A BEGINNER'S GUIDE TO QUERY TUNING SUPERSTARDOM

TIM FORD, SQL SERVER MVP

HEALTHCARE DBA ⊙ CONSULTANT ⊙ SPEAKER ⊙ MENTOR ⊙ DESIGNER ⊙ AUTHOR

GAMING THE QUERY ENGINE

TIM FORD, SQL SERVER MVP

HEALTHCARE DBA ⊙ CONSULTANT ⊙ SPEAKER ⊙ MENTOR ⊙ DESIGNER ⊙ AUTHOR

YOU ALWAYS HURT THE ONES YOU LOVE

TIM FORD, SQL SERVER MVP

HEALTHCARE DBA ⊙ CONSULTANT ⊙ SPEAKER ⊙ MENTOR ⊙ DESIGNER ⊙ AUTHOR

QUERIES: THE GOOD, THE BAD, AND THE UGLY

TIM FORD, SQL SERVER MVP

HEALTHCARE DBA ⊙ CONSULTANT ⊙ SPEAKER ⊙ MENTOR ⊙ DESIGNER ⊙ AUTHOR

HELPING PEOPLE STUCK WITH THE IMPORTANT, YET NON-GLAMOUROUS TASK OF GETTING INFORMATION OUT OF SQL SERVER AND INTO A BEAUTIFUL REPORT WITH PLENTY OF COLORFUL PICTURES (SO LONG AS THEY'RE NOT PIE CHARTS) DO SO WITHOUT HARMING THEMSELVES OR OTHERS.

TIM FORD, SQL SERVER MVP

HEALTHCARE DBA ⊙ CONSULTANT ⊙ SPEAKER ⊙ MENTOR ⊙ DESIGNER ⊙ AUTHOR

RE: me

EMAIL :	tim@thesqlagentman.com
TWITER :	@sqlagentman -&- @sqlcruise
CV :	linkedin.com/in/timothyford

SQL CRUISE EVENTS

SQLAGENTMAN CONSULTANCY

MICROSOFT SQL SERVER MVP (SINCE 2009)

THESQLAGENTMAN.COM

PROFESSIONAL ASSOCIATION FOR SQL SERVER DIRECTOR

AUTHOR

Fully.Qualified.Object.Names

SELECT *

JOINs

Sargability

DISTINCT Values

BONUS SPECTRUM DEMO EDITION EXTRAS		
Stored Procedures	Execution Plans	Statistics IO

RE: GOALS

Universal Happiness

SCHEMA

Fully Qualified Object Names

server_name.database.schema_name.object_name

database.schema.object_name

SELECT * TMI – Too Much Information

SELECT *

SELECT *STAR*

SELECT ~~*SUPERSTAR*~~

SELECT * TMI – Too Much Information

WRONG

```
SELECT *
FROM dbo.wait_history_int
WHERE wait_id = 3298
    AND running_pct < 80
ORDER BY running_pct ASC
OPTION (RECOMPILE);
```

SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 109 ms.
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 1 ms.

RIGHT

```
SELECT  date_stamp
    , wait_type
    , resource_wait_time_s
    , avg_resource_wait_time_ms
    , pct
    , running_pct
FROM dbo.wait_history_int
WHERE wait_id = 3298
    AND running_pct < 80
ORDER BY running_pct ASC
OPTION (RECOMPILE);
```

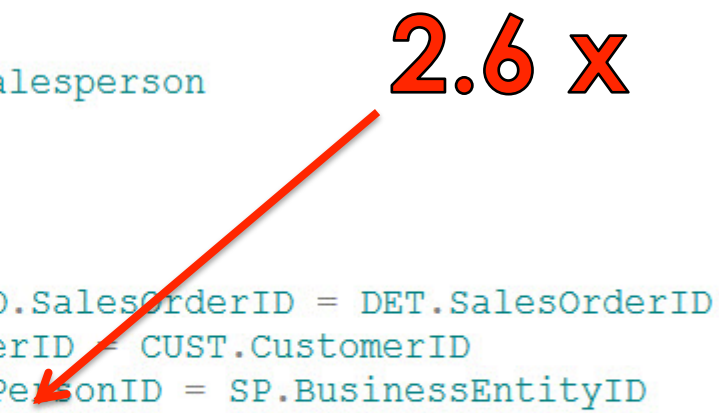
SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 46 ms.
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

