Regulatory Mechanisms in Biosystems

ISSN 2519-8521 (Print)
ISSN 2520-2588 (Online)
Regul. Mech. Biosyst.

Implementation and data mining of external biological databases

V. Langraf*, K. Petrovičová**, V. V. Brygadyrenko***, ****

**Constantine the Philosopher University in Nitra, Nitra, Slovakia*
***University of Agriculture in Nitra, Nitra, Slovakia*
****Oles Honchar Dnipro National University, Dnipro, Ukraine*
*****Dnipro State Agrarian and Economic University, Dnipro, Ukraine*

Article info
Received 10.05.2025
Received in revised form 12.06.2025
Accepted 08.07.2025

Department of Zoology and Anthropology, Constantine the Philosopher University in Nitra, Tr. A. Hlinku, 1, Nitra, 94901, Slovakia.
E-mail: langrafvladimir@gmail.com

Institute of Plant and Environmental Sciences, University of Agriculture in Nitra, Tr. A. Hlinku 2, Nitra, 94901, Slovakia.
E-mail: kornelia.petrovicova@gmail.com

Department of Biodiversity and Ecology, Oles Honchar Dnipro National University, Nauky av., 72, Dnipro, 49010, Ukraine.
E-mail: brigad@ua.fm

Department of Normal and Patological Anatomy of Farm Animal, Dnipro State Agrarian and Economic University, Sergiy Efremov st., 25, Dnipro, 49000, Ukraine.

Langraf, V., Petrovičová, K., & Brygadyrenko, V. V. (2025). Implementation and data mining of external biological databases. Regulatory Mechanisms in Biosystems, 16(3), e25114. doi:10.15421/0225114

The implementation of external biological databases is a key approach that allows researchers to consolidate scattered information from different sources into a collaborative unified system. In practice, this means that data from projects such as GenBank, UniProt, and Ensembl are automatically retrieved, transformed into a unified format, and stored in a relational or NoSQL database using ETL processes. This approach ensures that sequence data, gene annotations, or protein information are always consistent and ready for further analysis, eliminating the risk of manual copying or incorrect mapping of entities. The aim of this study was to design and implement a process for integrating data from an external ITIS (Integrated Taxonomic Information System) into a relational database in a Microsoft SQL Server environment. After analysing the ITIS schemas and data formats, we prepared tools for automated ETL (Extract, Transform, Load), which loaded 19 source files with taxonomic and metadata data using bulk import (BULK INSERT). Data normalisation and consistency checking ensured reliable linking of entities (identifiers, authors, comments, and vernaculars). To demonstrate the usefulness of the solution, we performed a preliminary SQL data extraction analysis: we found that the database contains 107,540 unique references to genera, of which the most numerous is the genus Euphorbia (5,009 records); the most comments on taxa were added in 2015 and 2001; and the highest frequency of publications was recorded in 2018-2023. These results confirm the suitability of MS SQL for systematic taxonomy studies and open up space for further automation of updates and expansion of the analysis to include temporal or geolocation trends.

Keywords: ITIS; SQL; query; data quality; SSMS.

Introduction

The implementation of external biological databases is a key element of modern bioinformatics solutions, enabling the integration, analysis, and visualisation of large biological data. External databases such as GenBank, UniProt, PDB or Ensembl provide well-documented APIs, structured formats and rich reference information that can be incorporated into your own systems. When designing the architecture for accessing these resources, it is important to choose the right communication model – be it a RESTful interface, SOAP services, or dedicated libraries such as BioPython while ensuring efficient management of large volumes of data, minimising latency and optimizing transmission (Baxevanis, 2011; Coronel & Morris, 2014).

The implementation is based on understanding of the data schemas of external systems. For example, GenBank manages sequence records using a hierarchical record structure with sections for organism information, annotations, and metadata (Benson et al., 2018). Similarly, UniProt provides protein records in RDF or XML format, which facilitates their automated parsing and transformation into a local relational or graph database (Cochrane et al., 2010). On the server side, robust client modules that support pagination, retries on failures, and parallel downloads need to be implemented, thereby achieving scalability to handle millions of records (Berman et al., 2000).

