Design of a Chemical Graph Database to Query on an Atom-Specific Level

Master Thesis Data Science in Business & Entrepreneurship

1st of December 2022

Student:
Steven Drenth

Academic Supervisors:
Dr. Rogier Brussee – Main supervisor
Prof. Dr. Jakob de Vlieg – Second supervisor

Company supervisor:
Dr. Victor Sojo – supervisor BASF

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

JADS Jheronimus Academy of Data Science

TILBURG UNIVERSITY

Abstract

Most molecule-oriented databases allow many different queries. E.g., for the leading biomolecular database PubChem, molecules can be queried on the molecule name, its identifier code (e.g., ICSC number), the molecular formula, one of (possibly non unique) SMILES strings of the molecule, as well as “metadata” such as the studies where information can be found, physical, chemical, and biological properties. Additionally, there can be searched for different sub-, and superstructures of a given molecule as well as similar molecules. Most reaction-oriented databases are also limited in the queries you can do. For example, the KEGG database can be queried on molecule name, biochemical process name and pathway identifiers. However, querying on both molecular structural properties and/or reaction properties is hard because information from two different databases is needed. Queries are currently limited in that because to access the internal structure of molecules a lot of detailed information has to be queried that needs to be post-processed.

An important reason for this is that to represent the structural formula of molecules, digital formats such as SMILES, SMARTS, and MOLfiles are used. These formats “flatten” the geometric structure of structural formulas in a table format or linear string that is convenient to store in relational databases but that post processing must reconstruct. Therefore, people often manually filter the visual representations of a structural formula. To solve this problem, this thesis presents a chemical graph database: a database that enables the user to query on an atom-specific level; queries can be run on specific parts of the reactions, molecules, bonds, atoms, rings, and their corresponding features. A graph database is chosen because a graph comes the closest to the structural formula representation of a molecule. Additionally, the graph database is also a good match for the reaction networks between molecules. Thus, reaction networks and molecules can be seamlessly integrated in the same database. Lastly, the query language of the graph database is easy to understand, and complex operations that access the reaction networks, molecular structure and metadata information can be performed with only a few query lines. The chemical graph database has been validated by testing queries on an atom-specific level that performed satisfactorily on ease of use and query length. Additionally, the data in the chemical graph database was transformed into three other data formats, which is possible with a few additional lines of python code per category.

Table of contents

1 Preface

To start, I would like to take this opportunity to thank some people. First, I'd like to thank Rogier Brussee, my first supervisor, for guiding me through the process, especially at the moments when I really needed it. I would also like to thank Jakob de Vlieg for his enthusiasm in the subject from the beginning and for being my second supervisor during difficult personal circumstances. Secondly, I'd like to thank BASF for letting me write my thesis at their location in Berlin. In the past six months, I got to know a lot of nice and enthusiastic people, which I really appreciate. A special thanks to Lars Mohren, who put a lot of effort in getting me to Berlin, and Victor Sojo, my company supervisor, who helped me in any way possible: coding, chemistry, English grammatic or just some random facts about basically anything, I have learned a lot of new things in the past months. Lastly, I would like to thank my family and friends for their support and for letting me forget, in a positive way, my thesis for some moments. I'd like to thank my lovely girlfriend, who also did this and made sure I did not starve to death or become one with my computer screen in especially the last couple of weeks.

Now, let's start with the thesis. As you might have guessed from the title and the abstract, this thesis has, in a positive way, a chemical taste to it. Doing my thesis at a chemical company was for me, as someone who had his last chemistry lessons in high school, a re-introduction to chemistry. Especially in the first months, I was (re)learning something new about chemistry every day. To correctly execute a project, it is important to generally understand with what information I'm working, but after all, I am doing a master's in data science and not chemistry. Therefore, the approach in this thesis will be that of a data scientist who tries to understand as much chemistry as possible. A significant amount of analysis research on chemical data from a data science perspective had already been done and this was a good start for me to get into the topic.

2 Introduction and problem definition

When studying different research of chemical data, I learned more about the chemical reactions, molecules, and their features as well as the different formats that represent them. It almost seems like every study is using different chemical data and/or representations of this data. Even when studies use the same data and representations, it is not always clear if the same features are used in both studies. However, two databases come back in a significant number of papers: the PubChem[1] database for molecule data and the KEGG[2] database for metabolic pathway[3] data. With the curiousness a data scientist has, I wanted to know how easy it is to retrieve chemical data from these databases.

The PubChem database allows you to search for molecules using different kinds of input, like the molecule name or molecular formula, and the returned molecule can be represented in different formats. Of the returned molecule, similar molecules or molecules that have a (partly) overlap can be found. However, you want to filter more specifically on the molecules, this is not possible, and a human is needed to perform this. For example, when you would like to filter the molecules on having 2 or fewer chemical rings, the database cannot do this, and human interpretation is needed. The KEGG database presents metabolic pathways sorted by category. When a pathway is selected, all molecules and their corresponding reactions are shown as a picture with a clickable link. However, searching for similar reactions or specific changes caused by a reaction is not possible. For a human, this is relatively easy to interpret for each individual study, but it would take a lot of time to do this for all of them.

The paragraph above already gives a lot of information for an introduction part, but it is important to show it as this will bring us to our problem definition. Because why do even the well-known databases, which are mentioned in a significant number of papers, not facilitate more specific filtering? When a human would have to do this, it is not possible for all available chemicals, so filtering on a selection of the data is needed, which can cause output to be left out. Having a solution for specific filtering/querying would save a significant amount of time and result in more information gain. This is not only applicable to the databases mentioned, but also to the chemical databases within a company. This solution can have a serious business impact as more data can be retrieved using significantly fewer resources. Therefore, the problem is defined as follows:

There is no general way of storing a combination of molecules and their corresponding reactions in a database so that there can be queried on a specific molecule and/or reaction level.

In this thesis, we will try to solve this problem, starting with the research questions in section 3. Note that throughout the entire document, footnotes are added to new chemical terms. These are not there as academic citations, but to give a reader without a chemical background an easier readthrough.

[1] https://pubchem.ncbi.nlm.nih.gov/
[2] https://www.genome.jp/entry/K00615+2.2.1.1+R01830
[3] https://en.wikipedia.org/wiki/Metabolic_pathway

3 Research questions

The research goal is to find a way of storing molecules and their reactions in a database so that they are quick and easy to query. To find an answer, the problem is divided into three main research questions, which each have two sub-questions.

3.1 RQ 1

First, research will be done on the possible representation methods and formats of molecules and/or chemical reactions. With this information, it can be decided what representation method or format would be the most suitable to use in a chemical database. Therefore, the first research question will be:

RQ 1: What are different representation methods and formats to represent molecules and/or chemical reactions and which method/format would be the most suitable to use in a chemical database?

To facilitate answering the main question, it has been split into two subparts. The first sub-question will try to answer what the digital formats of molecules and reactions are (trying to) represent. Chemical theory and its representation methods will be considered in this part. The first sub-question is formulated as follows:

RQ 1.1: *Looking from a chemical theory perspective, what representation do the digital formats (try to) represent?*

The second sub-question follows from the answer to the first. When it will be clear what the formats (try to) represent, it can be easier to compare them with each other. With this information, the most suitable representation method or format for the chemical database can be determined. From this, the second sub-question follows:

RQ 1.2*: What digital chemical representation methods and formats exist, and which is the most suitable to use for a chemical database?*

3.2 RQ 2

When the chemical representation for the database has been determined, this will be the base on which the database is built. Therefore, the second research question is:

RQ 2: How can the chemical database be constructed?

This question can be split into two sub-questions. First, the design concept and the structure of the database must be determined based on the chosen chemical representation. Therefore, the first sub-question will be:

RQ 2.1*: What is the conceptual design for the chemical database?*

After the conceptual design has been made, the input format(s) of the database will be determined. When the input format is known, it must be determined how it can be loaded into the new chemical database. With this, the second sub-question can be formulated:

RQ 2.2*: What are the most suitable input formats and how can the input format be transformed into the chemical database format?*

3.3 RQ 3

After the construction and implementation of the chemical database, the design will have to be validated, which brings us to our final research question:

RQ 3: How can the chemical database design be validated?

The validation of the database design will be done by looking at the querying of the chemical data from the database and transformation of the data to a different format. This brings us to both sub-questions:

RQ 3.1*: Which queries can be run and with what ease?*

RQ 3.2*: With what ease can the database format be transformed in other formats?*

4 Methodology

In this section, the methodology for answering the research questions will be given, together with the sections in which they will be answered.

In section 5.1, **RQ 1.1** will be answered by looking at the (chemical) graph theory and its representations of molecules and reactions. Then, in section 5.2, **RQ 1.2** will be answered by searching the literature for different digital representations of molecules and/or reactions and stating their (dis)advantages. With this information, the most suitable representation method or format can be chosen and more information about this method/format, together with already existing work, will be explained in section 5.3.

With the determined representation format, the conceptual design of the chemical database will be presented in section 6.1. Here, the molecules, reactions, corresponding features, and their connections will be defined. Together, this will answer **RQ 2.1**. In section 6.2, the transformation from the input format into the database format and what is needed to accomplish this is described. Additionally, a description of the code that performs this transformation is given and with this, an answer will be given to **RQ 2.2**.

After the database is constructed, the validation can start in section 6.3, which will answer **RQ 3.1**. This will be done by querying real-life problems on the database and testing e.g., the output, length of the queries, and ease of use. In section **Error! Reference source not found.**, the final question, **RQ 3.2**, will be answered. Here, the database will be validated by testing the ease of transforming to another data format. This is done by showing the transformation to another analysis tool with code examples, and an example analysis is performed to show the transformation worked.

Lastly, the business impact is described in section 7, the conclusion in section 8 and the limitations and further work in section 9.

5 Digital chemical representation

An important question to ask early on when creating a chemical database is in what format the compounds and reactions will be represented. This format will have a significant impact on how the database will be designed, as different data representations can require different database solutions. In this section, we will discuss what information the digital format must represent, what the most widely used digital representations of chemical compounds are, and which format would be the most suitable for our case.

5.1 What to represent

Before diving deeper into the digital representation methods and formats of molecules and chemical reactions, we will first look at what they try to represent. The simplest way of writing a molecule would be the molecular formula, which counts the number of times the atoms occur in a molecule. This amount is then represented as a subscript of the atom letter(s) (Hartshorn et al., 2015). For ethene, which has four hydrogen and two carbon atoms, this would be C_2H_4. A reaction can be depicted by using an arrow which represents the reaction, the plus (+) sign separates the different molecules on both sides of the arrow. The reagents (molecules that react) are typically written to the left of the arrow, and the products typically to the right. For example, the combustion of ethene in oxygen can be depicted as $C_2H_4+3O_2 \rightarrow 2CO_2+2H_2O$. However, this representation leaves out most of each molecule's structural information and can be interpreted in different ways. For example, C_2H_6O can be either dimethyl ether or ethanol, which are different chemical compounds with other properties. Therefore, we will look at the representation method described below.

5.1.1 Graphs

Graphs provide a more suitable representation from the chemical theory to represent molecules and their reactions. To understand why, we will first take a step back from chemistry to look into the graph theory, where a graph is defined as follows (Gross et al., 2014):

D1: A ***graph*** $G = (V, E)$ consists of two sets V and E.

- The elements of V are called ***vertices*** (or ***nodes***).
- The elements of E are called ***edges***.
- Each edge has a set of one or two vertices associated to it, which are called its ***endpoints***. An edge is said to ***join*** its endpoints.

Equation 1: Definition of an undirected graph (Gross et al., 2014)

An example of a simple graph drawn with the definition above is shown in figure 1. Here, V1, V2, V3 and V4 are the vertices/nodes and E1, E2, E3 and E4 the edges/relationships. As can be seen, nodes V1 and V2 are connected to each other by edge E1, whereas node V2 and V3 are connected to each other by Edge E2, et cetera. Furthermore, a *subgraph*[4] is a graph formed of a subset of the nodes and edges of the main graph. In the example in figure 1, nodes V1, V2 and V3, with their corresponding edges E1 and E2, would be a subgraph of the complete graph.

[4] https://en.wikipedia.org/wiki/Glossary_of_graph_theory#subgraph

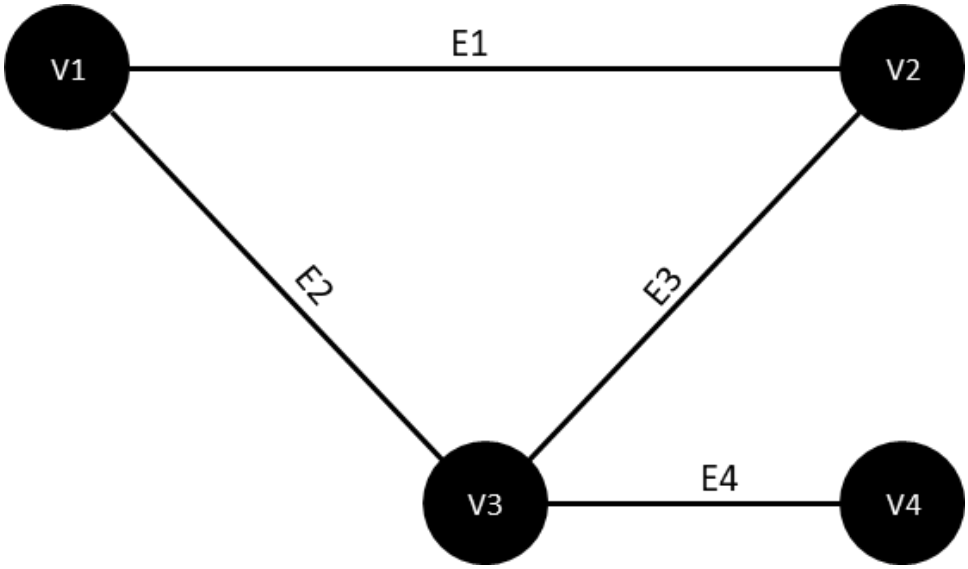


Figure 1: Example of an undirected graph

For every graph, an *adjacency matrix* can be constructed. This is a matrix of *n* by *n*, where *n* is the number of nodes. If there exists an edge between two nodes, the value is 1, otherwise the value is 0 (Gross et al., 2014). An example of the adjacency matrix of the graph in figure 1 can be seen in table 1 below. As this is an *undirected* graph, where the edges work in both directions, the adjacency matrix is mirrored across the diagonal.

	V1	V2	V3	V4
V1	0	1	1	0
V2	1	0	1	0
V3	1	1	0	1
V4	0	0	1	0

Table 1: Adjacency matrix of the graph in figure 1

In addition to the simple graph basics above, another important property of graphs is *isomorphism*, which occurs when two graphs are structurally identical (Gross et al., 2014). There is an isomorphism between graph G1 and G2 if there is a bijection between the nodes and the edges of the two graphs that preserves adjacency, i.e., if ϕ is the bijection between the nodes, then for the adjacency matrices $A^{(1)}$, $A^{(2)}$ of the graphs we have $A^{(1)}_{nm} = A^{(2)}_{\phi(n)\phi(m)}$.

In figure 2, an example of graph isomorphism between the graphs G and H can be seen (Gross et al., 2014). Above the graphs, the correspondence mapping between nodes *u* of graph G and nodes *v* of graph H is shown. Note that this specific example can have two kinds of correspondence mapping as the graphs are symmetrical. In addition to the given correspondence mapping in figure 2, the following correspondence would also be correct: *u1→v3, u1→v3, u1→v3 and u1→v3.*

$u_1 \mapsto v_1 \quad u_2 \mapsto v_2 \quad u_3 \mapsto v_4 \quad u_4 \mapsto v_3$

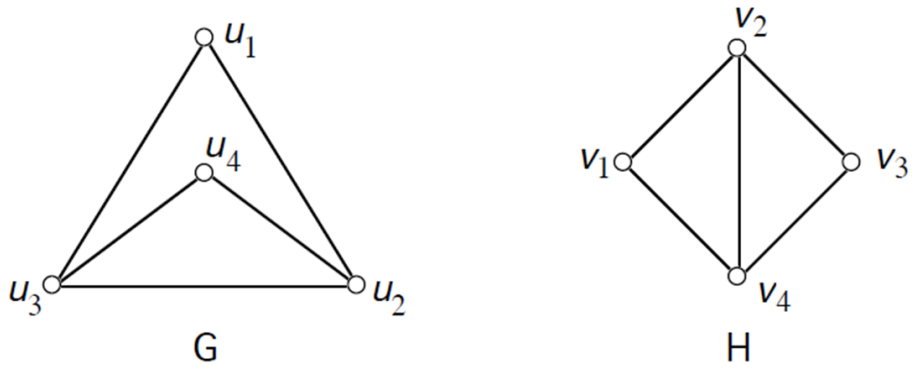


Figure 2: Example of graph isomorphism (Gross et al., 2014)

When creating an adjacency matrix of both graphs G and H of figure 2, we can see that these matrices are identical. To better visualize this identicality, the order of the nodes of one of the graphs has been adapted.

	u_1	u_2	u_3	u_4
u_1	0	1	1	0
u_2	1	0	1	1
u_3	1	1	0	1
u_4	0	1	1	0

G

	v_1	v_2	v_4	v_3
v_1	0	1	1	0
v_2	1	0	1	1
v_4	1	1	0	1
v_3	0	1	1	0

H

Table 2: Adjacency matrices of graph G and H

In addition to the graph isomorphism as described above, there is also *subgraph isomorphism* (Jiang et al., 2018). This occurs when the nodes and edges of graph G1 are a subset of the nodes and edges of graph G2. For example, the graph in figure 3 below has subgraph isomorphism with the graph in figure 1.

