PhageOperonDB – a MySQL Database of Bacteriophages' Operons

Joana Gabriel[1], Luís Melo[1], and Oscar Dias[1]

Centre of Biological Engineering, University of Minho, Campus of Gualtar, Braga, 4710-057, Portugal

Abstract. Bacteriophages (phages) are viruses that only infect bacteria and play an important role in controlling the global antibiotic resistance crisis. Most phage genomes contain ORFs with no homology to any other sequence in databases due to their ability to evolve easily. So it is important to better understand their transcription and replication mechanisms. Since an operon is a cluster of one or more genes delimited by a promoter and a terminator, it is important to identify these regions to better understand the transcription of the phages' genes. However, there is no database exclusive for operons of phages so this project aims to develop a MySQL database exclusive for them, called PhageOperonDB.This database was created but only has 13 operons, so there is a need to obtain more information on phages operons and update the PhageOperonDB.

Keywords: Bacteriophages · Database · Operons · MySQL

1 Introduction

1.1 Bacteriophages

Bacteriophages (phages) are viruses that only infect bacteria, these entities are the most abundant and diverse on the planet [30]. With these phages, we can learn more about the mechanisms of transcription and replication. Recently studies showed the importance of phages in therapeutics since the abuse of antibiotics has been causing a global antibiotic resistance crisis [33, 14]. And these phages could be used to kill these multi-resistant bacteria. They also could be an excellent model to understand more about genetics, molecular biology, drug-targeting, genomics, and proteomics [3, 14]. However, we need to know more about the mechanisms of infection of each phage and for that, we need to study phage-host interactions. These studies could reveal more information about the evolutionary process of phages, the co-evolution of phages and hosts, the mechanism of phage infection, the effects of phage on host metabolism, and the complexity and diversity of phage-bacterium interactions [11]. Since phages have a small genome, due to limitations of their size, they can evolve to use more and more resources of the host to produce the viral progeny and so most bacteria have an enormous change in their genes expression due to phage infection [15, 11]. These events cause such diversity in phage sequences, that most genomes contain ORFs with no homology to any other sequences in databanks [5].

Phages have their genome “divided” into 3 sections: the early, middle, and later genes. The early genes initiate the infection, and are the ones needed to recruit the host machinery; the middle genes are needed for the production of all the components of the phage and the later genes are responsible for the assembly and release of the phages [35]. The phages that do not have their own RNAP have various ways to recruit the host RNAP and other transcriptional activator/regulators [8, 32], it depends on the phage strategy [3, 26]. However some are still not very clear, even with the development of the “omics”, but with them, we can have a better knowledge of the phage/host interactions on a transcriptional level [35].

1.2 Operons

An operon is a cluster of one or more genes delimited by a promoter and a terminator. For operon prediction, we can use many features of the genome, including experimental and computational data. Since they usually have the same properties [6] their prediction is facilitated: 1) an operon consists of one or more genes on the same strand of a genome sequence; 2) the intergenic distances within an operon are generally shorter than the distances of genes pairs without operons; 3) Generally, several genes of an operon have a common promoter and terminator, but regions within an operon usually do not contain any promotor or terminator; 4) Genes in an operon usually have related functions, and most belong to the same functional category, such as a cluster of orthologous groups (COG), GO terms and KEGG pathways, and 5) the genes in an operon as a functional unit tend to be found in more conserved gene pairs[7] and more similar phylogenetic profiles [27, 34]. Some sequence elements are also used to predict operons such as Short DNA Motifs, length ratio between a pair of Genes, and Synonymous codon usage biases (SCUB)[34].

Operon databases are available, however, they are not specific for bacteriophages, and many don't even have information on them. Some examples are OperonDB[7], MicrobesOnline[2], Operon Database (ODB)[17], RegulonDB[9], DBTBS[12] and Database of Prokaryotic Operons (DOOR)[13].