Redundancy removal and data normalisation are the next important steps. When combining multiple sources (e.g. linking GenBank and Ensembl records at the gene identifier level), it is necessary to use official mapping files or online services such as BioMart or UniProt ID mapping (Kasprzyk 2011). These services allow the consolidation of heterogeneous identifiers and facilitate subsequent analysis. To maintain consistency, it is recommended to regularly compare the local state with versions of external databases, while it is possible to use data on the last update date available in API responses (Akoka et al., 2017). The central place in the integration is occupied by the ETL (Extract, Transform, Load) protocol. During extraction, it is important to use efficient streaming mechanisms, for example, parsing XML using SAX instead of loading the entire document into memory, or directly downloading in 'chunks' for large FASTA files (Kanehisa et al., 2020). Transformation includes format validation, entity normalisation, and mapping to internal data models, often implemented using frameworks such as Apache NiFi or custom Python scripts based on the BioPython and Pandas libraries. Finally, the data is loaded into a persistence layer – optimally a relational database (PostgreSQL, MySQL) or a NoSQL system (MongoDB, Neo4j), depending on the query and connection requirements. The security of access to external sources cannot be overlooked. Most databases require registration and allocate API keys that need to be stored in a secure repository (e.g., HashiCorp Vault) and never included directly in the code. Role-based authentication (RBAC), HTTPS transmission encryption, and encrypted backups are applied to protect sensitive data and control access on the local system (Afgan et al., 2018).

When integrating external biological databases, offline access and archiving of older versions should also be considered. Databases such as Sequence Read Archive (Leinonen et al., 2010) or GEO (Barrett et al., 2013) have huge volumes of data, so it is advisable to use local mirrors or cloud archives with low access costs (e.g. AWS S3). Version management is usually solved using metadata, where the source version and the file checksum are recorded with each download. Standardisation and interoperability, provided by ontologies such as Gene Ontology or OBO Foundry (Smith et al., 2007), allow for uniform labelling of biological entities and relatively easy extension of the implementation to other sources.

Data mining on external biological databases represents an interdisciplinary challenge that combines concepts from the fields of bioinformatics, data integration, and advanced analytical methods. The data mining process, often referred to as Knowledge Discovery in Databases (KDD), can be divided into several phases: selection and extraction of data from external sources, data preprocessing and cleaning, transfor-

mation and integration of heterogeneous formats, custom data mining algorithms, and finally interpretation and visualisation of the obtained patterns (Gligorijević & Pržulj, 2015).

Data mining in SQL is not exactly the same as data mining techniques using specialised tools such in Python, R, but SQL can be an excellent tool for preliminary analysis, aggregation, and pattern discovery – which is the basis of data mining. A key element of data mining is advanced SQL constructs. Common Table Expressions (CTE) and recursive queries allow efficient analysis of glycosylation hierarchies or lineage relationships of organisms, while functions (OVER, RANK, ROW_NUMBER) simplify the calculation of shifts in gene expression time series or the detection of extreme values in protein concentrations. Using GROUPING SETS or ROLLUP, summaries at multiple levels of detail can be generated without multiple separate queries (Luscombe et al., 2001).

The purpose of the contribution is to prepare code in Microsoft SQL Server Management Studio to implement data from the external ITIS database and subsequent data mining.

Materials and methods

We used source code written in SQL (Structured Query Language) to import data from the external Integrated Taxonomic Information System (ITIS) database. We then performed data mining on these data using Microsoft SQL Server Management Studio 2017 (Microsoft. Microsoft SQL Server. (2017): (RTM) - 14.0.1000.169 (X64) Aug 22 2017 17:04:49. Microsoft Corporation Express Edition (64-bit) on Windows 10 Home 10.0 [X64](Build 18362:)).

The integrated taxonomic information system stores taxonomic information about plants, animals, fungi, and microbes from North America and around the world. It provides a comprehensive taxonomy of global species that enables discovery, indexing, and linking of biodiversity information across all human activities.

Results

For practical processing of the implementation of external data, we chose the Integrated Taxonomic Information System database. On the website (www.itis.gov), ITIS offers the possibility of downloading data, which we can then import into our database. In the Get ITIS Data option, we select Full Database Download and then Database Files. The page will load, and we select the Full ITIS Data Set (MS SQL Server) option – itisMSSql.zip. We download the data to our computer and then unzip the itisMS file. After unzipping, we have data from the database in 19 files, which represent a data table in the database with the following names: comments, experts, geographic_div, hierarchy, jurisdiction, kingdoms, longnames, nodc_ids, other_sources, publications, reference_links, strippedauthor, synonym_links, taxon_authors_lkp, taxon_unit_types, taxon_unit_types, taxon_unit_types, vern_ref_links, vernaculars.

