

SQLServerFast.com

Execution Plan Video Training

Block 1: Understanding execution plans

Level: Basic

Chapter 1: What and why?

Programming paradigms

SQL – Structured Query Language

Fourth generation language

Also known as “declarative language”

Paradigm shift from third generation languages

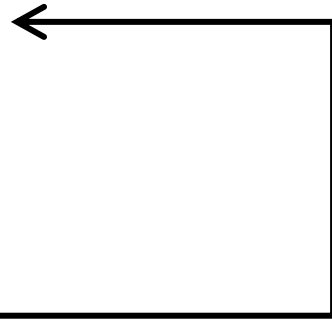
My morning routine (a

1. Switch off alarm
2. Get out of bed
3. Take a shower
4. Dry off
5. Get dressed
6. Brush teeth
7. Eat breakfast
8. Head to office



My morning routine (as I see it)

1. Switch off alarm
2. Get out of bed
3. Take a shower
4. Dry off
5. Get dressed
6. Brush teeth
7. Eat breakfast
8. Head to office

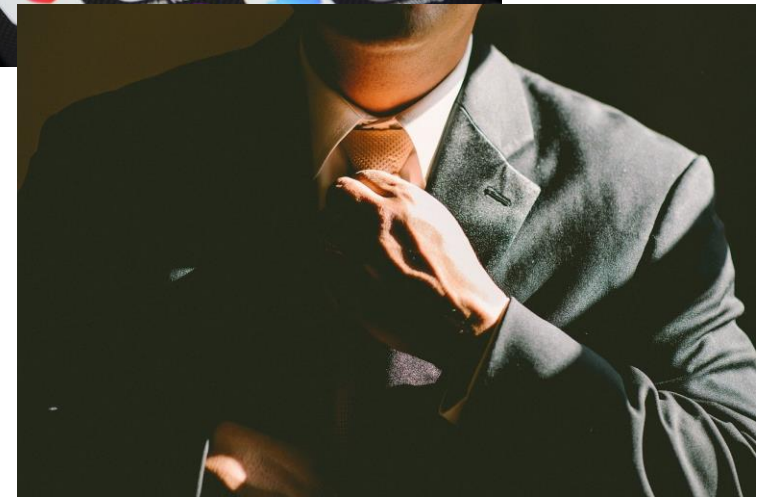


My morning routine (as I see it)

1. Switch off alarm
2. Get out of bed
3. Take a shower
4. Dry off
5. Get dressed
6. Brush teeth
7. Eat breakfast
8. Head to office



My morning routine (as my boss sees it)



My morning routine (as I see it)

1. Switch off alarm
2. Get out of bed
3. Take a shower
4. Dry off
5. Get dressed
6. Brush teeth
7. Eat breakfast
8. Head to office

Third generation programming language

1. Switch off alarm
2. Get out of bed
3. Take a shower
4. Dry off
5. Get dressed
6. Brush teeth
7. Eat breakfast
8. Head to office

```
1. public class LeapYear {
2.
3.     public static void main(String[] args) {
4.
5.         int year = 1900;
6.         boolean leap = false;
7.
8.         if(year % 4 == 0)
9.         {
10.            if( year % 100 == 0)
11.            {
12.                // year is divisible by 400, hence the year is a leap year
13.                if ( year % 400 == 0)
14.                    leap = true;
15.                else
16.                    leap = false;
17.            }
18.            else
19.                leap = true;
20.        }
21.        else
22.            leap = false;
23.
24.        if(leap)
25.            System.out.println(year + " is a leap year.");
26.        else
27.            System.out.println(year + " is not a leap year.");
28.    }
29. }
```

Third generation programming language

1. Assume no leap year
2. If divisible by 4:
 leap year
3. If divisible by 100:
 no leap year
4. If divisible by 400:
 leap year
5. Show result

```
1. public class LeapYear {  
2.  
3.     public static void main(String[] args) {  
4.  
5.         int year = 1900;  
6.         boolean leap = false;  
7.  
8.         if(year % 4 == 0)  
9.         {  
10.            if( year % 100 == 0)  
11.            {  
12.                // year is divisible by 400, hence the year is a leap year  
13.                if ( year % 400 == 0)  
14.                    leap = true;  
15.                else  
16.                    leap = false;  
17.            }  
18.            else  
19.                leap = true;  
20.        }  
21.        else  
22.            leap = false;  
23.  
24.        if(leap)  
25.            System.out.println(year + " is a leap year.");  
26.        else  
27.            System.out.println(year + " is not a leap year.");  
28.    }  
29. }
```