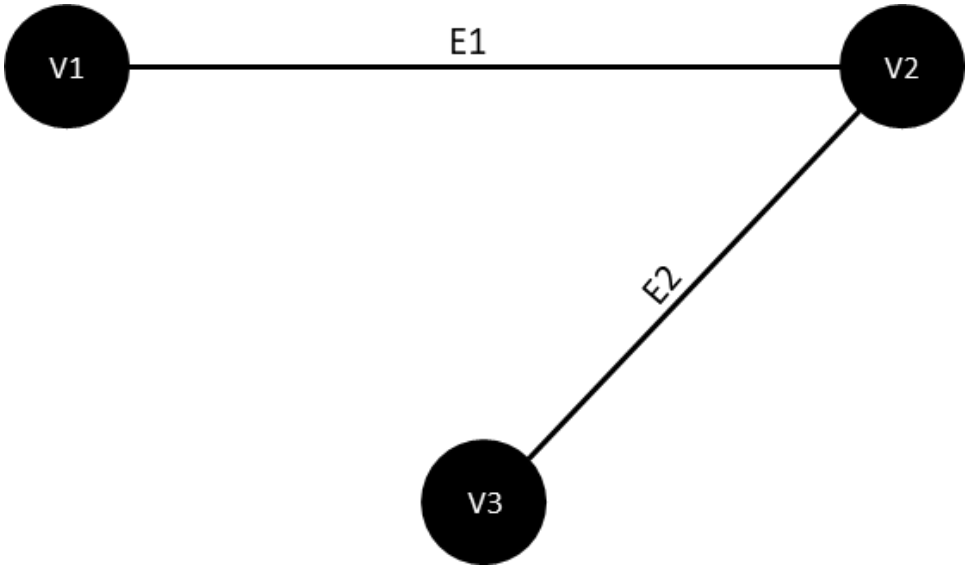


Figure 3: Example of a subgraph isomorphism of the graph in figure 1

Lastly, besides undirected graphs, there are also *directed graphs*. In these graphs, the nodes are connected by an arc/arrow, which consists of a tail and a head (Bang-Jensen & Gutin, 2009). The arc leaves from the tail side and enters at the head side. In figure 4, a simple example of a directed graph is given. Both arcs A1 and A2 have their head in node V2, and their tail in node V1 and V3 respectively, which means that the directions go from node V1 to V2 and from node V3 to V2.

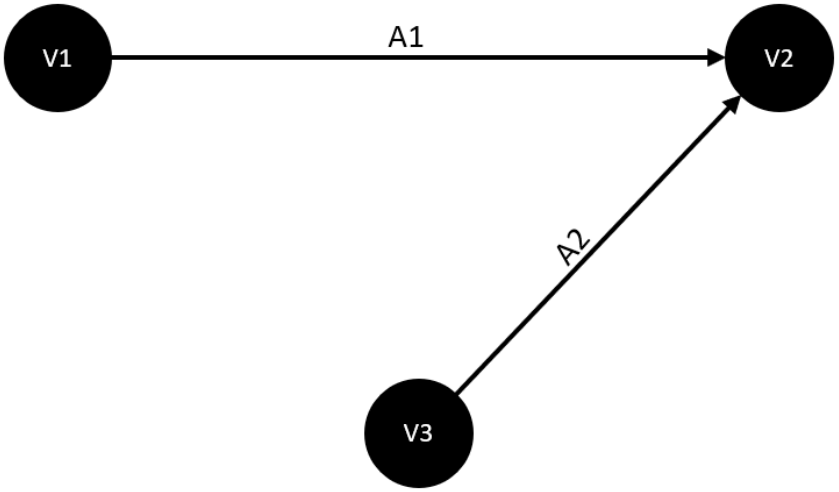


Figure 4: Example of a directed graph

As the example above is relatively abstract, therefore a more practical example is given in figure 5. Here, John and Mary are married to each other: John is married to Mary and, necessarily, Mary is married to John; therefore, the edge 'MARRIED_TO' is undirected. In the same graph, Mary has a father called Peter, but Peter does not simultaneously have Mary as his father. Therefore the 'HAS_FATHER' edge is directed, with the tail at Mary and the head at Peter. Note that the HAS_FATHER edge could equivalently be written inverted as HAS_CHILD.

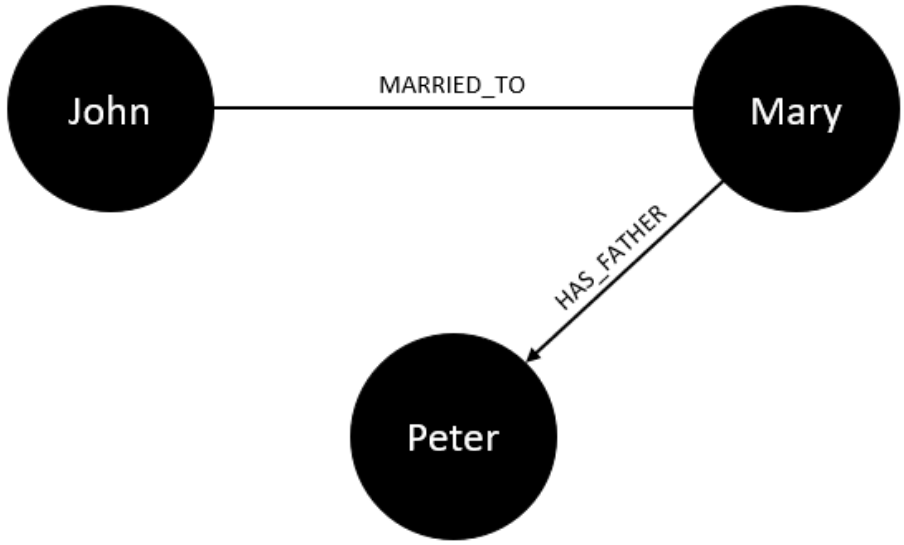


Figure 5: Example of a directed graph

5.1.2 Chemical graphs

Having covered the basics of graph theory, we can connect this to chemical theory. Chemical structures, especially in organic chemistry, are typically represented as graphs by labeling the node with the atom type and the undirected edge with the bond type; these graphs are called chemical graphs (Trinajstic, 1992). In these chemical graphs, the rules of (sub)graph isomorphism also apply and can therefore be used to look for identical (sub)structures. An example of the visualization of a chemical graph is shown in figure 6a below. Note that most chemists would not call this a chemical graph, but a structural formula[5]. The number of lines between the atoms determines the bond type; in this example, this would be a single bond between the carbon and hydrogen atoms and a double bond between the two carbon atoms. Here, the H atoms are shown, but there are also versions where the H atoms are implicit (absent from the depiction) to improve readability. A reaction can be represented with the same method as for the molecular formula, with an arrow for the reaction and a plus (+) to distinguish the different molecules on both sides of the arrow. An example can be found in figure 6b. More information about chemical graphs/structural formulas can be found in a book about the basic principles of organic chemistry (Roberts & Caserio, 1977).

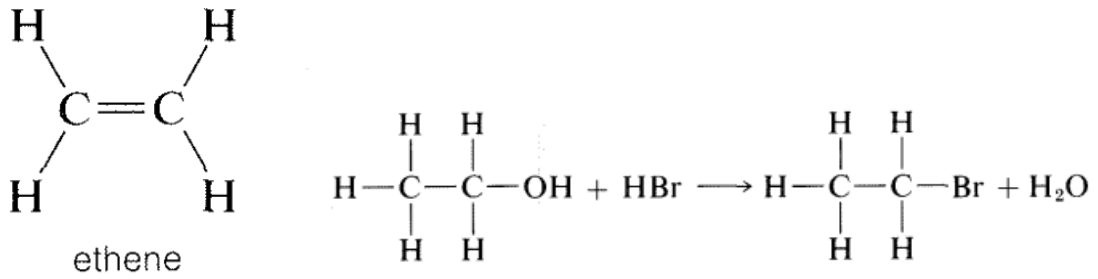


Figure 6: Example of the structural formula of ethene (a) and a reaction example (b) (Roberts & Caserio, 1977)

[5] https://en.wikipedia.org/wiki/Structural_formula

When a chemist looks at the visualization in figure 6, he or she can identify which molecule is being depicted and which changes have occurred when the molecule reacts by just looking at a related structure. For a computerized algorithm, however, this is harder to do, as the format is an image. It is possible to train an algorithm to perform image recognition to interpret these images, but this would require significant computing power and would be prone to error. However, several methods that represent atoms, molecules, and reactions in a machine-readable way can be found in the literature; some of these are described in the next section.

5.2 Different molecular representation methods and formats

In this section, multiple digital chemical representation formats are shortly explained together with their pros and cons. The formats reviewed are SMILES, SELFIES, SMARTS, MOLfiles, molecular fingerprints, and graph representation. All formats give their own representation of the structural graph as described in section 5.1.2.

5.2.1 SMILES

SMILES, an acronym for 'simplified molecular-input line-entry system' is a one-line text format that represents molecules, developed to be used in chemical information processing (Weininger, 1988). Two examples of this representation format are displayed in figure 7 below. Subbranches of the main branch are represented by using brackets and rings are represented by using numbers. The SMILES format does not represent the H atoms, as they are considered trivial and therefore redundant. Besides the H atoms, all the atoms of the molecule are represented in this format.

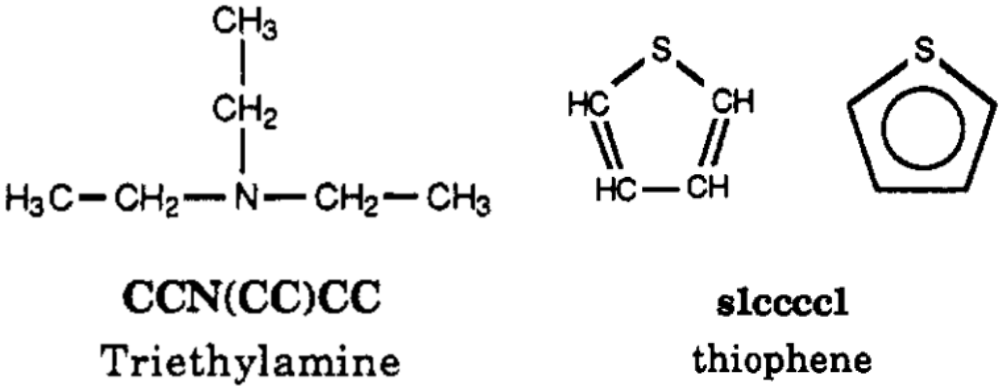


Figure 7: Triethylamine (left) and thiophene (right) as structural formula and SMILES format (Weininger, 1988)

Figure 8 below represents a more extensive, stepwise example of the construction of a SMILES string from a structural formula. In step B, all begin and end atoms of the molecule's rings[6] are chosen to construct a spanning tree. In step C, the main branch of the molecule is chosen, which in this case is the green branch. All other branches will be subbranches of this green branch, here displayed in other colors. There is no general rule for choosing this main branch, it could also have ended with, for example, the orange or dark blue part at the end. In part D, the branch and its subbranches are translated into SMILES.

[6] https://en.wikipedia.org/wiki/Ring_(chemistry)

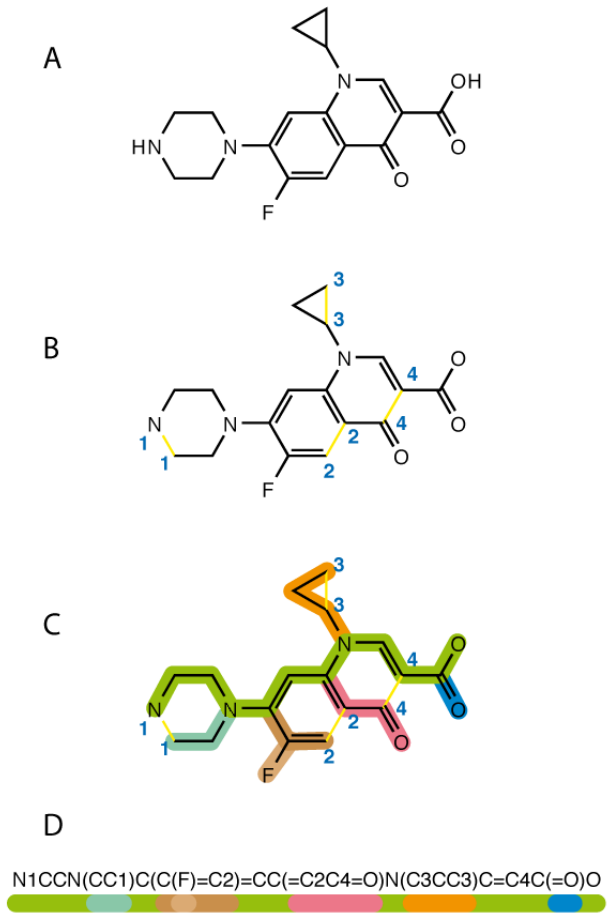


Figure 8: Example of the creation of a SMILES string (source: https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system#/media/File:SMILES.png)

Several advantages and disadvantages of working with SMILES in the machine learning age exist, which are described by Wigh et al. (2022). Advantages are that:

- SMILES are relatively easily readable for humans, which makes them easy to write.
- SMILES consist of only one line and are very compact, therefore they require very little storage space.
- The representation is widely accepted and can be used with most chemical software, like RDKit (Greg Landrum, 2019) and OpenBabel (O'Boyle et al., 2011). The representation is also a standard in several databases, e.g., PubChem upload (Kim et al., 2016), Pathbank (Wishart et al., 2020), and HMDB (Wishart et al., 2007).

Conversely, known disadvantages of SMILES are:

- No 2D or 3D coordinates of the atoms are given, and they cannot be inferred from the format.
- This format is created purely for the representation of molecules, their atoms, and bonds, not the reactions between molecules.
- There can be different correct SMILES strings that represent the same molecule. This problem occurs because there can be different methods of determining the spanning tree and the main branch, as described above. With the graphs of structural formulas, different versions of the same molecule would be recognized as graph isomorphism. However, this is not the case with a SMILES representation and this problem can cause difficulties in database lookups and other

analytical methods. Algorithms have been developed that can solve this by canonicalizing[7] the SMILES, but they are computationally expensive.

- If one wants to get data considering the number of rings, atoms, or bonds of a specific type, this is not possible without using a SMILES interpreter to provide this information. Doing this for all molecules in a database and for every query is very computationally expensive.
- SMILES can also be created for molecules that do not have a valid molecular graph; the smiles of molecules are not surjective on all possible smile strings[8] (Shen et al., 2021).

5.2.2 SELFIES

Self-Referencing Embedded strings (SELFIES) are an improvement of the SMILES format representation which aims to be better machine-readable . Like SMILES, this format also consists of one line with all the atom symbols, excluding the H atoms. Additionally, in contrast to the representation of the SMILES format, SELFIES prevent syntactically invalid strings and enforce the validity of molecules. This is done by storing identifiers for the branches and rings and the use of derivation rules. The best way to explain the difference is to show the examples of Krenn et al. (2019) in figures 9 and 10. In figure 9, an example of the branch representation is given, both the identifier '[branch]' and its size, in this case '[size=1]', are determined. The atom(s) behind the branch and the size identifier are part of this branch. In figure 4, an example of a ring can be seen. Here, the ring and its size are placed after the atoms that are part of the ring. In this case, this is '[ring1]' and '[size=7]', which means that the seven atoms before '[ring1]' belong to this ring. Using these methods, the information in the ring and branch strings is local and can make use of more efficient derivation rules compared to the SMILES representation (Krenn et al., 2020). The SELFIES representation format is relatively new, and several projects have been planned to research in the future (Krenn et al., 2022).

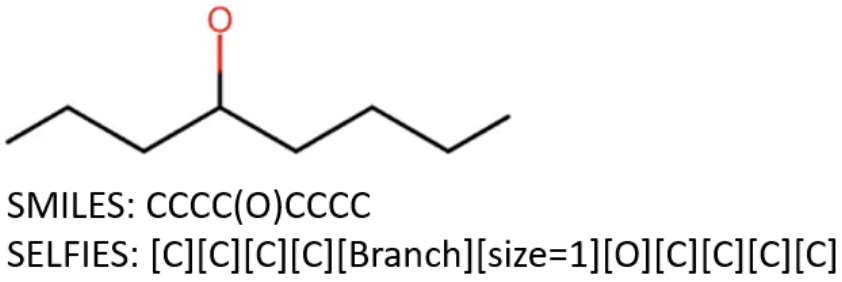


Figure 9: SELFIES branch example compared to SMILES[9]

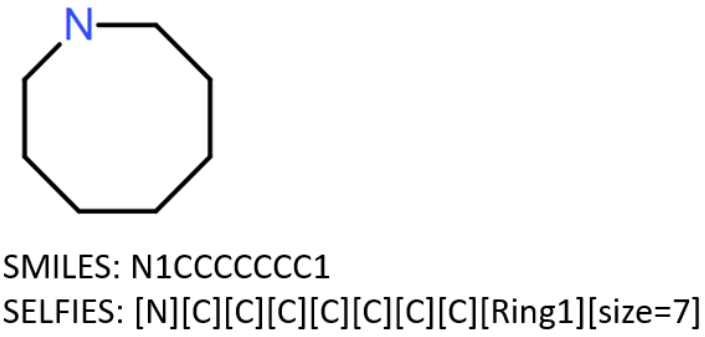


Figure 10: SELFIES ring example compared to SMILES[10]

[7] In this context this is creating one specific SMILES string for each molecule
[8] https://en.wikipedia.org/wiki/Surjective_function
[9] Example from: https://youtu.be/CaIyUmfGXDk?t=1473 (retreived October 19th, 2022)
[10] Example from: https://youtu.be/CaIyUmfGXDk?t=1473 (retreived October 19th, 2022)

The advantages of SELFIES are as follows:

- SELFIES are better readable by computers and therefore more effective to use with ML methods (Wigh et al., 2022).
- SELFIES only produce chemical valid molecules and can represent every molecule (Krenn et al., 2020).
- SELFIES consist of only one line and are very compact, therefore they require very little storage space.

