Approaches to Organization of Change Tables for Real-Time Business Intelligence

Anastasiia Morozova, Liliia Bielova and Ievgen Meniailov

V.N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine

Abstract

This paper provides an in-depth analysis of the methodologies for organizing change tables in real-time Business Intelligence (BI), a critical aspect of modern, data-driven decision-making processes. The primary focus is on the seamless integration of Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) systems, emphasizing utilizing change tables to facilitate this integration. The study meticulously examines various strategies for creating and maintaining change tables, including using triggers, asynchronous triggers, and log data capture methods. Comprehensive evaluations of the advantages and drawbacks of each approach are presented, offering insights into their operational efficiencies and constraints within different database environments. This paper extends its analysis to empirical investigations, employing real-world database management solutions from industry leaders like Oracle and MS SQL Server. Through this empirical lens, the paper elucidates the practical implications of each method, thereby bridging the gap between theoretical understanding and real-world application. The outcome of this research is a nuanced understanding of change table management in real-time BI systems, providing valuable guidelines for businesses and practitioners looking to optimize their data-handling capabilities in rapidly changing market conditions. This study not only advances the academic discourse in database management but also serves as a pragmatic guide for implementing effective change table strategies in real-time BI scenarios.

Keywords

BI, OLTP, OLAP, ETL script, DML, Data Capture, change tables, trig- ger, asynchronous trigger, database log, Log-based Change Data Capture, Golden Gate, Log Mine1

1. Introduction

Now is the time of rapidly developing technologies and business processes. That's why business owners today want to know what's happening now. Software systems must allow to react on events in real time, as they occur, and not a minute or an hour later [1]. Organizations are revamping and automating processes for faster financial reporting, operational intelligence, and performance management [2]. One of the way to achieve such result is real-time Business Intelligence (BI). BI is actual direction nowadays [3]. The primary purpose of BI is to improve the quality of decisions while decreasing the time it takes to make them. Fresh information can support fast-paced, time-sensitive business processes, such as operational BI, real-time management dashboards, just-in-time inventory, high-yield manufacturing, facility monitoring, call center information delivery, self-service information portals, recommendations in e-commerce, and so on [4]. The use of BI explains by its capabilities that include.

What concerns real-time BI. Why real-time is such important? Real-time software solutions analyze and monitor business processes to give a wide range of users the real-time visibility they need to see a opportunity or problem, make a fully informed decision, and then follow more favorable way.

ProfIT AI 2023: 3rd International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2023), November 20–22, 2023, Waterloo, Canada

a.morozova@karazin.ua (A. Morozova); l.belova@karazin.ua (L. Bielova); evgenii.menyailov@gmail.com (I. Meniailov)

0000-0003-2143-7992 (A. Morozova); 0009-0007-0805-4547 (L. Bielova); 0000-0002-9440-8378 (I. Meniailov)

Why is there a need to adapt classical OLAP system (star or snowflake) as component of business solution to work in real-time mode? This is due to the scope of OLAP. Chains of stores, airline services, railway and other transport systems, high-load banking systems, financial markets and exchanges are key economic segments using OLAP in work [5]. In conditions of high market fullness, fast-growing data it becomes necessary to adapt the solution in real-time mode for making promising economic decisions and increasing work efficiency of business solution.

For example, Continental Airlines as popular high-loaded international airline service has taken a $30M investment in hardware, software, and personnel to generate over $500M in revenue enhancements and cost savings, resulting in a ROI of over 1,000 percent [3].

Key aims of numerous applications of real-time BI include [6]:

- Understand customer behavior in real time across multiple channels, such as Web, mobile, social, and enterprise applications. Consequently improve the customer experience as it's happening.
- Evaluate sales performance in real time.
- Resolve the problem of product recurring in abandoned shopping carts on an e-commerce website. That way, start a promotion to close more sales of that product before interest in it wanes.
- Prevent potentially fraudulent activity as it is being perpetrate. Identify and direct a new social media sentiment or pattern.
- Improve level of accuracy, efficiency, and customer service for logistics sphere.
- Monitor the performance of interconnected infrastructures such as computer networks and manufacturing facilities.

