Experimental Comparison of Syntax and Semantics of DBS Oracle and MySQL

Michal Kvet
University of Zilina in Zilina
Zilina, Slovakia
Michal.Kvet@fri.uniza.sk

Lucia Fidesová, Karol Matiaško
University of Zilina in Zilina
Zilina, Slovakia

***Abstract*—Current information systems must deal with complex data management with regards on evolution over the time. There are several database systems with different data modelling principles, data handling and management, however concept of relational database is still most popular. The aim of this paper is to analyze and experimentally compare database systems Oracle and MySQL emphasizing the differences in syntax, semantics and performance characteristics. It consists of analysis of existing database systems, comparison of syntactic and semantic differences with regards on developed application functionality to transform commands to another system. Experimental part compares time consumption based on several characteristics.**

I. INTRODUCTION

Development of information systems and user requirements have brought the necessity for developing methods for data processing and storing. In the past, data were embedded in the program, so is was created as direct and inseparable part of the application. With the need to process large data amounts, there were significant challenges with regards on storage management, efficiency, fast processing and reliability. The solution of such problems is based on separating application logic from the physical data storage. Current information systems put great emphasis on speed, efficiency and low cost of the processing, so these systems begun to use database systems for complex data management. Nowadays, we can feel strict requirements for data processing in the whole time spectrum, data characteristics and methods evolve over the time, however current paradigm of database systems is still characterized by actual object states processing, which is inadequate [3], [8]. Thus, temporal characteristics have not been covered completely yet.

There are several types of database systems capable of storing, modifying and selecting large data amounts. The most commonly used database systems are still based on the relational data model. This group covers large database administration by products like Oracle, MySQL, PostgreSQL, Informix of DB2. Although all of them are based on SQL standard, each of them uses its own dialect. The differences between them can be therefore observed by syntax or semantic characteristics, but mostly by the performance of various systems influenced by the background processes and complex architecture.

Data stored in the database can be found in the vast majority of systems, in transport systems [9], industry [10], medicine or even web sites managing e-shops, photos and so on. For complex data and conditions evaluations, input data are the key part [12]. Many factors can influence access to process optimization or decision making [6], [7] for creating prognoses, therefore such data must be stored effectively, time to get reliable data must be shortened to the minimal possible value.

In this paper, we compare syntax and semantics of the two database systems – Oracle and MySQL, which are nowadays developed by the same company and will highlight the performance limitations and definitions, whereas one of them is used for commercial purposes, so the task is, how it influences performance.

II. ANALYSIS

Database system (DBS) can be understood as a set of interrelated data together with software tools that allow access to data stored in the physical layer [12]. However, users themselves do not have direct access to the physical data, to the datafiles, which cover defined data and structures. Such requirement is provided by the database system internal background processes, which ensure reliability, consistency and transfer of relevant data to the processing memory for consecutive processing. Logical layer is defined by the tablespace, which defines the locality of the physical data storage [11]. Thus, database system is composed by two groups – database (DB) and database management system (DMS).

Data can have various characteristics, which can be processed using available technical means. When dealing with the data processing concept, we mean persistency – data, which will be available even after shutting down the application. On the other hand, database management system is created by the program tools, which are intended for data definition, manipulation and protection covered by the access rules, consistency and transaction management.

III. HISTORY OF DATABASE SYSTEMS

Database systems, based on development, architectures and processing style can be divided into five following types [1], [14]:

- File systems. The development of operating systems has also brought the proposing techniques of files, which are processed and managed as part of the OS. It was necessary to define data directly, which were part of such application programs. File systems themselves have several disadvantages, such as no parallelism data

access support, no controlled redundancy or even consistency.

- Hierarchical and network based DBS. To address the weaknesses of file systems, first database management systems have been developed to ensure the independence of data from user programs. Moreover, it can provide data management using row level and backup planning. In hierarchical system, data model is modelled using the tree set providing depth and width scanning. On the other hand, in network based DBS, relationships are represented and modelled by the pointers. The main disadvantage of such defined system is the consistency problem, if any change of the data model is provided.
- Relational DBS is still most often used database concept. Theoretical part has been proposed in 70ties by the mathematician Edgar Codd. Processing is based on tables, relationships reducing data duplications and ensuring consistency. The root part is the relational algebra with several operations, which can be combined to get desired data. It uses structured query language – SQL. Thanks to that, as already mentioned, defined solution is independent from physical data location and structure [12].
- Object oriented DBS increases the level of abstraction, the principle is based on object oriented modelling and programming. However, this concept is not used very often, now.
- NoSQL is a new concept for processing wide data scale without direct table schema definition, even with no relationships. Data can be stored in various structures influencing performance and algorithmic complexity [15].

IV. MySQL vs. Oracle

Oracle database system is developed for wide range and wide amount of commercial data. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All requirements are accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery. It supports multiple platforms and operating systems including the most widely used Linux and MS Windows. As MySQL, also Oracle database system is developed by the Oracle company. For the purposes of this paper, we will use following version – Oracle Database 11g

Express Edition 11.2.0.2.0. Such information can be provided by executing following query. It also provides information about server OS [1], [4]:

```
select * from v$version;
```

Vice versa, MySQL is one of the most favourite open-source DBS mostly provided by web applications. As with Oracle DBS, we can also find significant support for Linux, Windows, Solaris, ... Users can select the best version based on several released editions. MySQL Community Edition is freely available for download on the official site of MySQL and is supported by a wide community of developers. In this paper, we will deal with MySQL Community Edition 5.7.9 version [13].

