DEVELOPMENT OF AN URBAN MATERIAL FLOW AND STOCK DATABASE STRUCTURE

Deliverable 4.2

Metabolism of Cities

Version	1.0 (2020-04-01)
WP	4
Dissemination level	Public
Deliverable lead	Metabolism of Cities (MoC), info@metabolismofcities.org
Authors	Paul Hoekman (MoC)
Reviewers	Carolin Bellstedt (MoC), Aristide Athanassiadis (MoC)
Abstract	This document provides the background of an urban metabolism database in use by Metabolism of Cities and describes the requirements that exist for the CityLoops project. This makes it clear why modifying the existing database structure is a useful development and necessary to take for the CityLoops project. More technical descriptions of the database schema are provided and the core components of the modified database structure are discussed, including the scope and extent of the modifications made for CityLoops.
Keywords	Urban metabolism; Stocks and flows; Database structure; Structured Query Language (SQL)
License	This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). See: https://creativecommons.org/licenses/by/4.0/

This document includes technical specifications and domain-specific language. A solid understanding of database administration and Structured Query Language (SQL) is required to understand the technical part of this document.

Contents

Acronyms and Abbreviations

AS-MFA	Activity-based Spatial MFA
CTE	Common Table Expression
DBMS	Database management system
GUMDB	Global Urban Metabolism Database
IOA	Input-Output Analysis
LCA	Life Cycle Assessment
MFA	Material Flow Analysis
MTU	Micro-territorial Unit
OMAT	Online Material Flow Analysis Tool
SCA	Sector-wide Circularity Assessment
SQL	Structured Query Language
STAFDB	Stocks and Flows Database
UCA	Urban Circularity Assessment
UMIS	Unified Materials Information System
WP	Work Package
YSTAFDB	Yale Stocks and Flows Database

Table of Figures

Table of Tables

1. Introduction

CityLoops is an EU Horizon 2020-funded project that brings together seven ambitious European cities to demonstrate a series of innovative tools and urban planning approaches, aimed at closing the loops of urban material flows and increasing their regenerative capacity. This report is part of Work Package (WP) 4: Urban Circularity Assessment. This WP has two objectives:

- To develop and implement a sector-wide material flow and stock accounting method, designed to help optimise demonstration activities through a detailed analysis of material flows, stakeholder involvement and valorisation pathways.
- To develop and demonstrate a comprehensive city-wide urban circularity assessment procedure, designed to enable cities to effectively integrate circularity into planning and decision making.

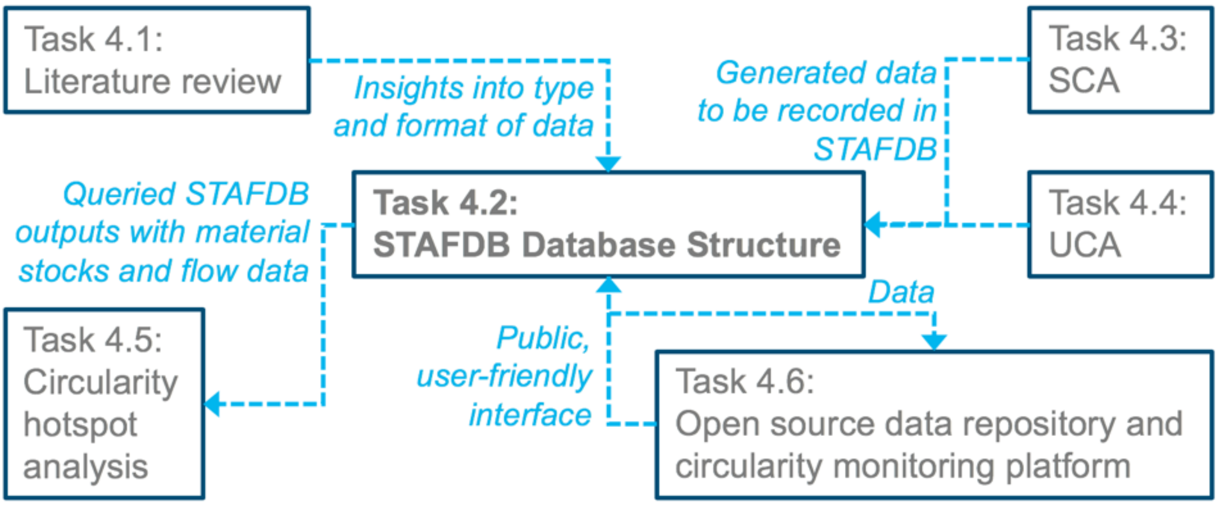


Figure 1: Relationship of tasks in WP4 and the information and/or function that they provide

This report is a deliverable of Task 4.2: Development of a flow and stock database structure. Within the associated WP4, there is no final decision yet on the material accounting method that will be used to undertake (city-wide and sector-wide) circularity assessments (Task 4.3 and 4.4). However, the development of a database structure is intentionally taking place before this decision is finalised, because the nature of the data collected and used in the following steps is already known. After completing a literature review on the different urban material flow and stock accounting methods (see Deliverable 4.1 from Task 4.1), there are already sufficient insights into the different types of methods that exist and that are relevant for CityLoops. All these methods use data on resource stocks and flows, and the database structure will need to cater to this by remaining as flexible and comprehensive as possible. Building a system around this database structure that allows for uploading, retrieving, and processing data in a way that is stipulated by the chosen method will take place separately from this task (in Task 4.6), and at that point it will be necessary to have a full understanding of the chosen method. However, until then, recording of data in a consistent format is key, which is the aim of this task. Finally, Figure 1 illustrates the relationship of the various tasks within WP4, showing what they provide to each other.

This document will outline the background of an urban metabolism database in use by Metabolism of Cities (Section 2.1) and then describe the requirements that exist for the CityLoops project (Section 2.2). This makes it clear why modifying the existing database structure is a necessary and useful route to take for the CityLoops project. More technical descriptions follow in Section 2.3, where the core components of the modified database structure are discussed. Section 2.4 describes in more detail the scope and extent of the modifications made for CityLoops. Finally, Chapter 3 summarises the future work to be done and how this database structure is expected to evolve during and beyond the CityLoops project.

The main output of Task 4.2, however, is not this report itself but instead Annex 1, which contains an SQL dump with the complete database structure. This report merely describes the rationale behind this schema and provides insights into the changes that were made. Annex 2 contains a sample spreadsheet format that shows in what format cities may collect and report data. This illustrates that the data collection and reporting format is a rather simple one, which will be transformed “behind the scenes” into the more complex data structure that is in place.

2. Database Structure

In this chapter, the database structure - including its background, recent changes, and future plans - is reviewed.

2.1. Background

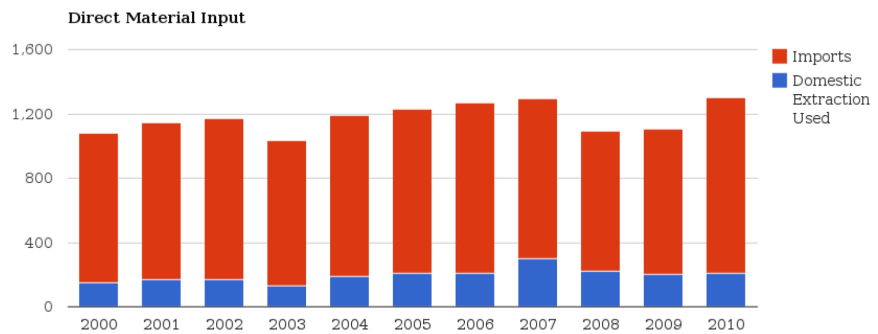
For many years, the open source, urban metabolism web platform developed by Metabolism of Cities (available at https://www.metabolismofcities.org) has been storing and sharing urban metabolism data in order to better understand the metabolism of urban systems. Over the years, the way data has been uploaded and used has changed due to a number of iterations to cater different purposes and users. In this section, the rationale behind these iteration steps are further detailed.

The first step, initiated in 2014, was an online tool to administer a material flow analysis (MFA), called OMAT (Online Material Flow Analysis Tool), which allowed users to record and manage material flow data for their own project(s). The system generates tables, indicators, and charts based on the entered data, see Figure 2. OMAT can be used for an economy-wide MFA or it can be used to perform an MFA on a specific sector. It has been used to allow students to jointly contribute data into the same MFA project (Villalba and Hoekman 2018), and it was one of the first web-based, open source tools to manage MFA datasets. Amongst similar tools that existed at that period was the offline software STAN which was an inspiration for OMAT.

Data

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Domestic Extraction Used	150	169	170	133	190	209	211	299	221	201	206
Imports	930	977	1,004	904	1,001	1,019	1,060	998	870	903	1,093
Direct Material Input	1,080	1,147	1,174	1,037	1,191	1,228	1,271	1,297	1,091	1,104	1,299

Graphs



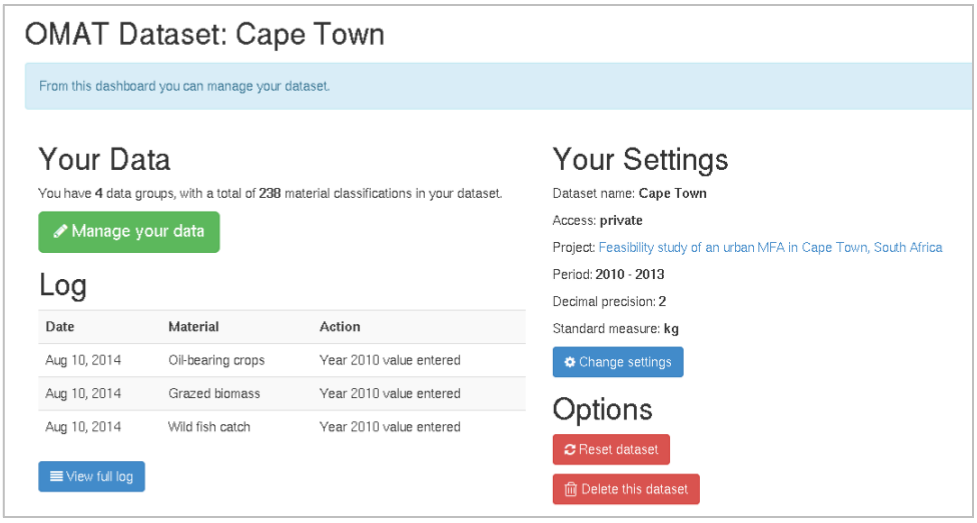
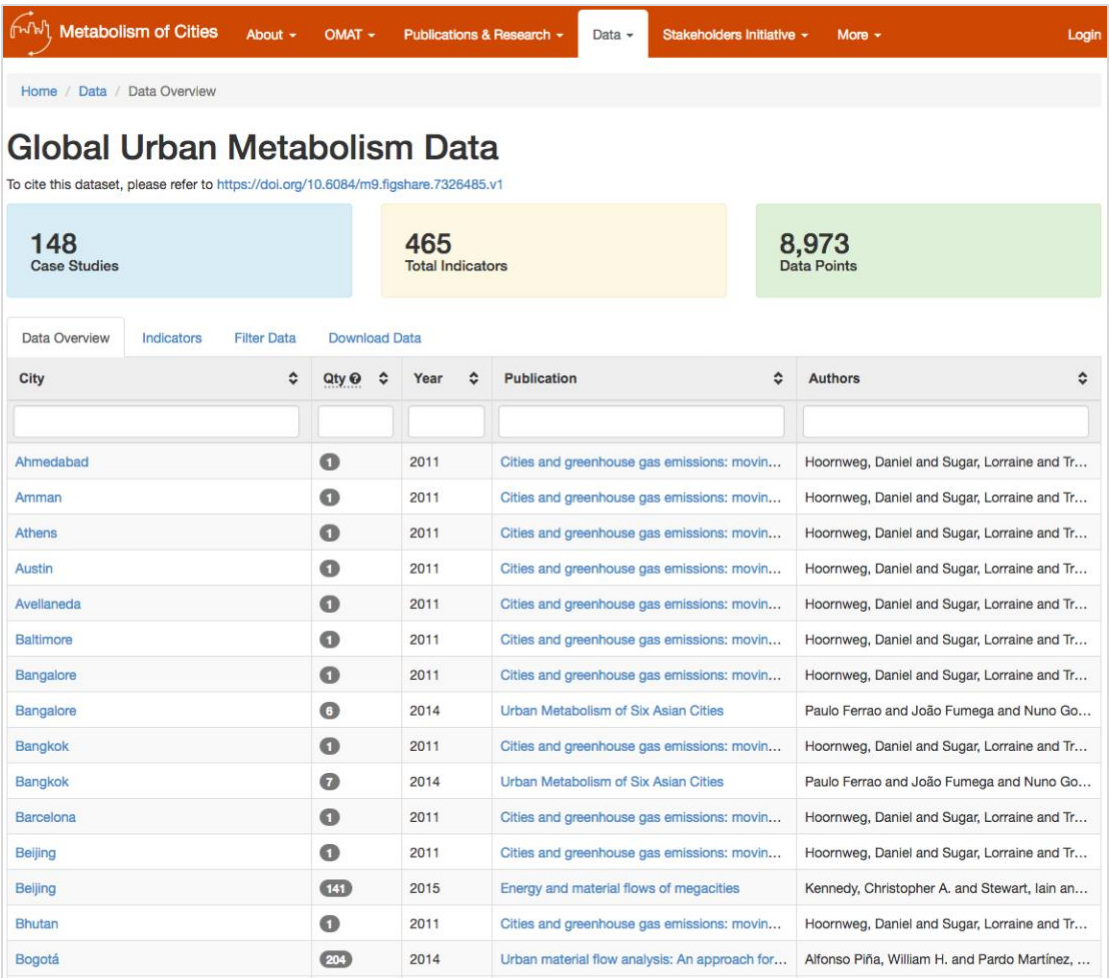
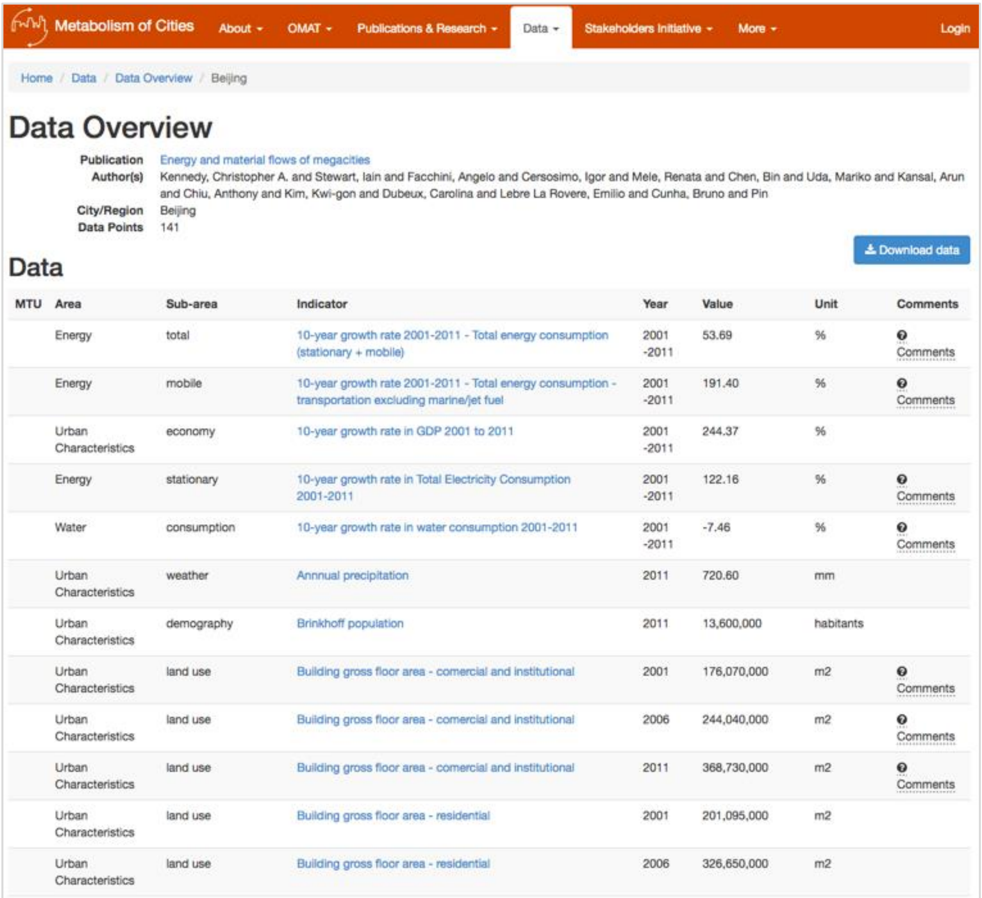


