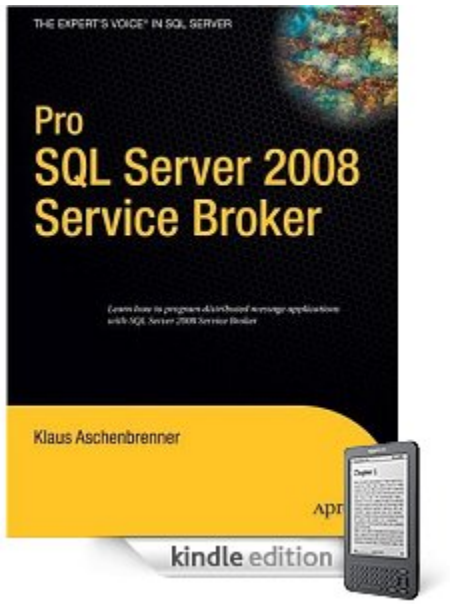
Joins in SQL Server

Klaus Aschenbrenner
Microsoft Certified Master SQL Server 2008
www.SQLpassion.at
Twitter: @Aschenbrenner

Bring your SQL Server installations to a new level of excellence!

About me

- CEO & Founder SQLpassion
- International Speaker, Blogger, Author
- SQL Server 2008 MCM
- "Pro SQL Server 2008 Service Broker"
- Twitter: @Aschenbrenner
- SQLpassion Academy
  - http://www.SQLpassion.at/academy
  - Free Newsletter, Training Videos



Agenda

- Physical Joins
- Logical Joins

Agenda

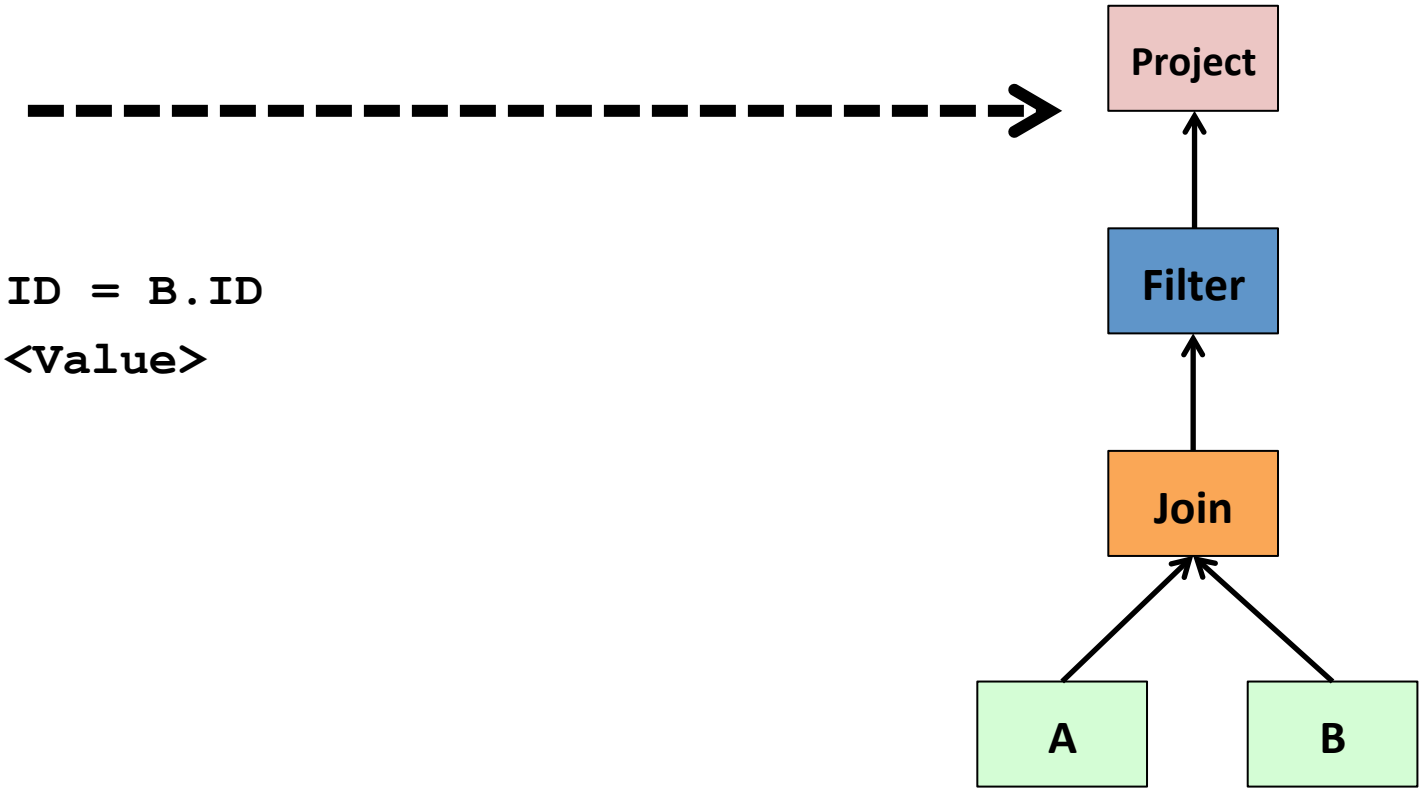
- Physical Joins
- Logical Joins

Query Trees

```
SELECT
    A.ID,
    B.ID
FROM A
JOIN B ON A.ID = B.ID
WHERE A.X = <Value>
```