1.3 Database

A database is a collection of information/data that are organized and structured. It is typically stored on a computer system, and is controlled by a database management system (DBMS) together became the database system (or simply database) [23]. A DBMS functions as an interface between the database and the end user or programs and some of the most popular are MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database, and dBASE [23].

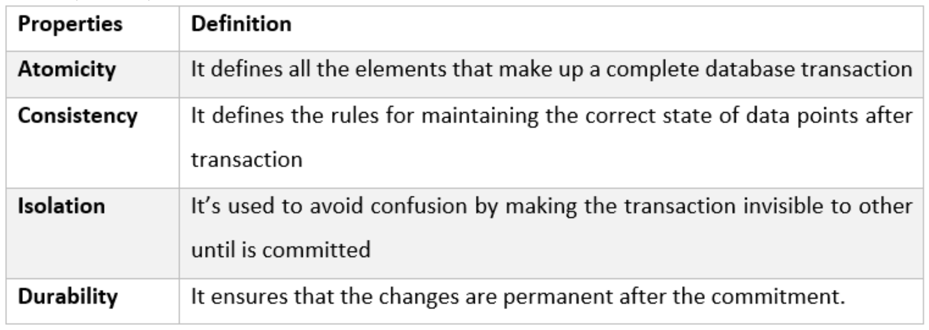
Nowadays data is typically organized in tables which consist in rows and columns which makes data querying more efficient. With this organization, the data can be easily accessed, managed, modified, updated, controlled, and organized. The Structured Query Language (SQL) is the language that is more popular for writing and querying data [23].

There are many types of databases classified depending on their system: the original system was the hierarchical database and the network database which were simple but inflexible. Then relational databases and object-oriented databases became popular [23]. With the growth of the internet, databases needed to become faster and be able of processing unstructured data, so the NoSQL databases appeared. However, there are a sixth and seventh types of databases, the cloud databases, and self-driving databases that were developed recently [23]. From now on we will be focused on Relational Databases.

Relational Databases In today's organizations, relational databases are one of the main technologies supporting information assets [28]. They store and provide access to related data points, they are based on the relational model that represents data in tables where each row is a record with a unique ID (key), and each column holds attributes of the data. In this way each record usually has a value for each attribute making it easy to establish the relationships among data points [24]. This model provides a standard way of representing and querying data that could be used by any application. It became even more popular with the use of SQL to write and query data in a database. This language is based on relational algebra and provides a consistent mathematical language that makes it easier to improve the performance of all database queries. For these reasons relational databases continue to be the most widely accepted model for databases [24].

When talking about policies for the commitments (permanent changes in the database) relational databases have strict rules. There are four crucial properties that define relational database transactions, referred to as ACID and they are described in Table 1 [24].

Table 1: The four crucial properties that define the relational database transactions (ACID) and their definition.



Properties	Definition
Atomicity	It defines all the elements that make up a complete database transaction
Consistency	It defines the rules for maintaining the correct state of data points after transaction
Isolation	It's used to avoid confusion by making the transaction invisible to other until is committed
Durability	It ensures that the changes are permanent after the commitment.

Relational Database Management System In order to Manage a Relational Database we need a Relational Database Management System (RDBMS) which is an advanced version of a DBMS because it can support distributed data. In these databases, the relations between tables are described formally in the "schema" [31]. Most relational DBMSs have network access to the database, have support for user authentication and access controls, which limit who can access the database and what they can do with the access, have support for backup and recovery of the database, and support for a variety of programming languages such as Java programming language, C, C++, Perl and Python [31].

A RDBMS permits the user to set constraints, these are a set of consistency rules and tests that can be used to prevent inappropriate values from being entered into the database and to maintain internal consistency. To avoid the problem of having multiple users modifying the same data, with an RDBMS we can lock a record so only one user at a time can be modifying that data. There are also "rolling back" techniques that permit undoing a modification [31].

Since RDBMS supports programming languages it allows programmers to write software that accesses the database to add or retrieve information. It also allows many database management tasks to be automated which is the key to providing a Web-based interface to the database [31].