Figure 2: Screenshots of OMAT, displaying the initial dashboard (bottom), and graphs and tables generated by the system (top).

In 2017, the Global Urban Metabolism Database (GUMDB) was set up as an initial experiment to centralise data points and indicators obtained from/by academic work (Figure 3) (Hoekman et al. 2019). Both GUMDB and OMAT have export functions that enable users to download data (either the entire project or a specific part of it) in CSV format.




Metabolism of Cities · About · OMAT · Publications & Research · Data · Stakeholders Initiative · More · Login

Home / Data / Data Overview / Beijing

Data Overview

Publication: Energy and material flows of megacities
Author(s): Kennedy, Christopher A. and Stewart, Iain and Facchini, Angelo and Cersosimo, Igor and Mele, Renata and Chen, Bin and Uda, Mariko and Kansal, Arun and Chiu, Anthony and Kim, Kwi-gon and Dubeux, Carolina and Lebre La Rovere, Emilio and Cunha, Bruno and Pin
City/Region: Beijing
Data Points: 141

Download data

Data

MTU	Area	Sub-area	Indicator	Year	Value	Unit	Comments
	Energy	total	10-year growth rate 2001-2011 - Total energy consumption (stationary + mobile)	2001 -2011	53.69	%	Comments
	Energy	mobile	10-year growth rate 2001-2011 - Total energy consumption - transportation excluding marine/jet fuel	2001 -2011	191.40	%	Comments
	Urban Characteristics	economy	10-year growth rate in GDP 2001 to 2011	2001 -2011	244.37	%	
	Energy	stationary	10-year growth rate in Total Electricity Consumption 2001-2011	2001 -2011	122.16	%	Comments
	Water	consumption	10-year growth rate in water consumption 2001-2011	2001 -2011	-7.46	%	Comments
	Urban Characteristics	weather	Annnual precipitation	2011	720.60	mm	
	Urban Characteristics	demography	Brinkhoff population	2011	13,600,000	habitants	
	Urban Characteristics	land use	Building gross floor area - comercial and institutional	2001	176,070,000	m2	Comments
	Urban Characteristics	land use	Building gross floor area - comercial and institutional	2006	244,040,000	m2	Comments
	Urban Characteristics	land use	Building gross floor area - comercial and institutional	2011	368,730,000	m2	Comments
	Urban Characteristics	land use	Building gross floor area - residential	2001	201,095,000	m2	
	Urban Characteristics	land use	Building gross floor area - residential	2006	326,650,000	m2	

Figure 3: Screenshots of GUMDB, showing a summary of available data (top), and a detailed list with data available for a specific city (bottom).

After running these two projects for a number of years, Metabolism of Cities started working on a system to capture material stocks and flow data from a greater variety of sources and with a larger degree of heterogeneity. This project, dubbed MultipliCity, was set up to allow for a much more fine-grained level of data capturing. MultipliCity makes it fairly easy for users to upload data, and it is built around the idea of crowdsourcing the collection of urban stocks and flow data. Data could be

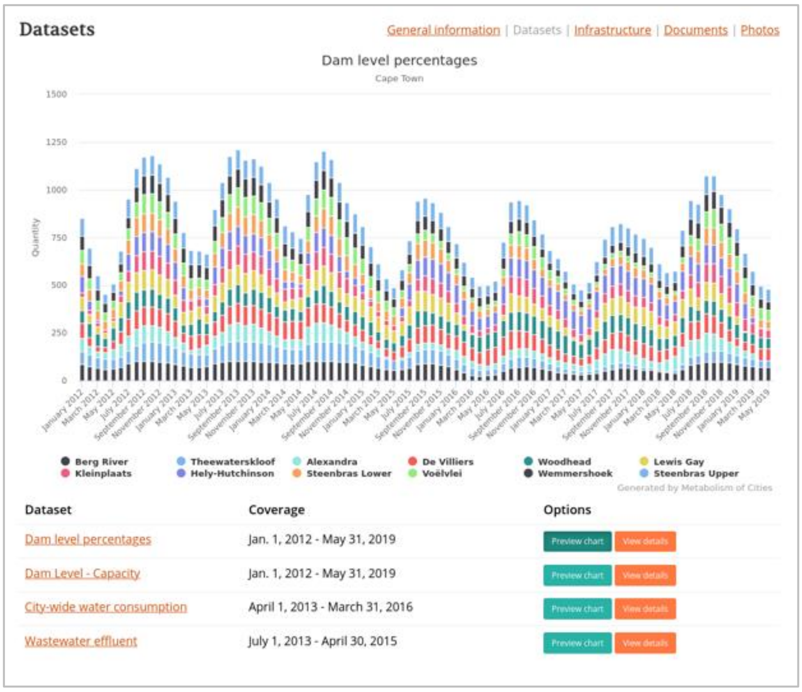


Figure 4: Screenshot of MultipliCity with overview of multiple datasets available within a sector (source)

recorded on a city-wide scale, but it could also be recorded on a suburb or neighbourhood level. Data could even be linked to individual infrastructure (e.g. a train station or wastewater treatment plant). Uploaded datasets are stored in a single database and data can be aggregated or disaggregated according to user needs. Figure 4 and Figure 5 depict screenshots of MultipliCity.

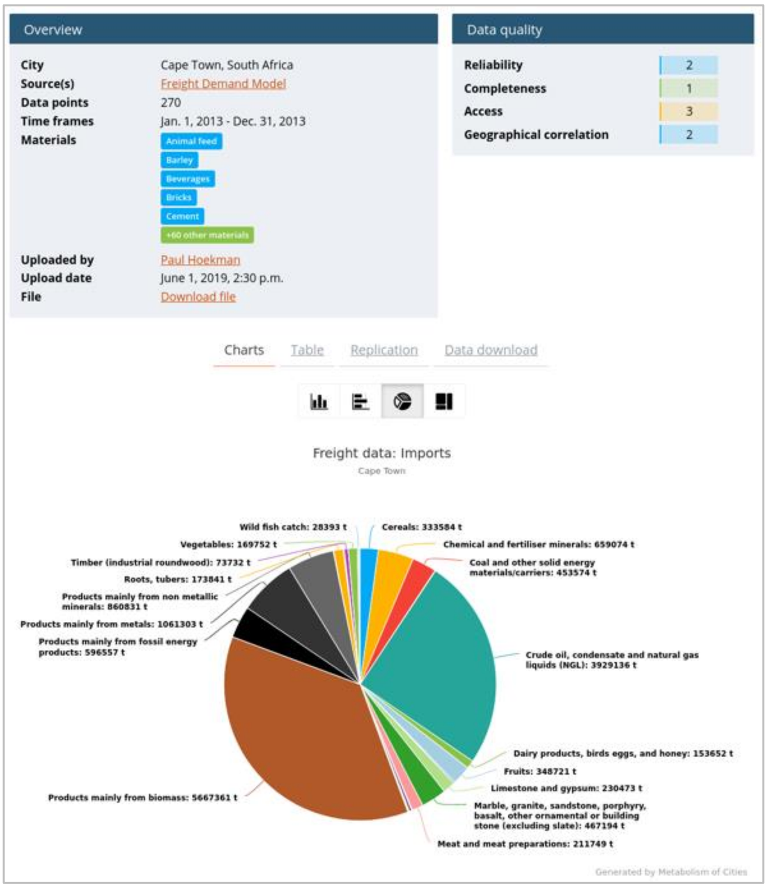


Figure 5: Screenshots of MultipliCity, featuring a record of a single dataset and its visualisations (source)

Both OMAT and GUMDB use two different MySQL database schemas, both of which are specifically made for their associated application. However, MultipliCity was set up with a more widespread use in mind. This system was built on the Unified Materials Information System (UMIS). UMIS was developed at Yale University (Myers et al. 2019), and was put to use in a database subsequently created to store material flows data obtained from decades of material systems research at Yale. This database, called the Yale Stocks and Flows Database (YSTAFDB), was one of the first functional databases where theoretical frameworks (like UMIS) are applied to a real-life scenario. This also meant that a database schema had to be developed alongside the theoretical framework. Both, the YSTAFDB database schema and the data points are published as open source works (Myers, Reck, and Graedel 2019).

Other material stocks and flows research groups have also developed databases or worked on consolidating the often incompatible formats. The industrial ecology data commons project (Pauliuk et al. 2019) provides a prototype database structure that aims to integrate other databases developed within a variety of disciplines. Other interesting work includes a database containing data on material intensity for buildings (Heeren and Fishman 2019), and a general system structure for socioeconomic metabolism information (Pauliuk, Majeau-Bettez, and Müller 2015).

YSTAFDB provided the most suitable starting point for the MultipliCity system. This system was one of the most applied database structures (rather than being a more theoretical framework), and the goals were well-aligned with Metabolism of Cities' data storage goals. However, from a technical perspective this database structure lacked some features. A principal shortcoming was the lack of database normalisation which may result in data redundancy and lack of data integrity (this means that when data is stored in multiple places, a change effected in one place may lead to a discrepancy if the same data point is not changed in another place, which becomes more likely if the database is not normalised). The initial implementation of an adjusted YSTAFDB in MultipliCity, called the Stocks and Flows Database (STAFDB), primarily consisted of applying database normalisation practices to the existing structure. It was in this form that it was implemented within the Metabolism of Cities website.

2.2. Requirements for data storage in CityLoops

Within CityLoops, there will be a multitude of material stocks and flows data that needs to be stored efficiently, in a central location, and that can be retrieved easily to suit a variety of needs (e.g. to generate data visualisations, to export to a spreadsheet, or to be used as an input into a model). For this, a suitable database structure needed to be developed that would be flexible enough to function for different methods and have some other key features, which will be elaborated on in this chapter.

