3. Data Management

In B2B electronic data interchange (EDI) messages, comprising product or service codes, transport unit identifications, as well as legal documents are exchanged among partners in a supply chain. They usually have the form of alphanumeric strings.

3.1 Information-Data-Knowledge

In computer-based information systems the representation of information varies depending on its purpose and use. It may be alphanumeric or binary considering the fact whether it represents a text, figure, sound or executable program… To be able to store, retrieve, process and transmit data of various types (e.g., numbers, characters, dates, currencies, etc.) they need to be properly encoded. Alphanumeric formats of data representation have their origins in the ASCII (ask-key) alphabet, having evolved in the sense of national character sets (e.g., standards 8859-1, Latin 1 and 8859-2, Latin 2) and finally progressed into international UTF-8 and UTF-16 formats. Thus, they enable common data interpretation by business partners belonging to different ethnic groups and geographic environments. While alphanumeric strings depend on their encoding, numeric data mainly differ in their size and/or precision.

The process of transforming data from their original analogue into the digital form is popularly termed digitization. Appropriately designed utility and application programs accepting data from various sources (e.g., optical scanners, electrical sensors, EDI, etc.) enable organizations to automate their data acquisition, storage, processing and transmission within and between their computer-based information systems.

When collected, data of various types may be joined and organized into tables of data, data bases, data warehouses (chronological order) or knowledge bases (conceptual order). Higher levels of data organization allow for automated classification, reasoning and representation of the hereby accumulated knowledge for analytic purposes.

3.2 Logistic Data

In logistics EDI is used to transmit transaction data between business partners. Since they may use different languages and applications, their conversion to a common format (e.g., XML, JSON) is necessary to be interpretable by different partner's information systems (W3schools, 2023). For quick identification and manipulation barcodes and RFID tags were devised.

To allow for international cooperation, globally accepted data formats needed to be defined for logistics purposes. Logistics data formats correspond with service and product labels, transport unit identifications and transaction codes, usually having the form of time-stamped alphanumeric strings. For simplicity of manipulation and processing speed these codes were standardized and encoded as optically readable bar-codes or electromagnetically readable radio-frequency identification (RFID) codes.

Bar-code (EAN/UCC) is a multi-sector and international form of numbering items (POS EAN-8 and EAN-13, changeable EAN-128, data bar, packaging ITF-14, QR, data matrix, etc.). They are used to identify products, product batches or shipments (1D codes) as well as services (2D codes). Among 2D bar codes the QR code is the most established one, readable also by smartphones, which increases their usability in different application areas.

RFID codes are primarily used in the same way as bar codes. They uniquely identify items or services. Typically, besides an optional bar code, RFID labels carry even more information on a chip the size of a pin head. Besides identification RFID tags enable the logging of tracking information, often required in logistics applications. In contrast to bar codes, RFID enables their scanning in without a direct line of sight and also multiple labels at once.

By the GS1 standard EPC Gen2 (ISO/IEC 18000-6:2013) a technological standard determining communication between RFID tags and readers was established. Similar to bar codes, EPCglobal standards link RFID technology with EPC tagging of products, logistic transport units, locations, inventory, returnable items, documents, etc. for direct, automated identification and tracking of logistics units within supply chains.

EPCglobal standards also represent the basis of the GDSN (Global Data Synchronization Network). It enables automated acquisition and interchange of specification data on products

and their packaging, hereby enabling enterprises to centrally manage these data to be used by them and their partners interchangeably.

Table 3.1 summarizes the various identification technologies with their applications. It reveals a variety of one and two-dimensional bar codes as well as different classes of RFID codes with their capabilities.

Table 3.1 Tagging technologies.

