Obligatorisk aflevering 3

Af Kristofer, Michael, Søren og Mads

Explore the data and formulate considerations about a hosting database.

To create a database, we must first explore and understand the different types of data in each dataset, as well as how the different datasets are connected to each other with the data they contain. In this section we will go over the different datasets and look at what columns they contain, and for each column we will look through the data contained within to determine what type of value they are. We will also try to give each a quick description to help us understand the relationships between all of the datasets.

Data exploration:

2016_-_Cities_Emissions_Reduction_Targets_20240207.csv:

Organisation: Looking at the column of data we can see that they are strings writing in the native language where the organization resides. The data are names of different government organizations overseeing a city. The value is unique.
Account No: Account number or id in non-sequential order. The value is unique to each organization.
Country: The country of which the organization is located within. The values are string and contain names. One country can have multiple organizations within it. Unlike the values of Organisation the values are all written in english.
City Short Name: The shortened name of the city where the organization resides. The values are string and some contain special characters.
C40: The value type is a string and represents if a city can be labeled as a C40 city.
Reporting Year: The year the organization gave the report from which the data of the row is gathered from. The data is an integer.
Sector: From which part of the organization the data has been gathered from. The values first read as an enum but with a few rows of data where multiple choices are included means the value should be a list of enums.
Target boundary: The limit where emission data has been collected within. The value is a string and is used as a description. The value in each row varies with some rows missing data.
Baseline year: The year where the baseline measurements were taken. The value is an integer.

Baseline emissions (metric tonnes CO2e): The emission value measured as the baseline. The value is a float and represents metric tons.
Percentage reduction target: The desired reduction of the baseline emission measured. The value is a float and represents percentage.
Target date: The year in which the organization aims to have the baseline emission reduced by the Percentage reduction target. The value is an integer.
Comment: String value containing a comment from the rapport. Some rows have missing values.
City Location: Coordinates of the city's location which most likely refers to the center of the city. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.
Country Location: Coordinates of the county’s location which most likely refers to the center of the country. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.

2016_-_Citywide_GHG_Emissions_20240207.csv:

Account Number: Account number or id in non-sequential order.
City Name: The full name of the city. The value is a string.
Country: The country of which the organization is located within. The values are string and contain names. One country can have multiple organizations within it. Unlike the values of Organisation the values are all written in english.
City Short Name: The shortened name of the city where the organization resides. The values are string and some contain special characters.
C40: The value type is a string and represents if a city can be labeled as a C40 city.
Reporting Year: The year the organization gave the report from which the data of the row is gathered from. The data is an integer.
Measurement Year: The year where the measurements were taken. The value is a date.
Boundary: The limit where emission data has been collected within. The value is a string and is used as a description. The value in each row varies with some rows missing data.
Primary Methodology: Name of the methodology used for the data collection. The value is a string.
Methodology Details: A description of the Primary Methodology used. The value is a string.
Gases included: What type of gases are included in the measurements. The value can contain multiple choices which could be considered enums since the gases that can be included appear to be static.
Total City-wide Emissions (metric tonnes CO2e): The value is a float and represents metric tons.
Total Scope 1 Emissions (metric tonnes CO2e): The value is a float and represents metric tons.
Total Scope 2 Emissions (metric tonnes CO2e): The value is a float and represents metric tons.
Increase/Decrease from last year If there have been any changes in their measurements compared to last year. The value is an enum which represents “=, >, <” or if it's their first measurement.

Reason for increase/decrease in emissions:
A description given in the report. The value is a string.

Current Population Year:
The year of the Current Population. The value is an integer.

Current Population:
The measured number of people living in the country. The value is an integer

City GDP:
The Gross Domestic Product of the market value within a city. The value is an integer.

GDP Currency:
What type of currency the City CDP refers to. The value is a string which can be separated into the long name and short name of the currency

Year of GDP:
The year of the City GDP was measured. The value is an integer.

GDP Source:
Where the information regarding the GDP was gathered. The value is a string and can both be a name of an organization or a link to a website.

Average annual temperature (in Celsius):
The average annual temperature of the city in celsius. The value is a float.

Land area (in square km):
The size of the city in square kilometers. The value is a float.

Average altitude (m):
The average altitude in meters of the city. The value is an integer.

City Location:
Coordinates of the city's location which most likely refers to the center of the city. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.

Country Location:
Coordinates of the county’s location which most likely refers to the center of the country. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.

2017_-_Cities_Community_Wide_Emissions.csv:

Account number:
Account number or id in non-sequential order. The value is unique to each organization.

Organization:
Looking at the column of data we can see that they are strings writing in the native language where the organization resides. The data are names of different government organizations overseeing a city. The value is unique.

City:
The full name of the city. The value is a string.

County:
The country of which the organization is located within. The values are string and contain names. One country can have multiple organizations within it. Unlike the values of Organisation the values are all written in english.

Region:
Where on the world map the country is located.
The value is a string and represents the different names of the world regions.

C40:
The value type is a string and represents if a city can be labeled as a C40 city.

Access:
If the report is accessible to the public or private. It would be reasonable to assume that there would be reports which are or would be marked as private then we can reflect that the data is or is not public. The value is a bool representing public or private.

Reporting year:
The year the organization gave the report from which the data of the row is gathered from. The data is an integer.

Accounting year:
The period of time where the data gathered in the report originated from. The value is a string containing two dates(YYYY-MM-DD) separated by a “-” and represents a period of time.

Boundary:
The limit where data has been collected within. The value is a string and is used as a description. The value in each row varies with some rows missing data.

Protocol:
The name of the protocol used for the gathering of information. The value is a string.

Protocol column:
A description of how the protocol was used to gather information for the report.
The value is a string.

Gases included:
A description of the type of gases included when gathering information. The value is a string and contains mostly a list of the chemical name of the gases but also contains a description.

Total emissions (metric tonnes CO2e):
The total measured emissions of CO2 in tons. The value is float. Most of the values in the column could be integers but the few who aren’t forces the others.

Scopes Included:
What scopes are included in the report. The value is a string.

Total Scope 1 Emissions (metric tonnes CO2e):
The value is a float and represents metric tons.

Total Scope 2 Emissions (metric tonnes CO2e):
The value is a float and represents metric tons.

Comment:
A comment in there may be in the report. The value is a string and can contain empty values.

Increase/Decrease from last year:
If there have been any changes in their measurements compared to last year. The value is an enum which represents “=, >, <” or if it's their first measurement.

Reason for increase/decrease in emissions:
A description given in the report. The value is a string.

Population:%
The measured number of people living in the country. The value is an integer

Population year:
The year of the Population. The value is an integer.

GDP:
The Gross Domestic Product of the market value within a city. The value is an integer.

GDP Currency:
What type of currency the City CDP refers to. The value is a string which can be separated into the long name and short name of the currency

GDP Year:
The year of the City GDP was measured. The value is an integer.

GDP Source:
Where the information regarding the GDP was gathered. The value is a string and can both be a name of an organization or a link to a website.

Average annual temperature (in Celsius):
The average annual temperature of the city in celsius. The value is a float.

Average altitude (m):
The average altitude in meters of the city. The value is an integer.

Land area (in square km):
The size of the city in square kilometers. The value is a float.

City Location:
Coordinates of the city's location which most likely refers to the center of the city. The

value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.
Country Location: Coordinates of the county’s location which most likely refers to the center of the country. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.

2017_-_Cities_Emissions_Reduction_Targets_20240207.csv:

Account No:
Account number or id in non-sequential order.

Organisation:
Looking at the column of data we can see that they are strings writing in the native language where the organization resides. The data are names of different government organizations overseeing a city. The value is unique.

City:
The full name of the city. The value is a string.

Country:
The country of which the organization is located within. The values are string and contain names. One country can have multiple organizations within it. Unlike the values of Organisation the values are all written in english.

Region:
Where on the world map the country is located.
The value is a string and represents the different names of the world regions.

Access:
If the report is accessible to the public or private. It would be reasonable to assume that there would be reports which are or would be marked as private then we can reflect that the data is or is not public. The value is a bool representing public or private.

C40:
The value type is a string and represents if a city can be labeled as a C40 city.

Reporting year:
The year the organization gave the report from which the data of the row is gathered from. The data is an integer.

Type of target:
The value type is an enum:

“Absolute target”
“Base year intensity target”
“Baseline scenario (business as usual) target”

Sector:
The value type is string. It would be an enum if not for “Other” which offers a string description.

Baseline year:
The year where the baseline measurements were taken. The value is an integer.

Baseline emissions (metric tonnes CO2e):
The emission value measured as the baseline. The value is a float and represents metric tons.

Percentage reduction target:
The desired reduction of the baseline emission measured. The value is a float and represents percentage.

Target date:
The year in which the organization aims to have the baseline emission reduced by the Percentage reduction target. The value is an integer.

Estimated business as usual absolute emissions in target year (metric tonnes CO2e):
How much CO2 is estimated to be produced as a byproduct of business.
The type is an integer and can be empty.

Intensity unit (emissions per):
The value type is a string and is a description of what “per” the measurements are taking.

Comment:
String value containing a comment from the rapport. Some rows have missing values.

Population:
The measured number of people living in the country. The value is an integer

Population Year:
The year of the Population. The value is an integer.

City Location:
Coordinates of the city's location which most likely refers to the center of the city. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.

Country Location:
Coordinates of the county’s location which most likely refers to the center of the country. The value is a string considering the use of parentheses. Within the parentheses there are two numbers separated by a comma: latitude and longitude.

2023_Cities_Climate_Risk_and_Vulnerability_Assessments_20240207.csv:

Questionnaire:
The value is a string. All the values within the column are the same: “Cities 2023”.

Organization Number:
Account number or id in non-sequential order.

Organization Name:
Name of the organization. The value is a string and the names within are written in the languages of the country it resides in.

City:
Name of the city the organization resides in. The value is a string and some values are empty.

Country/Area:
The country the city and organization resides in. The value is a string and the names are written in english.

CDP Region:
The state/region relevant CDP reporting platform.
The value is a string and contains the name of a region.

C40 City:
The value is a bool.

GCoM City:
The value is a bool.

Access:
If the report is accessible to the public or private. It would be reasonable to assume that there would be reports which are or would be marked as private then we can reflect that the data is or is not public. The value is a bool representing public or private.

Assessment attachment and/or direct link
A PDF of download link for an assessment document. The value may be meant to contain a pdf file but all values within the column is a string with some being a filename and extension name, and some being a web link.

Confirm attachment/link provided
Confirmation of which type of attachment was given. The values indicate an enum of “PDF”, “Link” but also contains “Other” where a description can be given, therefore making the value a string.

Boundary of assessment relative to jurisdiction boundary:
The value is an enum indicating the scale of the report.
“Smaller - covers only part of the jurisdiction, please explain exclusions: Área referente ao bairro de Porto de Pedra, correspondendo ao mapeamento de risco de movimento de massa e maior abrangência nas ruas Maria de Souza, Dom Marcos de Noronha, Senador Álvaro Uchoa.”

