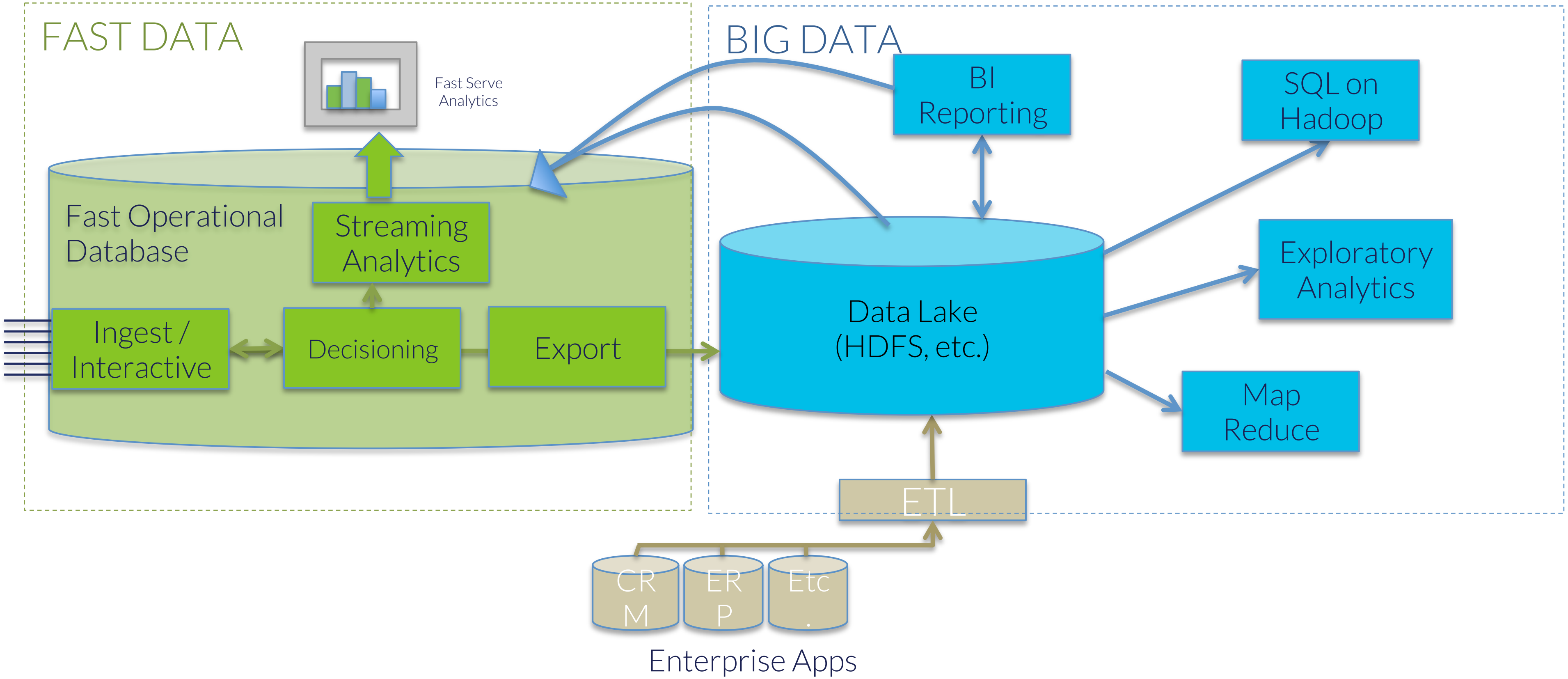
- Fast Data Application Patterns
- Digging Deeper: Looking at the Data
- Streaming Approach
- DB Approach
- Summary



Data leads to applications

Applications create more data

DATA ARCHITECTURE FOR FAST + BIG DATA



IN THE BIG CORNER

Systems facilitating exploration and analytics of large data sets

Example Technologies

Columnar OLAP warehouses

Hadoop Ecosystem

- MapReduce
- Hive, Pig
- SQL.next: Impala, Drill, Shark

Example Applications

- User segmentation & pre-scoring
- Seasonal trending
- Recommendation matrix calculations
- Building search indexes
- Data Science: statistical clustering, Machine learning

IN THE FAST CORNER

Systems facilitating real time ingest, analytics and decisions against incoming event feeds

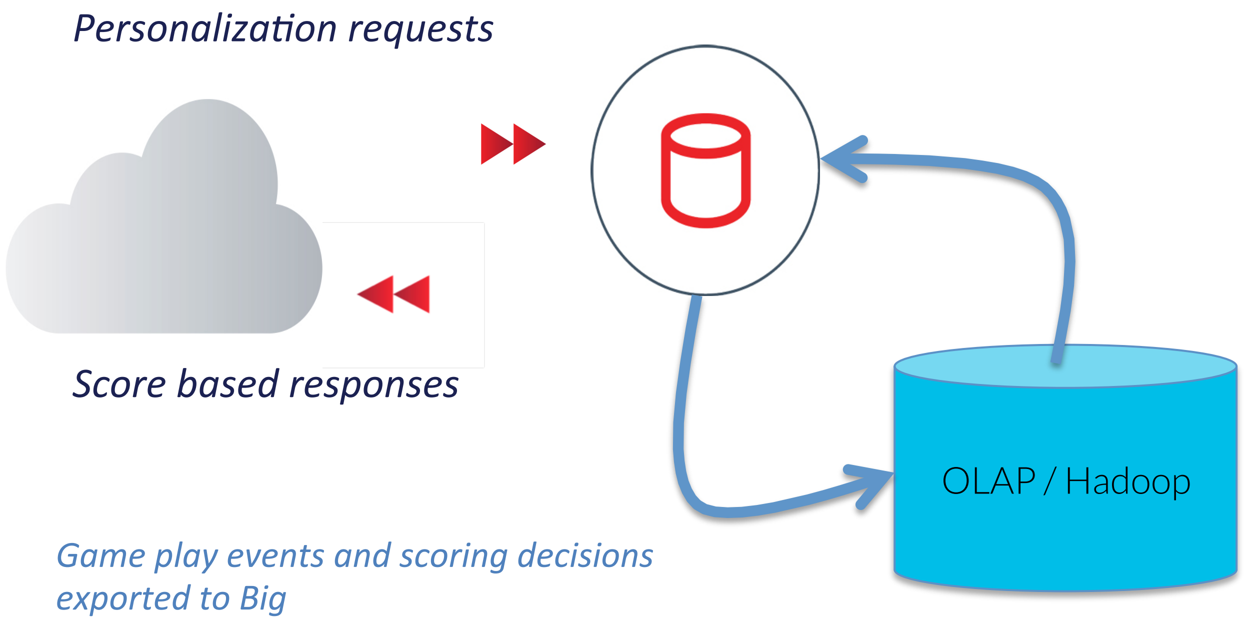
Example Technologies

- Streaming frameworks
- **VoltDB**

Example Applications

- Micro-personalization
- Recommendation serving
- Alerting/alarming
- Operational monitoring
- Data enrichment (ETL elimination)
- High throughput authorization
  - *Ex: API quota enforcement*

