FM on Bare Metal

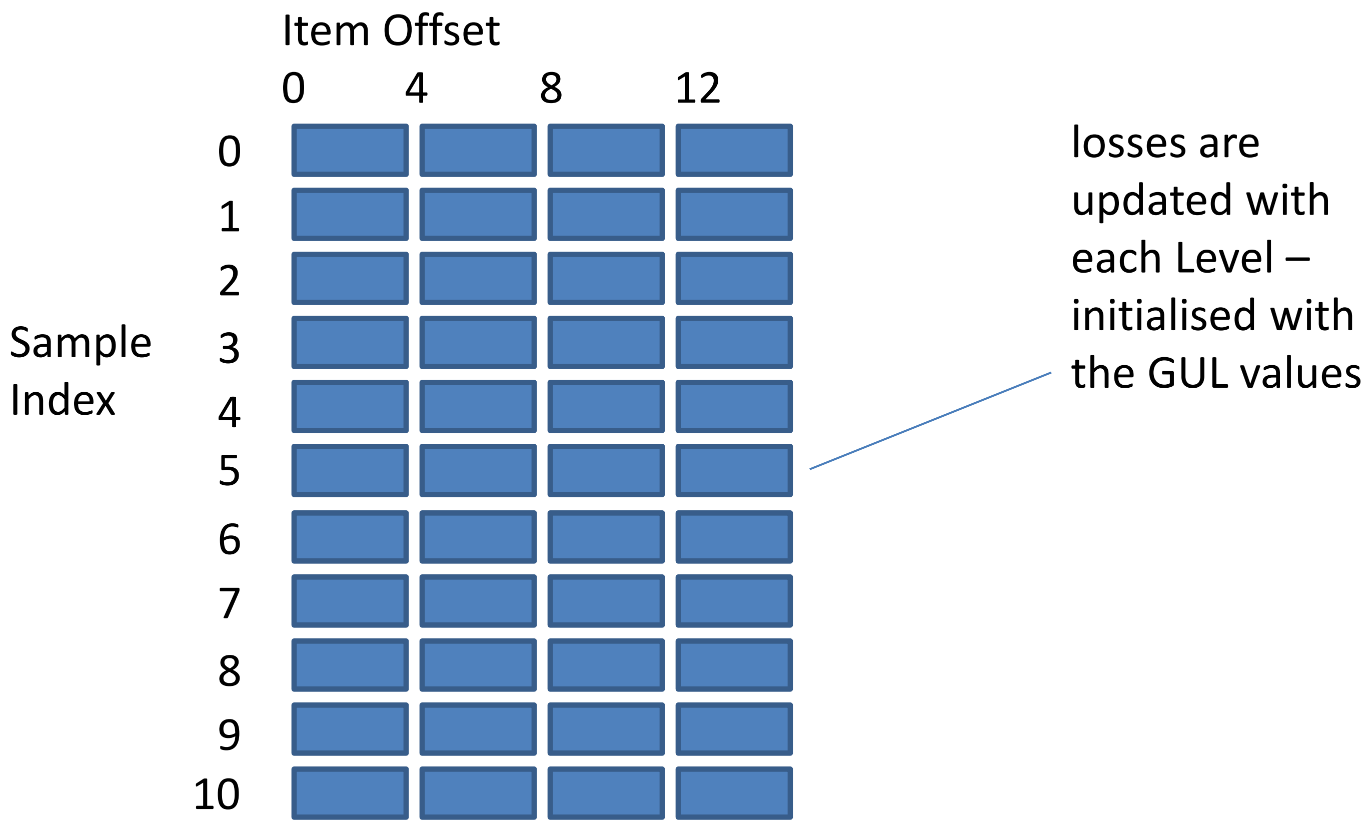
Jon G – Implementation Design Notes

DATA STRUCTURES

levelLossWork Table

- This is the main working data structure for the losses due to an event
- Each row corresponds to a sample index
- Each column corresponds to losses for an item
  - It is labelled with an offset, which is a multiple of 4 because the losses are 4 byte floating point values
- The table is updated for each level.