“Partial - covers part of the jurisdiction and adjoining areas, please explain exclusions/additions: Zona precordillerana y cordillerana de Peñalolén. Incluye la principal cuenca de la comuna, abarcando comuna aledaña” “Same - covers entire jurisdiction and nothing else” “Larger - covers the whole jurisdiction and adjoining areas, please explain additions: This is a county-wide plan”
Year of publication or approval: The year where the report was either publicized or approved. The value is an integer.
Factors considered in assessment: A description of what was considered during the assessment. The value is a string.
Primary author(s) of assessment: Who oversaw the assessment. The value is a string and can be empty.
Does the city have adaptation goal(s) and/or an adaptation plan?: The data indicates that the value is of type enum. “Adaptation goal(s) and adaptation plan” “Adaptation plan” “Adaptation goal(s)” “Incomplete report”
Population: The measured number of people living in the country. The value is an integer
Population Year: The year of the Population. The value is an integer.
City Location: The coordinates of the city. The value is a string. The coordinates are within two parentheses and separated with a space.
Last update: This value contains a date and represents when the values of the row was last updated.

Dataset relationship:

Here we will describe how the different datasets are connected to each other through their columns, values, data and a description of the overall meaning of the different reports represented in the datasets. We will also describe what each dataset represents.

From here each dataset will be know as:
Reduc16:
- 2016_-_Cities_Emissions_Reduction_Targets_20240207.csv

GHG:
- 2016_-_Citywide_GHG_Emissions_20240207.csv

Comm:
- 2017_-_Cities_Community_Wide_Emissions.csv

Reduc17:
- 2017_-_Cities_Emissions_Reduction_Targets_20240207.csv

RiskAndVul:
- 2023_Cities_Climate_Risk_and_Vulnerability_Assessments_20240207

Firstly we can see how they relate to each other by doing a quick comparison of some of the different columns in each dataset. We can see that many of them share a lot of data even if the names of the columns don't entirely match across the board. From doing this quick comparison we can also see that they all share an account number which we can use to easily create relationships within our database.

	GHG	Reduc16	Reduc17	Comm	RiskAndVul
Account number	Account Number	Account No	Account No	Account number	Organization Number
Organisation			Organisation	Organization	Organization Name
City	City Name		City	City	City
Country	Country	Country	Country	Country	Country/Area
Access			Access	Access	Access
C40	C40	C40	C40	C40	C40 City
Reporting year	Reporting Year	Reporting Year	Reporting year	Reporting year	
Baseline year	Measurement Year	Baseline year	Baseline year	Accounting year	
Baseline emission	Total City-wide Emissions (metric	Baseline emissions (metric tonnes CO2e)	Baseline emissions (metric tonnes CO2e)	Total emissions (metric tonnes CO2e)	

	tonnes CO2e)				

Organization account number:
Through each of the datasets there is a column, though with a different column name, that represents the account number of an organization that has submitted a report to at least one of the datasets. We can therefore connect all the different reports in the different datasets through their unique account number.

City, County and Coordinates:
The data can give us a picture of where the work against climate change is strongest. Throughout all five datasets there is a consistent representation of where in the world the data is coming from. The consistency only breaks in the newest dataset RiskAndVul, where the setup of the coordinates are different from the other datasets, though the data when extracted is still useful. In RiskAndVul the way the data has been save was: *“POINT (113.813 22.9175)”* and also with the possibility of empty data, where as the other datasets all setup the data as “(56.168393, 10.137373)” with no empty data.
Since the data is consistent over the different datasets we don’t need to consider this difference as long as we save the data so that when the coordinates are extracted as longitude and latitude numbers correctly.

City names and organization:
Looking at the data where both a column of the organization names and city names, can we see that each organization appears to be a government organization tied directly to the city it resides in.

Reduc16 and Reduc17:
Looking at the data within these two datasets and their names, we can see that they are of the same type of report separated by a year. We can however also see that the dataset from 2017 includes new columns of data which separates them from being completely identical to each other.

Formulation:

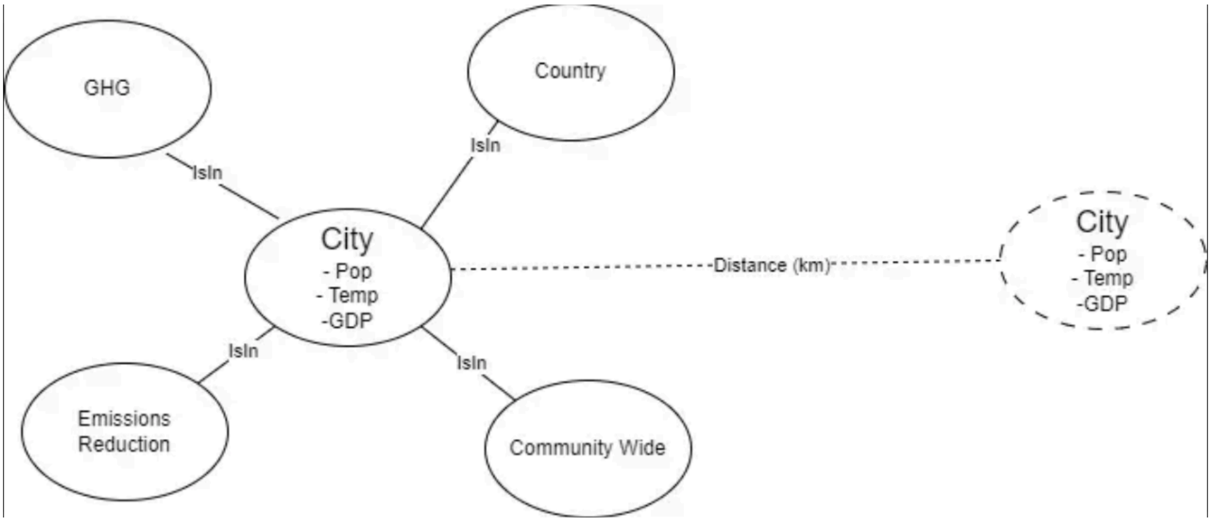
Here we will give a little overview of how we have chosen to set up our database.

Replicated:

We have decided to use transactional replication, primarily for load balancing and high availability. Load balancing can improve performance by distributing operations across multiple servers. Creating several replicas of the main database, ensures minimal downtime in the event of a primary database failure, which improves availability.

We have set up our database to use transactional replication, with a publisher/distributor server and 3 subscriber servers. We have initially created a snapshot of the publisher database, which is replicated to the subscribers, and from this point the database is set up to replicate any transaction received by the publisher to the subscribers as well. We have achieved this by creating separate users with specific roles and access rights, for example the snapshot user has full control, where the logreader only has read access, ensuring that the replication process operates smoothly and securely without compromising the integrity of the data or the performance of the system. This is set up in the Server Agent to handle the continued replication.

Design and develop proper database structure of the requested type.



Ingest the data into the database, including pre-processing of it, if necessary.

Here we will discuss how we introduce the data from the five datasets into our database. We will go over any data that we may or may not exclude, due to for example being unnecessary, and we will go over any data that may be the same in different datasets but with different data value types and how we will handle it.

Deciding include/exclude:

The first thing we need to push to our database is the organization's names and account number. We first need to determine whether or not we will include organizations with only an account number and no name tied to it. The reason for this is because only four of five of the datasets include a column representing the organization's name whilst they all include account number. It is in the *GHG* dataset that we have no column containing the names.We would first push the four datasets with names included so that there's only one row per unique account number with a name tied to it, and thereafter we could push from *GHG* any account number that doesn't already exist within the database. This would ensure all the data from the different datasets are included but could also lead to incomplete data when pulled for use. There is also the chance that we will find no new account number in *GHG* which hadn’t already been pushed before from the other datasets. It should be noted that while they don’t all contain the data of the names they do all include the name of the city it resides in, meaning the organization can be identified later if the name of the city and the country the city resides within are included. We have chosen to store organizations and cities together in one table called “City” with a primary key from the account number. This ensures that all account numbers are present even if they don’t contain all of the names associated, such as organization name, city name or city short name.

In the datasets *Reduc17* and *RiskAndVul* we see the column “Access” which we view as showing whether or not the general public have access to the report. The data within both datasets is however always “Public” which means we don’t need to push this data to the database as we can generalize that all data, from all five datasets, pushed and pulled to the database is public. We have however chosen to still include it in our database because we have determined it would be better for future proofing the database. While no data currently have any other value than “Public”, we would rather not run the risk of future reports missing this important difference should there be any that contain the value “Private”.

When we have to handle the overlapping data between the datasets we need to determine how we will handle each report type from the datasets. *Reduc16* and *Reduc17* both are the same type of report with the latter including new columns. With them being literally the same report but from different years mean we could store them together in the same table. This however would introduce many empties for the *Reduc16* data arriving from the new columns introduced from *Reduc17*. We have chosen to store the reports from the two datasets together in one table within our database.

Translating data type to new type:

For nearly every instance where a column contains data representing a year of the calendar they have a single integer number, but there are instances where the data type is represented by a variable type of date. This would introduce conflicts when pushing the data to the database. When deciding to translate an integer number to a date or a date to an integer number, we need to determine if the added information from the data that already exists as a date is important enough to store as translating an integer would not give us any new information regarding month, day or time. The added data wouldn’t be of much use considering that the reports are yearly based, so even if there are multiple reports from the same organizations there wouldn’t be multiple of the same target focus during one year. We can therefore discard the information of month, day and time and push only the year to our database.

In the dataset *RiskAndVul* we can see the column C40 being of type boolean whereas in other datasets with the column C40 being a string which can be empty. Since the string data are always empty or contain a string of “C40” then we can translate the strings to a boolean where an empty string equals false and a string of “C40” equals true.

Design and develop operations for maintenance of the database.

Considering that the type of data we will be storing within our database is not something that should change over time after it is first uploaded to our database and that any reports stored shouldn't be deleted unless there is a specific call to it, say if an organization is removed or replaced due to politics within a country or city then it would not make much sense to spend time or resources within our database, constantly checking the data within each table for value changes. Instead should any data need changing or deletion then a person should be overseeing the process.
Therefore we believe that time and resources would be better spent optimizing the speed for the request it would receive during its up time.

Backup/restore:

We have created a simple script to easily backup the database or optionally restore it when necessary. This script could be set up to be run at certain intervals or when changes are made to the database. However, as database backup procedures can be quite costly and changes to the database are not likely to be very frequent and should be overseen by a person anyway, it might be more efficient in this case to simply run this script manually when needed.

Indexing:

Considering the data from the datasets is mainly focused on where in the world it is coming from, we see a lot of data based on string values, primarily names. We can expect that a lot of data requests that our database would receive would be based on these names, so setting up text indexing based on the different names of the organizations, cities, countries would speed up the request time greatly:

```
CREATE TEXT INDEX node_text_index_city_name FOR (n:City) ON (n.city_name)
```

Creating range indexes for the account numbers, would also speed up the request time:

```
CREATE INDEX node_range_index_account_no FOR (n:City) ON (n.account_no)
```

The same can also be done on each of the different types of report nodes linked to the cities. Here it would mainly be on the account numbers which could be grouped based on their account number to speed up the gathering of report data for requests.

We now have to consider when we will maintain the indexing within our database with respect to when new data could be uploaded and when it is uploaded. Considering the type of data within the datasets, we can see that most cities uploaded yearly reports, with a few uploading more than once per year if the area of data gathering was different. Then we would consider the number of cities and countries that appear to be part of these reports.