REAL TIME SCORING EXAMPLE


User segmentation model Calculated on Big Side and cached in Fast Side
Personalization requests
Score based responses
OLAP / Hadoop
Game play events and scoring decisions exported to Big

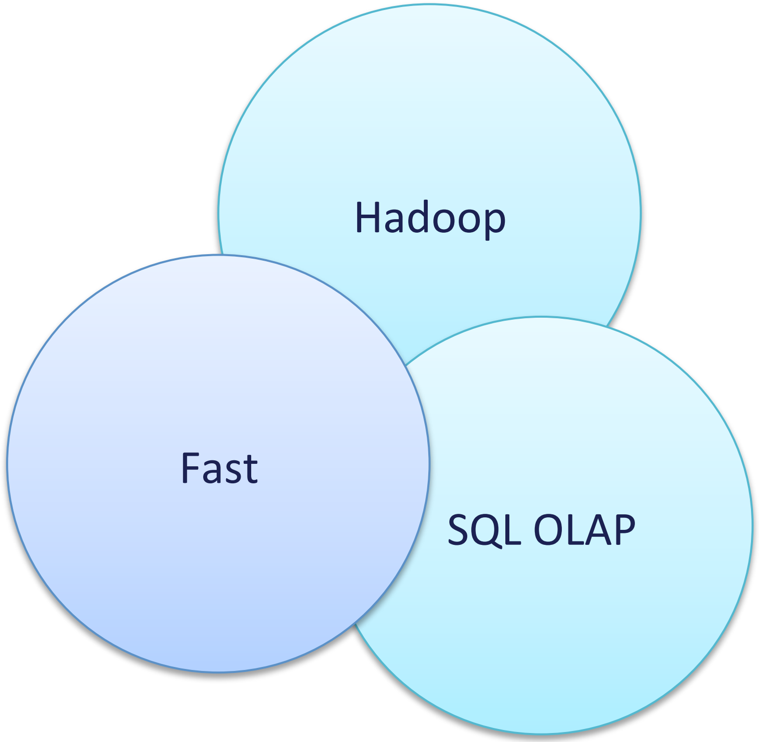
FAST AND BIG IN COMBINATION

- Fast Profile
  - In memory: user segmentation - GB to TB (300M+ rows)
  - 10k to 1M+ requests/sec
  - 99 percentile latency under 5ms. (5x9's under 50ms)
  - VoltDB export to Vertica

- Big Profile
  - TB to PB of historical data
  - Columnar analytics for fast reporting.
  - Real time ingest of historical data (possibly via VoltDB)
  - Vertica UDX to VoltDB

TYPICAL FAST QUESTIONS

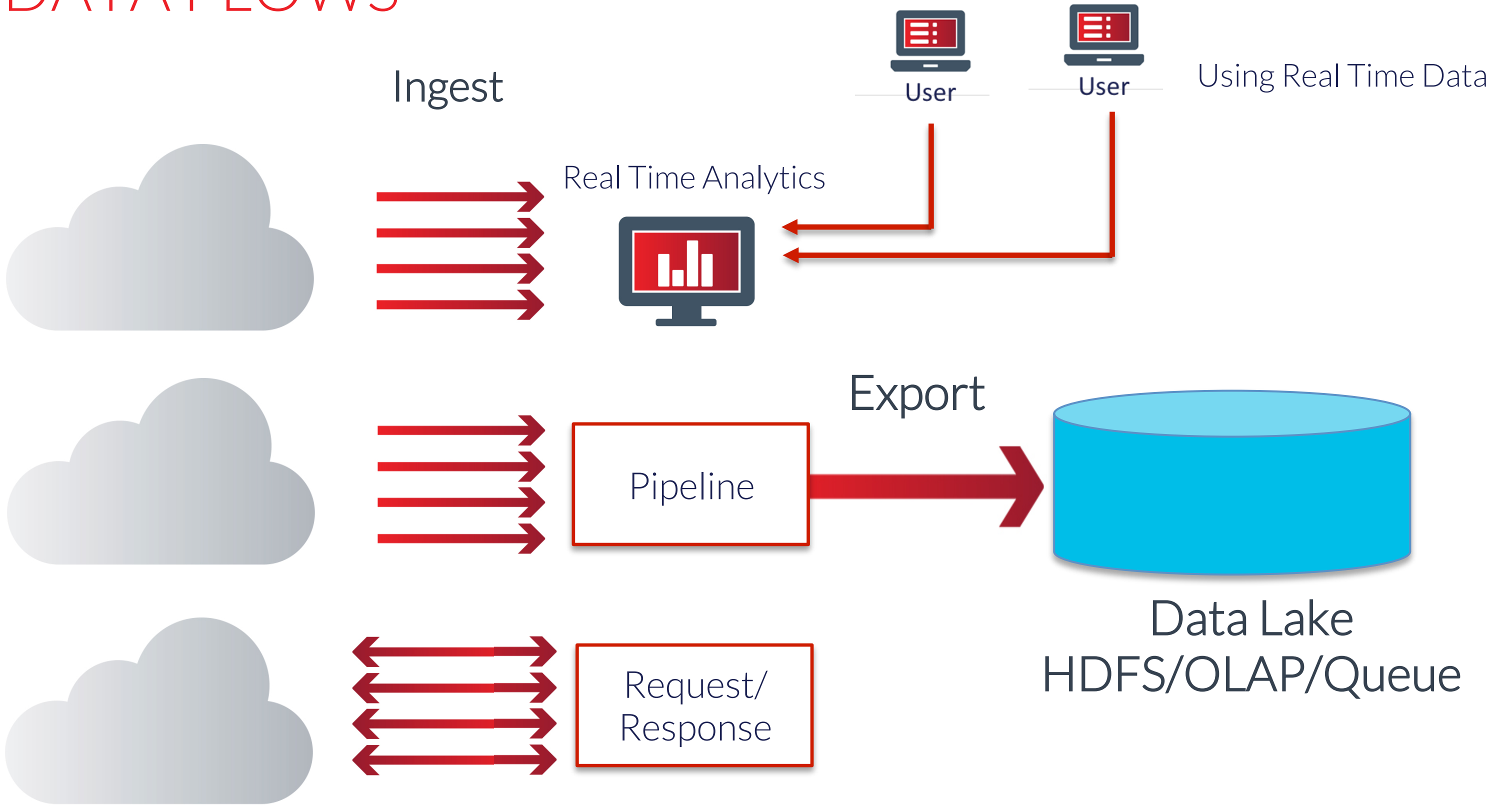
- **Is the fast layer streaming?**
  - It is often more like OLTP
- **How do the pieces communicate?**
  - OLAP analytics from Big -> Fast
  - New events from Fast -> Big
- **Where do “analytics” belong?**
  - Analytics with decisions: with Fast
  - Analytics against history: with Big
- **Are streaming frameworks equivalent?**
  - Traditional SQL CEP (Esper)
  - Tuple DAGs (Storm)
  - Window processors on Hadoop (Spark)



