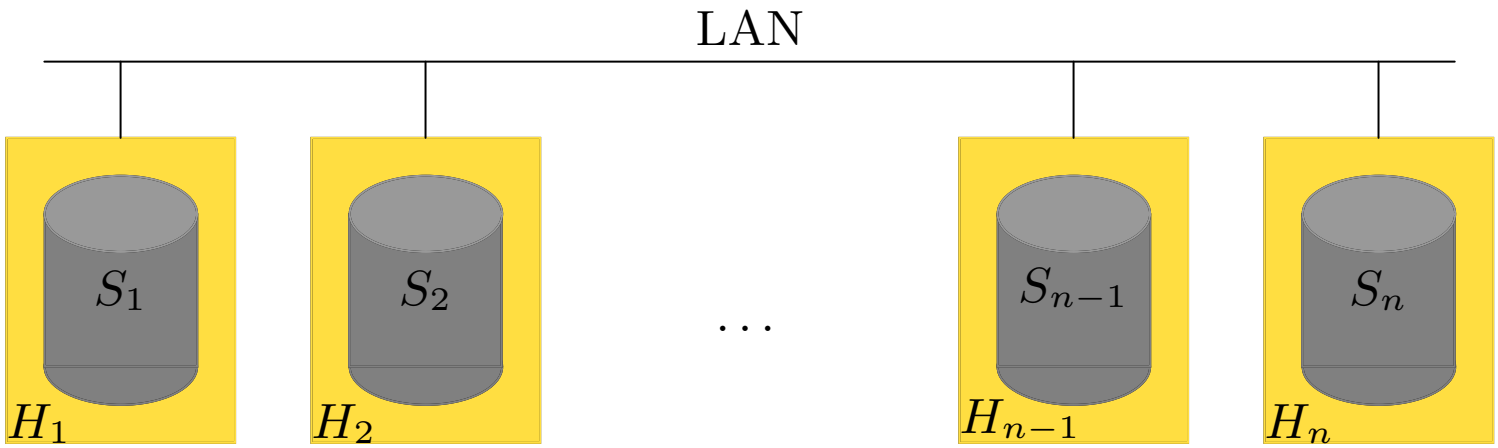
Big Data: Pig Latin

P.J. McBrien

Imperial College London

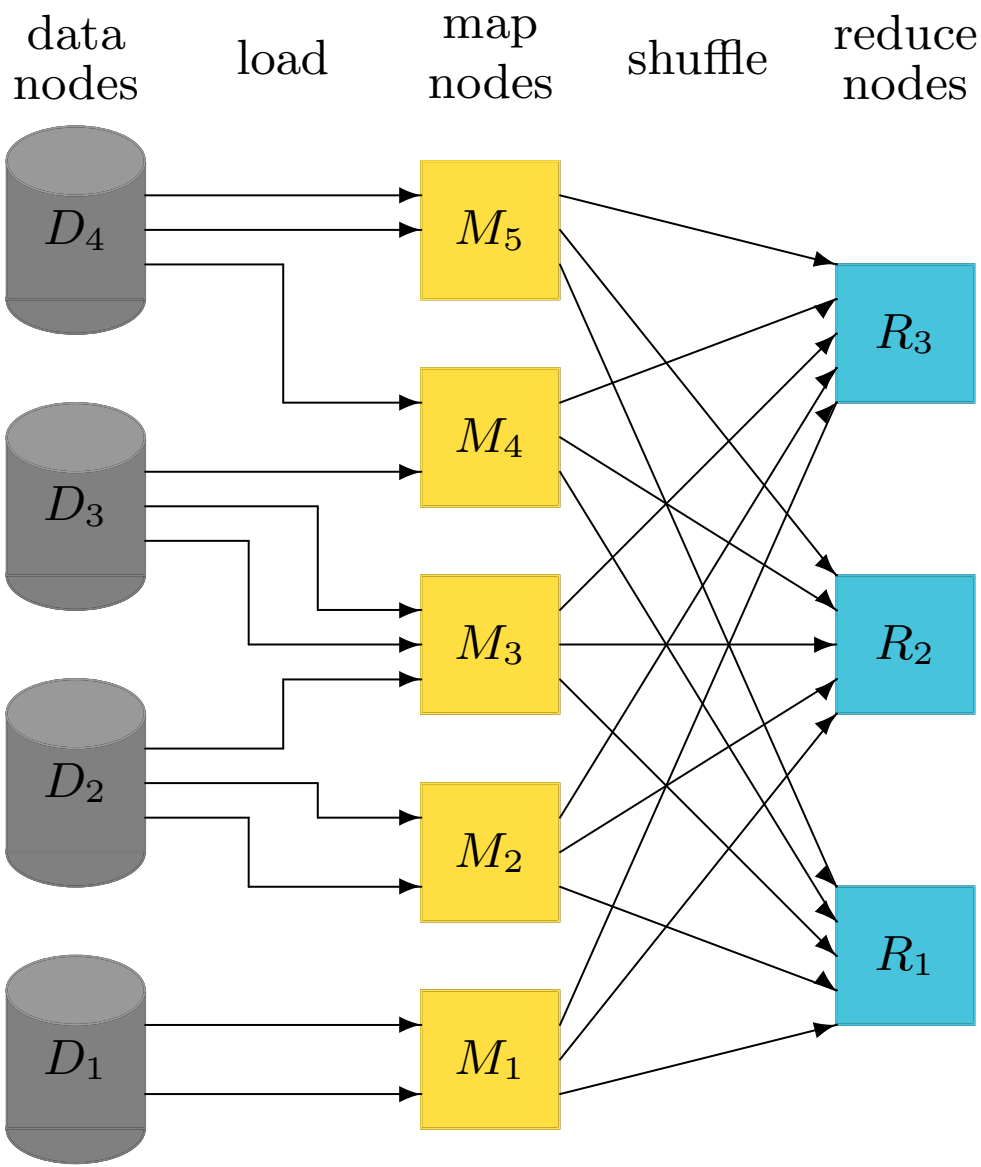
What is Big Data?