Technology	Application
Bar 1D	Retail items and product components
Bar 2D	Services (e.g., UPS, air-tickets), wholesale items requiring tracking
RFID class 1 (passive, R-tags)	Items requiring mass identification, access control
RFID class 2 (passive, RW-tags)	Items requiring tracking
RFID class 3 (semi-active, RW-tags)	Access control with added tracking information
RFID class 4 (active, RW-tags)	Closed space tracking and tracing
RFID class 5 (active tags/interrogators)	Open space tracking and tracing, proximity services with enabled devices, location-based services

Future trends in tagging, tracking and tracing follow two main directions: miniaturization and diversity. The data bar (1D) codes shall also enable tagging miniature items (e.g., medical capsules). Novel data matrix (2D) codes shall not only enable error correction while scanning but also data encryption.

RFID continues to spread to other usage areas like the identification by service providers (e.g., railcard, registration at work, etc.), contactless payments (e.g., wireless money transfers, payments at vending machines) and smart solutions (e.g., smart home management, remotely operating smart devices, etc.) as well as e-currencies.

3.3 Data organization

Apart from having a certain format the data may be organized in different ways to facilitate their management, processing and presentation. Although data on their input are mostly unstructured, by their storage, transmission and processing their organization increases. In the sequel the usual forms of data organization are presented by increasing complexity from semi-structured (e.g., CSV) to structured (e.g., spreadsheets, databases, etc.) formats.

Spreadsheets

The first form of data organization is by two-dimensional arrays of fields, also called tables or spreadsheets. Usually, the first line of a spreadsheet denotes the meanings of values stored in the underlying columns, followed by lines of data.

A table field or cell is the smallest unit of data. It has a certain type (string, number, date, currency, etc.). Its contents are addressable by the row and column markings (e.g., A1, representing the first row of column A).

Each table line is a group of related fields, representing a record (e.g.: transaction, student record, product data, etc.). Since all table lines have the same structure, we may define the type of a record as a list of attributes (e.g., Student data (name, family name, birthdate, birthplace, ID…)) of the corresponding data types.

Databases

A database file or table is a collection of records of the same type. A database (DB) is made up of multiple inter-connected tables. Hence, the ANSI definition of a database:

- DB data are interconnected and sorted
- DB data can be simultaneously used by multiple users
- Data in the DB are not repeated
- DB is stored in a computer

From the above definition one may draw some conclusions about the client-server architecture where the server holds the DB, being accessed by its clients. Of course, to be able to access the DB a communication network has to be established between the server and its clients. The

DB server is usually termed as its "back-end", while the clients represent its "front-end". The database management system (DBMS) at the server enables its clients access to the data stored in the DB via its application program interface (API) and DBM functions. The DBM functions are mechanisms that enable DB data input, retrieving, processing and presentation. To call up these functions standard query languages (SQL) have been defined.

Relational database model

There are various forms of DB organization with the relational model (RDB) being the most common. The basic idea behind this model is the fact that a user cannot know all the possible uses data stored in a DB upfront. Since there are usually no fixed paths of searching through the database files, various query languages have been devised for data retrieval and manipulation. The RDB model is based on the concept of entities and relations:

- An entity is a person/thing/concept that can be uniquely identified and has attributes.
- A relation represents a way to associate two or more entities.

RDB tables, representing entities or relations, are interconnected by means of keys. The set of attributes that uniquely identify an entity is termed its primary key. When a primary key appears as a field in another table with the goal to fulfil a relation with its original table, it is termed secondary or foreign key. While a table may contain only one primary key to uniquely identify its records, it may contain multiple secondary keys.

In general, there are two approaches to constructing an RDB: analytic and synthetic.

The analytic approach to RDB construction comprises the following four steps:

1. Real world analysis - global model
2. Determine entities and relations – conceptual model (e.g., E-R diagram)
3. Determine logical model – relational schema
4. Build the database (DBMS) – physical model

To illustrate the approach, let us consider an example of a convenience store chain and their suppliers (Figure 3.1). Each store has multiple suppliers. Every supplier may supply different stores. The replenishment cycle starts by an order from the store. In return a supplier delivers goods to the store. Orders are transactions in which the data on the store, supplier and goods delivered are combined (Figure 3.2).