When developing a database structure, it was taken into account that a number of different accounting method families have been identified in the literature review (Deliverable 4.1). There, the following method families were identified:

- Flow analysis methods
- Energy assessment methods
- Input/output (IO) methods
- Footprint methods
- Life cycle assessment (LCA) methods
- Integrated methods

These different types of methods have different data storage requirements. For all methods, there are quantities and material flows involved. Flow analysis methods focus on an origin and destination for materials. Energy assessments look at the upstream energy needs. Input/output methods unpack the interplay between components (e.g. sectors) within a system. LCA methods are concerned with the entire lifecycle of a material flow. Integrated methods are combinations of the aforementioned methods and generally do not have their own unique data storage requirements, following the ones of the combined methods.

From the outset of the CityLoops project, it has been envisioned that there would be a certain level of integration of material stocks and flows data within the open source data hub already developed by Metabolism of Cities. The existing database structure and the MultipliCity data visualisation system already met a number of the requirements of the CityLoops project. In order to fully cater to the CityLoops needs, a number of adjustments were made to this database structure. Various of these changes relate to the requirement of allowing data from different accounting methods to be stored in the same database, as discussed above. Other changes were focused on improving general shortcomings that were already observed in the initial roll-out of the system.

The following key features should be part of the database structure to ensure it fully suits the needs of the CityLoops project:

- **Multi-method**:
  Ability to record material stocks and flows data originating from different types of methodologies (MFA, LCA, IOA, and others where possible) - this is discussed in more detail below.
- **Multi-scale**:
  At a minimum, the system should work for city-wide data and sector-wide data. Since it will have to be flexible for these two levels, it should be possible to record national, global, or sub-national (regional) data as well.
- **Aggregation and disaggregation**:
  Users should be able to aggregate up (e.g. get the total material flows within a city based on the flows observed in all suburbs inside the city), and disaggregate (e.g. view individual data points for all those suburbs when exploring the city's data). Data should be recorded on the most fine-grained level (up to a sensible level), and then displayed as per the user's preference.
- **Hierarchical understanding of processes, sectors, materials, and reference spaces**:
  All these catalogues contain a certain hierarchical structure. This hierarchical structure should be maintained at a database level, to allow for aggregation and disaggregation using these same features. This requirement is more elaborated on in the following chapter.
- **Scalability**:
  Many millions of data points should be stored within the system without it adversely affecting performance (e.g. slow it down).
- **Balancing necessary defaults with customisation options**:
  To ensure users, in principle, use the same classification system for materials, processes, or reference spaces, there should be default catalogues in place that guide this - but at the same time users must retain a way to customise certain parts of their profile to suit their specific needs.
- **Community adoption**:
  The structure should be set up in such a way that a wide community of users can benefit from this system. The more users, the higher the likelihood that this project can continue to be developed and expanded. Community adoption is increased by a) licensing this

under an open source, permissive license, b) allowing a variety of methods, spatial scopes, and material scopes to be catered to, and c) encouraging of the development of tools and visualisations that are separate from CityLoops' and Metabolism of Cities' specific needs.

2.3. Principal CityLoops changes to STAFDB

The following sections explain the principal changes and the reason behind the core features of this database structure. These changes were crucial to enable STAFDB to be used for the CityLoops project. Some limitations affected the ability of the database to store crucial information or to provide flexibility to cater to the different data storage needs expected from the different cities, while other shortcomings would have had a negative effect in the long-term expansion and maintenance of the database.

2.3.1. Database normalisation and structural clean-up

While the initial implementation of STAFDB already included a database normalisation effort, these efforts were not yet finalised. Within STAFDB, the goal is to implement normalisation in order to reduce data redundancy and improve data integrity. However, this process may come at a cost in terms of structural complexity and performance. Achieving the sixth normal form (6NF), which in principle is the highest level of database normalisation, is not necessarily the goal. Instead, STAFDB aims to achieve the highest practical level of normalisation, weighed off against database complexity and performance. These trade-offs can be subjective and some of them will have to be studied and discussed as time goes by.

Another key activity was to critically review the database structure and evaluate the suitability of every single database field. A number of fields were implemented ad-hoc, especially in the first few months of bringing the MultipliCity system online, as unexpected user or admin needs arose. Several fields were furthermore made redundant but never removed. A structural evaluation and clean-up took place.

Some patterns that were observed in the previous roll-out of STAFDB related to database normalisation and clean-up include:

- Process (origin and destination) recording in the Data table. Every single flow within a specific time frame for a specific material and from and to a specific reference space will share the same process origin and destination if they describe the same part of the overall system flow diagram. In the original setup that meant that there was a high level of duplication of data. To overcome this, a new table was set up to record the processes involved in each vertex of the flow diagram. This new structure is elaborated on in more detail in the *Process diagrams (section 2.3.3)* section.
- Dataset, CSV, and Data interaction was improved. These three tables contain meta information about the dataset, details of the set of data points uploaded at any one time, and the actual individual data points, respectively. However, this structure was not

flexible enough to accommodate multiple people feeding data into the dataset, or to allow for people to edit or remove individual data points. These tables were restructured to accommodate this.

- Logging was improved to keep better track of which user uploads, edits, or deletes information in any part of the system. Where possible and practical, the original information is retained to keep a persistent record, and soft deletes are introduced.
- The “Topics” were completely removed. This system was superseded by better integration of economic sectors and the use of ubiquitous tags throughout the system.
- The “DatasetType” model was entirely removed. It became redundant after rolling out the new process diagrams.
- MTU (Micro-territorial unit) information was duplicated in the old structure (it had its own table, separate from the reference spaces). This system was superseded by an hierarchical geocode table.
- Reference spaces included separate fields for “Country”, “City”, and “Parent”, which was restructured to take out the parental link at reference space level and instead define this within the geocode system, and to then link to this.

2.3.2. Improved implementation of Adjacency Lists

Hierarchical data structures are commonplace in a stocks and flows database. Some examples of hierarchical data include:

- Processes (e.g. Mining is part of Extraction which is part of Pre-use transformative processes)
- Materials (e.g. Bananas is part of Fruits which is part of Crops which is part of Biomass)
- Reference spaces (e.g. Apeldoorn which is part of The Netherlands which is part of Europe).

There are a number of ways of storing hierarchical data in a database. Each technique comes with advantages and drawbacks, and it is a matter of weighing these pros and cons and selecting the technique most suitable for the use case. Some of the properties that should be evaluated when comparing options include:

- Whether standard SQL can be used or proprietary extensions are required
- Efficiency of finding descendants
- Efficiency of finding ancestors
- Ease of finding the children of a node
- Ease of finding a node’s parents
- Efficiency of aggregate queries
- Ease of tree reorganization

The topic of selecting an appropriate technique when recording hierarchical data has been of interest to programmers for many years. A 9 year old question on Stack Overflow[1] on the topic has attracted over 236 thousand views and 1157 stars to date. The following comparison table comes from the book SQL Design Patterns (Vadim Tropashko 2014) and compares four different techniques.

This comparison is not exhaustive and many other techniques could also be considered. For the STAFDB structure, Adjacency Lists were selected as the way to integrate hierarchical data. As can be seen in Table 1, where this is listed as "Adjacency relation", this technique ranks high on nearly all of the features that were analysed. The caveat is that finding ancestors and descendants is expensive under standard SQL. However, Common Table Expressions (CTE) alleviate this problem. CTEs are not supported in all database management systems, but because PostgreSQL does support this and PostgreSQL are considered the DBMS of choice, it was decided to implement adjacency lists.

Table 1: Comparison between Adjacency relation, Nested Sets, Materialized Path, and Nested Intervals (Vadim Tropashko 2014)

ADJACENCY RELATION (TREE EDGES; STANDALONE, OR COMBINED WITH THE TREE NODES)	NESTED SETS	MATERIALIZED PATH	NESTED INTERVALS VIA MATRIX ENCODING
Have to use proprietory SQL extensions for finding ancestors and descendants; although the queries are efficient	Standard SQL	Standard SQL	Standard SQL
Finding descendants is relatively efficient (i.e. proportional to the size of the subtree)	Finding descendants is easy and relatively efficient (i.e. proportional to the size of the subtree)	Finding descendants is easy and relatively efficient (i.e. proportional to the size of the subtree)	Same as MP: Finding descendants is easy and relatively efficient
Finding ancestors is efficient	Finding ancestors is easy but inefficient	Finding ancestors is tricky but efficient	Same as MP: Finding ancestors is tricky but efficient
Finding node's children is trivial	Finding node's children as all the descendants restricted to the next level is inefficient	Finding node's children as descendants on next level is inefficient	Same as AR: Finding node's children is trivial

[1] See: https://stackoverflow.com/q/4048151

	(e.g. consider root node)		
Finding node's parent is trivial	Finding node's parent as ancestor on the previous level is inefficient due to inefficiency of ancestors search	Finding node's parent as ancestor on the previous level is efficient	Same as AR: Finding node's parent is trivial
Aggregate queries are relatively efficient (i.e. proportional to the size of the subtree)	Aggregate queries are relatively efficient (except counting, which is super fast)!	Aggregate queries are relatively efficient (i.e. proportional to the size of the subtree)	Aggregate queries are relatively efficient (i.e. proportional to the size of the subtree)
tree reorganization is very simple	tree reorganization is hard	tree reorganization is easy	tree reorganization is easy (but not as simple as in AR)

Adjacency lists were already in use in processes and materials, and this implementation is illustrated in the table overviews in Figure 6 and Figure 7 respectively below. The `parent_id` field contains a foreign key to another record within the same table. Furthermore, the usage of CTE was relatively limited in the initial implementation, and usage has now been rolled out in all the tables where adjacency lists have been used.

Table: processes

Column	Type	Example
id	integer Auto Increment	204
name	character varying(255)	Mining
code	character varying(255) NULL	E04
description	text NULL	All activities related to mining operations
parent_id	integer NULL	442 (another record in the same table which is for the Extraction process)

Figure 6: Database table overview for processes

Table: materials

Column	Type	Example
id	integer Auto Increment	656
name	text	Fruit
code	character varying(255) NULL	MF1.1.8
description	text NULL	Any type of fruit as per the Eurostat classification
catalog_id	integer NULL	6 (this refers to the Eurostat classification)
parent_id	integer NULL	325 (this is the record Crops in this same table)

Figure 7: Database table overview for materials

The table with information about reference spaces had to be restructured. This table contains information of any physical system (a country, city, suburb, facility, etc). However, the hierarchy of these systems is ambiguous. Take the example of a city. This could be seen as being part of a province or state, but it could also be part of a larger subnational region. This could then fit within a country, or there could be a classification based on other boundaries. Table 2 illustrates this with an example.

Table 2: Two different hierarchical trees to place Apeldoorn

Level 1	Europe	Western Europe
Level 2	The Netherlands	The Netherlands
Level 3	Apeldoorn	East Netherlands
Level 4		Gelderland
Level 5		Apeldoorn

As can be seen in Table 2, there are multiple ways to situate Apeldoorn within a hierarchical structure. Many different standards and systems exist to locate spaces (for example FIPS, NUTS, and ISO 3166). Rather than dictating a single standard, STAFDB is set up so that it can accommodate any hierarchical geocoding scheme. This is done by creating a catalogue of geocoding systems, and then creating adjacency lists for all of the levels that exist within that catalogue. Finally, reference spaces are linked through a many-to-many relationship with specific levels within one or multiple geocoding systems. The relevant tables for the reference spaces are listed in Figure 8 - Figure 11.

Table: geocodesystem

Column	Type	Example
id	integer Auto Increment	5
name	character varying(255)	ISO 3166-2
description	text NULL	ISO 3166-2 is part of the ISO 3166 standard published by the International Organization for Standardization (ISO), and defines codes for identifying the principal subdivisions (e.g., provinces or states) of all countries coded in ISO 3166-1.
url	character varying(200) NULL	https://www.iso.org/standard/63546.html

Figure 8: Database table overview for geocode system for reference spaces

Table: geocode

Column	Type	Example	
id	integer Auto Increment	104	105
name	character varying(255)	Country	Subdivision
description	text NULL		
parent_id	integer NULL	NULL	104
system_id	integer	5	5

Figure 9: Database table overview for geocode for reference spaces

Table: referencespace

Column	Type	Example
id	integer Auto Increment	554
name	character varying(255)	Porto
description	text NULL	NULL
url	character varying(255) NULL	NULL
slug	character varying(255) NULL	porto
active	boolean	true
location_id	integer NULL	58393
parent_id	integer NULL	NULL

Figure 10: Database table overview for referencespace for reference spaces

Table: referencespace_geocode

Column	Type	Example
id	integer	14
referencespace_id	integer	554
geocode_id	integer	105

Figure 11: Database table contains the many-to-many relationship between geocode entries and reference spaces

2.3.3. Process diagrams

Within MultipliCity, the focus has been on storing MFA data, either on a city-wide level or on a micro-territorial unit level. Data recorded included an origin and destination place, and could be linked to a process, based on the NACE list. However, it was not possible to build a system overview. For instance, data could be stored on flows leaving and entering the city, but the system could not calculate the net addition to stock - all flows were seen as independent blocks without any correlation between them. The system was not set up to capture LCA or IOA datasets.