JOINs

Too Much of a Good Thing Is…

Spot The Issue…

```
SELECT HEAD.SalesOrderID
     , CUST.PersonID AS CustomerID
     , PERS.LastName + ', ' + PERS.FirstName AS salesperson
     , DET.OrderQty
     , PRO.ProductNumber
     , DET.LineTotal
FROM sales.SalesOrderHeader HEAD
     INNER JOIN sales.SalesOrderDetail DET ON HEAD.SalesOrderID = DET.SalesOrderID
     LEFT JOIN Sales.Customer CUST ON HEAD.CustomerID = CUST.CustomerID
     LEFT JOIN Sales.SalesPerson SP ON HEAD.SalesPersonID = SP.BusinessEntityID
--   LEFT JOIN Sales.SalesTerritory ST ON SP.TerritoryID = ST.TerritoryID
     LEFT JOIN Person.Person PERS ON SP.BusinessEntityID = PERS.BusinessEntityID
     LEFT JOIN Production.Product PRO ON DET.ProductID = PRO.ProductID
WHERE HEAD.CustomerID = 29825
OPTION(RECOMPILE);
```



SEEKS

It's All About SARGABILITY!

SEEKS

Sargability: Seeks Good; Scans Bad

Image source: http://www.earth911.com/content/uploads/2011/06/Phone-Books.jpg used under Creative Commons licensing.

SEEKS

Operators & Sargability

SARGABLE	SARGABLE-*ISH*	NOT-SARGABLE
=>	<>	LIKE '%---'
<	IN	
>=	OR	
<=	NOT IN	NOT LIKE '%---'
BETWEEN	NOT EXISTS	
LIKE '---%'	NOT LIKE '---%'	
EXISTS		

SEEKS

Functions != Sargability

SELECT ... WHERE Year(date) = 1986	NOT-SARGABLE
SELECT ... WHERE date >= '01-01-1986' AND date < '01-01-1987'	SARGABLE
Select ... WHERE isNull(LastName, 'Jonze') = 'Jonze'	NOT-SARGABLE
Select ... WHERE ((LastName = 'Jonze') OR (LastName IS NULL))	SARGABLE
Select ... WHERE SUBSTRING(StateName,2) = 'Mi'	NOT-SARGABLE
Select ... WHERE StateName Like 'Mi%'	SARGABLE
Select ... WHERE DateDiff(mm,PartyDate,GetDate()) >= 20	NOT-SARGABLE
Select ... WHERE PartyDate < DateAdd(mm,-20,GetDate())	SARGABLE

