Designing Indexes to Improve Query Performance: Part 1

Gail Shaw
TECHNICAL LEAD

@SQLintheWild http://sqlinthewild.co.za

Overview

- Introducing nonclustered indexes
- Common query predicates
- Indexing for equality
- Indexing for inequality
- Indexing for ORs
- Indexing for joins
- Include columns
- Filtered indexes

Introducing Nonclustered Indexes

Indexes
- Architecture, 25-32
- Clustered, 38
- Include, 67
- Nonclustered, 45
- Pages, 26, 28
- Scans, 22, 52-54
- Seeks, 20, 48-50

Nonclustered Indexes

Separate structures from the table

Multiple allowed per table

Don’t have to contain all columns

Always in sync with the table

Equality Predicates

Inequality Predicates

Predicates combined with OR

Joins

```
WHERE ClientID = 105

WHERE ClientID = 105 AND
Priority = 2

WHERE Amount > 10000

WHERE Amount > 10000 AND
TransactionType = 'D'

WHERE Mass > 100 AND Volume > 50

WHERE HasTemperatureControlled =
1 OR HasLivestock = 1

FROM Shipments s INNER JOIN
ShipmentDetails sd ON
s.ShipmentID = sd.ShipmentID
```

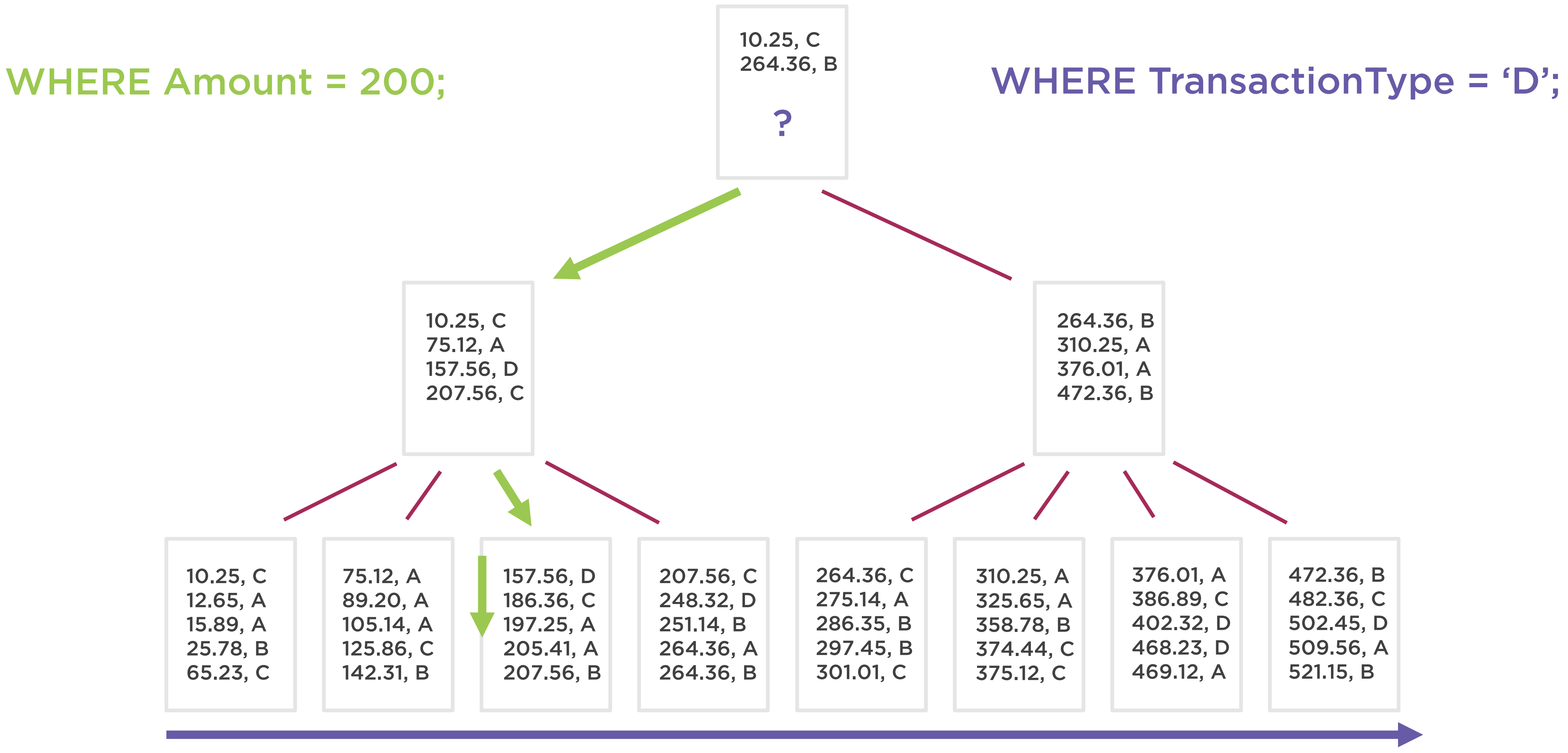
Indexing Rules for Equalities

Query must filter on a left-based subset of the index key

Order of index columns doesn’t matter for a single query

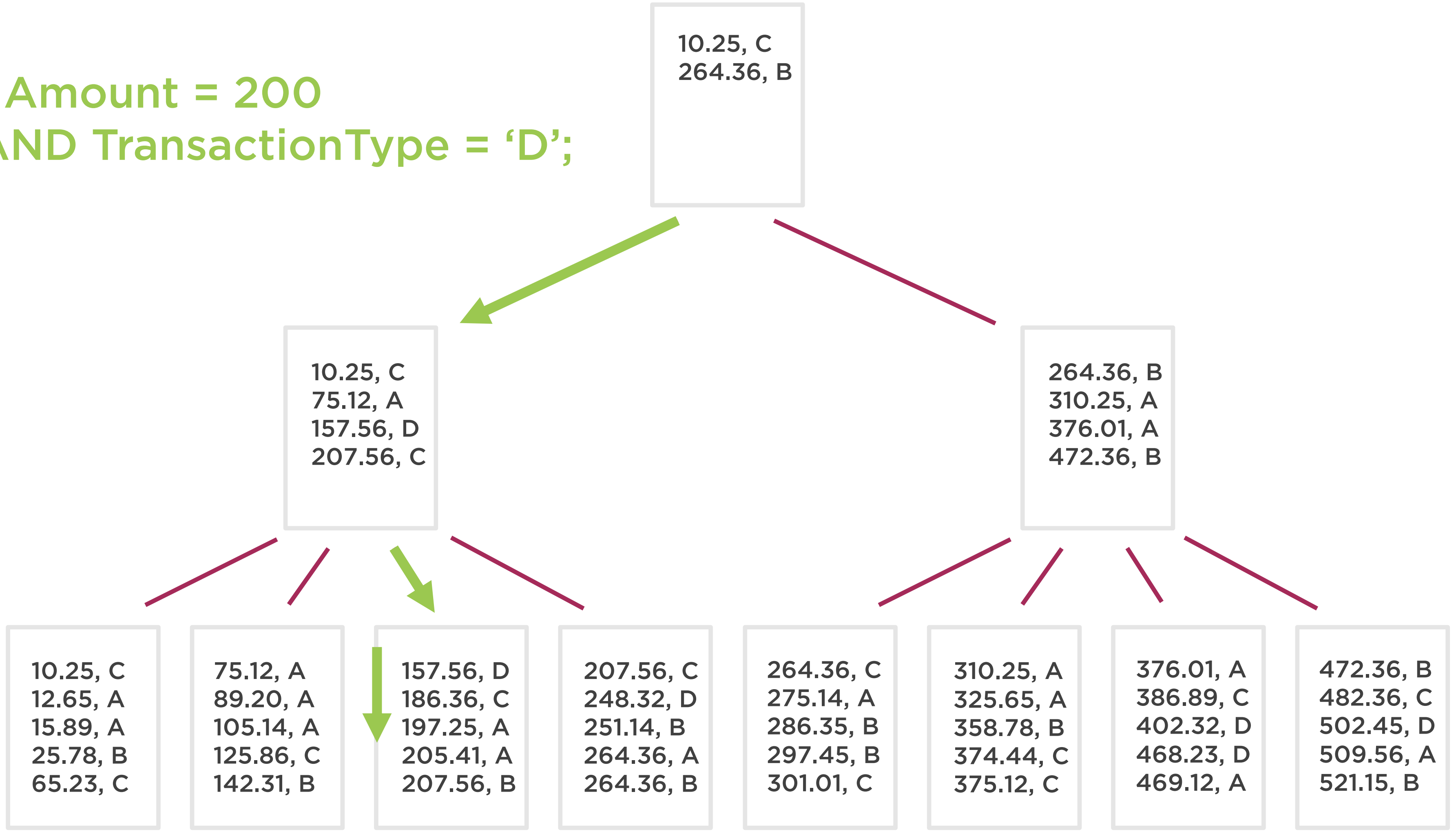
Order does matter when trying to get multiple queries to use a single index

Left-based Subset of the Index Key



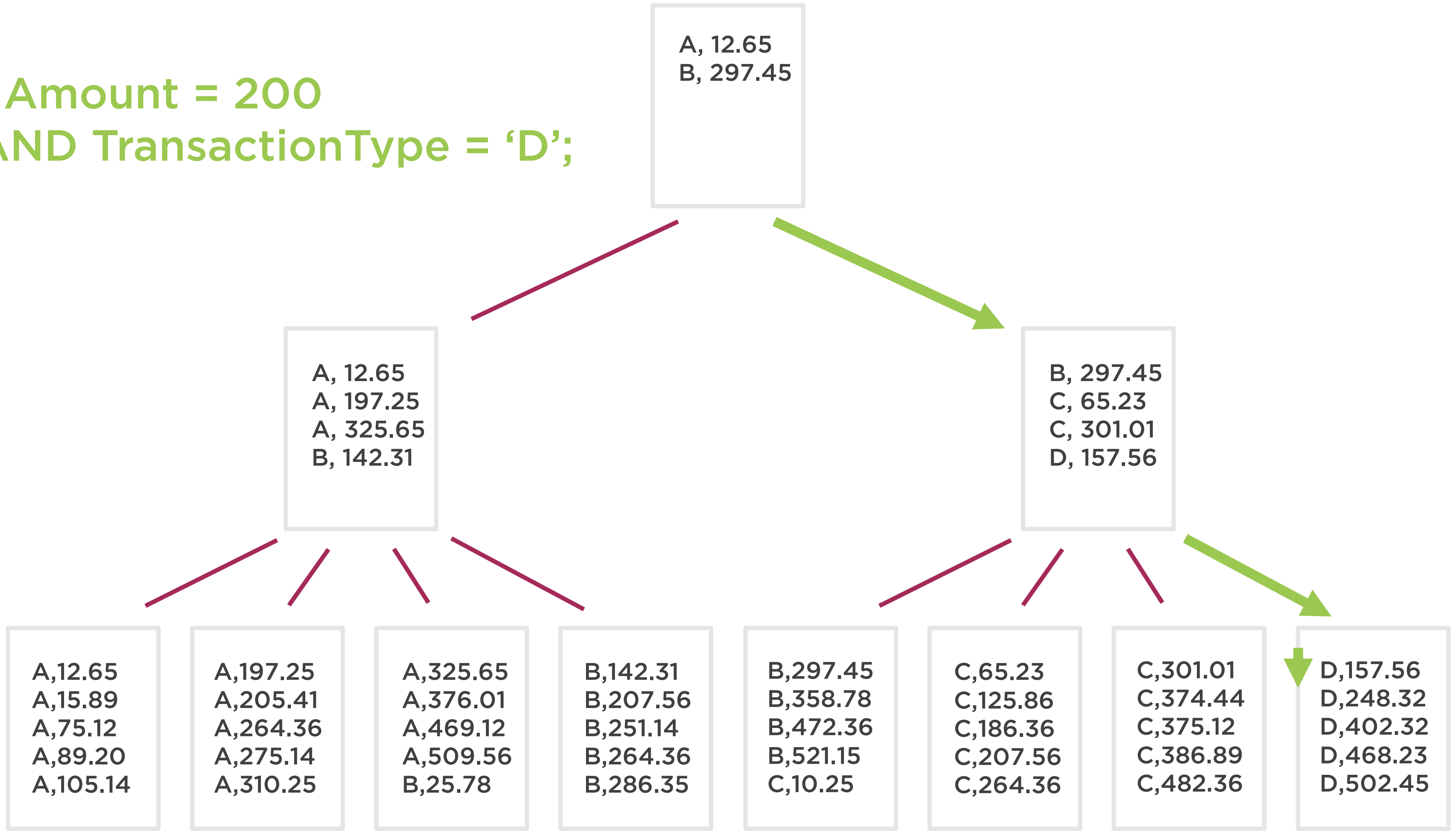
Order of Index Columns

WHERE Amount = 200
AND TransactionType = ‘D’;