After downloading the data, we created a database called ITIS in Microsoft SQL Server Management Studio using the CREATE DATABASE ITIS command. We then wrote the code (SQL) to create the tables that we downloaded. The columns in the tables had data types assigned to them according to the data that were stored in them. The numeric values in the columns had data types int and smallint. The columns where the text formats were stored had the data types varchar and char. The dates and time updates had data types datetime and smalldatetime. The columns where data must be entered are marked NOT NULL, those where data does not have to be stored have NULL. Columns with an assigned primary key are marked as PRIMARY KEY, and a foreign key as FOREIGN KEY. We used the USE function to call the ITIS database. After running the code, the programme (SSMS) will write a message about the creation of tables. The programmed code (SQL) to create the tables in the database is shown below.

```
USE [ITIS]
GO

CREATE TABLE ITIS.dbo.comments (
    comment_id int NOT NULL CONSTRAINT comments_pk PRIMARY KEY CLUSTERED,
        commentator varchar (100) NULL,
        comment_detail char (2000) NOT NULL,
        comment_time_stamp datetime NOT NULL,
        update_date smalldatetime NOT NULL);
GO

CREATE TABLE ITIS.dbo.experts (
        expert_id_prefix char (3) NOT NULL,
        expert_id int NOT NULL,
        expert varchar (100) NULL,
        exp_comment varchar (500) NULL,
        update_date smalldatetime NULL,
    CONSTRAINT experts_pk PRIMARY KEY CLUSTERED (expert_id_prefix,expert_id));
GO

CREATE TABLE ITIS.dbo.geographic_div (
        tsn int NOT NULL,
        geographic_value varchar (45) NOT NULL,
        update_date smalldatetime NULL,
    CONSTRAINT geographic_div_pk PRIMARY KEY NONCLUSTERED (tsn,geographic_value));
GO

CREATE TABLE ITIS.dbo.jurisdiction (
        tsn int NOT NULL,
        jurisdiction_value varchar (30) NOT NULL,
        origin varchar (19) NULL,
        update_date smalldatetime NULL,
    CONSTRAINT jurisdiction_pk PRIMARY KEY NONCLUSTERED (tsn,jurisdiction_value));
GO

create table dbo.kingdoms (
    kingdom_id int NOT NULL CONSTRAINT kingdoms_pk PRIMARY KEY CLUSTERED,
        kingdom_name char (10) NOT NULL,
        update_date smalldatetime NOT NULL);
GO

create table dbo.longnames (
    tsn int NOT NULL CONSTRAINT longnames_pk PRIMARY KEY CLUSTERED,
        completename varchar (164) NULL);
GO

create table dbo.nodc_ids (
        nodc_id char (12) NULL,
        update_date smalldatetime NULL,
        tsn int NULL);
GO

create table dbo.other_sources (
        source_id_prefix char (3) NOT NULL,
        source_id int NOT NULL,
        source_type char (10) NULL,
        source varchar (64) NULL,
        version char (10) NULL,
        acquisition_date smalldatetime NULL,
        source_comment varchar (500) NULL,
        update_date smalldatetime NULL,
    CONSTRAINT other_sources_pk PRIMARY KEY CLUSTERED (source_id_prefix,source_id));
GO

create table dbo.publications (
        pub_id_prefix char (3) NOT NULL,
        publication_id int NOT NULL,
```

```
        reference_author varchar (100) NULL,
        title varchar (255) NULL,
        publication_name varchar (255) NULL,
        listed_pub_date datetime NULL,
        actual_pub_date datetime NULL,
        publisher varchar (80) NULL,
        pub_place varchar (40) NULL,
        isbn varchar (16) NULL,
        issn varchar (16) NULL,
        pages varchar (15) NULL,
        pub_comment varchar (500) NULL,
        update_date smalldatetime NULL,
    CONSTRAINT publications_pk PRIMARY KEY CLUSTERED
(pub_id_prefix,publication_id));
    GO

    create table dbo.strippedauthor (
        taxon_author_id int NOT NULL CONSTRAINT
PK_strippedauthor PRIMARY KEY NONCLUSTERED,
        shortauthor varchar (100) NOT NULL);
    GO

    create table dbo.synonym_links (
        tsn int NOT NULL,
        tsn_accepted int NOT NULL,
        update_date smalldatetime NOT NULL,
    CONSTRAINT synonym_links_pk PRIMARY KEY
CLUSTERED (tsn,tsn_accepted));
    GO

    create table dbo.taxon_authors_lkp (
        taxon_author_id int NOT NULL CONSTRAINT tax-
on_authors_lkp_pk PRIMARY KEY NONCLUSTERED,
        taxon_author varchar (100) NOT NULL,
        update_date smalldatetime NOT NULL,
        kingdom_id smallint NOT NULL,
      short_author varchar(100));
    GO

    create table dbo.taxon_unit_types (
        kingdom_id int NOT NULL,
        rank_id smallint NOT NULL,
        rank_name char (15) NOT NULL,
        dir_parent_rank_id smallint NOT NULL,
        req_parent_rank_id smallint NOT NULL,
        update_date smalldatetime NOT NULL,
    CONSTRAINT taxon_unit_types_pk PRIMARY KEY
CLUSTERED (kingdom_id,rank_id));
    GO

    create table dbo.taxonomic_units (
        tsn int NOT NULL CONSTRAINT taxonomic_units_pk
PRIMARY KEY CLUSTERED,
        unit_ind1 char (1) NULL,
        unit_name1 char (35) NOT NULL,
        unit_ind2 char (1) NULL,
        unit_name2 varchar (35) NULL,
        unit_ind3 varchar (7) NULL,
        unit_name3 varchar (35) NULL,
        unit_ind4 varchar (7) NULL,
        unit_name4 varchar (35) NULL,
        unnamed_taxon_ind char (1) NULL,
        usage varchar (12) NOT NULL,
        unaccept_reason varchar (50) NULL,
        credibility_rtng varchar (40) NOT NULL,
        completeness_rtng char (10) NULL,
        currency_rating char (7) NULL,
        phylo_sort_seq smallint NULL,
        initial_time_stamp datetime NOT NULL,
        parent_tsn int NULL,
        taxon_author_id int NULL,
        hybrid_author_id int NULL,
        kingdom_id smallint NOT NULL,
        rank_id smallint NOT NULL,
        update_date smalldatetime NOT NULL,
        uncertain_prnt_ind char (3) NULL,
        name_usage varchar(12) NOT NULL,
        complete_name varchar(300));
    GO

    create table dbo.tu_comments_links (
        tsn int NOT NULL,
        comment_id int NOT NULL,
        update_date smalldatetime NOT NULL,
    CONSTRAINT tu_comments_links_pk PRIMARY KEY
CLUSTERED (tsn,comment_id));
    GO

    create table dbo.reference_links (
    tsn int NOT NULL CONSTRAINT refer-
ence_links_fk_taxonomic_units FOREIGN KEY REFERENCES
dbo.taxonomic_units (tsn),
        doc_id_prefix char (3) NOT NULL,
        documentation_id int NOT NULL,
        original_desc_ind char (1) NULL,
        init_itis_desc_ind char (1) NULL,
        change_track_id int NULL,
        vernacular_name varchar (80) NULL,
        update_date smalldatetime NOT NULL,
    CONSTRAINT reference_links_pk PRIMARY KEY
CLUSTERED (tsn,doc_id_prefix,documentation_id));
    GO

    create table dbo.vernaculars (
    tsn int NOT NULL CONSTRAINT vernacu-
lars_fk_taxonomic_units FOREIGN KEY REFERENCES
dbo.taxonomic_units (tsn),
        vernacular_name varchar (80) NOT NULL,
        language varchar (15) NOT NULL,
        approved_ind char (1) NULL,
        update_date smalldatetime NOT NULL,
        vern_id int NOT NULL,
    CONSTRAINT vernaculars_pk PRIMARY KEY
NONCLUSTERED (tsn,vern_id));
    GO

    create table dbo.vern_ref_links (
        tsn int NOT NULL,
        doc_id_prefix char (3) NOT NULL,
        documentation_id int NOT NULL,
        update_date smalldatetime NOT NULL,
        vern_id int NOT NULL,
    CONSTRAINT vern_ref_links_pk PRIMARY KEY
CLUSTERED (tsn,doc_id_prefix,documentation_id,vern_id),
    CONSTRAINT vern_ref_links_fk_vernaculars FOREIGN KEY
(tsn,vern_id) REFERENCES dbo.vernaculars (tsn,vern_id));
    GO

    create table dbo.hierarchy (
    hierarchy_string varchar(300) NOT NULL CONSTRAINT hier-
archy_pk PRIMARY KEY CLUSTERED,
        TSN int NOT NULL,
        Parent_TSN int,
        level int NOT NULL,
        ChildrenCount int NOT NULL);
    GO
```