When dealing with MySQL, it is worth to mention also MariaDB community, which has been founded by the original MySQL developers. They maintain high compatibility of MariaDB with MySQL [13].

V. Benchmark data model

Next chapters focus on the comparison of syntactic and semantic differences in SQL statements, when dealing with both databases. While defined DBS are being developed by the same company, each of them is intended for another usage and has its own dialect, thus the syntax and functionality is different. All these factors have partial impact on definition, management and in final stage - performance.

Whereas there is necessity to execute multiple statements for the analysis, to declare performance characteristics and limitations, our experiments are based on *aircraft data model*, which is part of our benchmark model. The *l_flight_ticket* table has more than 2 millions of rows. Cardinality of the table *l_person* is 100 rows, we have 100 employees in the *l_employee* table. We have almost 40 000 flights stored. Although it can be considered limited and insufficient, as we will see in the experiment part, even this solution provides reliable and significant performance changes and can show the limitations of the system. Fig. 1 shows the significant part of the benchmark data model, which can be also downloaded – *benchmark aircraft* [http://frdsa.fri.uniza.sk/~kvet/benchmark_aircraft.zip]. Provided benchmark data model consists of 11 tables and can be divided into several modelling parts:

- Persons – are modelled using *l_person* table and *l_employee* reflecting employees of the air company. Each person is identified by composite primary key (*id_card, card_type*). This category consists also of *l_air_company* management.
- Flights management, which is modelled by *l_flight* and M:N association between flights and persons – *l_flight_ticket* table. It is also categorized by the flight classes – *l_class* table.
- Technical data – plane type definition and association to the specific flight – *l_plane, l_plane_type.*
- Positional data – *l_country, l_town, l_airport.*

For the purposes of this paper, we will mostly deal with *l_person, l_flight_ticket* and *l_flight* table (Fig. 1). Partially, we will access *l_airport* table to get departure and arrival airport modelled by two non-identifying 1:N relationships.

VI. Syntax and semantics comparison

When dealing with multiple database systems, it is necessary to identify equivalents for particular data types. Nowadays, there are wide range of sources for describing equivalents, however, many of them do not determine them correctly based

on structure, precision and size. Some data types have more than one equivalents. This chapter describes the basic data types of the defined and compared DBS [1], [14], [17].

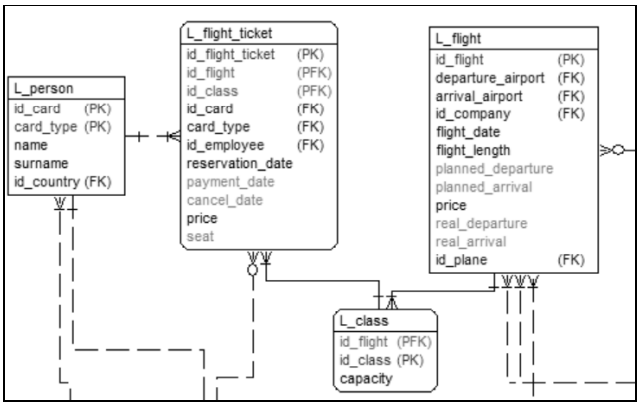

Fig. 1. Benchmark data model

A. Numeric data types

MySQL supports all of the numeric SQL standard data types: integer types, data types with fixed and floating precision point.

MySQL integer types include INTEGER (or INT) and SMALLINT, but also supported types are TINYINT, MEDIUMINT and BIGINT.

Table I. describes the different data types: the allocation requirements, range, minimum and maximum values, which may take.

TABLE I. MYSQL INTEGER DATA TYPES

Data type	Allocation (bytes)	Minimal value With(out) sign	Maximal value With(out) sign
TINYINT	1	-128	127
		0	255
SMALLINT	3	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INTEGER (INT)	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-922337203 6854775808	92233720368 54775807
		0	184467440737 09551615

Data types with fixed precision point is used, when it is necessary to keep exact numerical values. This category contains types DECIMAL and NUMERIC. In MySQL, NUMERIC type is represented as DECIMAL. Type DECIMAL (m, n) - "m" represents the total number of digits and "n" refers to how many of these numbers will be used to represent the decimal part, DECIMAL (m, 0) to replace the equivalent of DECIMAL (m). The maximum number of digits for the numeric type is set for 65.

Data types with floating point represent types characterized as FLOAT and DOUBLE. Four bytes are allocated for single-precision (0-23 decimal places), for double precision (0-53 decimal places) - eight bytes are allocated. If the number contains more decimal places than defined, such value is rounded to the decimals specified in the definition type. As a consequence, such value is approximate and therefore poses problems to accuracy for comparing.

Numeric data types in the DBS allow Oracle to store positive and negative numbers, fixed and floating precision point. Data type NUMBER allows you to represent whatever value (fixed, floating point) from the range with the 38 digits of precision – positive numbers range is $1*10^{-130}$ – $9,99...9*10^{125}$, negative numbers from $-1*10^{-130}$ – $9,99...9*10^{125}$ (for 38 precision digits). It is also possible to use INTEGER data types, which is, however, automatically converted to NUMBER(38). Moreover, Oracle can use also two data types for storing numbers with floating point: BINARY_FLOAT and BINARY_DOUBLE. These types use binary precision, which enables faster arithmetic calculations and have less memory requirements.