Order of Index Columns

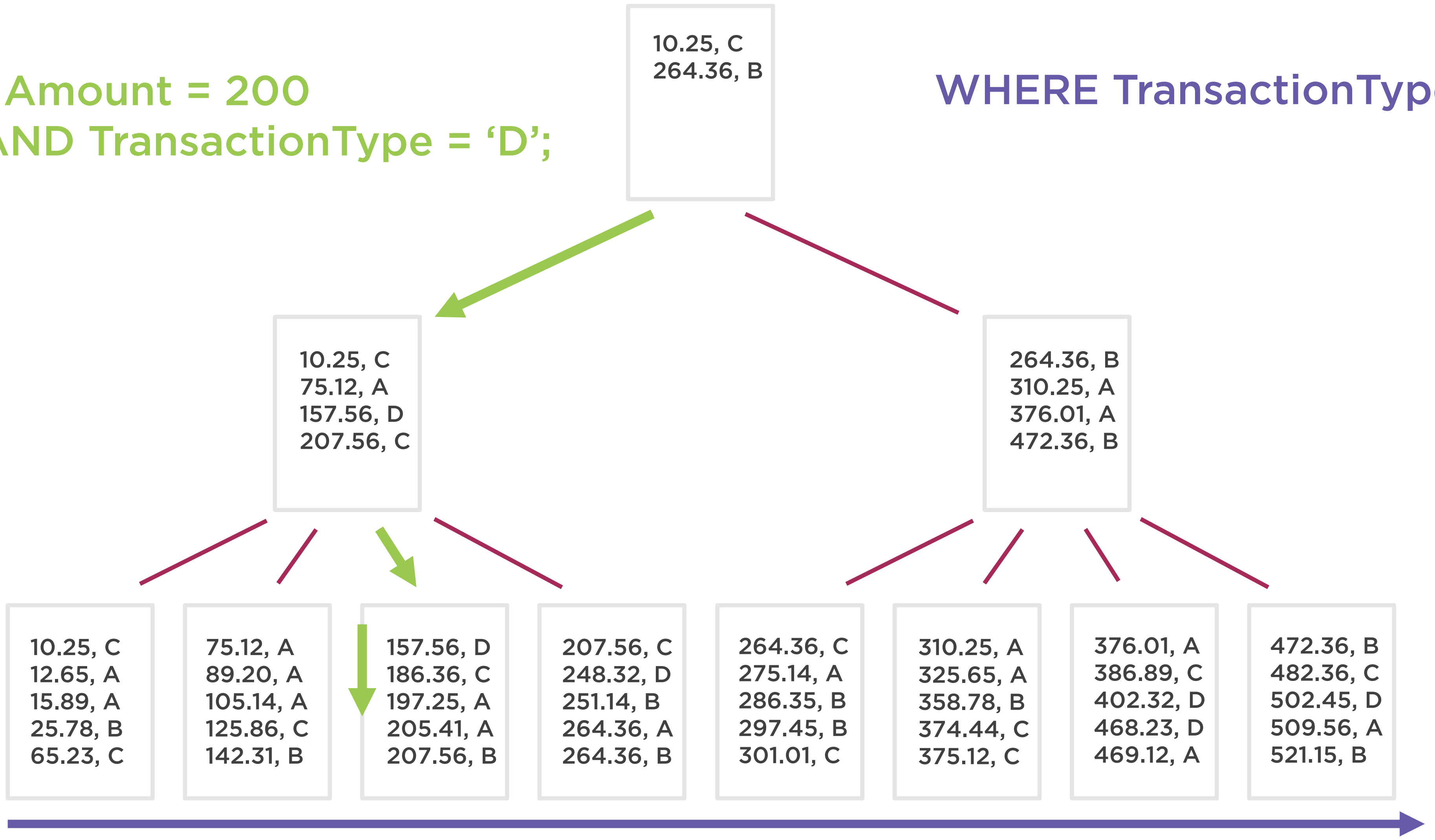
```
WHERE Amount = 200
      AND TransactionType = ‘D’;
```



Order of Index Columns

WHERE Amount = 200
AND TransactionType = ‘D’;

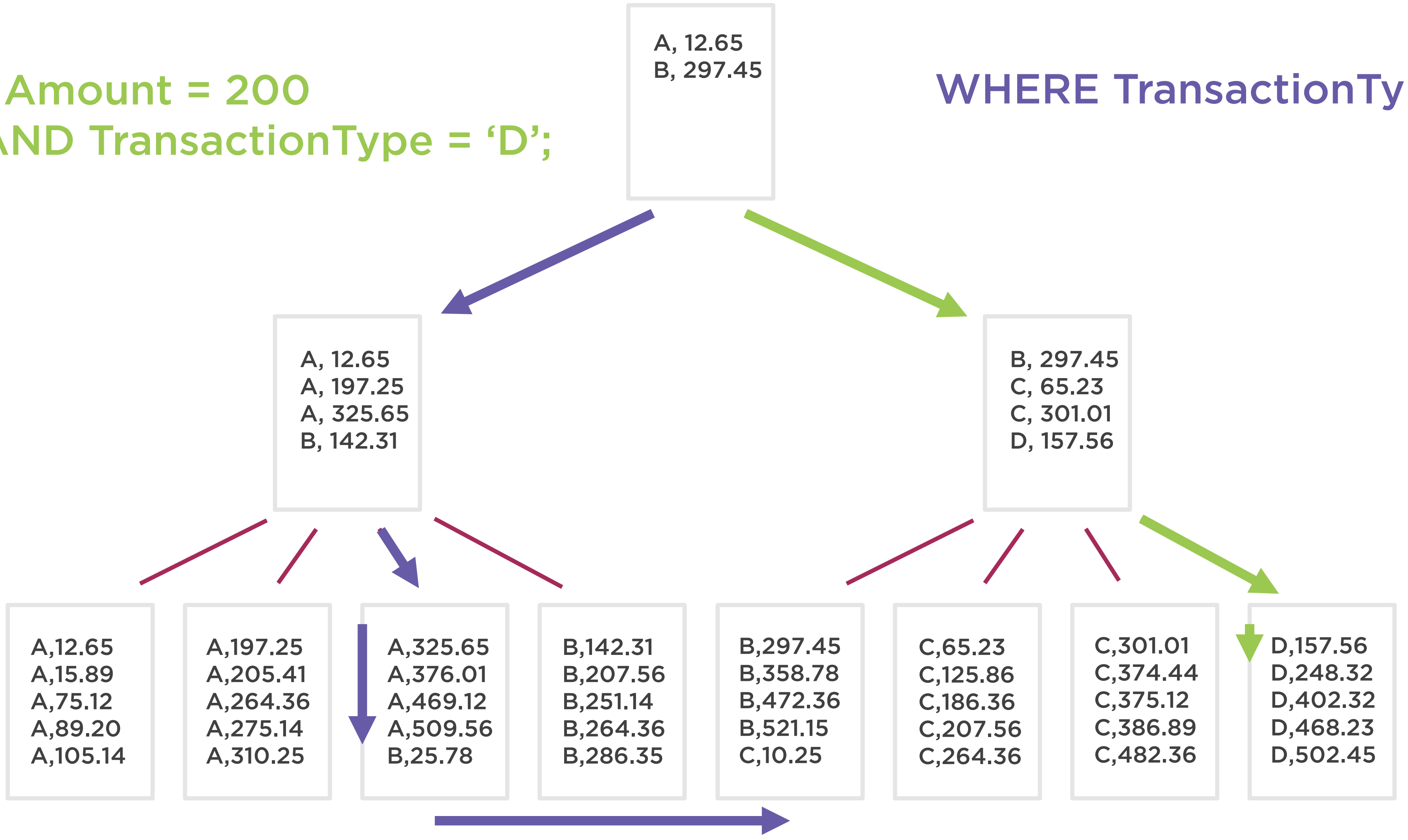
WHERE TransactionType = ‘B’;