Relation database products include Oracle [21], Microsoft SQL Server, IBM's DB2, and Informix. And also two popular open-source products: PostgreSQL and MySQL (that we will be focusing on) which are widely used in bioinformatics applications [31].

MySQL MySQL is an open-source relational database management system based on SQL. It was designed and optimized for web applications. MySQL became the platform of choice for web developers and web-based applications because it is designed to process millions of queries and thousands of transactions [23]. MySQL runs on Unix systems, Windows 2000, Linux, and on Macintosh, OS X. Is known for being fast and became the RDBMS of choice for Web site operations because it can operate large loads although it does not provide the full range of integrity checking that other offer and have an incomplete implementation of SQL. Many biological schemas are available for MySQL, and many full software systems use MySQL as a back-end such as EnsEMBL and UCSC genome browsers, and Airbnb, Uber, LinkedIn, Facebook, Twitter, and YouTube [22, 31].

1.4 Objectives

In this project, we aim to create PhageOperonDB, a MySQL database of phage operons known in the literature, in order to facilitate future studies on phage replication, their mechanisms of infection, and their transcriptome architecture.

2 Materials and Methods

The aim of this project is to develop a database on phages' operons, to do so the project was divided into two main steps. Firstly the collection of the phages' operons data to integrate the PhageOperonDB, starting with the database phiSITE[10] to find promoters and terminators of phages and then search the literature for operons containing these regulatory sequences. Secondly the construction of the MySQL Database, with the help of Python to better automate the process, inserting the data found in the first step. All the files that originated from this work were stored on GitHub (https://github.com/pg42870/PhageOperonDB).

2.1 Data Collection

To construct the dataset on phages' operons we used phiSITE database [10] to retrieve information available on promoters and terminators of phages, resulting in two FASTA files, one for promoters, and the other for terminators. This information was crossed to determine the list of phage species with identified promoters and terminators with the help of pandas [25], numpy [18], and BioPython[4] packages. This list was used to conduct a more direct search in the literature for known phages' operons, including the genes involved using the *Genome* database. The data collected was stored in a table format on a .xlsx file ('phages.xlsx').

2.2 Construction of the Database

The development of the PhageOperonDB was done using the information retrieved in the last step and with the help of *Python* to create scripts to manipulate this data and construct the database, using the mysql-connector-python[19] package. This consisted of various steps such as the construction of the Database Schema, the creation of the database and connection to the server, the creation of the database tables followed by the insertion of the data, and lastly the creation of Views to better access the information on the database.

The Database Schema Using MySQL Workbench[20] the PhageOperonDB schema was created. To choose the attributes and tables of the database we used the data collected and apply the normalization rules for the relational method.

Creation of the Database The creation of the PhageOperonDB was done using Python to access the MySQL server on a Jupyter Notebook to facilitate the connection and visualization of the results with the code below:

```
from getpass import getpass
import mysql

mydb = connect(host="localhost",
          user=input("Enter username: "),
          password=getpass("Enter password: "))

mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE PhageOperonDB")
```

For the creation of each table, was created an individual file to facilitate the updating of the different tables. And a file ('table_order.txt') with the order from each of the tables should be added to the database (in the Supplementary Files). And functions in Python were created to read these files and create the tables in the PhageOperonDB:

```
import mysql
def create_tables(connection, cursor, file_list):
    f = open(file_list, 'r')
    files = f.readlines()
    for t in files:
        query = create_query(t.strip())
        cursor.execute(query)
    connection.commit()

def create_query(file):
    f = open(file, 'r')
    query = " ".join(f.readlines())
    return query

create_tables(mydb, mycursor, 'table_order.txt')
```