Third generation programming language

Algorithm

Method for solving a task
Often multiple choices

Developer chooses

Speed
Size of code
Ease of maintenance
Other reasons

```
1. public class LeapYear {  
2.  
3.     public static void main(String[] args) {  
4.  
5.         int year = 1900;  
6.         boolean leap = false;  
7.  
8.         if(year % 4 == 0)  
9.         {  
10.            if( year % 100 == 0)  
11.            {  
12.                // year is divisible by 400, hence the year is a leap year  
13.                if ( year % 400 == 0)  
14.                    leap = true;  
15.                else  
16.                    leap = false;  
17.            }  
18.            else  
19.                leap = true;  
20.        }  
21.        else  
22.            leap = false;  
23.  
24.        if(leap)  
25.            System.out.println(year + " is a leap year.");  
26.        else  
27.            System.out.println(year + " is not a leap year.");  
28.    }  
29. }
```

Third generation programming language

Developer chooses

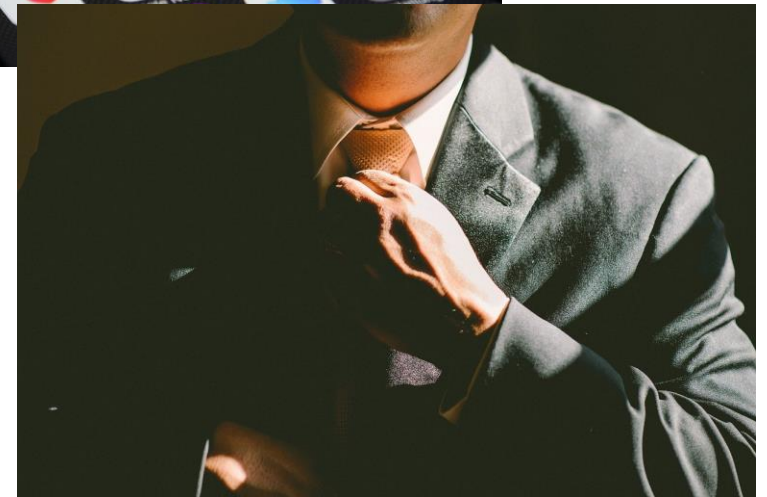
Developer is responsible!

Fix your bad choices

Fix your predecessor's
bad choices

```
1. public class LeapYear {  
2.  
3.     public static void main(String[] args) {  
4.  
5.         int year = 1900;  
6.         boolean leap = false;  
7.  
8.         if(year % 4 == 0)  
9.         {  
10.            if( year % 100 == 0)  
11.            {  
12.                // year is divisible by 400, hence the year is a leap year  
13.                if ( year % 400 == 0)  
14.                    leap = true;  
15.                else  
16.                    leap = false;  
17.            }  
18.            else  
19.                leap = true;  
20.        }  
21.        else  
22.            leap = false;  
23.  
24.        if(leap)  
25.            System.out.println(year + " is a leap year.");  
26.        else  
27.            System.out.println(year + " is not a leap year.");  
28.    }  
29. }
```

My morning routine (as my boss sees it)



Fourth generation programming language



```
SELECT    c.CountryName,  
          (SELECT COUNT(*)  
           FROM    dbo.People AS p  
           WHERE   p.CountryCode = c.CountryCode)  
                                                AS CountryPopulation  
FROM      dbo.Countries AS c  
ORDER BY  c.CountryName ASC;
```