a **big data** system is able to handle:

- Tera-bytes of data
- data spread over hundreds or thousands of servers
- failures of nodes without loss of data

MapReduce


data nodes
load
map nodes
shuffle
reduce nodes
D_4
D_3
D_2
D_1
M_5
M_4
M_3
M_2
M_1
R_3
R_2
R_1

MapReduce: Map Phase of Word Count

M_1

The First Lord of the Admiralty in his speech the other night went even farther. He said, 'We are always reviewing the position'. Everything, he assured us is entirely fluid. I am sure that that is true. Anyone can see what the position is. The Government

(the,1) (first,1) (lord,1) (of,1) (the,1) (admiralty,1) (in,1) (his,1) (speech,1) (the,1) (other,1) (night,1) (went,1) (even,1) (farther,1) (he,1) (said,1) (we,1) (are,1) (always,1) (reviewing,1) (the,1) (position,1) (everything,1) (he,1) (assured,1) (us,1) (is,1) (entirely,1) (fluid,1) (i,1) (am,1) (sure,1) (that,1) (that,1) (is,1) (true,1) (anyone,1) (can,1) (see,1) (what,1) (the,1) (position,1) (is,1) (the,1) (government,1)

M_2

simply cannot make up their minds, or they cannot get the Prime Minister to make up his mind. So they go on in strange paradox, decided only to be undecided, resolved to be irresolute, adamant for drift, solid for fluidity, all-powerful to be impotent.

(simply,1) (cannot,1) (make,1) (up,1) (their,1) (minds,1) (or,1) (they,1) (cannot,1) (get,1) (the,1) (prime,1) (minister,1) (to,1) (make,1) (up,1) (his,1) (mind,1) (so,1) (they,1) (go,1) (on,1) (in,1) (strange,1) (paradox,1) (decided,1) (only,1) (to,1) (be,1) (undecided,1) (resolved,1) (to,1) (be,1) (irresolute,1) (adamant,1) (for,1) (drift,1) (solid,1) (for,1) (fluidity,1) (all-powerful,1) (to,1) (be,1) (impotent,1)

MapReduce: Shuffle Phase of Word Count

M_1
(the,1) (first,1) (lord,1) (of,1)
(the,1) (admiralty,1) (in,1) (his,1)
(speech,1) (the,1) (other,1)
(night,1) (went,1) (even,1)
(farther,1) (he,1) (said,1) (we,1)
(are,1) (always,1) (reviewing,1)
(the,1) (position,1) (everything,1)
(he,1) (assured,1) (us,1) (is,1)
(entirely,1) (fluid,1) (i,1) (am,1)
(sure,1) (that,1) (that,1) (is,1)
(true,1) (anyone,1) (can,1) (see,1)
(what,1) (the,1) (position,1) (is,1)
(the,1) (government,1)

M_2
(simply,1) (cannot,1) (make,1)
(up,1) (their,1) (minds,1) (or,1)
(they,1) (cannot,1) (get,1) (the,1)
(prime,1) (minister,1) (to,1)
(make,1) (up,1) (his,1) (mind,1)
(so,1) (they,1) (go,1) (on,1) (in,1)
(strange,1) (paradox,1) (decided,1)
(only,1) (to,1) (be,1) (undecided,1)
(resolved,1) (to,1) (be,1)
(irresolute,1) (adamant,1) (for,1)
(drift,1) (solid,1) (for,1) (fluidity,1)
(all-powerful,1) (to,1) (be,1)
(impotent,1)

R_1
(first,1) (admiralty,1) (in,1) (his,1)
(even,1) (farther,1) (he,1) (are,1)
(always,1) (everything,1) (he,1)
(assured,1) (is,1) (entirely,1)
(fluid,1) (i,1) (am,1) (is,1)
(anyone,1) (can,1) (is,1)
(government,1) (cannot,1)
(cannot,1) (get,1) (his,1) (go,1)
(in,1) (decided,1) (be,1) (be,1)
(irresolute,1) (adamant,1) (for,1)
(drift,1) (for,1) (fluidity,1)
(all-powerful,1) (be,1) (impotent,1)

R_2
(the,1) (lord,1) (of,1) (the,1)
(speech,1) (the,1) (other,1)
(night,1) (went,1) (said,1) (we,1)
(reviewing,1) (the,1) (position,1)
(us,1) (sure,1) (that,1) (that,1)
(true,1) (see,1) (what,1) (the,1)
(position,1) (the,1) (simply,1)
(make,1) (up,1) (their,1) (minds,1)
(or,1) (they,1) (the,1) (prime,1)
(minister,1) (to,1) (make,1) (up,1)
(mind,1) (so,1) (they,1) (on,1)
(strange,1) (paradox,1) (only,1)
(to,1) (undecided,1) (resolved,1)
(to,1) (solid,1) (to,1)