Insertion of the data in the Database In order to insert the data on the corresponding tables of the database, first manipulation of the data collected was needed, using pandas[25] and numpy[18] packages. The data was divided for the different tables and a Python file originated, with two variables: the query and the values as shown in the example below for the Operon Table:

```
insert_operons="""
    INSERT INTO PhageOperonDB.Operon (Name, idReferences)
    VALUES (%s, %s)
    """

operon_records=
    [('gem operon','1'),('Mom operon','2'), ('I operon','3'),
    ('lys operon','3'), ('P operon','3'),('middle operon',
    '4'), ('T7 early operon','5'), ('phage lambda leftward
    operon','5'), ('phage lambda rigthward operon','6'),
    ('phage lambda later operon','6'),('lambda rexAB','7'),
    ('gtrABC operon','8'),('PL operon','9'),]
```

To execute the insertion of the data on PhageOpeornDB, the Python file was run followed by this command:

```
mycursor.executemany(insert_operons, operon_records)
mydb.commit()
```

Creation of Views Some views were originated to better research the PhageOperonDB. Those views were `'Genes_by_Operons'`, `'Promoter_by_Operons'`, `'Terminator_by_Operons'`, `'Organisms_per_Operon'`, `'Number_of_Operons'`, `'Number_of_Genes'`, `'Number_of_Organisms'` and `'Operons'`.

3 Results and Discussion

The construction of a database for phages' operons can present several challenges.

3.1 Data Collect

The first step to collecting information on phages' operons, since there wasn't a database with them, was to try to automate the process and search the GEO database for operons in phages. However, this approach didn't work. The majority of the results were of operons in the phages' host and not on the phage itself. Also, since the phages genome is divided into three sections, and this division is a temporal one, this results in not describing or naming the operons, and just calling it by early genes, middle genes, or late genes, without specifying if it's an operon or not. So the next step consisted of finding a database that has promoters and/or terminators of phages. With this information is easier to find if these motifs belong to an operon or not. The database found was phiSITE[10], that have information on gene regulation in phages. In this database, we could find 488 promoters, from which 329 were experimentally confirmed, and 177 terminators, from which 177 were experimentally confirmed. After combining this information we ended up with 29 species of phages that have promoters and/or terminators confirmed (complete list in 'bacteriohages_list.txt' file in the Supplementary files). This shows us that there are many species of phages that don't have confirmed information about these regulatory sequences, so more studies, and tools to predict operons in phages should be done.

With this information, was done a manual search through the literature with the support of the Genome[16] database, to find operons and the genes involved in them.

This search resulted in 13 operons found, from 4 species *(Enterobacteria phage Mu, Enterobacteria phage T7, Enterobacteria phage lambda and Salmonella phage P22)*, 22 promoters, 15 terminators, and 103 genes (Complete table with all information on the file 'phages.xlsx' on the Supplementary files). This is a very low number of operons, so in the future, more should be added.

3.2 Database Schema

To construct PhageOperonDB with the information collected, first, we needed to create the Database Schema (Figure 1), to determine which tables and attributes the database should have.

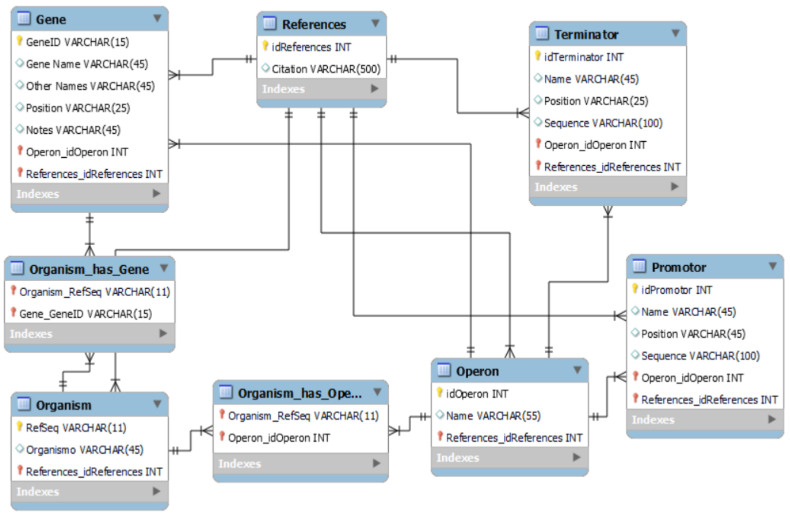


Fig. 1: The PhageOperonDB schema that shows all the tables in the database, their attributes and the relations between all the tables.