Structured

Query

Language

Fourth generation programming language

Rank ↕	Country (or dependent territory) ↕	Population ↕	% of World Population ↕	Date ↕	Source
–	 Abkhazia ^[s]	244,832	0.00316%	1 Jan 2018	National estimate ^[167]
43	 Afghanistan	32,225,560	0.415%	1 Jul 2019	National annual estimate ^[43]
134	 Albania	2,862,427	0.0369%	1 Jan 2019	National annual estimate ^[127]
32	 Algeria	42,200,000	0.544%	1 Jan 2018	National annual projection ^[32]
–	 American Samoa (US)	56,700	0.000731%	1 Jul 2018	National annual estimate ^[95]
184	 Andorra	76,177	0.000982%	31 Dec 2018	National estimate ^[181]
47	 Angola	30,175,553	0.389%	1 Jul 2019	National projection ^[47]
–	 Anguilla (UK)	14,869	0.000192%	1 Jul 2019	UN projection ^[2]
183	 Antigua and Barbuda	96,453	0.00124%	1 Jul 2019	National annual projection ^[179]
31	 Argentina	44,938,712	0.579%	1 Jul 2019	National annual projection ^[31]
133	 Armenia	2,957,500	0.0381%	30 Sep 2019	National quarterly estimate ^[126]
–	 Aruba (Netherlands)	112,309	0.00145%	31 Mar 2019	National quarterly estimate ^[173]
54	 Australia	25,564,537	0.330%	26 Dec 2019	National population clock ^[52]
97	 Austria	8,898,457	0.115%	1 Oct 2019	Quarterly provisional figure ^[93]
90	 Azerbaijan	10,027,874	0.129%	1 Aug 2019	National estimate ^[87]
169	 Bahamas	385,340	0.00497%	1 Jul 2019	National annual projection ^[161]
148	 Bahrain	1,543,300	0.0199%	1 Jul 2019	National annual projection ^[140]
8	 Bangladesh	167,820,572	2.16%	26 Dec 2019	National population clock ^[9]
173	 Barbados	287,025	0.00370%	1 Jul 2019	UN projection ^[2]
93	 Belarus	9,454,800	0.122%	1 Oct 2019	National quarterly estimate ^[89]
80	 Belgium	11,505,732	0.148%	1 Oct 2019	Monthly National estimate ^[77]
168	 Belize	408,487	0.00527%	1 Jul 2019	National estimate ^[160]

(source: Wikipedia; retrieved 2019-12-26)

SQLServerFast.com execution plan training, block 1, basic level, chapter 1: What and why? - (c) Hugo Kornelis

```
SELECT      c.CountryName,
            (SELECT COUNT(*)
             FROM   dbo.People AS p
             WHERE  p.CountryCode = c.CountryCode)
            AS CountryPopulation
FROM        dbo.Countries AS c
ORDER BY   c.CountryName ASC;
```

Structured

Query

Language

Fourth generation programming language

```
SELECT      c.CountryName,  
            COUNT(*) AS CountryPopulation  
FROM        dbo.Countries AS c  
INNER JOIN  dbo.People    AS p  
            ON           p.CountryCode = c.CountryCode  
GROUP BY    c.CountryCode  
ORDER BY    c.CountryName ASC;
```

```
SELECT      c.CountryName,  
            (SELECT COUNT(*)  
             FROM    dbo.People AS p  
             WHERE   p.CountryCode = c.CountryCode)  
                                                    AS CountryPopulation  
FROM        dbo.Countries AS c  
ORDER BY    c.CountryName ASC;
```

```
SELECT      c.CountryName,  
            pt.CountryPopulation  
FROM        dbo.Countries AS c  
CROSS APPLY  
            (SELECT COUNT(*) AS CountryPopulation  
             FROM    dbo.People AS p  
             WHERE   p.CountryCode = c.CountryCode) AS pt  
ORDER BY    c.CountryName ASC;
```