MapReduce: Reduce Phase of Word Count

R_1	(first,1) (admiralty,1) (in,1) (his,1) (even,1) (farther,1) (he,1) (are,1) (always,1) (everything,1) (he,1) (assured,1) (is,1) (entirely,1) (fluid,1) (i,1) (am,1) (is,1) (anyone,1) (can,1) (is,1) (government,1) (cannot,1) (cannot,1) (get,1) (his,1) (go,1) (in,1) (decided,1) (be,1) (be,1) (irresolute,1) (adamant,1) (for,1) (drift,1) (for,1) (fluidity,1) (all-powerful,1) (be,1) (impotent,1)	⇨	(adamant,1) (admiralty,1) (all-powerful,1) (always,1) (am,1) (anyone,1) (are,1) (assured,1) (be,3) (can,1) (cannot,2) (decided,1) (drift,1) (entirely,1) (even,1) (everything,1) (farther,1) (first,1) (fluid,1) (fluidity,1) (for,2) (get,1) (go,1) (government,1) (he,2) (his,2) (i,1) (impotent,1) (in,2) (irresolute,1) (is,3)
R_2	(the,1) (lord,1) (of,1) (the,1) (speech,1) (the,1) (other,1) (night,1) (went,1) (said,1) (we,1) (reviewing,1) (the,1) (position,1) (us,1) (sure,1) (that,1) (that,1) (true,1) (see,1) (what,1) (the,1) (position,1) (the,1) (simply,1) (make,1) (up,1) (their,1) (minds,1) (or,1) (they,1) (the,1) (prime,1) (minister,1) (to,1) (make,1) (up,1) (mind,1) (so,1) (they,1) (on,1) (strange,1) (paradox,1) (only,1) (to,1) (undecided,1) (resolved,1) (to,1) (solid,1) (to,1)	⇨	(lord,1) (make,2) (mind,1) (minds,1) (minister,1) (night,1) (of,1) (on,1) (only,1) (or,1) (other,1) (paradox,1) (position,2) (prime,1) (resolved,1) (reviewing,1) (said,1) (see,1) (simply,1) (so,1) (solid,1) (speech,1) (strange,1) (sure,1) (that,2) (the,7) (their,1) (they,2) (to,4) (true,1) (undecided,1) (up,2) (us,1) (we,1) (went,1) (what,1)

MapReduce: Combine Phase on Map Nodes

Combine

Often (and in particular for aggregate operators on grouped data), the Reduce process may be partially calculated on the Map nodes. Such a partial Reduce process is called a **Combine** operations.
For example $Sum(B) = Sum(B_1) + Sum(B_2)$ if $B = B_1 \cup B_2$ using bag semantics

Applying Combine to the WordCount problem

- Map phase identifies words from text
- Combine phase counts the number of times each word appears on each Map node
- Reduce phase sums per word the output of all Combine phases

MapReduce: Combine Phase of Word Count

M_1

(the,1) (first,1) (lord,1) (of,1)
(the,1) (admiralty,1) (in,1) (his,1)
(speech,1) (the,1) (other,1)
(night,1) (went,1) (even,1)
(farther,1) (he,1) (said,1) (we,1)
(are,1) (always,1) (reviewing,1)
(the,1) (position,1) (everything,1)
(he,1) (assured,1) (us,1) (is,1)
(entirely,1) (fluid,1) (i,1) (am,1)
(sure,1) (that,1) (that,1) (is,1)
(true,1) (anyone,1) (can,1) (see,1)
(what,1) (the,1) (position,1) (is,1)
(the,1) (government,1)

(i,1) (am,1) (he,2) (in,1) (is,3) (of,1)
(us,1) (we,1) (are,1) (can,1) (his,1)
(see,1) (the,6) (even,1) (lord,1)
(said,1) (sure,1) (that,2) (true,1)
(went,1) (what,1) (first,1) (fluid,1)
(night,1) (other,1) (always,1)
(anyone,1) (speech,1) (assured,1)
(farther,1) (entirely,1) (position,2)
(admiralty,1) (reviewing,1)
(everything,1) (government,1)

M_2

(simply,1) (cannot,1) (make,1)
(up,1) (their,1) (minds,1) (or,1)
(they,1) (cannot,1) (get,1) (the,1)
(prime,1) (minister,1) (to,1)
(make,1) (up,1) (his,1) (mind,1)
(so,1) (they,1) (go,1) (on,1) (in,1)
(strange,1) (paradox,1) (decided,1)
(only,1) (to,1) (be,1) (undecided,1)
(resolved,1) (to,1) (be,1)
(irresolute,1) (adamant,1) (for,1)
(drift,1) (solid,1) (for,1) (fluidity,1)
(all-powerful,1) (to,1) (be,1)
(impotent,1)

(be,3) (go,1) (in,1) (on,1) (or,1)
(so,1) (to,4) (up,2) (for,2) (get,1)
(his,1) (the,1) (make,2) (mind,1)
(only,1) (they,2) (drift,1) (minds,1)
(prime,1) (solid,1) (their,1) (cannot,2) (simply,1) (adamant,1)
(decided,1) (paradox,1) (strange,1)
(fluidity,1) (impotent,1) (minister,1) (resolved,1) (undecided,1)
(irresolute,1) (all-powerful,1)

