Hash-based Grouping and Aggregation

R

co-grouping vs join processing

-> sort merge join

-> generalized co-grouped join

Hash-based Grouping and Aggregation

R

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

schema

```
SELECT R.x, aggregate(R.y)
FROM R
GROUP BY R.x;
```

Hash-based Grouping and Aggregation

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

HashBasedGrouping(R, aggregate()):

```
HashMap hm = new HashMap();                    //initialize hash map
List group = NULL;                             //handle to group of R.x

ForEach r in R:                                //for every tuple in R
    If NOT hm.contains( r.x ):                 //check if already seen this key
        group = new List();                    //create new group
    Else:                                      //i.e. key already in hash map (key seen before)
```

42

Hash-based Grouping and Aggregation

R

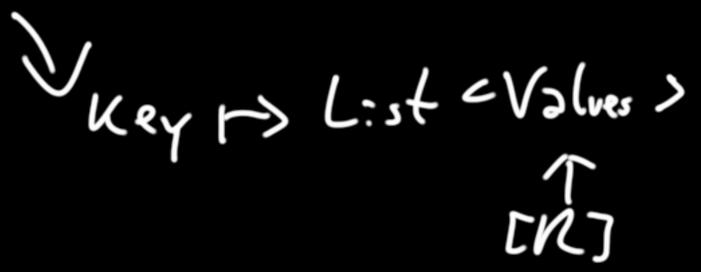
grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

HashBasedGrouping(R, aggregate()):

```
    HashMap hm = new HashMap();          //initialize hash map
    List group = NULL;                   //handle to group of R.x

    ForEach r in R:                      //for every tuple in R
        If NOT hm.contains( r.x ):       //check if already seen this key
            group = new List();          //create new group
        Else:                            //i.e. key already in hash map (key seen before)
            group = hm.get( r.x );       //get existing group from hash map
```



Hash-based Grouping and Aggregation

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

```
HashBasedGrouping( R, aggregate() ):
    HashMap hm = new HashMap();                    //initialize hash map
    List group = NULL;                             //handle to group of R.x

    ForEach r in R:                                //for every tuple in R
        If NOT hm.contains( r.x ):                 //check if already seen this key
            group = new List();                    //create new group
        Else:                                      //i.e. key already in hash map (key seen before)
            group = hm.get( r.x );                 //get existing group from hash map
        group.append( r );                         //append element to group
    // hm.put( r.x, group );                       //insert/replace group in hash map
```

Hash-based Grouping and Aggregation

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

HashBasedGrouping(R, aggregate()):

```
    HashMap hm = new HashMap();                    //initialize hash map
    List group = NULL;                             //handle to group of R.x

    ForEach r in R:                                //for every tuple in R
        If NOT hm.contains( r.x ):                 //check if already seen this key
            group = new List();                    //create new group
        Else:                                      //i.e. key already in hash map (key seen before)
            group = hm.get( r.x );                 //get existing group from hash map
        group.append( r );                         //append element to group
        hm.put( r.x, group );                      //insert/replace group in hash map

    ForEach key in hm:                             //loop over all existing keys in hash map
```

Hash-based Grouping and Aggregation

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

HashBasedGrouping(R, aggregate()):

```
    HashMap hm = new HashMap();                    //initialize hash map
    List group = NULL;                             //handle to group of R.x

    ForEach r in R:                                //for every tuple in R
        If NOT hm.contains( r.x ):                 //check if already seen this key
            group = new List();                    //create new group
        Else:                                      //i.e. key already in hash map (key seen before)
            group = hm.get( r.x );                 //get existing group from hash map
        group.append( r );                         //append element to group
        hm.put( r.x, group );                      //insert/replace group in hash map

    ForEach key in hm:                             //loop over all existing keys in hash map
        group = hm.get( key );                     //retrieve result group for that key
```