Disadvantages of SELFIES would be that:

- A new familiarity with this representation format would have to be created by humans to properly read it (Krenn et al., 2020).
- No 2D or 3D coordinates of the atoms are given, and they cannot be inferred from this format, the same disadvantage as the SMILES representation has.
- This format is created purely for the representation of molecules, their atoms, and bonds, not the reactions between the molecules, the same disadvantage as the SMILES representation has.
- They are not as well-known and well-used compared to the other representation formats.
- As for SMILES, if one wants to get data considering the number of rings, atoms, or bonds of a specific type, this is not possible without using a SELFIES interpreter to provide this information. Doing this for all molecules in a database for every query is very computationally expensive.
- The SELFIES representation is not unique.

5.2.3 SMARTS

SMILES arbitrary target specification (SMARTS) describes patterns in the molecules using an adapted version of the SMILES format (Daylight Chemical Information, n.d.). Additionally, reaction SMARTS can also be used to represent reactions between molecules. In this representation format, the different molecules are separated by a '.' and the reactants and products by '>>'. Every atom group in the reaction SMARTS is numbered on both sides of the reaction arrow so that they can be traced back to each other. In figure 11, an example of a reaction SMARTS is given together with the structural formula.

Reaction SMARTS

[O$(O(C)([CX4])):8][C:7](=[O:9])[CH:6][C:5][C:4][C:3][C:2]([O$(O(C)([CX4])):10]) =[O:1]>>[O:8][C:7](=[O:9])[C:6]1[C:5][C:4][C:3][C:2]1=[O:1]

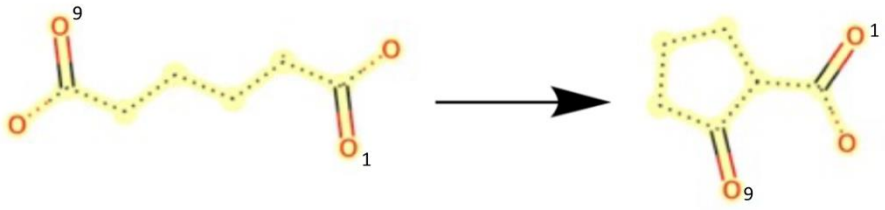


Figure 11: reaction SMARTS example (Gao et al., 2021)

The advantages of SMARTS are that:

- It represents both molecules and the reaction in the same format (Daylight Chemical Information, n.d.).

- The atoms or atom groups on both sides of the reaction can be tracked because they are numbered. However, this also has a downside, which is stated below.
- The representation consists only of one line and therefore they require very little storage space.

The disadvantages of SMARTS are that:

- They focus less on the molecules and more on the reactions. Because of this, valuable information about the molecules and atoms that do not react can be lost. (Coley et al., 2017)
- The atoms of the molecules on both sides of the reaction arrow are numbered to correspond to each other. This causes the structural formula's graph isomorphism not to be maintained by the representation of the SMARTS format. For example, in figure 11, the molecule on the left of the reaction arrow will be the same if you turn it 180 degrees. SMARTS assign the numbers 9 and 1 to the oxygen atoms with a double bond and thereby say that only the oxygen atom with number 9 can be bonded to the ring right of the reaction arrow. In reality, this could have been both number 9 and 1, SMARTS are stating the specific atom that will be bonded to the ring whereas this cannot be specified (Cioslowski & Nanayakkara, 1993).
- Without a visualizer, the format is harder to interpret for humans.
- No 2D or 3D coordinates of the atoms are given, and they cannot be inferred from this format, the same disadvantage the SMILES and SMARTS representations have.
- As for SMILES and SELFIES, retrieving specific data from SMARTS can cause problems as the only data visible in a database is the SMARTS string. With this, it can only be said if two molecules are the same and even this can cause problems, looking at the previous disadvantage. If one wants to get data considering the number of rings, atoms, or bonds of a specific type, this is not possible without using a SMARTS interpreter to provide this information. Doing this for all molecules in a database for every query is very computationally expensive.

5.2.4 MOLfile (Ctab)

The MOLfile is a chemical table format that can represent one molecule and consists of three parts: a counts line, a connection table (Ctab), and the end line (Biovia, 2020; Dalby et al., 1992). In figure 12, an example of a leucine molecule represented by the MOLfile format can be seen. The counts line shows the general facts about the molecule, like the number of atoms, bonds, and if there is chirality. The Ctab consists of two 'blocks': the atom block, where the features of each atom are stated. Of these the x-, y-, and z-coordinates are the most important. These are cartesian coordinates, which are calculated with the information on bond lengths, valance angles, torsional angles, and dihedral angles (Hilderbrandt, 1969). Additionally, twelve other columns refer to different atom features[11]. In the bond block, the two atoms are connected by a bond, and the bond type, stereochemistry, and topology are stated (Dalby et al., 1992). There are two versions of the MOLfile, the V2000 and the V3000 version, where the V3000 version is an improvement of the V2000 version (Wigh et al., 2022). The V3000 version has a bigger counts line (>999 for each input), improved support for new chemical properties, and a better description of stereochemistry.

[11] These are the mass difference, charge, atom stereo parity, hydrogen count, apply stereo configuration, valence, H0 designator, atom-atom mapping number, inversion/retention flag and exact change flag. Only ten features are specified because, counting from the chemical symbol column, the eighth and ninth column are not used.

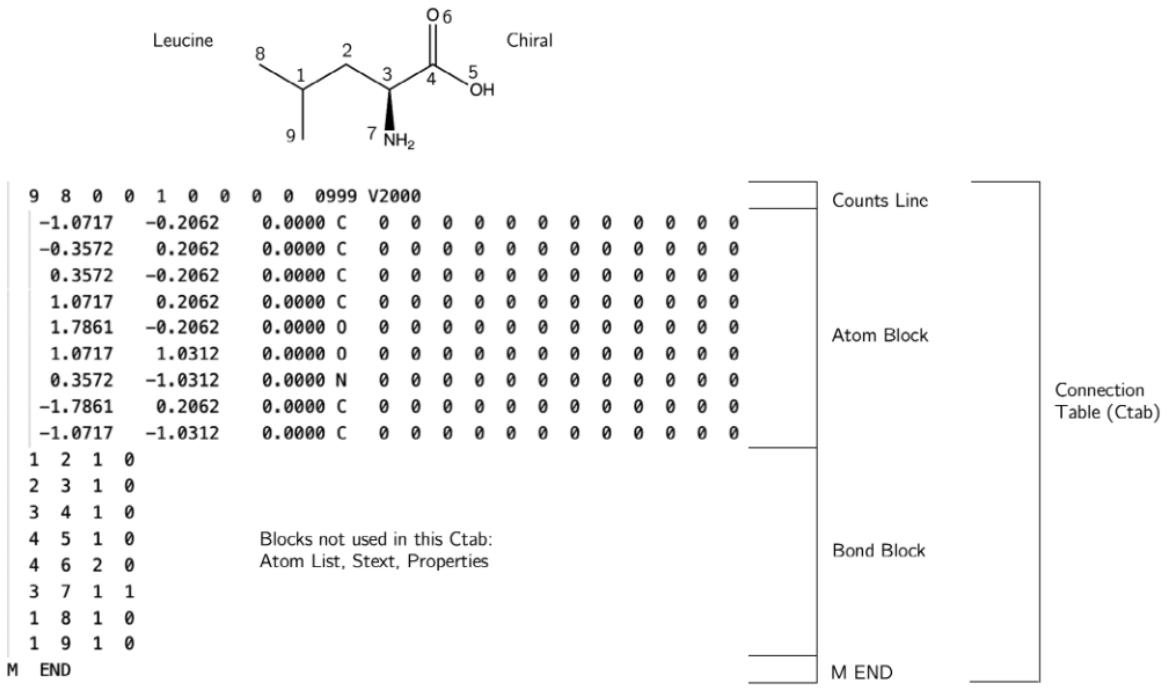


Figure 12: Example of a MDL MOLfile (Dalby et al., 1992)

The advantages of the MOLfile format are that:

- It is used as a standard format to work with in databases like PubChem upload (Kim et al., 2016), Pathbank (Wishart et al., 2020), HMDB (Wishart et al., 2007) and is widely used in general (Wigh et al., 2022). It also works with chemical tools like RDKit (Greg Landrum, 2019).
- The format stores x-, y-, and z-coordinates and other additional atom features.
- Because of the numbered atoms in the atom block and the specified bonds between them in the bond block, the MOLfiles are more consistent in their representation compared to SMARTS and SMILES.

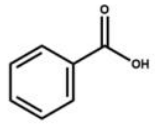
The disadvantages of the MOLfile format are that:

- With the same atom block different bond blocks can be created that represent the same atom. This problem occurs with aromatic[12] rings: in the MOLfile, the bonds in an aromatic ring can be represented as aromatic or with single and double bonds. If the aromatic ring is represented with single and double bonds, there are at least two different ways of doing this, depending on the ring size. An example with benzoic acid can be seen in figure 13 below. Here MOL I, IV and V all have a different bond block, but represent the exact same benzoic acid molecule. MOL I and IV have a different ordering of the single and double bonds, and MOL V has a bond type aromatic. The problem with the single or double bonds in aromatic rings also occurs with the SMILES, SMARTS, and SELFIES formats.

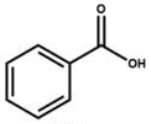
[12] https://goldbook.iupac.org/terms/view/A00441

Atom table for each of MOL I, MOL IV, and MOL V (same atom set!)

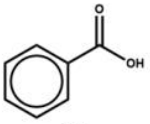
```
9  9  0  0  0  0  0  0  0  0999 V2000
  -1.4289   -0.0000    0.0000 C   0  0  0  0
  -1.4289   -0.8250    0.0000 C   0  0  0  0
  -0.7145   -1.2375    0.0000 C   0  0  0  0  ...
   0.0000   -0.8250    0.0000 C   0  0  0  0
   0.0000   -0.0000    0.0000 C   0  0  0  0  ...
  -0.7145    0.4125    0.0000 C   0  0  0  0
   0.7145    0.4125    0.0000 C   0  0  0  0  ...
   1.4289    0.0000    0.0000 O   0  0  0  0
   0.7145    1.2375    0.0000 O   0  0  0  0
```



I



IV



V

MOL I bond table

1	2	1	0
2	3	2	0
3	4	1	0
4	5	2	0
5	6	1	0
6	1	2	0
5	7	1	0
7	8	1	0
7	9	2	0

MOL IV bond table

1	2	2	0
2	3	1	0
3	4	2	0
4	5	1	0
5	6	2	0
6	1	1	0
5	7	1	0
7	8	1	0
7	9	2	0

MOL V bond table

1	2	4	0
2	3	4	0
3	4	4	0
4	5	4	0
5	6	4	0
6	1	4	0
5	7	1	0
7	8	1	0
7	9	2	0

Figure 13: Example of different bond blocks with the same atom block that all represent the molecule benzoic acid (Robert Belford, 2017)

- This format is created purely for the representation of molecules, their atoms, and bonds, not the reactions between the molecules, the same disadvantage as the SMILES and SELFIES representations have. Additionally, if you would create a MOLfile of the reactant and product of a reaction, the same problem as with the SMARTS will occur as in the MOLfile each atom is numbered to link using the bond block. This causes the structural formula's graph isomorphism not to be maintained by the representation of the MOLfile format.
- Especially for bigger molecules, the format is harder to interpret for humans as the file will consist of a substantial number of lines.
- Both the V2000 and V3000 version are used interchangeably. As the V3000 format cannot be read by V2000 interpreters, the format is as strong as its weakest link, which is version V2000 (Wigh et al., 2022).
- The MOLfiles, are, as the name says, separate files that contain information about a molecule. To run more specific queries, the files have to be read and interpreted in order to output the right information for each molecule for each query, which is computationally expensive.

5.2.5 Molecular fingerprints

The concept of a molecular fingerprint is to describe the molecular structure and substructure of a molecule using bit strings[13], which makes it a useful format to see if there are any (dis)similarities between molecules (Willett et al., 1998). A substantial number of different fingerprints has been developed over the years, each for different purposes (Todeschini & Consonni, 2008). In general, two types of fingerprints can be distinguished: one type uses the substructure as a base for the fingerprint and the other uses the topological path of the fingerprint (Cereto-Massagué et al., 2015).

As there are a lot of different fingerprints, the extended-connectivity fingerprint (ECFP) will be considered for this study (Rogers & Hahn, 2010). This is the most frequently used fingerprint (Wigh et al., 2022) and a variant of the Morgan fingerprint (Morgan, 1964). An ECFP is constructed by using several iterations, in each iteration, the nearest neighbors are found and of this, an identifier is generated. A visualization of this is shown in figure 14, here it can be seen that in every iteration, the method looks for all atoms that are connected to the already discovered area. In every iteration, in the figure called the diameter, the identifiers of each part are displayed on the right side. Note that the algorithm does not store additional identifiers that are duplicates. In the example, there are six atoms but only five are displayed in iteration 1 because there are two carbon atoms with two bonds, which are duplicates.

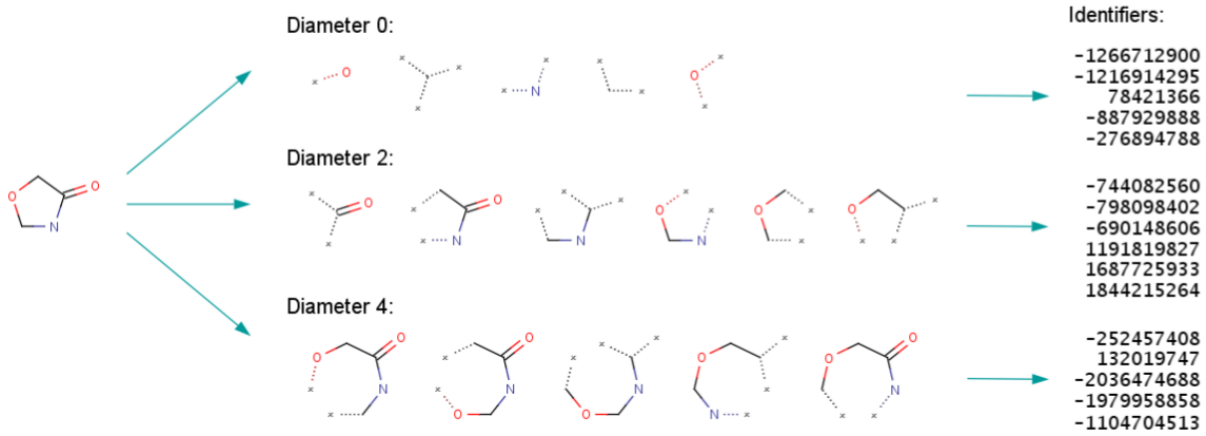


Figure 14: Visualization example of the different iterations (diameters) of the creation an ECFP of leucine (source: https://docs.chemaxon.com/display/docs/extended-connectivity-fingerprint-ecfp.md#src-1806333-extendedconnectivityfingerprintecfp-configuration)

After all the iterations have been completed, the identifiers are derived to a bit string, which can be of any chosen length (the default often is 1024 bits). A smaller string costs less computing power but a smaller string has a higher chance of a bit collision[14]. This occurs when more than one identifier corresponds to a specific bit in the bit string. In figure 15, the identifiers of the example in figure 14 are translated to the bit string and as can be seen, two collisions occur.

[13] An array storing only ones and zeros

[14] Also called hash collision: https://en.wikipedia.org/wiki/Hash_collision

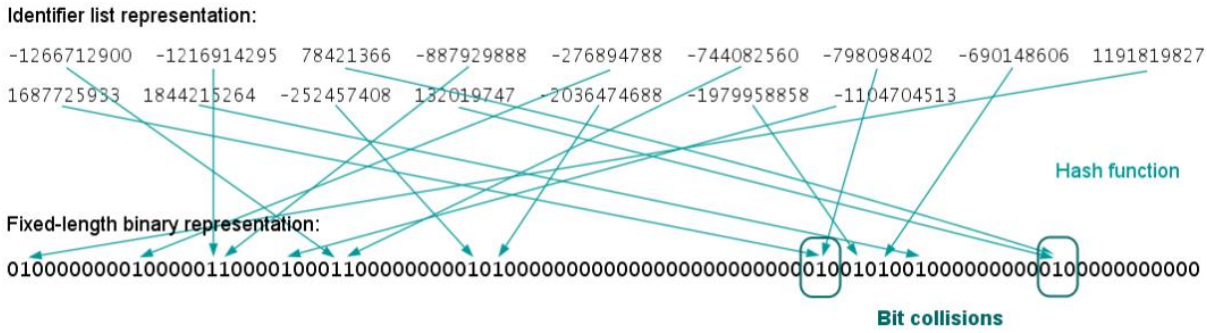


Figure 15: Example of the construction of the fingerprint bit string (source: https://docs.chemaxon.com/display/docs/extended-connectivity-fingerprint-ecfp.md#src-1806333-extendedconnectivityfingerprintecfp-configuration)

The advantages of the ECFP are that:

- The fingerprints are quick to compute and represent some of the additional key features of a molecule (Wigh et al., 2022)
- The format can be used to predict molecule activity, similarity searching, clustering, and virtual screening and is suitable to be used in ML models for chemistry (Rogers & Hahn, 2010)

The disadvantages are that:

- The format is not readable by humans without the use of an interpreter.
- Fingerprints are not suitable for all machine learning methods (Rogers & Hahn, 2010).
- A part of the features of the atoms and molecule is not represented by this format (Wigh et al., 2022).
- Fingerprints can be used as a good way to quickly see similarities between molecules, but to actually see what these similarities are, an interpreter of the fingerprints is needed. Just as for the formats mentioned above, an interpreter is needed for every query and molecule to get more specific information, which is computationally expensive.

