Need for **cost-based optimization**

Small range:
- Index search, nested loops join

Large range:
- Large scans, hash join
- Need for cost-based optimization
Integrated hashing operation

A single hash operation does both grouping and join ... and saves time.
Hash team root & member

No overflow files or I/O costs for intermediate result due to team
Index intersection

One table, four predicates, four indexes exploited
Multiple indexes covering a query

After joining two indexes of one table, all required columns are present – expensive record fetching is avoided
Nested query becomes semi-join

Exploit join algorithms designed for large inputs
Multiple optimization techniques are needed to find this plan

- Join clause inferred between line item & part supply
- Group-by list reduced by functional dependencies
- Grouping (on alternative column) pushed down through join
- “Interesting orderings” between scans, joins, grouping
Multiple optimization techniques in a hash-based plan

Same as previous example, plus

- Integrated hash operation …
- … within a hash team
- Disk-order scans
Star joins: Cartesian products

Cartesian product of two dimension tables prior to join with the fact table

Let the large fact table participate in fewer joins – reduce star join cost without any query plan hints!
Star joins: semi-join reduction

First, **join** each dimension table with an index of the fact table; then, **intersect** bookmark lists; finally, **fetch** fact table rows

All star join technologies now also in SQL Server!