Hash-based Grouping and Aggregation

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

HashBasedGrouping(R, aggregate()):

```
    HashMap hm = new HashMap();                    //initialize hash map
    List group = NULL;                             //handle to group of R.x

    ForEach r in R:                                //for every tuple in R
        If NOT hm.contains( r.x ):                 //check if already seen this key
            group = new List();                    //create new group
        Else:                                      //i.e. key already in hash map (key seen before)
            group = hm.get( r.x );                 //get existing group from hash map
        group.append( r );                         //append element to group
        hm.put( r.x, group );                      //insert/replace group in hash map

    ForEach key in hm:                             //loop over all existing keys in hash map
        group = hm.get( key );                     //retrieve result group for that key
        aggregationResult = aggregate( group );    //call aggregation function on that group
        output( key,  aggregationResult );         //output result pairs
```

Sort-based Grouping and Aggregation

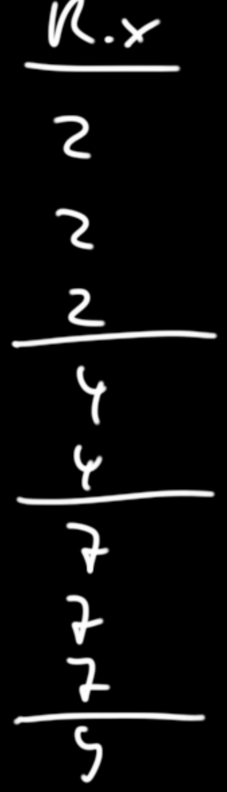
R

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

SortBasedGrouping(R, aggregate()):

```
    sort( R on R.x );                          //sort R on grouping attribute R.x
    Pointer PR = R[0];                         //initialize pointer to first tuple of R
    Value currentGroupValue = R[0].x;          //value of R.x for this group
    List group = new List();                   //create new group
    Do:                                        //start loop
        If PR.x != currentGroupValue:          //if current tuple belongs to same group
```



Sort-based Grouping and Aggregation

R

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

```
SortBasedGrouping( R, aggregate() ):
    sort( R on R.x );                                   //sort R on grouping attribute R.x
    Pointer PR = R[0];                                  //initialize pointer to first tuple of R
    Value currentGroupValue = R[0].x;                   //value of R.x for this group
    List group = new List();                            //create new group
    Do:                                                 //start loop
        If PR.x != currentGroupValue:                   //if current tuple belongs to same group
            aggregationResult  = aggregate( group );    //aggregate existing result group
            output( currentGroupValue,  aggregationResult );  //output result pairs
            group = new List();                         //create new group
            currentGroupValue = PR.x;                   //re-init value of R.x for this group
        group.append( PR );                             //append this tuple to result group anyway
        PR++;                                           //move pointer forward to next tuple
    While PR != R.end;                                  //end of table?
```

group closing

Sort-based Grouping and Aggregation

grouping attribute: R.x

aggregation function: aggregate(MultiSet<R‘>): MultiSet<R‘> ↦ <ScalarType>, [R‘] ⊆ [R]

SortBasedGrouping(R, aggregate()):

```
sort( R on R.x );                                          //sort R on grouping attribute R.x
Pointer PR = R[0];                                         //initialize pointer to first tuple of R
Value currentGroupValue = R[0].x;                          //value of R.x for this group
List group = new List();                                   //create new group
Do:                                                        //start loop
    If PR.x != currentGroupValue:                          //if current tuple belongs to same group
        aggregationResult  = aggregate( group );           //aggregate existing result group
        output( currentGroupValue,  aggregationResult );   //output result pairs
        group = new List();                                //create new group
        currentGroupValue = PR.x;                          //re-init value of R.x for this group
    group.append( PR );                                    //append this tuple to result group anyway
    PR++;                                                  //move pointer forward to next tuple
While PR != R.end;                                         //end of table?
aggregationResult = aggregate( group );                    //aggregate existing result group
output( currentGroupValue,  aggregationResult );           //output result pairs
```