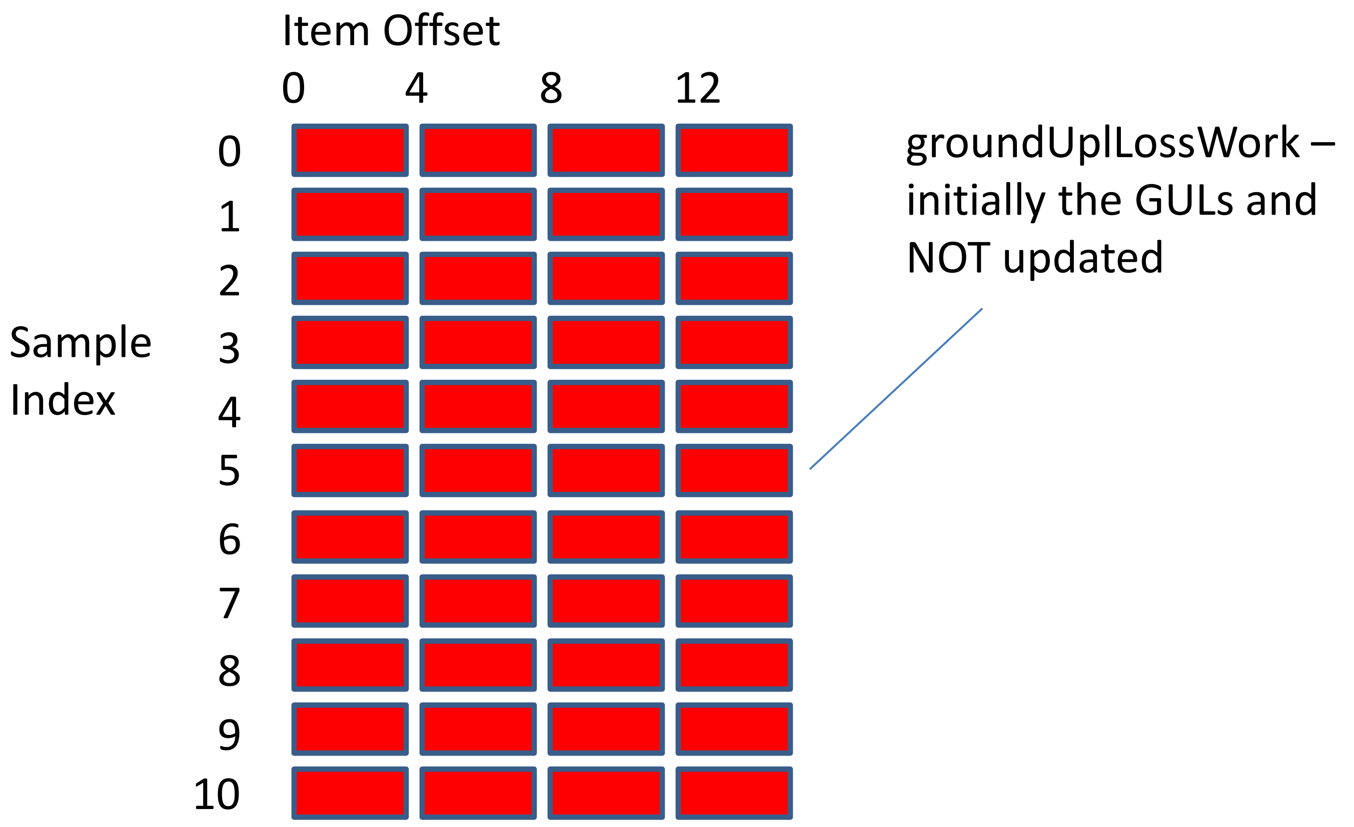
levelLossWork Table for 4 items


Item Offset
0
4
8
12
Sample Index
0
1
2
3
4
5
6
7
8
9
10
losses are updated with each Level – initialised with the GUL values

groundUpLossWork Table

- This is table of ground up losses used for certain back allocation calculations.
- It is initialised to be identical to the loss work table, but it is not updated for each level.