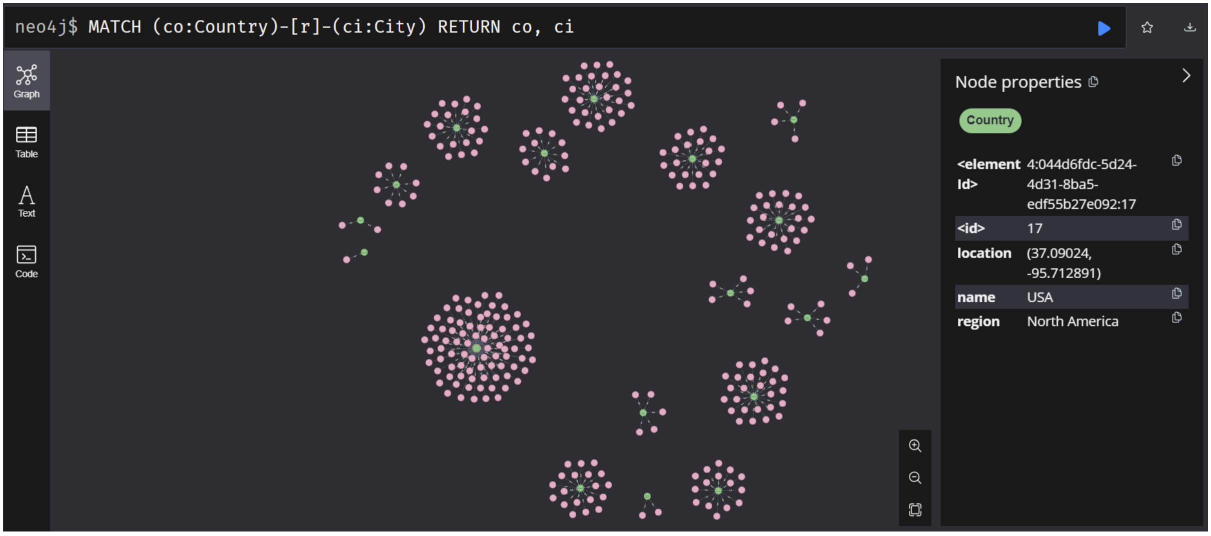
Overall we believe that indexing when new data is uploaded would be the best solution as the number of uploads is relatively few in comparison to how many requests there could appear should the database be used in a public scenario.

Formulate ten relevant questions for extracting information from the database, design and develop database functionality for implementing the information extraction.

1. All countries with their cities

A very simple match statement can provide the desired result for this question, however it gives a really clear insight over which countries have the most cities registered in the database. As seen in the image below, North America (cluster in the bottom left side) clearly dominates in numbers of cities in the database.

```
MATCH (co:Country)-[r]-(ci:City)
RETURN co, ci
```



```
neo4j$ MATCH (co:Country)-[r]-(ci:City) RETURN co, ci
```

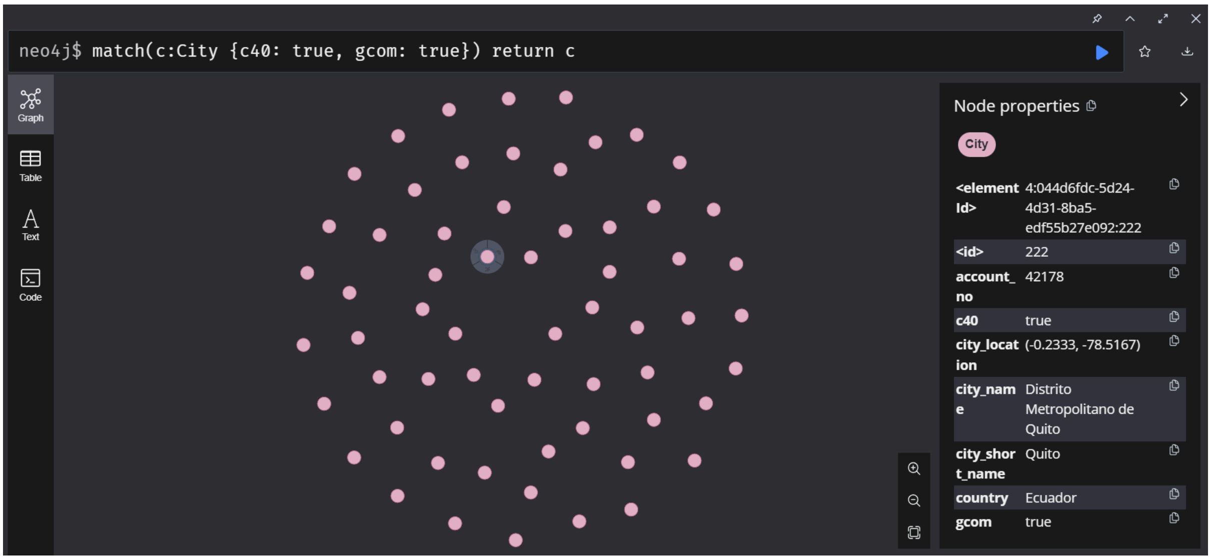
Node properties

Country

<element Id>	4:044d6fdc-5d24-4d31-8ba5-edf55b27e092:17
<id>	17
location	(37.09024, -95.712891)
name	USA
region	North America

2. All C40/GCOM city:

```
match(c:City {c40: true, gcom: true}) return c
```



```
neo4j$ match(c:City {c40: true, gcom: true}) return c
```

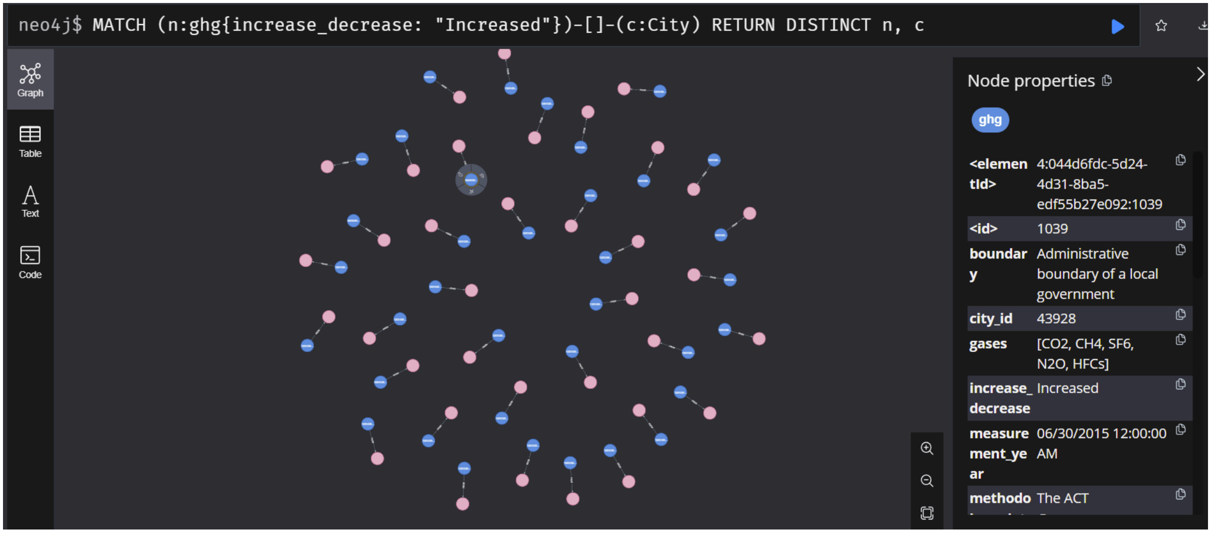
Node properties

City

<elementId>	4:044d6fdc-5d24-4d31-8ba5-edf55b27e092:222
<id>	222
account_no	42178
c40	true
city_location	(-0.2333, -78.5167)
city_name	Distrito Metropolitano de Quito
city_short_name	Quito
country	Ecuador
gcom	true

3. Decrease/Increase by city:

```
MATCH (n:ghg{increase_decrease: "Increased"})-[]-(c:City)
RETURN DISTINCT n, c
```



```
neo4j$ MATCH (n:ghg{increase_decrease: "Increased"})-[]-(c:City) RETURN DISTINCT n, c
```

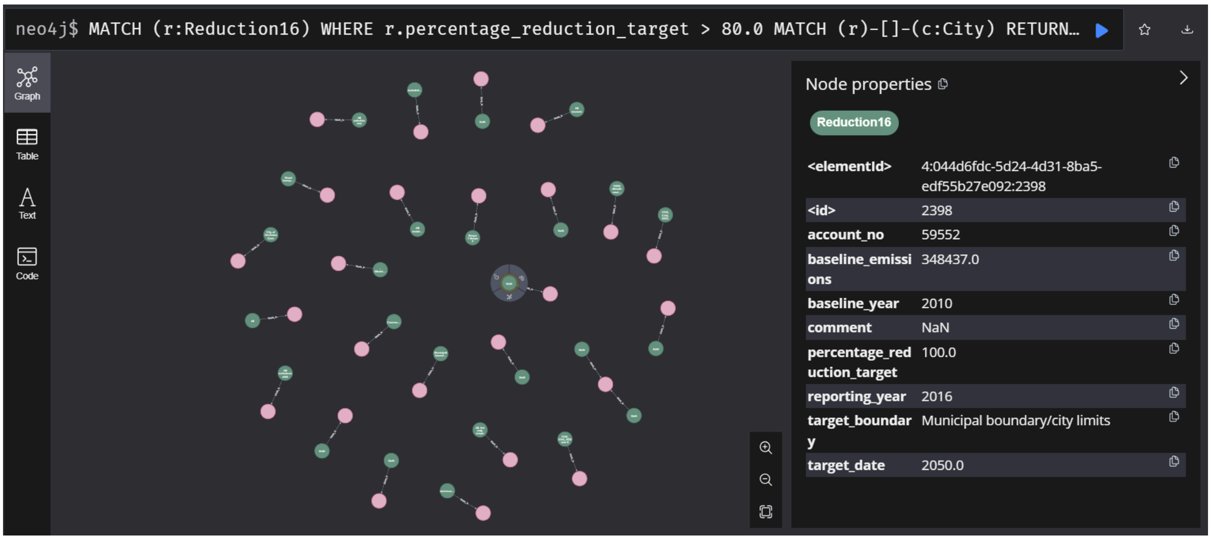
Node properties

ghg

<elementId>	4:044d6fdc-5d24-4d31-8ba5-edf55b27e092:1039
<id>	1039
boundary	Administrative boundary of a local government
city_id	43928
gases	[CO2, CH4, SF6, N2O, HFCs]
increase_decrease	Increased
measurement_year	06/30/2015 12:00:00 AM
methodo	The ACT

4. 2016 Percentage reduction target over ~80%:

```
MATCH (r:Reduction16)
WHERE r.percentage_reduction_target > 80.0
MATCH (r)-[]-(c:City)
RETURN DISTINCT r, c
```



```
neo4j$ MATCH (r:Reduction16) WHERE r.percentage_reduction_target > 80.0 MATCH (r)-[]-(c:City) RETURN…
```

Node properties

Reduction16

<elementId>	4:044d6fdc-5d24-4d31-8ba5-edf55b27e092:2398
<id>	2398
account_no	59552
baseline_emissions	348437.0
baseline_year	2010
comment	NaN
percentage_reduction_target	100.0
reporting_year	2016
target_boundary	Municipal boundary/city limits
target_date	2050.0