MapReduce: Reduce Phase of Word Count after Combine

R_1	(admiralty,1) (always,1) (am,1) (anyone,1) (are,1) (assured,1) (can,1) (entirely,1) (even,1) (everything,1) (farther,1) (first,1) (fluid,1) (government,1) (he,2) (his,1) (i,1) (in,1) (is,3) (adamant,1) (all-powerful,1) (be,3) (cannot,2) (decided,1) (drift,1) (fluidity,1) (for,2) (get,1) (go,1) (his,1) (impotent,1) (in,1) (irresolute,1)	⇨	(adamant,1) (admiralty,1) (all-powerful,1) (always,1) (am,1) (anyone,1) (are,1) (assured,1) (be,3) (can,1) (cannot,2) (decided,1) (drift,1) (entirely,1) (even,1) (everything,1) (farther,1) (first,1) (fluid,1) (fluidity,1) (for,2) (get,1) (go,1) (government,1) (he,2) (his,2) (i,1) (impotent,1) (in,2) (irresolute,1) (is,3)
R_2	(lord,1) (night,1) (of,1) (other,1) (position,2) (reviewing,1) (said,1) (see,1) (speech,1) (sure,1) (that,2) (the,6) (true,1) (us,1) (we,1) (went,1) (what,1) (make,2) (mind,1) (minds,1) (minister,1) (on,1) (only,1) (or,1) (paradox,1) (prime,1) (resolved,1) (simply,1) (so,1) (solid,1) (strange,1) (the,1) (their,1) (they,2) (to,4) (undecided,1) (up,2)	⇨	(lord,1) (make,2) (mind,1) (minds,1) (minister,1) (night,1) (of,1) (on,1) (only,1) (or,1) (other,1) (paradox,1) (position,2) (prime,1) (resolved,1) (reviewing,1) (said,1) (see,1) (simply,1) (so,1) (solid,1) (speech,1) (strange,1) (sure,1) (that,2) (the,7) (their,1) (they,2) (to,4) (true,1) (undecided,1) (up,2) (us,1) (we,1) (went,1) (what,1)

MapReduce Implementations

- **Hadoop**: Java API to write MapReduce programmes
- **HBase**: Java API (implemented over Hadoop) to provide **BigTable**
- **Hive**: SQL based database system (implemented over Hadoop)
- **Pig Latin**: High Level scripting language to implement MapReduce as a sequence of primitive operators (implemented over Hadoop) in a dataflow

Pig: Accessing Data

LOAD

The LOAD operator makes available a data source as a relation.

Pig: Accessing Data

LOAD

The LOAD operator makes available a data source as a relation.

account.tsv

100[tab]current[tab]McBrien, P.[tab][tab]67
101[tab]deposit[tab]McBrien, P.[tab]5.25[tab]67
103[tab]current[tab]Boyd, M.[tab][tab]34
107[tab]current[tab]Poulovassilis, A.[tab][tab]56
119[tab]deposit[tab]Poulovassilis, A.[tab]5.50[tab]56
125[tab]current[tab]Bailey, J.[tab][tab]56

Pig: Accessing Data

LOAD

The LOAD operator makes available a data source as a relation.

account.tsv

100[tab]current[tab]McBrien, P.[tab][tab]67
101[tab]deposit[tab]McBrien, P.[tab]5.25[tab]67
103[tab]current[tab]Boyd, M.[tab][tab]34
107[tab]current[tab]Poulovassilis, A.[tab][tab]56
119[tab]deposit[tab]Poulovassilis, A.[tab]5.50[tab]56
125[tab]current[tab]Bailey, J.[tab][tab]56

Reading a TSV file

```
account =
    LOAD '/vol/automed/data/bank_branch/account.tsv'
    AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);
```

Running Pig Scripts

copy_account.pig

```
account =
    LOAD '/vol/automed/data/bank_branch/account.tsv'
    AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);

STORE account INTO 'account_copy' USING PigStorage(',');
```

Non-interactive

```
pig -x local copy_account.pig
```

Running Pig Scripts

copy_account.pig

```
account =
    LOAD '/vol/automed/data/bank_branch/account.tsv'
    AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);

STORE account INTO 'account_copy' USING PigStorage(',');
```

Interactive

```
pig -x local
grunt>account =
        LOAD '/vol/automed/data/bank_branch/account.tsv'
        AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);
grunt>STORE account INTO 'account_copy' USING PigStorage(',');
```

Running Pig Scripts

copy_account.pig

```
account =
    LOAD '/vol/automed/data/bank_branch/account.tsv'
    AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);

STORE account INTO 'account_copy' USING PigStorage(',');
```

Interactive: inspecting schemas and viewing results

```
pig -x local
grunt>account =
        LOAD '/vol/automed/data/bank_branch/account.tsv'
        AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);
grunt>DESCRIBE account;
grunt>DUMP account;
```

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Project π

FOREACH ⟨alias⟩ GENERATE ⟨colname⟩,...
Projects certain column names from an alias

π_{sortcode} account

```
account_sortcode_bag=
    FOREACH account
    GENERATE sortcode;

account_sortcode=
    DISTINCT account_sortcode_bag;
```

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Select σ

FILTER ⟨alias⟩ BY ⟨predicate⟩
Only passes those tuples in ⟨alias⟩ that match the ⟨predicate⟩