B. Date and time data types

1) MySQL

The proposed data types, which can represent time or date are DATE, DATETIME and TIMESTAMP.

DATE data type is used for values, that do not contain time spectrum. Supported range is from 01.01.1000 to 31.12.9999 and is influenced by the server format settings. DATETIME and TIMESTAMP types are used to represent date, but also time. The output format is similar to the type DATE by adding a section representing the time: 'YYYY-MM-DD HH: MM: SS'. For both data types, automatic conversion to the current time can be done. Unlike DATETIME, TIMESTAMP values allows automatic conversion with regards to the set time zone that is server set. Invalid values are converted to zero ('0000-00-00', or '0000-00-00 00:00:00'), exception is raised. MySQL represents the date and time as a character string and allows to delimit pieces of the date and time between the parts by using any character.

2) Oracle

DBS Oracle uses two types to represent the date and time format - DATE and TIMESTAMP. Values of these types can be also automatically converted. Both values contain not only date, but also time, which is, however, very often main mistake, when comparing values. Moreover, TIMESTAMP allows you to save fractions of seconds (11 bytes) and time zone (13 bytes). Values in DATE format requires 7 bytes.

C. String data types

1) MySQL

Data types CHAR and VARCHAR are similar, they have declared the maximum number of characters that can be stored. They differ in the method of storing, and the length. Length of the data type CHAR may range from 0 to 255. Variable length VARCHAR type is in the range 0 - 65536 (effective MySQL version 5.0.3). To allocate memory for VARCH.AR, 1 or 2 bytes are used to store the length (number of bytes of data storage) and then, there are the data

themselves. The table below provides a practical example of limitations - when to use CHAR and VARCHAR (Table II.).

TABLE II. STRING SIZE CHARACTERISTICS

Input data	CHAR(4)	Size	VARCHAR(4)	Size
‘’	‘’	4 bytes	‘’	1 byte
‘ab’	‘ab’	4 bytes	‘ab’	3 bytes
‘abcd’	‘abcd’	4 bytes	‘abcd’	5 bytes
‘abcdef’	‘abcd’	4 bytes	‘abcd’	5 bytes

BLOB (Binary Large Object) may include various data that are represented as binary strings. There are 4 types: TINYBLOB, BLOB, MEDIUMBLOB and LONGBLOB. They differ in the maximum length of the values they may contain. TEXT values are strings of characters and has 4 types: TINYTEXT, TEXT, MEDIUMTEXT and LONGTEXT. If you try to insert BLOB or TEXT data into the database exceeding the maximum defined length, alert is generated and data are truncated.

2)Oracle

The CHAR data type is used for fixed-length strings. It is necessary to specify a length ranging from 1 to 2000 characters (or bytes). For storing variable length strings VARCHAR2 is used, respectively, for backward compatibility with earlier versions can be used VARCHAR. Size (length) may not exceed 4,000 characters (bytes). If no size is defined, default value 1 is used. BLOB and CLOB types can store up to 128 terabytes of data. BLOB stores them in binary form and CLOB as a characters.

Based on the definition, Table III. and Table IV. show the equivalents for Oracle and MySQL database system. Table III. characterizes numerical datatypes, Table IV. compares date and time formats.

TABLE III. NUMERICAL EQUIVALENTS

MySQL	Size (in bytes)	Oracle
TINYINT	1	NUMBER(3,0)
SMALLINT	2	NUMBER(5,0)
MEDIUMINT	3	NUMBER(7,0)
INTEGER (INT)	4	NUMBER(10,0)
BIGINT	8	NUMBER(19,0)
DOUBLE	8	BINARY_FLOAT BINARY_DOUBLE
FLOAT(x<=24)	4	BINARY_FLOAT
FLOAT(25<=x<=53	8	BINARY_FLOAT
DECIMAL, REAL	8	BINARY_FLOAT

TABLE IV. TIME ATTRIBUTE EQUIVALENTS

MySQL	Size (in bytes)	Oracle
DATE	3	DATE
DATETIME	8	DATE
TIMESTAMP	4	DATE, TIMESTAMP

VII. SQL

Structured query language (SQL) is the most widely used database language. This language belongs to the non-procedural languages and describes what is required from the database, but does not specify how it is performed. It can be divided into four groups [12], [14]:

A. Data definition language (DDL)

With commands (CREATE, ALTER, DROP, TRUNCATE) that belong to this group, we can create, drop and modify database objects, like tables, views, schemas, domains, or integrity constraints. In Oracle, it is allowed to use CREATE OR REPLACE clause type for some objects (e.g. views, procedures, functions). It means, that object is created regardless the status of existing one, thus if it exists, system automatically redefines it. MySQL does not support this syntax. In case you want to create an object, first run the command DROP [IF EXISTS] - the object is deleted, if already defined in the database, and subsequently CREATE command must be executed. CREATE OR REPLACE syntax cannot be used to define tables at all. Changes are performed by ALTER command in that case.

TRUNCATE command is used when we want to erase all data from the table. The syntax is the same in both DBS, however in MySQL database, it is not necessary to use the keyword "TABLE" in the database such as Oracle does.