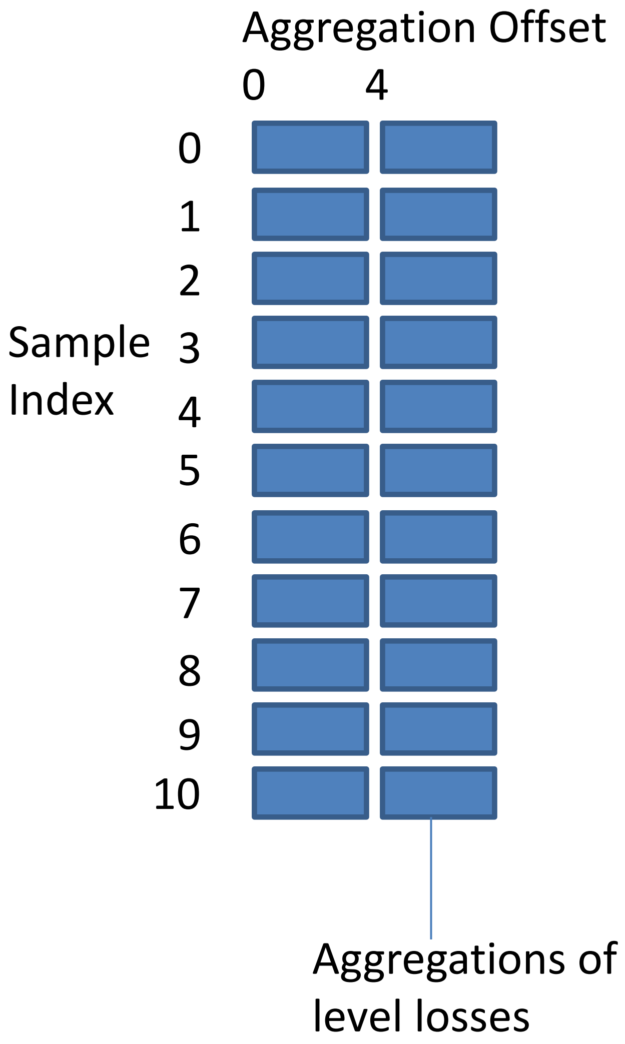
groundUpLossWork Table for 4 items

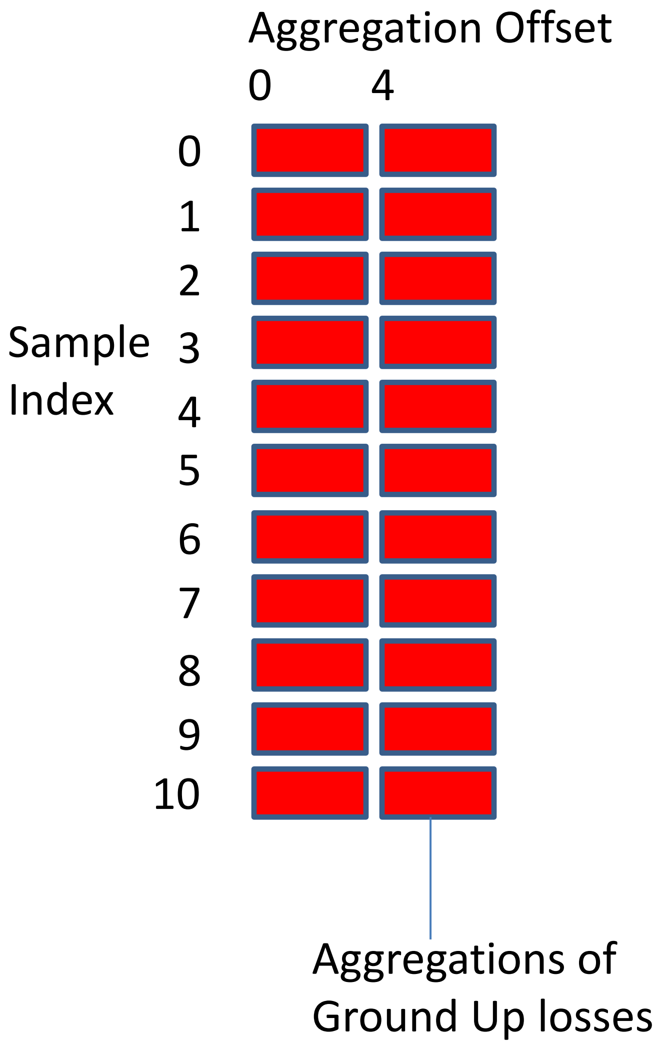


aggLevelLossWork

- aggLevelLossWork is a table for storing the results of the aggregations.
- It has one row for each sample.
- It has one column for each aggregation.
- There is a corresponding table for the aggregations of the Ground-Up Losses – aggGroundUpLossWork.
- Both get set up and torn down for each level, as the number of aggregations is different for each level.