$\sigma_{\text{rate}>0}$ account

```
account_with_rate=
    FILTER account
    BY rate >0.0;
```

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Product $\times$

CROSS $\langle$alias$\rangle$,$\langle$alias$\rangle$

Produce the Cartesian product of two relations

branch $\times$ $\sigma_{\text{rate}>0}$ account

```
branch_account_with_rate =
    CROSS branch, account_with_rate;
```

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Join $\bowtie$

JOIN ⟨alias⟩ BY ⟨colname⟩, ⟨alias⟩ BY ⟨colname⟩
Perform a equi-join between two relations on the specified columns.

branch $\bowtie$ $\sigma_{\text{rate}>0}$ account

```
branch_with_interest_account =
    JOIN branch BY branch::sortcode,
         account_with_rate BY account_with_rate::sortcode;
```

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Union $\cup$

UNION ⟨alias⟩,⟨alias⟩
Perform a bag based union between two relations

π_{sortcode} branch $\cup$ π_{no} account

```
branch_sortcode=
    FOREACH branch
    GENERATE sortcode;
account_no=
    FOREACH account
    GENERATE no;
all_ids_bag=
    UNION branch_sortcode,account_no
all_ids=
    DISTINCT all_ids_bag;
```

Pig: Implementation of the RA

- Project π
- Select σ
- Product $\times$
- Join $\bowtie$
- Union $\cup$
- Difference $-$

account

no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Difference $-$

No direct implementation. Can achieve the same result by performing a LEFT join, and then elimating rows with null values.

π_{no} account $-$ π_{no} movement

```
account_and_movement=
    JOIN account BY no LEFT,
    movement BY no;
account_without_movement=
    FILTER account_and_movement
    BY movement::no IS NULL;
account_no_without_movement=
    FOREACH account_without_movement
    GENERATE no
```

Quiz 1: Understanding Pig Scripts (1)

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

```
a = FILTER account BY type=='current';
ap = FOREACH a GENERATE no,sortcode;
```

What is the value of ap in the Pig Script?

A

ap	
no	sortcode
100	67
103	34
107	56
125	56

B

ap	
no	sortcode
100	67
103	34
107	56

C

ap	
no	sortcode
100	67
107	56

D

ap
sortcode
67
34
56
56

Quiz 2: Understanding Pig Scripts (2)

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

```
a = FILTER branch BY cash<50000;
b = FILTER account BY type=='deposit';
ab = JOIN a BY sortcode, b BY sortcode;
abp = FOREACH ab GENERATE a::sortcode AS sortcode;
```

What is the value of abp in the Pig Script?

A

abp
sortcode
56
67

B

abp
sortcode
56

C

abp
sortcode
67

D

abp
sortcode
34

Quiz 3: RA and Pig Equivalence

```
a = FILTER branch BY cash<50000;
b = FILTER account BY type=='deposit';
ab = JOIN a BY sortcode, b BY sortcode;
abp = FOREACH ab GENERATE a::sortcode AS sortcode;
adpd = DISTINCT abp;
```

Which RA expression is equivalent to abpd in the Pig Script?

A

$\pi_{\text{sortcode}}(\sigma_{\text{cash}<50000}\ \text{branch} \cup \sigma_{\text{type}=\text{'deposit'}}\ \text{account})$

B

$\pi_{\text{sortcode}}(\sigma_{\text{cash}<50000}\ \text{branch} \cap \sigma_{\text{type}=\text{'deposit'}}\ \text{account})$

C

$\pi_{\text{sortcode}}\ \sigma_{\text{cash}<50000}\ \text{branch} \cup \pi_{\text{sortcode}}\ \sigma_{\text{type}=\text{'deposit'}}\ \text{account}$

D

$\pi_{\text{sortcode}}\ \sigma_{\text{cash}<50000}\ \text{branch} \cap \pi_{\text{sortcode}}\ \sigma_{\text{type}=\text{'deposit'}}\ \text{account}$

Worksheet: Translating RA to Pig

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

movement			
mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

key branch(sortcode)
key branch(bname)
key movement(mid)
key account(no)

movement(no) $\stackrel{fk}{\Rightarrow}$ account(no)

account(sortcode) $\stackrel{fk}{\Rightarrow}$ branch(sortcode)

1. π_{no} movement
2. $\pi_{\text{cname,mid,amount}}\, \sigma_{\text{amount}<0.0}(\text{account} \bowtie \text{movement})$
3. π_{sortcode} branch $-$ $\pi_{\text{sortcode}}\, \sigma_{\text{type}=\text{'deposit'}}$

Worksheet: Translating RA to Pig (1)

π_{no} movement

```
movement_no_bag =
    FOREACH movement
    GENERATE no;

movement_no =
    DISTINCT movement_no_bag;

DUMP movement_no;
```

Worksheet: Translating RA to Pig (2)

$\pi_{\text{cname,mid,amount}}\, \sigma_{\text{amount}<0.0}(\text{account} \bowtie \text{movement})$

```
withdrawl =
    FILTER movement
    BY     amount<0;

account_with_withdrawl =
    JOIN account BY no,
         withdrawl BY no;

account_and_withdrawl_amount =
    FOREACH account_with_withdrawl
    GENERATE cname, mid, amount;

DUMP account_and_withdrawl_amount;
```