For that, we created 6 tables with different attributes:

1) **References Table**: id (primary key) and Citation;

2) **Operon Table**: id (of the Operon as a primary key), Name (of the Operon), idReferences (if there is a citation associated as Foreign key to related with the References Table);

3) **Organism Table**: RefSeq (of the Organism as a primary key), Organism (Name of the Organism), idReferences (if there is a citation associated as Foreign key to related with the References Table);

4) **Gene Table**: id (GeneID of NCBI as a primary key), Name, Other Names, Position (of the gene in the genome), Notes, Operon_id (id of the Operon as a Foreign key to relating with the Operon Table) and idReferences (if there is a citation associated as Foreign key to related with the References Table));

5) **Promoter Table**: id (of the Promoter as a primary key), Name (of the promoter), Position (of the promoter on the genome of the organism), Sequence (of the promoter), Operon_id (id of the Operon as a Foreign key to relating with the Operon Table) and idReferences (if there is a citation associated as Foreign key to related with the References Table));

6) **Terminator Table**: id (of the Terminator as a primary key), Name (of the terminator), Position (of the terminator on the genome of the organism), Sequence (of the terminator), Operon_id (id of the Operon as a Foreign key to relating with the Operon Table) and idReferences (if there is a citation associated as Foreign key to related with the References Table));

Since there is many to many relationship between the Organism Table and the Operon Table there was a need to add a junction table. The same occurs in the relation between the Organism Table and the Gene Table:

1) **Organism_has_Operon Table**: RefSeq (of the Organism) and Operon_id (both being at the same time primary and foreign keys);

2) **Organism_has_Gene Table**: RefSeq (of the Organism) and GeneID (both being at the same time primary and foreign keys).

All these tables have the essential information to understand an operon, by having its name, the organism it occurs on, the genes involved, and the promoters and terminators, however, this could be improved. We could add more attributes in order to simplify the search for other scientists, for example, links for other databases such as NCBI[29], the function of the resulting transcripts of these genes, and even Uniprot [1] links for the known proteins that are involved in these operons.

3.3 Creation of the Database

This database called PhageOperonDB is in localhost, however, if needed it can be transferred to another host in order to become public and be accessed by other scientists.

3.4 Insertion of the data

By using a Python file for each table to insert the data, it became easier to add more information to the database by simply modifying and running the file. However, for the Organism_has_Operon and Organism_has_Gene Tables on the construction of the Python files the values for one of the attributes were inserted manually, it was the easiest way to correlate these attributes. Not all the attributes in all the tables have data, some of them are Null. This is due to the fact that the information collected was not complete for all the attributes on the database, however, they can be added in the future, only by the INSERT command, not needed to update the structure and relations of the tables.

On the Table References we have 9 references, all for the operons. Since we have 13 Operons on the Operon Table, this means that some operons shared a reference. For the Organims Table, we have 4 organisms, it will be encouraged to find more operons in different species, to diversify as much as possible the PhageOperonDB. As for the Gene Table, we have found 103 genes, in relation to the number of operons this is a good number, but is due to the fact that the search for these genes was done manually through the Gene database of NCBI[29]. Resulting in 22 promoters and 15 terminators on the database. We could have added all the promoters and terminators of the phiSITE[10], but there wouldn't be a corresponding operon. These numbers also showed us that for some operons that weren't found the corresponding terminator.

3.5 Creation of the Views

In a relational database, the data is organized in tables with established relations between them, and the data could be accessed through the SELECT command, however, some of these commands are more complex, and many of recurrent use, so a VIEW is created. This is the case with PhageOperonDB, if we wanted to see the information of an operon, including, the promoter, terminator, and genes involved with said operon we will need a big SELECT command. So instead we have a need to create a VIEW.