THREE FAST DATA APPLICATION PATTERNS

- Real-Time Analytics
  - Real-time analytics for operations
  - Real-time KPI measurement
  - Real-time analytics for apps
- Data Pipelines
  - Streaming data enrichment
  - Sessionization / re-assembly
  - Correlation (by time, by location, by id)
  - Filtering
  - Pre-aggregation
- Fast Request/Response
  - Mobile Authorization
  - Campaign Authorization
  - Fast API Quota Enforcement
  - Micro-Personalization
  - Recommendation Serving

DATA FLOWS


Ingest
User
User
Using Real Time Data
Real Time Analytics
Export
Pipeline
Data Lake
HDFS/OLAP/Queue
Request/
Response

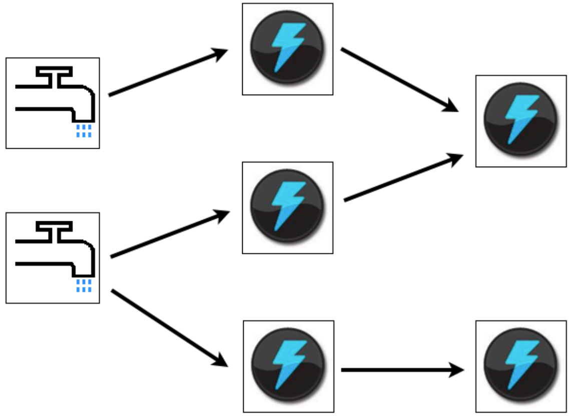
THE INPUT FEED IS ONLY A SMALL PART OF FAST DATA

Data	Temporality	Examples
Input Feed	Event Stream	Click stream, tick stream, sensors, metrics
Real-Time Analytic Results	Persistent/ Queryable	Counters, streaming aggregates, Time-series rollups
Event metadata	Persistent (Look-Ups)	Device version, location, user profiles, point of interest data
OLAP Analytics Used in Real-Time Decisions	Persistent (Look-Ups)	Scoring models, seasonal usage, demographic trends
Responses	Event Stream	Policy enforcement decisions, Personalization recommendations
Pipeline Output	Event Stream	Enriched, filtered, correlated transform of input feed

THREE REQUIREMENTS CREATE STATE

1. RT analytics outputs must be queryable
2. Metadata, dimension data, “lookup tables” to create groupings for analytics and to supply enrichment data
3. Grouping, filtering and aggregating generate intermediate state – open sessions, partially assembled logical events

STORM: A COMMON ALTERNATIVE



- Spouts and Bolts
- Streaming computation
- Run snippets of java against each event
- Connect queues to backends with intermediate code

But…

1. Need "lookup" database for dimension data.
2. Need a "serving" database for analytic results
3. Need additional management clusters (ZooKeeper)
4. No ad-hoc queries.
5. Lots of custom code (rarely declarative).

STREAMING OPERATORS NEED STATE

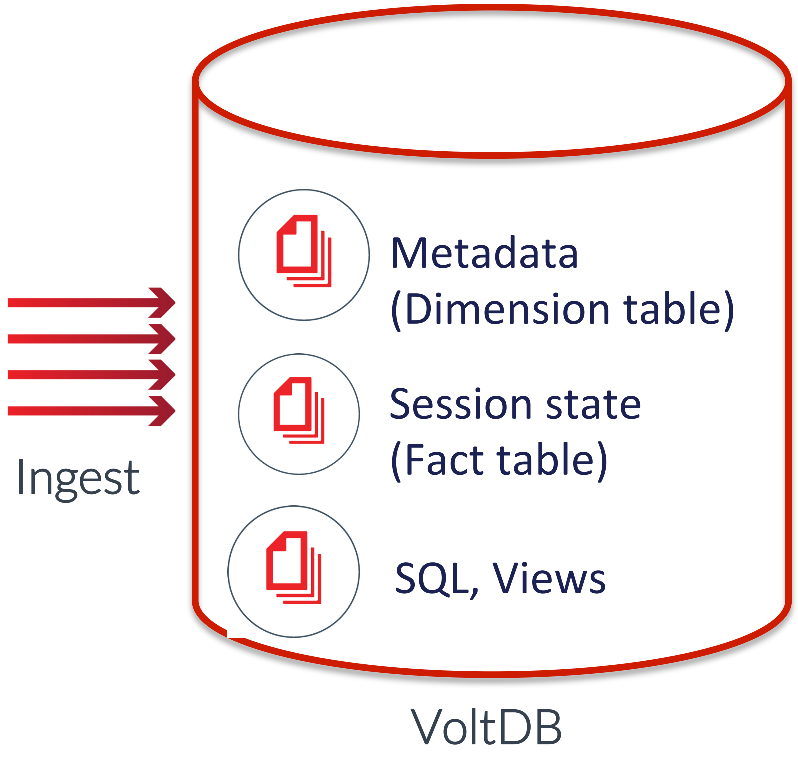
Require State

- Filter
- Join
- Aggregate
- Group By

Stateless

- Partition

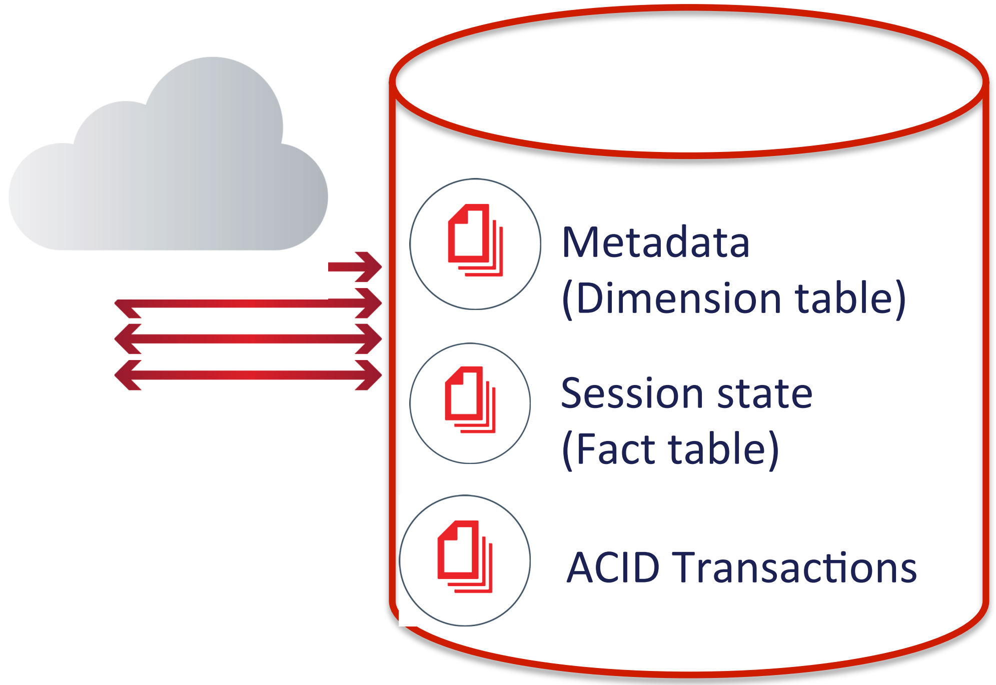
VOLTDB: REAL-TIME ANALYTICS



- In-memory MPP SQL over ODBC/JDBC
- Cheap + correct materialized views for streaming aggregations