5. Percentage reduction target under ~20% including comment:

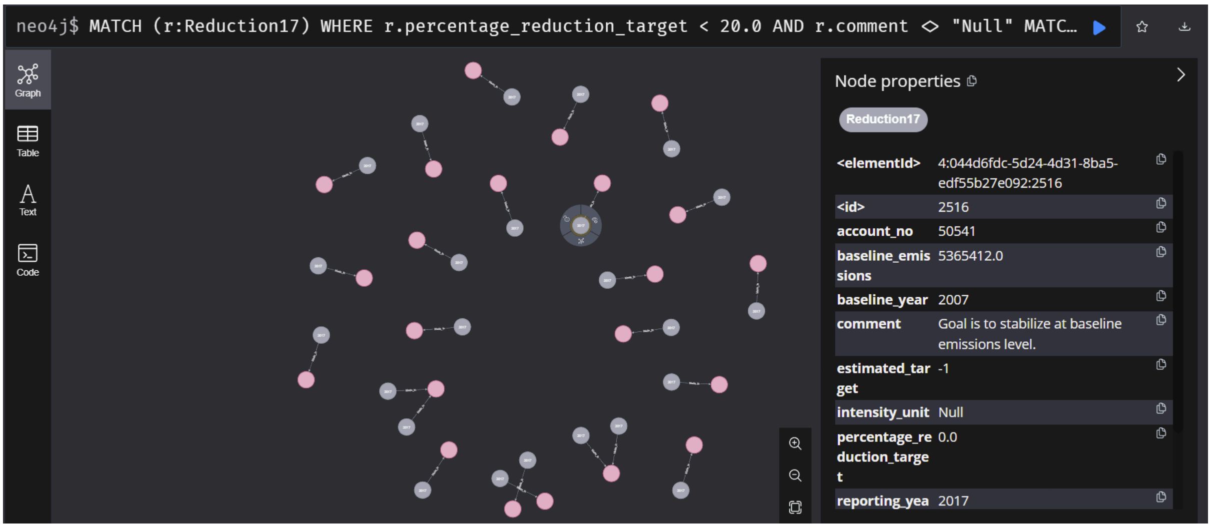
```
MATCH (r:Reduction17)
WHERE r.percentage_reduction_target < 20.0
AND r.comment <> "Null"
MATCH (r)-[]-(c:City)
RETURN DISTINCT r, c
```

It is very common to have a search related to null values. If for example a null value should not be included in a search result, a common syntax would be the following.

```
MATCH (e:Example)
WHERE e.value IS NOT NULL
RETURN DISTINCT e
```

In this database however, comments that don't have a value are written as a string saying null with uppercase, "Null". This is why it is necessary to check for a string value that equals "Null" using the following syntax like done in the database search above.

```
AND r.comment <> "Null"
```



```
neo4j$ MATCH (r:Reduction17) WHERE r.percentage_reduction_target < 20.0 AND r.comment <> "Null" MATC…
```

Node properties

Reduction17

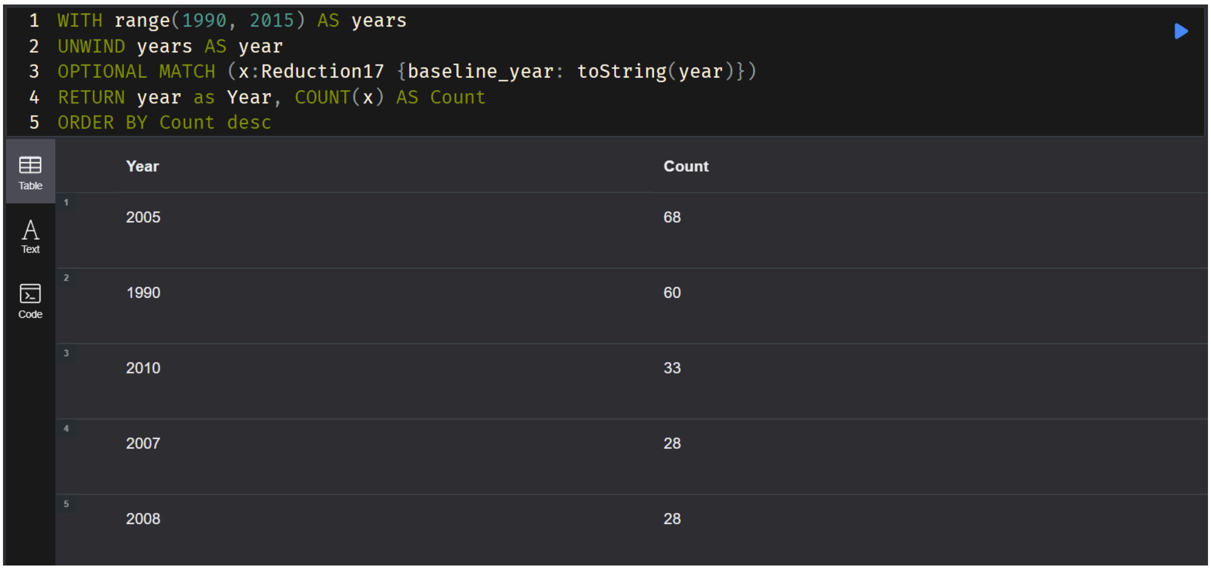
<elementId>	4:044d6fdc-5d24-4d31-8ba5-edf55b27e092:2516
<id>	2516
account_no	50541
baseline_emissions	5365412.0
baseline_year	2007
comment	Goal is to stabilize at baseline emissions level.
estimated_target	-1
intensity_unit	Null
percentage_reduction_target	0.0
reporting_yea	2017

6. How many entries from each baseline year

```
WITH range(1990, 2015) AS years
UNWIND years AS year
OPTIONAL MATCH (x:Reduction17 {baseline_year: toString(year)})
RETURN year as Year, COUNT(x) AS Count
ORDER BY Count desc
```

This code loops through all the years from 1990 to 2015 and counts how many entries have that specific baseline_line. We “Unwind” (in most programming languages known as loop) through the variable “years” which is a range from 1990 to 2015.
It then does an “OPTIONAL MATCH” to filter only the occurrences that match said year. A regular match looks through occurrences and if nothing is found, nothing is returned. However, an optional match doesn’t require at least one occurrence, so years with no entries, like 1998, still get included with a count of 0 (not shown in the

picture, but the year 1991 to 1997 all have a count of 0).

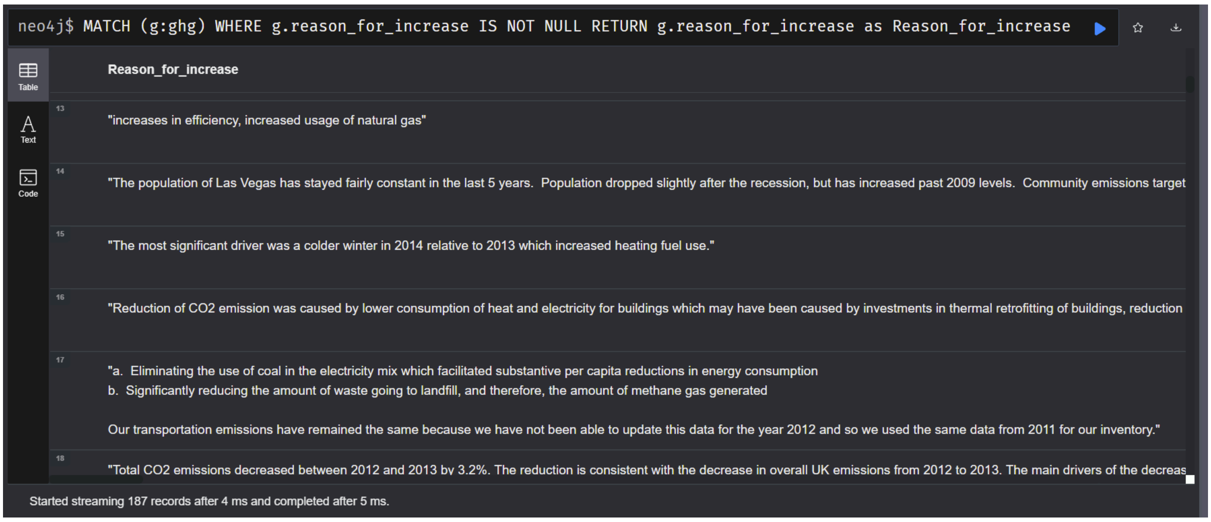


```
WITH range(1990, 2015) AS years
UNWIND years AS year
OPTIONAL MATCH (x:Reduction17 {baseline_year: toString(year)})
RETURN year as Year, COUNT(x) AS Count
ORDER BY Count desc
```

	Year	Count
1	2005	68
2	1990	60
3	2010	33
4	2007	28
5	2008	28

7. List of reasons for increase

```
MATCH (g:ghg)
WHERE g.reason_for_increase IS NOT NULL
RETURN g.reason_for_increase as Reason_for_increase
```



```
neo4j$ MATCH (g:ghg) WHERE g.reason_for_increase IS NOT NULL RETURN g.reason_for_increase as Reason_for_increase
```

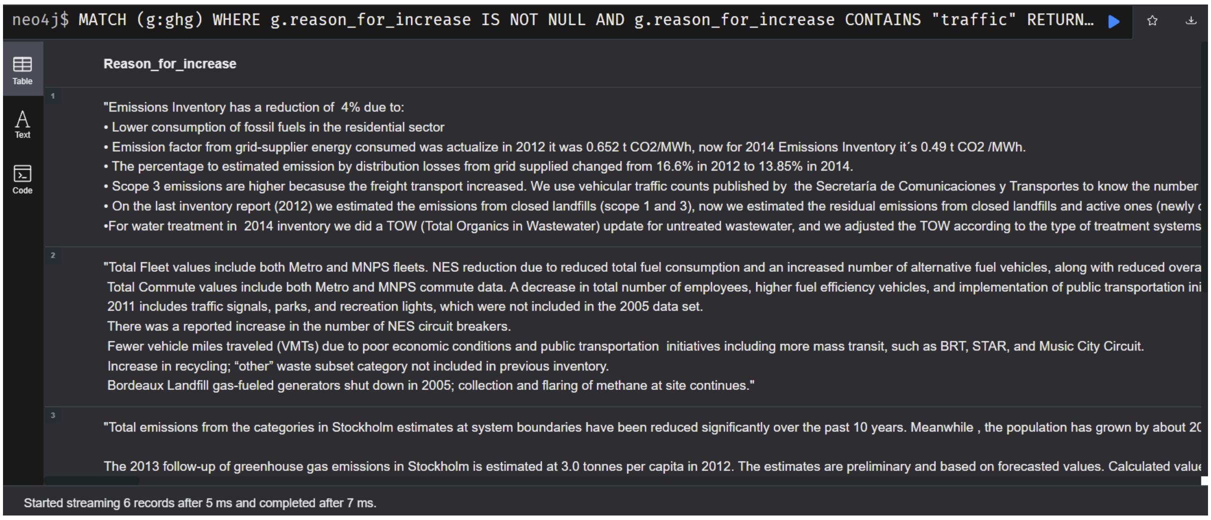
	Reason_for_increase
13	"increases in efficiency, increased usage of natural gas"
14	"The population of Las Vegas has stayed fairly constant in the last 5 years. Population dropped slightly after the recession, but has increased past 2009 levels. Community emissions target
15	"The most significant driver was a colder winter in 2014 relative to 2013 which increased heating fuel use."
16	"Reduction of CO2 emission was caused by lower consumption of heat and electricity for buildings which may have been caused by investments in thermal retrofitting of buildings, reduction
17	"a. Eliminating the use of coal in the electricity mix which facilitated substantive per capita reductions in energy consumption b. Significantly reducing the amount of waste going to landfill, and therefore, the amount of methane gas generated Our transportation emissions have remained the same because we have not been able to update this data for the year 2012 and so we used the same data from 2011 for our inventory."
18	"Total CO2 emissions decreased between 2012 and 2013 bv 3.2%. The reduction is consistent with the decrease in overall UK emissions from 2012 to 2013. The main drivers of the decreas

Started streaming 187 records after 4 ms and completed after 5 ms.