After creating the database and its tables, external data is imported. To call the ITIS database, we used the USE function. We enter the path where the data are stored as follows: “set @home_path=‘C:itis’”. After

running the code in SQL, the data will be loaded into the created tables. The programmed code (SQL) for importing data into the tables in the database is shown below.

```
USE ITIS;

declare @home_path as varchar(500);
set @home_path='C:\itis\';
declare @filename as varchar(50);
declare @sql as varchar(1000);
set @filename='comments';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='experts';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='geographic_div';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='jurisdiction';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='kingdoms';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='nodc_ids';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='other_sources';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='publications';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='taxon_authors_lkp';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='taxon_unit_types';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='tu_comments_links';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='synonym_links';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='taxonomic_units';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='reference_links';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='vernaculars';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='vern_ref_links';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='longnames';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='strippedauthor';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
print @sql;
exec (@sql);

set @filename='hierarchy';
set @sql='BULK INSERT dbo.' + @filename + ' from ''' +
@home_path + @filename + ''' with (CODEPAGE = 1252,
FIELDTERMINATOR = "|", rowterminator = "\n")'
```

print @sql;
exec (@sql);

The description of the data in the imported tables is as follows:

Table comments – stores comments on changes in systematic classification.

table experts – stores the names of systematic experts with their addresses.

Table geographic_div stores information on the continents of occurrence of the species.

table hierarchy – the table serves as a converter for assigning other data.