Structural changes were made to enable for flows to be correlated. This is done by allowing the user to build a system diagram as a first step. This diagram can be envisioned as a flow diagram, in which any number of blocks are linked to each other, and flows are drawn between each of these blocks. Such a process diagram is common in LCA and in SFA, where the entire value chain or life cycle is drawn out, and the size of each flow is calculated. Other methods also use system diagrams to link flows to specific activities. An example is Activity-based Spatial MFA (AS-MFA), developed within the REPAiR project (Geldermans et al. 2017). Figure 12 and Figure 13 below illustrate process diagrams for material flows from existing literature.

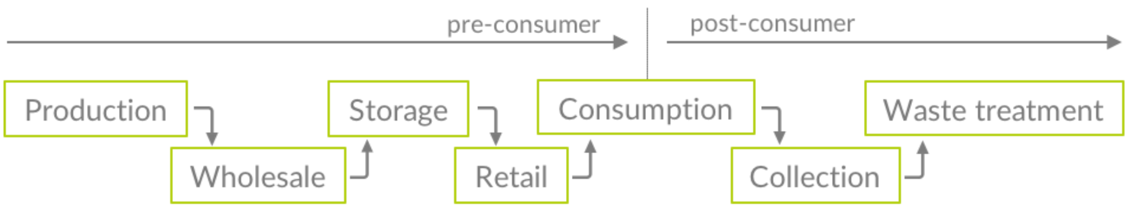


Figure 12: Typical supply chain visualised within the REPAiR project (Geldermans et al. 2017)

Process flow diagrams were already part of UMIS. However, because these diagrams were not relevant to MultipliCity in its first MFA-based version, they were excluded from STAFDB. In UMIS, the diagrams are developed by creating so-called subsystem specifications which are then linked to each other. A subsystem defines a particular process-based activity within the anthroposphere or within the natural environment. Within such a subsystem, the user defines a transformative process (e.g. *Quarrying*), a material flow (e.g. *200t of iron ore*), and a distributive process (e.g. *Transporting the ore to smelters*). This subsystem could also have a storage process, but this is not required. Once the subsystem is defined, it is given a specific code based on the activities and the position within the bigger system (e.g. *PEM.1;1;4*). Within

UMIS data points are linked to a specific step in the process by including the subsystem code within the data table.

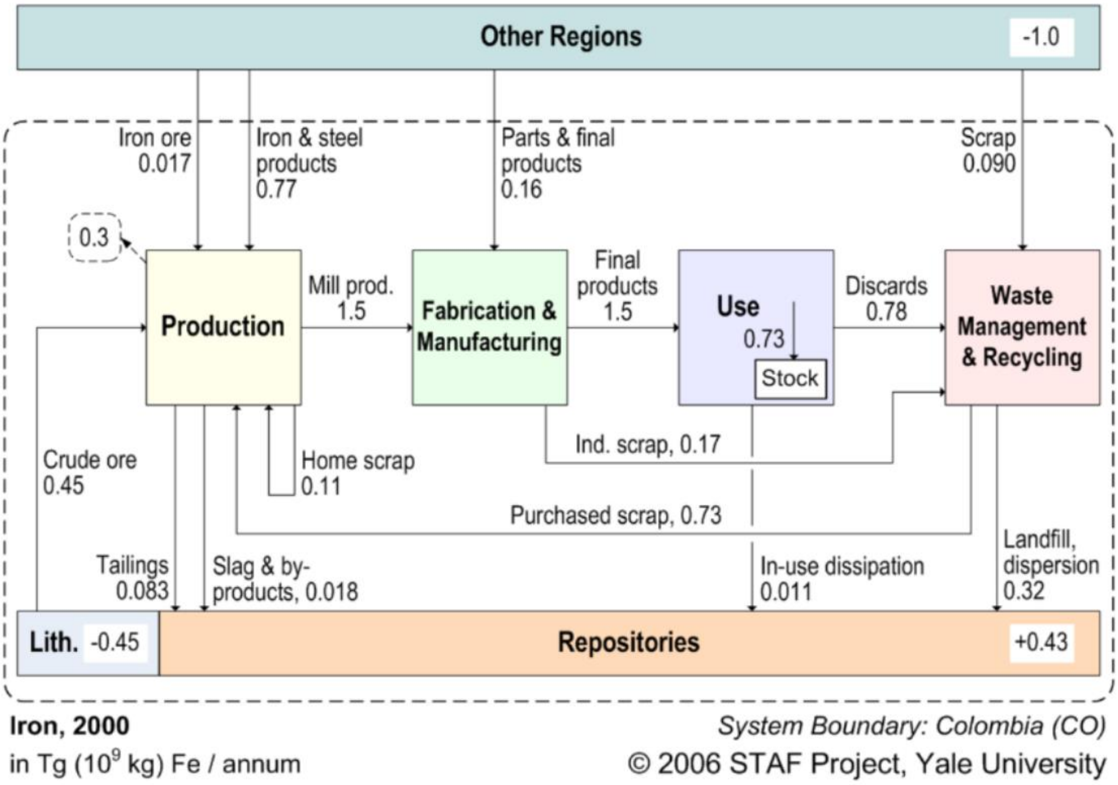


Figure 13: Material cycle diagram from a research project on the global iron cycle (Wang, Müller, and Graedel 2007).

In order to embed a more scalable and normalised implementation of the flow diagram system, STAFDB was equipped with a number of new tables. Firstly, a table was created to store metadata about a particular flow diagram (see Figure 14). The idea behind this is that a particular flow diagram (e.g. one that describes the water sector in a city) may be used by multiple datasets. In fact, it is likely beneficial to the system if users are encouraged to re-use existing flow diagrams in order to enhance comparability and standardise flow diagram-based data visualisations. The table contains a limited number of fields, as shown in Figure 14.

Table: flowdiagram

Column	Type	Example
id	integer Auto Increment	32
name	character varying(255)	Water cycle
description	text NULL	This describes the lifecycle of potable water, from extraction to wastewater.

Figure 14: Database table containing metadata about a flow diagram

A separate table (Figure 15) contains the individual elements within the flow diagram. Each edge or vertex of this diagram is recorded independently, and the process table is referenced to indicate both the origin and the destination (the material flow will move from origin to destination).

Table: flowblocks

Column	**Type**	**Example**
id	integer Auto Increment	140
description	text NULL	This is the flow of water going from the water treatment plants to the final user.
origin_id	integer	22 (links to process 22 = Water treatment)
destination_id	integer	493 (links to process 493 = Use)
diagram_id	integer	32 (links to the flowdiagram table)

Figure 15: Database table containing the individual blocks of the flow diagram

2.3.4. Independent structure

Lastly, the decision was made to develop STAFDB as a standalone app within the larger Metabolism of Cities project. Within Django (the python framework used to build the Metabolism of Cities website), apps are independent collections of files that contain models, views, and static files which may be moved between different projects. In the previous structure of the Metabolism of Cities platform, the stocks and flows database, the MultipliCity data visualisation platform, and the other parts of the website were set up as highly correlated fragments that could not function independently.

In the new structure, the STAFDB system is seen as a database structure that is intimately integrated with the front- and back-end tools that are used to insert, edit, extract, and visualise data. This makes it possible for this system to be used elsewhere. Integration of this system in the Metabolism of Cities website is just one of multiple possible uses of this system. This is expected to enhance uptake of this system, which ultimately enhances the longevity of this tool and makes it more likely that improvements and updates continue to be made, to the benefit of all that use this tool.

2.4. Structural overview

This section reflects on the core structure of STAFDB, after the CityLoops changes were applied. Understanding these key components will help understand the general database layout, and make it easier to unpack the SQL data dump that is provided in Annex 1. While there are many more tables and the data structure is more complex than this, the core of the STAFDB is formed by only a handful of tables. The focus for this part is on presenting this simplified core structure as a foundation for understanding the full data structure.

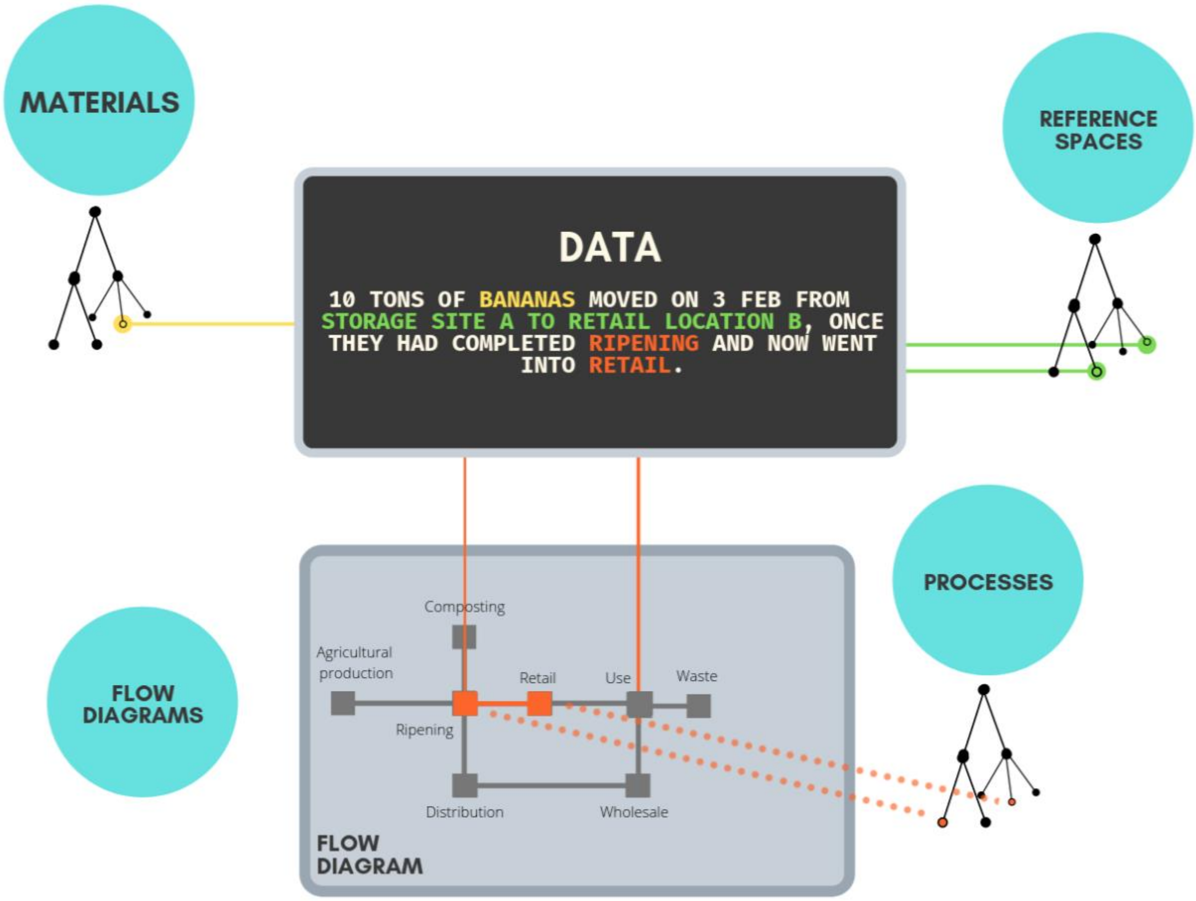


Figure 16: A simplified overview of the key tables within STAFDB, and how they relate to the data table

The principal tables within the database structure are the following:

- **Materials**
  This is a hierarchical materials list that contains all possible products and materials that may be tracked. The lists are based on existing standard classifications like the Harmonised System, CPA and others used by different statistical and international organisations.
- **Processes**
  These are economic or natural activities, structured in a hierarchical format. Various activity catalogues can be used (for instance, the statistical classification of economic activities in the European Community, abbreviated as NACE[2]).

2 https://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:Statistical_classification_of_economic_activities_in_the_European_Community_%28NACE%29

- **Reference spaces**
  Any type of system under study. A reference space refers to a physical place and could range from a household or company premises to a continent or the entire planet.
- **Flow Diagrams**
  A chain of connected processes that describe the life cycle or value chain of a product, sector, or the process-based structure of the system under study. A flow diagram describes how materials move through the system.
- **Data**
  This table contains the actual data points (quantifying the flows), and it links to the aforementioned tables. Each data point describes a material, which has a physical origin and destination (reference spaces), and this flow is linked to a specific flow within the larger system diagram (and thus also indirectly linked to specific processes).

These various tables are illustrated in Figure 16. The figure shows what a simplified setup looks like and demonstrates the key tables and how they link to the data table. Figure 17 illustrates the schematic overview of the tables in the database and how they are linked to each other.