8. List of reason including a keyword

```
MATCH (g:ghg)
WHERE g.reason_for_increase IS NOT NULL
AND g.reason_for_increase CONTAINS "traffic"
RETURN g.reason_for_increase as Reason_for_increase
```

This is very similar to the previous search, however this includes a condition of a specific word or phrase appearing. As shown in the bottom of the picture, 6 entries mention “traffic” in the reason for the increase.

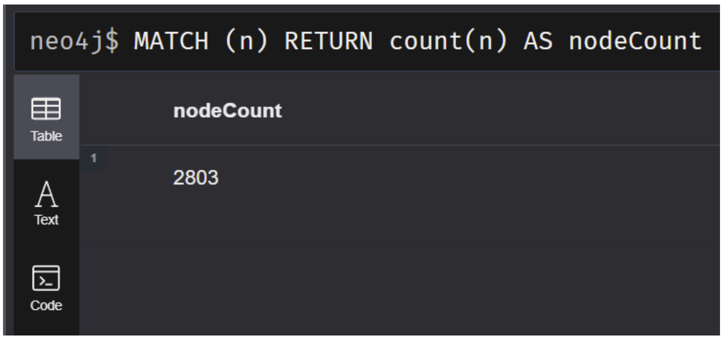


```
neo4j$ MATCH (g:ghg) WHERE g.reason_for_increase IS NOT NULL AND g.reason_for_increase CONTAINS "traffic" RETURN…
```

```
Reason_for_increase

1   "Emissions Inventory has a reduction of  4% due to:
    • Lower consumption of fossil fuels in the residential sector
    • Emission factor from grid-supplier energy consumed was actualize in 2012 it was 0.652 t CO2/MWh, now for 2014 Emissions Inventory it´s 0.49 t CO2 /MWh.
    • The percentage to estimated emission by distribution losses from grid supplied changed from 16.6% in 2012 to 13.85% in 2014.
    • Scope 3 emissions are higher becasuse the freight transport increased. We use vehicular traffic counts published by  the Secretaría de Comunicaciones y Transportes to know the number
    • On the last inventory report (2012) we estimated the emissions from closed landfills (scope 1 and 3), now we estimated the residual emissions from closed landfills and active ones (newly
    •For water treatment in  2014 inventory we did a TOW (Total Organics in Wastewater) update for untreated wastewater, and we adjusted the TOW according to the type of treatment systems

2   "Total Fleet values include both Metro and MNPS fleets. NES reduction due to reduced total fuel consumption and an increased number of alternative fuel vehicles, along with reduced overa
     Total Commute values include both Metro and MNPS commute data. A decrease in total number of employees, higher fuel efficiency vehicles, and implementation of public transportation ini
     2011 includes traffic signals, parks, and recreation lights, which were not included in the 2005 data set.
     There was a reported increase in the number of NES circuit breakers.
     Fewer vehicle miles traveled (VMTs) due to poor economic conditions and public transportation  initiatives including more mass transit, such as BRT, STAR, and Music City Circuit.
     Increase in recycling; “other” waste subset category not included in previous inventory.
     Bordeaux Landfill gas-fueled generators shut down in 2005; collection and flaring of methane at site continues."

3   "Total emissions from the categories in Stockholm estimates at system boundaries have been reduced significantly over the past 10 years. Meanwhile , the population has grown by about 20

    The 2013 follow-up of greenhouse gas emissions in Stockholm is estimated at 3.0 tonnes per capita in 2012. The estimates are preliminary and based on forecasted values. Calculated value

Started streaming 6 records after 5 ms and completed after 7 ms.
```

Validate and test all database operations.

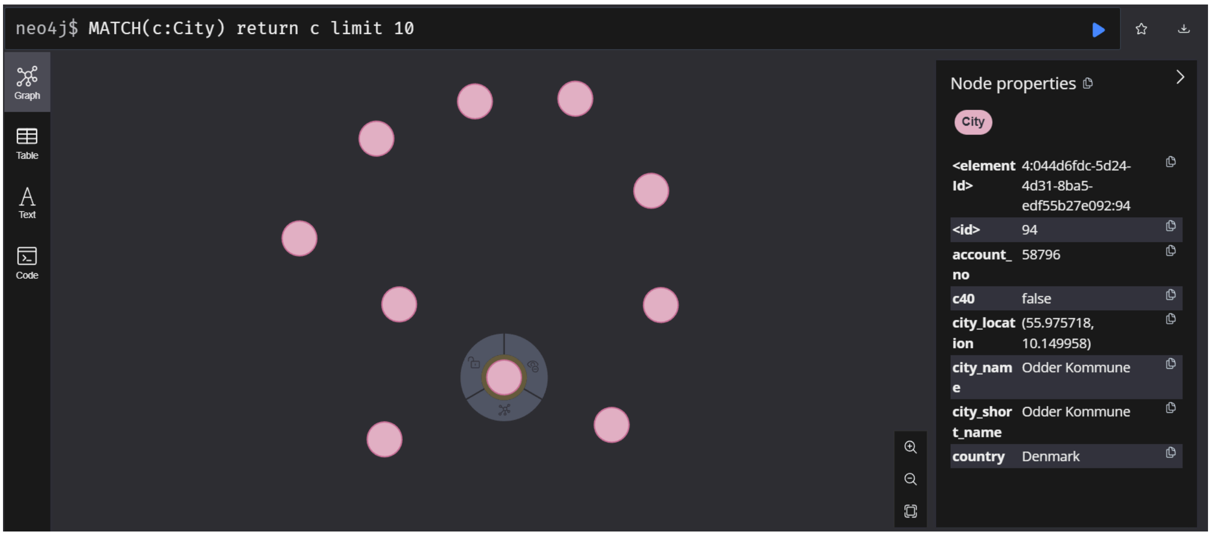
It is important to validate and test the database to make sure the data has been inserted correctly. In order to do that, it is necessary to run a few common database operations and see if the result of the queries seem realistic. The first query is a simple count, to see how many nodes exist in the database. As shown in the picture on the right, there's 2803 total nodes, which seems accurate enough, when adding together all the cities, countries, reports and so on.



```
neo4j$ MATCH (n) RETURN count(n) AS nodeCount
```

	nodeCount
1	2803

To make sure the nodes actually hold the data they're supposed to, we'll extract a few nodes and see what data they hold and whether it seems correct.



```
neo4j$ MATCH(c:City) return c limit 10
```

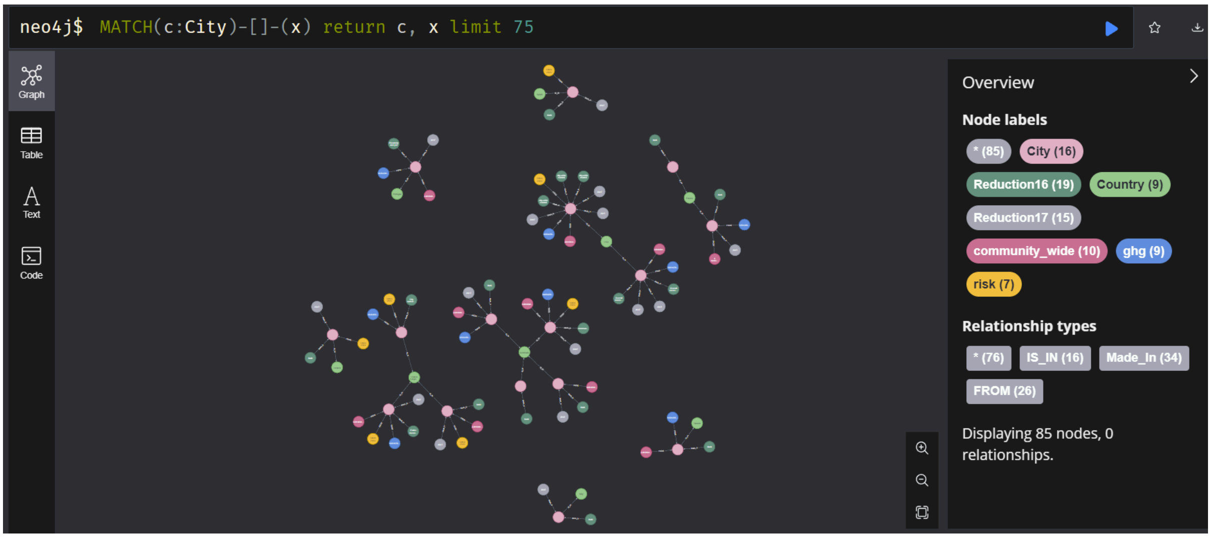
Node properties

City

<elementId>	4:044d6fdc-5d24-4d31-8ba5-edf55b27e092:94
<id>	94
account_no	58796
c40	false
city_location	(55.975718, 10.149958)
city_name	Odder Kommune
city_short_name	Odder Kommune
country	Denmark

As shown in the picture, we extract 10 cities using a fairly simple match statement. When taking a look at the node representing "Odder Kommune", the attributes of the node contain legit data. The city nodes appear to be correctly ingested, but in Neo4j, the relations between the nodes are an absolute necessity too.

In order to test the relations between the nodes, we'll query a few nodes with all their relations to other nodes and check whether it seems to be in order.



```
neo4j$ MATCH(c:City)-[]-(x) return c, x limit 75
```

Overview

Node labels

* (85) City (16) Reduction16 (19) Country (9) Reduction17 (15) community_wide (10) ghg (9) risk (7)

Relationship types

* (76) IS_IN (16) Made_In (34) FROM (26)

Displaying 85 nodes, 0 relationships.

This query returns cities with their related nodes with a limit of 75. As seen on the right, the result includes an evenly distributed result of different node types. There's a handful of cities, countries, risk reports, ghg and so on. It is also shown well in the visualization of the nodes, how the nodes are related and very well connected with each other.

Through these three queries, it is possible to see that the total number of nodes is accurate, the nodes contain data correctly and the relationships between the nodes are set up as they're supposed to. Overall, this validation and these tests show that the database is set up correctly and accurately, with all the data and relations included.

Evaluate the database’s performance and suggest measures for improving it.

To test the database we will be using the queries used for the questions to test the performance. We will be using the profiler to get a better understanding of executions of the queries.