Thus adaptation of classical OLAP system to real-time BI is actual problem in the world of high-speed information technologies and fast-growing economic and business processes. This paper does not offer an innovative solution to organization of data capture processes in classical OLAP system. It considers and analyzes existing approaches to OLAP organization in real-time BI context.

2. OLTP and OLAP integration

Let's move to On-line Transaction Processing (OLTP) and On-line Analytical Processing (OLAP) integration. OLTP environments use database technology to transact and query against data, and support the daily operational needs of the business enterprise. OLTP is characterized by a large number of short on-line modified transactions. The main emphasis for OLTP systems is put on very fast query processing, maintaining data integrity in multi-access environments and an effectiveness measured by number of transactions per second.

OLAP is one of BI tools. OLAP environments use database technology to support analysis and mining of long data horizons, and to provide decision makers with a platform from which to generate decision making information [7].

OLTP systems are designed to quickly process individual sequences of transactions, while BI systems must have the data organized in a dimensional model to support querying, analysis, and "slicing and dicing" the data.

Look at the Figure 1 to understand the interaction between OLTP and OLAP.

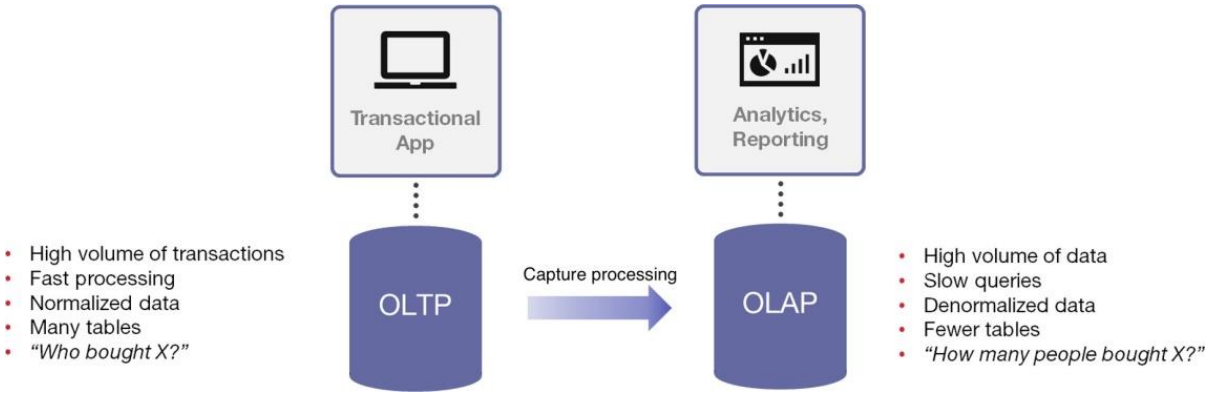

Figure 1: OLTP and OLAP integration

What allows integration between OLTP and OLAP? The answer to this question is using ETL scripts. ETL means three data management stages as: extraction, transformation and loading. ETL tools are used to:

- Extract data from homogeneous or heterogeneous data sources.
- Transform the data for storing it in proper format or structure for querying and analysis purpose.
- Load it into the final target (database, more specifically, operational data store, data mart, or data warehouse).

OLAP technology as part of BI system developed to enable fast extraction of requisite data from the data warehouse is populated from OLTP database through ETL tools. OLAP makes it available to carry out data modeling, perform business planning and forecasting, adopt budgeting and draw up financial reports. Owing to the system a user is able to conduct ad hoc analysis and apply 'what if' scenarios.

As for extract process we distinguish three main types of extraction: full extract, incremental extract and update notification [8].

Full extract is the most traditional approach. Performance is a key factor of system processing. As the demand for information grows, the load on your systems grows with it. If a report takes 5 minutes to run, that is putting 5 minutes of heavy load on your source systems, compromising any other users that are working at the time. Such way does not allow to perform ETL script in real-time mode because it can break functioning of highly loaded system.

And how does incremental approach work? At a specific point in time, only the data that has changed since a well-defined event back in history will be extracted. This event may be the last time of extraction or a more complex business event like the last booking day of a fiscal period. To identify this delta change there must be a possibility to identify all the changed information since this specific time event. Incremental extraction suggests using subsidiary change tables.