Table jurisdiction – stores information on the states of occurrence of the species.

table kingdoms – is a code book with the listed systematic groups.

Table longnames – stores Latin names of taxa.

table nodc_ids – the table serves as a converter for assigning other data.

Table Other_sources - stores Internet sources from which the database on systematic classification of taxa draws.

Table publications – stores publication sources from which the database on systematic classification of taxa is drawn.

Table reference_links – the table serves as a converter for assigning other data.

Table strippedauthor – serves as a codebook for the names of people who described individual taxa (species, class, family, etc.)

Table synonym_links – serves as a converter from old taxon names to new ones.

Table taxon_authors_lkp stores the names of people who described individual taxa (species, class, family, etc.).

Table taxon_unit_types – serves as a converter for assigning codes to systematic groups (Kingdom, Subkingdom, Phylum, Subphylum, etc.) assigned to the kingdom.

table taxonomic_units – serves as a frequency table with repeated entries.

Table tu_comments_links – serves as a converter for assigning other data.

table vern_ref_links – serves as a converter for assigning other data.

Table Vernaculars - serves as a converter for assigning Latin species names to vernacular names.

After implementing data from an external database, we want to perform data mining, which is different in SQL compared to specialised libraries in Python. However, it is a good tool for preliminary analysis, aggregation, and pattern search, which is the basis of data mining. From the frequency table taxonomic_units, we want to find the frequency of genera recorded in the table. That is, the frequency of occurrence of records. To obtain this information, we used the following query:

```
SELECT distinct([unit_name1]) as genus,
        count([unit_name1]) as number_taxonov
FROM [ITIS].[dbo].[taxonomic_units]
GROUP BY [unit_name1]
ORDER BY count([unit_name1]) desc
```

After running the selection, we get a result from which we can say that the most represented genus is the genus *Euphorbia*, which has 5,009 records. The total number of records without duplicates is 107,540 (Fig. 1).

Each genus has an associated species and comments on individual species aimed at their protection or changing the scientific name. To find the number of occurrence of these comments for individual taxonomic groups, we will use the following query:

```
SELECT A.[completename],
        count(B.[update_date]) as count
FROM [ITIS].[dbo].[longnames] as A
LEFT JOIN [ITIS].[dbo].[tu_comments_links] as B on A. [tsn]=B. [tsn]
LEFT JOIN [ITIS].[dbo].[comments] as C on B.[comment_id]=C.[comment_id]
WHERE B.comment_id is not null
GROUP BY A. [completename]
ORDER BY count(B.[update_date]) desc
```

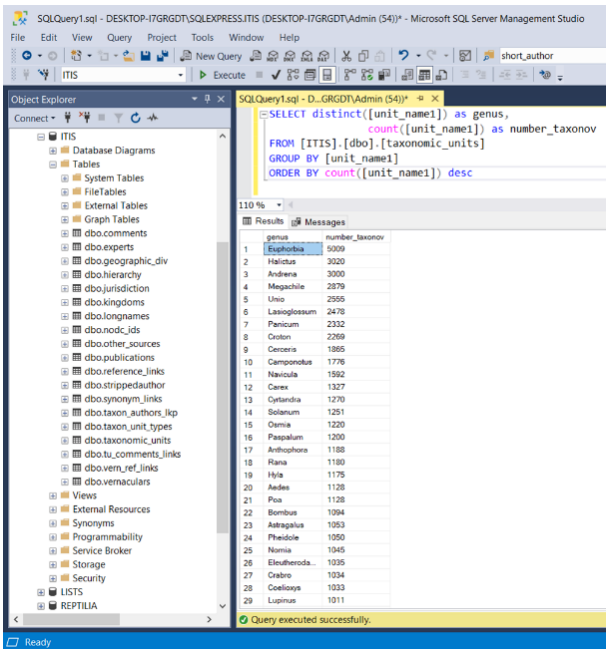

Fig. 1. The result of the abundance of genus

From the results, we found that the taxonomic groups with the highest number of comments were *Oncorhynchus tshawytscha* (16 comments), *Oncorhynchus mykiss* (14 comments), *Echinorhynchus*, *Articulata* (9 comments), *Moraxella caviae*, *Moraxella ovis*, *Tanais cavolinii*, *Anatanais normani*, *Pikea*, *Sphingobacterium composti*, *Idotea balthica*, *Lirceus culveri*, *Trichia*, *Chione* (8 comments) (Fig. 5). The remaining taxa had 7 or fewer comments, 18 taxa had 7 comments, 85 taxa had 6 comments, 220 taxa had 5 comments, 1752 taxa, had 4 comments, 13,775 taxa had 3 comments, 25,082 taxa had 2 comments, 91,844 taxa had 1 comment and 827,345 taxa had 0 comments (Fig. 2).