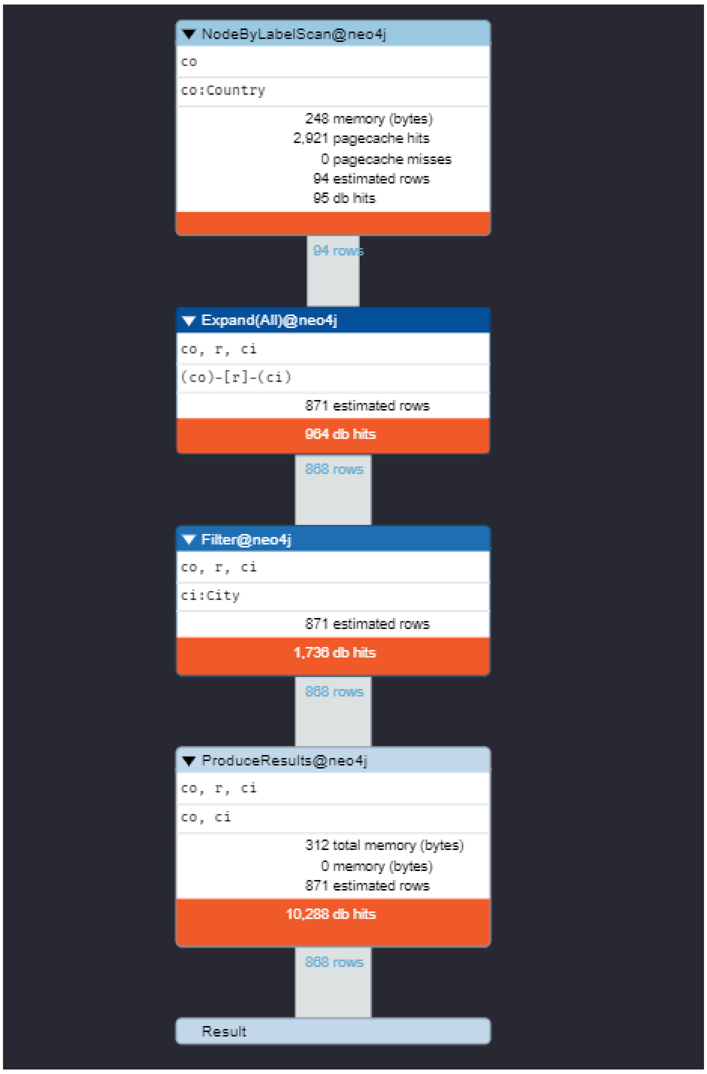
1. All countries with their cities

Total execution time: 4ms

Query:

```
MATCH (co:Country)-[r]-(ci:City)
RETURN co, ci
```

Execution plan:



```
▼ NodeByLabelScan@neo4j
co
co:Country
248 memory (bytes)
2,921 pagecache hits
0 pagecache misses
94 estimated rows
95 db hits

94 rows

▼ Expand(All)@neo4j
co, r, ci
(co)-[r]-(ci)
871 estimated rows
964 db hits

868 rows

▼ Filter@neo4j
co, r, ci
ci:City
871 estimated rows
1,736 db hits

868 rows

▼ ProduceResults@neo4j
co, r, ci
co, ci
312 total memory (bytes)
0 memory (bytes)
871 estimated rows
10,288 db hits

868 rows

Result
```

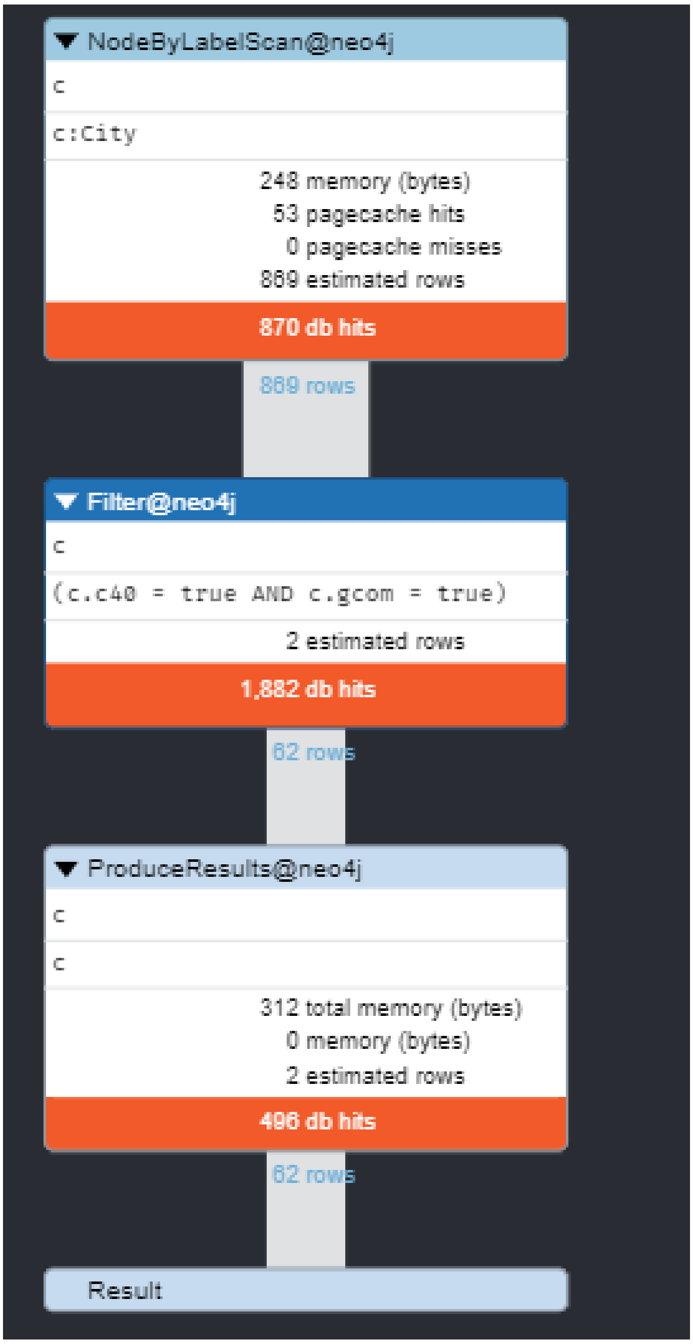
2. All C40/GCOM city:

Total execution time: 2ms

Query:

```
MATCH (co:Country)-[r]-(ci:City)
RETURN co, ci
```

Execution plan:



```
▼ NodeByLabelScan@neo4j
c
c:City
248 memory (bytes)
53 pagecache hits
0 pagecache misses
869 estimated rows
870 db hits
869 rows

▼ Filter@neo4j
c
(c.c40 = true AND c.gcom = true)
2 estimated rows
1,882 db hits
62 rows

▼ ProduceResults@neo4j
c
c
312 total memory (bytes)
0 memory (bytes)
2 estimated rows
496 db hits
62 rows

Result
```

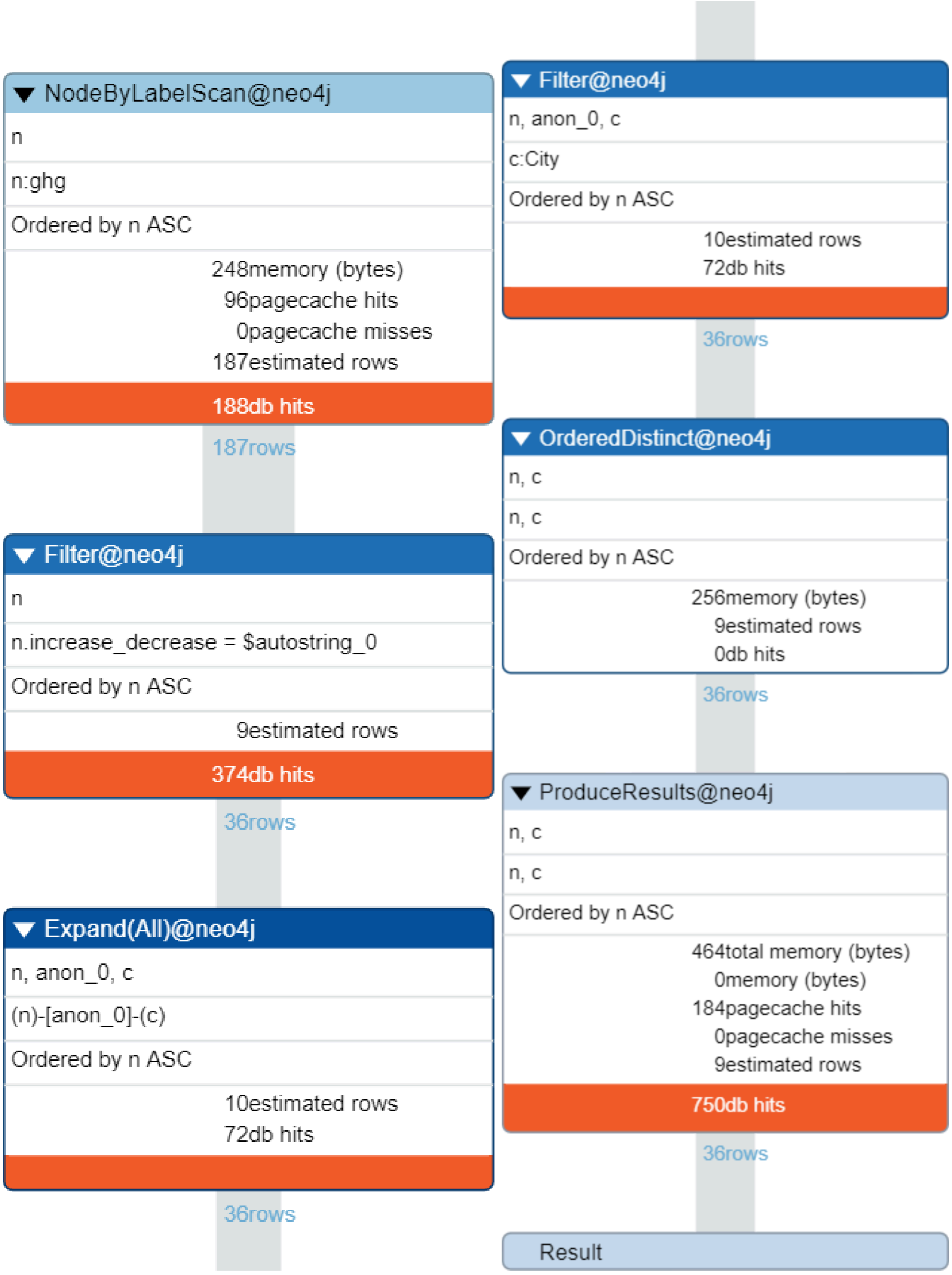
3. Decrease/Increase by city:

Total execution time: 2ms

Query:

```
MATCH (n:ghg{increase_decrease: "Increased"})-[]-(c:City)
RETURN DISTINCT n, c
```

Execution plan:



NodeByLabelScan@neo4j

n

n:ghg

Ordered by n ASC

248memory (bytes)
96pagecache hits
0pagecache misses
187estimated rows
188db hits

187rows

Filter@neo4j

n

n.increase_decrease = $autostring_0

Ordered by n ASC

9estimated rows
374db hits

36rows

Expand(All)@neo4j

n, anon_0, c

(n)-[anon_0]-(c)

Ordered by n ASC

10estimated rows
72db hits

36rows

Filter@neo4j

n, anon_0, c

c:City

Ordered by n ASC

10estimated rows
72db hits

36rows

OrderedDistinct@neo4j

n, c

n, c

Ordered by n ASC

256memory (bytes)
9estimated rows
0db hits

36rows

ProduceResults@neo4j

n, c

n, c

Ordered by n ASC

464total memory (bytes)
0memory (bytes)
184pagecache hits
0pagecache misses
9estimated rows
750db hits

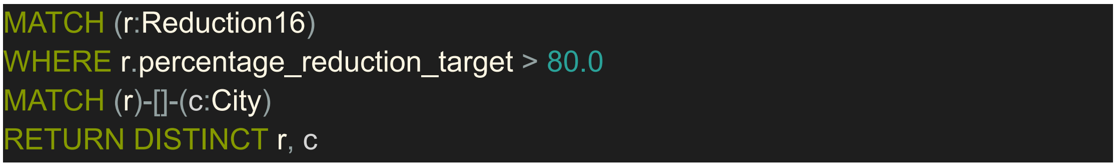
36rows

Result

4. 2016 Percentage reduction target over ~80%:

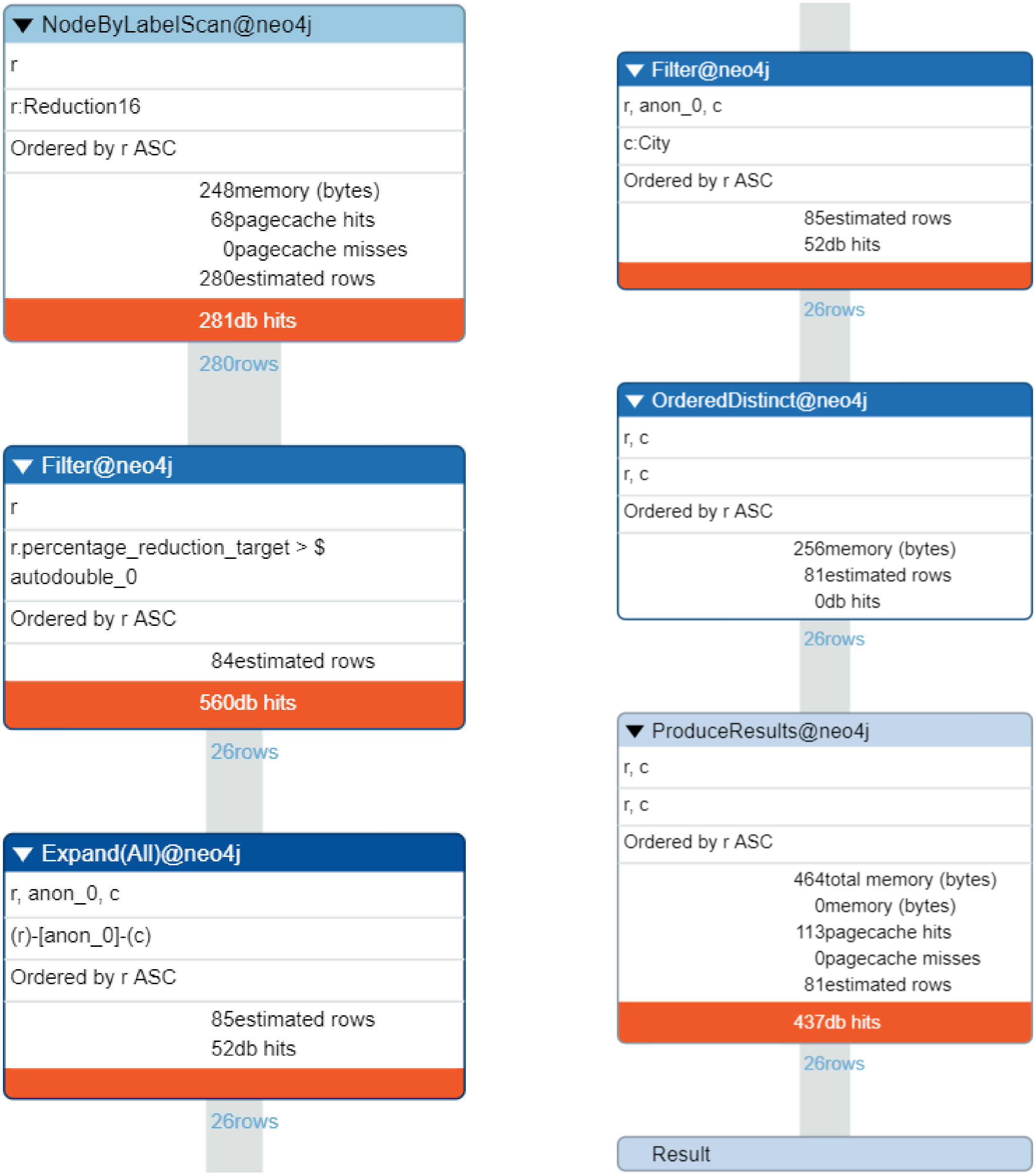
Total execution time: 2ms

Query:



```
MATCH (r:Reduction16)
WHERE r.percentage_reduction_target > 80.0
MATCH (r)-[]-(c:City)
RETURN DISTINCT r, c
```

Execution plan:



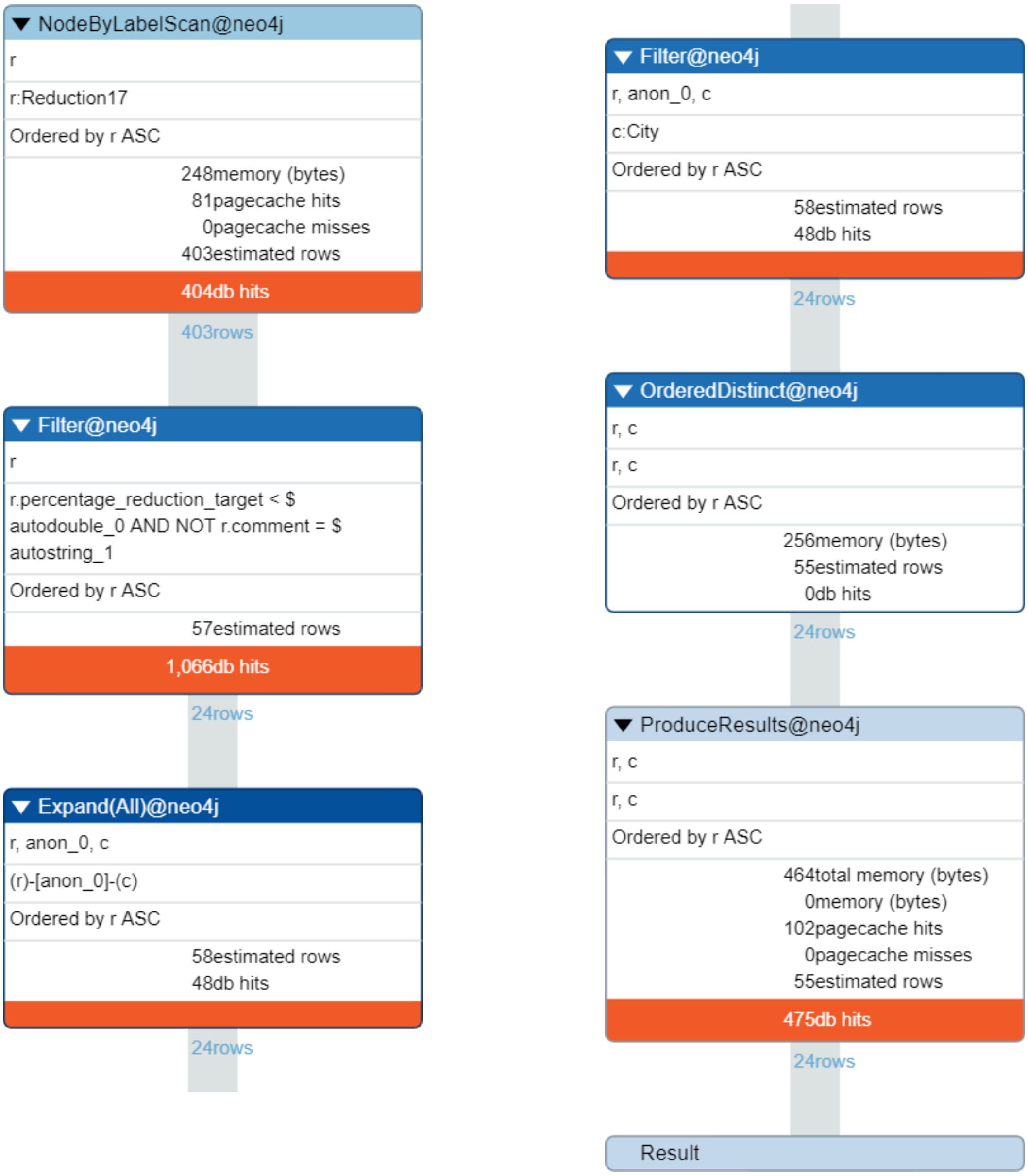
```
▼ NodeByLabelScan@neo4j
r
r:Reduction16
Ordered by r ASC
248memory (bytes)
68pagecache hits
0pagecache misses
280estimated rows
281db hits
280rows

▼ Filter@neo4j
r
r.percentage_reduction_target > $
autodouble_0
Ordered by r ASC
84estimated rows
560db hits
26rows

▼ Expand(All)@neo4j
r, anon_0, c
(r)-[anon_0]-(c)
Ordered by r ASC
85estimated rows
52db hits
26rows

▼ Filter@neo4j
r, anon_0, c
c:City
Ordered by r ASC
85estimated rows
52db hits
26rows

▼ OrderedDistinct@neo4j
r, c
r, c
Ordered by r ASC
256memory (bytes)
81estimated rows
0db hits
26rows

▼ ProduceResults@neo4j
r, c
r, c
Ordered by r ASC
464total memory (bytes)
0memory (bytes)
113pagecache hits
0pagecache misses
81estimated rows
437db hits
26rows

Result
```

5. Percentage reduction target under ~20% including comment:

Total execution time: 3ms
Query:

```
MATCH (r:Reduction17)
WHERE r.percentage_reduction_target < 20.0
AND r.comment <> "Null"
MATCH (r)-[]-(c:City)
RETURN DISTINCT r, c
```

Execution plan:



NodeByLabelScan@neo4j

r

r:Reduction17

Ordered by r ASC

248memory (bytes)
81pagecache hits
0pagecache misses
403estimated rows

404db hits

403rows

Filter@neo4j

r

r.percentage_reduction_target < $ autodouble_0 AND NOT r.comment = $ autostring_1

Ordered by r ASC

57estimated rows

1,066db hits

24rows

Expand(All)@neo4j

r, anon_0, c

(r)-[anon_0]-(c)

Ordered by r ASC

58estimated rows
48db hits

24rows

Filter@neo4j

r, anon_0, c

c:City

Ordered by r ASC

58estimated rows
48db hits

24rows

OrderedDistinct@neo4j

r, c

r, c

Ordered by r ASC

256memory (bytes)
55estimated rows
0db hits

24rows

ProduceResults@neo4j

r, c

r, c

Ordered by r ASC

464total memory (bytes)
0memory (bytes)
102pagecache hits
0pagecache misses
55estimated rows