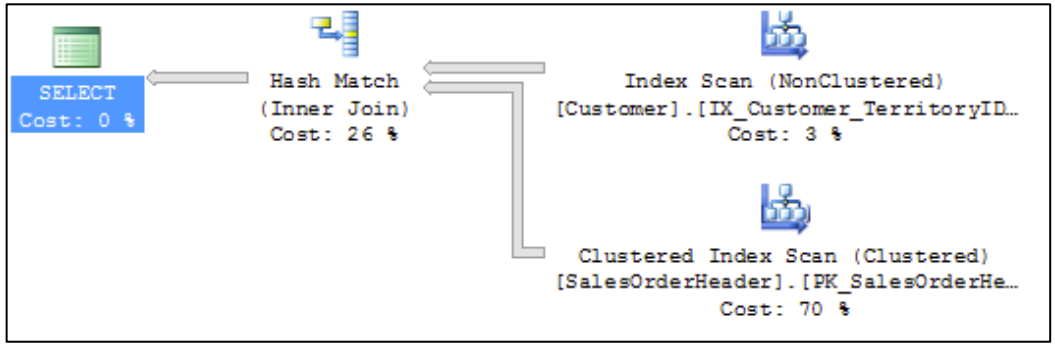
Query Trees

```
SELECT
    A.ID,
    B.ID
FROM A
JOIN B ON A.ID = B.ID
WHERE A.X = <Value>
```

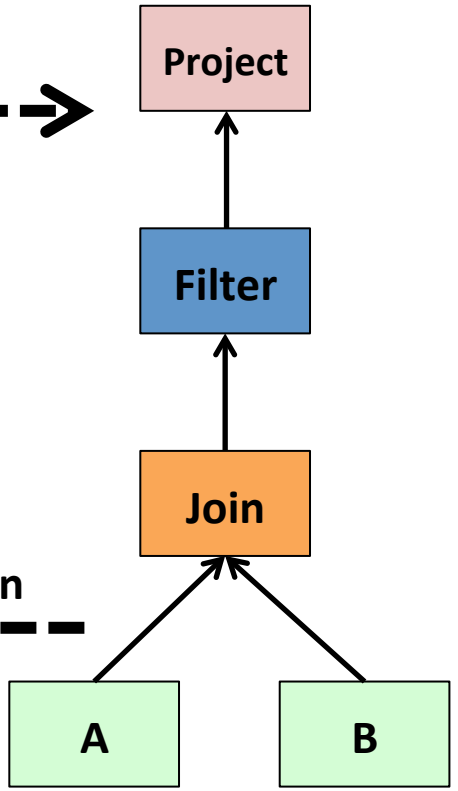


Query Trees

```
SELECT
    A.ID,
    B.ID
FROM A
JOIN B ON A.ID = B.ID
WHERE A.X = <Value>
```



Optimization



Nested Loop

- "For each Row" operator
- Takes output from one step and executes another operation "for each" output row
- Outer Loop, Inner Loop
- Only join type that supports inequality Predicates

Merge Join

- Needs at least one equijoin predicate
- Used when joined columns are indexed (sorted)
- Otherwise (expensive) sorting is needed
  - Plan may include explicit sort

Hash Join

- Needs at least one equijoin predicate
- Hashes values of join columns from one side (smaller table)
  - Based on Statistics
- Probes them with the join columns of the other side (larger table)
- Stop and Go for the Probe Phase
- Needs memory grants to build the hash table
  - If the memory grants are exceeded, the Hash Join is spilled to TempDb
  - Performance decreases!

Demo

Physical Joins

Agenda

- Physical Joins
- Logical Joins

Joins

- Operates on 2 input tables
- 3 fundamental types
  - CROSS JOIN
  - INNER JOIN
  - OUTER JOIN
- Differ in how they apply their logical query processing phases

Logical Phases

- Each Join type applies a different set of phases
  - Cartesian Product
  - Filter
  - Add Outer Rows

Cross Joins

- Simplest type of Join
- Implement only one logical query processing phase
  - Cartesian Product
  - Each row from each input is matched with the rows from each other input
  - Result: m * n rows

Demo

Cross Joins

Inner Joins

- Applies 2 logical query processing phases
  - Cartesian Product
  - Filters rows based on a predicate
- Only returns rows where the predicate returns TRUE
  - FALSE, UNKNOWN are not returned
- INNER keyword is optional
- Use the new ANSI SQL-92 syntax to avoid Cross Joins
  - ON clause is needed

Non-Equi Joins

- Join condition involves an non-equality operator
  - <
  - >
  - <=
  - >=
  - <>
- Implemented as a Nested Loop operator

Demo

Inner Joins

Outer Joins

- Apply 3 logical query processing phases
  - Cartesian Product
  - Filter rows based on a predicate
  - Adding Outer Rows
- One table is marked as „preserved“
  - **LEFT** OUTER JOIN
  - **RIGHT** OUTER JOIN
  - **FULL** OUTER JOIN
- OUTER keyword is optional
- Rows from the preserved table are added to the other table
  - With NULL values as placeholders

Demo

Outer Joins

Summary

- Physical Joins
- Logical Joins

SQL Server Performance Tuning Workshop

- June 1 – 5 in London
- Agenda
  - Database Internals
  - Execution Plans
  - Indexing
  - Statistics
  - Locking, Blocking, Deadlocking
  - Performance Troubleshooting
- More Information
  - http://www.SQLpassion.at/academy/perftuning
  - 10% Discount!