5.2.6 Graph representation

In the past sections, we have seen several representations of a structural formula. Each of these formats are interpretations of how a structural formula can be represented but in the end, they are interpretations that each have their pros and cons. In addition to these interpretations of a structural formula, it is also possible to represent the structural formulas as the graphs they are. For the molecules, this has already been done in several studies, of which Ralaivola et al. (2005) was one of the first. In their study, the molecules are undirected graphs where the nodes represent the atoms, and the edges represent the bonds. Here, the nodes have the atom symbol as a feature and the edge the bond type (e.g., single, double, etc.). This is also called an attributed molecular graph (Wigh et al., 2022), which is used for several machine learning and neural network applications (Jin et al., 2017; Shi et al., 2020; Wigh et al., 2022; You et al., 2018).

In addition to representing molecules, it is also possible to represent the reactions between molecules and reaction networks with this representation method. Harada et al. (2020) do this by using a 'graph in graph' method. Here, the molecules are represented as the attributed molecular graph described above. The reactions between the molecules are represented with directed graphs, here the molecule graphs are the nodes, and the reaction arcs. The reactant of the reaction is on the tail side of the arc, and the product is on the head side. A visualization of this can be seen in figure 16 below.

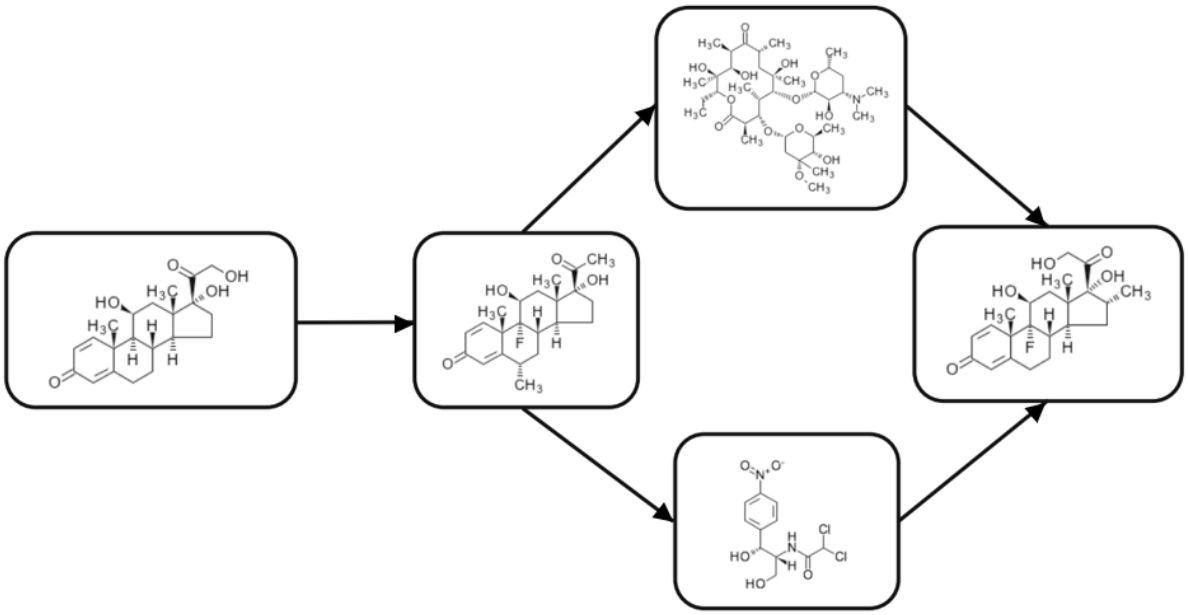


Figure 16: Example of a graph in graph, with the internal graphs being molecules and the external graphs reactions (Harada et al., 2020)

The advantages of this representation are that:

- The graphs are similar to the structural formula representation, as they are both graphs. The difference is that the graphs described in this section are better machine-readable in comparison to the representation as a picture/drawing of the structural formula. Where the other representations try to store as much information as possible to represent the structural formula, the graph representation is the structural formula itself.
- Both molecules and reactions can be represented by this representation method without losing information about the molecules or reactions.

Disadvantages are that:

- Without a suitable visualizer, the graph representation is harder to interpret for humans.
- Most studies do not specify what, or if, additional features are stored besides the atom symbol and the bond type, no general data set of rules of what should be stored exists. Two studies that do describe the additional features that are used are from Kearnes et al. (2016) and Duvenaud et al. (2015).

5.3 Chemical representation for the database

As stated in the previous section, the first step of constructing a database is deciding what the most suitable chemical representation is. In the section above, the different options for molecule and/or reaction representation and their (dis)advantages are explained. Based on this, a graph is choses for both the molecule and reaction representation. This choice has been made because of the following reasons:

- The other formats try to describe the structural formula in the best way possible. The graph representation, however, is not interpreting this structural formula differently, it is, like the structural formula, a graph that represents the molecules. The difference is that this graph will be created using an approach that makes this graph better machine-readable instead of a picture format.
- Additionally, reactions can be created by linking the molecules to each other. This can be done without losing information about the molecules, unlike the other formats that represent reactions and molecules.

5.3.1 Graph database (dis)advantages

Besides the arguments in favor of a graph representation on a molecule and reaction level, arguments from a database perspective can be given as well. Graph databases allow for a more natural way of modeling data, as all the information about a specific entity can be stored in the node, and the additional information can be inferred from the other nodes it is connected to (Angles & Gutierrez, 2008). In addition to this, the queries that can be run on a graph database directly refer to its graph structure. This means that operations, like determining subgraphs[15], can be generated with relatively short queries. Also, it is not necessary to know the whole structure of the graph database to formulate useful queries (Angles & Gutierrez, 2008).

Relational vs graph databases

In comparison to a normal relational database, which consists of linked tables, a graph database has a more flexible structure. The more different tables the relational database will have, the poorer it will perform on analytics tasks that are not based on aggregation because linking these different tables costs a lot of processing power (Angles & Gutierrez, 2008). A relational database with a large number of tables also quickly becomes unwieldy to use for a programmer. Furthermore, when most data are indirectly connected to each other, which is the case with both the atoms in the molecules and the reactions, graph databases are considered the best database to work with this data (Bajer et al., 2021). A graph database excels in being flexible and can deal with different kinds of heterogeneous data in a simple way compared to a relational database. The queries to return data are relatively short and easy to understand for users of the database. Besides these advantages, there are also disadvantages connected to the use of a graph database. Due to this flexibility, there is less control on the data and the chance of wrong data input is higher compared to a relational database. Graph databases also perform relatively poorly for data that can be organized effectively as a small number of fixed format tables with many entries. Additionally, a graph database generates large volumes of data at a fast speed, which requires a lot of working memory (Pokorný, 2015). To overcome this problem, one could try to recreate the functionality of a graph database in a relational database with SQL as query language. However, because of the heterogeneous nature of the data, doing this will create a complicated network of linked tables that will also cause messy and difficult queries to retrieve data. A visualization of a comparison between the two database formats can be seen in figure 17.

[15] https://en.wikipedia.org/wiki/Glossary_of_graph_theory#subgraph

In conclusion, graph databases store and process heterogenous data efficiently and can be very useful for our case.

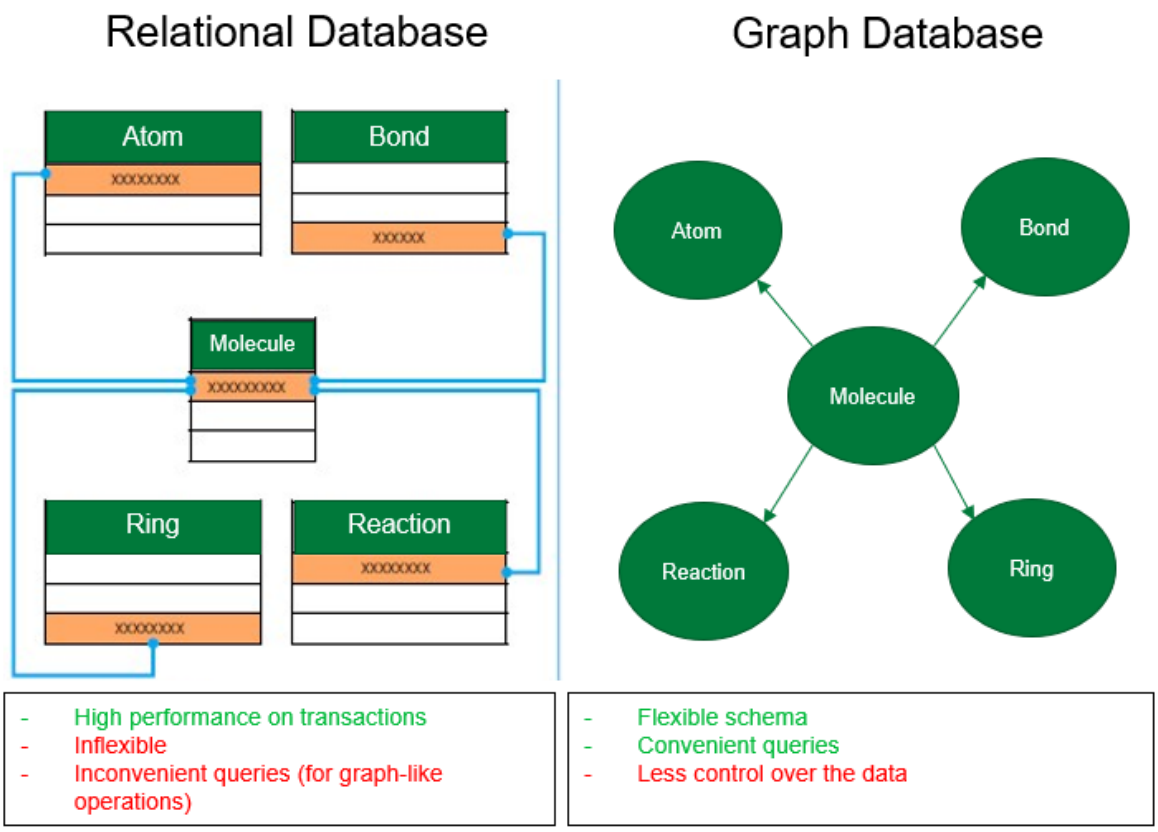


Figure 17: A relational compared to a graph database (base concept from: https://www.nextplatform.com/2018/09/19/the-graph-database-poised-to-pounce-on-the-mainstream/)

5.3.2 Existing chemical graph databases

Graph databases are already heavily used for several applications, like social networks, web graphs, chemical biological networks, master data management, and recommender systems (Tyagi & Singh, 2017). However, considering chemical databases, only two studies could be found that have a connection with both graph databases and chemical molecules and/or reactions, these are explained below.

In the study of Murali et al. (2020), they connect two heterogeneous chemical databases, ChEMBL and DrugBank into a single graph database to discover (dis)similarities between them. Unfortunately, neither molecules nor reactions are represented as graphs in this study.

Sidorov et al. (2021) present a graph database in which the reaction pathway data between different molecules is stored. In this study, the focus is more on the reactions, the molecule(s) are mapped into one node. A simplified example graph of such a pathway can be found in figure 18. Here, the reactants node (C1) and the products node (C2) are connected to the reaction node (R1) with a directed edge.

Figure 18: Simple example of the reaction pathway

5.3.3 Design requirements

As can be seen from the previous sections, there are quite some arguments to create a chemical graph database with both the molecules and reactions in it. Furthermore, this has not been done in any research known to the author, which makes it interesting non-trivial research. To construct the first version of this chemical graph database, the following design requirements are determined:

1. A set of basic features will be created and implemented for each node. For the molecules, this will be based on the features of Duvenaud et al. (2015) and Kearnes et al. (2016). For the reactions general reaction literature and BASF experts will be consulted. Furthermore, it should be possible to add more features over time.
2. Determine the nodes and edges in the graph database so that queries can be run in the most efficient way possible.
3. It should be possible to query data from the database for multiple purposes without having to alter the output data too much.

In the following section, we will work towards a design that complies with these requirements.

6 Chemical graph database

As stated in the literature section, the chemical molecules and reactions will be represented by graphs in the graph database. In this section, we will first look at the different types of nodes and their features and how they are connected to each other. After this, we will explain how the database is constructed in practice and some use cases will be given. The goal of this graph database is to have one database that can store all chemical data available for the molecules and reactions of choice. From this main database, it is possible to obtain all kinds of sub-graphs that represent the needed data for a specific task.

(Sub)part	Feature	Source
Molecule	Hdonors* (Amount of hydrogen donors)	Kearnes et al. (2016)
	SMILES string	
	Chirality*	Kearnes et al. (2016)
--Bond	Type (single, double, triple, aromatic)	Duvenaud et al. (2015); Kearnes et al. (2016)
	Distance*	
	Conjugated*	Kearnes et al. (2016)
--Atom	Symbol	Duvenaud et al. (2015); Kearnes et al. (2016)
	Charge*	Kearnes et al. (2016)
	Hybridization (sp, sp^2, sp^3)	Kearnes et al. (2016)
	Valence	Duvenaud et al. (2015)
	XYZ coordinates	MOLfile (Biovia, 2020)
	Type (C.3, Ar, An)	Duvenaud et al. (2015); Kearnes et al. (2016)
	Degree*	Kearnes et al. (2016)
--Ring	Type (aliphatic, aromatic, aromatic system)	Duvenaud et al. (2015); Kearnes et al. (2016)
	Number of atoms	Kearnes et al. (2016)
Reaction	Enzyme (catalyzed by)	BASF experts
	Organism	BASF experts
	Tissue	BASF experts

Table 3: Key features of the molecules and reactions in the graph database

6.1 Key features and conceptual design

To create the basis of the chemical graph database, the key features of the molecules and reactions have to be defined. For this, the features described by Duvenaud et al. (2015) & Kearnes et al. (2016) together with the MOLfile documentation by Biovia (2020) will be used. Note that these features only form a foundation and that it is possible to add any additional features of choice to the database. In table 3 below, an overview of the different chemical (sub)parts together with their features are stated. The chemical parts are the molecules, reactions, and their features. The chemical subparts are only applicable to the molecule part, which consists of the bonds, atoms, and rings subparts. A further explanation of these (sub)parts will be given in the sections below. Furthermore, the first version of the database will be constructed using the chemical data available, which, in our case, will be MOLfiles or SMILES. Unfortunately, these input formats bring some limitations with them, and they cause some features to be partially altered. If this is the case, the feature is marked with a * and this will be further

clarified in the section of each (sub)part. Additionally, the reaction features are solely added based on the advice of BASF experts, note that these features are highly dependent on the type of reaction.

6.1.1 Nodes & Edges

A fundamental part of the graph, and therefore also the graph database, is what is defined as a node (Gross et al., 2014). Below, each node that can be found in the graph database is described together with its features from table 3. In this graph database representation of the molecules and reactions, it is chosen to represent all chemical (sub)parts that have features (including bonds and reactions) as nodes in this database. It is also possible to assign features to edges but using only nodes for feature storage creates a database that is easier to overview and query (Tyagi & Singh, 2017).

Molecules

The molecules are represented as nodes, which form the basis of the molecule. The molecule node has two features, which are the number of Hdonors and the SMILES string. The SMILES string is added as it is a simple one-line representation of the molecule. Especially with new molecules, that do not have a name yet, this can be an easier way for users of the database to recognize the molecule. The number of Hdonors indicates how many hydrogen bonds can be 'donated' by this molecule. The more hydrogen donors a molecule has, the more likely it is that it will react with another chemical compound[16] (Hasan et al., 2020). Kearnes et al (2016) describe the number of hydrogen donors on an atom level, but in the created graph database this is done on a molecule level as the input formats do not allow an atom-specific version. Lastly, Kearnes et al. (2016) describe chirality[17] as a feature, but this feature could not be calculated with the available tools.

Bonds

In both the digital graphs that represent molecules (Ralaivola et al., 2005) and in chemical graph theory (Trinajstic, 1992), bonds are represented as edges. However, for the creation of our graph database, it is chosen to represent the molecule's bonds as nodes instead of edges. This decision has been made because a bond has its own features, it is not just an edge without features that connects two atom nodes. Representing the bond as a node, the different feature values of the bond can be stored, retrieved, and as stated earlier, it can be better overviewed and queried.

The bonds have two key features. The first feature is the type of bond: this can be single, double, triple, or aromatic[18] (Duvenaud et al., 2015; Kearnes et al., 2016). The bond type is aromatic when the specific bond is part of an aromatic ring. The second feature is the distance, which refers to the distance between the two atoms the bond connects (Kearnes et al., 2016). In the database, the distance feature is represented as a float. Duvenaud et al (2015) also mention conjugation[19] as a key feature of a bond. Unfortunately, this feature cannot be retrieved with the current input format and tools.