Order of Index Columns

WHERE Amount = 200
AND TransactionType = ‘D’;

WHERE TransactionType = ‘B’;



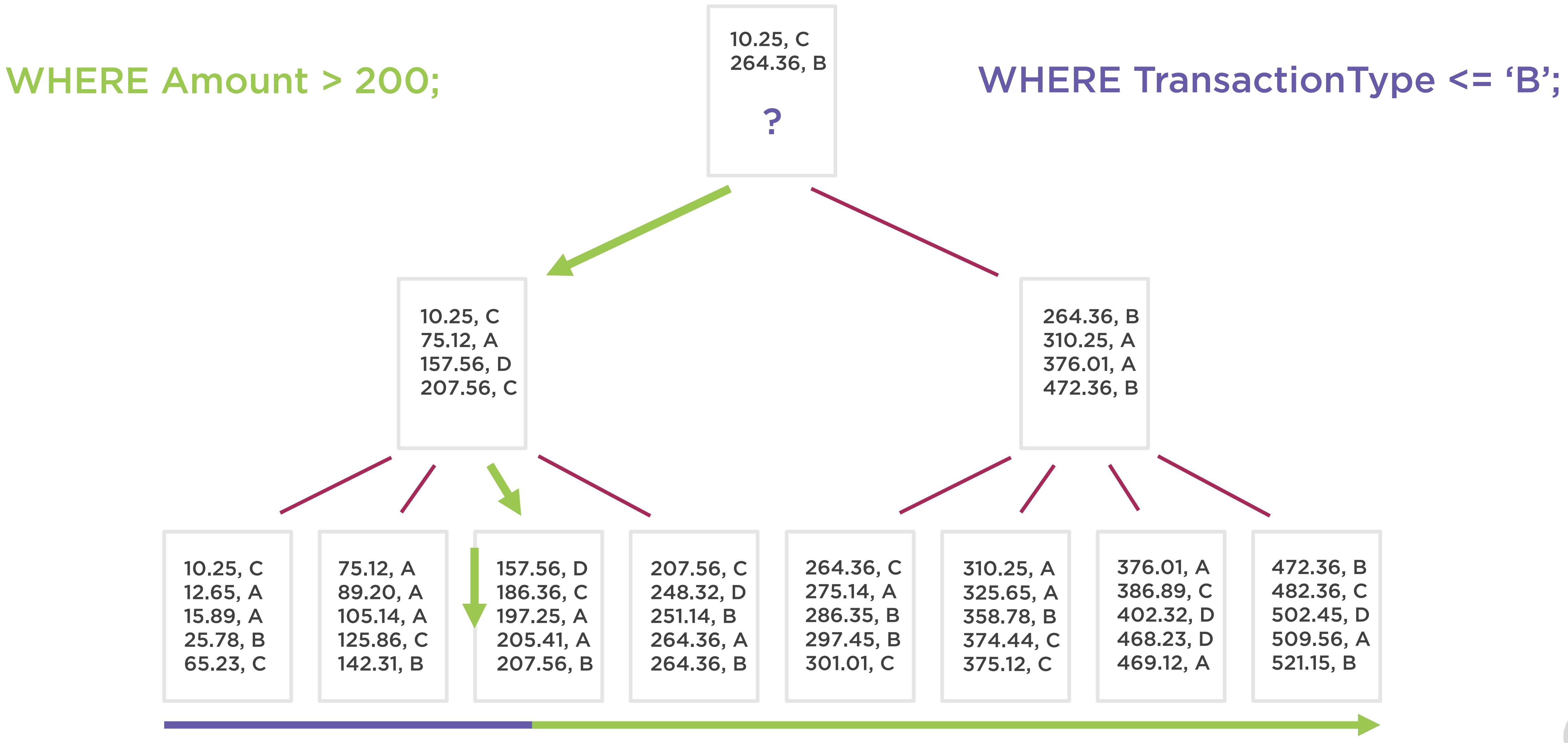
Indexing for Inequalities

Left-based subset of the index key

Equality columns before inequality columns

