Normal Forms

ISM 6218

Due on September 10

The Avengers Team

"We will avenge every problem on our way"

Aitemir Yeskenov (Team Lead)

Nagarjuna Kanneganti

Sai Suraj Argula

Vinay Kumar Reddy Baradi

Table of Contents

Business Process Supported

The database created for the Normal Forms assignment is the employee database. The database consists of a 2NF table EmployeeDetails that is supposed to satisfy the 3rd, 5th, and the 6th Normal Forms. Our assumption is that normalization of the table helps a DB user to achieve efficiency and get rid of redundant records such as records of the attributes dependent either partially or transitively.

Requirement Described

1. Create RS demonstrating 3rd, 5th and 6th normal form designs.

2. Generate separate database versions using creating table designs implementing 3rd, 5th, and 6th. You will need to make use of separate servers (dev-test-stage) for each database.

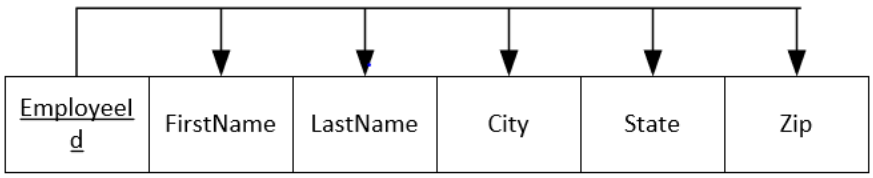
3. Implement “look up tables” for static attributes such as city, state, zip.

4. Run query series experiments comparing designs to evaluate differences in performance. You will need to make use of execution plan and live statistics settings to complete this exercise; and complete query across servers.

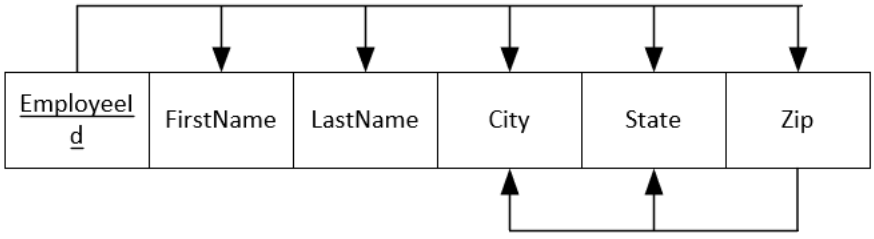
5. Generate report of experiment results with supported user stories and derived requirements.

1) Creation of RS demonstrating 3NF, 5NF, and 6NF designs

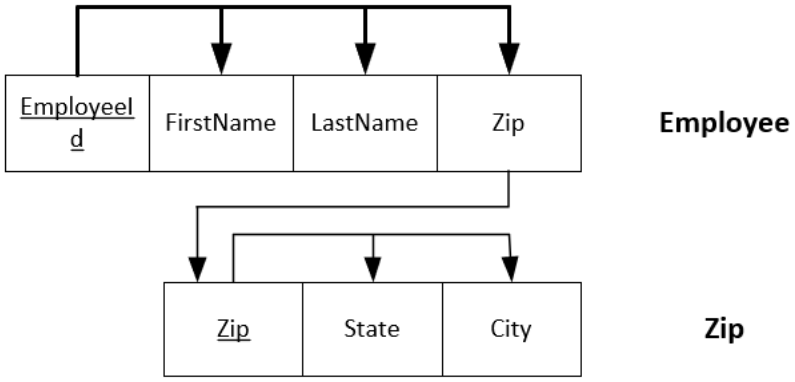
For this assignment, we decided to create a database from scratch. Initially, the database has one table – **EmployeeDetails,** which needs to be normalized for a better design and better performance in the future. The **EmployeeDetails** has a general information about the employee: **first name**, **last name**, **city**, **state**, and **zip**. All the attributes functionally depend on the primary key that is **EmployeeId**. Below is the initial state of the table when our team just came up with it:



After carefully looking at the table we idenfitied that in order to normalize the table and satisfy the 3NF condition we have to get rid of the transitive dependency. The transitive dependency is clearly identified:



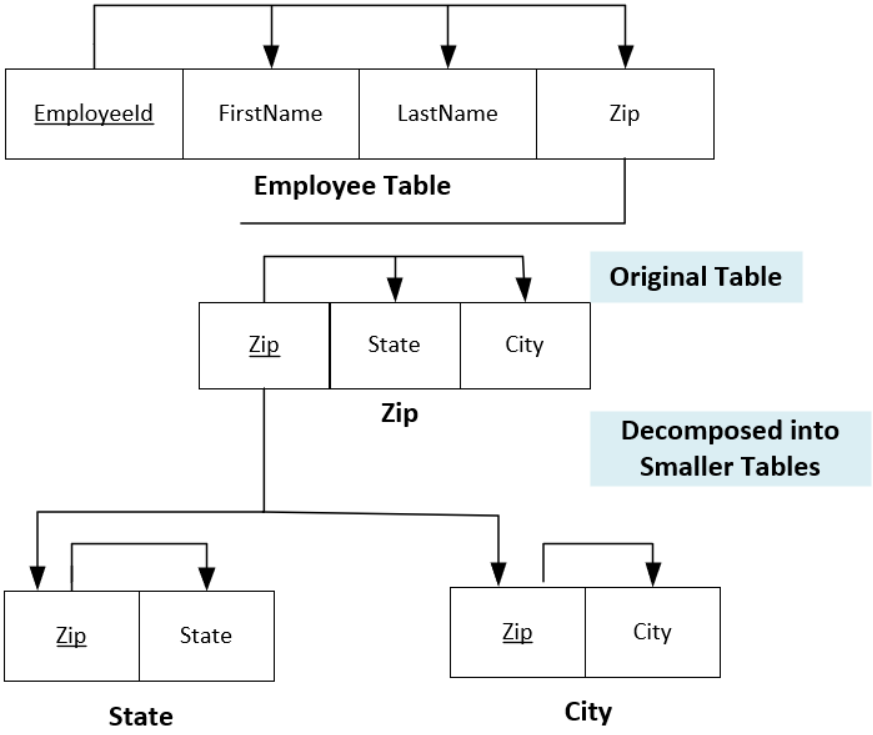
City and State oftentimes depend on the Zip attribute. To satisfy the 3NF, we broke the table into two tables:



