

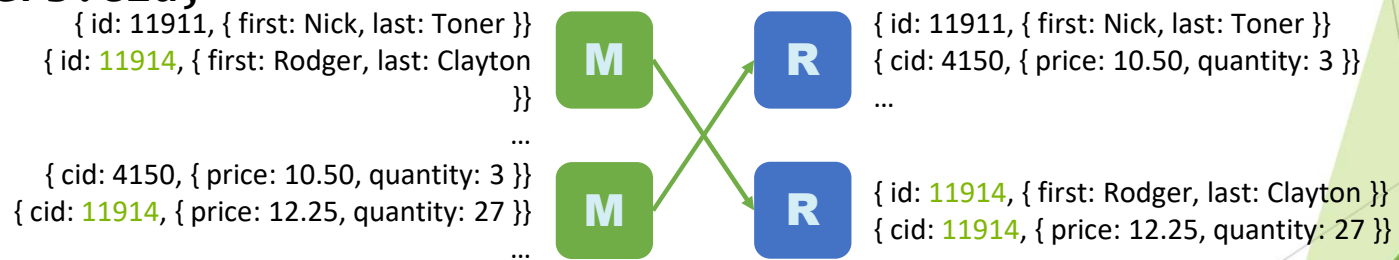
Hive Join Strategies

Type	Approach	Pros	Cons
Shuffle Join	Join keys are shuffled using MapReduce, and joins are performed on the reduce side.	Works regardless of data size or layout.	Most resource-intensive and slowest join type.
Map (Broadcast) Join	Small tables are loaded into memory in all nodes, mapper scans through the large table, and joins.	Very fast, single scan through largest table.	All but one table must be small enough to fit in RAM.
Sort-Merge-Bucket Join	Mappers take advantage of co-location of keys to do efficient joins.	Very fast for tables of any size.	Data must be sorted and bucketed ahead of time.

Shuffle Joins

customer				orders		
first	last	id		cid	price	quantity
Nick	Toner	11911		4150	10.50	3
Jessie	Simonds	11912		11914	12.25	27
Kasi	Lamers	11913		3491	5.99	5
Rodger	Clayton	11914		2934	39.99	22
Verona	Hollen	11915		11914	40.50	10

SELECT * FROM customer JOIN orders ON customer.id =
orders.cid;



Map (Broadcast) Joins

customer			orders		
first	last	id	cid	price	quantity
Nick	Toner	11911	4150	10.50	3
Jessie	Simonds	11912	11914	12.25	27
Kasi	Lamers	11913	3491	5.99	5
Rodger	Clayton	11914	2934	39.99	22
Verona	Hollen	11915	11914	40.50	10

```
SELECT * FROM customer JOIN orders ON customer.id =  
orders.cid;
```


{ id: 11914, { first: Rodger, last: Clayton } }
{ cid: 11914, { price: 12.25, quantity: 27 } },
cid: 11914, { price: 12.25, quantity: 27 } }



Records are joined during
the map phase.

Sort-Merge-Bucket Joins

customer			orders		
first	last	id	cid	price	quantity
Nick	Toner	11911	4150	10.50	3
Jessie	Simonds	11912	11914	12.25	27
Kasi	Lamers	11913	11914	40.50	10
Rodger	Clayton	11914	12337	39.99	22
Verona	Hollen	11915	15912	40.50	10



```
SELECT * FROM customer join orders ON customer.id =  
orders.cid;
```

**Distribute and sort by the most common
join key.**

```
CREATE TABLE orders (cid int, price float, quantity int)  
CLUSTERED BY(cid) SORTED BY(cid) INTO 32 BUCKETS;
```

```
CREATE TABLE customer (id int, first string, last string)  
CLUSTERED BY(id) SORTED BY(id) INTO 32 BUCKETS;
```

Invoking a Hive UDF

```
ADD JAR /myapp/lib/myhiveudfs.jar;  
CREATE TEMPORARY FUNCTION  
ComputeShipping  
    AS 'hiveudfs.ComputeShipping';  
  
FROM orders SELECT  
    address,  
    description,  
    ComputeShipping(zip, weight);
```

Lab: Writing a Hive UDF

Computing ngrams in Hive

```
select ngrams(sentences(val), 2, 100)
from mytable;
```

```
select context_ngrams(sentences(val),
    array("error", "code", null),
    100)
from mytable;
```

Demo: Computing ngrams

Lesson Review

1. A Hive table consists of a schema stored in the Hive _____ and data stored in _____.
2. **True or False:** The Hive metastore requires an underlying SQL database.
3. What happens to the underlying data of a Hive-managed table when the table is dropped?
4. **True or False:** A Hive external table must define a **LOCATION**.
5. List three different ways data can be loaded into a Hive table.
6. When would you use a skewed table?
7. What will the folder structure in HDFS look like for the **movies** table?

```
create table movies (title string, rating string, length double) partitioned by  
(genre string);
```

8. Explain the output of the following query:
select * from movies order by title;

9. What does the following Hive query compute?

```
from mytable select  
explode(ngrams(sentences(val),3,100)) as myresult;
```

10. What does the following Hive query compute?

```
Talentum Group Technology  
from mytable select explode(  
context_ngrams(sentences(val),  
array("I","liked",null),10)) as myresult;
```

Lab: Joining Datasets in Hive

Lab: Computing ngrams of Emails in Avro Format