Atoms

The study of (Ralaivola et al., 2005) is one of the first studies that represents molecules as machine-readable graphs. In their study, the atoms and their features are represented as nodes, which will also be done for the creation of our graph database. As the atoms carry the most information, the atom nodes will also have the most features relative to the other nodes. The first and most important feature of the atom is the symbol, which determines the chemical element (Duvenaud et al., 2015;

[16] https://www.chem.ucla.edu/~harding/IGOC/H/hydrogen_bond_donor.html
[17] https://goldbook.iupac.org/terms/view/C01058
[18] https://goldbook.iupac.org/terms/view/A00441
[19] https://goldbook.iupac.org/terms/view/C01267

Kearnes et al., 2016). Other features are the charge of the atom, the hybridization[20], valence[21], x-, y-, and z-coordinates, atom type, and degree. Kearnes et al (2016) mention both formal[22] and partial[23] charge, however, due to input limitations, only the partial charge can be given as a feature, which is a positive or negative integer. The hybridization as mentioned by Kearnes et al (2016) is implemented and can be sp, sp^2, or sp^3. The valence stated by Duvenaud et al (2015) is taken into the database, which is a positive integer. The x-, y-, and z-coordinates of each atom are added to the database, where the z coordinate is available depending on the type of MOLfile (MOL or 3DMOL (Biovia, 2020)). Lastly, an extra type is added as an atom feature, which can show if the atom is aromatic, as stated as an important feature by both Duvenaud et al. (2015) and Kearnes et al. (2016). In addition to this, the feature can contain a lot of other atom types, which can be found in the documentation of (TRIPOS, 2005) on respectively pages 53 and 54. Lastly, it has not been clearly specified what is meant with the degree by Duvenaud et al. (2015). Therefore, the degree calculation of RDKit is used, which defines the degree as the number of directly bonded neighbors of an atom.

Rings

Both Duvenaud et al (2015) and Kearnes et al (2016) represent a chemical ring of a molecule by storing it as features for both the atoms and bonds. This representation stores the aromatic ring information for all the individual bonds and atoms. However, when, for example, you want to query every atom and bond in the dataset that has a certain ring type, this method would be highly inefficient. This inefficiency is caused by the fact that all atoms and bonds must be checked individually to find out if they are a part of a ring or not. To overcome this inefficiency, it is decided to represent a ring as a node in the graph database. By doing this, the bond and atom nodes that are in a ring can be linked directly to the specific ring node they belong to. Additionally, if there are more rings of the same type in a molecule, it is directly clear to which ring the atoms and bonds belong with this representation.

A ring node has two features: the ring type and the number of atoms. The ring type feature has three options: aromatic, aromatic system[24], and aliphatic[25]. The ring has an aromatic type if it concerns an aromatic ring. If there are aromatic rings with overlapping atoms, the ring type would be an aromatic system. If the ring is not aromatic, it will have the type aliphatic. In the example in figure 19 below, the ring in the red area is aromatic, the rings in the green area are aliphatic, and the rings in the blue area form one aromatic system. The other feature of the ring is the number of atoms that are in the specific ring or system.

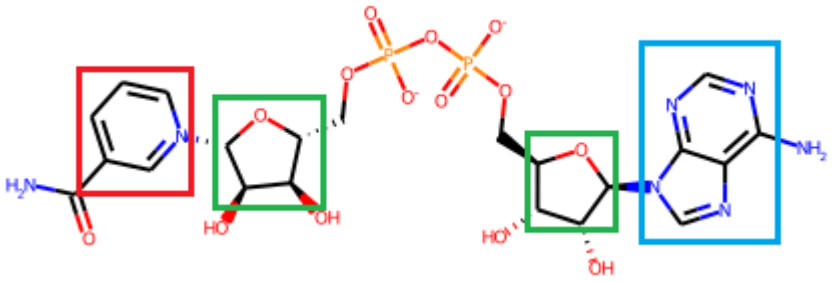


Figure 19: Example of the different ring types

[20] https://goldbook.iupac.org/terms/view/H02874
[21] https://goldbook.iupac.org/terms/view/V06588
[22] https://en.wikipedia.org/wiki/Formal_charge
[23] https://en.wikipedia.org/wiki/Partial_charge
[24] Two or more aromatic rings that are connected to each other by sharing at least one atom
[25] https://goldbook.iupac.org/terms/view/A00217

Reactions

In literature, reactions are represented as an edge between the reactant and the product (Harada et al., 2020; Schilling et al., 1999). However, for the same reasons as for the bond nodes, the reactions will be also represented as nodes in our graph database. The features of the reaction nodes are highly dependent on the reactions that one wants to store in the graph database. As the data of the use case consists of metabolic pathway data, the type of the enzyme, organism, and the kind of tissue are added as reaction features, based on the advice of BASF experts.

6.1.2 Edges

The edges between the nodes are essential for the graph database, as they show what information is available about the nodes including how the nodes are connected to each other to represent chemical structure. We will describe for every node mentioned in the previous section how they are connected with edges. Each edge type has been given a name, which will be specified below.

We will start with the molecule node, which has directed edges to the bond, ring, atom, and reaction nodes. A molecule node is connected to:

1. An atom node with the 'HAS_ATOM' edge, where the tail is at the molecule node and the head at the atom node.
2. A bond node with the 'HAS_BOND' edge, where the tail is at the molecule node and the head at the bond node.
3. A ring node with the 'HAS_RING' edge, where the tail is at the molecule node and the head at the ring node.
4. A reaction node with the 'REACTS_IN' or 'PRODUCES' edge. For the 'REACTS_IN' edge, a molecule node is at the tail and a reaction node at the head. This molecule node will be a reactant[26] of the specific reaction. For the 'PRODUCES' edge, a reaction node is at the tail and a molecule node at the head. This molecule node will be a product[27] of the reaction.

In the case of edges 1, 2, and 3, we choose for a directed edge because atoms, bonds, and rings together form a molecule, whereas the molecule node facilitates a combination of atom, bond, and ring nodes to be found more effectively in the database. The molecule nodes are, in other words, on another 'level' than the atom, bond, and ring nodes.

Atom and bond nodes are connected to each other with a directed 'BONDED_WITH' edge, where the tail is at the atom node and the head is at the bond node. We choose for a directed edge because an atom is bonded in the bond and a bond is not bonded in the atom. Exactly two atoms are connected to one bond node.

The ring nodes are connected with directed 'HAS_BOND' edges to the bond and atom nodes that are part of the ring. Here, the edge tail will be at the ring node and the head at the other nodes. The explanation for the choice of directed edges is similar to the explanation of the molecule node: the bond and atom nodes make the ring itself, whereas the ring node facilitates the ring to be found more effectively in the database.

A benefit of implementing all edges as directed is that the program that is used to implement the database, Neo4j[28], assumes that all edges are directed. Therefore, it could be less computationally demanding to work with directed instead of undirected edges in this database.

[26] https://goldbook.iupac.org/terms/view/P04861
[27] https://goldbook.iupac.org/terms/view/R05163
[28] https://neo4j.com/

A visualization of the combined node and edges of the chemical graph database can be seen in figure 20.

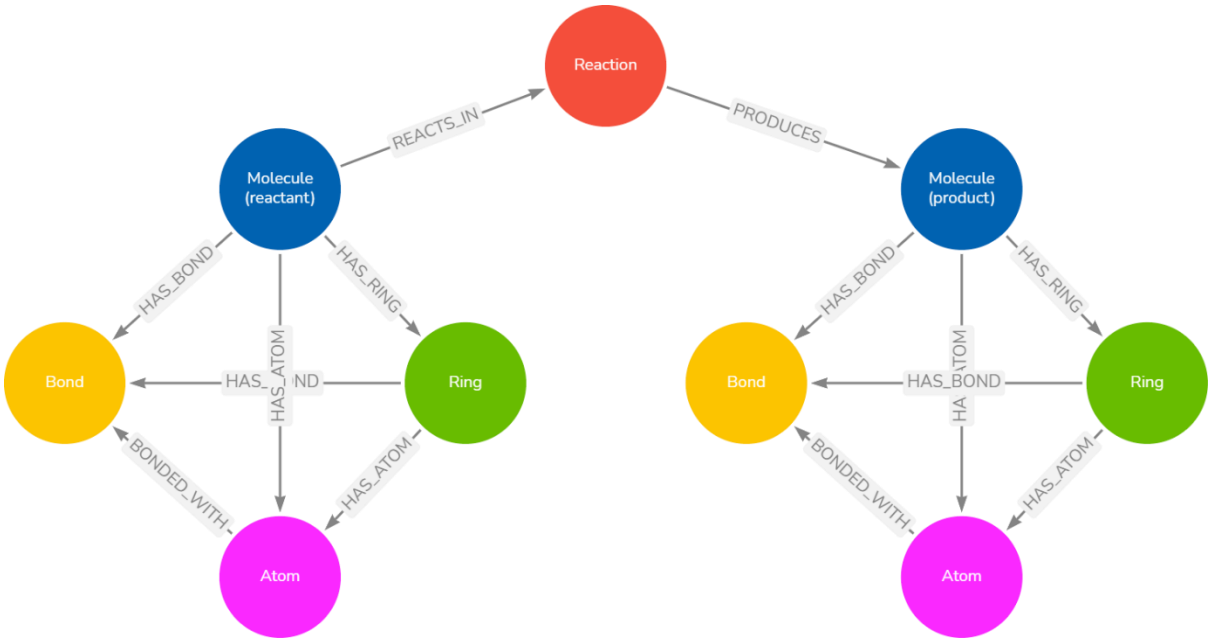


Figure 20: Example of the graph database's nodes and edges

The figure above can also be written down as a web ontology language (OWL), in this case, Terse RDF Triple Language (Turtle). This language is used as there is no convenient way to do this in Neo4j. The Turtle representation can be seen below:

```
#node types
chem:Reaction a owl:Class.

chem:Bond a owl:Class.

chem:MolecularSubGroup a owl:Class
chem:Molecule a owl:Class;
      rdfs:subClassOf chem:MolecularGroups.

chem:Ring a owl:Class;
      rdfs:subClass chem:MolecularSubGroup.
```

```
chem:Atom a owl:Class;
      rdfs:subClassOf chem:MolecularSubGroup.

chem:REACTS_IN a owl:ObjectProperty;
      rdfs:domain chem:Molecule;
      rdfs:range chem:Reaction.

chem:PRODUCES a owl:ObjectProperty;
      rdfs:domain chem:Reaction;
      rdfs;range chem:Molecule.

chem:BONDED_WITH a owl:ObjectProperty;
      rdfs:domain chem:Molecule;
      rdfs:range chem:Bond.

chem:HAS_MOLECULAR_SUBGROUP a owl:ObjectProperty;
      owl:transitiveProperty.

chem:HAS_ATOM a owl:ObjectProperty;
      rdfs:subProperty owl:HAS_MOLECULAR_SUBGROUP;
      rdfs:domain chem:MolecularSubGroup;
      rdfs:range chem:Atom.


chem:HAS_RING a owl:ObjectProperty;
      rdfs:subProperty chem:HAS_MOLECULAR_SUBPGROUP;
      rdfs:domain chem:Molecule;
      rdfs:range chem:Ring.


chem:BONDED_WITH a owl:ObjectProperty;
      rdfs:domain chem:Atom;
      rdfs:range chem:Bond.
```

Furthermore, an example of test data is displayed in figure 21. Here, the orange nodes are the molecules, the beige nodes are the reactions and the red and blue are the atoms and bonds, respectively. In this example, the metabolic pathway can clearly be noticed: the molecule on the left is the first reactant of the pathway and the two molecules on the right are the last products.

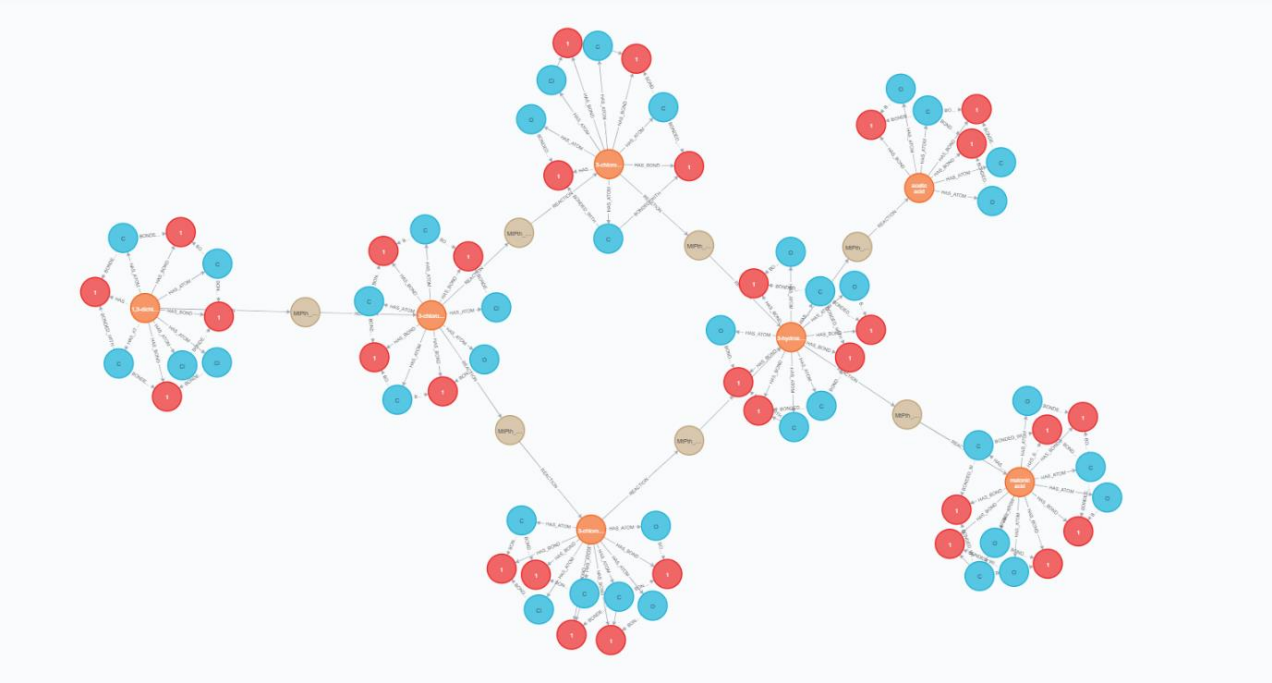


Figure 21: Example of a metabolic pathway in the graph database

6.2 Graph database in practice (implementation)

In the previous section, the conceptual idea of our graph database has been explained. In this section, we will explain how this database is constructed using this concept and which tools were used. The written code and accompanying files can be found on the GitHub page[29].

6.2.1 Used tools

The main tool used to convert the chemical formats to the graph database is python in combination with several packages. Python was used because this is one of the most common programming languages for data-related projects and the author was the most familiar with it. The python packages used to interpret the different chemical formats are RDKit (Greg Landrum, 2019) and OpenBabel (O'Boyle et al., 2011).

For the graph database type, there has been chosen for Neo4j[30] which has its own query language, called cypher. Neo4j was chosen because it's a popular and leading graph database (Tyagi & Singh, 2017) with clear visualizations of the graphs that have been created. Additionally, a connection with python can be made quite easy to automatically store or query data in or from the graph database.

6.2.2 Testing data

For developing and testing the database design, two datasets are used. The first dataset consists of basic oxidations of alcohol, with all molecules stored in MOLfile format. There are 13 reactants with corresponding products and five molecules that do not react. The second dataset consists of data from metabolic pathway studies, pro-processed by BASF. Metabolic pathways are sequences of reactions that occur inside an organism (Nelson et al., 2008). In this dataset, there are 89 different metabolic pathway studies, which have, combined, 671 molecules. The molecule format is SMILES and each molecule is assigned a unique ID, which is used to link the reactants and products of one study to each other. Furthermore, a study type (plant or animal) and the molecule name can be found in the dataset. Note that in the GitHub folder, another, smaller SMILES dataset is given to run the code due to confidentiality issues with the original files. The original SMILES dataset has been seen and assessed by both supervisors, as stated in Appendix A (Section 14.1).

The input format of the chemical data is important, as it influences to what level certain features of molecules and reactions can be added to the graph database. This has already been briefly discussed when explaining the key features of the nodes in the previous section. Between the MOLfiles and SMILES there also exists a difference, which is the lack of coordinates for the SMILES format. Therefore, this feature will be unavailable for the SMILES data in the graph database. Additionally, there is a difference between the processing of MOLfile and SMILES formats, which will be further elaborated in sections 6.2.3 and 6.2.4 below.

Both datasets are transformed into graphs: molecule, atom, and bond nodes and, if applicable ring and reaction nodes, all with their corresponding features and connecting edges. This graph data will be used to test the database output queries, described in the validation of section 6.3.

6.2.3 From MOLfile to graph database

In this section, the process of transforming the MOLfiles dataset into a graph database will be explained. We will give an overview of the design choices and implementation methods to create a better understanding of what has been done. For additional, more in-depth information, there can be

[29] https://github.com/StevenDrenth/ChemicalGraphDatabase
[30] https://neo4j.com/

looked at the corresponding Jupyter Notebook[31], where the code along with comments and explanation can be found. This code consists of a total of 350 lines, including the defined functions that belong to it.