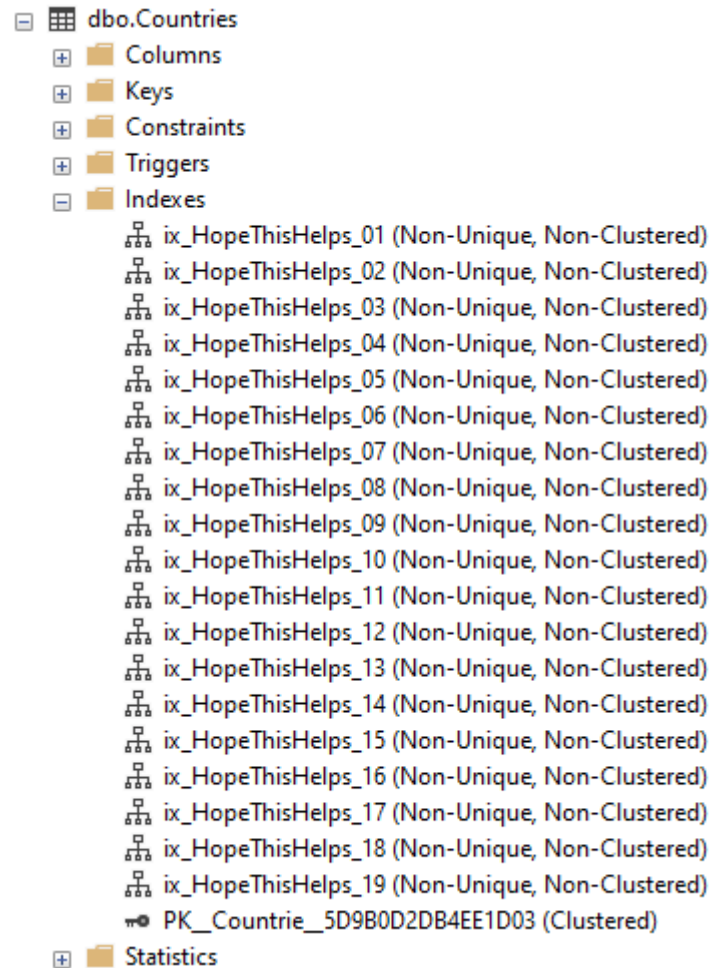
Fourth generation programming language



🔍 Help! I need to make my query go faster or my boss will fire me!

```
SELECT    c.CountryName,  
          (SELECT COUNT(*)  
           FROM    dbo.People AS p  
           WHERE   p.CountryCode = c.CountryCode)  
          AS CountryPopulation  
FROM      dbo.Countries AS c  
ORDER BY  c.CountryName ASC;
```

Fourth generation programming language



```
SELECT    c.CountryName,
          (SELECT COUNT(*)
           FROM    dbo.People AS p
           WHERE   p.CountryCode = c.CountryCode)
                                     AS CountryPopulation
FROM      dbo.Countries AS c
ORDER BY  c.CountryName ASC;
```

Fourth generation programming language

Execution plan

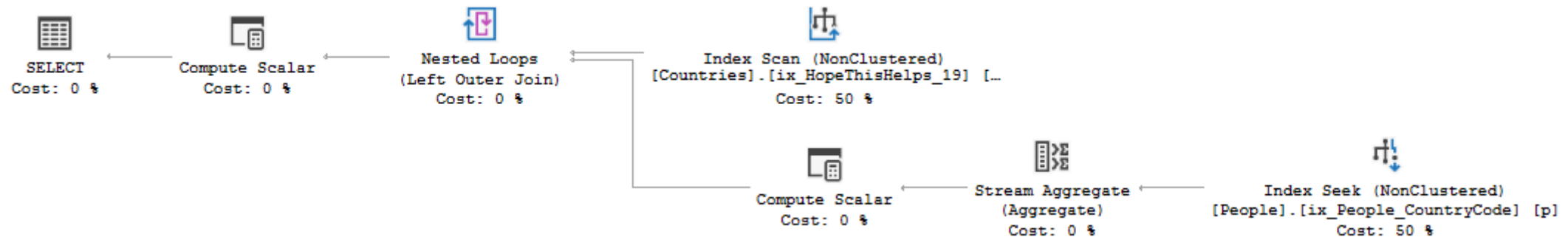
Created by Query Optimizer

Based on query, indexes, statistics, etc.

Shows how query executed

Reveals where time was lost

```
SELECT    c.CountryName,  
          (SELECT COUNT(*)  
           FROM    dbo.People AS p  
           WHERE   p.CountryCode = c.CountryCode)  
          AS CountryPopulation  
FROM      dbo.Countries AS c  
ORDER BY  c.CountryName ASC;
```



Summary

Relevance of execution plans

- Show the algorithms used to execute a query

- Reveal root cause of bad performance

- Allow effective, targeted tuning

Next chapters

Chapter 2: Requesting an execution plan

- Three ways to request an execution plan for a query

- Differences and similarities

- How to get them

Chapter 3: How to read an execution plan

Chapter 4: Properties

Chapter 5: Where to find execution plans

Chapter 6: Cardinality in the execution plan

Chapter 7: Percentages in the execution plan