DISTINCT The Expensive, The Sloppy and the Lazy

DISTINCT & User-Defined Functions

DISTINCT & 1-to-Many Joins

DISTINCT to Hide Bad/Lazy Joins

DISTINCT

DISTINCT & User-Defined Functions

WRONG

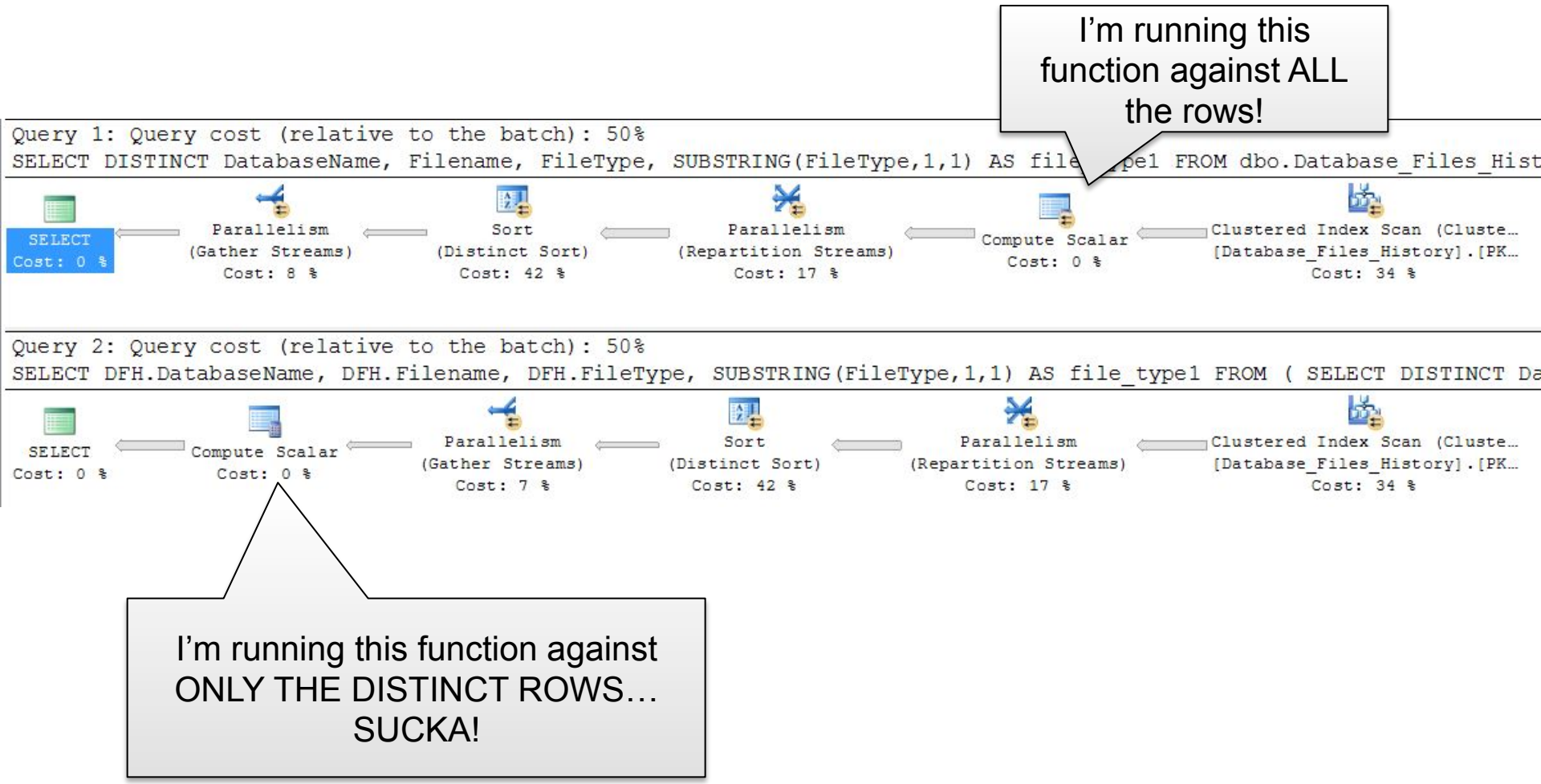
```
SELECT DISTINCT DatabaseName
     , Filename
     , FileType
     , SUBSTRING(FileType,1,1) AS file_type1
FROM dbo.Database_Files_History;

-----------------------------------------------------------
```

RIGHT

```
SELECT DFH.DatabaseName
     , DFH.Filename
     , DFH.FileType
     , SUBSTRING(FileType,1,1) AS file_type1
FROM
     (
          SELECT DISTINCT DatabaseName, Filename, FileType
          FROM dbo.Database_Files_History
     ) AS DFH;
```

DISTINCT & User-Defined Functions



DISTINCT & User-Defined Functions

WRONG

(3226 row(s) affected)

SQL Server Execution Times:
CPU time = 171 ms, elapsed time = 330 ms.

RIGHT

(3226 row(s) affected)

SQL Server Execution Times:
CPU time = 78 ms, elapsed time = 149 ms.