As you can see, now we have two tables – **Employee** and **Zip**.

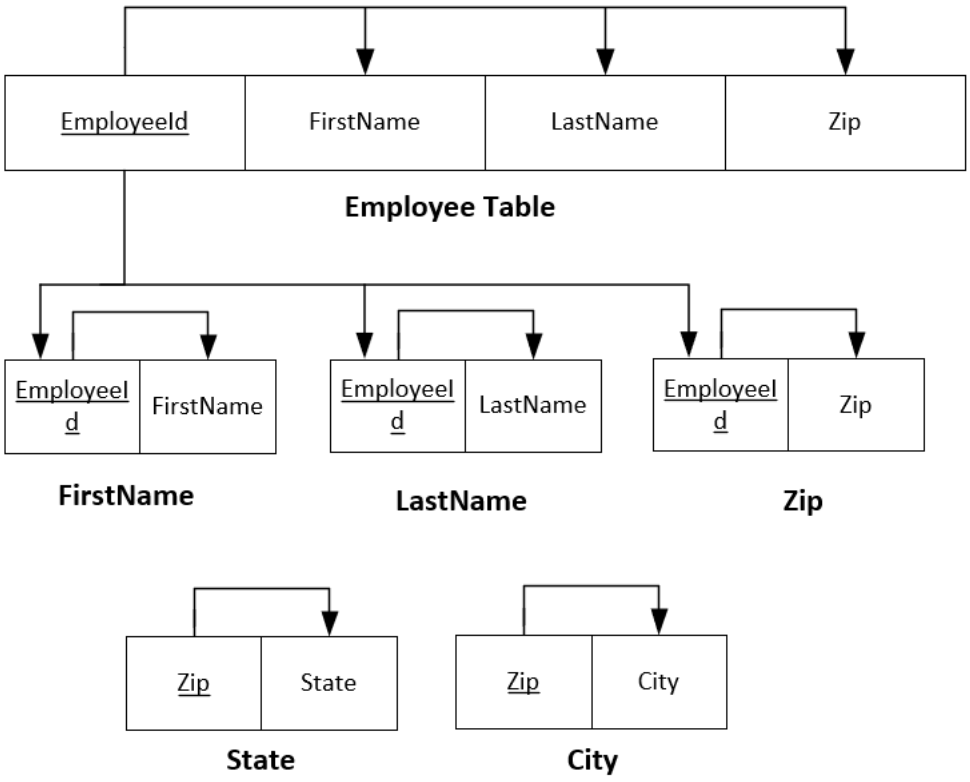
For a better performance we decided to go further and satisfy the **5NF** form. The expectation is to have more records in the future. This means that the run time becomes highly important for the DB user. Lossless decomposition test is what we need. In other words, the efficient way to run queries faster is to break the existing tables into smaller component tables:

5th Normal Form



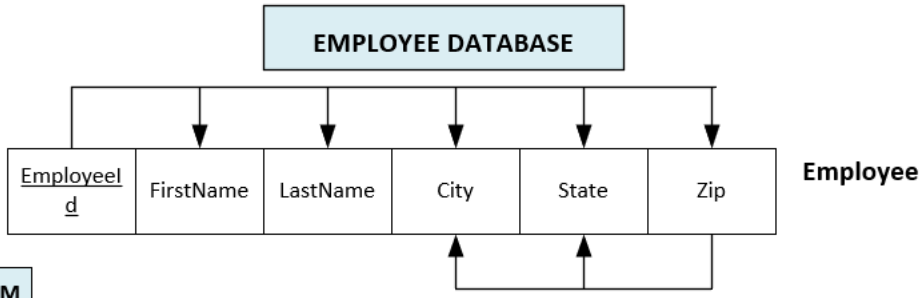
As you can see, we now have two more tables – **State** and **City**. In both cases, the primary key is the **Zip** attribute.

After experimenting with the "enhanced" vertical partition by satisfying the 5th normal form, we thought that it would be also useful to try to achieve the 6th normal form and then compare both form's performances. Here is what we did for the 6NF:

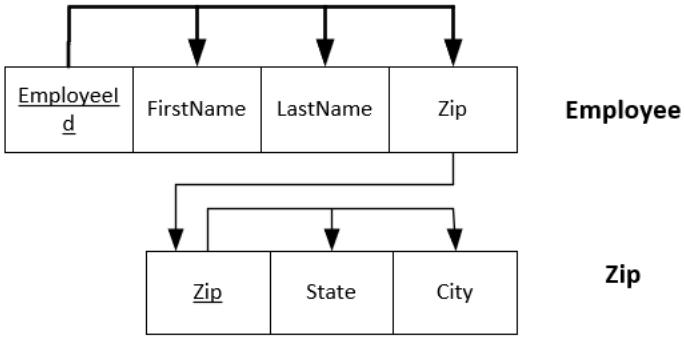


As you can see, each table now has the fewest number of columns that is exactly 2. As a result, we have 5 tables: **FirstName**, **LastName**, **Zip**, **State**, and **City**.