Using the ALTER command, it is possible to create primary and foreign key in an existing table, add column to the table, or edit or delete the column. This command has four options:

- ADD,
- MODIFY,
- DROP,
- RENAME.

With the keyword RENAME, user usually wants to rename a table or a particular column. However, this command works only in the DBS Oracle. In MySQL, it can be used only to rename a table, renaming column must use performed by using the keyword CHANGE. Also data type must be specified regardless the change.

B. Data manipulation language (DML)

This type defines a database query. DML category commands allow inserting, modifying and deleting data.

INSERT, DELETE, and UPDATE statements can be used only for one table. However, these commands can be combined with a SELECT statements, so a set of data can be managed even based on another table data conditions. In SELECT statements, several tables can be listed and joined after the FROM clause, which is not possible to be done in another DML statements.

If we want to add new data to the column that has data type DATE, TIMESTAMP, or DATETIME, it is necessary to use conversion function from the string. For this purpose, DBS

Oracle uses function TO_DATE, MySQL provides STR_TO_DATE function. Vice versa, when transforming date value to the string, DBS Oracle uses TO_CHAR function, DATE_FORMAT is MySQL equivalent.

Select statements are mostly affected by the syntax specifics. All requirements can be rewritten to another syntax, however, DBS Oracle provides significantly larger amount of possibilities. Transformation of particular values can be done in MySQL using CASE functionality, which is naturally available for Oracle DBS, but there is also DECODE functionality and NVL for dealing with undefined (NULL) values.

```
-- Oracle
Select name, surname, id_flight_ticket,
        decode(id_class, 1, 'premium',
                         2, 'business', 3, 'economy',
                         'special')
  from l_person
    join l_flight_ticket using(id_card, card_type);
```

```
-- MySQL
Select name, surname, id_flight_ticket,
  Case id_class
    when 1 then 'premium'
    when 2 then 'business'
    when 3 then 'economy',
    else 'special'
  end)
  from l_person
    join l_flight_ticket using(id_card, card_type);
```

When using a set, which has been created by the Select statement and such set is then processed by another Select statement, MySQL requires to use naming convensions like standard tables provided by aliases. For the simplicity, we will name only solution for MySQL. Oracle can deal with it without alias definition necessity.

```
-- MySQL
Select name, surname
  from
(Select name, surname, count(*) num_fl
  from l_person
    join l_flight_ticket using(id_card, card_type)
  group by name, surname, id_card, card_type)
    tempTable
where num_fl > 10
  order by surname, name;
```

String concatenation is managed by the CONCAT function, which can manage only 2 strings as parameters. Thus, if more strings must be grouped together, multiple CONCAT functions are necessary to be used. Oracle DBS simplifies it by pipe definition (||). Following code shows the principle of concatenating 5 strings – name, surname, flight ticket identifier, departure and arrival destination.

```
– Oracle
select name ||' '|| surname ||' '|| id_flight_ticket
        ||' '|| dept.airport_name
        ||' '|| arr.airport_name)
…
```

```
– MySQL
select concat(concat(concat(concat(
        concat(concat(concat(concat(name, ' '),
        surname), ' '),id_flight_ticket),' '),
        dept.airport_name),' '),
        arr.airport_name)
from l_person
  join l_flight_ticket using(id_card, card_type)
  join l_flight using(id_flight)
  join l_airport dept on
    (l_flight.departure_airport=dept.id_airport)
  join l_airport arr on
    (l_flight.arrival_airport=arr.id_airport);
```

Result of two dates substracting is the difference expressed in day granularity. Whereas there is no special automatic converstion in MySQL, explicit call of the DATEDIFF function must be done. Oracle calculates it automatically. Adding positive value to the date increases the number of days. If the number of days in a given month is exceeded, it is automatically reflected to the following month or year.

ADD_MONTHS function, which parameters are date and number to shift the defined number of months is used to correct month management. If the second parameter is positive, number of months is added, if not, defined number is substracted with regards to leap years and complex date and year management.

MySQL handles dates using DATE_ADD and DATE_SUB feature based on defined interval as a parameter (e.g. DAY, WEEK, MONTH or YEAR).

C. Data control language (DCL)

DCL ensures access control to data, security.

This category contains SQL commands for granting (GRANT) and revoking (REVOKE) access rights to database objects such as tables or views. Using keywords GRANT OPTION and ADMIN OPTION can be assigned to declare possibility for particular user to allocate rights to another user. The difference is based on roles and categories of rights. Moreover, if the privilege is removed from the user with ADMIN OPTION rights, privileges defined by him are not removed, which does not apply to GRANT OPTION (rights are managed in cascade way in this type). There is no difference of individual commands for DBS Oracle and MySQL, syntax is the same.

D. Transaction control language (TCL)

TCL is also well known as *Data Integrity Statements* (DIS) providing commands for transaction management. It also ensures correctness and data security.

SQL transaction is defined as a sequence of operations including database commands that are considered as atomic in the terms of data recovery.

The transaction begins with the first SQL command after logging, or directly after the end of the previous transaction. Transaction ends with the ROLLBACK statement, which returns database to the state before start of the transaction. Vice versa, COMMIT statement ensures, that changes made in the transaction become permanent. Both mentioned database systems allow AUTOCOMMIT functionality definition, thus each statement automatically reaches COMMIT. However, significant position has DDL statement in transaction processing. In Oracle, each DDL statement automatically executes COMMIT, thus transaction is ended successfully. However, DDL statement is MySQL does not influence processed transaction. Syntax of the statements is also a bit different, however, functionality is the same.