aggLevelLossWork and aggGroundUpLossWork



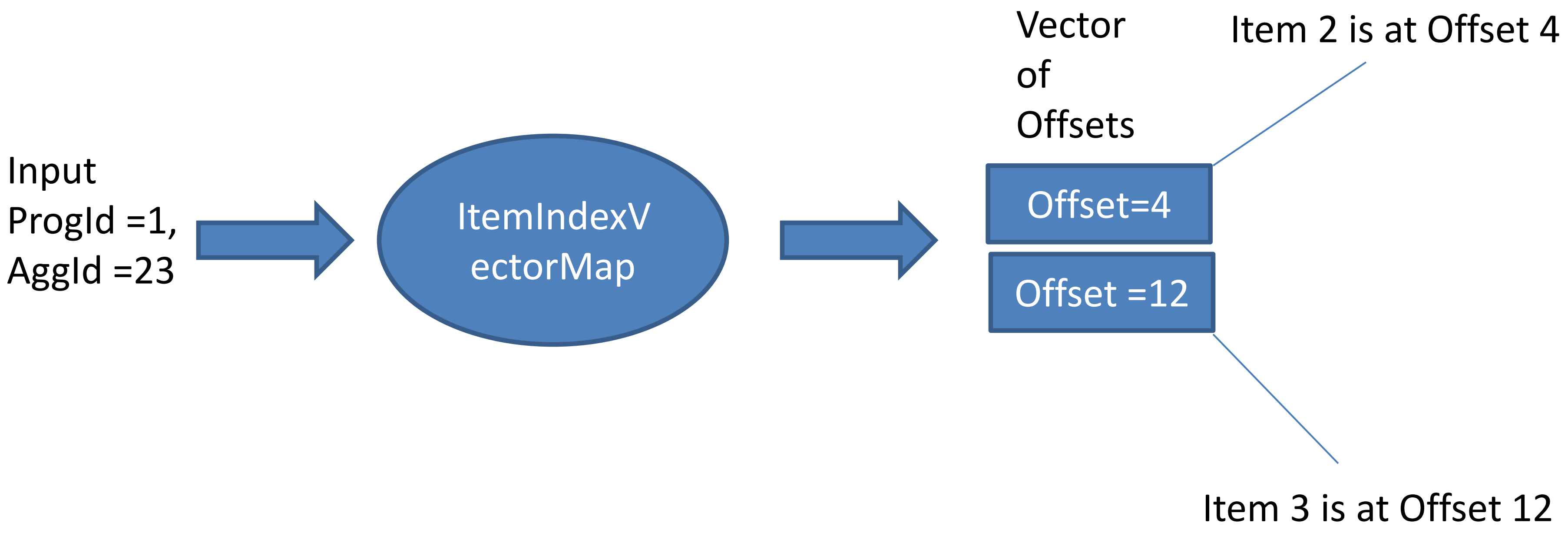


ItemIndexVector

- Maps from a pair – progId, aggId – to a vector of offsets of items which need to be summed.
- The key is a pair because different progIds may use the same aggIds.
- The vectors hold offsets into each rows of the levelLossWork table and the groundUpLossWork table – their layouts are identical
- Offsets of items are multiples of 4. This avoids a multiplication when calculating the position of the item in the table.

ItemIndexVector

For Prog Id 1, AggId 23 sums Items 2 and 3.



Note that offsets refer to each row of the levelLossWork table – ie we sum the items at the corresponding positions within each row for each sample.

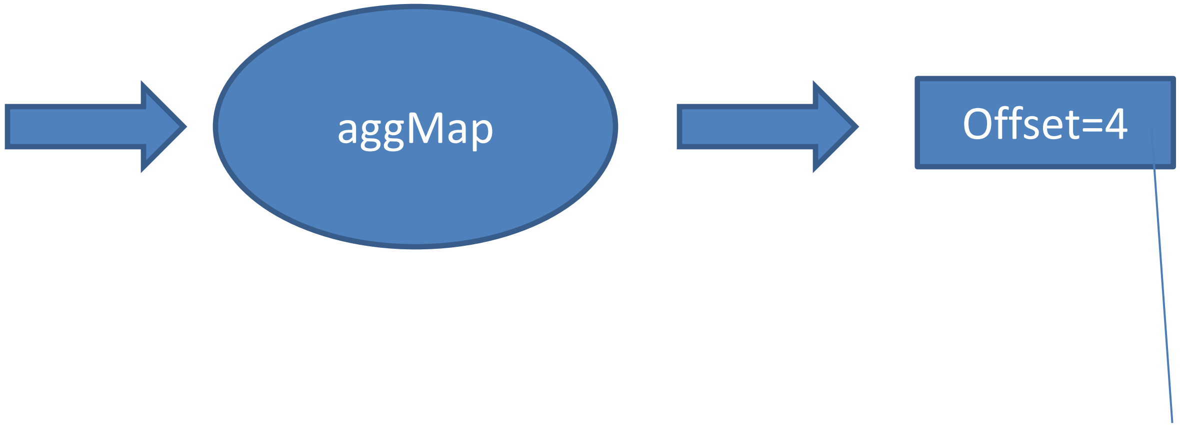
aggMap

- Maps from a pair – progId, aggId – to a single offsets into each row of the aggLevelLossWork and groundUpLevelLossWork tables.
- Again, it is a multiple of 4.

aggMap

AggId 23 sums Items 2 and 3 under Prog Id 1

Input
ProgId =1,
AggId =23



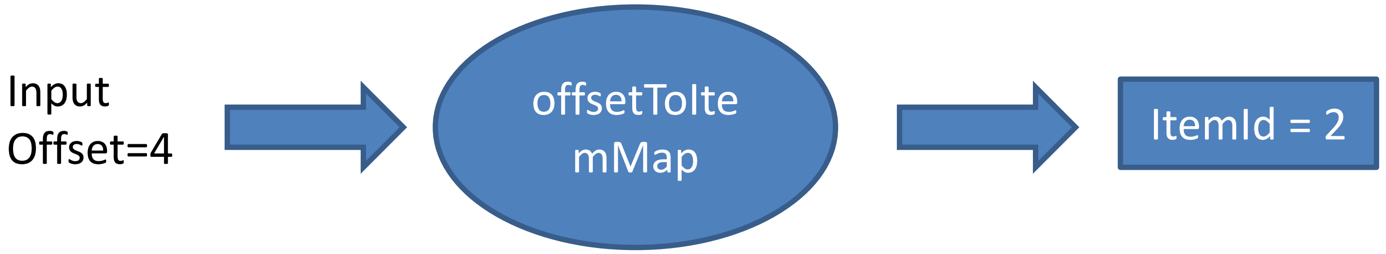
Aggregation 23 is at
Offset 4 in the
aggregation table