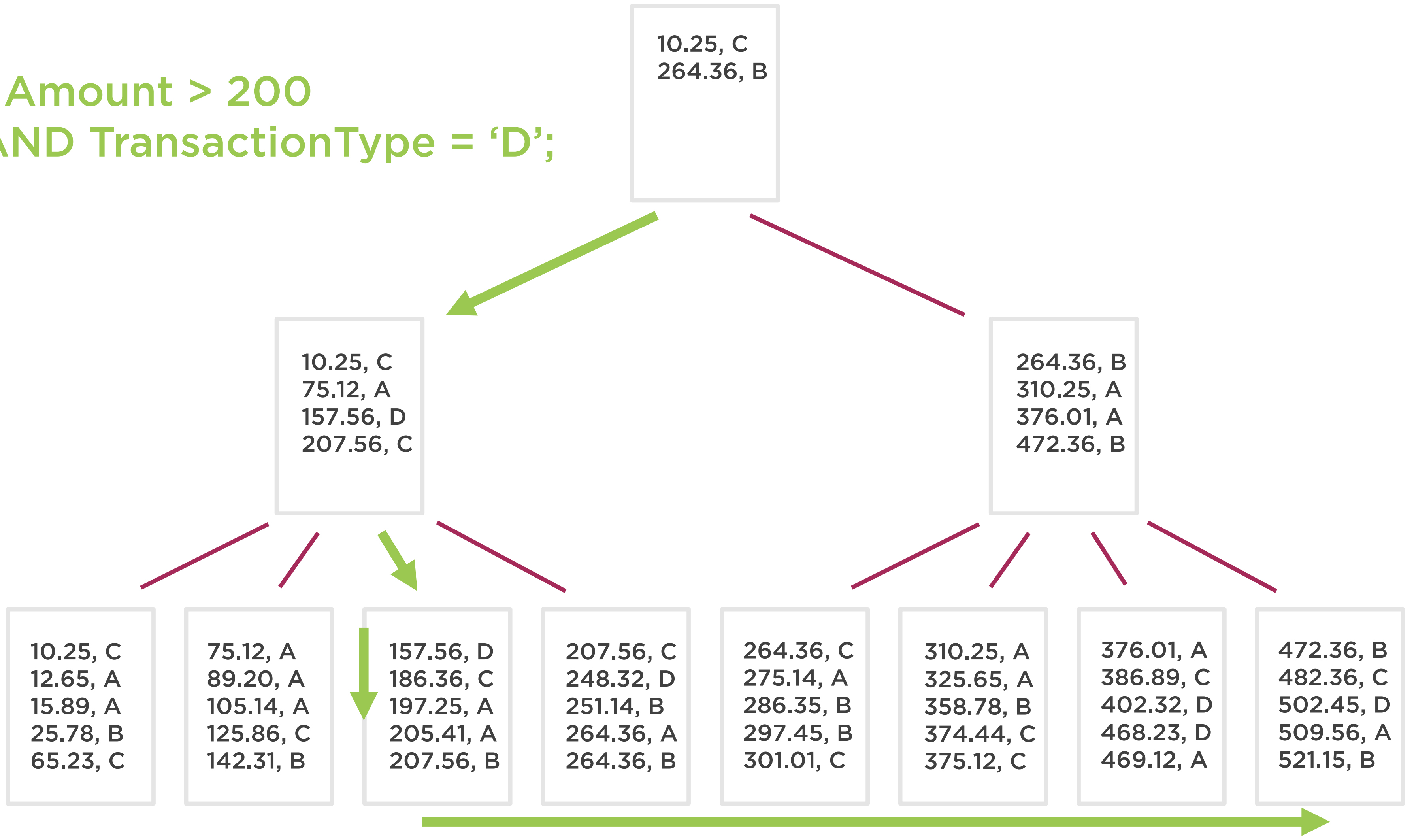
Multiple inequalities are hard to index well

Left-based Subset of the Index Key



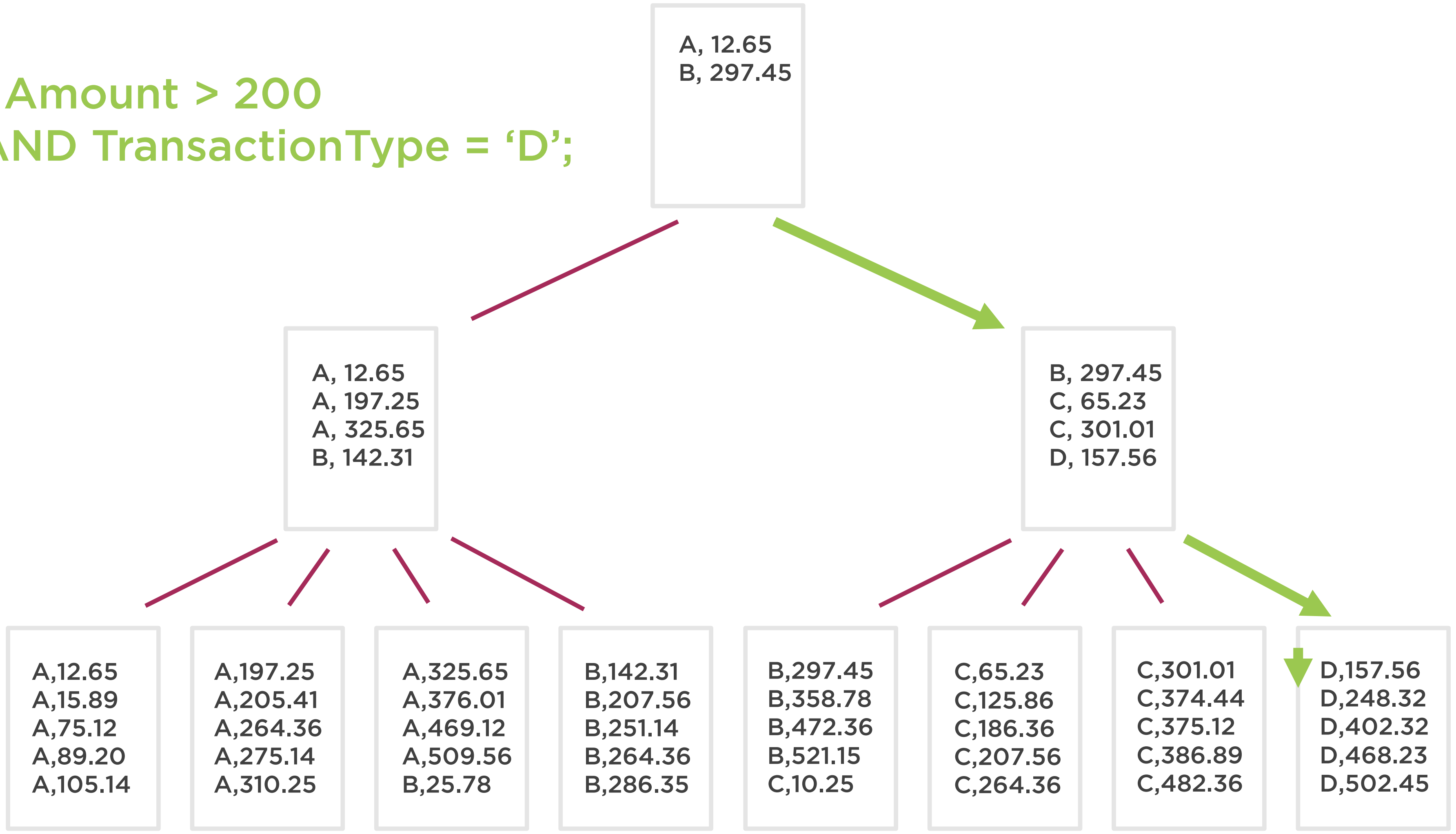
Equality Columns before Inequality Columns

```
WHERE Amount > 200
      AND TransactionType = 'D';
```