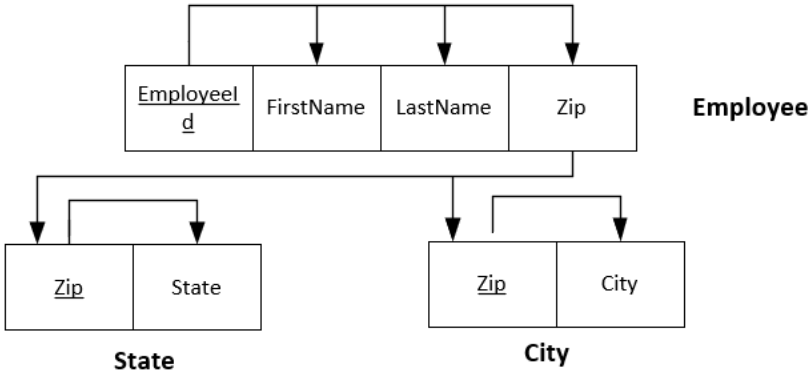
The diagram below shows the **EmployeeDetails** table on different stages:



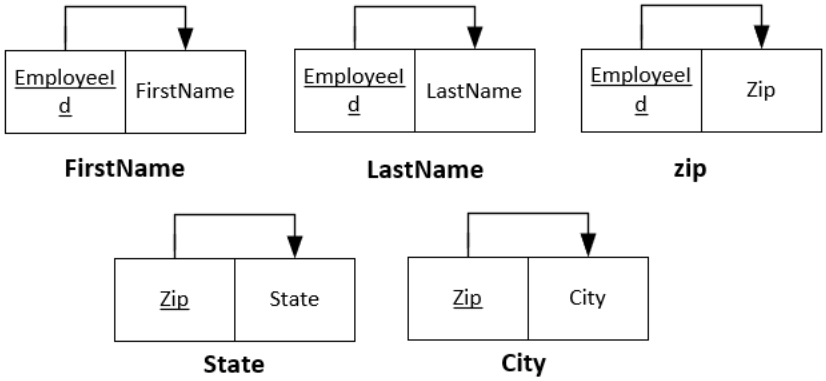
3rd NORMAL FORM



5Th NORMAL FORM



6th NORMAL FORM

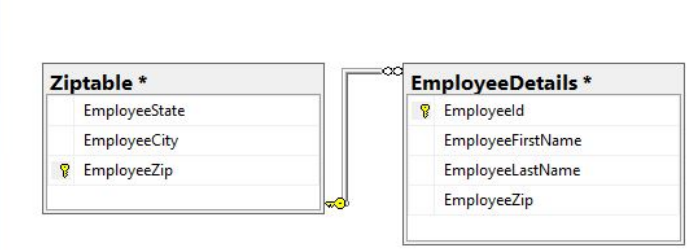


As you can see, at the end we do not even have such table as **EmployeeDetails** since we had to split it into more and more component tables.

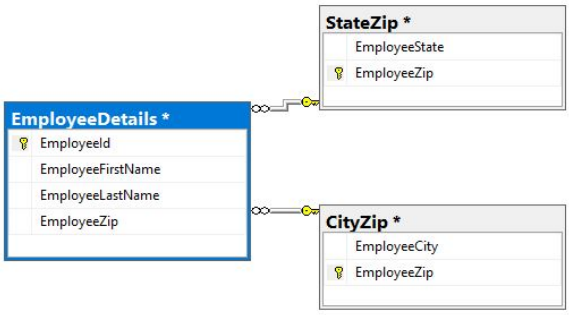
2) Generate DB versions from the designed RS

The next step is to create the database within the SQL Server Management Studio. We generated 3 different databases (3rd, 5th, and 6th forms accordingly) on 3 different servers (dev-test-stage).

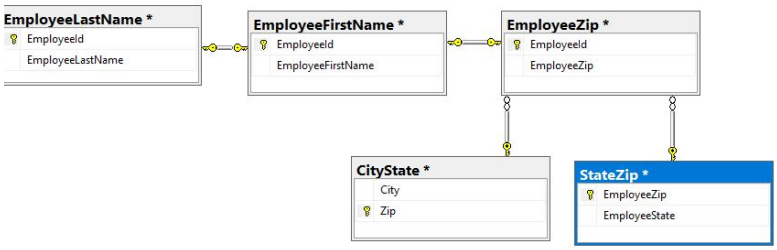
3NF:



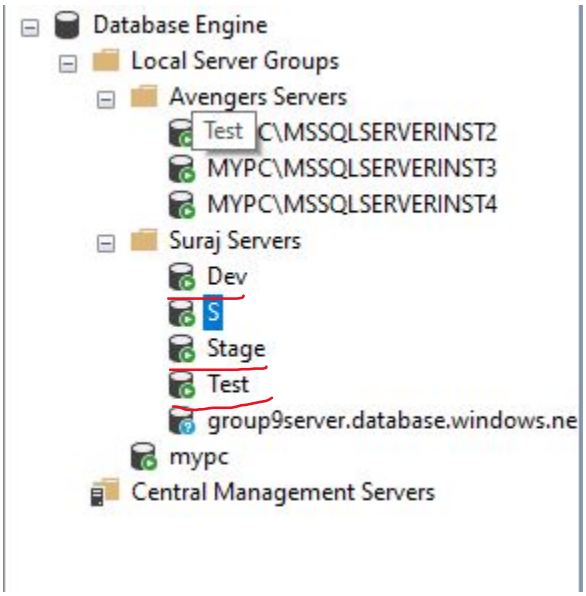
5NF:



6NF:



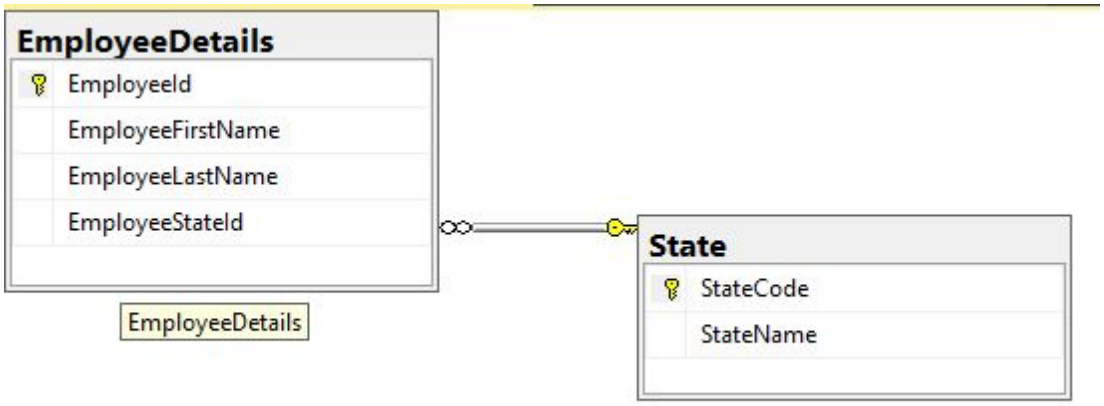
3 servers used for this task:



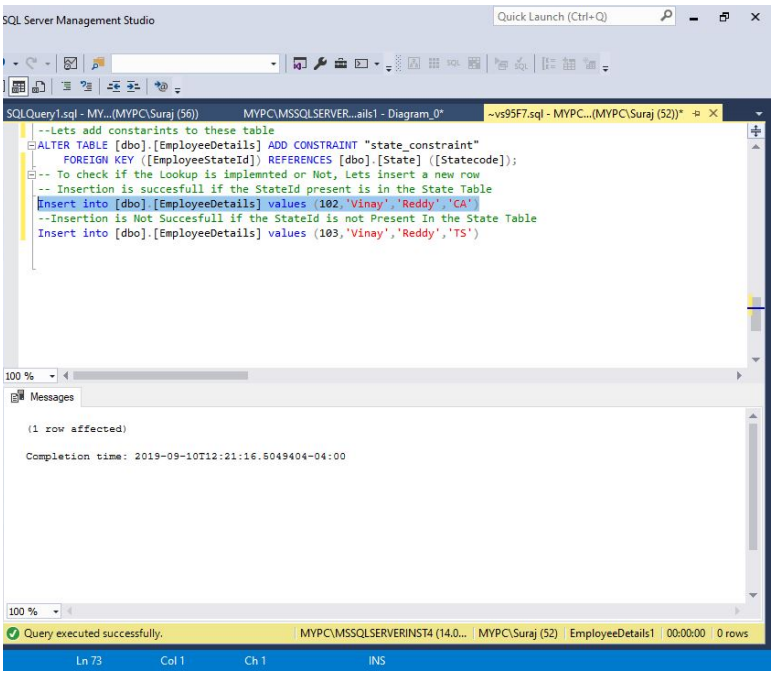
Note that for the experiment we created Dev, Test, and the Stage servers.

3) Look up table implementation

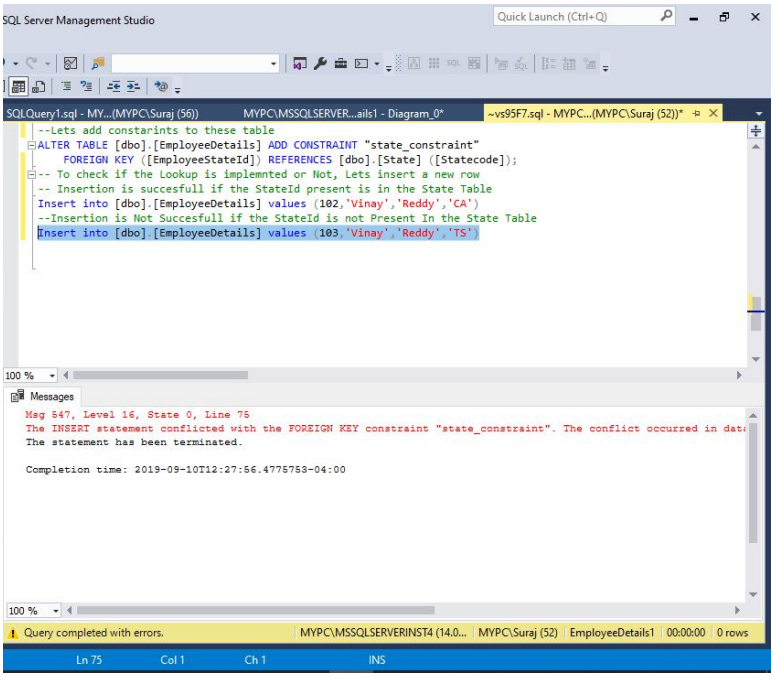
Implementation of look up tables for static attributes such as Zip, State, and City is also important. As an example, we created a State table that serves as a lookup table providing the name of the state:



There is a constraint that we also added that checks specific state that a DB user can add. As you can see on the screenshots, it runs the INSERT query with the state of California “CA”

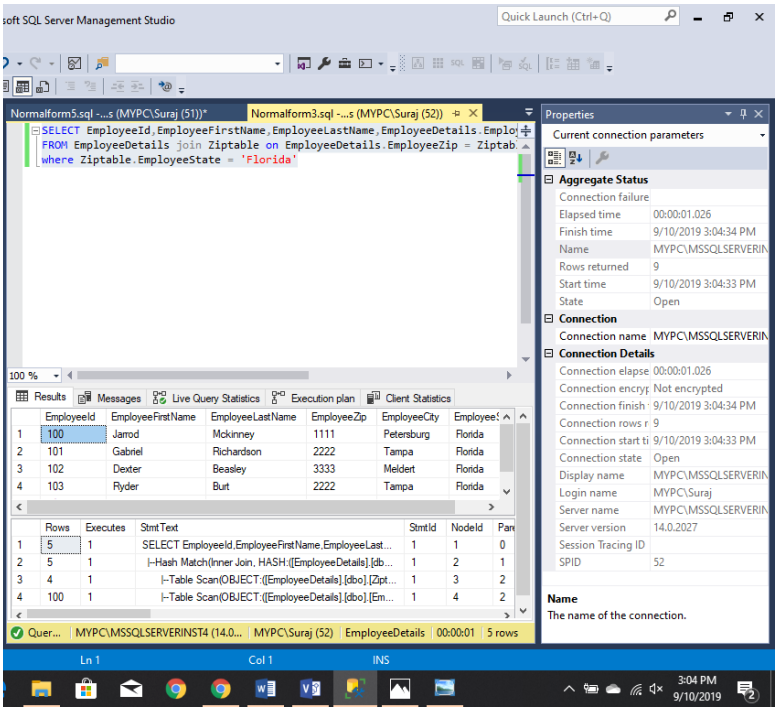


However, due to the constraint, similar query, but with a different value of state outside of the US will not execute. See the example below with “TS”.

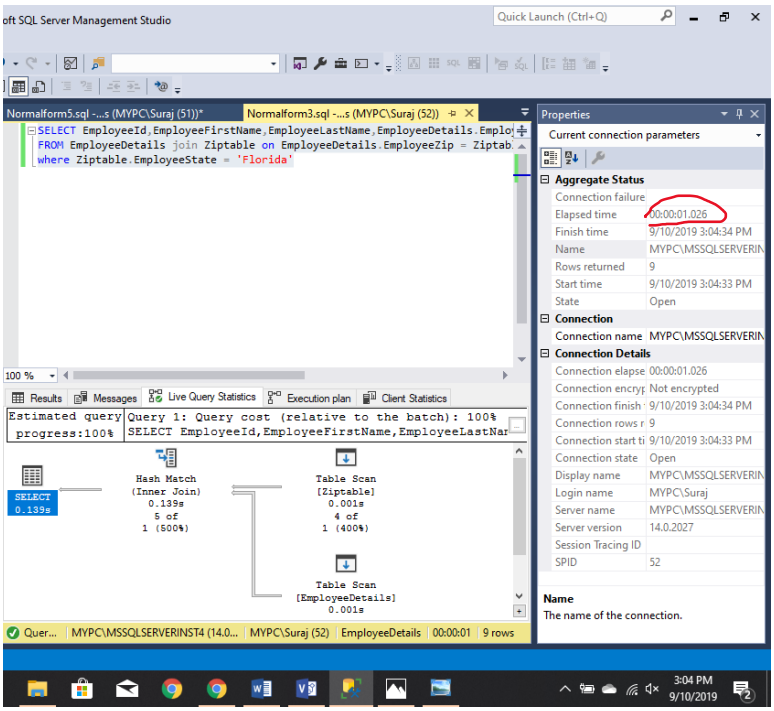


4) Run queries to experiment with design comparison. Evaluation of difference in performance.

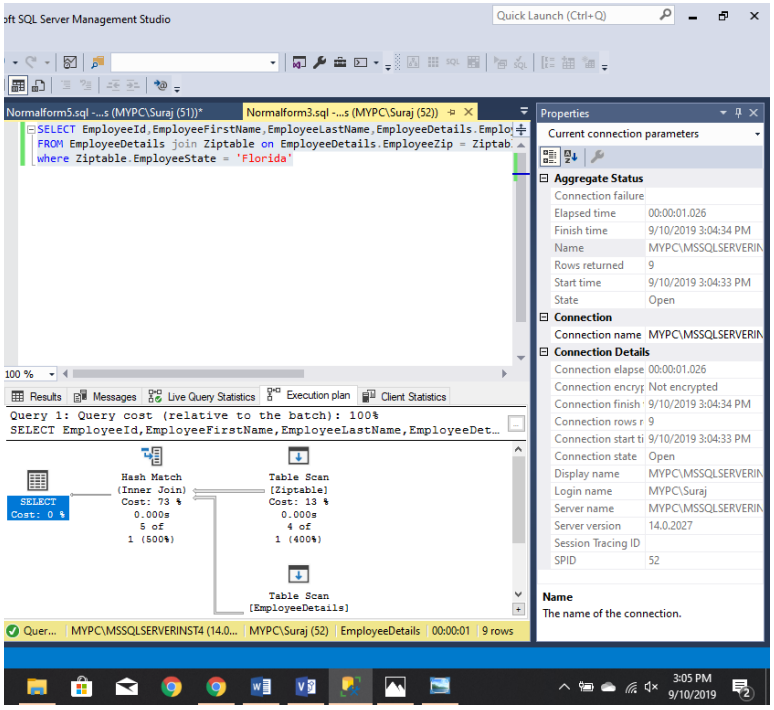
For the 3NF designed table we ran the **SELECT** query to return records that condition requires attribute **State** to have value of “Florida”. In other words, every record returns employees from Florida:



As you can see, we see the output with the data. Below is the Live Query Statistics:

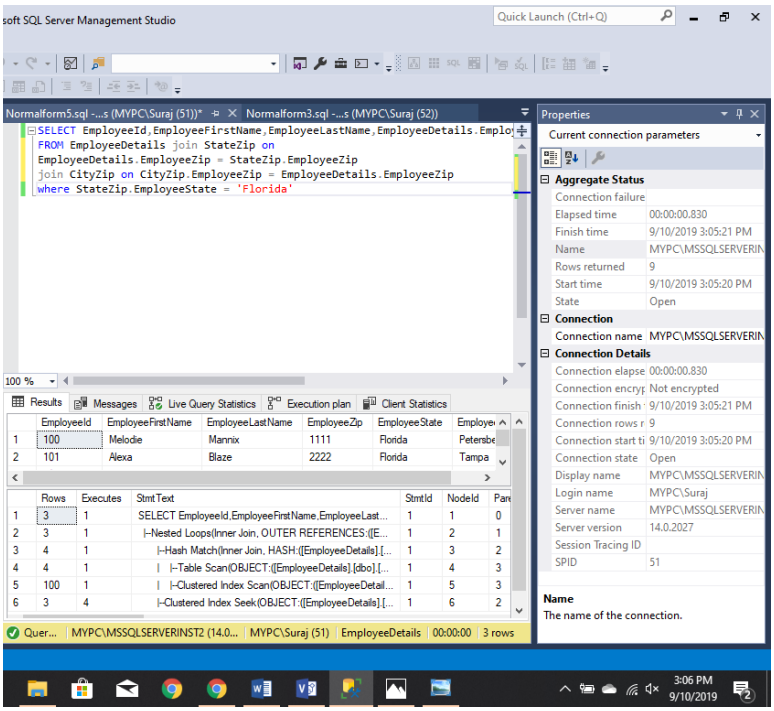


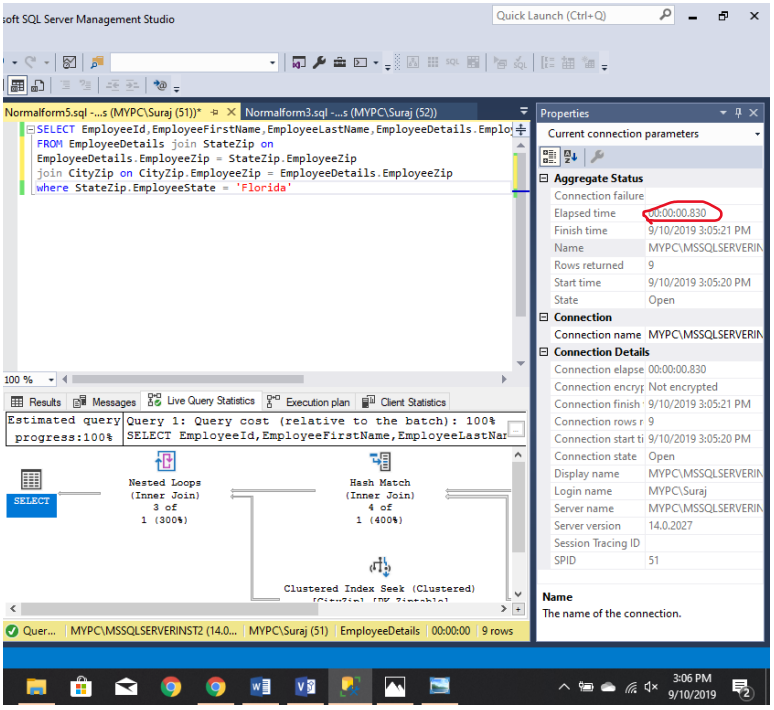
The execution time is 1.026. And also we can see the execution plan:



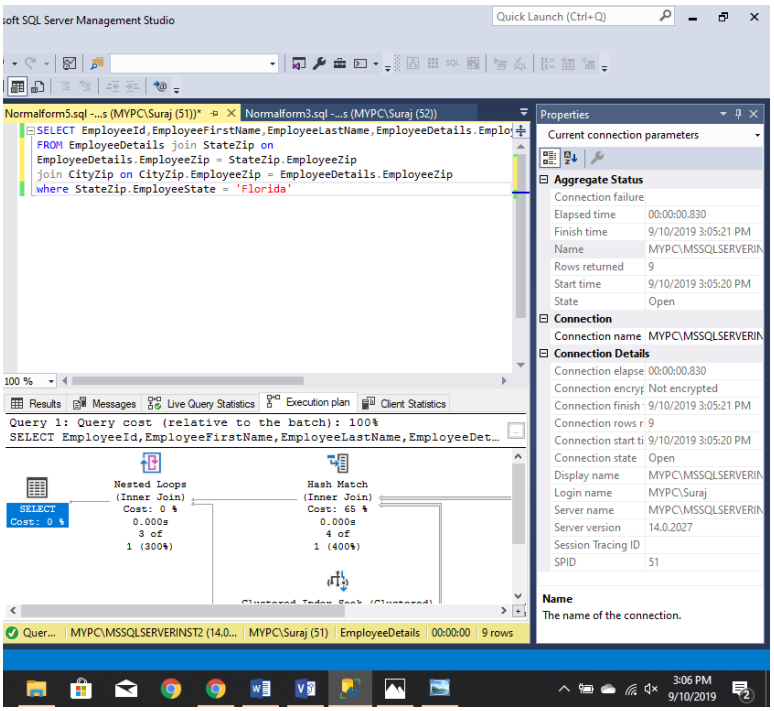
We ran exactly same operation for the 5NF and 6NF tables:

5NF table:

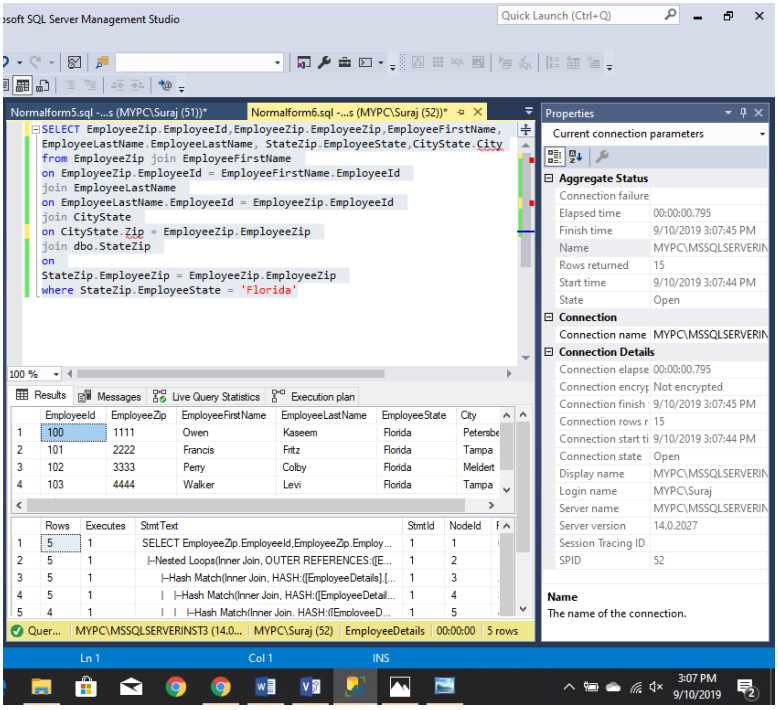


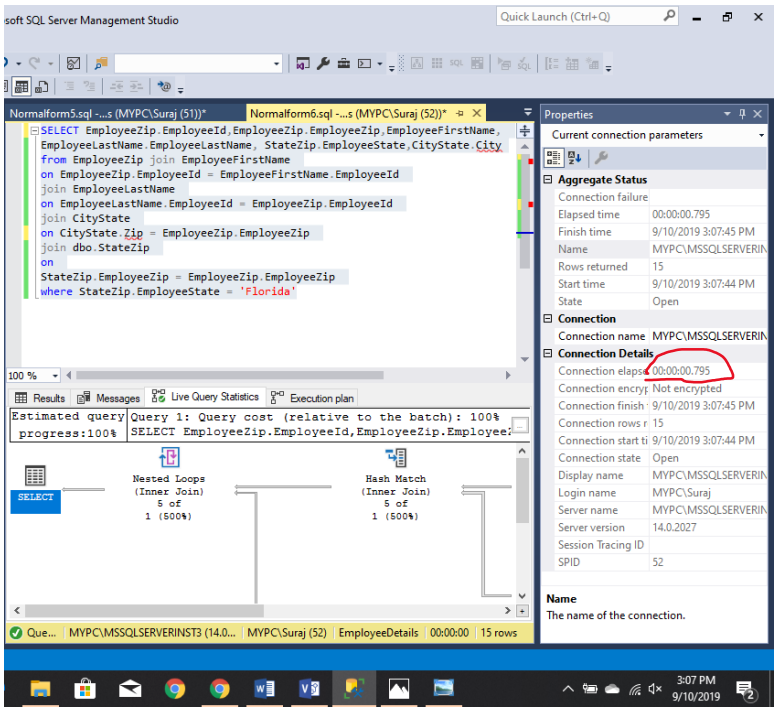


The execution time is 0.830. And also we can see the execution plan:

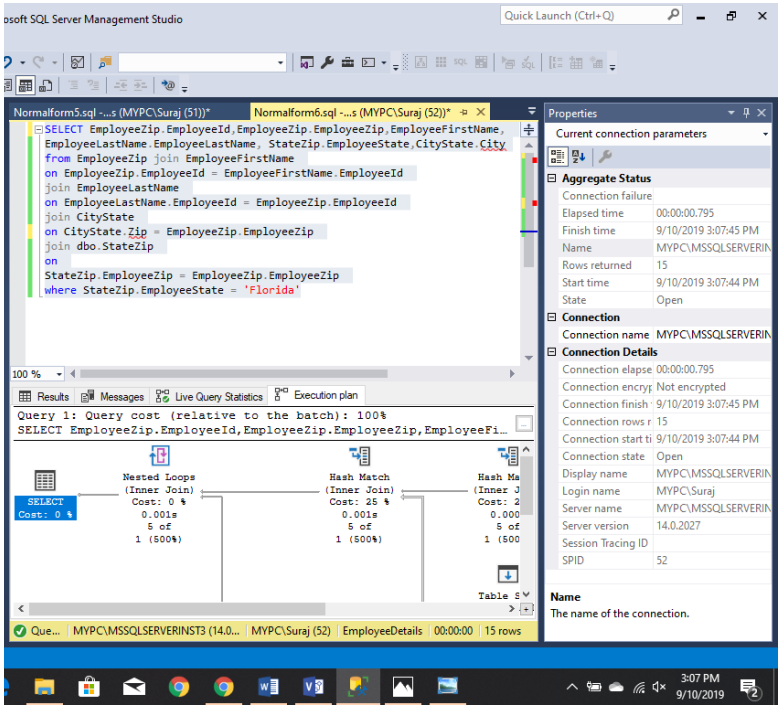

Quick Launch (Ctrl+Q)
Normalform5.sql -...s (MYPC\Suraj (51))*
Normalform3.sql -...s (MYPC\Suraj (52))
SELECT EmployeeId,EmployeeFirstName,EmployeeLastName,EmployeeDetails.Emplo
FROM EmployeeDetails join StateZip on
EmployeeDetails.EmployeeZip = StateZip.EmployeeZip
join CityZip on CityZip.EmployeeZip = EmployeeDetails.EmployeeZip
where StateZip.EmployeeState = 'Florida'
Results
Messages
Live Query Statistics
Execution plan
Client Statistics
Query 1: Query cost (relative to the batch): 100%
SELECT EmployeeId,EmployeeFirstName,EmployeeLastName,EmployeeDet...
SELECT
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 0 %
0.000s
3 of
1 (300%)
Hash Match
(Inner Join)
Cost: 65 %
0.000s
4 of
1 (400%)
Properties
Current connection parameters
Aggregate Status
Connection failure
Elapsed time 00:00:00.830
Finish time 9/10/2019 3:05:21 PM
Rows returned 9
Start time 9/10/2019 3:05:20 PM
State Open
Connection
Connection name
Connection Details
Login name MYPC\Suraj
Server name
Server version 14.0.2027
Session Tracing ID
SPID 51
Name
The name of the connection.
MYPC\MSSQLSERVERINST2 (14.0...
MYPC\Suraj (51)
EmployeeDetails
00:00:00
9 rows
3:06 PM
9/10/2019

6NF table:


Quick Launch (Ctrl+Q)
Normalform5.sql -...s (MYPC\Suraj (51))*
Normalform6.sql -...s (MYPC\Suraj (52))*
SELECT EmployeeZip.EmployeeId,EmployeeZip.EmployeeZip,EmployeeFirstName,
EmployeeLastName.EmployeeLastName, StateZip.EmployeeState,CityState.City
from EmployeeZip join EmployeeFirstName
on EmployeeZip.EmployeeId = EmployeeFirstName.EmployeeId
join EmployeeLastName
on EmployeeLastName.EmployeeId = EmployeeZip.EmployeeId
join CityState
on CityState.Zip = EmployeeZip.EmployeeZip
join dbo.StateZip
on
StateZip.EmployeeZip = EmployeeZip.EmployeeZip
where StateZip.EmployeeState = 'Florida'
Results
Messages
Live Query Statistics
Execution plan
EmployeeId EmployeeZip EmployeeFirstName EmployeeLastName EmployeeState City
1 100 1111 Owen Kaseem Florida Petersbe
2 101 2222 Francis Fritz Florida Tampa
3 102 3333 Perry Colby Florida Meldert
4 103 4444 Walker Levi Florida Tampa
Rows Executes StmtText StmtId NodeId
1 5 1 SELECT EmployeeZip.EmployeeId,EmployeeZip.Employ... 1 1
2 5 1 |--Nested Loops(Inner Join, OUTER REFERENCES:([E... 1 2
3 5 1 |--Hash Match(Inner Join, HASH:([EmployeeDetails].[... 1 3
4 5 1 |--Hash Match(Inner Join, HASH:([EmployeeDetail... 1 4
5 4 1 |--Hash Match(Inner Join, HASH:([EmployeeD... 1 5
Properties
Current connection parameters
Aggregate Status
Connection failure
Elapsed time 00:00:00.795
Finish time 9/10/2019 3:07:45 PM
Rows returned 15
Start time 9/10/2019 3:07:44 PM
State Open
Connection
Connection name
Connection Details
Login name MYPC\Suraj
Server name
Server version 14.0.2027
Session Tracing ID
SPID 52
Name
The name of the connection.
MYPC\MSSQLSERVERINST3 (14.0...
MYPC\Suraj (52)
EmployeeDetails
00:00:00
5 rows
Ln 1
Col 1
INS
3:07 PM
9/10/2019



The execution time is 0.795. And also we can see the execution plan:



After the experiment, we can compare the live query statistics. We found out that the time taken for execution is less in the 6th Normal Form comparing to the 5NF and 3NF.

Here is the report on experiment series with the execution time per the table:

3NF - 1.026s

5NF - 0.830s

6NF - 0.795s

The 6NF turned out to be the fastest. Note: it is important that when we have a small number of records (1-1000) the difference in execution time might not be significant. It may even take more time for 5NF than for the 3NF form in rare scenarios. Once the volume of records is increased, the DB users start to notice the difference. For example, if we added 100,000 more records, the difference between the 3NF, 5NF, and 6NF would be even more clear.