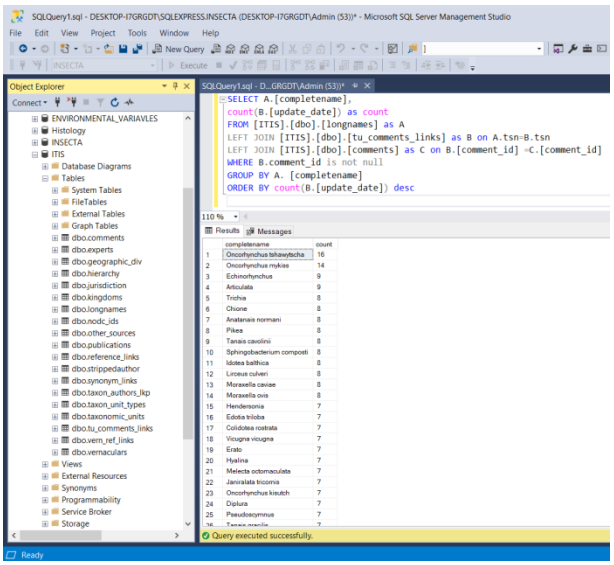

Fig. 2. Result of the number of comments

We also want to find out the frequency of comments so that we know in which years the most frequent changes in information about various states in relation to taxonomy occurred. We can obtain the necessary information by selecting the following:

```
SELECT frequency.[date] as year,
```

```
        count(frequency.[date]) as frequency
FROM
(SELECT A. [completename],
right(CONVERT(varchar(10),B.[update_date],104),4) as date,
                C. [commentator],
                C. [comment_detail]
FROM [ITIS].[dbo].[longnames] as A
LEFT JOIN [ITIS].[dbo].[tu_comments_links] as B on A. [tsn]=B. [tsn]
LEFT JOIN [ITIS].[dbo].[comments] as C on B. [comment_id] = C. [comment_id]) as frequency
GROUP BY frequency. [date]
ORDER BY frequency. [date]  desc
```

We confirmed that the highest frequency of comments was during the years 2015 (51,173 entries), 2001 (29,492 entries), and 2010 (21,517 entries), 2005 (13,091 entries). The lowest frequency of comments was during the years 1997 (2 entries), and, 1998 (180 entries) 2003 (212 entries) (Fig. 3).

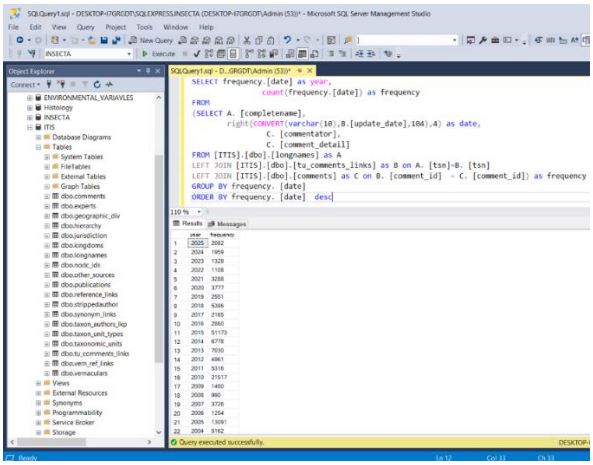

Fig. 3. Result of the frequency of comments

Changes in comments also come from published sources of articles, so these parts are related and the frequency of writing sources is interesting. To find out, we used the following select:

```
SELECTright(CONVERT(varchar(10),[update_date],104),4)as year,
        count(publication_name) as frequency
FROM [ITIS].[dbo].[publications]
GROUP BY right(CONVERT(varchar(10),[update_date],104),4)
ORDER BY count(publication_name) desc
```

We confirmed that the highest frequency of sources was during the years 2023 (2,910 sources), 2018 (2,588 sources), and 2021 (2,324 sources), 2024 (2,113 sources). The lowest frequency of sources was during the years 1998 (9 sources), 1997 (15 sources) 2001 (19 sources) (Fig. 4).

Discussion

Implementing data integration from external biological databases, such as ITIS, required a rigorous design of a relational model that could capture complex taxonomic relationships and metadata in a unified Microsoft SQL Server environment. Data from ITIS, managed by the Smithsonian Institution and linked to GBIF and the Catalogue of Life, were extracted via an official API and subsequently transformed into three main entities: taxa, synonyms, and comments on individual items. At the same time, this solution builds on approaches such as BioWarehouse, which show that storing heterogeneous biological databases in MySQL or Oracle enables efficient multi-database queries in SQL (Galperin, 2004; Lee et al., 2006).