- Operational analytics and monitoring
- RT analytics enabling user-facing applications
- KPI for internal BI/Dashboards

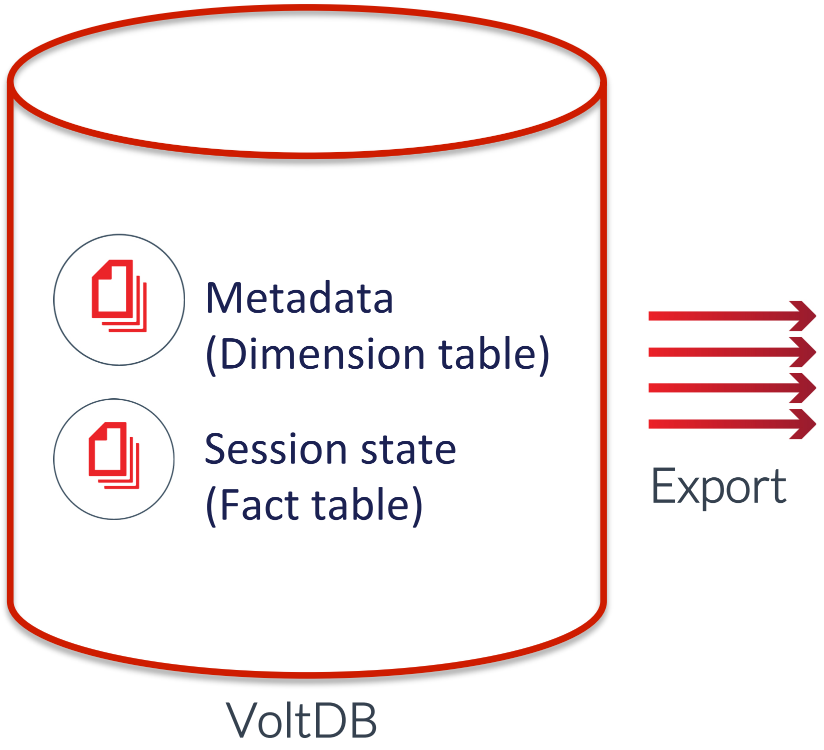
VOLTDB: REQUEST/RESPONSE DECISIONS



- Fully ACID transaction model.
- Thousands to Millions per second
- At less than 5ms latencies

- Authorization
  - RT balance checks, quota enforcement
- Personalization and Recommendation Serving
  - Combine pre-score with immediate context

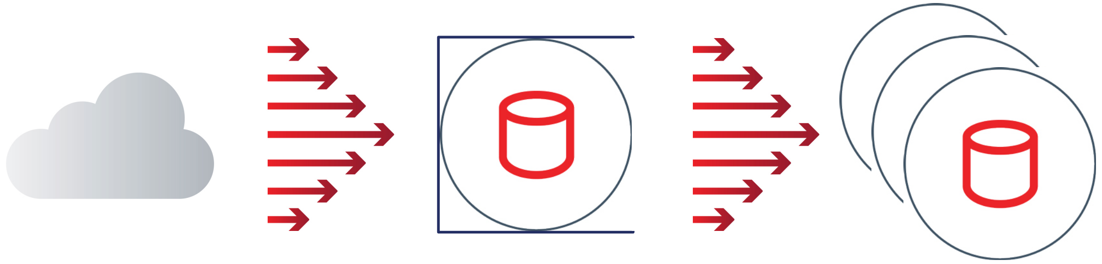
VOLTDB: DATA PIPELINES WITH EXPORT



- MPP streaming Export
- Row data, Thrift messages, CSV
- OLAP, HDFS and message queues

- Filtering (ex: only RFID / iBeacon readings that show change from previous location).
- Sessionization
- Common version re-writing
- Data enrichment

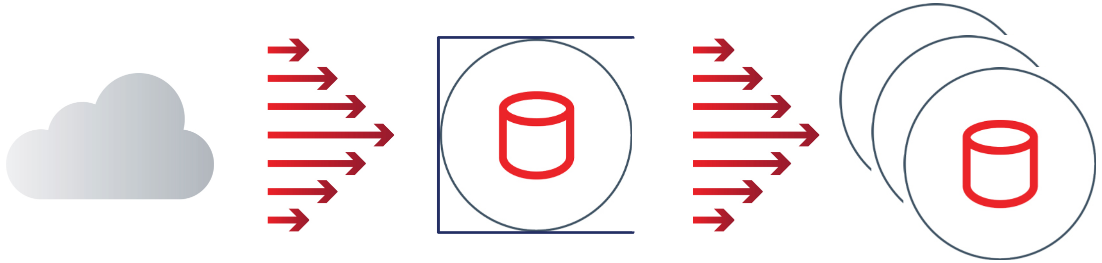
PIPELINE DEPLOYED: VOLTDB…



Manages game state for online poker and archives completed games to **Hadoop**.

Ingests smart meter readings from concentrators, supports **real time applications** and buffers data for end of day **billing mediation systems**.

PIPELINE DEPLOYED: VOLTDB…



Ingests RFID readings, supports real time applications that **push social media updates** based on VoltDB leaderboards.

Processes **clickstream logs and exports correlated USERID** records for use at CDN endpoints for advertising targeting

Processes **SKU catalogs** from suppliers to produce correlated catalog that is exported to indexing and post-processing for an **online retailer**.

VOLTDB EXPORT ANSWERS THE QUESTIONS:

How do I stream filtered, enriched, updated results to OLAP/HDFS systems?

How do I send alerts, alarms, SMS, or messages to downstream applications?