offsetToItem maps

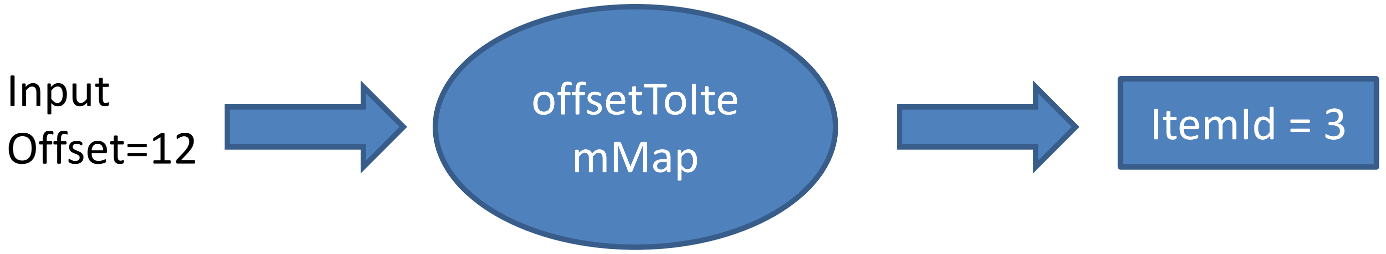
- These are maps to allow the look up of the itemId corresponding to an offset into the levelLossWorkTable.
- This is used for output, which we will do (for some options) by iterating over the itemIndexVector map.
  - The aggId is part of the key, but the value looked up is a vector of offsets, each of which need converting to item Ids in order to generate output losses.

offsetToItem maps

Offset 4 is Item 2



Offset 12 is Item 3



OPERATION OF LEVELS

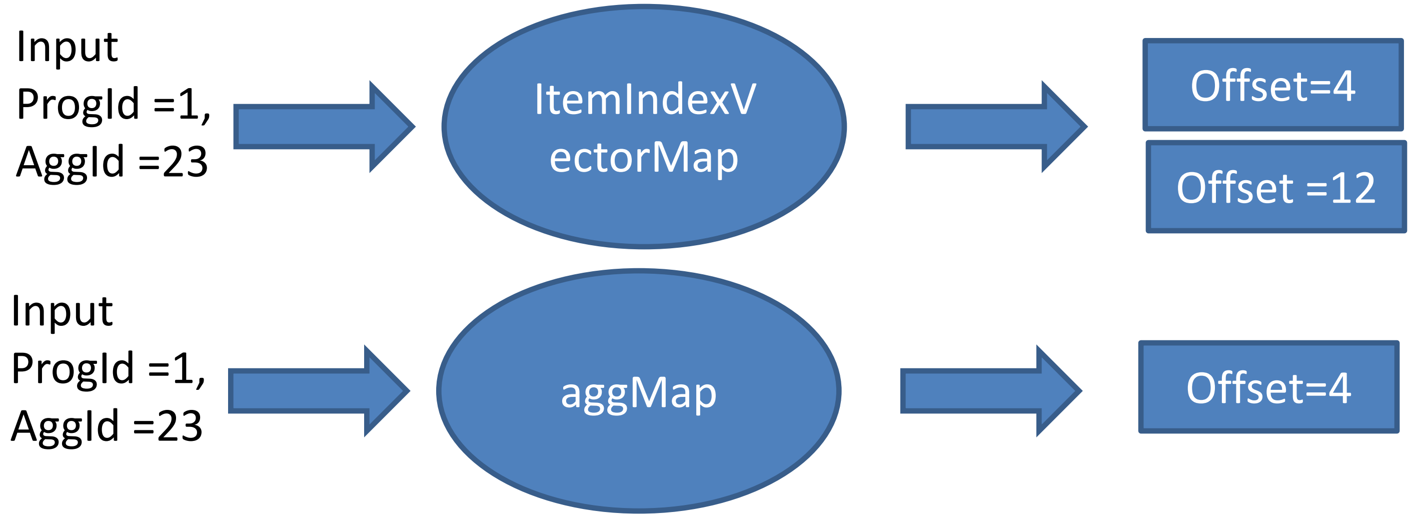
Steps in a Level Calculation if there is a single item in the aggregation

- Identify the offset into levelLossWork for each sample row given by the itemOffsetVector
- Perform the required calculation on the lossWork Item– eg deductibles, limits etc.
- Write the result straight back to the lossWork table.