Worksheet: Translating RA to Pig (3)

π_{sortcode} branch $- \pi_{\text{sortcode}}\, \sigma_{\text{type}=\text{'deposit'}}$

```
deposit =
    FILTER account
    BY     type=='deposit';

branch_account =
    JOIN branch BY sortcode LEFT,
         deposit BY sortcode;

branches_without_deposit =
    FILTER branch_account
    BY no IS NULL;

sortcodes_without_desposit =
    FOREACH branches_without_deposit
    GENERATE branch::sortcode AS sortcode;

DUMP sortcodes_without_desposit;
```

Relations as attributes: GROUP and FLATTEN

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int, no:int, amount:double, tdate:bytearray);
```

movement			
mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

Relations as attributes: GROUP and FLATTEN

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int, no:int, amount:double, tdate:bytearray);

account_movements =
    GROUP movement
    BY    no;
```

account_movements	
group	movement
100	{⟨1000,100,2300.0,1999-01-05⟩,⟨1002,100,-223.45,1999-01-08⟩,⟨1006,100,10.23,1999-01-15⟩}
101	{⟨1001,101,4000.0,1999-01-05⟩,⟨1008,101,1230.0,1999-01-15⟩}
103	{⟨1005,103,145.5,1999-01-12⟩}
107	{⟨1004,107,-100.0,1999-01-11⟩,⟨1007,107,345.56,1999-01-15⟩}
119	{⟨1009,119,5600.0,1999-01-18⟩}

Relations as attributes: GROUP and FLATTEN

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int, no:int, amount:double, tdate:bytearray);

account_movements =
    GROUP movement
    BY    no;

movement_copy =
    FOREACH account_movements
    GENERATE FLATTEN(movement);
```

movement_copy			
mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

Relations as attributes: GROUP and FLATTEN

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int,no:int,amount:double,tdate:bytearray);

account_movements =
    GROUP movement
    BY    no;


account_balance =
    FOREACH account_movements
    GENERATE group AS no,
             SUM(movement.amount) AS balance;
```

account_balance	
no	balance
100	2086.78
101	5230.00
103	145.50
107	245.56
119	5600.00

Aggregates Operators in Pig

Function	Result
int COUNT(bag)	Returns the number of not null values in the bag.
int COUNT_STAR(bag)	Returns the number of values in the bag (including any null values).
double AVG(bag)	Returns the average of values in the bag.
double MAX(bag)	Returns the maximum value in the bag.
double MIN(bag)	Returns the minimum value in the bag.
double SUM(bag)	Returns the sum of values in the bag.
bag DIFF(bag a,bag b)	Returns those tuples in a that do not appear in b

To achieve the equivalent of SQL's GROUP BY and use of aggregate operators:

- Use GROUP to build a bag of tuples for each value in the group
- Apply a Pig aggregate operator to the bag

Quiz 4: Understanding Pig Scripts (3)

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

```
ab = JOIN account BY no LEFT, movement BY no;
abg = GROUP ab BY account::no;
abr = FOREACH abg GENERATE group,COUNT(ab.movement::no) AS no_mv;
```

What is the value of abr in the Pig Script?

A

abr	
group	no_mv
100	1
101	1
103	1
107	1
119	1
125	0

B

abr	
group	no_mv
100	3
101	2
103	1
107	2
119	1
125	1

C

abr	
group	no_mv
100	3
101	2
103	1
107	2
119	1
125	0

D

abr	
group	no_mv
100	3
101	2
103	1
107	2
119	1

Optimisation of Scripts: Project Early

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int, no:int, amount:double, tdate:bytearray);
```

movement			
mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

Optimisation of Scripts: Project Early

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int,no:int,amount:double,tdate:bytearray);

movement_data =
    FOREACH movement
    GENERATE no,amount;

account_movements =
    GROUP movement_data
    BY    no;
```

account_movements	
group	movement
100	{⟨100,2300.0⟩,⟨100,-223.45⟩,⟨100,10.23⟩}
101	{⟨101,4000.0⟩,⟨101,1230.0⟩}
103	{⟨1103,145.5,⟩}
107	{⟨107,-100.0⟩,⟨107,345.56⟩}
119	{⟨119,5600.0⟩}

Optimisation of Scripts: Project Early

```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int,no:int,amount:double,tdate:bytearray);

movement_data =
    FOREACH movement
    GENERATE no,amount;

account_movements =
    GROUP movement_data
    BY    no;

movement_project =
    FOREACH account_movements
    GENERATE FLATTEN(movement);
```

movement_project	
no	amount
100	2300.00
101	4000.00
100	-223.45
107	-100.00
103	145.50
100	10.23
107	345.56
101	1230.00
119	5600.00

Optimisation of Scripts: Project Early

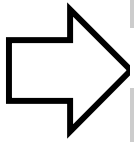
```
movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int,no:int,amount:double,tdate:bytearray);

movement_data =
    FOREACH movement
    GENERATE no,amount;

account_movements =
    GROUP movement_data
    BY    no;

account_balance =
    FOREACH account_movements
    GENERATE group AS no,
             SUM(movement.amount) AS balance;
```

account_balance	
no	balance
100	2086.78
101	5230.00
103	145.50
107	245.56
119	5600.00

Nested Statements

SQL Query to find total of credits and of debits

```
SELECT    account.no,
          COUNT(movement.mid) AS no_trans,
          SUM(CASE WHEN amount>0.0 THEN amount ELSE 0.0 END) AS credit,
          SUM(CASE WHEN amount<0.0 THEN amount ELSE 0.0 END) AS debit
FROM      account LEFT JOIN movement ON account.no=movement.no
GROUP BY account.no
```



no	no_trans	credit	debit
100	3	2310.23	-223.45
103	1	145.50	0.0
119	1	5600.00	0.0
125	0	0.0	0.0
107	2	345.56	-100.00
101	2	5230.00	0.0

Powerful feature of SQL is that any statement that returns a value may appear in an aggregate function.