VOLTDB EXPORT UI

ddl.sql

```
CREATE TABLE events (
   EventID INTEGER,
   time    TIMESTAMP,
   msg     VARCHAR(128));
EXPORT TABLE events;
```

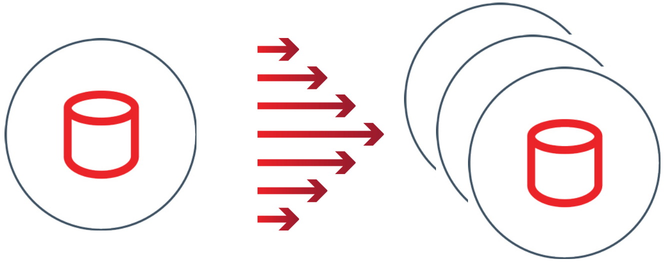
Application SQL

```
INSERT into TABLE values…
```

deployment.xml

```
<export enabled="true"
target="file">
```

EXPORTING TO HDFS



```
<export enabled="true" target="http">
  <configuration>
    <property name="endpoint">
      http://hadoopserver/webhdfs/v1.0/%t/%p.%t.%g.csv
    </property>
  </configuration>
</export>
```

EXPORT FORMATS

- CSV
- TSV
- Avro container
- Raw data

EXTENSIBLE API

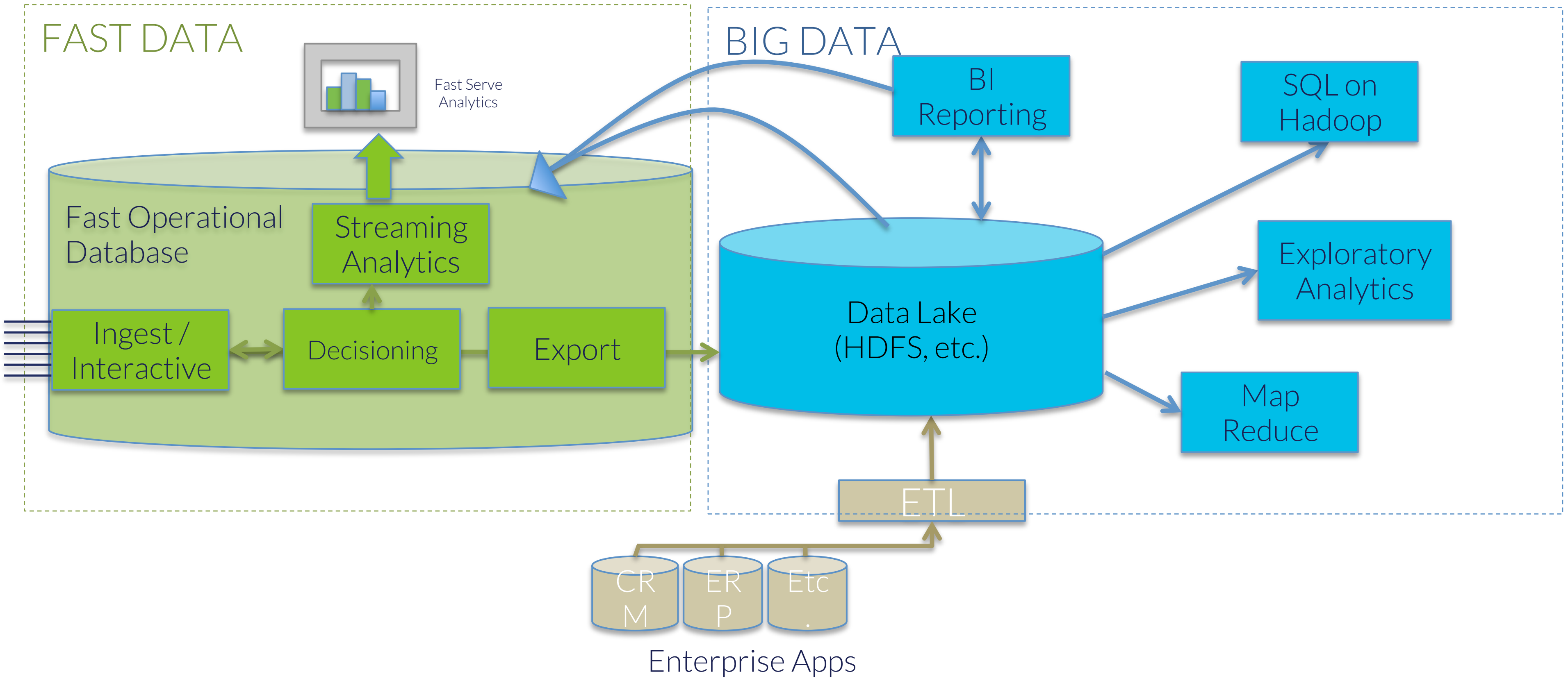
All of these export connectors are hosted plugins to the VoltDB database. VoltDB manages HA, fault tolerance, configuration, and MPP scale-out.

```
public void onBlockStart() throws RestartBlockException;{
}

public boolean processRow(int rowSize, byte[] rowData) throws
    RestartBlockException {
}

public void onBlockCompletion() throws RestartBlockException {
}
```

DATA ARCHITECTURE FOR FAST + BIG DATA



STREAMING APPS ARE REALLY DATABASE APPS WHEN YOU USE A DATABASE THAT'S FAST ENOUGH.

Try VoltDB

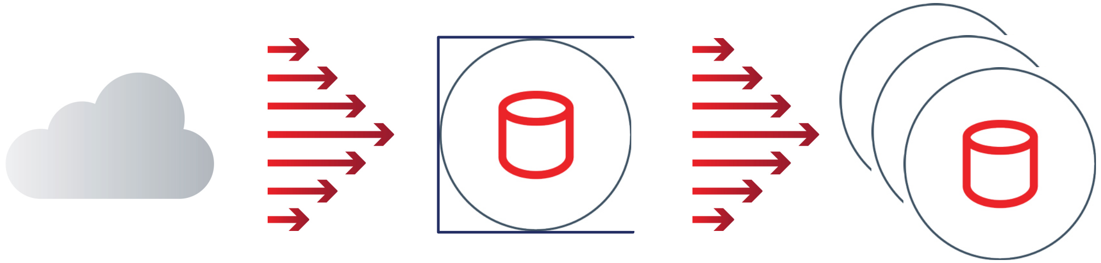
voltdb.com

SMART DATA FAST.™ VOLTDB

THANK YOU!