475db hits

24rows

Result

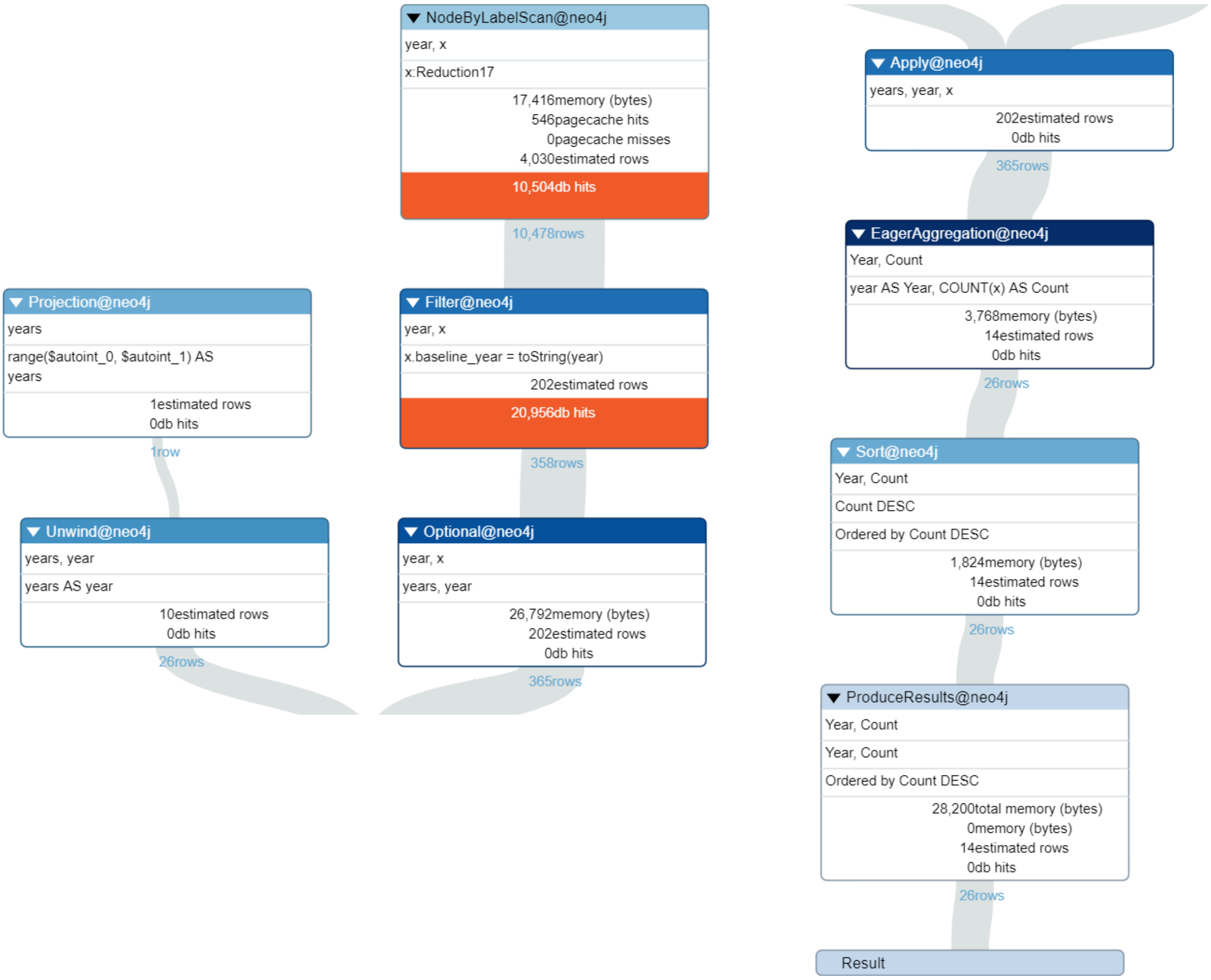
6. How many entries from each baseline year

Total execution time: 7ms

Query:

```
WITH range(1990, 2015) AS years
UNWIND years AS year
OPTIONAL MATCH (x:Reduction17 {baseline_year: toString(year)})
RETURN year as Year, COUNT(x) AS Count
ORDER BY Count desc
```

Execution plan:



▼ Projection@neo4j
years
range($autoint_0, $autoint_1) AS years
1estimated rows
0db hits

1row

▼ Unwind@neo4j
years, year
years AS year
10estimated rows
0db hits

26rows

▼ NodeByLabelScan@neo4j
year, x
x:Reduction17
17,416memory (bytes)
546pagecache hits
0pagecache misses
4,030estimated rows
10,504db hits

10,478rows

▼ Filter@neo4j
year, x
x.baseline_year = toString(year)
202estimated rows
20,956db hits

358rows

▼ Optional@neo4j
year, x
years, year
26,792memory (bytes)
202estimated rows
0db hits

365rows

▼ Apply@neo4j
years, year, x
202estimated rows
0db hits

365rows

▼ EagerAggregation@neo4j
Year, Count
year AS Year, COUNT(x) AS Count
3,768memory (bytes)
14estimated rows
0db hits

26rows

▼ Sort@neo4j
Year, Count
Count DESC
Ordered by Count DESC
1,824memory (bytes)
14estimated rows
0db hits

26rows

▼ ProduceResults@neo4j
Year, Count
Year, Count
Ordered by Count DESC
28,200total memory (bytes)
0memory (bytes)
14estimated rows
0db hits

26rows

Result

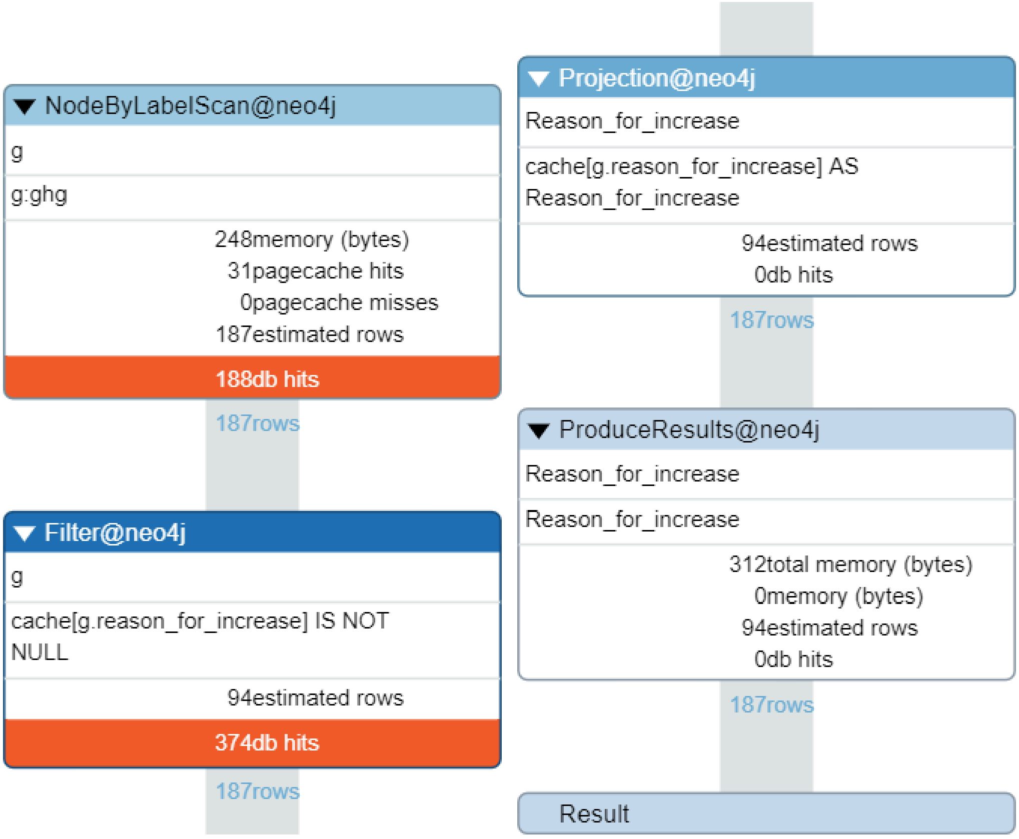
7. List of reasons for increase

Total execution time: 1ms

Query:

```
MATCH (g:ghg)
WHERE g.reason_for_increase IS NOT NULL
RETURN g.reason_for_increase as Reason_for_increase
```

Execution plan:



```
▼ NodeByLabelScan@neo4j
g
g:ghg
248memory (bytes)
31pagecache hits
0pagecache misses
187estimated rows
188db hits
187rows

▼ Filter@neo4j
g
cache[g.reason_for_increase] IS NOT
NULL
94estimated rows
374db hits
187rows

▼ Projection@neo4j
Reason_for_increase
cache[g.reason_for_increase] AS
Reason_for_increase
94estimated rows
0db hits
187rows

▼ ProduceResults@neo4j
Reason_for_increase
Reason_for_increase
312total memory (bytes)
0memory (bytes)
94estimated rows
0db hits
187rows

Result
```

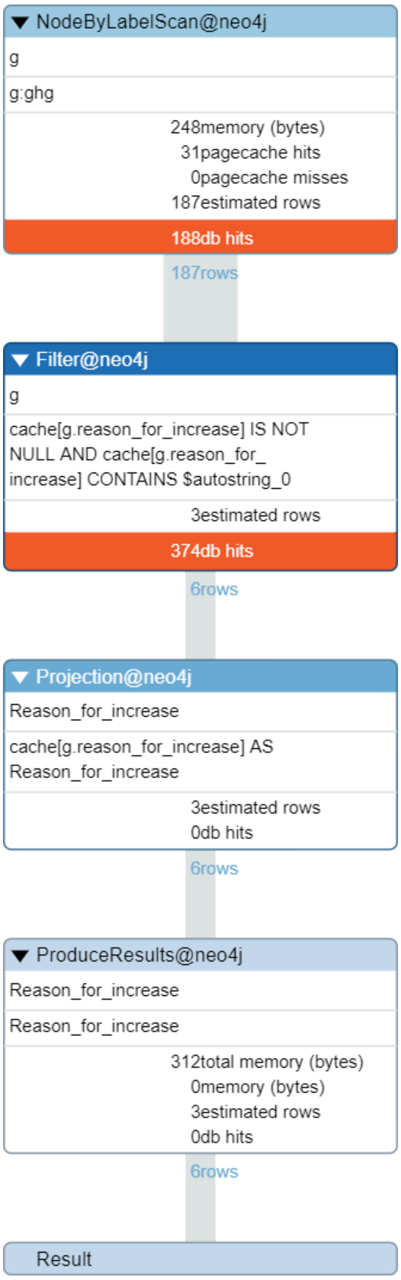
8. List of reasons for increased with keyword

Total execution time: 2ms

Query:

```
MATCH (g:ghg)
WHERE g.reason_for_increase IS NOT NULL
AND g.reason_for_increase CONTAINS "traffic"
RETURN g.reason_for_increase as Reason_for_increase
```

Execution plan:



```
▼ NodeByLabelScan@neo4j
g
g:ghg
248memory (bytes)
31pagecache hits
0pagecache misses
187estimated rows
188db hits
187rows

▼ Filter@neo4j
g
cache[g.reason_for_increase] IS NOT
NULL AND cache[g.reason_for_
increase] CONTAINS $autostring_0
3estimated rows
374db hits
6rows

▼ Projection@neo4j
Reason_for_increase
cache[g.reason_for_increase] AS
Reason_for_increase
3estimated rows
0db hits
6rows

▼ ProduceResults@neo4j
Reason_for_increase
Reason_for_increase
312total memory (bytes)
0memory (bytes)
3estimated rows
0db hits
6rows

Result
```

Summary of Performance

Overall the database performs very quickly. The longest query takes only 7ms which is query 6, “How many entries from each baseline year“. It is an expensive query due to the many executions and many hits which suggest resource heavy tasks. However a query of 7ms is very fast considering its the slowest one.

There is little to improve on the database to improve its performance. The high performance of the database is more due to the simplicity of the database. Nodes are rarely interconnected and do not go more than 3 levels deep. As all of the nodes are not connected to each other, but are formed by clusters, means we can not create very complex queries or .

Formulate conclusions and recommendations:

We have gone through the five datasets given to us and from those have explored the data they contained, had discussions and made our conclusions as to how we best could store the data within a database. From our conclusions regarding the datasets we have discussed our options for setting up a database where our conclusion is the result of our work. With a diagram drawn in DrawIO, of the structure of the database which we used to set up a local instance and created scripts for uploading the data from the five datasets into the database instance.

From there we have made discussions, evaluations and conclusions for future improvement recommendations that could be made to our database as well as some improvements which we have implemented into our database.