Nested Statements

SQL Query to find total of credits and of debits

```
SELECT    account.no,
          COUNT(movement.mid) AS no_trans,
          SUM(CASE WHEN amount>0.0 THEN amount ELSE 0.0 END) AS credit,
          SUM(CASE WHEN amount<0.0 THEN amount ELSE 0.0 END) AS debit
FROM      account LEFT JOIN movement ON account.no=movement.no
GROUP BY account.no
```

Pig Script to find total of credits and of debits

```
account_and_movement =
    JOIN account BY no LEFT,
         movement BY no;
account_detail =
    GROUP account_and_movement BY account::no;
account_credits_and_debits =2
    FOREACH account_detail {
        credit =
           FILTER account_and_movement
           BY     amount>0.0;
        debit =
           FILTER account_and_movement
           BY     amount<0.0;
        GENERATE group AS no,
                 COUNT(account_and_movement) AS no_trans,
                 SUM(credit.amount) AS credit,
                 SUM(debit.amount) AS debit;
    }
```

Worksheet: Translating SQL to Pig

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

movement			
mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

key branch(sortcode)
key branch(bname)
key movement(mid)
key account(no)

movement(no) $\stackrel{fk}{\Rightarrow}$ account(no)

account(sortcode) $\stackrel{fk}{\Rightarrow}$ branch(sortcode)

Worksheet: Translating SQL to Pig (1)

```
SELECT  branch.bname,
        account.no
FROM    branch
        JOIN account ON branch.sortcode=account.sortcode
        JOIN movement ON account.no=movement.no
WHERE   movement.amount<0
```

```
withdrawl =
    FILTER movement
    BY     amount<0;

account_with_withdrawl_bag =
    JOIN account BY no,
         withdrawl BY no;

account_with_withdrawl =
    DISTINCT account_with_withdrawl_bag;

branch_with_withdrawl =
    JOIN account_with_withdrawl BY sortcode,
         branch BY sortcode;

branch_with_withdrawl_no =
    FOREACH branch_with_withdrawl
    GENERATE bname,account::no;

DUMP branch_with_withdrawl_no;
```

Worksheet: Translating SQL to Pig (2)

```
SELECT account.cname,
       SUM(movement.amount) AS balance
FROM   account
       JOIN movement ON account.no=movement.no
GROUP BY account.cname
```

```
branch =
    LOAD '/vol/automed/data/bank_branch/branch.tsv'
    AS (sortcode:int,bname:chararray,cash:float);

account =
    LOAD '/vol/automed/data/bank_branch/account.tsv'
    AS (no:int,type:chararray,cname:chararray,rate:float,sortcode:int);

movement =
    LOAD '/vol/automed/data/bank_branch/movement.tsv'
    AS (mid:int,no:int,amount:double,tdate:bytearray);

account_movement =
    JOIN account BY no LEFT, movement BY no;

customer_details =
    GROUP account_movement BY account::cname;

customer_balance =
    FOREACH customer_details
    GENERATE group AS cname, SUM(account_movement.movement::amount) AS balance;
```

Worksheet: Translating SQL to Pig (3)

```
SELECT    branch.sortcode,
          branch.bname,
          COUNT(CASE WHEN type='current' THEN no ELSE NULL END) AS current,
          COUNT(CASE WHEN type='deposit' THEN no ELSE NULL END) AS deposit
FROM      account JOIN branch ON account.sortcode=branch.sortcode
GROUP BY branch.sortcode, branch.bname
ORDER BY branch.sortcode, branch.bname
```

```
branch_account =
    JOIN branch BY sortcode, account BY sortcode;
branch_detail =
    GROUP branch_account BY (branch::sortcode,branch::bname);
branch_account_types =
    FOREACH branch_detail {
        current =
            FILTER branch_account
            BY type == 'current';
        deposit =
            FILTER branch_account
            BY type == 'deposit';
        GENERATE group.sortcode AS sortcode,
                 group.bname AS bname,
                 COUNT(current.no) AS current,
                 COUNT(deposit.no) AS deposit;
    }
branch_account_types_ordered =
    ORDER branch_account_types
    BY    sortcode, bname;
```

Pig to Hadoop Translation

Pig scripts are interpreted into a sequence of Hadoop Map, Combine, Shuffle, and Reduce operations.

- In general, a Pig script may require multiple MapReduce processes to be run.
- Map and Combine processes run on nodes containing data.
- Number of Reduce nodes used specified in the Pig script (and defaults to 1!)
- Temporary files are used to allow output of one MapReduce process to be fed back as input to another MapReduce process.
- Projects (from GENERATE in Pig) are automatically pushed inside Joins, but otherwise little optimisation is performed by the Pig interpreter.

Quiz 5: Pig Operations in Map Reduce

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

Which Pig Operator may be executed entirly on a Map Process?