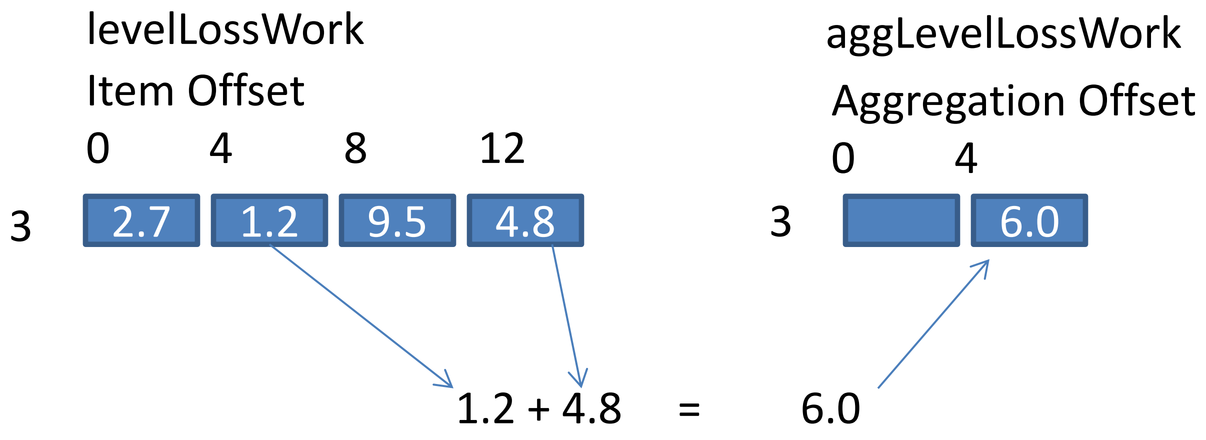
Steps in a Level Calculation if there is more than one item in the aggregation and no back allocation involving GUL

- Sum the items required for each aggregation for each sample using the vectors of offsets into levelLossWork in the itemOffsetVector map
- Place the results at the offsets into aggLevelLossWork given by the levelAggMap
- Perform the required calculation on the aggregation – eg deductibles, limits etc.
- Back allocate the result according to the allocation rule – ie in proportion to level loss.