As it can be seen from schema in extraction process using two group of tables: source and change. Why is it necessary to have such table organization? Answer to this question is processing speed. Only imagine, in source tables there are 2 million records and only 100 another are added. At full extraction ETL script processes all the records of data storage, but in the same time change tables allows incremental extraction reading only new 100 records. Thus approach with using change tables significantly increases productivity.

Based on the fact of change tables efficiency different approaches to organization of change tables will be considered and analyzed further.

3. ETL Process Adaptation for Real-Time BI

This adaptation is based on integration and tracking of changes in OLPT database. The schema of ETL process adaptation for Real-Time BI is shown in Figure 2. These changes will be saved as Change Tables.

The independence among all the structures is based on their organization. For instance, all of the connectors are presented in form of interfaces. So as for different databases and storage their realizations can vary.

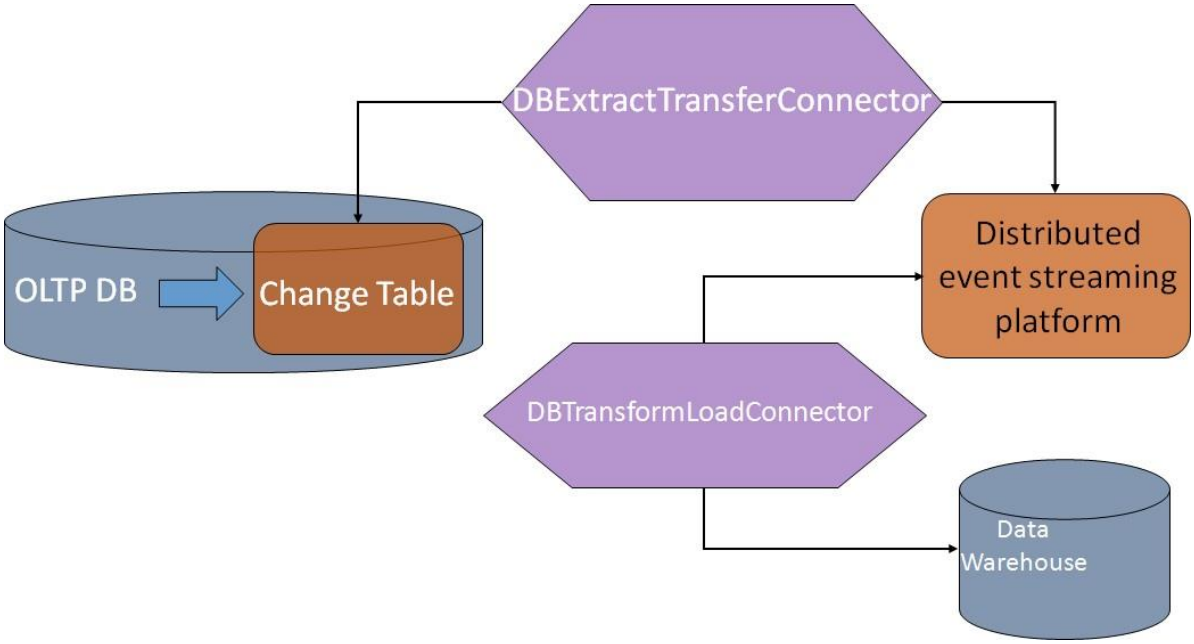

Figure 2: OLTP and OLAP integration

4. Approaches to processing change tables

There are several different approaches to extraction data into change tables. For a start let'sfocus on trigger solution.

4.1. Trigger

There is significant segment of extraction solutions based on trigger. This is due to the fact that trigger based extraction is enough simple to implement and it doesn't require any changes to application in source system. A trigger is a special kind of procedural code programmed to fire when a data manipulation language (DML) event occurs in the database server. As for DML events they include UPDATE, INSERT, or DELETE statements issued against a table [9]. There are three types of DML triggers to audit information changes such as: before-data, after-data and schedule. According to type name before-data trigger fires before the data set is executed and after trigger captures information after the data engine executes all data sets. These types of trigger check event and run associated code when the event occurs. Such approach allows keeping real-time processing [10]. In the same time schedule trigger doesn't provide such flexibility because it is executed at the time a report job is scheduled to run. As you can see trigger approach is not complicated, reasonable and understandable. It is important to note universality and autonomy of trigger technique:

- Not necessary additional special software or license because any database management system includes trigger solution;
- Simply to modify data capture strategy;
- Possible to create a separate trigger for each of the existing operations;
- Uncomplicated introduction to pre-existing functioning system.