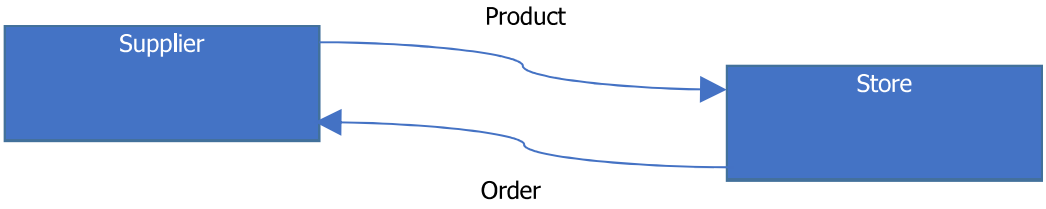

Figure 3.1 Global model.

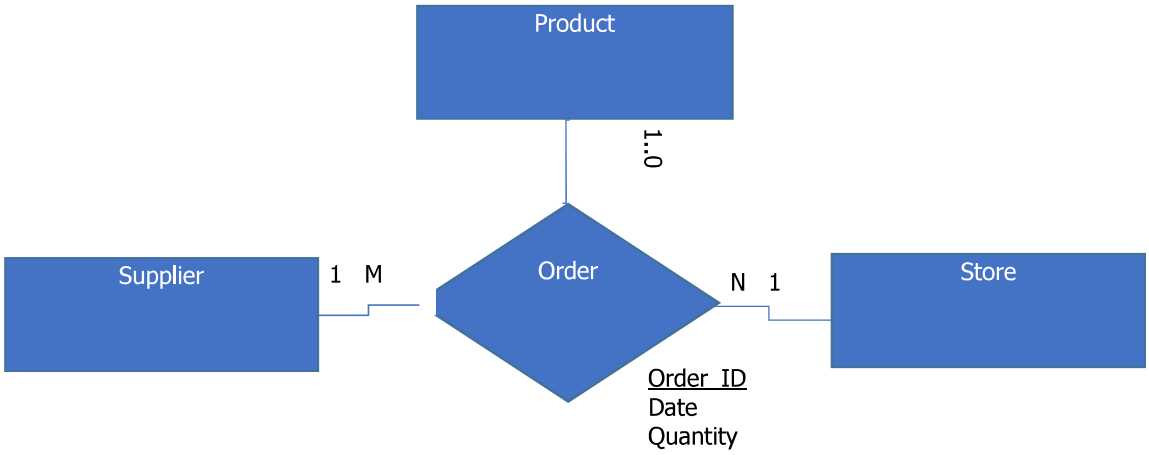

Figure 3.2 Conceptual model.

SUPPLIER (Supplier_ID#, Supplier_name, Supplier_contact)

STORE (Store_ID#, Store_name, Store_address)

ORDER (Supplier_ID#, Store_ID#, Order_ID#, Date, Quantity, EPC)

PRODUCT (EPC#, Product_name, Product_price)

Figure 3.3 Logical model.

The logical model represents the RDB tables by the types of their records. In the logical model (Figure 3.3) certain attributes contain the hashtag (#) symbol. This signifies that the field they represent is or belongs to a (composite) primary key. Some fields are underlined. They are called secondary or foreign keys, since they reference the primary keys of related tables.

The synthetic approach to an RDB comprises the following three steps:

1. Data analysis – list of all relevant attributes
2. Determine logical model by normalization – relational schema

3. Physical model (DBMS)

Normalization or canonical synthesis (Kent, 1983) ensures that by reverse engineering from the relevant attributes a DB is formed fulfilling the conditions of a RDB (cp. Figure 4). While the initial normal form (0NF) of attributes represents a table of unordered attributes, the subsequent normal forms, as defined by (Codd, 1970), represent higher levels of data organization. One may claim that by reaching the 3rd normal form one has achieved a schema fulfilling the requirements for a RDB logical model.