VOLTDB:

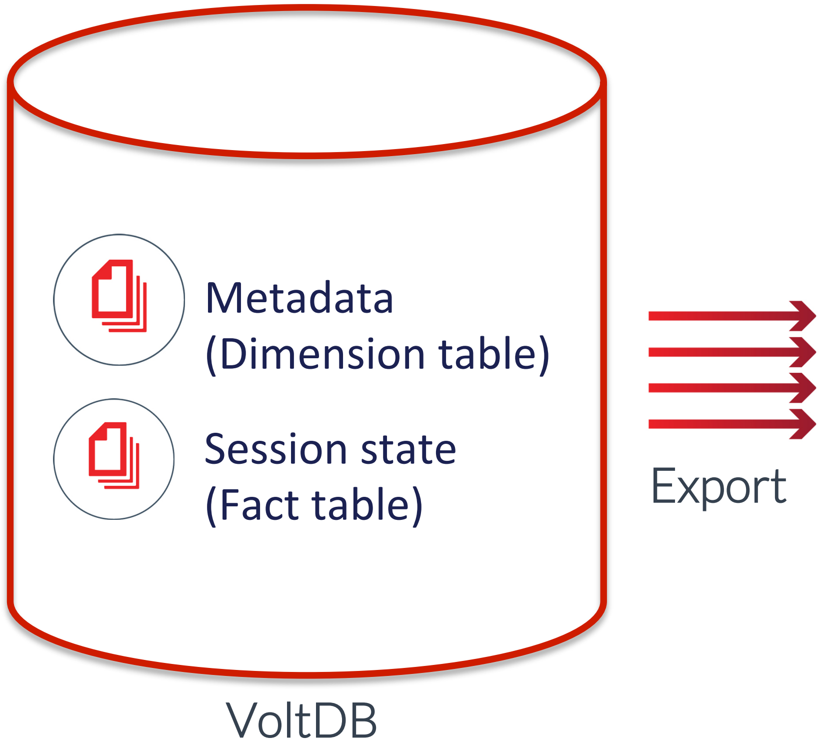
- We say “Ingest &Export” vs. Spout and Bolt
- Scale (**ACID**) snippets of Java for each incoming event.



AND…

- **Actually serve real time analytics via SQL**
- **Metadata for lookup/enrichment implicit in DB function**
- **Integrate with OLAP systems to use OLAP reports with event-based processing**
- **Generate fast transactional responses**
- **Support ad-hoc queryability**
- **Declarative aggregations vs. code**
- **Fast: no need to micro-batch**

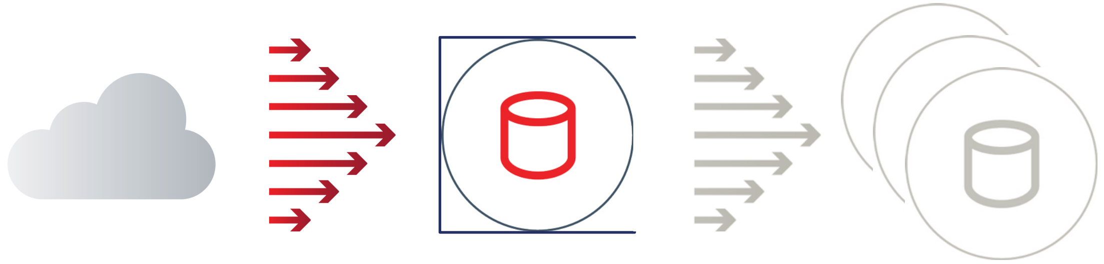
VOLTDB: DATA PIPELINES WITH EXPORT



- MPP streaming Export
- Row data, Thrift messages, CSV
- OLAP, HDFS and message queues

- Filtering (ex: only RFID / iBeacon readings that show change from previous location).
- Sessionization
- Common version re-writing
- Data enrichment

INTEGRATING DATA SOURCES WITH VOLTDB



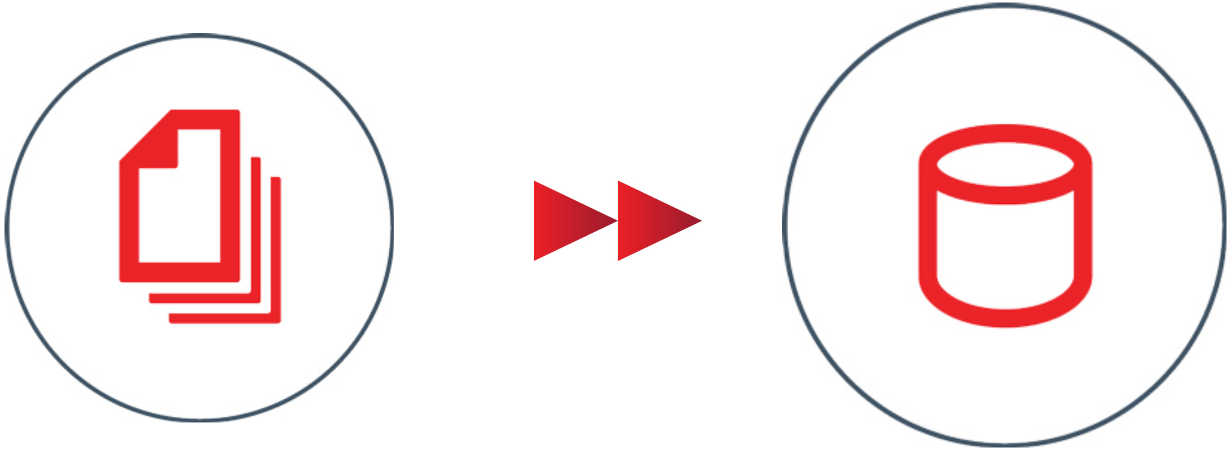
BULK LOADERS

- CSV loader
- Kafka loader
- JDBC loader
- Vertica UDx
- Extensible loader API

APPLICATION INTERFACES

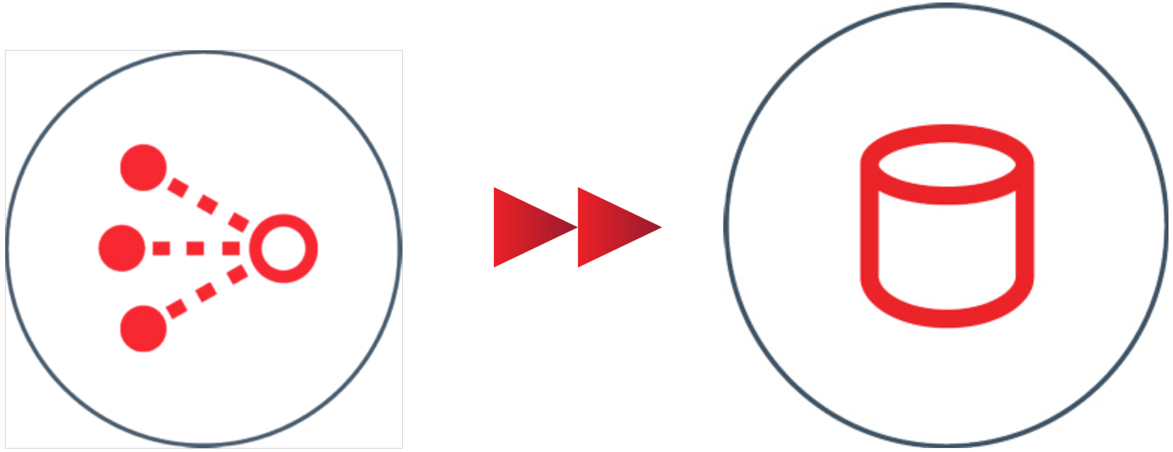
- JDBC
- ODBC
- HTTP JSON
- Native client drivers / SDKs