And what disadvantages does such solution include? Trigger technique can affect performance on the source system, and this impact must be carefully considered before implementation on a production source system. For performing data capture manipulation

database triggers are used in shadow tables. The shadow tables may store the entire row to keep track of every single column change, or only the primary key is stored as well as the operation type (insert, update or delete). However, a few challenges arising in data capture process should be noted:

- Firing the trigger, and storing the row changes in a shadow table, introduces overhead.
- In an extreme case CDC may introduce 100% overhead on the transaction i.e. instead of 0.1 second it may take 0.2 seconds to complete a transaction. In the same time it must be emphasized that transferring data to change tables is also a costly operation.
- If changes are made to tables then triggers and shadow tables may also have to be modified, recreated and/or recompiled which introduces extra overhead to manage and maintain the database.
- Inefficiency of trigger in capture of updating data. For example, you're interested in change of one row of table, but definite record is modified in another row. Trigger fires by any change in record. Therefore, data capture performs in any cases of updating. Just think about it: 500 records with not interested for you attributes are updated and then captured. It decreases the efficiency of system.

4.2. Asynchronous trigger

The distinctive property of synchronous trigger is that original DML statement which caused the trigger to fire will not be released until the trigger itself is complete. If the trigger implements some complex logic, it can take enough long a while to execute. Such trigger can cause a delay even when executed simple DML operations. That's why there is need for asynchronous in the performing. Work of asynchronous trigger is specific and complex. In case of asynchronous trigger the trigger doesn't actually perform the requested work. Instead, it creates a message that contains information about the work to be done and sends this message to a service that performs the work. Then the trigger returns. Thus the program implementing the service performs the work in a separate transaction. By performing this work in a separate transaction, the original transaction can commit immediately. The application avoids system slowdowns that result from keeping the original transaction open while performing the work. It must be emphasized that asynchronous trigger allows more opportunities than synchronous one:
Supporting not only DML capture events and also data definition language (DDL) ones;
Capture level expands to global and schema;
The complicity of rules associated with the capture doesn't influence decreasing system performance consequently rules can be either simple or complex.

As for capture processes of asynchronous trigger they have some advanced abilities that are not available with synchronous one: combined and downstream capture. The key disadvantage of approach is the fact that only enterprise version provides asynchronous trigger therefore the solution is not publicly available in usage.

4.3. Log-based Change Data Capture

There is transaction log in transaction databases that stores all changes in order to recover the committed state of the database should the database crash for whatever reason. Log-based Change Data Capture (CDC) takes advantage of this aspect of the transaction database to read the changes from the log. To understand the principle of the log-based CDC approach, look at the Figure 3.

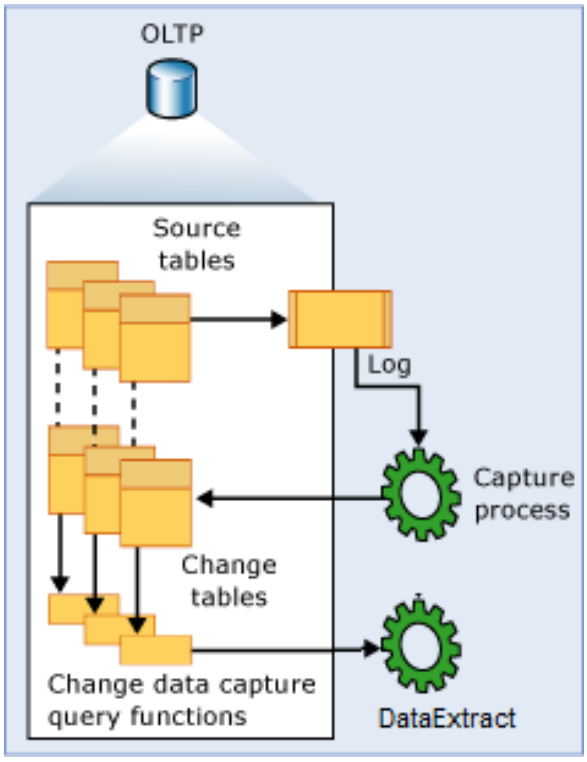

Figure 3: Log-based Change Data Capture

What are the benefits of this approach? Among the most significant:

1. Minimal performance impact on the master database that allows to use in systems with extremely high transaction volumes;
2. It also requires no change to tables or the application;
3. The asynchronous nature of CDC: changes are captured independent of the source application performing the changes.
4. There are also several characteristics making the usage of log-based CDC approach quite challenging. Such as:
5. Transaction log formats are proprietary to each database vendor hence there are no documented standards on how the changes are stored and interpreting the changes in the transaction log is difficult;
6. Many database vendors lack sufficient documentation because they consider the transaction log formats to be an internal format;
7. Database vendors may not provide an interface to the transaction logs – documented or not – and even if there is one it may be relatively slow and/or resource intensive;
8. Most databases have been optimized to only use internal identifiers to recover database row changes which is insufficient to perform data capture and record the changes on a different system;
9. Most, if not all, transaction log formats are in a binary format.

5. Results of Testing

All suggested approaches to organize of change tables have been tested. This section contains some scripts and testing results. First, we are going to describe the test environment. We run our test on workstation with 64 GB RAM and 8 CPUs. We've created test table and run tests for two DBMS: SQL Server and Oracle. Test table contains simple data types like int, spatial data type and LOB data type.

We've reproduced OLPT load with short DML operations (INSERT/DELETE/UPDATE) and measured performance before and after implementation corresponding approaches.

5.1. Step 1. Preliminary Test

First, we have run DML tests and measured performance without any changes. The results are shown in Table 1.

Table 1
Preliminary DML test results.

Number of records	Time in sec (SQL Server)	Time in sec (Oracle)
100,000	20	22
200,000	37	51
1,000,000	192	215

5.2. Step 2. Synchronous Trigger

This subsection contains results of using synchronous trigger to populate change tables. First, we need to create change table for every table we want to monitor. This table can have the same structure as a source table or contains less or extra columns. For our tests we create change table that has the same structure as a corresponding source table and contains two extra columns to get code of DML operation (1 for DELETE, 2 for INSERT, 3 and 4 for UPDATE) and time of change.

At the next step we've created DML trigger to capture DML changes from source table into change table.

We reproduced the same workload as in the Step 1. The results are shown in shown in Table 2.

Table 2
Synchronous Trigger results.

Number of records	Time in sec (SQL Server)	Time in sec (Oracle)
100,000	42	46
200,000	71	64
1,000,000	360	478

5.3. Step 3. Autonomous Trigger

This subsection describes approach to use Autonomous Trigger for gathering information into change tables. Oracle and SQL Server have difference approach to creation of autonomous trigger.

Oracle uses AUTONOMOUS_TRANSACTION pragma inside code of trigger. The AUTONOMOUS_ TRANSACTION pragma changes the way a subprogram works within a transaction. A sub- program marked with this pragma can do SQL operations and commit or roll back those operations, without committing or rolling back the data in the main transaction [11]. To create autonomous trigger for Oracle, just add PRAGMA AUTONOMOUS_TRANSACTION; as a first statement inside trigger body.

Autonomous Trigger for SQL Server is based on Service Broker. First, we should create service broker objects such as MESSAGE, Response and Request Queue, Receiving Service, Sending Service, procedure to send items to the queue for asynchronous trigger and procedure which processes items and adds modified data into change table. We aren't going to listing all scripts to create service broker objects just show the results. The results of testing Autonomous Trigger are shown in Table 3.

Table 3
Autonomous Trigger results.

Number of records	Time in sec (SQL Server)	Time in sec (Oracle)
100,000	96	54
200,000	190	112
1,000,000	900	558

5.4. Step 4. Log Data Capture

This subsection contains the results of the last approach we tested. Log Data Capture approach is based on reading changes from database log file. To track data changes SQL Server provides Change Data Capture (CDC) feature. Change Data Capture is deprecated in Oracle Database 12c, so we haven't tested it.