Various VIEWS were created:

1) The Genes_by_Operons shows the different Operons and the genes that are involved in which one; We could also query the View in order to select only one of the operons and the corresponding genes (as shown in the example below where we want to see the genes involved in the PL operon);

```
# genes of the PL operon
mycursor.execute('SELECT GeneID, Gene_Name
                  FROM Genes_by_Operons
                  WHERE Operon_Name = "PL operon"')
result = mycursor.fetchall()
for row in result:
     print(row)

#output (GeneID, Gene_Name)
('1262785', 'kil')
('1262787', 'c3')
('1262789', '17')
('1262791', 'orf87')
('1262800', 'erf')
('1262802', 'abc1')
('1262804', 'abc2')
('1262810', 'sieB')
('1262811', 'ral')
('1262841', '24')
('1262843', 'orf78')
('1262850', 'arf')
('2944229', 'esc')
```

2) The Promoter_by_Operons and Terminator_by_Operons worked in the same way as the Gene_by_Operons but instead of genes we have promoters and terminators respectively;

3) The Organisms_per_Operons that show for each Organism the Operons associated, we could also specify which organism we want to query, for example,

if we want only the operons present in the *Entereobacteria phage Mu* we could run the code below:

```
# operons for the Entereobacteria phage Mu
mycursor.execute('SELECT Operon_id, Operon_Name
                  FROM Organims_per_Operon
                  WHERE Organism = "Entereobacteria phage Mu"')
result = mycursor.fetchall()
for row in result:
    print(row)

#output (Operon_id, Operon_Name)
(1, 'gem operon')
(2, 'Mom operon')
(3, 'I operon')
(4, 'lys operon')
(5, 'P operon')
(6, 'middle operon')
```

4) The Number_of_Operons, Number_of_Genes, and Number_of_Organisms count the number of operons, genes, and organisms respectively that exist in the database. These views are needed to better understand the database and its content. And in case of a new insertion of data, we could see if it really added it or not;

5) Lastly we have the Operons views, which consist of querying the database for all the information related to the operons at the same time. The SELECT command for this view is relatively big, so this is the most needed view of the PhageOperon. Here are the first 5 results of this view:

```
    ('gem operon', 'Entereobacteria phage Mu ',
    'gemA', 'pGEM promoter', 't9.2 terminator')
    ('gem operon', 'Entereobacteria phage Mu ',
    ' gemB', 'pGEM promoter', 't9.2 terminator')
    ('Mom operon', 'Entereobacteria phage Mu ',
    'Mup54 (gpCom)', 'pMom promoter', 'tMom terminator')
    ('Mom operon', 'Entereobacteria phage Mu ',
    'Mup55 (Mom)', 'pMom promoter', 'tMom terminator')
    ('lys operon', 'Entereobacteria phage Mu ',
    'Mup24', 'pLyz promoter', 'tLys(b) terminator')
```

This view shows us a resume of the database, however, it could be updated to show more information, such as the RefSeq of the Organisms, GeneID of the Genes, Position of the Promoters and Terminators, and the Reference of the operon.

The Operons View has a problem that consists in not showing the operons that don't have a defined terminator or promoter, to solve this problem we will need to add 'unknown' to those null terminators and promoters of the operon. However, this will create a need to update the Number_of_Terminators and Number_of_Promoters views to not count the 'unknown'.

Even with all these views we should create more in the future, especially if we added more data. And some Python functions should be defined too, in order to facilitate the searching, for example, we could create a function that receives the Operon from which we need the information and the function returns the results of the Operons View but only for that operon, and there would not be a need to query the database ourselves.

At this point there is no need to create an index for the database since it doesn't have that much data, however, that will be a good addition to the database when it has more data, to increase the performance of the querying processes.