INTEGRATING WITH CSV DATA



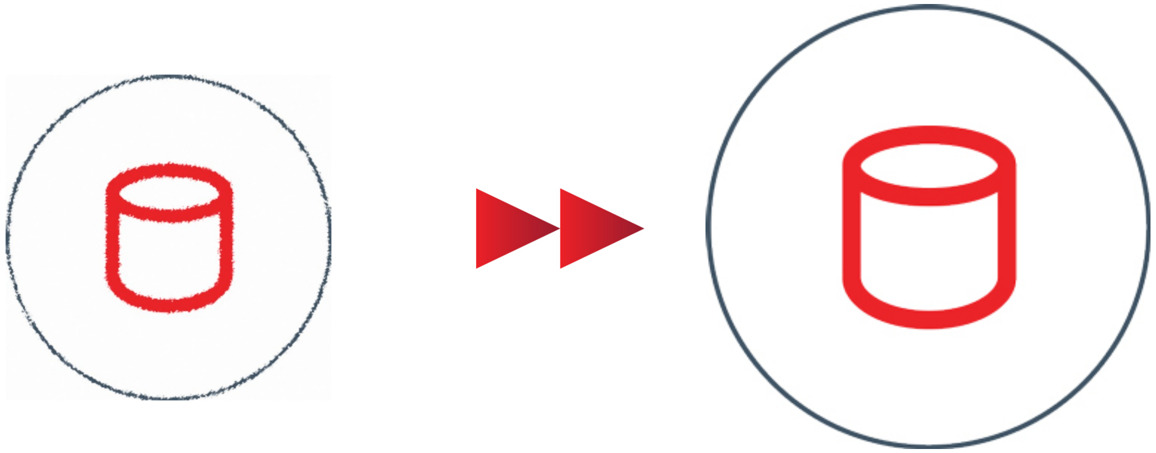
```
csvloader volttable –f data.csv
```

INTEGRATING WITH KAFKA



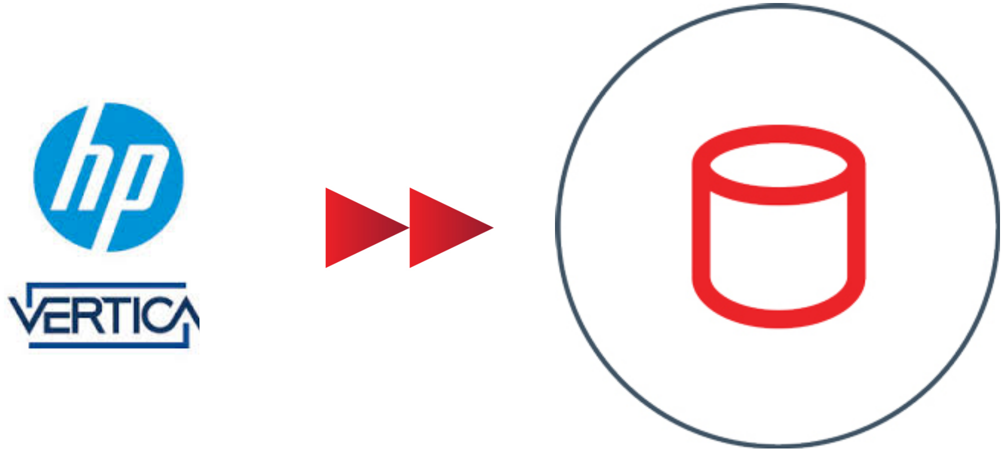
```
kafkaloader volttable \
  --zookeeper=zkserver:2181 \
  --topic=topicname
```

INTEGRATING WITH JDBC SOURCES



```
jdbcloader volttable \
  --jdbcurl=jdbc:postgresql://server/db \
  --jdbcdriver=org.postgresql.Driver \
  --jdbctable=table
```

INTEGRATING WITH HP VERTICA UDX



```
SELECT voltdbload(c1, c2, c3
       USING PARAMETERS voltservers='localhost’,
                        volttable=‘volttable’)
FROM T;
```

EXTENSIBLE VOLTBULKLOADER: BULK SMASH

All of these tools are built on a MIT licensed extensible API that provides performance optimizations, batching, load balancing.

```
VoltBulkLoader loader =
    client.getNewBulkLoader(tableName, maxBatchSize,
    failureCallback);
for (…) {
     loader.insertRow(handle, values);
}
loader.drain();
loader.close();
```

NATIVE CLIENT LIBRARIES

- Java
- C++
- PHP

- Node.js
- Go
- Python
- Erlang
- Ruby

Or, just...

```
curl ‘http://localhost:8080/api/1.0/?\
   Procedure=Vote&Parameters=[1,1,0]’
```

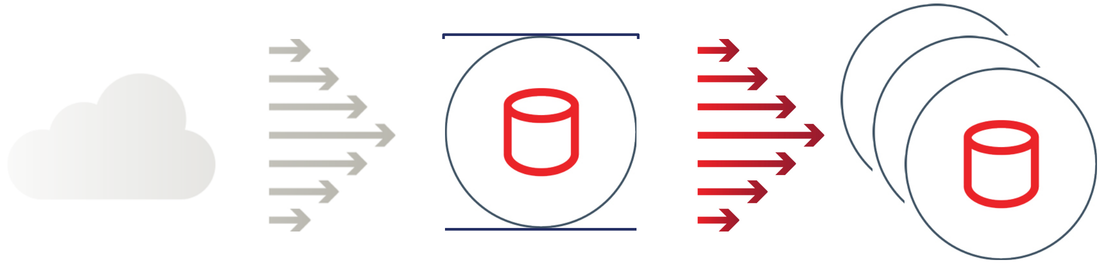
VOLTDB EXPORT TOPOLOGIES

- VoltDB -> Queue (Kafka, RabbitMQ)
- VoltDB -> HDFS (for Pig/Hive/etc. processing)
- VoltDB -> OLAP (Vertica, Netezza..)
- VoltDB -> HTTP Endpoint, i.e: ElasticSearch