For SQL Server it took 230 sec for INSERT/DELETE/UPDATE 1,000,000 rows. CDC runs background process to collect changes into "change table". It took about 18 min to finish this background process. CDC in SQL Server has some peculiar properties for collecting LOB data types such geometry, geography, varbinary(max), varchar(max) or nvarchar(max). Also you can configure Capture and Cleanup jobs to change performance of CDC. This paper omit all this configurations and uses its default values.

Unfortunately, most Oracle solutions have only paid license that's why they weren't tested.

Oracle provides log data capture decision named Golden Gate. Its modular architecture gives the flexibility to extract and replicate selected data records, transaction changes, and changes to DDL across a variety of topology. By this cause Oracle Golden Gate can allow to support numerous business requirements such as: business continuance and high availability, initial load and database migration, data integration and decision support.

It should be noted there is more powerful log-based tool developed by Oracle as Log Miner. Log Miner directly accesses the Oracle redo logs, which are complete records of all activities performed on the database. That's why this solution provides more advanced functionality. Log Miner supports such important and flexible possibilities as: database recovery operations (even fine-grained recovery at the transaction level), auditing of DML statements, the order in which transactions were committed. Usage of such features allows to improve system efficiency, rollback logical data corruptions or user errors, tune performance and capacity planning through trend analysis.

6. Extract Transform phase

This section contains the structure for Extract Transform phase of the adapted ETL process for real time. The purpose of the developed algorithm is primarily the preparation and transmission of information using ETL-technology. The data received by the system will be removed from the relational database, regardless of its own structure and features. In this way, we can ensure the flexibility of the system to expand, use and implement in different environments without being tied to corporate identity.

Based on the analyzed data, it was decided that the most complex and interesting process that requires detailed improvement and development of a new architecture is the process of extracting data from storage, converting them into a specialized form and sending them to a distributed message queue.

At the moment, there are many paid platforms with the implementation of data analysis, in which ETL technology is only an integral part. However, they cannot and do not allow the transfer of information and do not provide access to verify the architecture of their application, so it is possible that leaks of corporate data may occur very often.

In order to track incremental data, it will be advantageous to use and take as a basis for the architecture of the CDC.

The first step of our algorithm will be to create a flow in which the distribution of the processes of extraction, processing and transportation of data by tables will be organized.

After that, threads are created that implement these actions. Each of them first receives the changed information.

The next step is to adjust the data to the format that best suits the distributed message queue.

From the beginning, all start tables are placed in a queue synchronized with all threads. Before processing, the table is taken out of it, the number of threads will increase. After all operations and data transportation, the table is added to the end of this queue and frees up space for new processing. This method was chosen because the processing time of tables of different sizes will not be the same, which would complicate the operation of the system in real time. This process is endless, as the analytical platform needs to constantly receive fresh data for the most effective recommendations for tactical changes in information. An important feature is the storage of the current transaction that has been delivered to the system. In the event of unexpected platform disruptions or unexpected errors, the system needs to develop a repository to allow for instant resumption. The Sequence Diagram and Class Diagram of suggested approach are shown in Figure 4 and Figure 5.

Conclusion

We tested different solutions to get incremental data into change tables for two database vendors (SQL Server, Oracle). All suggested approaches have overhead for DML operations. We summarize results of all tested approaches for 1,000,000 rows into Table 4.

Change data capture using database triggers lowers the overhead to extract the changes but increases the overhead to record the changes. The disadvantage is the limited scalability and performance of trigger procedures, making them optimal for use cases with light to medium loads. It should be noted that the performance of the system is also affected by the complexity of rules associated with the capture for trigger approach. Although log-based CDC is generally considered the superior approach to data capture that can be applied to all possible scenarios including systems with extremely high transaction volumes it is enough complex in implementation by cause of the lack of document standards. Thus it is impossible to single out the best approach that can be suitable in all cases. For right choice of data capture approach you need to consider the significant characteristics of systems influenced its performance.

Table 4
Autonomous Trigger results.