4 Conclusion

The goal of this project was to create a MySQL database for phages' operons called PhageOperonDB. This database has only 13 operons from 4 organisms. It has 6 tables and 2 junctions tables. As 8 Views that query the database. It is a local database with the objective of becoming a public one to help scientists with the study of PhageOperons. However, the PhageOperonDB needs to be updated with more operons, and more information for each one. In the future, all solutions presented in this project could be explored to improve the PhageOperonDB. With that, this database could be used as a dataset to explore the prediction of operons on phages.

References

1. Uniprot: the universal protein knowledgebase in 2023. Nucleic Acids Research **51**(D1), D523–D531 (2023)
2. Alm, E.J., Huang, K.H., Price, M.N., Koche, R.P., Keller, K., Dubchak, I.L., Arkin, A.P.: The microbesonline web site for comparative genomics. Genome research **15**(7), 1015–1022 (2005)
3. Berdygulova, Z., Westblade, L.F., Florens, L., Koonin, E.V., Chait, B.T., Ramanculov, E., Washburn, M.P., Darst, S.A., Severinov, K., Minakhin, L.: Temporal regulation of gene expression of the thermus thermophilus bacteriophage p23-45. Journal of molecular biology **405**(1), 125–142 (2011)
4. BioPython: Biopython, https://biopython.org/
5. Blasdel, B.G., Chevallereau, A., Monot, M., Lavigne, R., Debarbieux, L.: Comparative transcriptomics analyses reveal the conservation of an ancestral infectious strategy in two bacteriophage genera. The ISME journal **11**(9), 1988–1996 (2017)
6. Chen, X., Su, Z., Dam, P., Palenik, B., Xu, Y., Jiang, T.: Operon prediction by comparative genomics: an application to the synechococcus sp. wh8102 genome. Nucleic acids research **32**(7), 2147–2157 (2004)
7. Ermolaeva, M.D., White, O., Salzberg, S.L.: Prediction of operons in microbial genomes. Nucleic acids research **29**(5), 1216–1221 (2001)

8. Hsieh, M.L., James, T.D., Knipling, L., Waddell, M.B., White, S., Hinton, D.M.: Architecture of the bacteriophage t4 activator mota/promoter dna interaction during sigma appropriation. Journal of Biological Chemistry **288**(38), 27607–27618 (2013)
9. Huerta, A.M., Salgado, H., Thieffry, D., Collado-Vides, J.: Regulondb: a database on transcriptional regulation in escherichia coli. Nucleic acids research **26**(1), 55–59 (1998)
10. Klucar, L., Stano, M., Hajduk, M.: phisite: database of gene regulation in bacteriophages. Nucleic acids research **38**(suppl_1), D366–D370 (2010)
11. Li, T., Zhang, Y., Dong, K., Kuo, C.J., Li, C., Zhu, Y.Q., Qin, J., Li, Q.T., Chang, Y.F., Guo, X., et al.: Isolation and characterization of the novel phage jd032 and global transcriptomic response during jd032 infection of clostridioides difficile ribotype 078. MSystems **5**(3) (2020)
12. Makita, Y., Nakao, M., Ogasawara, N., Nakai, K.: Dbtbs: database of transcriptional regulation in bacillus subtilis and its contribution to comparative genomics. Nucleic Acids Research **32**(suppl_1), D75–D77 (2004)
13. Mao, F., Dam, P., Chou, J., Olman, V., Xu, Y.: Door: a database for prokaryotic operons. Nucleic acids research **37**(suppl_1), D459–D463 (2009)
14. Miller, E.S., Kutter, E., Mosig, G., Arisaka, F., Kunisawa, T., Ruger, W.: Bacteriophage t4 genome. Microbiology and molecular biology reviews **67**(1), 86–156 (2003)
15. Mojardín, L., Salas, M.: Global transcriptional analysis of virus-host interactions between phage ϕ29 and bacillus subtilis. Journal of virology **90**(20), 9293–9304 (2016)
16. NCBI: Genome databse, https://www.ncbi.nlm.nih.gov/genome/
17. Okuda, S., Katayama, T., Kawashima, S., Goto, S., Kanehisa, M.: Odb: a database of operons accumulating known operons across multiple genomes. Nucleic acids research **34**(suppl_1), D358–D362 (2006)
18. Oliphant, T.E.: A guide to NumPy, vol. 1. Trelgol Publishing USA (2006)
19. Oracle: Mysql, https://dev.mysql.com/doc/connector-python/en/connector-python-introduction.html
20. Oracle: Mysql workbench, https://dev.mysql.com/doc/workbench/en/
21. Oracle: Oracle, https://www.oracle.com/
22. Oracle: Quickly develop and deploy cloud applications, https://www.oracle.com/mysql/
23. Oracle: What is a database?, https://www.oracle.com/database/what-is-database/
24. Oracle: What is a relational database?, https://www.oracle.com/database/what-is-a-relational-database/
25. Pandas Development Team, T.: Pandas-dev/pandas. Pandas (2020)
26. Pavlova, O., Lavysh, D., Klimuk, E., Djordjevic, M., Ravcheev, D.A., Gelfand, M.S., Severinov, K., Akulenko, N.: Temporal regulation of gene expression of the escherichia coli bacteriophage phieco32. Journal of molecular biology **416**(3), 389–399 (2012)
27. Pellegrini, M., Marcotte, E.M., Thompson, M.J., Eisenberg, D., Grothe, R., Yeates, T.O.: Assigning protein functions by comparative genome analysis protein phylogenetic profiles (May 13 2003), uS Patent 6,564,151
28. Ramalho, J.C., Ferreira, B., Faria, L., Ferreira, M.: Beyond relational databases: preserving the data. New Review of Information Networking **25**(2), 107–118 (2020)