E. Triggers and views

When creating triggers, as well as procedures and functions, syntax CREATE OR REPLACE keyword can be used in Oracle DBS. In MySQL, command DROP [IF EXISTS], followed by the CREATE must be executed.

Before creating the trigger in MySQL, own DELIMITER must be defined. It is a consequence of the fact, that trigger contains multiple statements separated by semicolon, so the system must have explicitely defined position of the end point of the block. Thus the problem is to find where the body of the trigger ends. Such procedure is not necessary for DBS Oracle. Each block of statements is delimited by the flash. MySQL can define only two triggers for each operation distinguished by the BEFORE and AFTER keyword. Oracle can define multiple triggers for any operations, however, before the version 11g, there is no possibility to influence the order of execution (by the FOLLOWS clause). Morever, one trigger can be in DBS Oracle defined for multiple operations (like BEFORE INSERT OR UPDATE), however this is not possible to do in MySQL, so such defined code should be separated to named block and called from multiple triggers. Syntax and dialect of the DBS Oracle provides possibilities to define attributes (e.g. OF column_name ON table_name), which border the firing actions of the trigger. Only if the values are changed or processed, trigger is executed reducing the processing time and processor costs [2], [5], [17].

Accordingly, records delimiting values to be managed are processed another way based on their sense. Specifically, in MySQL, these records are defined as references and therefore they cannot be set using SELECT statements directly - addition of another execution of code to assign values is required:

```
-- Oracle
Create or replace trigger log_trig
  Before insert or update on table_name
    For each row
Begin
  Select user, sysdate
      into :new.username, :new.ch_date
    from dual;
end;
/
```

```
-- MySQL
Delimiter $$
  Drop trigger if exists log_trig$$
  Create trigger log_trig
   Before insert on table_name
    For each row
  Begin
    Declare v_username varchar(20);
    Declare v_ch_date date;
     Select user(), now()
        into v_username, v_ch_date;
     set new.username = v_username;
     set new.ch_date = v_ch_date;
  end;$$
delimiter ;
  -- This is only code for insert operation, similar
  solution must be defined also for update operation.
```

The difference in syntax can also be found in defining and managing error codes. Oracle database is designed to call RAISE_APPLICATION_ERROR and MySQL can use the expression SIGNAL.

F. Procedures and functions

As with triggers, even when dealing with procedures and functions, it is necessary to specify own delimiter.

DBS Oracle allows using clause "*table_name.column_name%type*" for parameters, constants and variables definition. It assigns the data type based on table attribute. Thanks to that, if the structure of the table is changed (data type), defined procedure or function does not need to be recompiled. However, MySQL does not support such syntax at all.

The difference in syntax can be found in declaring variables. Variables are defined after IS | AS clause in Oracle. MySQL database variables must be declared in the body of the named block using the keyword DECLARE.

Each function must be defined with particular data type, which value will be returned. Oracle DBS uses clause *RETURN data_type* positioned before the clause IS | AS. MySQL uses *RETURNS data_type* located before the body (before BEGIN) [17].

VIII. APLLICATION FOR SYNTAX TRANSFORMATION

Based on knowledge and of syntactic and semantic differences, complex conversion functionality has been created, from DBS Oracle syntax to MySQL, but also vice versa. Developed solution application core part is Java programming language version 8, which offers wide range of opportunities and features for dealing with strings, which is in this case very important criterion both for development as well as for working with final solution.

Application consists of two packages:

- Gui – this package contains directly executable class MainGui.
- Translator – contains all classes representing particular statements types and also auxiliary classes for transforming specific cases of the syntax.

Application can be started by running the file *Translator.jar*. In the first part of the window, several settings are located. Code can be written directly or loaded from the file. All input files should be in SQL format and extension, the rebuild process is done online. In the left part, we can see original code, right part represents processed and converted code. Fig. 2 shows the screen of the application.

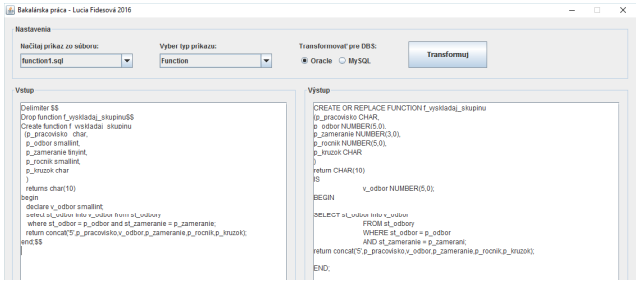

Fig. 2. Application screen

IX. EXPERIMENTS

Focus of the experiment processing and comparison is based on total processing time and costs of the CPU. Both of the mentioned database systems provide ways to determine duration of each part of the execution with regards on used sources, indexes and so on. Problem, however, occurs, because each DBS measures such values in another units and different accuracy, so it is not possible to get strict value. MySQL database system types use SHOW PROFILE commands to get details of the execution. The most important part for the experimental comparison in our system was total duration and how much of this time requires CPU to process command. This information can be read from the table and are given in seconds with several decimal places accuracy. One of the most favorite performance evaluation feature in DBS Oracle is AUTOTRACE displaying execution information depending on the settings and parameters user set. Duration is displayed in seconds, but it is rounded to the nearest one. CPU costs are expressed in percentage based on optimizer calculations of manipulated data sample, thus such value is only estimation. As a consequence, such values cannot be compared each other's and used for evaluating results [1], [4], [11], [13], [14].