The schema design focused on normalisation so that each taxon, its author, and comment had a unique identifier, and relationships between entities were clearly represented. This allowed the use of recursive Common Table Expressions (CTE) to construct a full taxonomic hierarchy, greatly simplifying the analysis of relationships between genera and species. Furthermore, the addition of spatial SQL extensions allows geographic queries to be performed directly in the database, opening up the potential for studies of spatiotemporal distribution patterns of biodiversity (Triplet & Butle, 2013; Tiwari et al., 2018; Siegel et al., 2022).

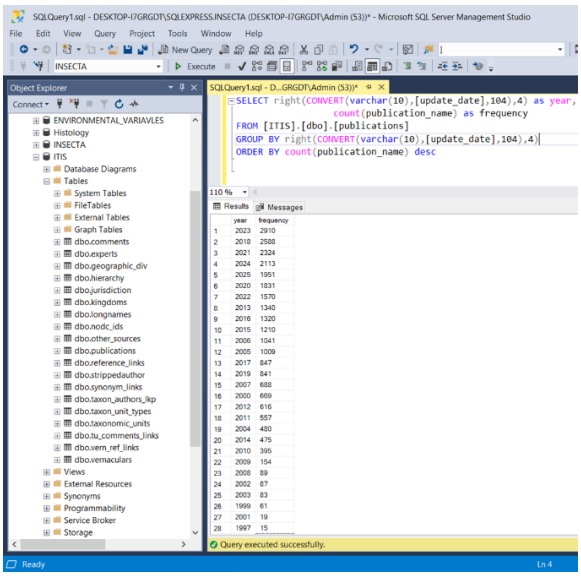

Fig. 4. Result of the frequency of literary sources

Preliminary data mining analysis revealed that 107,540 references to genera are found in the 19 imported source files, of which the genus *Euphorbia* with 5,009 records represents more than 4.5% of the total taxonomic component. This proportion corresponds to phylogenetic studies confirming the high diversity of the genus Euphorbia based on analysis of ITS and ndhF sequences (Steinmann & Porter 2002; Wang et al., 2022).

Analysis of qualitative feedback showed that the most comments were received from the important aquaculture species *Oncorhynchus tshawytscha* (Chinook salmon) and *Oncorhynchus mykiss*, with 16 and 14 comments, respectively. These figures reflect the need for a regular review of taxonomic status (De Lorenzo et al., 2018) and conservation strategies for these species, which are of crucial ecological importance; (Scott, C. 2003. "Oncorhynchus tshawytscha" (On-line), Animal Diversity Web. Accessed July 7, 2025 at https://animaldiversity.org/accounts/Oncorhynchus_tshawytscha)

The temporal structure of comment edits shows four significant peaks of activity: 2015 (51,173 changes), 2001 (29,492), 2010 (21,517), and 2005 (13,091), with the lowest frequency (1997–1998, 2003) correlating with the beginnings of the digital transformation and the limited capacities of database systems at that time. These trends show that large-scale taxonomic revisions often occur in cycles stimulated by technological innovation and grant funding (Federhen 2012; Schoch et al., 2020).

Likewise, the dynamics of the increase in references shows a sharp increase in the years 2018–2023, with a peak in 2023 (2,910 new sources), reflecting the improvement of ETL processes and the greater availability of tools for processing big data in bioinformatics (Birney & Clamp 2004; Hogeweg 2011; Yang et al., 2017).

Conclusions

The implementation of external data from the ITIS database was carried out in a Microsoft SQL Server environment in three steps: (1) creating a schema and tables with correctly defined data types, (2) bulk loading of 19 basic files with taxonomic and metadata data, and (3) verifying the quality and completeness of the imported data through SQL queries for statistical analysis. Subsequently, we performed data mining focused on the frequency of genera registration, the time evolution of comments, and publication sources. The results confirm that the ITIS database in SQL is a reliable basis for systematic studies of taxon-

omy and evolutionary trends. The preliminary analysis allowed us to identify key areas of expert activity (comments) and the dynamics of adding new publications. The process designed in this way and the first results demonstrate the strong role of the relational database model in data mining in the field of biological systematics.

This research was supported by the grants VEGA 1/0603/25 Data integration (Big data) for spatial modeling of biodiversity in different ecosystem conditions, KEGA No. 010UKF-4/2025 Data science for biology and No. 037SPU-4/2024 Data integrity in biological and ecological databases.

References

Afgan, E., Baker, D., Batut, B., van den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B. A., Guerler, A., Hillman-Jackson, J., Hiltemann, S., Jalili, V., Rasche, H., Soranzo, N., Goecks, J., Taylor, J., Nekrutenko, A., & Blankenberg, D. (2018). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. Nucleic Acids Research, 46(W1), W537–W544.

Akoka, J., Comyn-Wattiau, I., & Laoufi, N. (2017). Research on Big Data – a systematic mapping study. Computer Standards and Interfaces, 54, 105–115.