29. Sayers, E.W., Beck, J., Bolton, E.E., Bourexis, D., Brister, J.R., Canese, K., Comeau, D.C., Funk, K., Kim, S., Klimke, W., et al.: Database resources of the national center for biotechnology information. Nucleic acids research **49**(D1), D10 (2021)
30. Srinivasiah, S., Bhavsar, J., Thapar, K., Liles, M., Schoenfeld, T., Wommack, K.E.: Phages across the biosphere: contrasts of viruses in soil and aquatic environments. Research in Microbiology **159**(5), 349–357 (2008). https://doi.org/https://doi.org/10.1016/j.resmic.2008.04.010, https://www.sciencedirect.com/science/article/pii/S0923250808000612, exploring the prokaryotic virosphere
31. Stein, L.: Creating databases for biological information: an introduction. Current protocols in bioinformatics (1), 9–1 (2003)
32. Twist, K.A.F., Campbell, E.A., Deighan, P., Nechaev, S., Jain, V., Geiduschek, E.P., Hochschild, A., Darst, S.A.: Crystal structure of the bacteriophage t4 late-transcription coactivator gp33 with the β-subunit flap domain of escherichia coli rna polymerase. Proceedings of the National Academy of Sciences **108**(50), 19961–19966 (2011)
33. Viertel, T.M., Ritter, K., Horz, H.P.: Viruses versus bacteria—novel approaches to phage therapy as a tool against multidrug-resistant pathogens. Journal of Antimicrobial Chemotherapy **69**(9), 2326–2336 (05 2014). https://doi.org/10.1093/jac/dku173, https://doi.org/10.1093/jac/dku173
34. Wang, Y., Zhou, Y., Zhou, C., Wang, S., Du, W., Zhang, C., Liang, Y.: Approaches and Methods for Operon Prediction based on Machine Learning Techniques, chap. 21, pp. 449–477. John Wiley Sons, Ltd (2011). https://doi.org/https://doi.org/10.1002/9780470892107.ch21, https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470892107.ch21
35. Yang, H., Ma, Y., Wang, Y., Yang, H., Shen, W., Chen, X.: Transcription regulation mechanisms of bacteriophages. Bioengineered **5**(5), 300–304 (2014). https://doi.org/10.4161/bioe.32110, https://doi.org/10.4161/bioe.32110, pMID: 25482231