Since it was not possible to compare the processed data effectively and accurately using profiling tools, whereas they are reported in different units, accuracy and principles, it was necessary to find and develop another way of comparison, which can be applicable for both DBS and has no impact on output result sets. Another problem is based on data caching into the memory, consequencing, that data are loaded into memory and another query performance would be rapidly faster by accessing memory, not the disc files. However, it would distort results of the experiments. Thus, to solve this problem, all tables are removed after query execution and are recreated once again. Execution order consists of these steps:

- Executing script with cursor for dropping all tables in the database.
- Database tables and import data themselves.
- After import is done, program for measuring duration of the query is started. It is written in PHP and consists of database connector using PDO interface. Both databases are working on the same machine accessed locally. Variables store current system time, database query is performed, which is loaded into the memory from file. After that, new current system time is loaded and the difference with the first one is the stored as result of the processing.

To remove local dependencies and impacts of another processing, each statement has been performed five times, average values are used for statistical analysis. Yellow row in the experiment results table expresses average values.

Experimental comparison has been performed using two machines with following parameters (Table V.).

There is statistical analysis of achieved results in this chapter. Evaluation of the particular experiment consists of these parts:

- command, which is executed and tested,
- data obtained from the database,
- evaluation.

TABLE V. MACHINE PARAMETERS

	Machine 1	**Machine 2**
CPU	Intel Core i7 4710HQ, 3300 MHz, 4 cores, 8 threads	Intel Core i5 4200M, 3100 MHz 2 cores, 4 threads
RAM	16 GB	8 GB
HDD	Samsung SSD 840 EVO SATA3	Samsung SSD 850 EVO SATA3
OS	Windows 8.1 Pro	Windows 10 Pro
PHP	7.0.0 64bit thread safe	
MySQL	MySQL Community Edition 5.7.9	
Oracle	Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit	

Data in the database can be stored using multiple ways. Methods of the data storage influence processing techniques

and data access. Therefore, we decided to include two most commonly used storage engines – InnoDB and MyISAB. Using particular engine should depend on database requirements. If there is large data amounts and needs for foreign key and wide range of transactions, InnoDB is preferred. On the other hand, if there is frequent execution of the destructive operations or even full text search is necessary, MyISAM engine is better [13], [16].

Multiple experiments have been performed and evaluated, we will highlight some specifics of proposed systems (Table VI.). All data in the tables are expressed in milliseconds.

A. Experiment 1 – using DISTINCT clause

```
select DISTINCT
        name, surname, l_country.id_country
  from l_person join l_country
   on (l_person.id_country=l_country.id_country)
    join l_flight_ticket
  on(l_person.id_card=l_flight_ticket.id_card and
    l_person.card_type=l_flight_ticket.card_type);
```

TABLE VI. EXPERIMENT 1 RESULTS

Machine 1			Machine 2		
Oracle	MySQL		Oracle	MySQL	
	InnoDB	MyISAM		InnoDB	MyISAM
257	010	012	497	013	032
340	011	010	472	014	023
348	012	010	482	012	041
348	0120	036	671	011	040
341	011	011	482	014	050
347	**010**	**011**	**521**	**013**	**037**

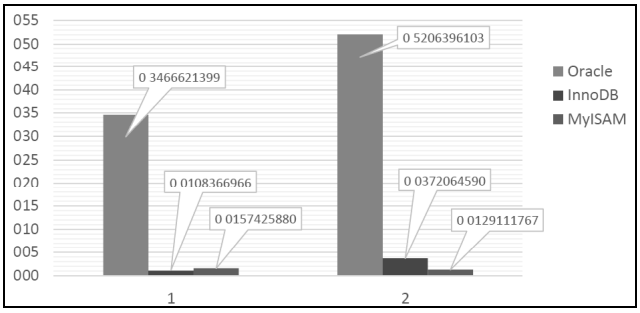

Fig. 3. Experiment 1 results (ms)

Duration of the execution of defined Select statement varies considerably across DBS (all values are expressed in milliseconds). Let show the EXECUTION PLAN in the DBS Oracle using AUTOTRACE feature. We can percieve, that the whole set of the data is obtained in the first phase, which must be then processed - over that set, HASH and SORT operations to ensure uniquencess of the results are performed.

```
-------------------------------------
| Id  | Operation                     |
-------------------------------------
|   0 | SELECT STATEMENT              |
|   1 |  HASH UNIQUE                  |
|   2 |   MERGE JOIN                  |
|   3 |    TABLE ACCESS BY INDEX ROWID|
|   4 |     INDEX FULL SCAN           |
|*  5 |    SORT JOIN                  |
|   6 |     VIEW                      |
|   7 |      HASH UNIQUE              |
|*  8 |       TABLE ACCESS FULL       |
-------------------------------------
```

Fig.4. Execution plan – Experiment 1

As we will see, it is only one statement type, where Oracle DBS achieves worse results. However, same functionality can be also reached by using GROUP BY clause, therefore the second experiment will deal with removing duplicates. There is no aggregate function, so there is only one aim – remove duplicate tuples from the result set.