A table is in 1NF, if it represents a relation. By this it is ensured that all repeating groups of data are kept separately and hence do not repeat.

A table in 2NF is in 1NF. Additionally, no key-attributes may be partially functionally dependent on the primary key. Hereby, all keys that uniquely identify certain attributes are kept separately. Mainly this is to ensure that only attributes that are dependent on all (parts of) the primary key are kept in a single table.

A table in 3NF is in 2NF. Additionally, no non-key attributes are transitively dependent on the primary key. This means that all non-key attributes that may represent a key to certain other non-key attributes are kept in a separate table, whereby only their key is maintained in the original table as foreign key.

By following the normalization steps from the 1NF to the 3NF one ends up with a logical model that corresponds to the regulations of a relational database (RDB). Higher forms of normalization are mainly for RDB model optimization.

A table in Boyce-Codd NF (BCNF) is in 3NF; additionally, every determinant is a key. This removes all relations not covered by existing ordering of keys from the original table, hereby forming additional tables for every candidate key. A table in 4NF is in 3NF and BCNF. Additionally, every multi-valued attribute that is partially dependent on the key is in its own table. 4NF is meant to remove all possible remaining multi-valued attributes from the original table. A table is 5NF is in 4NF; additionally, every JOIN operation is foreseen by the keys. A table in 6NF is in 5NF; additionally, all non-trivial JOIN-dependencies are accounted for.

One can observe that by higher forms of organization, the number of tables increases with each step. Hence, it is sensible to observe the fragmentation of data in order to prevent unnecessary infrequently accessed tables from being created.

Table 3.2 RBD Normalization example.

ONF ORDER • Supplier_ID* • Supplier_name • Supplier_contact • Store_ID* • Store_name • Store_address • Order_ID* • Date • EPC • Quantity • Product_name • Product_price	1NF ORDER • Supplier_ID# • Supplier_name • Supplier_contact • Store_ID# • Store_name • Store_address • Order_ID# • Date • EPC • Quantity • Product_name • Product_price
* Candidate keys of the repeating group of attributes, uniquely identifying an order	
2NF SUPPLIER • Supplier_ID# • Supplier_name • Supplier_contact STORE • Store_ID# • Store_name • Store_address	ORDER • Order_ID# • Supplier_ID# • Store_ID# • EPC • Product_name • Product_price • Date • Quantity
3NF ORDER • Supplier_ID# • Store_ID# • Order_ID# • Date • Quantity • Product_ID	PRODUCT • EPC# • Product_name • Product_price

Query languages

They are mainly of two types: Structured Query Language (SQL) and Query by Example (QBE). While SQL is considered a programming language and a DBMS's API, QBE is mainly used with the DBMS directly for DB management and data warehousing.

The standard SQL (ISO/IEC 9075, 1986-2016) is a fourth-generation programming language for database manipulation. It enables searching, adding, modifying as well as deleting data records. Despite its standardization there are slight differences in its implementation with different database management systems (DBMS).

In the sequel the language is briefly presented with the most common options. By convention the SQL keywords are written in capital letters and every sentence ends with a semicolon. The sentences are presented with links to associated reference materials offering further information.

Every database manipulation starts by its creation. The sentence

CREATE DATABASE *database_name*;

creates a new empty database with the specified name.

As elaborated above, the data within databases are organized in tables of data records of a certain type where all rows share a common structure. To create a table the following sentence is used:

CREATE TABLE *table_name* (
column1 type1,
column2 type2,
column3 type3,
….);

Every named column represents an attribute with a certain data type. For example, in:

CREATE TABLE *Store* (Store_*ID* int NOT NULL PRIMARY KEY,…);

CREATE TABLE *Order* (Order_*ID* int NOT NULL PRIMARY KEY, …, Product_*Id* int FOREIGN KEY REFERENCES *Product* (*EPC*));

two tables are created. The first contains the data on the customers, while the second contains the data on their orders, referencing the first table via the customer number as foreign key.