Approach	Time in sec (SQL Server)	Time in sec (Oracle)
no changes	192	215
synchronous trigger	360	478
synchronous trigger	900	558
CDC	230	-

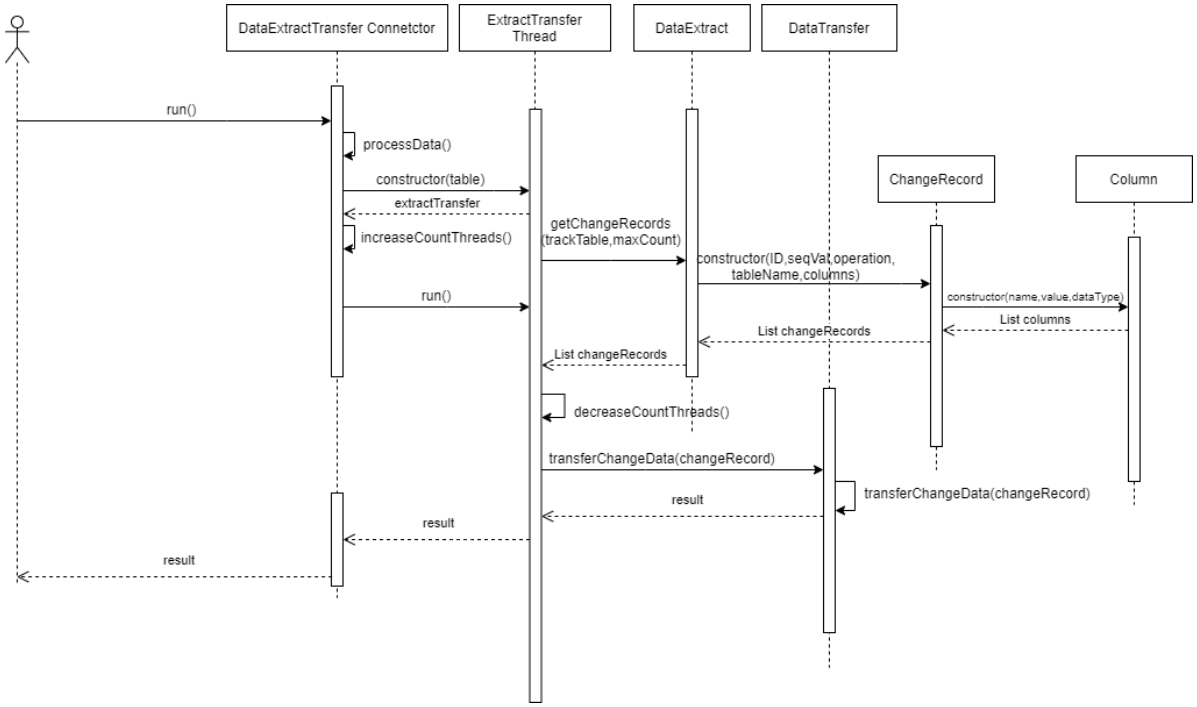

Figure 4: Sequence diagram

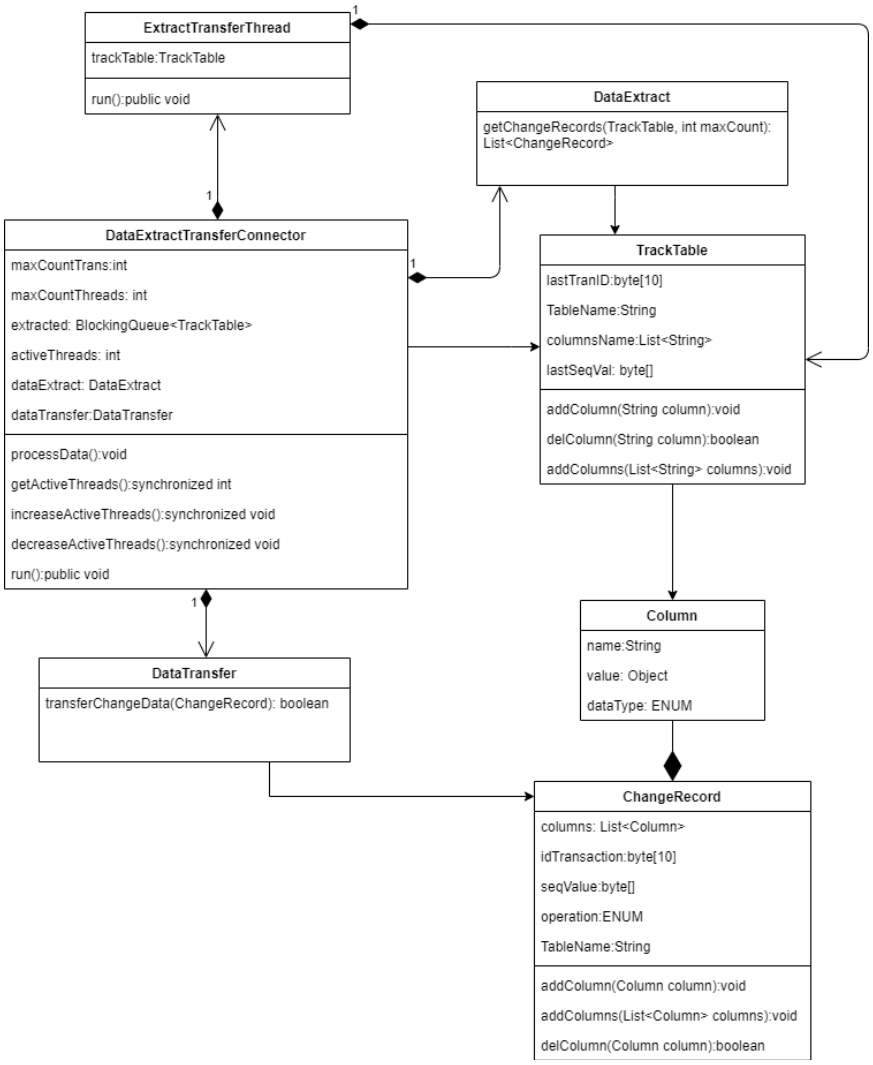

Figure 5: System class diagram

References

[1] Dotsenko, N, Chumachenko, D., Chumachenko, I. Modeling of the processes of stakeholder involvement in command management in a multi-project environment, International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2018, vol. 1, pp. 29-32. doi: 10.1109/STC-CSIT.2018.8526613

[2] Bazilevych, K., et al., Stochastic modelling of cash flow for personal insurance fund using the cloud data storage, International Journal of Computing, 2018, 17 (3), pp. 153-162.

[3] Shabanov, D., et al., Simulation as a Method for Asymptotic System Behavior Identification (e.g. Water Frog Hemiclonal Population Systems), *Communications in Computer and Information Science*, vol. 1175, 2020, pp. 392-414. doi: 10.1007/978-3-030-39459-2_18

[4] Dhaouadi A, Bousselmi K, Gammoudi MM, Monnet S, Hammoudi S., Data Warehousing Process Modeling from Classical Approaches to New Trends: Main Features and Comparisons. *Data*. 2022; 7(8):113. doi: 10.3390/data7080113

[5] Dotsenko, N., Chumachenko, D., Chumachenko, I., Project-oriented management of adaptive teams' formation resources in multi-project environment, CEUR Workshop Proceedings, 2019, 2353, pp. 911-923.