The code can be summarized in the following 11 steps, which are run for every molecule in the dataset:

1. Sort the MOLfiles in the right order to be processed.
2. Load the MOLfiles with RDKit, this is used to get most of the features for the different nodes.
3. Create the molecule node and its features.
4. Convert to a MOL2 file with OpenBabel for additional atom features.
5. Use the ringinfo[32] function from the functions.py file to get ring information.
6. If applicable, create the ring node(s) and its 'HAS_RING' edge with the molecule.
7. Create the atom nodes, the 'HAS_ATOM' edges with the molecule node, and the 'HAS_ATOM' edges with the ring node(s) if this is applicable.
8. Create the bond nodes, the 'HAS_BOND' edge with the molecule node, and the 'BONDED_WITH' edges with the atom nodes.
9. The queries created in the eight steps above are stored in a cypher file, run this file in the Neo4j database.
10. Create all 'HAS_BOND' edges between the rings and their corresponding bonds, if applicable, and run it in Neo4j.
11. If the molecule node is a product, search for the corresponding reactant molecule node and create a corresponding reaction node. Afterward, the molecule and reaction nodes are connected with the 'CONNECTED_WITH' and 'PRODUCES' edges and the created query is run in Neo4j.

Step 1

In the dataset, the name of the MOLfile determines if the molecule is a reactant, product, or non-reacting molecule. A reactant starts with the letter R, for example, R07, and the corresponding product with the letter P, P07, respectively. If the file name starts with the letter R, but there is no corresponding file starting with the letter P, the molecule is non-reacting. For certain output formats of the graph database, it is important to input the reactant and after this the corresponding product. By doing this, they will have successive internal ID numbers in Neo4j, which cannot be changed after the data has been stored in the graph database. To accomplish this, the MOLfiles are sorted so that the reactant-product pairs are processed successively.

Step 2

After the sorting, the processing of the MOLfiles can start. To read the input format, two interpreters are used: RDKit and OpenBabel. Both interpreters facilitate the extraction of the molecule features and can calculate additional features given the information from the input files.

Step 3

The molecule node is the first node to be created. The features described in section 6.1 are added and in addition to this the file name and unique ID. This is all added in a cypher code file to be queried in Neo4j later.

Step 5 & 6

Next, the ring bond is created, which requires an adequate way of dealing with the ring types. As stated, these types can be an aliphatic, aromatic, or aromatic system. RDKit is used to determine which

[31] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/1.1-MOLfiles%20to%20Neo4j.ipynb
[32] Self-written function, which can be found in the functions.py file

atoms of the molecule are in a ring and it can determine if this ring is aromatic or not. With this, the distinction between an aliphatic and aromatic ring is easily made. However, we also want the algorithm to recognize aromatic systems, which consist of two or more aromatic rings. In these aromatic systems, some atoms are part of several, or all, rings of the system. It is important that the algorithm sees all the atoms that belong to the rings in this system as a part of one aromatic system, in graph terms one ring node for all atoms, and not (also) as individual rings. For a system consisting of two rings, this was relatively easy to do, but we wanted the algorithm to be able to deal with any kind of aromatic system. Therefore, the code was tested with polycyclic aromatic hydrocarbon (PAH), a molecule that is one large aromatic system consisting of 13 rings, as can be seen in figure 22. The algorithm also classifies this as one system, with one ring node and therefore we can say that the ring type recognition is performing satisfactorily. The creation of the ring, together with the features in section 6.1 and a unique ring ID are written to the cypher query file.

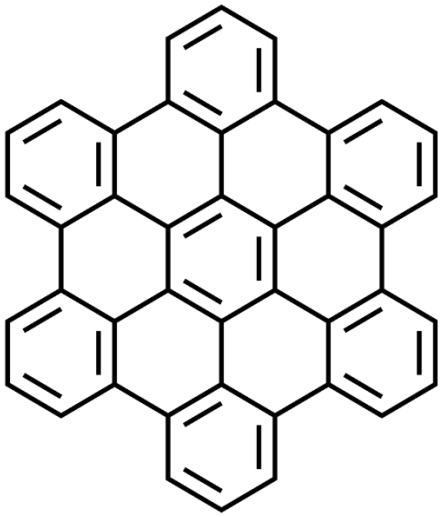


Figure 22: polycyclic aromatic hydrocarbon (PAH) [33]

Step 8 & 9

After the rings, the atom nodes are created. Here, the features stated in section 6.1 are added together with a unique ID. After this, the created atom node is connected to the corresponding molecule with a directed edge. Then, it is checked if this specific atom is part of a ring or aromatic system, if so, the atom is connected to this specific ring node with a directed edge. The cypher commands for this are added to the main cypher file.

Step 10

The last node to add is the bond node, to which all features mentioned in section 6.1 are added. The bond is connected to the corresponding molecule with a directed edge and to the two atom nodes that it connects, also by a directed edge. These cypher commands are written to the main cypher file, which is run in Neo4j and deleted after this. Then an additional cypher command is run in the Neo4j database that connects all the bonds that are part of a ring with the corresponding ring node, using a directed edge.

Step 11

Lastly, we want the reactant and product molecule nodes to be connected with the reaction node. This connection is only made if both the reaction and product molecule have been stored in the

[33] https://en.wikipedia.org/wiki/Polycyclic_aromatic_hydrocarbon

database, as both have to exist in the database in order to make the connection. First, the existing cypher file is deleted, then a reaction node with a unique ID is created, and no additional features are stored as this isn't applicable to the alcohol dataset. After this, the corresponding reactant and product molecule nodes are connected with the reaction node using directed edges. The commands are added to an empty cypher file, and this is run in Neo4j. For the molecules that are not reacting, this process is not performed, as they do not have a corresponding product molecule.

The above steps are repeated until all molecules are stored in the Neo4j database.

6.2.4 From SMILES to graph database

The processing from the SMILES dataset to the graph database is similar for the most part of the process. The differences with the MOLfile algorithm are stated in this section, and the code can be found in the accompanying Jupyter notebook[34].

The first difference is the fact that the SMILES dataset does not have to be sorted for the reactants and products to be successive, as they correspond to each other by using their unique molecule IDs in the database. This brings us to another difference, which is the fact that this database consists of metabolic pathways. There will be several molecules linked to each other by reaction nodes; a product of reaction A can be the reactant of reaction B, etcetera. It is also possible that a reaction, and therefore a reaction node, has several reactants and/or product molecule nodes. To cope with this, an adaptation to the algorithm had to be made. For this dataset, the algorithm looks if the specific molecule has any reactants in the database, therefore checking if it is a product molecule node. If this is the case, it connects the reactant node(s) and the product node to the reaction node using directed edges. Another difference is the lack of coordinates in the SMILES format. With the available computation methods, it was not possible to add these to the database and therefore they are not included as node features.

Additionally, it is important to state that the disadvantage of SMILES strings, where different SMILES can represent the same molecule does not occur in this database. This is because of the following two reasons:

- All SMILES strings are loaded into python with RDKit and the canonical SMILES string that RDKit gives as output is used in further processing.
- The SMILES format can present different spanning trees of the same structural formula, which is a graph. As the representation in the database is a graph and not a spanning tree, this will not cause problems in the database.

[34] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/1.2-SMILES%20to%20Neo4j.ipynb

6.3 Design validation

In this section, we will validate our graph database by running queries that will provide output for different kinds of uses. For this validation, we will mainly show the output of the database created from the SMILES dataset. This is used because this dataset is more complex and, therefore, more complex queries can be shown with this database. If, for a specific reason, the database based on the MOLfile dataset is used, this will be stated explicitly.

6.3.1 Cypher validation examples

Below, different examples of cypher queries are stated together with an explanation of the output will be given.

Example 1

```
MATCH (m:Molecule {Hdonors:4})
RETURN m.name, m.smiles
```

The cypher query above returns a list of the names and SMILES of all molecules in the database that have exactly 4 Hdonors. The output can be seen in table 4:

m.name	m.smiles
"Thiamine pyrophosphate"	"CC1=C(CCOP(O)(=O)OP(O)(O)=O)SC=[N+]1CC1=CN=C(C)N=C1N"
"3-Deoxy-D-manno-octulosonate 8-phosphate"	"C1C(C(C(OC1(C(=O)[O-])O)C(COP(=O)([O-])[O-])O)O)O"
"Pyrophosphate "	"OP(O)(=O)OP(O)(O)=O"

Table 4: Output of example 1

Example 2

```
MATCH (m:Molecule)-[:HAS_RING]->(r:Ring {ring_type:'aromaticsystem'})
RETURN m.name, m.smiles
```

In this query outputs a list of the molecule names and SMILES of every molecule node that is connected to a ring node with ring type ‘aromatic system’. A part of the output is displayed in table 5.

m.name	m.smiles
"Coenzyme A"	"CC(C)(COP(O)(=O)OP(O)(=O)OC[C@H]1O[C@H]([C@H](O)[C@@H]1OP(O)(O)=O)N1C=NC2=C1N=CN=C2N)[C@@H](O)C(=O)NCCC(=O)NCCS"
"Adenosine triphosphate"	"NC1=NC=NC2=C1N=CN2[C@@H]1O[C@H](COP(O)(=O)OP(O)(=O)OP(O)(O)=O)[C@@H](O)[C@H]1O"
"Adenosine monophosphate"	"NC1=C2N=CN([C@@H]3O[C@H](COP(O)(O)=O)[C@@H](O)[C@H]3O)C2=NC=N1"
…	…

Table 5: Part of the output of example 2

Example 3

```
MATCH (m:Molecule {studyid:'MtPth_Map_031'})
MATCH (rxn:Reaction {studyid:'MtPth_Map_031'})
RETURN m, rxn
```

The query above matches all the molecule and reaction nodes with the study ID 'MtPth_Map_031'and returns the connected molecule and reaction nodes and the corresponding edges in between[35]. As this is the first query that returns nodes and edges, the output visualization is shown in figure 23. Here the red molecule nodes and the beige reaction nodes and the directed edges can be seen. It is also possible to 'expand' a molecule so that it shows all its connections, in this case the atom, bond, and ring nodes. The expansion of one molecule of this example is displayed in figure 24, with blue for the atom, yellow for the bond and green for the ring node(s).

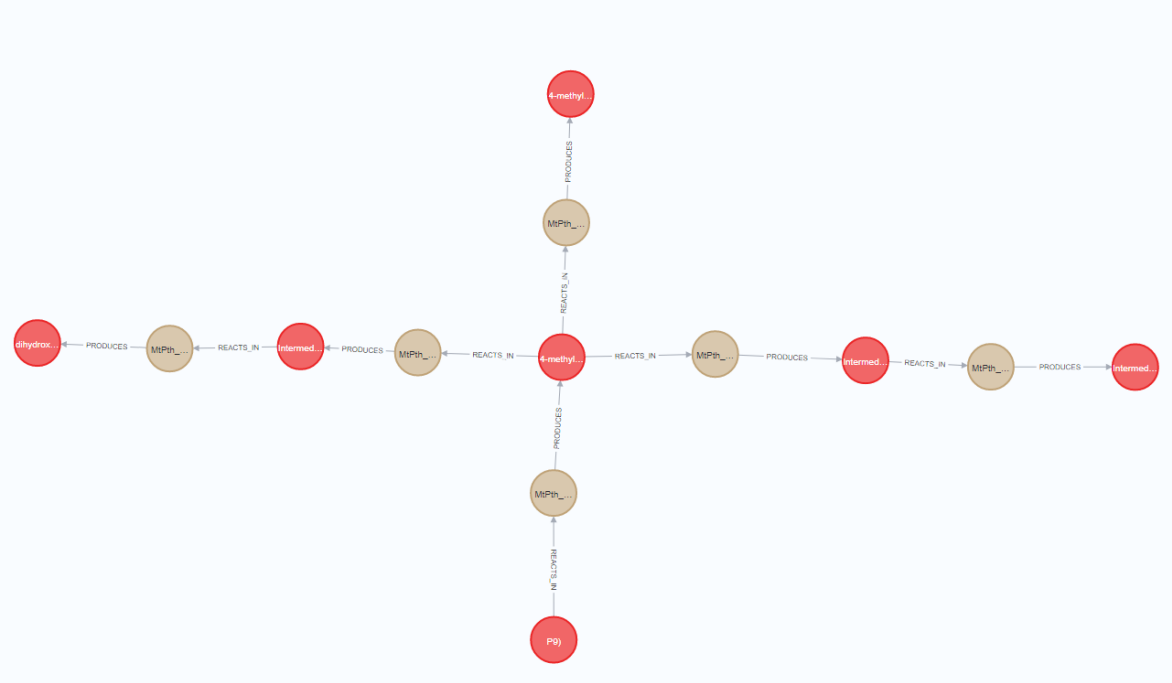


Figure 23: Output of example 3

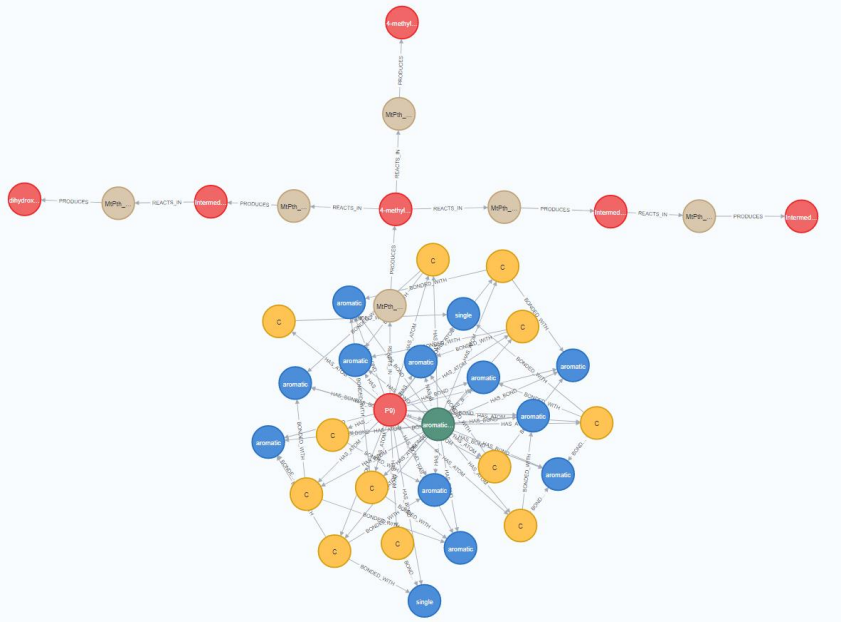


Figure 24: Output of example 3 with an 'expanded' molecule

[35] With the given dataset in the GitHub file, this example cannot be reproduced as it is part of another dataset. One can create a similar output by changing 'MtPth_Map_031' to 'MtPth_Map_001', for example.

The figures shown above are the visualization of the output, but the underlying part of the output from Neo4j is in JSON. In table 6, a part of the JSON output from example 3 is stated, which consists of a molecule and reaction node column. The visualization in figure 23 has 6 reaction and 7 molecule nodes. Therefore, the JSON output table of this visualization consists of 6*7=42 rows in total, of which the first two are displayed.

Molecule node (m)	**Reaction node (rxn)**
{ "identity": 352, "labels": ["Molecule"], "properties": { "Hdonors": 0, "smiles": "Cc1ccc(C)c2ccccc12", "name": "P9)", "studyid": "MtPth_Map_031", "id": "MtPth_Map_031_6c2ca3b2_bebc_464e_93b5_888f1edf7e58" } }	{ "identity": 581, "labels": ["Reaction"], "properties": { "study_type": "Goat", "name": "MtPth_Map_031_3cb1b67c_40d3_4fa0_90ce_61f4d5d100f6_999579a7_cda3_4597_931f_6986da757d04", "studyid": "MtPth_Map_031" } }
{ "identity": 352, "labels": ["Molecule"], "properties": { "Hdonors": 0, "smiles": "Cc1ccc(C)c2ccccc12", "name": "P9)", "studyid": "MtPth_Map_031", "id": "MtPth_Map_031_6c2ca3b2_bebc_464e_93b5_888f1edf7e58" } }	{ "identity": 582, "labels": ["Reaction"], "properties": { "study_type": "Goat", "name": "MtPth_Map_031_ad1c8afb_3758_41b3_886c_d98896df1713_3cb1b67c_40d3_4fa0_90ce_61f4d5d100f6", "studyid": "MtPth_Map_031" } }
…	…

Table 6: Part of the JSON output of example 3

The output format shown for this example is also applicable to the other examples below but will not be shown as it is harder to visualize these examples.

Example 4

```
MATCH (r:Ring {ring_type:'aliphatic'})<-[:HAS_RING]-(m:Molecule)-[:REACTS_IN]
->(rxn:Reaction)-[:PRODUCES]->(m2:Molecule)-[:HAS_RING]->(r2:Ring
{ring_type:'aromatic'})
RETURN m, m2, rxn, r, r2
```

This query is a bit more complicated compared to the previous ones. It checks if there are molecule nodes in the database that are a reactant and have a ring node with an aliphatic ring type. Then, it checks if this reactant reacts into a molecule node, which is a product, and has a ring node with an

aromatic ring type. The nodes of the molecules, reactions, and rings involved are given as output, where the corresponding nodes are connected to each other with their edges.