While data in a table is already sorted by the primary key, it can be additionally sorted by other attributes, provided it is indexed. We can index it by creating an index on the provided attribute(s) by the following sentence:

CREATE INDEX *index_name* ON *table_name* (*column_name*);

Every data manipulation on an indexed table takes a little longer, since for its consistency, not only the data provided by keys needs to be checked and the data ordered appropriately, but also other attributes from the specified index.

The most common operation on a database is data query enabled by the SELECT statement:

SELECT *column1, column2*, …
FROM *table_name*;

This data query returns data in column1, column2, etc. from the table. The query sentences are usually formed by providing additional options, filtering out data, fulfilling the specified condition(s):

WHERE specifies a condition, which determines the records selection criteria.

GROUP BY joins records, having a common property to enable aggregate functions.

HAVING specifies aggregate functions on groups defined by GROUP BY statements.

ORDER BY specifies the attributes on which the return records are ordered.

For example:

```
SELECT "Store"."Store_Name", "Product"."Product_name", "Order"."Quantity" FROM "Order",
"Product", "Supplier", "Store" WHERE "Order"."Product_ID" = "Product"."EPC" AND
"Order"."Supplier_ID" = "Supplier"."Supplier_ID" AND "Order"."Store_ID" =
"Store"."Store_ID" ORDER BY "Store"."Store_Name" ASC
```

returns a list of stores with their ordered products and quantities, ordered by store name.

The most important operation in the selection process is the JOIN operation. Often, it replaces the WHERE condition as JOIN ON, followed by the condition. It compares the column values and determines, based on the comparison, whether or not they should be included in the result. In LEFT JOIN the record is returned, if the criteria are fulfilled in the left table and vice versa in the RIGHT JOIN operation. As outlined above, the condition needs to be fulfilled in

both tables in order to comply with the INNER JOIN or FULL JOIN operation. Since the latter is most commonly used, one can use JOIN as synonym. Referring to the conditions of the 5th and 6th normal forms, this is the same JOIN operation, which needs to be fulfilled to comply with the terms of the corresponding NF.

To input new data into the table the INSERT INTO operation is used:

INSERT INTO *table_name* (*column1*, [*column2*, ...])
VALUES (*value1*, [*value2*, ...]);

To be successful, the values in the operation need to fulfil all the conditions of the attributes denoted by column names. One does not need to specify column names in case all values are listed. In case some DEFAULT values are foreseen in the table, one does not need to specify them, unless they are different.

Once the data are input, they can be modified by an UPDATE statement:

UPDATE *table_name*
SET *column1=value1, column2=value2*,...
WHERE *some_column=some_value*;

In the statement new values for the fields in the listed columns are given. The row selection criteria are denoted by the WHERE specifier, which determines all column values to which the UPDATE statement applies. In order to prevent unwanted changes, extra caution needs to be applied with the formulation of the selection criteria.

A record or multiple records can be deleted from a table by the DELETE operation:

DELETE FROM *table_name*
WHERE *some_column=some_value*;

As with the UPDATE statement, the WHERE specifier is used to determine all rows that should be deleted.

Of course, database management doesn't end here. Every element of the database can also be removed, altered and/or replaced by a new one. In case an index, a table or a database is to be removed, the following statements can be applied:

DROP INDEX *index_name* ON *table_name*;

DROP TABLE *table_name*;

DROP DATABASE *database_name*;

If one only wishes to remove data from a table the TRUNCATE statement may be used:

TRUNCATE TABLE *table_name*;

In case one wishes to add or remove an attribute (column) to/from a table, one can do so by the ALTER statement:

ALTER TABLE *table_name* ADD *column_name datatype*;

ALTER TABLE *table_name* DROP COLUMN *column_name*;

This concludes this short overview of the SQL language and its most common usage scenarios. SQL is commonly used in client-server architectures with the DBMS hosted by the server. To access it, SQL statements are issued, either by a client application programme or the server's DBMS Web-interface.