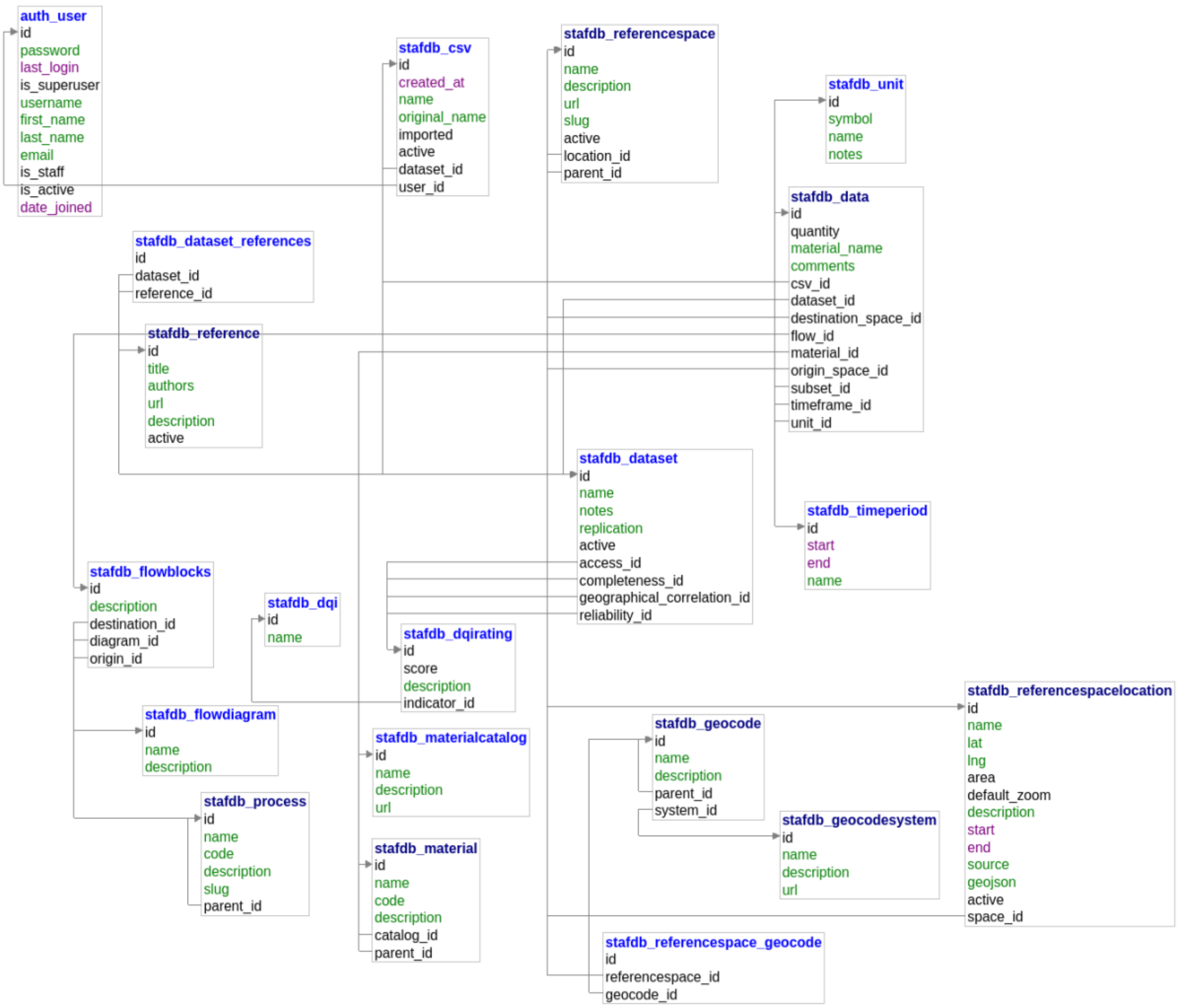


Figure 17: Visual view of the different tables in the STAFDB schema. Annex 1 provides the underlying SQL code for all the fields and tables.

In order to illustrate the complexity of information that is handled by the different tables, Figure 18 serves to show the material flow system diagram from an actual research project, and within it highlighted areas that are linked to specific tables. The colour framed boxes and text highlight where different components are stored in STAFDB.

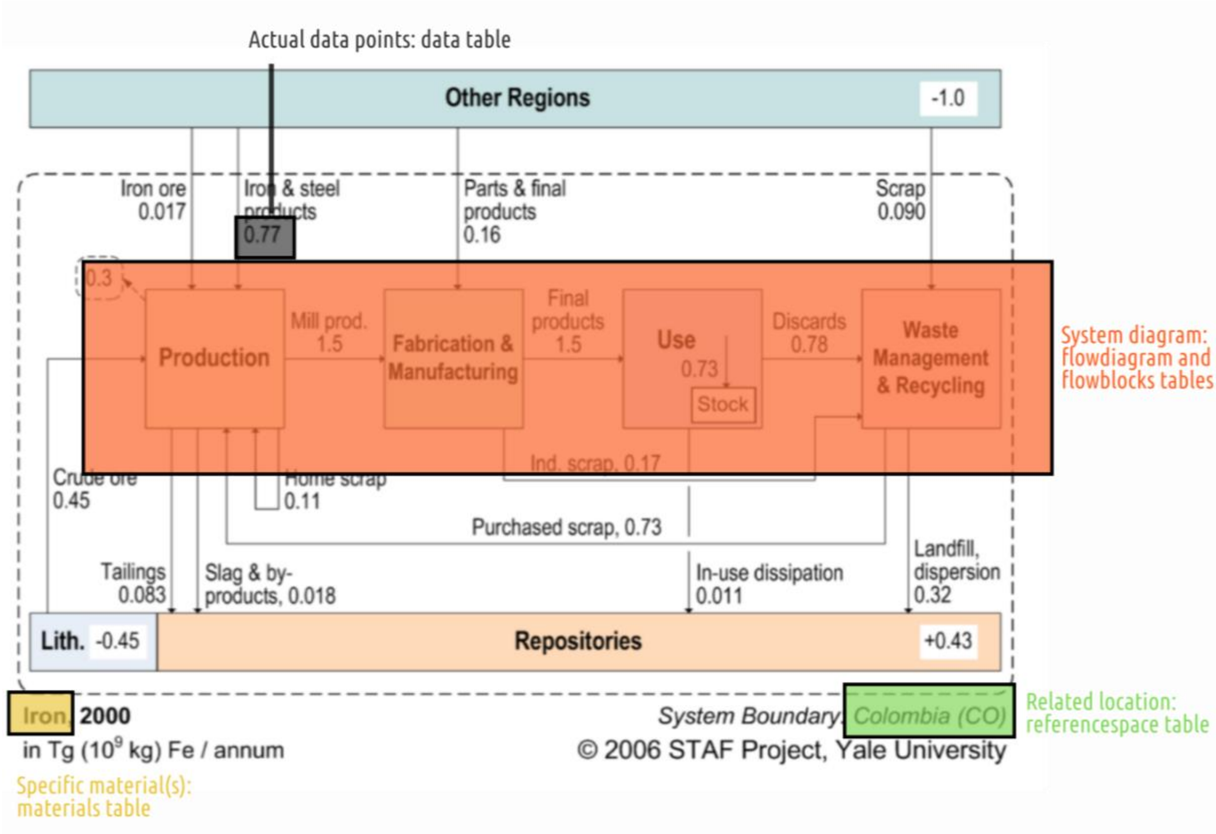


Figure 18: Overview of the key tables used for different components of a dataset (corresponding to the same colours as in Figure 16), illustrated with a material cycle diagram from a research project on the global iron cycle (Wang, Müller, and Graedel 2007).

3. Going forward

It is unlikely that this new database structure represents a perfect, final version. Just like the initial roll-out of STAFDB, this should be considered a work-in-progress that will continue to be improved as it is being used. However, it is already based on a real-life implementation and has been exposed to a diverse set of user needs. The implemented changes will make this system more robust and provide flexibility where it has shown to be needed. Certain changes and improvements are already expected and where possible catered for. The expectation is that the new structure is solid enough to not require drastic changes that heavily affect front- and back-end parts of the website when they are applied.

The following components are either already expected to be added later, or are so new that they will be tested and are likely subject to refinement throughout the CityLoops project (and possibly beyond this timeframe as well):

- Implementation of transfer coefficients. This is outside of the scope of the CityLoops project, but it will be a welcome addition to enable usage of the system for managing data gaps. We expect that transfer coefficients can be implemented in the future as an add-on without requiring structural upheaval.
- Implementation of reference materials. Sometimes a system is analysed with respect to a specific material, or material flow data is obtained in which the flow needs to be associated with a reference material (for example, when recording the concentration of one material inside another). This reference material can likely be easily implemented within either the meta data of the dataset, or at the level of the process diagram blocks.
- Inclusion of data from IOA and LCA. As mentioned before, these methods were not part of the first phase and the recent restructuring should now cater to data obtained using these methods. However, only after seeing it in practice and having third parties use the system can we tell if these changes are sufficient. A group of academic trial users will assist in this process.
- Use of the CTEs in the tables with adjacency lists. This system was only tentatively embedded in the first phase, and this wider roll-out will have to be tested. Performance and ease-of-use of the CTEs will have to be reviewed as the actual system is being developed.
- Material balancing and identification of gaps or information clashes. Through the use of the process diagrams, it should be possible to identify where data gaps exist, or where conflicting information is present (e.g. two data points providing different figures within the same diagram). The system should be able to pick up on this and present these discrepancies to the end user. This system needs to be tried.
- Nesting of reference spaces using the geocode systems. This is a new setup which also has to be tested in a real-life context. It is expected that ISO 3166 will be as the default geocode catalogue, but the suitability of this catalogue also has to be tested.

- Integration of the NACE codes and the STAFDB-specific process structure. The NACE codes have provided a useful and consistent hierarchical list of economic activities. However, the grouping of these activities is not compatible with the grouping required for STAFDB to efficiently use the process diagram layer. These two lists have to be merged in a way that the top-level classification is based on STAFDB requirements, while the lower-level grouping consists of existing NACE structures and thus can maintain a structure that is in effect equal to NACE data. This merge is not too complicated to carry out, as it only entails the moving of economic activity (NACE) codes into a limited number of top-level categories.
- Conversion of existing data and attempt to scale up. There is already data present in the existing system which will have to be converted to this new format. Furthermore, the database has always been developed with scaling in mind. Whether or not millions and millions of data points, reference spaces, or other related entries can be efficiently managed within this structure is to be tested.

Insights into the STAFDB schema and the structure itself will continuously be shared and released within the open source repository of the Metabolism of Cities website. Once the structure has been sufficiently tested additional documentation will be published and disseminated.

Bibliography

Geldermans, Bob, Carolin Bellstedt, Enrico Formato, Viktor Varju, Zoltan Grunhut, Maria Cerreta, Libera Amenta, Pasquale Inglese, Janneke van der Leer, and Alexander Wandl. 2017. 'Resource Management in Peri-Urban Areas (REPAiR): D3.1 Introduction to Methodology for Integrated Spatial, Material Flow and Social Analyses'. http://h2020repair.eu/wp-content/uploads/2018/03/Deliverable_3.1_Introduction_to_methodology.pdf.

Heeren, Niko, and Tomer Fishman. 2019. 'A Database Seed for a Community-Driven Material Intensity Research Platform'. *Scientific Data* 6 (1): 23. https://doi.org/10.1038/s41597-019-0021-x.

Hoekman, Paul, Aristide Athanassiadis, João Meirelles de Miranda, Franziska Meinherz, Gabriela Fernandez, and Yves Bettignies Cari. 2019. 'The Global Urban Metabolism Database'. https://doi.org/10.6084/m9.figshare.7326485.v1.

Myers, Rupert J., Tomer Fishman, Barbara K. Reck, and T. E. Graedel. 2019. 'Unified Materials Information System (UMIS): An Integrated Material Stocks and Flows Data Structure'. *Journal of Industrial Ecology* 23 (1): 222–40. https://doi.org/10.1111/jiec.12730.

Myers, Rupert J., Barbara K. Reck, and T. E. Graedel. 2019. 'YSTAFDB, a Unified Database of Material Stocks and Flows for Sustainability Science'. *Scientific Data* 6 (1): 1–13. https://doi.org/10.1038/s41597-019-0085-7.

Pauliuk, Stefan, Niko Heeren, Mohammad Mahadi Hasan, and Daniel B. Müller. 2019. 'A General Data Model for Socioeconomic Metabolism and Its Implementation in an Industrial Ecology Data Commons Prototype'. *Journal of Industrial Ecology* 23 (5): 1016–27. https://doi.org/10.1111/jiec.12890.

Pauliuk, Stefan, Guillaume Majeau-Bettez, and Daniel B. Müller. 2015. 'A General System Structure and Accounting Framework for Socioeconomic Metabolism'. *Journal of Industrial Ecology* 19 (5): 728–41. https://doi.org/10.1111/jiec.12306.

Vadim Tropashko. 2014. *SQL Design Patterns Book*. Rampant TechPress. http://www.rampant-books.com/book_0601_sql_coding_styles.htm.

Villalba, Gara, and Paul Hoekman. 2018. 'Using Web-Based Technology to Bring Hands-On Urban Material Flow Analysis to the Classroom'. *Journal of Industrial Ecology* 22 (2): 434–42. https://doi.org/10.1111/jiec.12553.

Wang, Tao, Daniel B. Müller, and T. E. Graedel. 2007. 'Forging the Anthropogenic Iron Cycle'. *Environmental Science & Technology* 41 (14): 5120–29. https://doi.org/10.1021/es062761t.