A

JOIN

B

DISTINCT

C

GENERATE

D

UNION

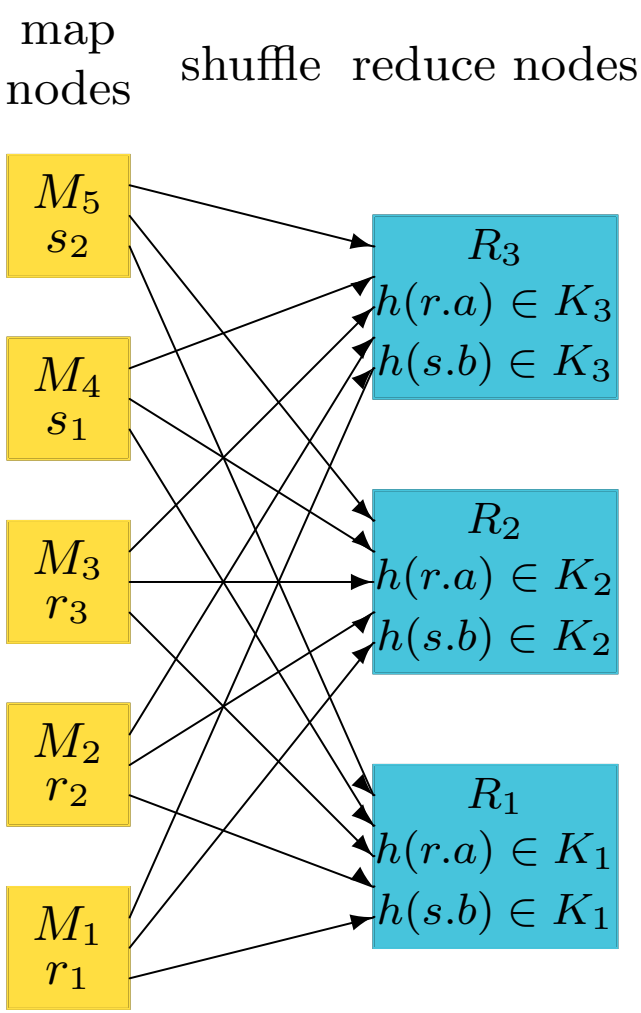
Pig Operators in MapReduce

Pig Operator	Map or Reduce
FILTER R BY $A == val$	Map
FOREACH R GENERATE $A, B, \ldots$	Map
CROSS R, S	Reduce
GROUP R BY A	Combine,Reduce
JOIN R BY A, S BY B	Reduce
JOIN R BY A LEFT OUTER, S BY B;	Reduce
JOIN R BY A RIGHT OUTER, S BY B;	Reduce
UNION R, S	Reduce

Parallism in Reduce Operators

- Control number of reduce nodes by a PARALLEL option at the end of reduce operator.
- Default is the have one reduce node.

Types of Join

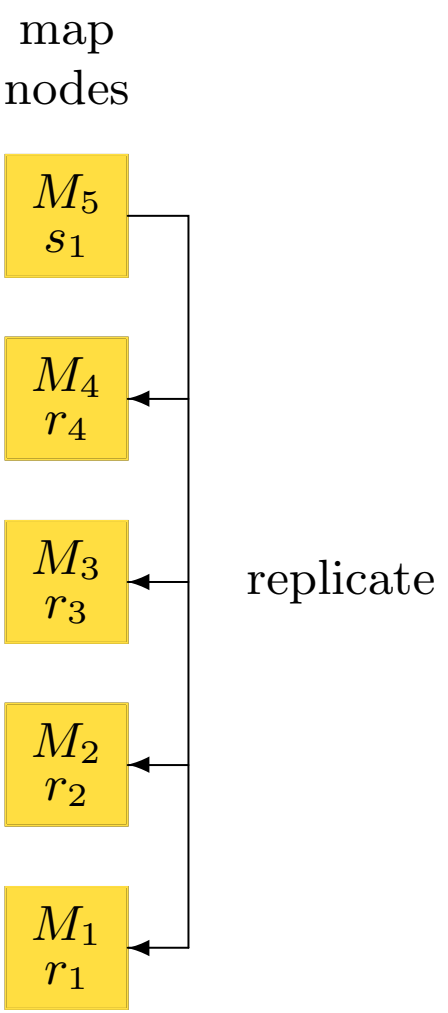


Default implementation of Join

```
t_u = JOIN r BY a, s BY b
```

- Standard JOIN will use a shuffle to distribute the tables of the join over the reduce nodes
- uses the Java hashCode method

Types of Join



Replicated Joins

```
t_u = JOIN r BY a, s BY b USING 'replicated'
```

- JOIN with the replicated option causes the entire right hand table to be copied onto the all map nodes holding the left hand table.
- replicated joins executed as a Map process.

Quiz 6: Pig Joins

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

The size of **branch** is such it easily fits on one node, whilst **account** does not.

Which Pig Script is invalid?

A

```
ba = JOIN account BY sortcode, branch BY sortcode;
```

B

```
ba = JOIN account BY sortcode RIGHT, branch BY sortcode USING 'replicated';
```

C

```
ba = JOIN account BY sortcode LEFT, branch BY sortcode USING 'replicated';
```

D

```
ba = JOIN account BY sortcode, branch BY sortcode USING 'replicated';
```