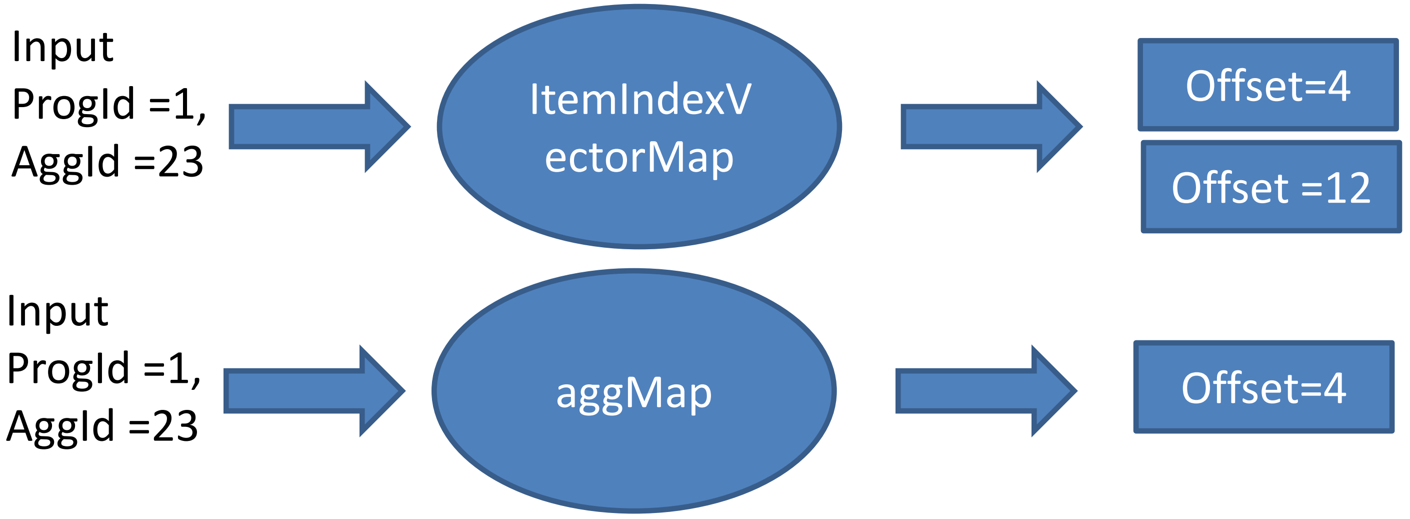
Basic Level Calculation - Aggregation



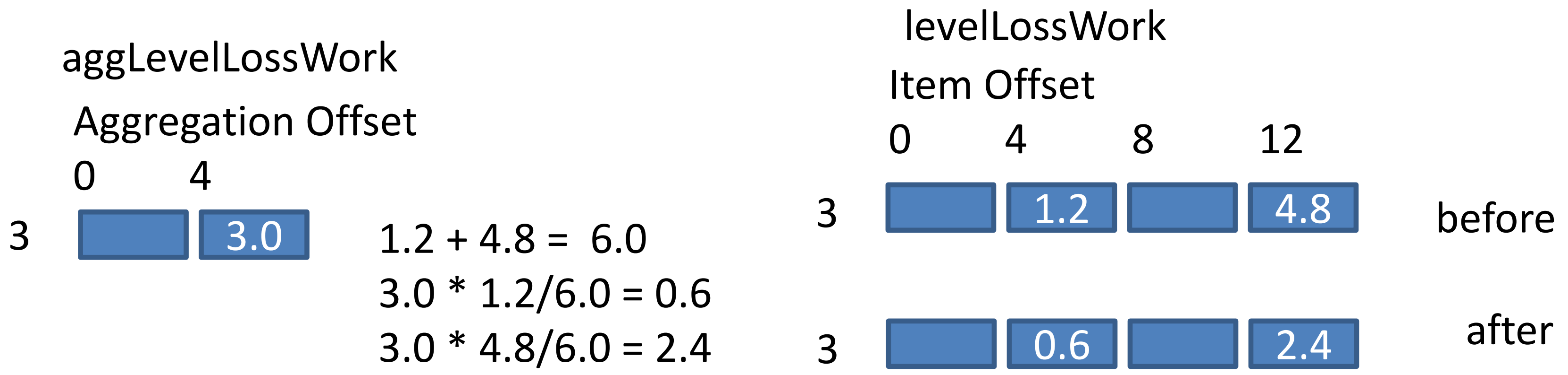
For each sample – eg sample 3 - sum the level losses at offsets 4 and 12 in each row of the levelLossWork table and put the result at offset 4 of the aggLevelLossWork table



Basic Level Calculation – Back Allocation



For each sample – eg sample 3 - allocate back in proportion to the losses before the level.



1.2 + 4.8 = 6.0
3.0 * 1.2/6.0 = 0.6
3.0 * 4.8/6.0 = 2.4

Back Allocation by Ground Up Loss (GUL) or Retained Loss (RL)

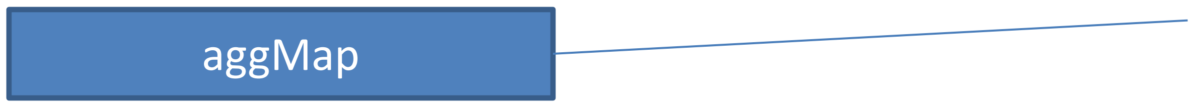
- Back allocations by GUL or RL require the sum of the GULs for the aggregation to be calculated – the offsets into the sample rows are the same as for the level Losses.
- Back allocation is in proportion to the GUL or RL for the item as a proportion of the loss summed across the items.

OPERATION OF OUTPUT

Output of Aggregated Losses with a Single Layer

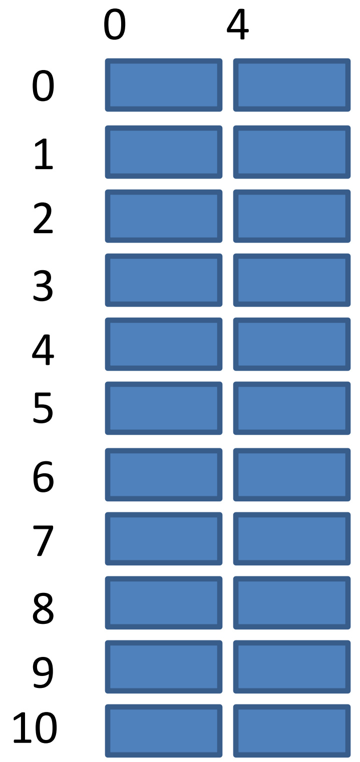
- If AllocRule_ID is zero, the aggregated losses are output.
- We iterate through the aggMap
- The Prog Id and Aggregation ID are found from the key to the map.
- The value stored in aggMap is offset for the aggregation and hence we can look up the aggregated loss for each sample.

Output of Aggregated Losses with a Single Layer



Iterate over aggMap to get the aggId (23) and offset into aggLevelLossWork (4)

Key is progId, aggId
Value is offset into aggLevelLossWork



For each mapping item in aggMap:
1) output the header first – eventId, outputId(=aggId) etc
2) iterate over the aggLevelLossWork table to output the sample index and value eg at position 4 in each row

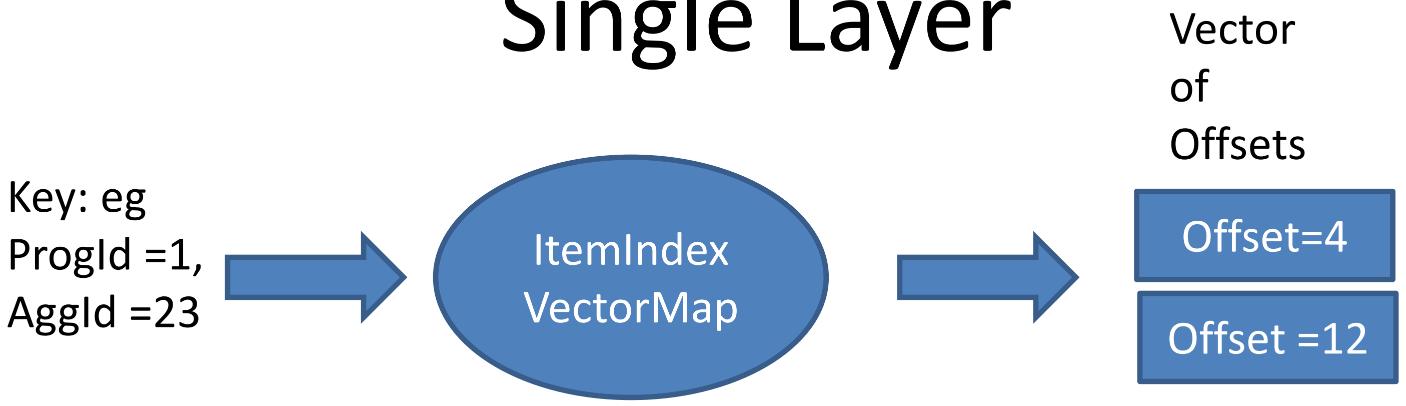
Output of Aggregated Losses with multiple Layers