Annex 1

SQL dump with the complete database structure

```
--
-- PostgreSQL database dump
--

-- Dumped from database version 11.3 (Debian 11.3-1.pgdg90+1)
-- Dumped by pg_dump version 11.3 (Debian 11.3-1.pgdg90+1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- Name: auth_user; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.auth_user (
    id integer NOT NULL,
    password character varying(128) NOT NULL,
    last_login timestamp with time zone,
    is_superuser boolean NOT NULL,
    username character varying(150) NOT NULL,
    first_name character varying(30) NOT NULL,
    last_name character varying(150) NOT NULL,
    email character varying(254) NOT NULL,
    is_staff boolean NOT NULL,
    is_active boolean NOT NULL,
    date_joined timestamp with time zone NOT NULL
);


ALTER TABLE public.auth_user OWNER TO postgres;

--
-- Name: auth_user_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.auth_user_id_seq
    AS integer
```

```
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.auth_user_id_seq OWNER TO postgres;

--
-- Name: auth_user_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.auth_user_id_seq OWNED BY public.auth_user.id;


--
-- Name: stafdb_csv; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_csv (
    id integer NOT NULL,
    created_at timestamp with time zone NOT NULL,
    name character varying(255) NOT NULL,
    original_name character varying(255) NOT NULL,
    imported boolean NOT NULL,
    active boolean NOT NULL,
    dataset_id integer,
    user_id integer NOT NULL
);


ALTER TABLE public.stafdb_csv OWNER TO postgres;

--
-- Name: stafdb_csv_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_csv_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_csv_id_seq OWNER TO postgres;

--
-- Name: stafdb_csv_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_csv_id_seq OWNED BY public.stafdb_csv.id;
```

```
--
-- Name: stafdb_data; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_data (
    id integer NOT NULL,
    quantity double precision,
    material_name character varying(500),
    comments text,
    csv_id integer,
    dataset_id integer NOT NULL,
    destination_space_id integer,
    flow_id integer NOT NULL,
    material_id integer,
    origin_space_id integer,
    subset_id integer,
    timeframe_id integer NOT NULL,
    unit_id integer
);


ALTER TABLE public.stafdb_data OWNER TO postgres;

--
-- Name: stafdb_data_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_data_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_data_id_seq OWNER TO postgres;

--
-- Name: stafdb_data_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_data_id_seq OWNED BY public.stafdb_data.id;


--
-- Name: stafdb_dataset; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_dataset (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
    notes text,
    replication text,
    active boolean NOT NULL,
    access_id integer,
```

```
    completeness_id integer,
    geographical_correlation_id integer,
    reliability_id integer
);


ALTER TABLE public.stafdb_dataset OWNER TO postgres;

--
-- Name: stafdb_dataset_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_dataset_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_dataset_id_seq OWNER TO postgres;

--
-- Name: stafdb_dataset_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
postgres
--

ALTER SEQUENCE public.stafdb_dataset_id_seq OWNED BY public.stafdb_dataset.id;


--
-- Name: stafdb_dataset_references; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_dataset_references (
    id integer NOT NULL,
    dataset_id integer NOT NULL,
    reference_id integer NOT NULL
);


ALTER TABLE public.stafdb_dataset_references OWNER TO postgres;

--
-- Name: stafdb_dataset_references_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.stafdb_dataset_references_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER TABLE public.stafdb_dataset_references_id_seq OWNER TO postgres;

--
-- Name: stafdb_dataset_references_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.stafdb_dataset_references_id_seq OWNED BY
public.stafdb_dataset_references.id;


--
-- Name: stafdb_dqi; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_dqi (
    id integer NOT NULL,
    name character varying(40) NOT NULL
);


ALTER TABLE public.stafdb_dqi OWNER TO postgres;

--
-- Name: stafdb_dqi_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_dqi_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_dqi_id_seq OWNER TO postgres;

--
-- Name: stafdb_dqi_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_dqi_id_seq OWNED BY public.stafdb_dqi.id;


--
-- Name: stafdb_dqirating; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_dqirating (
    id integer NOT NULL,
    score smallint NOT NULL,
    description character varying(255) NOT NULL,
    indicator_id integer,
```

```
    CONSTRAINT stafdb_dqirating_score_check CHECK ((score >= 0))
);


ALTER TABLE public.stafdb_dqirating OWNER TO postgres;

--
-- Name: stafdb_dqirating_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_dqirating_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_dqirating_id_seq OWNER TO postgres;

--
-- Name: stafdb_dqirating_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
postgres
--

ALTER SEQUENCE public.stafdb_dqirating_id_seq OWNED BY public.stafdb_dqirating.id;


--
-- Name: stafdb_flowblocks; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_flowblocks (
    id integer NOT NULL,
    description text,
    destination_id integer NOT NULL,
    diagram_id integer NOT NULL,
    origin_id integer NOT NULL
);


ALTER TABLE public.stafdb_flowblocks OWNER TO postgres;

--
-- Name: stafdb_flowblocks_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_flowblocks_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER TABLE public.stafdb_flowblocks_id_seq OWNER TO postgres;

--
-- Name: stafdb_flowblocks_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_flowblocks_id_seq OWNED BY public.stafdb_flowblocks.id;


--
-- Name: stafdb_flowdiagram; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_flowdiagram (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
    description text
);


ALTER TABLE public.stafdb_flowdiagram OWNER TO postgres;

--
-- Name: stafdb_flowdiagram_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_flowdiagram_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_flowdiagram_id_seq OWNER TO postgres;

--
-- Name: stafdb_flowdiagram_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_flowdiagram_id_seq OWNED BY public.stafdb_flowdiagram.id;


--
-- Name: stafdb_geocode; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_geocode (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
```

```
    description text,
    parent_id integer,
    system_id integer NOT NULL
);


ALTER TABLE public.stafdb_geocode OWNER TO postgres;

--
-- Name: stafdb_geocode_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_geocode_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_geocode_id_seq OWNER TO postgres;

--
-- Name: stafdb_geocode_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
postgres
--

ALTER SEQUENCE public.stafdb_geocode_id_seq OWNED BY public.stafdb_geocode.id;


--
-- Name: stafdb_geocodesystem; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_geocodesystem (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
    description text,
    url character varying(200)
);


ALTER TABLE public.stafdb_geocodesystem OWNER TO postgres;

--
-- Name: stafdb_geocodesystem_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.stafdb_geocodesystem_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
```

```
    CACHE 1;


ALTER TABLE public.stafdb_geocodesystem_id_seq OWNER TO postgres;

--
-- Name: stafdb_geocodesystem_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_geocodesystem_id_seq OWNED BY public.stafdb_geocodesystem.id;


--
-- Name: stafdb_material; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_material (
    id integer NOT NULL,
    name text NOT NULL,
    code character varying(255),
    description text,
    catalog_id integer,
    parent_id integer
);


ALTER TABLE public.stafdb_material OWNER TO postgres;

--
-- Name: stafdb_material_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_material_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_material_id_seq OWNER TO postgres;

--
-- Name: stafdb_material_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_material_id_seq OWNED BY public.stafdb_material.id;


--
-- Name: stafdb_materialcatalog; Type: TABLE; Schema: public; Owner: postgres
--
```

```
CREATE TABLE public.stafdb_materialcatalog (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
    description text,
    url character varying(255)
);


ALTER TABLE public.stafdb_materialcatalog OWNER TO postgres;

--
-- Name: stafdb_materialcatalog_id_seq; Type: SEQUENCE; Schema: public; Owner: 
postgres
--

CREATE SEQUENCE public.stafdb_materialcatalog_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_materialcatalog_id_seq OWNER TO postgres;

--
-- Name: stafdb_materialcatalog_id_seq; Type: SEQUENCE OWNED BY; Schema: public; 
Owner: postgres
--

ALTER SEQUENCE public.stafdb_materialcatalog_id_seq OWNED BY 
public.stafdb_materialcatalog.id;


--
-- Name: stafdb_process; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_process (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
    code character varying(255),
    description text,
    slug character varying(255),
    parent_id integer
);


ALTER TABLE public.stafdb_process OWNER TO postgres;

--
-- Name: stafdb_process_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--
```

```
CREATE SEQUENCE public.stafdb_process_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_process_id_seq OWNER TO postgres;

--
-- Name: stafdb_process_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
postgres
--

ALTER SEQUENCE public.stafdb_process_id_seq OWNED BY public.stafdb_process.id;


--
-- Name: stafdb_reference; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_reference (
    id integer NOT NULL,
    title character varying(255) NOT NULL,
    authors character varying(255) NOT NULL,
    url character varying(255) NOT NULL,
    description text,
    active boolean NOT NULL
);


ALTER TABLE public.stafdb_reference OWNER TO postgres;

--
-- Name: stafdb_reference_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_reference_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_reference_id_seq OWNER TO postgres;

--
-- Name: stafdb_reference_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
postgres
--

ALTER SEQUENCE public.stafdb_reference_id_seq OWNED BY public.stafdb_reference.id;
```

```
--
-- Name: stafdb_referencespace; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_referencespace (
    id integer NOT NULL,
    name character varying(255) NOT NULL,
    description text,
    url character varying(255),
    slug character varying(255),
    active boolean NOT NULL,
    location_id integer,
    parent_id integer
);


ALTER TABLE public.stafdb_referencespace OWNER TO postgres;

--
-- Name: stafdb_referencespace_geocode_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.stafdb_referencespace_geocode_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_referencespace_geocode_id_seq OWNER TO postgres;

--
-- Name: stafdb_referencespace_geocode; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_referencespace_geocode (
    id                              integer                              DEFAULT
nextval('public.stafdb_referencespace_geocode_id_seq'::regclass) NOT NULL,
    referencespace_id integer NOT NULL,
    geocode_id integer NOT NULL
);


ALTER TABLE public.stafdb_referencespace_geocode OWNER TO postgres;

--
-- Name: stafdb_referencespace_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.stafdb_referencespace_id_seq
    AS integer
```

```
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_referencespace_id_seq OWNER TO postgres;

--
-- Name: stafdb_referencespace_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.stafdb_referencespace_id_seq OWNED BY
public.stafdb_referencespace.id;


--
-- Name: stafdb_referencespacelocation; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_referencespacelocation (
    id integer NOT NULL,
    name character varying(255),
    lat character varying(20),
    lng character varying(20),
    area double precision,
    default_zoom smallint,
    description text,
    start date,
    "end" date,
    source character varying(255),
    geojson text,
    active boolean NOT NULL,
    space_id integer NOT NULL,
    CONSTRAINT stafdb_referencespacelocation_default_zoom_check CHECK
((default_zoom >= 0))
);


ALTER TABLE public.stafdb_referencespacelocation OWNER TO postgres;

--
-- Name: stafdb_referencespacelocation_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.stafdb_referencespacelocation_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER TABLE public.stafdb_referencespacelocation_id_seq OWNER TO postgres;

--
-- Name: stafdb_referencespacelocation_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_referencespacelocation_id_seq OWNED BY
public.stafdb_referencespacelocation.id;


--
-- Name: stafdb_timeperiod; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_timeperiod (
    id integer NOT NULL,
    start date NOT NULL,
    "end" date,
    name character varying(255) NOT NULL
);


ALTER TABLE public.stafdb_timeperiod OWNER TO postgres;

--
-- Name: stafdb_timeperiod_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_timeperiod_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_timeperiod_id_seq OWNER TO postgres;

--
-- Name: stafdb_timeperiod_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
postgres
--

ALTER SEQUENCE public.stafdb_timeperiod_id_seq OWNED BY
public.stafdb_timeperiod.id;


--
-- Name: stafdb_unit; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.stafdb_unit (
    id integer NOT NULL,
```

```
    symbol character varying(255) NOT NULL,
    name character varying(255) NOT NULL,
    notes text
);


ALTER TABLE public.stafdb_unit OWNER TO postgres;

--
-- Name: stafdb_unit_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.stafdb_unit_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.stafdb_unit_id_seq OWNER TO postgres;

--
-- Name: stafdb_unit_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public.stafdb_unit_id_seq OWNED BY public.stafdb_unit.id;


--
-- Name: auth_user id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.auth_user ALTER COLUMN id SET DEFAULT
nextval('public.auth_user_id_seq'::regclass);


--
-- Name: stafdb_csv id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_csv ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_csv_id_seq'::regclass);


--
-- Name: stafdb_data id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_data_id_seq'::regclass);


--
-- Name: stafdb_dataset id; Type: DEFAULT; Schema: public; Owner: postgres
```

```
--

ALTER TABLE ONLY public.stafdb_dataset ALTER COLUMN id SET DEFAULT nextval('public.stafdb_dataset_id_seq'::regclass);


--
-- Name: stafdb_dataset_references id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset_references ALTER COLUMN id SET DEFAULT nextval('public.stafdb_dataset_references_id_seq'::regclass);


--
-- Name: stafdb_dqi id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dqi ALTER COLUMN id SET DEFAULT nextval('public.stafdb_dqi_id_seq'::regclass);


--
-- Name: stafdb_dqirating id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dqirating ALTER COLUMN id SET DEFAULT nextval('public.stafdb_dqirating_id_seq'::regclass);


--
-- Name: stafdb_flowblocks id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_flowblocks ALTER COLUMN id SET DEFAULT nextval('public.stafdb_flowblocks_id_seq'::regclass);


--
-- Name: stafdb_flowdiagram id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_flowdiagram ALTER COLUMN id SET DEFAULT nextval('public.stafdb_flowdiagram_id_seq'::regclass);


--
-- Name: stafdb_geocode id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_geocode ALTER COLUMN id SET DEFAULT nextval('public.stafdb_geocode_id_seq'::regclass);


--
```

```
-- Name: stafdb_geocodesystem id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_geocodesystem ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_geocodesystem_id_seq'::regclass);


--
-- Name: stafdb_material id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_material ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_material_id_seq'::regclass);


--
-- Name: stafdb_materialcatalog id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_materialcatalog ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_materialcatalog_id_seq'::regclass);


--
-- Name: stafdb_process id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_process ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_process_id_seq'::regclass);


--
-- Name: stafdb_reference id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_reference ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_reference_id_seq'::regclass);


--
-- Name: stafdb_referencespace id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_referencespace_id_seq'::regclass);


--
-- Name: stafdb_referencespacelocation id; Type: DEFAULT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.stafdb_referencespacelocation ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_referencespacelocation_id_seq'::regclass);
```

```
--
-- Name: stafdb_timeperiod id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_timeperiod ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_timeperiod_id_seq'::regclass);


--
-- Name: stafdb_unit id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_unit ALTER COLUMN id SET DEFAULT
nextval('public.stafdb_unit_id_seq'::regclass);


--
-- Name: auth_user auth_user_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.auth_user
    ADD CONSTRAINT auth_user_pkey PRIMARY KEY (id);


--
-- Name: auth_user auth_user_username_key; Type: CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.auth_user
    ADD CONSTRAINT auth_user_username_key UNIQUE (username);


--
-- Name: stafdb_csv stafdb_csv_pkey; Type: CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.stafdb_csv
    ADD CONSTRAINT stafdb_csv_pkey PRIMARY KEY (id);


--
-- Name: stafdb_data stafdb_data_pkey; Type: CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_pkey PRIMARY KEY (id);


--
-- Name: stafdb_dataset stafdb_dataset_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--
```

```
ALTER TABLE ONLY public.stafdb_dataset
    ADD CONSTRAINT stafdb_dataset_pkey PRIMARY KEY (id);


--
-- Name: stafdb_dataset_references stafdb_dataset_references_dataset_id_reference_id_fd6d744c_uniq; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset_references
    ADD CONSTRAINT stafdb_dataset_references_dataset_id_reference_id_fd6d744c_uniq UNIQUE (dataset_id, reference_id);


--
-- Name: stafdb_dataset_references stafdb_dataset_references_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset_references
    ADD CONSTRAINT stafdb_dataset_references_pkey PRIMARY KEY (id);


--
-- Name: stafdb_dqi stafdb_dqi_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dqi
    ADD CONSTRAINT stafdb_dqi_pkey PRIMARY KEY (id);


--
-- Name: stafdb_dqirating stafdb_dqirating_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dqirating
    ADD CONSTRAINT stafdb_dqirating_pkey PRIMARY KEY (id);


--
-- Name: stafdb_flowblocks stafdb_flowblocks_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_flowblocks
    ADD CONSTRAINT stafdb_flowblocks_pkey PRIMARY KEY (id);


--
-- Name: stafdb_flowdiagram stafdb_flowdiagram_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY public.stafdb_flowdiagram
    ADD CONSTRAINT stafdb_flowdiagram_pkey PRIMARY KEY (id);


--
-- Name: stafdb_geocode stafdb_geocode_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.stafdb_geocode
    ADD CONSTRAINT stafdb_geocode_pkey PRIMARY KEY (id);


--
-- Name: stafdb_geocodesystem stafdb_geocodesystem_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_geocodesystem
    ADD CONSTRAINT stafdb_geocodesystem_pkey PRIMARY KEY (id);


--
-- Name: stafdb_material stafdb_material_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.stafdb_material
    ADD CONSTRAINT stafdb_material_pkey PRIMARY KEY (id);


--
-- Name: stafdb_materialcatalog stafdb_materialcatalog_pkey; Type: CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_materialcatalog
    ADD CONSTRAINT stafdb_materialcatalog_pkey PRIMARY KEY (id);


--
-- Name: stafdb_process stafdb_process_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.stafdb_process
    ADD CONSTRAINT stafdb_process_pkey PRIMARY KEY (id);


--
-- Name: stafdb_reference stafdb_reference_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.stafdb_reference
    ADD CONSTRAINT stafdb_reference_pkey PRIMARY KEY (id);
```

```
--
-- Name: stafdb_referencespace_geocode stafdb_referencespace_ge_referencespace_id_geocod_48b2ba6a_uniq; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace_geocode
    ADD CONSTRAINT stafdb_referencespace_ge_referencespace_id_geocod_48b2ba6a_uniq UNIQUE (referencespace_id, geocode_id);


--
-- Name: stafdb_referencespace_geocode stafdb_referencespace_geocode_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace_geocode