Barrett, T., Wilhite, S. E., Ledoux, P., Evangelista, C., Kim, I. F., Tomashevsky, M., Marshall, K. A., Phillippy, K. H., Sherman, P. M., Holko, M., Yefanov, A., Lee, H., Zhang, N., Robertson, C. L., Serova, N., Davis, S., & Soboleva, A. (2013). NCBI GEO: Archive for functional genomics data sets – update. Nucleic Acids Research, 41, D991–D995.

Baxevanis A. D. (2011). The importance of biological databases in biological discovery. Current Protocols in Bioinformatics, 34(1), 1–6.

Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Ostell, J., Pruitt, K. D., & Sayers, E. W. (2018). GenBank. Nucleic acids research, 46(D1), D41–D47.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., & Bourne, P. E. (2000). The protein data bank. Nucleic Acids Research, 28(1), 235–242.

Birney, E., & Clamp, M. (2004). Biological database design and implementation. Briefings in Bioinformatics, 5(1), 31–38.

Cochrane, G., Karsch-Mizrachi, I., & Nakamura, Y. (2010). The International Nucleotide Sequence Database Collaboration. Nucleic Acids Research, 39, D15–D18.

Coronel, C., & Morris, S. (2014). Database systems: Design, implementation and management. 11th ed. Cengage Learning, Stamford.

De Lorenzo, V., Prather, K. L., Chen, G., O'Day, E., von Kameke, C., Oyarzún, D. A., Hosta-Rigau, L., Alsafar, H., Cao, C., Ji, W., Okano, H., Roberts, R. J., Ronaghi, M., Yeung, K., Zhang, F., & Lee, S. Y. (2018). The power of synthetic biology for bioproduction, remediation and pollution control. EMBO Reports, 19(4), e45658.

Federhen, S. (2011). The NCBI taxonomy database. Nucleic Acids Research, 40(D1), D136–D143.

Galperin, M. Y. (2004). The molecular biology database collection: 2004 update. Nucleic Acids Research, 32, 3–22.

Gligorijević, V., & Pržulj, N. (2015). Methods for biological data integration: perspectives and challenges. Journal of the Royal Society, Interface, 12(112), 20150571.

Hogeweg, P. (2011). The roots of bioinformatics in theoretical biology. PLoS Computational Biology, 7(3), e1002021.

Kanehisa, M., Furumichi, M., Sato, Y., Ishiguro-Watanabe, M., & Tanabe, M. (2020). KEGG: integrating viruses and cellular organisms. Nucleic Acids Research, 49(D1), D545–D551.

Kasprzyk A. (2011). BioMart: Driving a paradigm change in biological data management. Database, 2011, bar049.

Lee, T. J., Pouliot, Y., Wagner, V., Gupta, P., Stringer-Calvert, D. W., Tenenbaum, J. D., & Karp, P. D. (2006). BioWarehouse: A bioinformatics database warehouse toolkit. BMC Bioinformatics, 7, 170.

Leinonen, R., Sugawara, H., & Shumway, M. (2010). The sequence read archive. Nucleic Acids Research, 39, D19–D21.

Luscombe, N. M., Greenbaum, D., & Gerstein, M. (2001). What is bioinformatics? An introduction and overview. Yearbook of Medical Informatics, 10(1), 83–100.

Schoch, C. L., Ciufo, S., Domrachev, M., Hotton, C. L., Kannan, S., Khovanskaya, R., Leipe, D., Mcveigh, R., O'Neill, K., Robbertse, B., Sharma, S., Soussov, V., Sullivan, J. P., Sun, L., Turner, S., & Karsch-Mizrachi, I. (2020). NCBI Taxonomy: A comprehensive update on curation, resources and tools. Database, 2020, baaa062.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L., & Lewis, S. (2007). The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology, 25(11), 1251–1255.

Steinmann, V. W., & Porter, J. M. (2002). Phylogenetic relationships in Euphorbieae (Euphorbiaceae) based on ITS and ndhF sequence data. Annals of the Missouri Botanical Garden, 89(4), 453–490.

Tiwari, S., Wee, H. M., & Daryanto, Y. (2018). Big data analytics in supply chain management between 2010 and 2016: Insights to industries. Computers and Industrial Engineering, 115, 319–330.

Triplet, T., & Butler, G. (2013). A review of genomic data warehousing systems. Briefings in Bioinformatics, 15(4), 471–483.

Wang, Y.-L., Jian, X., & Wang, S. (2022). Characterization of the complete chloroplast genome of *Euphorbia pekinensis* Rupr. (Euphorbiaceae). Mitochondrial DNA Part B, 7(8), 1550–1552.

Yang, A., Troup, M., & Ho, J. W. K. (2017). Scalability and validation of big data bioinformatics software. Computational and Structural Biotechnology Journal, 15, 379–386.