DISTINCT

DISTINCT & Joins

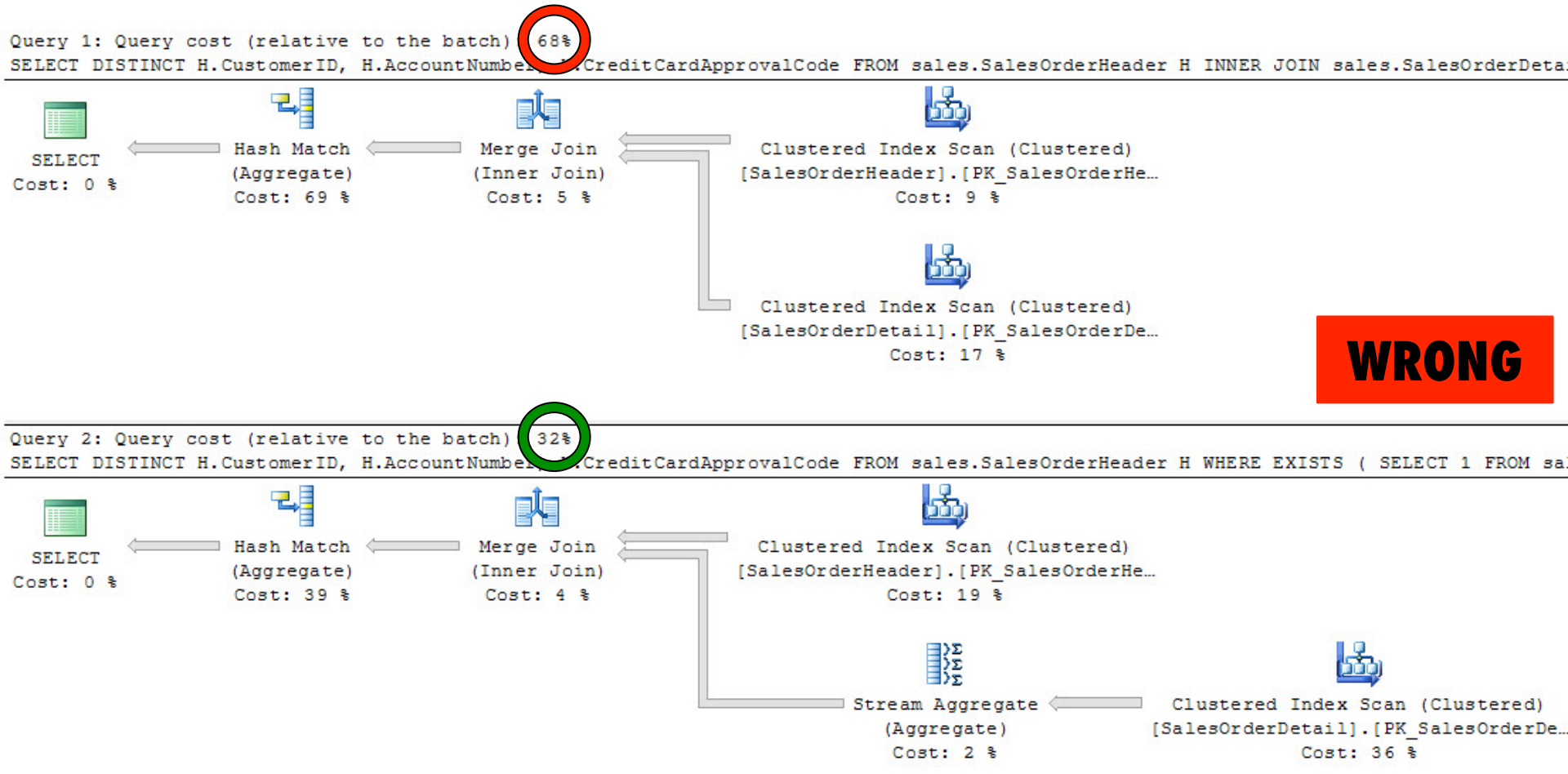
WRONG

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales.SalesOrderHeader H
    INNER JOIN sales.SalesOrderDetail D
        ON H.SalesOrderID = D.SalesOrderID
OPTION (RECOMPILE);
```

RIGHT

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales.SalesOrderHeader H
WHERE EXISTS
    (
        SELECT 1
        FROM sales.SalesOrderDetail D
        WHERE H.SalesOrderID = D.SalesOrderID
    )
OPTION (RECOMPILE);
```

DISTINCT & Joins



DISTINCT & Joins

WRONG

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales.SalesOrderHeader H
    INNER JOIN sales.SalesOrderDetail D
        ON H.SalesOrderID = D.SalesOrderID
OPTION (RECOMPILE);
```

SQL Server Execution Times:
CPU time = 125 ms, elapsed time = 387 ms.
SQL Server parse and compile time:
CPU time = 8 ms, elapsed time = 8 ms.

RIGHT

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales.SalesOrderHeader H
WHERE EXISTS
    (
        SELECT *
        FROM sales.SalesOrderDetail D
        WHERE H.SalesOrderID = D.SalesOrderID
    )
OPTION (RECOMPILE);
```

SQL Server Execution Times:
CPU time = 63 ms, elapsed time = 271 ms.
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