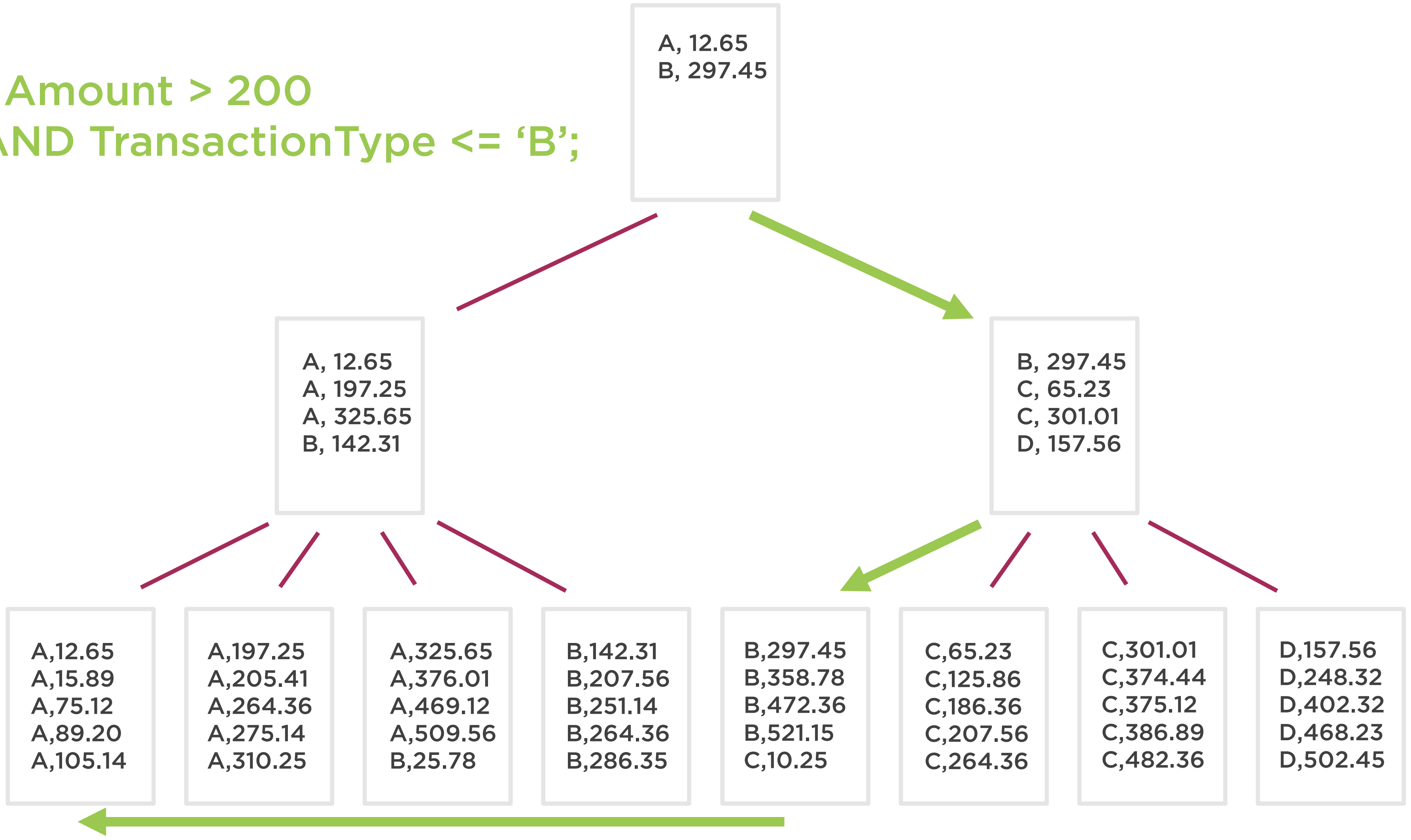
Equality Columns before Inequality Columns

```
WHERE Amount > 200
      AND TransactionType = 'D';
```



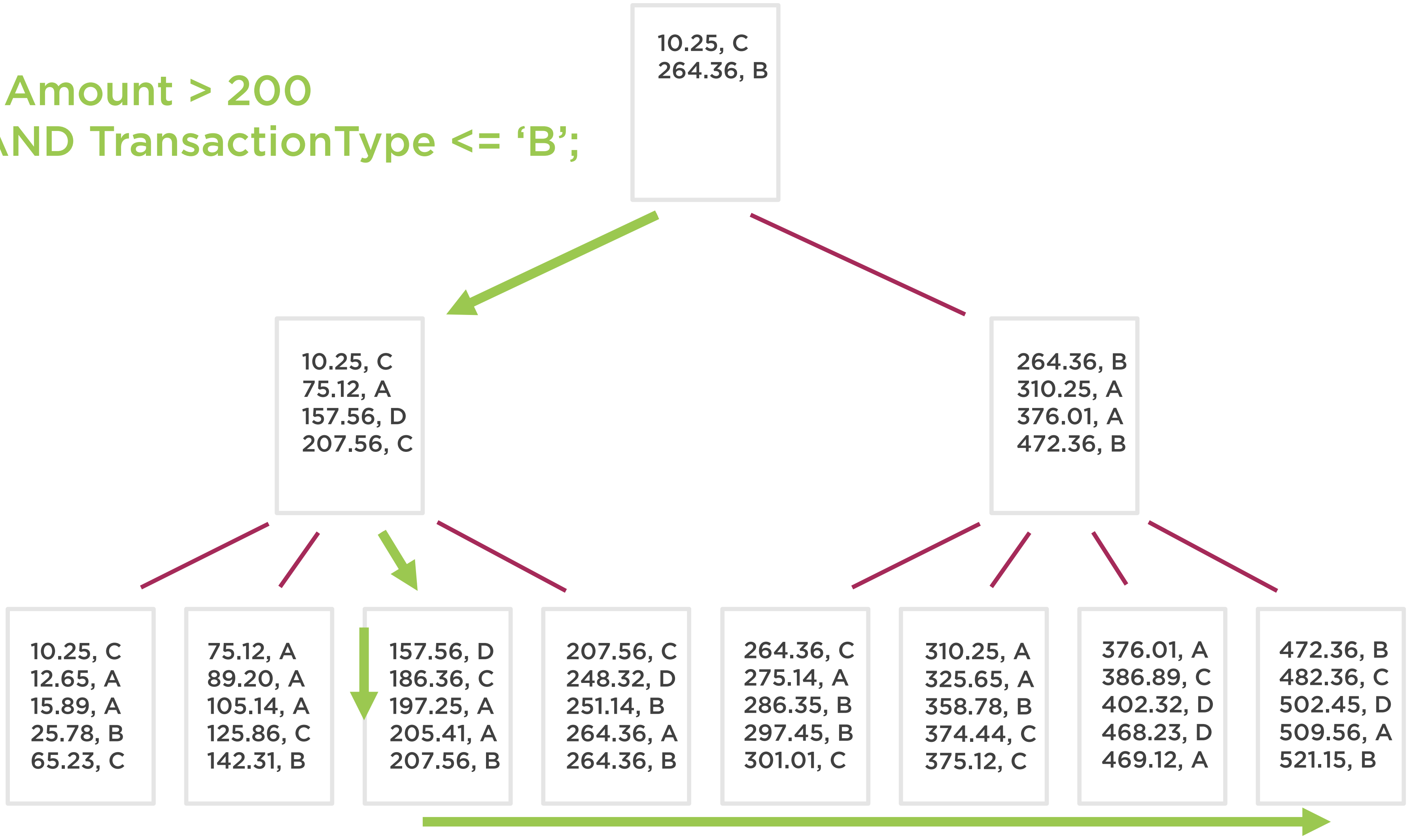
Multiple Inequalities Are Hard to Index Well

```
WHERE Amount > 200
        AND TransactionType <= 'B';
```