VOLTDB EXPORT PROGRAMMING CONTRACT

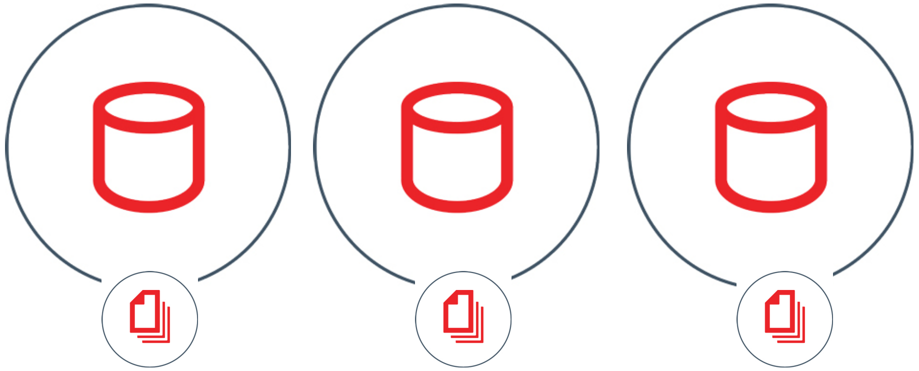
- Export data is durable until exported
- MPP scale-out of export data flows
- At-least-once delivery during HA events
- Built-in row ids (for uniqueness filtering)
- Extensible API for open source connectors

INTEGRATING VOLTDB WITH EXPORT TARGETS



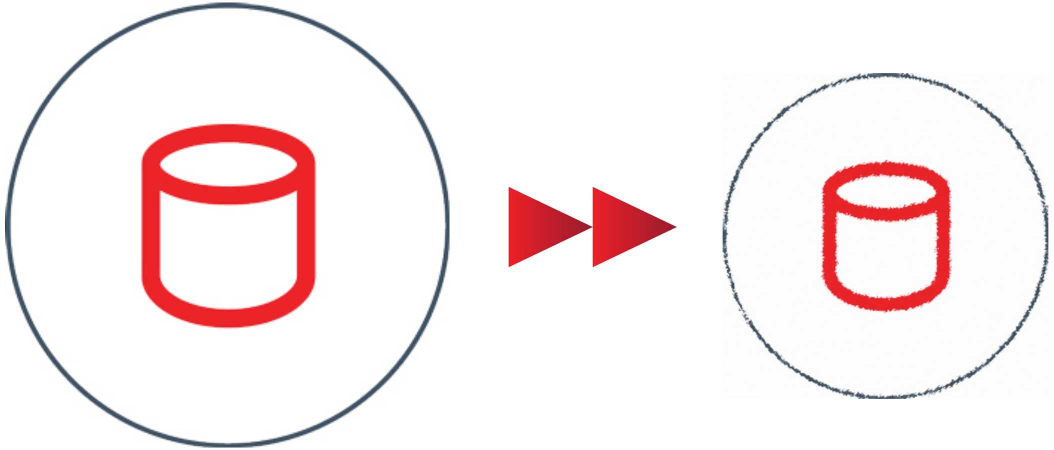
- Local file system export
- JDBC export
- Kafka export
- RabbitMQ export
- HDFS export
- HTTP export
- Extensible API

EXPORTING TO LOCAL FILE SYSTEM



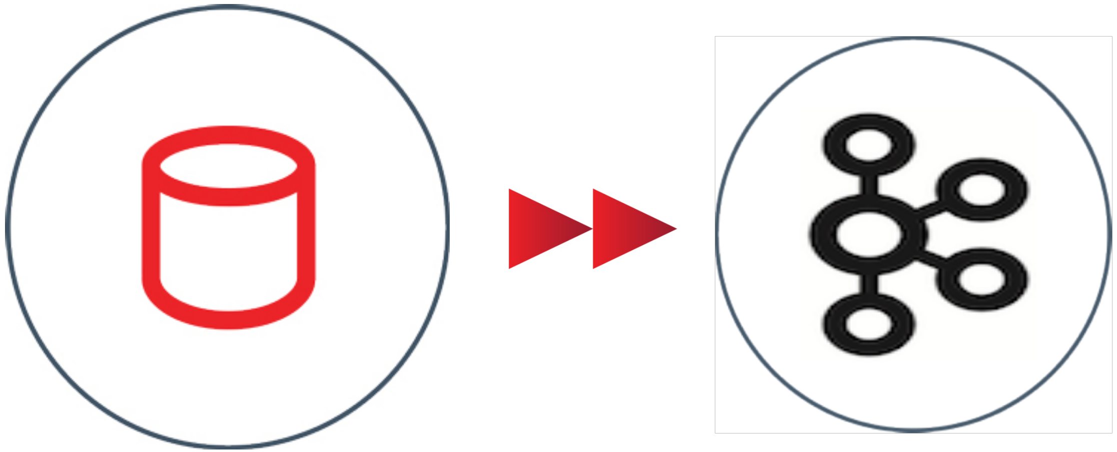
```
<export enabled="true" target="file">
  <configuration>
    <property name="type">csv</property>
    <property name="nonce">MyExport</property>
  </configuration>
</export>
```

EXPORTING TO JDBC DESTINATIONS



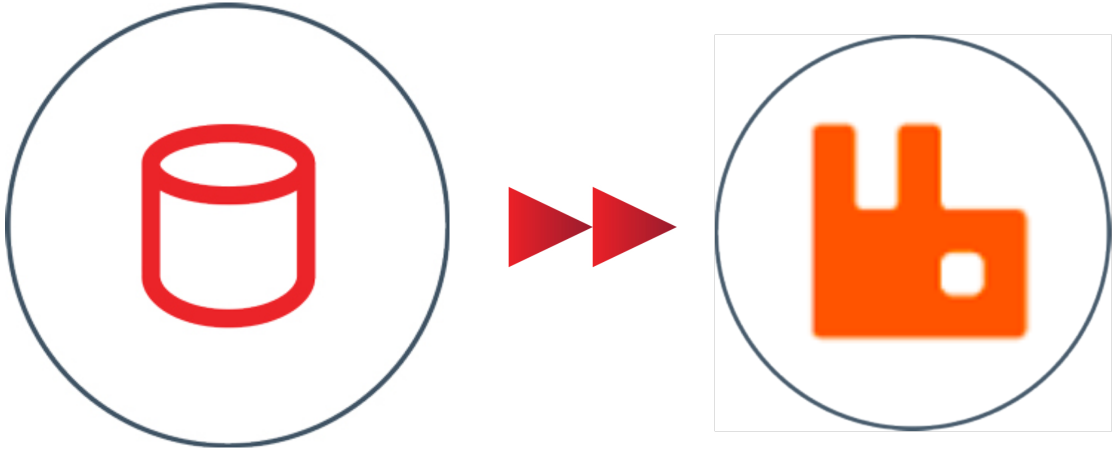
```
<export enabled="true" target=”jdbc">
  <configuration>
    <property name=”jdbcurl">jdbc:postgresql://server/db</
property>
    <property name=”jdbcuser">guest</property>
  </configuration>
</export>
```

EXPORTING TO KAFKA



```
<export enabled="true" target=”kafka">
  <configuration>
    <property name=“metadata.broker.list”>server1</property>
  </configuration>
</export>
```

EXPORTING TO RABBITMQ



```
<export enabled="true" target="rabbitmq">
  <configuration>
    <property name="broker.host”>server1</property>
  </configuration>
</export>
```