    ADD CONSTRAINT stafdb_referencespace_geocode_pkey PRIMARY KEY (id);


--
-- Name: stafdb_referencespace stafdb_referencespace_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace
    ADD CONSTRAINT stafdb_referencespace_pkey PRIMARY KEY (id);


--
-- Name: stafdb_referencespacelocation stafdb_referencespacelocation_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespacelocation
    ADD CONSTRAINT stafdb_referencespacelocation_pkey PRIMARY KEY (id);


--
-- Name: stafdb_timeperiod stafdb_timeperiod_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_timeperiod
    ADD CONSTRAINT stafdb_timeperiod_pkey PRIMARY KEY (id);


--
-- Name: stafdb_unit stafdb_unit_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_unit
    ADD CONSTRAINT stafdb_unit_pkey PRIMARY KEY (id);
```

```
--
-- Name: auth_user_username_6821ab7c_like; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX auth_user_username_6821ab7c_like ON public.auth_user USING btree (username varchar_pattern_ops);


--
-- Name: stafdb_csv_active_5ada15df; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_csv_active_5ada15df ON public.stafdb_csv USING btree (active);


--
-- Name: stafdb_csv_dataset_id_f5cbc137; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_csv_dataset_id_f5cbc137 ON public.stafdb_csv USING btree (dataset_id);


--
-- Name: stafdb_csv_user_id_43e6dd03; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_csv_user_id_43e6dd03 ON public.stafdb_csv USING btree (user_id);


--
-- Name: stafdb_data_csv_id_464851c6; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_data_csv_id_464851c6 ON public.stafdb_data USING btree (csv_id);


--
-- Name: stafdb_data_dataset_id_4bb36b25; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_data_dataset_id_4bb36b25 ON public.stafdb_data USING btree (dataset_id);


--
-- Name: stafdb_data_destination_space_id_8b3a623c; Type: INDEX; Schema: public; Owner: postgres
--
```

```
CREATE INDEX stafdb_data_destination_space_id_8b3a623c ON public.stafdb_data USING
btree (destination_space_id);


--
-- Name: stafdb_data_flow_id_3cfa90e0; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_data_flow_id_3cfa90e0 ON public.stafdb_data USING btree
(flow_id);


--
-- Name: stafdb_data_material_id_1dd63459; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_data_material_id_1dd63459 ON public.stafdb_data USING btree
(material_id);


--
-- Name: stafdb_data_origin_space_id_503fc524; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_data_origin_space_id_503fc524 ON public.stafdb_data USING btree
(origin_space_id);


--
-- Name: stafdb_data_subset_id_c24da2b1; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_data_subset_id_c24da2b1 ON public.stafdb_data USING btree
(subset_id);


--
-- Name: stafdb_data_timeframe_id_8663d339; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_data_timeframe_id_8663d339 ON public.stafdb_data USING btree
(timeframe_id);


--
-- Name: stafdb_data_unit_id_f4f15ba8; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_data_unit_id_f4f15ba8 ON public.stafdb_data USING btree
(unit_id);
```

```
--
-- Name: stafdb_dataset_access_id_5c5fc94f; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_access_id_5c5fc94f ON public.stafdb_dataset USING btree (access_id);


--
-- Name: stafdb_dataset_active_b7e12305; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_active_b7e12305 ON public.stafdb_dataset USING btree (active);


--
-- Name: stafdb_dataset_completeness_id_911eee85; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_completeness_id_911eee85 ON public.stafdb_dataset USING btree (completeness_id);


--
-- Name: stafdb_dataset_geographical_correlation_id_2c6ca2b7; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_geographical_correlation_id_2c6ca2b7 ON public.stafdb_dataset USING btree (geographical_correlation_id);


--
-- Name: stafdb_dataset_references_dataset_id_a5be5731; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_references_dataset_id_a5be5731 ON public.stafdb_dataset_references USING btree (dataset_id);


--
-- Name: stafdb_dataset_references_reference_id_74570b6a; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_references_reference_id_74570b6a ON public.stafdb_dataset_references USING btree (reference_id);


--
```

```
-- Name: stafdb_dataset_reliability_id_a3430559; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dataset_reliability_id_a3430559 ON public.stafdb_dataset USING btree (reliability_id);


--
-- Name: stafdb_dqirating_indicator_id_772236ae; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_dqirating_indicator_id_772236ae ON public.stafdb_dqirating USING btree (indicator_id);


--
-- Name: stafdb_flowblocks_destination_id_7d04261c; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_flowblocks_destination_id_7d04261c ON public.stafdb_flowblocks USING btree (destination_id);


--
-- Name: stafdb_flowblocks_diagram_id_206862fd; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_flowblocks_diagram_id_206862fd ON public.stafdb_flowblocks USING btree (diagram_id);


--
-- Name: stafdb_flowblocks_origin_id_4b13114d; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_flowblocks_origin_id_4b13114d ON public.stafdb_flowblocks USING btree (origin_id);


--
-- Name: stafdb_geocode_parent_id_c40eb671; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_geocode_parent_id_c40eb671 ON public.stafdb_geocode USING btree (parent_id);


--
-- Name: stafdb_geocode_system_id_1943b420; Type: INDEX; Schema: public; Owner: postgres
```

```
--

CREATE INDEX stafdb_geocode_system_id_1943b420 ON public.stafdb_geocode USING btree
(system_id);


--
-- Name: stafdb_material_catalog_id_32bd10bf; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_material_catalog_id_32bd10bf ON public.stafdb_material USING
btree (catalog_id);


--
-- Name: stafdb_material_code_80d3fe5e; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_material_code_80d3fe5e ON public.stafdb_material USING btree
(code);


--
-- Name: stafdb_material_code_80d3fe5e_like; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_material_code_80d3fe5e_like ON public.stafdb_material USING
btree (code varchar_pattern_ops);


--
-- Name: stafdb_material_name_66b1f923; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_material_name_66b1f923 ON public.stafdb_material USING btree
(name);


--
-- Name: stafdb_material_name_66b1f923_like; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_material_name_66b1f923_like ON public.stafdb_material USING
btree (name text_pattern_ops);


--
-- Name: stafdb_material_parent_id_dd728ec4; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_material_parent_id_dd728ec4 ON public.stafdb_material USING
btree (parent_id);
```

```
--
-- Name: stafdb_process_code_4f6d7229; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_process_code_4f6d7229 ON public.stafdb_process USING btree
(code);


--
-- Name: stafdb_process_code_4f6d7229_like; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_process_code_4f6d7229_like ON public.stafdb_process USING btree
(code varchar_pattern_ops);


--
-- Name: stafdb_process_name_0f9e0fe1; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_process_name_0f9e0fe1 ON public.stafdb_process USING btree
(name);


--
-- Name: stafdb_process_name_0f9e0fe1_like; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_process_name_0f9e0fe1_like ON public.stafdb_process USING btree
(name varchar_pattern_ops);


--
-- Name: stafdb_process_parent_id_bc4c539b; Type: INDEX; Schema: public; Owner:
postgres
--

CREATE INDEX stafdb_process_parent_id_bc4c539b ON public.stafdb_process USING btree
(parent_id);


--
-- Name: stafdb_process_slug_14bbbbf2; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_process_slug_14bbbbf2 ON public.stafdb_process USING btree
(slug);


--
-- Name: stafdb_process_slug_14bbbbf2_like; Type: INDEX; Schema: public; Owner:
postgres
```

```
--

CREATE INDEX stafdb_process_slug_14bbbbf2_like ON public.stafdb_process USING btree (slug varchar_pattern_ops);


--
-- Name: stafdb_referencespace_active_24962fe3; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_active_24962fe3 ON public.stafdb_referencespace USING btree (active);


--
-- Name: stafdb_referencespace_geocode_geocode_id_f11d0396; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_geocode_geocode_id_f11d0396 ON public.stafdb_referencespace_geocode USING btree (geocode_id);


--
-- Name: stafdb_referencespace_geocode_referencespace_id_3bb6d502; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_geocode_referencespace_id_3bb6d502 ON public.stafdb_referencespace_geocode USING btree (referencespace_id);


--
-- Name: stafdb_referencespace_location_id_e55f2a80; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_location_id_e55f2a80 ON public.stafdb_referencespace USING btree (location_id);


--
-- Name: stafdb_referencespace_name_5b76bb39; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_name_5b76bb39 ON public.stafdb_referencespace USING btree (name);


--
-- Name: stafdb_referencespace_name_5b76bb39_like; Type: INDEX; Schema: public; Owner: postgres
--
```

```
CREATE INDEX stafdb_referencespace_name_5b76bb39_like ON public.stafdb_referencespace USING btree (name varchar_pattern_ops);


--
-- Name: stafdb_referencespace_parent_id_14e1ebb6; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_parent_id_14e1ebb6 ON public.stafdb_referencespace USING btree (parent_id);


--
-- Name: stafdb_referencespace_slug_afc2f623; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_slug_afc2f623 ON public.stafdb_referencespace USING btree (slug);


--
-- Name: stafdb_referencespace_slug_afc2f623_like; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespace_slug_afc2f623_like ON public.stafdb_referencespace USING btree (slug varchar_pattern_ops);


--
-- Name: stafdb_referencespacelocation_active_edf887a0; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespacelocation_active_edf887a0 ON public.stafdb_referencespacelocation USING btree (active);


--
-- Name: stafdb_referencespacelocation_end_83501dc3; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespacelocation_end_83501dc3 ON public.stafdb_referencespacelocation USING btree ("end");


--
-- Name: stafdb_referencespacelocation_space_id_02983803; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespacelocation_space_id_02983803 ON public.stafdb_referencespacelocation USING btree (space_id);
```

```
--
-- Name: stafdb_referencespacelocation_start_a48f177b; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_referencespacelocation_start_a48f177b ON public.stafdb_referencespacelocation USING btree (start);


--
-- Name: stafdb_timeperiod_end_d5bad739; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_timeperiod_end_d5bad739 ON public.stafdb_timeperiod USING btree ("end");


--
-- Name: stafdb_timeperiod_name_3dab2e31; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_timeperiod_name_3dab2e31 ON public.stafdb_timeperiod USING btree (name);


--
-- Name: stafdb_timeperiod_name_3dab2e31_like; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_timeperiod_name_3dab2e31_like ON public.stafdb_timeperiod USING btree (name varchar_pattern_ops);


--
-- Name: stafdb_timeperiod_start_a9fa7bad; Type: INDEX; Schema: public; Owner: postgres
--

CREATE INDEX stafdb_timeperiod_start_a9fa7bad ON public.stafdb_timeperiod USING btree (start);


--
-- Name: stafdb_csv stafdb_csv_dataset_id_f5cbc137_fk_stafdb_dataset_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_csv
    ADD CONSTRAINT stafdb_csv_dataset_id_f5cbc137_fk_stafdb_dataset_id FOREIGN KEY (dataset_id) REFERENCES public.stafdb_dataset(id) DEFERRABLE INITIALLY DEFERRED;
```

```
--
-- Name: stafdb_csv stafdb_csv_user_id_43e6dd03_fk_auth_user_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_csv
    ADD CONSTRAINT stafdb_csv_user_id_43e6dd03_fk_auth_user_id FOREIGN KEY (user_id) REFERENCES public.auth_user(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_csv_id_464851c6_fk_stafdb_csv_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_csv_id_464851c6_fk_stafdb_csv_id FOREIGN KEY (csv_id) REFERENCES public.stafdb_csv(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_dataset_id_4bb36b25_fk_stafdb_dataset_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_dataset_id_4bb36b25_fk_stafdb_dataset_id FOREIGN KEY (dataset_id) REFERENCES public.stafdb_dataset(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_destination_space_id_8b3a623c_fk_stafdb_re; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_destination_space_id_8b3a623c_fk_stafdb_re FOREIGN KEY (destination_space_id) REFERENCES public.stafdb_referencespace(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_flow_id_3cfa90e0_fk_stafdb_flowblocks_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_flow_id_3cfa90e0_fk_stafdb_flowblocks_id FOREIGN KEY (flow_id) REFERENCES public.stafdb_flowblocks(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_material_id_1dd63459_fk_stafdb_material_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_material_id_1dd63459_fk_stafdb_material_id FOREIGN
KEY (material_id) REFERENCES public.stafdb_material(id) DEFERRABLE INITIALLY
DEFERRED;


--
-- Name: stafdb_data stafdb_data_origin_space_id_503fc524_fk_stafdb_re; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_origin_space_id_503fc524_fk_stafdb_re FOREIGN KEY
(origin_space_id) REFERENCES public.stafdb_referencespace(id) DEFERRABLE INITIALLY
DEFERRED;


--
-- Name: stafdb_data stafdb_data_subset_id_c24da2b1_fk_stafdb_data_id; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_subset_id_c24da2b1_fk_stafdb_data_id FOREIGN KEY
(subset_id) REFERENCES public.stafdb_data(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_timeframe_id_8663d339_fk_stafdb_timeperiod_id;
Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_timeframe_id_8663d339_fk_stafdb_timeperiod_id
FOREIGN KEY (timeframe_id) REFERENCES public.stafdb_timeperiod(id) DEFERRABLE
INITIALLY DEFERRED;


--
-- Name: stafdb_data stafdb_data_unit_id_f4f15ba8_fk_stafdb_unit_id; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_data
    ADD CONSTRAINT stafdb_data_unit_id_f4f15ba8_fk_stafdb_unit_id FOREIGN KEY
(unit_id) REFERENCES public.stafdb_unit(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_dataset stafdb_dataset_access_id_5c5fc94f_fk_stafdb_dqirating_id;
Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset
```

```
    ADD CONSTRAINT stafdb_dataset_access_id_5c5fc94f_fk_stafdb_dqirating_id FOREIGN KEY (access_id) REFERENCES public.stafdb_dqirating(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_dataset stafdb_dataset_completeness_id_911eee85_fk_stafdb_dqirating_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset
    ADD CONSTRAINT stafdb_dataset_completeness_id_911eee85_fk_stafdb_dqirating_id FOREIGN KEY (completeness_id) REFERENCES public.stafdb_dqirating(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_dataset stafdb_dataset_geographical_correla_2c6ca2b7_fk_stafdb_dq; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset
    ADD CONSTRAINT stafdb_dataset_geographical_correla_2c6ca2b7_fk_stafdb_dq FOREIGN KEY (geographical_correlation_id) REFERENCES public.stafdb_dqirating(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_dataset_references stafdb_dataset_refer_dataset_id_a5be5731_fk_stafdb_da; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset_references
    ADD CONSTRAINT stafdb_dataset_refer_dataset_id_a5be5731_fk_stafdb_da FOREIGN KEY (dataset_id) REFERENCES public.stafdb_dataset(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_dataset_references stafdb_dataset_refer_reference_id_74570b6a_fk_stafdb_re; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dataset_references
    ADD CONSTRAINT stafdb_dataset_refer_reference_id_74570b6a_fk_stafdb_re FOREIGN KEY (reference_id) REFERENCES public.stafdb_reference(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_dataset stafdb_dataset_reliability_id_a3430559_fk_stafdb_dqirating_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY public.stafdb_dataset
    ADD   CONSTRAINT   stafdb_dataset_reliability_id_a3430559_fk_stafdb_dqirating_id
FOREIGN  KEY  (reliability_id)  REFERENCES  public.stafdb_dqirating(id)  DEFERRABLE
INITIALLY DEFERRED;


--
-- Name: stafdb_dqirating stafdb_dqirating_indicator_id_772236ae_fk_stafdb_dqi_id;
Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_dqirating
    ADD CONSTRAINT stafdb_dqirating_indicator_id_772236ae_fk_stafdb_dqi_id FOREIGN
KEY (indicator_id) REFERENCES public.stafdb_dqi(id) DEFERRABLE INITIALLY DEFERRED;


--
--                              Name:                                stafdb_flowblocks
stafdb_flowblocks_destination_id_7d04261c_fk_stafdb_process_id;       Type:       FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_flowblocks
    ADD  CONSTRAINT  stafdb_flowblocks_destination_id_7d04261c_fk_stafdb_process_id
FOREIGN   KEY   (destination_id)   REFERENCES   public.stafdb_process(id)   DEFERRABLE
INITIALLY DEFERRED;


--
--                              Name:                                stafdb_flowblocks
stafdb_flowblocks_diagram_id_206862fd_fk_stafdb_flowdiagram_id;       Type:       FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_flowblocks
    ADD  CONSTRAINT  stafdb_flowblocks_diagram_id_206862fd_fk_stafdb_flowdiagram_id
FOREIGN   KEY   (diagram_id)   REFERENCES   public.stafdb_flowdiagram(id)   DEFERRABLE
INITIALLY DEFERRED;


--
--                              Name:                                stafdb_flowblocks
stafdb_flowblocks_origin_id_4b13114d_fk_stafdb_process_id;   Type:   FK   CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_flowblocks
    ADD     CONSTRAINT     stafdb_flowblocks_origin_id_4b13114d_fk_stafdb_process_id
FOREIGN KEY (origin_id) REFERENCES public.stafdb_process(id) DEFERRABLE INITIALLY
DEFERRED;


--
--   Name:   stafdb_geocode   stafdb_geocode_parent_id_c40eb671_fk_stafdb_geocode_id;
Type: FK CONSTRAINT; Schema: public; Owner: postgres
```

```
--

ALTER TABLE ONLY public.stafdb_geocode
    ADD CONSTRAINT stafdb_geocode_parent_id_c40eb671_fk_stafdb_geocode_id FOREIGN KEY (parent_id) REFERENCES public.stafdb_geocode(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_geocode stafdb_geocode_system_id_1943b420_fk_stafdb_geocodesystem_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_geocode
    ADD CONSTRAINT stafdb_geocode_system_id_1943b420_fk_stafdb_geocodesystem_id FOREIGN KEY (system_id) REFERENCES public.stafdb_geocodesystem(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_material stafdb_material_catalog_id_32bd10bf_fk_stafdb_ma; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_material
    ADD CONSTRAINT stafdb_material_catalog_id_32bd10bf_fk_stafdb_ma FOREIGN KEY (catalog_id) REFERENCES public.stafdb_materialcatalog(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_material stafdb_material_parent_id_dd728ec4_fk_stafdb_material_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_material
    ADD CONSTRAINT stafdb_material_parent_id_dd728ec4_fk_stafdb_material_id FOREIGN KEY (parent_id) REFERENCES public.stafdb_material(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_process stafdb_process_parent_id_bc4c539b_fk_stafdb_process_id; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_process
    ADD CONSTRAINT stafdb_process_parent_id_bc4c539b_fk_stafdb_process_id FOREIGN KEY (parent_id) REFERENCES public.stafdb_process(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_referencespace_geocode stafdb_referencespac_geocode_id_f11d0396_fk_stafdb_ge; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY public.stafdb_referencespace_geocode
    ADD CONSTRAINT stafdb_referencespac_geocode_id_f11d0396_fk_stafdb_ge FOREIGN
KEY (geocode_id) REFERENCES public.stafdb_geocode(id) DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_referencespace
stafdb_referencespac_location_id_e55f2a80_fk_stafdb_re; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace
    ADD CONSTRAINT stafdb_referencespac_location_id_e55f2a80_fk_stafdb_re FOREIGN
KEY (location_id) REFERENCES public.stafdb_referencespacelocation(id) DEFERRABLE
INITIALLY DEFERRED;


--
-- Name: stafdb_referencespace
stafdb_referencespac_parent_id_14e1ebb6_fk_stafdb_re; Type: FK CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace
    ADD CONSTRAINT stafdb_referencespac_parent_id_14e1ebb6_fk_stafdb_re FOREIGN KEY
(parent_id) REFERENCES public.stafdb_referencespace(id) DEFERRABLE INITIALLY
DEFERRED;


--
-- Name: stafdb_referencespace_geocode
stafdb_referencespac_referencespace_id_3bb6d502_fk_stafdb_re; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespace_geocode
    ADD CONSTRAINT stafdb_referencespac_referencespace_id_3bb6d502_fk_stafdb_re
FOREIGN KEY (referencespace_id) REFERENCES public.stafdb_referencespace(id)
DEFERRABLE INITIALLY DEFERRED;


--
-- Name: stafdb_referencespacelocation
stafdb_referencespac_space_id_02983803_fk_stafdb_re; Type: FK CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.stafdb_referencespacelocation
    ADD CONSTRAINT stafdb_referencespac_space_id_02983803_fk_stafdb_re FOREIGN KEY
(space_id) REFERENCES public.stafdb_referencespace(id) DEFERRABLE INITIALLY
DEFERRED;


--
-- PostgreSQL database dump complete
--
```