Example 5

```
MATCH (m:Molecule {name:"Phthalic acid / PA"})--(rxn:Reaction)--(m2:Molecule)
RETURN DISTINCT  m2.name
```

The query above returns the distinct names of all molecule nodes that are a reactant or product in a reaction with the molecule 'Phthalic acid / PA'. A part of the output can be seen in table 7 below.

m2.name
Phthalamic acid / PaA
Phthalic anhydride / PAanh
Phosmet
Phthalimide / Pi
…

Table 7: Partial output of example 5

The cypher query stated above consists only of two lines of code in order to obtain the output table. It is interesting to make a comparison with the SQL query language, which is used for most relational databases. If you would create a graph-like database that can be queried with SQL, the query in figure 25 below would be needed to get the same output. As can be seen, the SQL code lines needed for retrieving the data are more than 10 times as long as the cypher query.

```
WITH q0 AS ( -- GetEdges: (m:Molecule)-[e1]->(rxn:Reaction) | attributes: m.name
SELECT "from" AS "m", edge_id AS "e1", "to" AS "rxn",… AS "m.name" FROM edge …),
q1 AS ( -- GetEdges: (m:Molecule)<-[e1]-(rxn:Reaction) | attributes: m.name
SELECT "from" AS "rxn", edge_id AS "e1", "to" AS "m",… AS "m.name" FROM edge …),
q2 AS ( -- UnionAll: q0 U q1
SELECT … FROM q0 UNION ALL SELECT … FROM q1),
q3 AS ( -- Selection: p.name = 'Alice'
SELECT * FROM q2 WHERE ("m.name" = 'Phthalic acid / PA')),
q4 AS ( -- GetEdges: (rxn:Reaction)-[e2]->(m2:Molecule) | attributes: m2.name
SELECT "from" AS "rxn", edge_id AS "e2", "to" AS "m2",… FROM edge …),
q5 AS ( -- GetEdges: (rxn:Reaction)<-[e2]-(m2:Molecule) | attributes: m2.name
SELECT "from" AS "m2", edge_id AS "e2", "to" AS "rxn",… FROM edge …),
q6 AS ( -- UnionAll: q4 U q5
SELECT … FROM q4 UNION ALL SELECT … FROM q5),
q7 AS ( -- Join
SELECT "left"."m", "left"."m.name", "left"."e1", "left"."rxn",
"right"."e2", "right"."m2", "right"."m2.name"
FROM q3 AS "left" INNER JOIN q6 AS "right" ON "left"."rxn" = "right"."rxn"),
q8 AS ( -- AllDifferent
SELECT * FROM q7 WHERE is_unique(ARRAY["e1", "e2"]))
-- Projection
SELECT "m2.name" FROM q8
```

Figure 25: SQL query

Example 6

```
MATCH (m:Molecule)-[b:HAS_ATOM]->(a:Atom)
WITH m, count(*) as at, collect(a) as atoms
WHERE at < 8
RETURN m, atoms
```

This query matches all molecules in the database that have 8 atoms or less and returns these molecule and atom nodes.

Summary

As can be seen in the different examples, the cypher queries to retrieve the data are relatively short and easy to understand. With a few lines of code, a quite complex query can be constructed, which would take a significantly larger amount of code in, for example, SQL. Additionally, the output visualization can help the user with better understanding and checking the query output.

6.3.2 Compatibility with graph analysis tools

In addition to the output stated in the section above, it is also valuable to be able to convert the data to other formats that have additional analysis options. One of these formats is a tensor, which is suitable to represent and do computations on graphs (Duchenne et al., 2011). Tensors can have several formats, but for the graph representation, with tensors, a n-dimensional array of a uniform type will be meant. A python package that is completely built to work with tensors that represent graphs is Pytorch Geometric[36] (PyG). In this section, three methods will be described that convert the graph in our database into input that works with PyG.

TUdata representation

For the first method, we looked at the examples of molecule representations for PyG that already exist in their standard datasets[37]. Here, two could be found: MoleculeNet[38] and the TUDataset[39]. The graph datasets in the TUDataset are from the TU Dortmund and are from different chemical studies. In their studies, they represent an atom as a node and a bond as an undirected edge. They have also built their own data loader for their data in PyG, which wants to receive the different tensors in .txt format. To create a graph with PyG, the interpreter needs at least the following tensors:

1. An adjacency matrix, that determines which atoms are linked to each other.
2. Node labels, which are the atom types.
3. A 'graph indicator', which determines to which graph the atoms belong.
4. Graph labels, in case of supervised graph analysis

Additionally, the following tensors can be added:

5. Edge labels, for additional information storage about the bonds.
6. Edge attributes, also for additional bond information.
7. Node attributes, for additional information storage of the atoms.
8. Graph attributes, for additional information about the molecules.

For our graphs to be represented in this format, the required tensors are 1-3 and 5, 7, and 8 in .txt format. Using the Neo4j driver for python, the needed output can be queried from the database, and with a few lines of code, the desired .txt format can be created. The data loader of the TUDataset does

[36] https://pytorch-geometric.readthedocs.io/en/latest/
[37] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html
[38] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html#torch_geometric.datasets.MoleculeNet
[39] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html#torch_geometric.datasets.TUDataset

not allow external data to be processed. Therefore, their code, which is available in the PyG documentation[40], has been adjusted to work with our data. This process can be found in the accompanying Jupyter notebook[41].

An example of the python code that is used to create the adjacency matrix (tensor 1) and edge labels (tensor 5) is shown below. As can be seen, it takes only a few lines of code to transform the Neo4j data. A part of the output of the code below is shown in table 8, where the ID(a1) and ID(a2) columns form the adjacency matrix and the bond_type column the edge labels. Note that, between two atoms, two directed edges in both ways must be created to be interpreted as an undirected edge for this format. This means, one directed edge from atom A to atom B and one directed edge from atom B to atom A. Additionally, the adjacency matrices of all molecules in the dataset are given in column ID(a1) and column ID(a2). For example, in table 8, 0 and 1 belong to a different molecule than 2 and 3.

```
# Cypher query to get the adjacency matrix from only the atoms
query = f"MATCH (m:Molecule)-[:HAS_ATOM]->(a1:Atom)-[:BONDED_WITH]->(b:Bond)<-
[:BONDED_WITH]-(a2:Atom)<-[:HAS_ATOM]-(m) RETURN ID(a1), ID(a2), b.bond_type"
bonds = pd.DataFrame(conn.query(query), columns=['ID(a1)', 'ID(a2)', 'bond_type'])

# Transform all categorical values to integers
bonds = to_int(bonds)

# Ensure that all IDs are sequential for the adjacency matrix
bonds2 = bonds
bonds2[['ID(a1)', 'ID(a2)']] = bonds2[['ID(a1)',
'ID(a2)']].replace(list(bonds['ID(a1)'].unique()), range(0, bonds['ID(a1)'].nunique()))
```

ID(a1)	ID(a2)	bond_type
0	1	1
1	0	1
2	3	2
3	2	2
4	6	1
…	…	…

Table 8: Example output of the python code

Using the above method, the graphs of the MOLfile database have been loaded into the PyG and a simple graph neural network (GNN) could be created. This GNN classifies quite accurately if a certain molecule, in this case, an alcohol, it is given will react or not. Also, for the graphs of the SMILES database, the data has been loaded successfully and a GNN has been created. The code can be found in the Jupyter notebook[42].

Heterogenous graph for PyG

In the PyG format above, it is not possible to create nodes with a different feature length because the format assumes that all nodes are in the same category (atoms). As a solution for this, PyG has created

[40] https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/datasets/tu_dataset.html
[41] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/2.1-MOLfiles%20Neo4j%20to%20pytorch%20TUdata.ipynb
[42] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/3.1-MOLfiles%20PyG%20TUData.ipynb

the so-called heterogeneous graph learning[43]. As the name implies, this lets you create graphs with different categories of nodes and bonds and each category can have its own number of features. To implement this for our graph data, the following should be given as input (all tensors in .txt format):

- The node files, for each different node category the features of every node are stored, in our case, these are the following nodes:
    - Molecules
    - Atoms
    - Bonds
    - Rings
    - Reactions
- The edge files, where for each edge category an adjacency matrix has to be made, it also has to be specified between which nodes these edges occur. This is done by stating 'Node_name_1', 'Edge_name', 'Node_name_2' in this order. When doing this for all nodes and edges the following is created:
    - 'molecule', 'contains_atom', 'atom'
    - 'molecule', 'contains_bond', 'bond'
    - 'molecule', 'contains_ring', 'ring'
    - 'atom', 'bonded_with', 'bond'
    - 'ring', 'has_atom', 'atom'
    - 'ring', 'has_bond', 'bond'
    - 'molecule', 'reacts_in', 'reaction'
    - 'reaction', 'produces', 'molecule'

As was the case for the previous method, the needed files can be created using the Neo4j driver for python to query from the database together with a few lines of python code[44]. For both databases, the data can be loaded in this format, and it is also possible to create a working GNN[45]. However, no significant performance results have been made yet.

MoleculeNet

As stated in the description of the TUdata, MoleculeNet is another data loader that is available in PyG and is built to load several datasets[46] from the study of Wu et al. (2017). In our case, the following .txt files have to be returned from the graph database to function as input for the MoleculeNet format:

1. An adjacency matrix, which determines which atoms are linked to each other.
2. Edge labels, which are the bond types.
3. Node attributes, the features of the atoms.

These files are generated by the '2.4-MOLfiles Neo4j to pytorch MoleculeNet.ipynb' Jupyter notebook[47] and loaded into PyG with the '3.4-MOLfiles PyG MoleculeNet.ipynb' Jupyter notebook[48]. For this data format, no additional test analysis is performed. Furthermore, it has only been tested with the MOLfiles dataset, not the SMILES.

[43] https://pytorch-geometric.readthedocs.io/en/latest/notes/heterogeneous.html
[44] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/2.3A-MOLfiles%20Neo4j%20to%20pytorch%20heterodata.ipynb and https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/2.3B-SMILES%20Neo4j%20to%20pytorch%20heterodata.ipynb.ipynb
[45] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/3.3-MOLfiles%20PyG%20heterodata.ipynb
[46] https://moleculenet.org/datasets-1
[47] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/2.4-MOLfiles%20Neo4j%20to%20pytorch%20MoleculeNet.ipynb
[48] https://github.com/StevenDrenth/ChemicalGraphDatabase/blob/main/3.4-MOLfiles%20PyG%20MoleculeNet.ipynb

6.4 Summary

In this section, the data input, output, and transformation of the graph database have been shown. Section 6.2 shows how MOLfiles or SMILES are loaded into the database with the description of the python code. Section 6.3 shows how the data can be queried using cypher and how the graph database data format can be transformed into other formats.

The transformation to the graph representation in section 6.2 works satisfactorily with the used datasets. However, the running time of the code can add up to hours when thousands of molecules have to be transformed and loaded into the database, which is something to consider. However, loading the data only has to be done only once, after which the queries are much quicker to carry out.

As stated in section 5.3.1, a graph database has both pros and cons, also compared to the standard relational database. Of the cypher examples in section 6.3.1, the first two examples can also be carried out by a relational database, which will probably perform better on a computational level. But our graph database is more convenient when it has to return the nodes, as only a few short and understandable query lines are needed to perform a complex operation. If, for instance, example 4 has to be written for a relational database, this would require a table of every node category, being the molecule, atom, bond, ring, and reaction. Then, each node in every node category table has to have a unique identifier to link them to the other corresponding nodes. A node can be connected to several other nodes, which has to be correctly described by the SQL query to get this data. For example, an atom node is connected to a molecule node and to at least one bond node. Additionally, this atom node could be connected to a ring node, which itself is connected to other atom and bond nodes and the molecule node. When this has to be specified for all node categories of the molecules in a reaction in order to query which reaction changes an aliphatic to an aromatic ring, the query will be long, messy, and full of different node categories and identifier names. This makes writing and understanding a SQL query code for these cases relatively hard and time-consuming. This can also be seen in example 5, where retrieving the same output with a SQL query is more than 10 times as long as the cypher query. Because the cypher query is easier to understand and relatively short, it is also less prone to human error.

Lastly, as stated in section 6.3.2, transforming the graph database data to another format is possible with only a few lines of additional python code for each category. This makes the database useful not only for programs that can work with the standard output formats of the graph database but also for a wide range of other applications. One of these tested applications a several input formats (tensors) for data loaders of Pytorch Geometric.

In conclusion, it can be the case that the graph database is not the fastest method concerning computing speed. However, this is compensated by the fact that the queries in cypher are significantly shorter and easier to understand than they would be with a relational database and SQL, which is more convenient. Therefore, the chemical graph database concept has more pros than cons.

7 Business impact

To determine the business impact, we will first look at the current situation. Then, the expected impact of this study is described, assuming that the graph database solution can be implemented at the company. Lastly, the possible future steps the company can take to get to the expected impact are stated.

7.1 Current situation

Presently, molecules and reactions are stored in different databases within the company, depending on the kind of research they belong to. The molecules are grouped by study ID and are in most situations represented as SMILES strings in combination with a visualized structural formula. For every study, the reactions are represented as arrows between these structural formulas, and they are also stored with IDs in a relational database.

The current situation facilitates the storage of chemical studies with their molecules and reactions. To retrieve the data, one can filter on study purposes and a similarity match can be done on the SMILES strings (given that the same SMILES string is used for each molecule). However, if one wants to know more specific details of a specific study, a human is needed to interpret the molecular structures and the corresponding reactions. Querying on specific molecule or reaction features is not possible in this representation. In other words, it is possible to filter on a molecule-specific level, but not on an atom-specific level: it is not possible to filter on the molecular structure.

7.2 Expected impact

The implementation of a chemical graph database will enable the user to query chemical data not only on a molecule-specific but also on an atom-specific level. This enables querying on specific parts of the reactions, molecules, bonds, atoms, rings, and their corresponding features. Therefore, an atom-specific query does not have to be performed by a human, that has to check every molecule in all selected studies individually, but the process can be automatized. The queries can run across all (selected) studies instead of looking at them separately, which is the case in the current situation. Therefore, the queried data can be returned at once for all the studies. Additionally, the query language of Neo4j, cypher, is easy to understand and complex operations can be performed with only a few query lines.

The chemical graph database should not be seen as a database where a part of the chemical data is stored, but as a general database where the entire chemical data of the company with all available information can be located. The desired output, in the desired format, can be queried from this database. This can be for chemical purposes, but also other data representations like tensors, as is shown in section 6.3.

Concluding, the implementation of the chemical graph database has the following potential impact on the company:

1. Queries can be run on specific parts of the reactions, molecules, bonds, atoms, rings, and their corresponding features. It is possible to filter/query on an atom level instead of a molecule level and complex operations can be performed with only a few query lines.
2. Because of point 1, employees can perform their tasks faster, as it allows them to search for reactions and molecules on an atom level across all (selected) studies.
3. The output is less prone to human error because a computer performs all aspects of the needed filtering. Additionally, a cypher query consists of significantly less lines than the same operation with, for example, a SQL query. This also leads to less errors that can be made in writing the query.

4. All chemical data is in the same place, which will improve data consistency.
5. It is possible to integrate data from different perspectives, studies, and departments, with which new insights can be created that were previously not reachable.

7.3 Future steps

To achieve the complete impact mentioned in the previous section, a significant amount of work must be done. In this short business impact section, it is impossible to compose a roadmap for this, but it is possible to give some steps that need to be taken to achieve the predicted impact:

- To get all chemical information in one graph database, the data in the different existing databases should be standardized in a way that it can be used as input for the chemical graph database. Additionally, determining which data should be entered in the graph database is important, as the database has to serve a broad spectrum of purposes in the ideal situation.
- All applicable data in the different databases must be transformed into graphs. This can be done by transforming the data of each database at its own speed. When one database is joined with another, they can be checked for duplicate reaction pathways and, if they exist, they can be joined.
- Newly created data should be immediately implemented as graphs in the graph database; this reduces the chance of input errors.

8 Conclusion

In this section, the research questions and their sub-questions stated in section 3 are answered. In this thesis, a chemical graph database has been created that enables running queries on specific parts of the molecules, reactions, bonds, atoms, rings, and their corresponding features. Additionally, the queries are short in length and easy to understand. Combined, this forms an efficient tool that can create output that was not previously possible with such ease.

8.1 Research question 1

The first research question asks what the different methods to represent molecules and chemical reactions are. To answer the first sub-question, we consider what the digital formats (try to) represent, which are chemical-structure graphs. For the second sub-question, we consider six digital representation formats of molecules and/or reactions together with their (dis)advantages. We searched for a format that represents the structural formula as directly as possible and that is relatively easy to query. The answer was found in the graph representation method because this method does not try to represent the structural formula, but it is a graph in essence. This answers the second sub-question. Together, the two sub-questions answer the first research question: It is beneficial to represent the structural formulas as directly as possible, and therefore a graph database is a legitimate choice.

8.2 Research question 2

The second research question asks how the chemical database can be constructed. This is split into two parts with two corresponding sub-questions. The first sub-question addresses what the conceptual design of the chemical graph database should be. For the design, the nodes and edges of the graph database together with their features are determined. The nodes, with corresponding features, represent the molecules, atoms, bonds, rings, and reactions, that all have a chemical reality. The nodes are connected with each other by labeled directed edges, that do not have features. The molecule consists of a molecule, atom, bond, and, if applicable, ring node, which is visualized in figure 26. A bond node connects two atom nodes with directed edges, where the edge has its head at the bond node for both atoms. The ring node is connected with all bond and atom nodes that are in this ring with a directed edge that has the tail at the ring node. All these nodes, which belong to the same molecule, are connected to the molecule node with a directed edge, that has its tail at the molecule node (not visualized in the picture).