On the other hand, QBE is also often used with relational DBMS with a graphical user interface (GUI), like the MS Access or LibreOffice Base. With QBE, the database and its tables are created much more interactively and their structure is easier to maintain. As its name suggests, it also offers a simpler form of data input and discovery. To perform searches, one needs to assemble all tables that are used in the search and then establish conditions as patterns in column fields to filter out the relevant data (Figure 3.4). The query formulation is supplemented by the existing relations between tables. As usual, the result from such a query is another table with the resulting data, which can be further processed later on. This way also cascading or multi-phase queries can be formed.

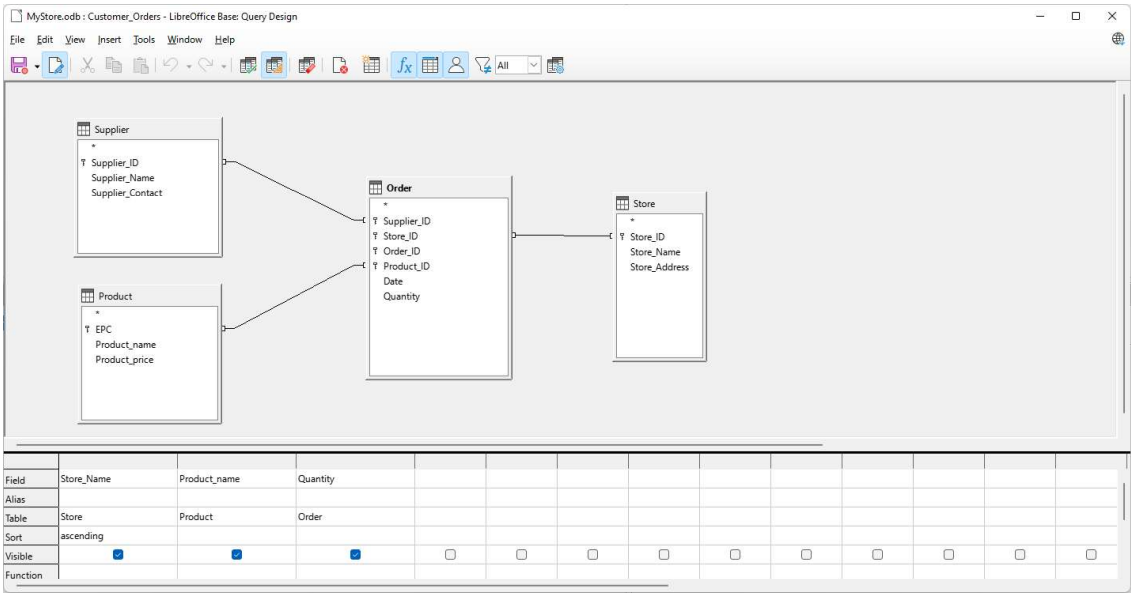

Figure 3.4 Query by Example equivalent to the above SQL query.

Data filters and masks

In order to prevent erroneous or incomplete data, which could interfere with their processing and interpretation, additional precautions should be applied:

1. Filtering out empty rows and columns
2. Applying strong data typing to prevent computational errors
3. Defining input masks to prevent input of wrong data
4. Data biasing to prevent erroneous results

Empty rows and columns are a common source of errors that originate mainly from poor interfaces in data collection applications. Cancelled or incomplete transactions usually result in empty lines or missing data in transaction logs. They can only partly be dealt with spreadsheet applications where empty lines and missing data can be detected by checking the total number vs. the number of non-zero data in rows/columns. They can be filtered out by removing the empty rows/columns, however, this may not always be the desired action, since we might lose some valuable data as well. The best way to prevent this from happening is by using a DBMS which would only allow complete transaction data to be entered on one hand, whereas on the other, it would also prevent empty rows/columns, since in databases there are none.