DISTINCT

DISTINCT & Sheer Laziness/Bad Joins

WRONG

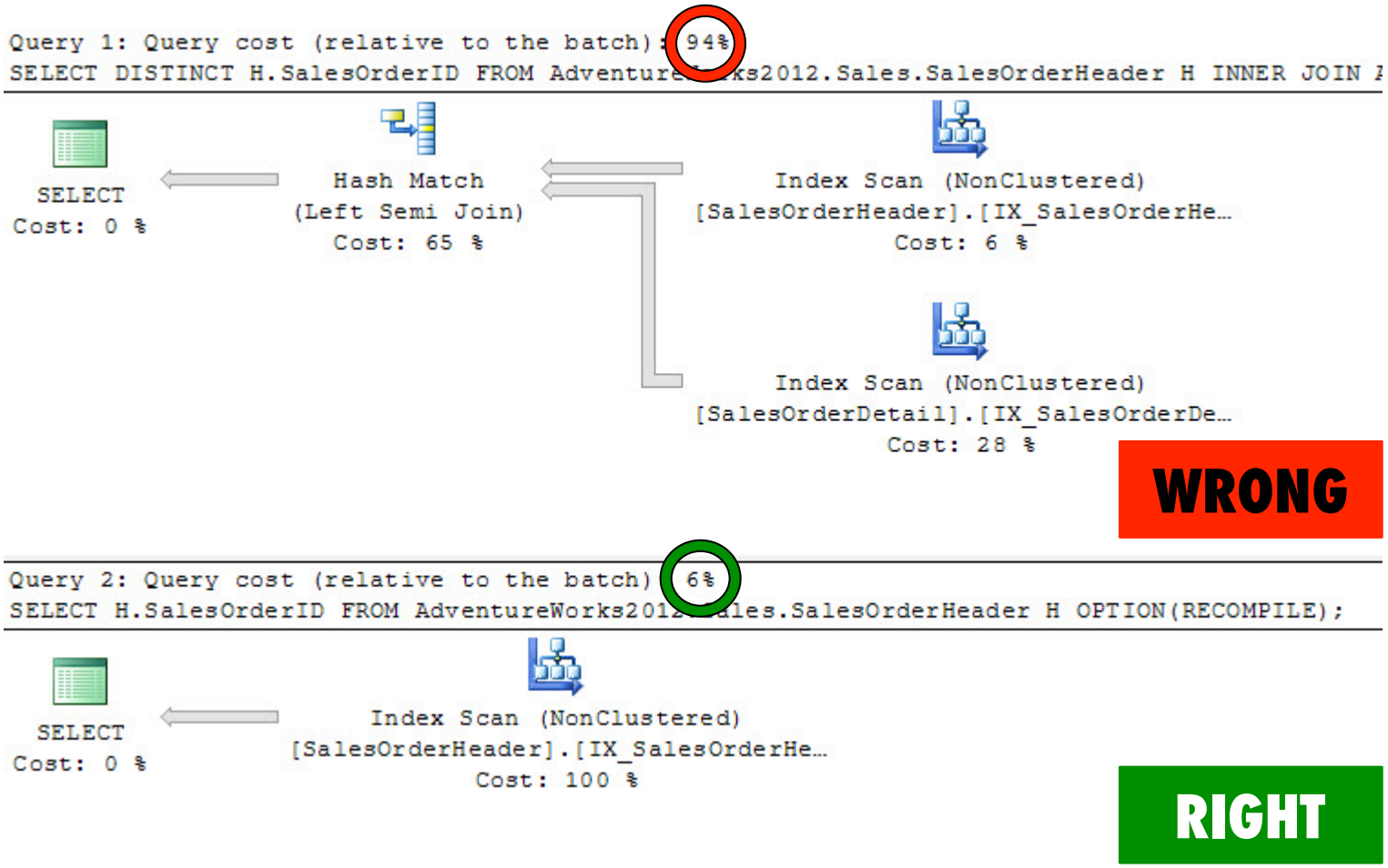
```
SELECT DISTINCT
    H.SalesOrderID
FROM AdventureWorks2012.Sales.SalesOrderHeader H
    INNER JOIN AdventureWorks2012.Sales.SalesOrderDetail D
        ON H.SalesOrderID = D.SalesOrderID
OPTION(RECOMPILE);
```

RIGHT

```
SELECT H.SalesOrderID
FROM AdventureWorks2012.Sales.SalesOrderHeader H
OPTION(RECOMPILE);
```

DISTINCT

DISTINCT & Sheer Laziness/Bad Joins



DISTINCT

DISTINCT & Sheer Laziness/Bad Joins

(31465 row(s) affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesOrderDetail'. Scan count 1, logical reads 339, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesOrderHeader'. Scan count 1, logical reads 70, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
CPU time = 47 ms, elapsed time = 241 ms.
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

WRONG

(31465 row(s) affected)
Table 'SalesOrderHeader'. Scan count 1, logical reads 70, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
CPU time = 15 ms, elapsed time = 138 ms.
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

RIGHT

Stored Procedures

Execution Plans

STATISTICS IO / TIME

RE: Q & A

RE: links

SLIDE DECK :	thesqlagentman.com/go/presentations/
BLOG :	thesqlagentman.com
CONSULTING :	tim@thesqlagentman.com
SQL CRUISE EVENTS :	sqlcruise.com