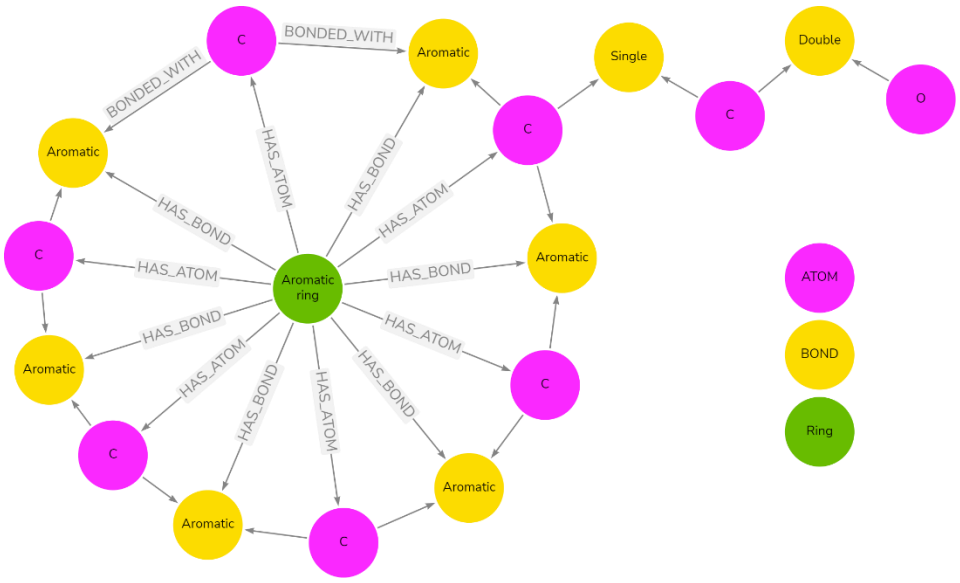


Figure 26: Example molecule representation of benzaldehyde

A reaction consists of one reaction node and two molecule nodes, an example is visualized in figure 27, where the blue nodes are the molecules, and the red node is the reaction. The reactant molecule node is connected to the reaction node with a directed edge that has its head at the reaction node. The product molecule node is connected to the reaction node with a directed edge that has its head at the molecule node.

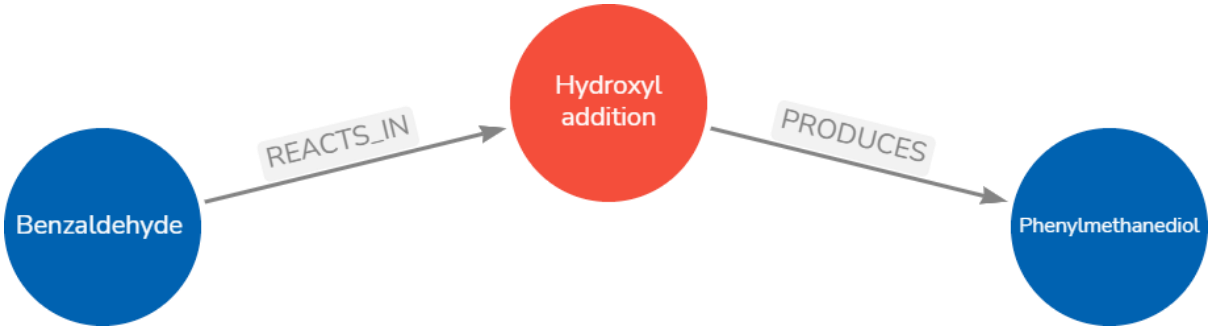


Figure 27: Example reaction

With the concept defined, the second sub-question is answered by stating the input format. The used formats are SMILES and MOLfiles because they are most commonly used and are available as test datasets. The input data is converted into graphs in Neo4j using a python script and the cypher query language.

8.3 Research question 3

The last research question asks how the created chemical graph database can be validated. Query examples are created of which the major part queries the molecules and reaction on a molecule-specific level. The queries are tested on their ease of use and length, and the output on correctness and ease of interpretation. On both, the database performed satisfactorily, which answers sub-question 3.1. Hereafter, the data in the chemical graph database is transformed into three other formats. For each format, this transformation is possible with just a few additional lines of python code per category. The two sub-questions give a proper answer to research question 3.

9 Limitations and further work

In this thesis, the design, implementation, and validation of the chemical graph database are presented. This section will elaborate on the limitation and further work.

9.1 Limitations

In this study, a proof of concept of a chemical graph database has been implemented and validated with the resources available. As is the case with most projects, there are some limitations to be taken into consideration with this study:

- SMILES strings are currently used as input data for the database, which means that two disadvantages of the SMILES format mentioned in section 5.2 also apply to the database. These disadvantages are that invalid molecules can be put in the database and no atom coordinates are available in this format.
- The input code for the MOLfiles dataset has only been tested with single reactions, as this was the MOLfile data that was available. The code has not been tested with reaction pathways, where a molecule can be a product of one reaction (or more) and a reactant of another reaction (or more).
- The chemical graph database has been tested with relatively small datasets for both the MOLfiles and SMILES input. Therefore, it is not known how a bigger dataset would affect the querying time, as well as the computation time of the data input and transformation.
- Only general chemical features, described by the studies of Duvenaud et al. (2015) and Kearnes et al. (2016), are stored in the different nodes of the chemical graph database. Not all of these features could be implemented in the same way and some features had to be left out due to the combination of input format and software limitations, as stated in section 6.1.
- The chemical graph database has been validated in section 6.3 but it would preferable if additional validations were done. These validations should focus on the scalability as well as the realizability of the chemical graph database.
- Representing not only the atoms, but also the bonds, rings, molecules, and reactions as nodes to be used as input for a GNN has not been done in any study known to the author. Getting all these nodes, their information, and connections are relatively hard to do, but this is facilitated by the chemical graph database. A starting effort has been made to create a GNN with the heterogeneous data loader of Pytorch Geometric, but no usable output has been generated yet.

9.2 Further work

Based on the limitations in the previous section, the following suggestions for further work are formulated:

- To further validate and adjust the graph database, testing and consulting should be done with several users from different backgrounds.
- When continuing to use SMILES as input, invalid SMILES should be filtered out of the input or adapted to SMILES that represent a valid molecule (e.g. SELFIES).
- To create a more stable input for reaction pathways, the functionality to create reaction pathways from MOLfiles can be added.
- Tests with larger datasets for both the MOLfiles as SMILES input can be performed to analyze what impact this has on the querying, input, and transformation of the data regarding computation power and time.

    - If the input speed would be too slow, a solution can be to write this code in a faster programming language. C++ would be a good option, as this is also the main language of RDKit. However, the added value of this should be further investigated.
- Tests to validate if the transformation to other file formats is impacted by larger and more complex data in the chemical graph database can be performed. It can be investigated by what means the computational power and computation time are affected.
- To make the chemical graph database more complete, additional features can be added, starting with adding the remaining features mentioned by Duvenaud et al. (2015) and Kearnes et al. (2016). Additionally, a method to obtain the atom coordinates when using SMILES as an input format should be found. Monitoring the ease of adding these features can be another validation of the database.
- More research on the heterogenous Pytorch Geometric GNNs in combination with the graph consisting of atom, bond, ring, molecule, and reaction nodes can be done. The impact of adding nodes with features and corresponding edges can be investigated.

10 References

Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, *40*(1). https://doi.org/10.1145/1322432.1322433

Bang-Jensen, J., & Gutin, G. Z. (2009). *Digraphs*. Springer London. https://doi.org/10.1007/978-1-84800-998-1

Biovia. (2020). *CTFILE FORMATS BIOVIA DATABASES 2020*. https://www.3ds.com/support/

Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., & Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, *71*(C), 58–63. https://doi.org/10.1016/j.ymeth.2014.08.005

Cioslowski, J., & Nanayakkara, A. (1993). Similarity of Atoms in Molecules. In *J. Am. Chem. Soc* (Vol. 115). https://pubs.acs.org/sharingguidelines

Coley, C. W., Barzilay, R., Jaakkola, T. S., Green, W. H., & Jensen, K. F. (2017). Prediction of Organic Reaction Outcomes Using Machine Learning. *ACS Central Science*, *3*(5), 434–443. https://doi.org/10.1021/acscentsci.7b00064

Dalby, A., Nourse, J. G., Hounshell, W. D., G., A. K., G., D. L., L., & B. A., & L. J. (1992). Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited. *Journal of Chemical Information and Computer Sciences*, *32*(3), 244–255. https://pubs.acs.org/sharingguidelines

Daylight Chemical Information. (n.d.). *SMARTS - A Language for Describing Molecular Patterns*. Retrieved October 19, 2022, from https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html

Duchenne, O., Bach, F., Kweon, I. S., & Ponce, J. (2011). A tensor-based algorithm for high-order graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(12), 2383–2395. https://doi.org/10.1109/TPAMI.2011.110

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R. P. (2015). *Convolutional Networks on Graphs for Learning Molecular Fingerprints*.

Gao, W., Mercado, R., & Coley, C. W. (2021). *Amortized Tree Generation for Bottom-up Synthesis Planning and Synthesizable Molecular Design*. http://arxiv.org/abs/2110.06389

Greg Landrum. (2019). *RDKit Documentation* (2019.09.01).

Gross, L. J., Jay Yellen, & Ping Zhang. (2014). *Handbook of graph theory* (Kenneth Rosen, Ed.; 2nd ed.).

Harada, S., Akita, H., Tsubaki, M., Baba, Y., Takigawa, I., Yamanishi, Y., & Kashima, H. (2020). Dual graph convolutional neural network for predicting chemical networks. *BMC Bioinformatics*, *21*. https://doi.org/10.1186/s12859-020-3378-0

Hartshorn, R. M., Hellwich, K. H., Yerin, A., Damhus, T., & Hutton, A. T. (2015). Brief guide to the nomenclature of inorganic chemistry. *Pure and Applied Chemistry*, *87*(9–10), 1039–1049. https://doi.org/10.1515/pac-2014-0718

Hasan, M. M., Habib, M. L., Anwaruzzaman, M., Kamruzzaman, M., Khan, M. N., & Rahman, M. M. (2020). Processing techniques of chitosan-based interpenetrating polymer networks, gels, blends, composites and nanocomposites. In *Handbook of Chitin and Chitosan: Volume 2: Composites and Nanocomposites from Chitin and Chitosan, Manufacturing and Characterisations* (pp. 61–93). Elsevier. https://doi.org/10.1016/B978-0-12-817968-0.00003-2

Hilderbrandt, R. L. (1969). Cartesian Coordinates of Molecular Models. *The Journal of Chemical Physics*, *51*(4), 1654–1659. https://doi.org/10.1063/1.1672229

Jiang, L., Zhao, X., Ge, B., Hu, S., Xiao, W., Shang, H., & Jing, Y. (2018). On minimal unique induced subgraph queries. *Applied Sciences (Switzerland)*, *8*(10). https://doi.org/10.3390/app8101798

Jin, W., Coley, C. W., Barzilay, R., & Jaakkola, T. (2017). *Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network*.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., & Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, *30*(8), 595–608. https://doi.org/10.1007/s10822-016-9938-8

Kim, S., Thiessen, P. A., Bolton, E. E., Chen, J., Fu, G., Gindulyte, A., Han, L., He, J., He, S., Shoemaker, B. A., Wang, J., Yu, B., Zhang, J., & Bryant, S. H. (2016). PubChem substance and compound databases. *Nucleic Acids Research*, *44*(D1), D1202–D1213. https://doi.org/10.1093/nar/gkv951

Krenn, M., Ai, Q., Barthel, S., Carson, N., Frei, A., Frey, N. C., Friederich, P., Gaudin, T., Gayle, A. A., Jablonka, K. M., Lameiro, R. F., Lemm, D., Lo, A., Moosavi, S. M., Nápoles-Duarte, J. M., Nigam, A., Pollice, R., Rajan, K., Schatzschneider, U., … Aspuru-Guzik, A. (2022). *SELFIES and the future of molecular string representations*. http://arxiv.org/abs/2204.00056

Krenn, M., Häse, F., Nigam, A., Friederich, P., & Aspuru-Guzik, A. (2020). *Self-Referencing Embedded Strings (SELFIES): A 100% robust molecular string representation*. https://doi.org/10.1088/2632-2153/aba947

Krystyna Bajer, Anne Seidlitz, Sascha Steltgens, & Bastian Wormuth. (2021). Graph Databases. In *The Digital Journey of Banking and Insurance: Vol. Volume III* (pp. 34–49). Springer International Publishing. https://doi.org/10.1007/978-3-030-78821-6

Morgan, H. L. (1964). The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, *5*(2), 107–113. https://pubs.acs.org/sharingguidelines

Murali, V., Königs, C., Deekshitula, S., Nukala, S., Santhi, M. D., & Athri, P. (2020). CompoundDB4j: Integrated Drug Resource of Heterogeneous Chemical Databases. *Molecular Informatics*, *39*(9). https://doi.org/10.1002/minf.202000013

Nelson, D. L., Lehninger, A. L., & Cox, M. M. (2008). *Lehninger principles of biochemistry*. https://archive.org/details/lehningerprincip00lehn_1/page/n1303/mode/2up

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., & Hutchison, G. R. (2011). Open Babel: An Open chemical toolbox. *Journal of Cheminformatics*, *3*(10). https://doi.org/10.1186/1758-2946-3-33

Pokorný, J. (2015). Graph databases: Their power and limitations. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9339*, 58–69. https://doi.org/10.1007/978-3-319-24369-6_5

Ralaivola, L., Swamidass, S. J., Saigo, H., & Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Networks*, *18*(8), 1093–1110.

Robert Belford. (2017). Anatomy of a MOL file. In *UALR 4399/5399: ChemInformatics*. https://chem.libretexts.org/@go/page/83691

Roberts, J. D., & Caserio, M. C. (1977). *Basic Principles of Organic Chemistry* (second edition).

Rogers, D., & Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, *50*(5), 742–754. https://doi.org/10.1021/ci100050t

Schilling, C. H., Schuster, S., Palsson, B. O., & Heinrich, R. (1999). Metabolic pathway analysis: Basic concepts and scientific applications in the post-genomic era. *Biotechnology Progress*, *15*(3), 296–303. https://doi.org/10.1021/bp990048k

Shen, C., Krenn, M., Eppel, S., & Aspuru-Guzik, A. (2021). Deep molecular dreaming: Inverse machine learning for de-novo molecular design and interpretability with surjective representations. *Machine Learning: Science and Technology*, *2*(3). https://doi.org/10.1088/2632-2153/ac09d6

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., & Tang, J. (2020). *GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation*. http://arxiv.org/abs/2001.09382

Sidorov, P., Varnek, A., Gimadiev, T., Nugmanov, R., Batyrshin, D., Madzhidov, T., & Maeda, S. (2021). Combined graph/relational database management system for calculated chemical reaction pathway data. *Journal of Chemical Information and Modeling*, *61*(2), 554–559. https://doi.org/10.1021/acs.jcim.0c01280

Todeschini, R., & Consonni, V. (2008). *Handbook of molecular descriptors*. John Wiley & Sons.

Trinajstic, N. (1992). *CHEMICAL GRAPH THEORY SECOND EDITION* (2nd ed.). CRC press.

TRIPOS. (2005). *Tripos Mol2 File Format*.

Tyagi, N., & Singh, N. (2017). Graph Database-An Overview of its Applications and its Types. *International Journal of Computer Science Engineering*, *2*(3), 6–10. http://www.ijcsejournal.org

Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, *28*(1), 31–36. https://pubs.acs.org/sharingguidelines

Wigh, D. S., Goodman, J. M., & Lapkin, A. A. (2022). A review of molecular representation in the age of machine learning. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *12*(5). https://doi.org/10.1002/wcms.1603

Willett, P., Barnard, J. M., & Downs, G. M. (1998). Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, *38*(6), 983–996. https://doi.org/10.1021/ci9800211

Wishart, D. S., Li, C., Marcu, A., Badran, H., Pon, A., Budinski, Z., Patron, J., Lipton, D., Cao, X., Oler, E., Li, K., Paccoud, M., Hong, C., Guo, A. C., Chan, C., Wei, W., & Ramirez-Gaona, M. (2020). PathBank: A comprehensive pathway database for model organisms. *Nucleic Acids Research*, *48*(D1), D470–D478. https://doi.org/10.1093/nar/gkz861

Wishart, D. S., Tzur, D., Knox, C., Eisner, R., Guo, A. C., Young, N., Cheng, D., Jewell, K., Arndt, D., Sawhney, S., Fung, C., Nikolai, L., Lewis, M., Coutouly, M. A., Forsythe, I., Tang, P., Shrivastava,

S., Jeroncic, K., Stothard, P., … Querengesser, L. (2007). HMDB: The human metabolome database. *Nucleic Acids Research*, *35*(SUPPL. 1). https://doi.org/10.1093/nar/gkl923

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., & Pande, V. (2017). *MoleculeNet: A Benchmark for Molecular Machine Learning*. http://arxiv.org/abs/1703.00564

You, J., Liu, B., Ying, R., Pande, V., & Leskovec, J. (2018). *Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation*.

11 Figures

12 Tables

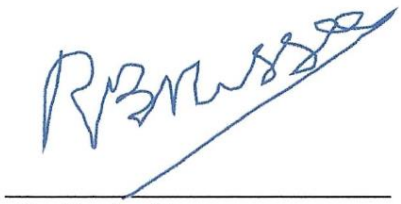
13 Equations

14 Appendices

14.1 Appendix A: Confidential data statement

Confidential Code and Data Statement

We hereby declare that we have seen and assessed the confidential SMILES dataset that Steven Drenth used in his thesis and that we approve of his work which is carried out with this dataset.



Dr. Rogier Brussee
Main supervisor
25-11-2022

Prof.dr.Jakob de Vlieg
Second assessor
25-11-2022