Annex 2

Example spreadsheet for data collection

Timeframe name	Frome (date)	To (date)	Material name	Material code	Quantity	Unit	Origin (reference space)	Destination (reference space)	Origin (process)	Destination (process)	Comments
Q1 2020	2010-01-01	2010-03-31	Cement	EW202	4034	t	Germany	Apeldoorn	Mining	Manufacturing	
2019	2019-01-01	2019-12-31	Glass	EW399	110	t		Apeldoorn	Distribution	Construction	

CITYLOOPS

CityLoops is an EU-funded project focusing on construction and demolition waste (CDW), including soil, and organic waste (OW), where seven European cities are piloting solutions to be more circular.

Høje-Taastrup and Roskilde (Denmark), Mikkeli (Finland), Apeldoorn (the Netherlands), Bodø (Norway), Porto (Portugal) and Seville (Spain) are the seven cities implementing a series of demonstration actions on CDW and soil, and OW, and developing and testing over 30 new tools and processes.

Alongside these, a sector-wide circularity assessment and an urban circularity assessment are to be carried out in each of the cities. The former, to optimise the demonstration activities, whereas the latter to enable cities to effectively integrate circularity into planning and decision making. Another two key aspects of CityLoops are stakeholder engagement and circular procurement.

CityLoops started in October 2019 and will run until September 2023.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 821033.