B. Experiment 2 – using GROUP BY clause

```
select name, surname, l_country.id_country
   from l_person join l_country
    on (l_person.id_country=l_country.id_country)
     join l_flight_ticket
   on(l_person.id_card=l_flight_ticket.id_card and
     l_person.card_type=l_flight_ticket.card_type)
  group by name, surname, l_country.id_country,
          l_person.id_card, l_person.card_type;
```

This statement provides the same results as *Experiment 1*, but it is rewritten by using GROUP BY clause. Although Oracle is better in comparison with MySQL, it is significantly worse when dealing with DISTINCT (more than 100 times).

TABLE VII. EXPERIMENT 2 RESULTS

Machine 1			Machine 2		
Oracle	MySQL		Oracle	MySQL	
	InnoDB	MyISAM		InnoDB	MyISAM
780	1 259	1 222	916	1 372	1 381
888	1 307	1 215	963	1 385	1 375
765	1 295	1 222	959	1 387	1 653
762	1 257	1 218	947	1 386	1 385
765	1 258	1 228	947	1 379	1 389
792	**1 275**	**1 221**	**955**	**1 382**	**1 437**

Experiment 2 shows, that Oracle provides improvement with value 35,13% for machine 1 and value 28,00% for machine 2 (MySQL MyISAM reference 100%). When dealing with MySQL InnoDB (reference 100%), database system Oracle provides improvement – 37,88% (machine 1) and 30,76% (machine 2) – Table VII.

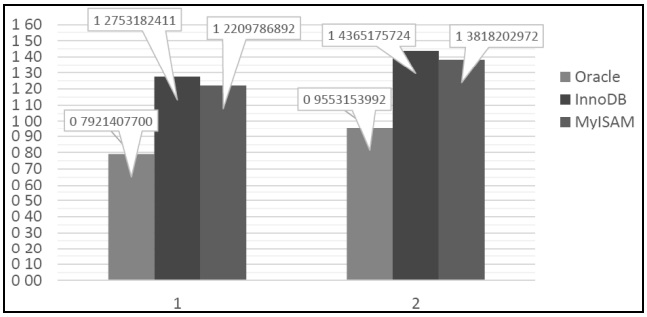

Fig. 5. Experiment 2 results (ms)

C. *Experiment 3*

The *Experiment 3* deals with time, conversion function to string is used (TO_CHAR for Oracle DBS, DATE_FORMAT for MySQL). Actual date is lowered by one month (function ADD_MONTHS for Oracle, DATE_SUB for MySQL). Also DISTINCT clause is used.

```
-- Oracle
Select distinct name, surname
  from l_person
       join l_flight_ticket using(id_card, card_type)
   where to_char(cancel_date, 'MM.YYYY' =
  to  char(add  months(sysdate, -1), 'MM.YYYY');
```

```
-- MySQL
Select distinct name, surname
  from l_person
       join l_flight_ticket using(id_card, card_type)
   where date_format(cancel_date, '%m.%y') =
   date_format(date_sub(now(),
                        interval 1 month), '%m.%y');
```

Thus, following Select statement lists the names and surnames of the people, who canceled their flights last month (Table VIII).

TABLE VIII. EXPERIMENT 3 RESULTS

Machine 1			Machine 2		
Oracle	MySQL		Oracle	MySQL	
	InnoDB	MyISAM		InnoDB	MyISAM
134	1 269	1 768	358	2 110	2 122
135	1 265	1 808	354	2 150	2 131
135	1 268	1 760	333	2 114	2 099
139	1 273	1 754	346	2 115	2 127
132	1 257	1 766	356	2 108	2 120
137	**1 267**	**1 771**	**349**	**2 119**	**2 120**

As we can see, conversion function processing is fare more effectively managed in DBS Oracle. Moreover, there is powerful automatic conversion handling.

Oracle DBS performance provides following improvements: (MySQL is reference 100%):

- Machine 1
  - 89, 27% (InnoDB)
  - 92, 26% (MyISAM)
- Machine 2
  - 83, 54% (InnoDB)
  - 83, 53% (MyISAM)

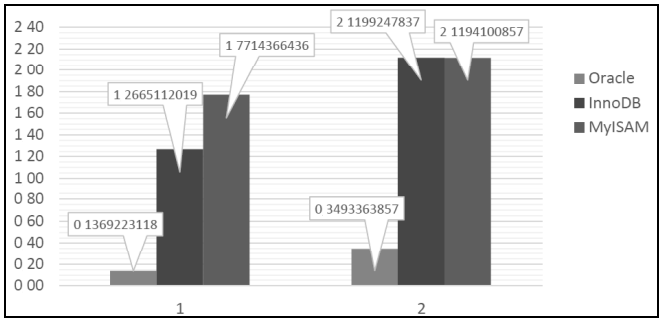

Fig. 5. Experiment 3 results (ms)

D. *Experiment 4*

The last experiment handles two significant parts – aggregate function with GROUP BY section and sorting criterion. also in this case, DBS Oracle provides better solutions, however, this experiment highlights the impact of technical parameters of hardware.

```
– Oracle
Select f.id_flight, f.id_class, sum(f.price)
  from l_class c
    join l_flight_ticket f on (f.id_flight = c.id_class)
    where f.cancel_date is NULL
      and to_char(reservation_date, 'YYYY')=2006
  group by f.id_flight, f.id_class
    order by 1,2;
```

For database system MySQL, syntax is almost the same, TO_CHAR function is replaced by the equivalent DATE_FORMAT (principle is expressed in *Experiment 3*).