Multiple Inequalities Are Hard to Index Well

```
WHERE Amount > 200
      AND TransactionType <= 'B';
```



Demo

Index design with equalities and inequalities

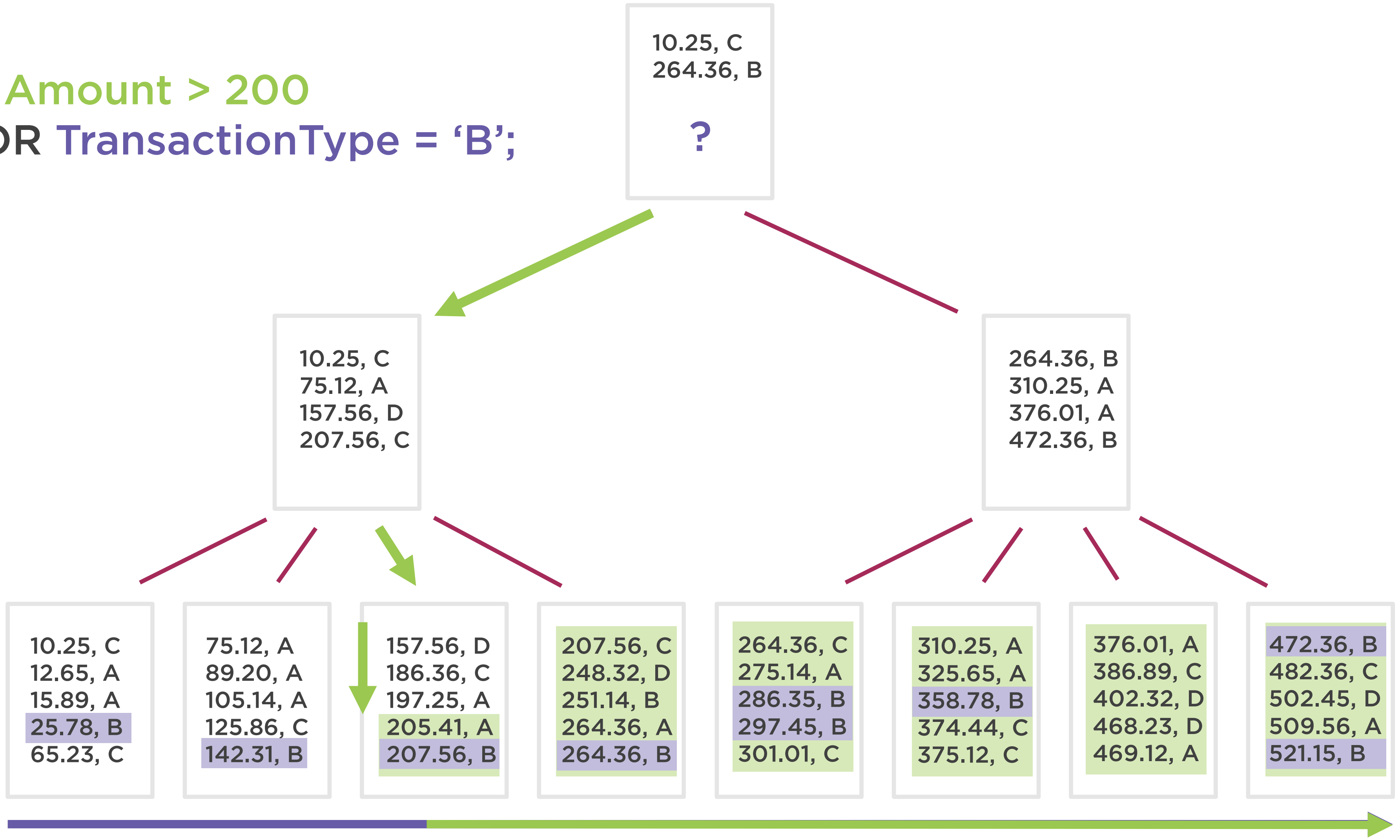
Predicates Combined with an OR

Additional predicates increase the results

Needs multiple indexes for best performance

Multiple Indexes

WHERE Amount > 200
OR TransactionType = 'B';