[6] Luo, X.(R). and Chang, F.-K., Toward a design theory of strategic enterprise management business intelligence (SEMBI) capability maturity model, Journal of Electronic Business & Digital Economics,2023. doi: 10.1108/JEBDE-11-2022-0041

[7] Naseema Shaikl, Dr. Wali Ullah, Dr. G. Pradeepni, OLAP Mining Rules: Association of OLAP with Data Mining, *American Journal of Engineering Research (AJER),* Volume 5, Issue 2, 2016, pp. 237–240.

[8] Seenivasan, Dhamotharan, ETL for Data Warehousing, 2023. URL: https://www.researchgate.net/publication/368300555_ETL_for_Data_Warehousing

[9] Audit SQL databases using a SQL Server trigger tool, 2020. URL: http:// https://solutioncenter.apexsql.com/audit-sql-databases-using-the-sql-server-trigger-tool.

[10] Chumachenko, D., On intelligent multiagent approach to viral Hepatitis B epidemic processes simulation, Proceedings of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP 2018, 2018, pp. 415-419. doi: 10.1109/DSMP.2018.8478602

[11] Oracle. Oracle Database Documentation, 2018. URL: https://docs.oracle.com/en/database/oracle/oracle-database/index.html

[12] David Arnott, Felix Lizama, Yutong Song, Patterns of business intelligence systems use in organizations,Decision Support Systems, Volume 97, 2017,Pages 58-68. doi:10.1016/j.dss.2017.03.005.