- The aggregations and outputs are produced for each layer in turn.
- Note that the aggregation table is over-written for each layer.

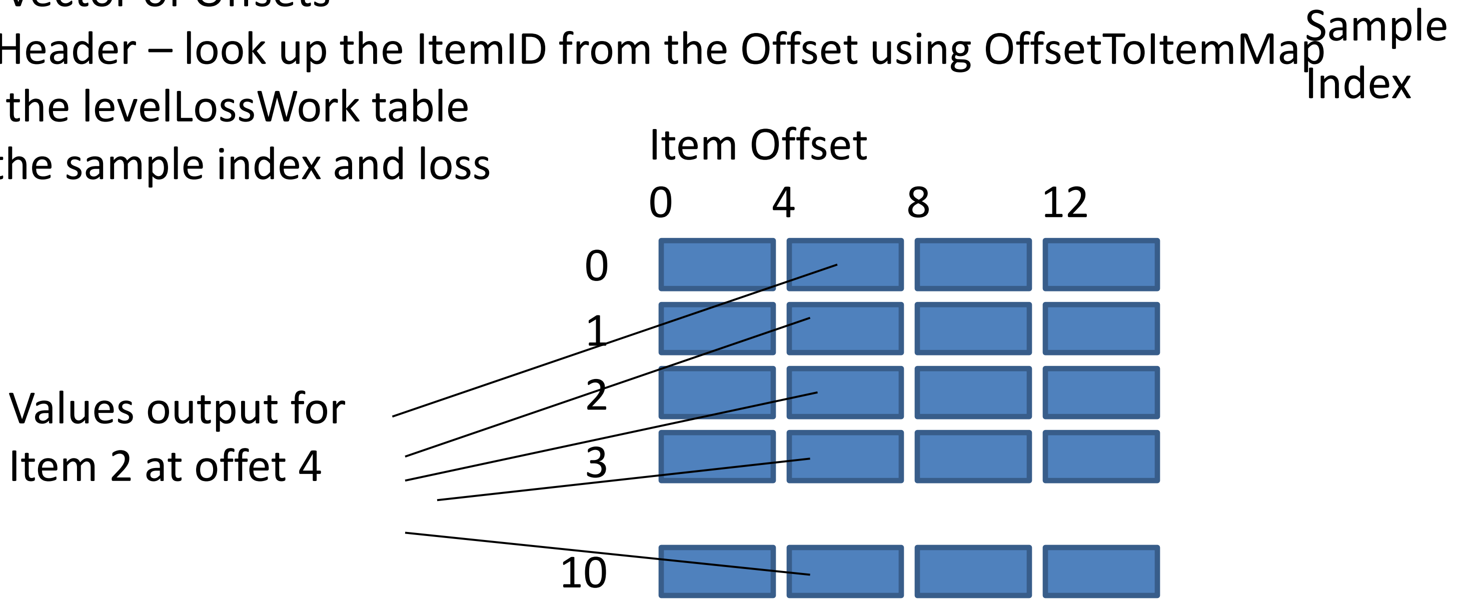
Output of Back Allocated Losses with a Single Layer

- If AllocRule_ID is -1, 1,2,3,etc, the back allocated losses are output.
- In this case, we iterate through the itemOffsetVector map.
- For each offsetVector, we get the itemIds using the offsetToItem map and use the offset itself to obtain the loss for each sample.
- Ie we only produce output for items involved in an aggregation in the final level.

Output of Back Allocated Losses with a Single Layer



```
Iterate over the ItemIndexVectorMap:
    Iterate over the Vector of Offsets
        Output the Header – look up the ItemID from the Offset using OffsetToItemMap
        Iterate over the levelLossWork table
            output the sample index and loss
```



Output of Back Allocated Losses with Multiple Levels

- For each level in turn
  - calculate the aggregations and back allocations
  - Output the losses from the losswork table as before
- Note that the losswork table is over-written each time.

NOTES

Known To Do Items

- Tidy up the code by setting up typedefs – eg for iterators.
- Remove the repeated aggregation – once for the loss calculation, once for the back allocation