Demo

Index design for predicates combined with ORs

Indexing for Joins

Temp Icons! Replace in editing

Nested loop joins benefit from an index on the inner table

Merge joins may benefit from indexes to provide necessary ordering

Hash joins don’t benefit from indexes

Demo

Indexing for joins

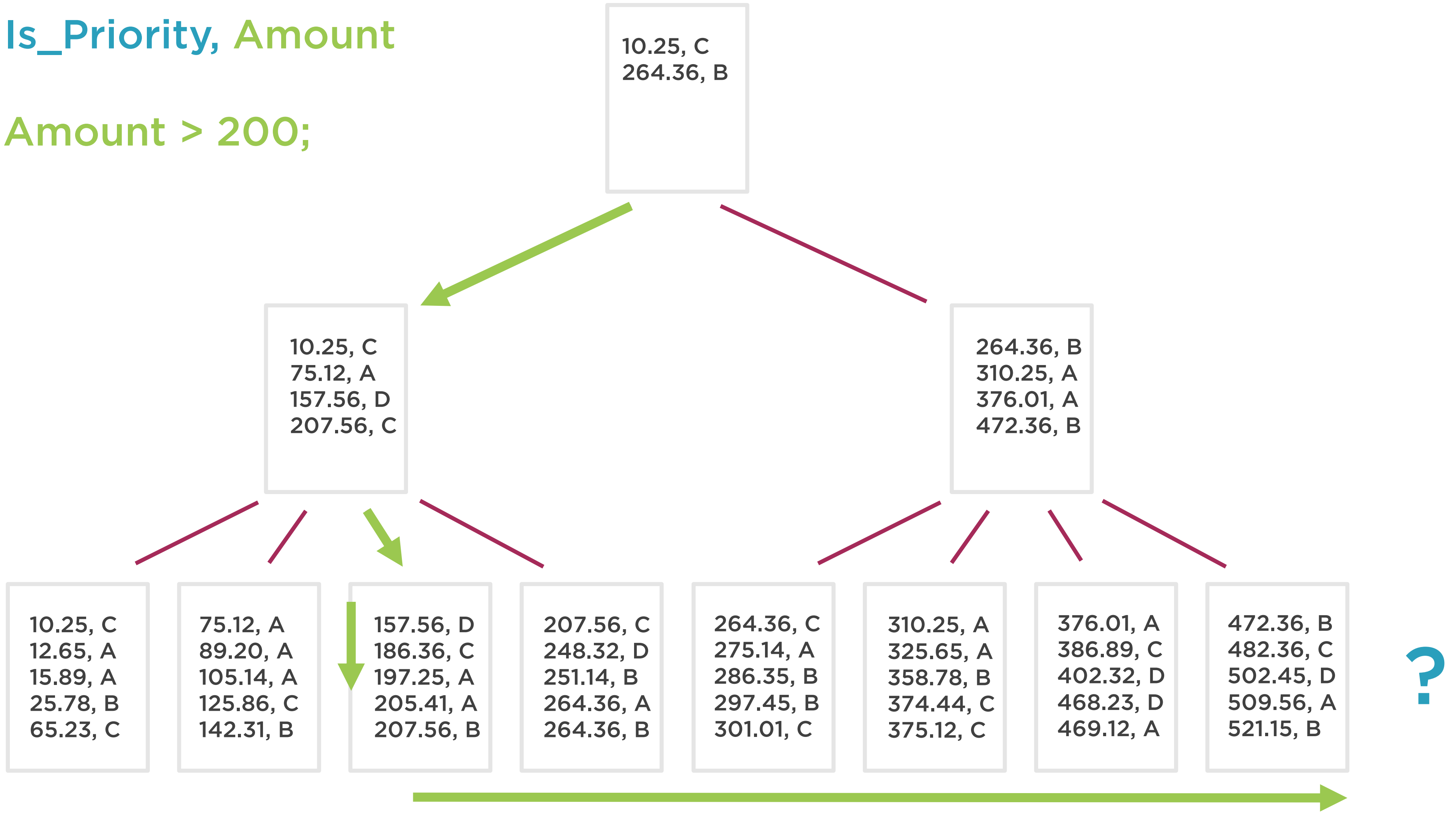
Index Include Columns

Additional columns at the leaf level of the index

Used to avoid expensive key lookups

Include Columns in an Index

```
SELECT Is_Priority, Amount
...
WHERE Amount > 200;
```



Key Lookups

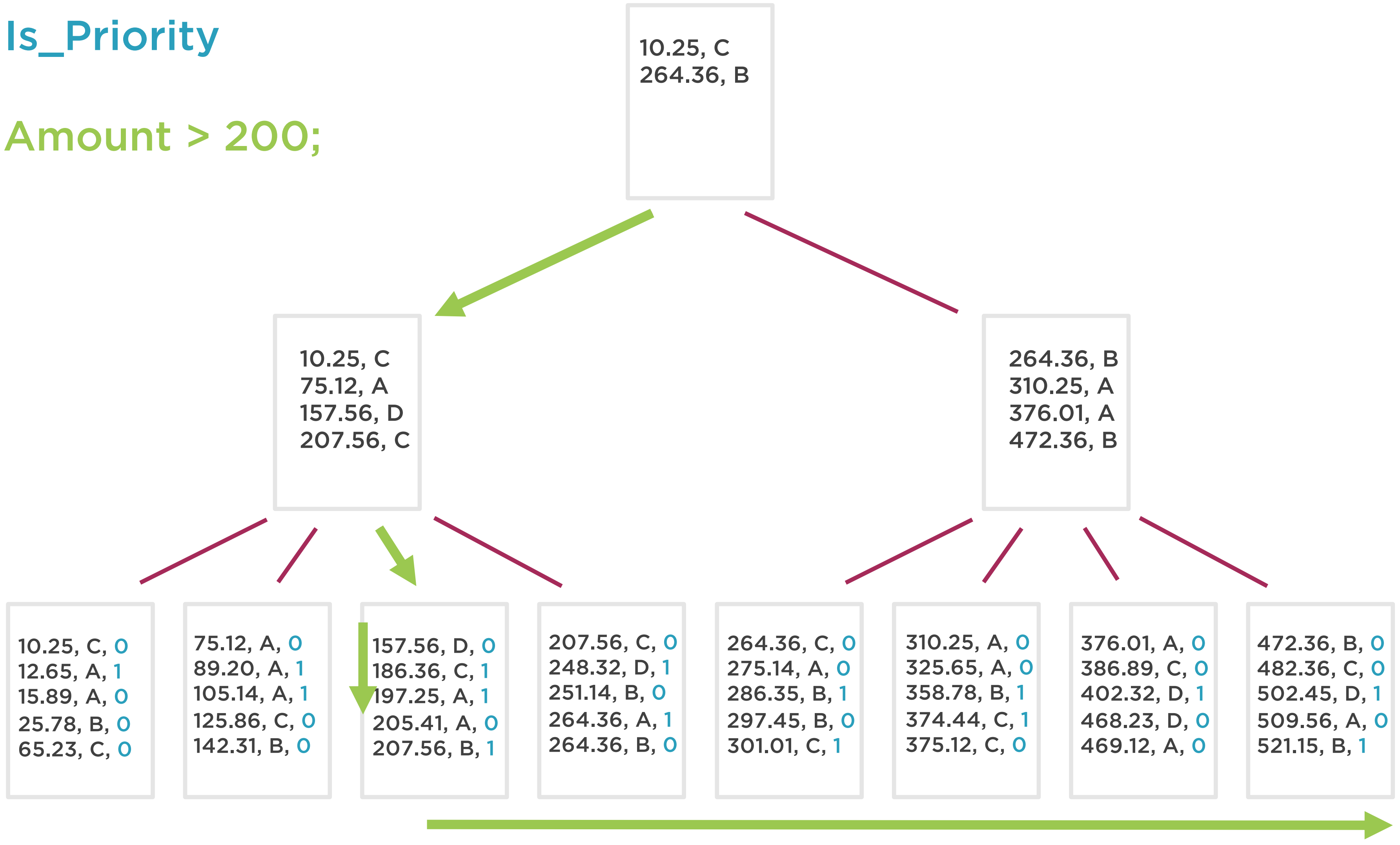
Single-row seek against the clustered index

Fetch columns which are required, but not in the key of the index used

Slow if there are a large number

Include Columns in an Index

```
SELECT Is_Priority
...
WHERE Amount > 200;
```



Demo

Include columns

Filtered Indexes

Indexes on a subset of rows in the table

Can be useful on tables with skewed data

Also useful for complex unique constraints

Don’t work with parameterised queries

Demo

Filtered indexes

How Many Nonclustered Indexes?

As many as you need for the workload

And no more

Summary

Introducing nonclustered indexes

Indexing for equality

Indexing for inequalitry

Indexing for ORs

Indexing for joins

Include columns

Filtered indexes