Weak data typing is another common source of errors. If alphanumeric data, like date or the amount of currency, is entered in place of numeric data, this would result in errors when processing these data. In spreadsheets as well as in databases the individual data cells, representing attribute values in data records, can be assigned data types, which would alarm us, when entering data in the wrong format. Thus, by this measure one can prevent wrong data to interfere with their processing.

When constructing databases, restrictions can be applied on the data fields representing entity or relation attributes. Apart from assigning them an appropriate type, input masks can be defined allowing data, like dates, currencies, EAN codes, etc., to be input only in a certain format. This usually resolves many misconceptions, which could otherwise occur while processing the data.

Another common source of errors are unbiased data, representing data that are orders of magnitude greater or smaller than expected. Again, they could interfere with our processing, yielding erroneous results. They are harder to detect and can only be filtered out by looking at the data. In spreadsheet applications a good common practice would be to determine the minimum, maximum and mean values of data in the corresponding columns to detect possible deviations. If they are detected, they can then be highlighted and dealt with manually, if they are a few, or filtered out and modified by a query on a database table, if they are many. Either way they should be assessed carefully, in order not to make the situation even worse, in which case it would be better to remove these data.

Data warehouses and knowledge bases

Data in data warehouses are collected from RDB and catalogued in chronological order. Usually, there are some business analyses performed on the data and the results stored for later references by the analytical department as well. Once they are stored in the warehouse, these data are usually not changed to preserve their consistency.

Besides chronological order, contextual orders are considered when building knowledge bases. Here, the entities are represented as derivatives of a top-level entity or a subsidiary entity thereof. Their relations are established more freely as they are meant to be updated and upgraded as they are used. They are established in the form of rules, based on entity properties. Hence, the form in which they are stored is somewhat different. Often, they are stored in the form of ontologies containing a deeper knowledge on the collected data. Similar

to storing query results in data warehouses, queries in knowledge bases are also stored for later use to render current results, as entities, relations and data instances change.

As opposed to databases and warehouses, which are application specific, knowledge bases may be application independent and are often used cross-domain by different applications. An example is presented in (Gumzej et. al., 2023).

3.4 Conclusion

In this chapter different aspects of data management in logistics have been dealt with. Apart from data representations and data storage standards, data organization and retrieval mechanisms have been presented. Finally, some common mistakes in automated data processing have been addressed to keep the conscious reader alert. Besides the listed examples, more can be discovered in the associated learning materials.

Reference Chapter 3

- Codd E.F. (1970). A relational model of data for large shared data banks. Communications. ACM 13, 6, pp. 377–387.
- Gumzej, R., Kramberger, T., Dujak, D. (2023). A knowledge base for strategic logistics planning, Proceedings of the 23rd International Scientific Conference Business Logistics in Modern Management: October 5-6, 2023, Osijek, Croatia, Dujak, Davor (ed.) Osijek: Josip Juraj Strossmayer University of Osijek, Faculty of Economics and Business, pp. 317-330. [available at: https://blmm-conference.com/past-issues/, access November 3rd, 2023]
- GS1 (2023). GS1 Standards. [available at: https://www.gs1.org/standards, access October 27th, 2023]
- Kent, W. (1983). A Simple Guide to Five Normal Forms in Relational Database Theory, Communications of the ACM, vol. 26, pp. 120-125.
- W3schools (2023). XML Tutorial [available at: https://www.w3schools.com/xml/, access November 3rd, 2023]
- W3schools (2023). JSON - Introduction [available at: https://www.w3schools.com/js/js_json_intro.asp, access November 3rd, 2023]

- W3schools (2023). Database Normalization [available at: https://www.w3schools.in/DBMS/database-normalization/, access December 7th, 2023]