Defined experimental environment provides us following results (Table IX.).

TABLE IX. EXPERIMENT 4 RESULTS

Machine 1			Machine 2		
Oracle	MySQL		Oracle	MySQL	
	InnoDB	MyISAM		InnoDB	MyISAM
439	890	622	581	715	708
433	909	620	570	715	718
434	893	618	579	707	711
441	893	623	584	723	708
439	890	621	640	717	707
437	**895**	**621**	**591**	**716**	**710**

Based on experiments, performance improvement of Oracle DBS in comparison with MySQL is 51,17% (InnoDB reference 100%) and 29,63% (MyISAM reference 100%). When dealing with worse hardware characteristics, difference is not so significant:

- 16, 76% (InnoDB reference 100%),
- 17, 46% (MyISAM reference 100%).

Thus, it can be mentioned, that database server configuration and parameters influence the performance difference between DBS Oracle and MySQL.

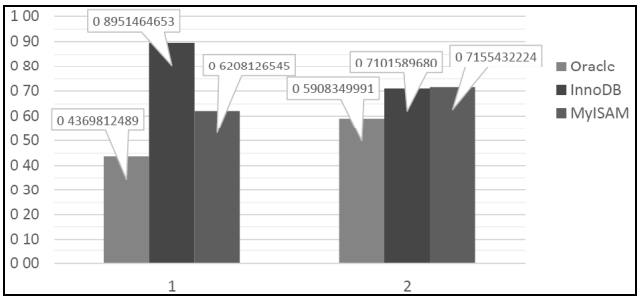

Fig. 6. Experiment 4 results (ms)

X. Summary

Based on these experiments, we can say that in most cases Oracle database processing command requirements are less than a MySQL database (processing time). Database system Oracle, however, lags significantly in choosing unique record values provided by the DISTINCT feature.

It was further found that a change in the method of storing data in the MySQL database has greater impact on the experiments for machine 1 in comparison with machine 2. When dealing with date formats, storage engine InnoDB je faster than MyISAM. On the other hand, when managing aggregate functions and ORDER BY clause in the Select statements, MyISAM is better choice.

XI. Conclusions

Database is the most significant source and part of almost every information system. Data are managed effectively by the SQL commands. Focus on the data processing and amount of data processed over the time even highlights the necessity. Although several database system types have been proposed with various quality and complexity, relational model is still most often used due to several characteristics. Nowadays, there is wide range of systems, which can manage data in this manner. In this paper, we focus on two systems developed by Oracle, whereas one of them is open free and the second one is commercial. Several characteristics and limitations are proposed and compared. Moreover, transformation software has been proposed, which automatically translates code from one DBS to another. Thanks to that, data transfer to another solution is no longer problem. Experiment section deals with several queries characteristics like processing and resulting unique values, managing aggregate functions or result set sorting.

In the future, we will deal with the temporality and extension to NoSQL database systems and fusion with relational concept. We would also like to optimize translated code based on optimization principles of the particular database type.

Acknowledgment

This publication is the result of the project implementation:

Centre of excellence for systems and services of intelligent transport II., ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

"Podporujeme výskumné aktivity na Slovensku
Projekt je spolufinancovaný zo zdrojov EÚ"

This work was partially supported by the project:

Creating a new diagnostic algorithm for selected cancers, ITMS project code: 26220220022 co-financed by the EU and the European Regional Development Fund.

References

[1] L. Ashdown and T. Kyte, *Oracle database concepts*, Oracle Press, 2015.

[2] E. Bertino, G. Guerrini, I. Merlo, *Trigger inheritance and overriding in an active object database system*, IEEE Transactions on Knowledge and Data Engineering (Volume:12, Issue: 4), 2000.

[3] C. J. Date, *Date on Database*, Apress, 2006.

[4] S. Feueuerstein, S, *Oracle PL/SQL Programming*, O'Reilly, 2014.

[5] W. K. Hauger, M. S. Olivier, *The role of triggers in database forensics*, Information Security for South Africa, 2014.

[6] J. Janáček, M. Kvet, *Min-Max Optimization Of Emergency Service System By Exposing Constraints*, in Communications: Scientific Letters of the University of Žilina, volume 2/2015, 2015, pp. 15 – 22

[7] J. Janáček, M. Kvet, *Public service system design by radial formulation with dividing points*, in Procedia computer science Vol. 51, 2015, pp. 2277 – 2286

[8] T. Johnston and R. Weis, *Managing Time in Relational Databases*, Morgan Kaufmann, 2010.

[9] M. Kvet and K. Matiaško, *Transaction Management.* 2014. CISTI, Barcelona, pp.868-873.

[10] M. Kvet and M. Vajsová, *Transaction Management in Fully Temporal System*, 2014. UkSim, Pisa, pp. 147-152.

[11] T. Kyte and D. Kurn, *Expert Oracle Database Architecture*, Apress, 2014.

[12] K. Matiaško, et al., *Database systems.* EDIS, 2008.

[13] M. McLaughlin, *Database 11g & MySQL 5.6 Developer Handbook*, Oracle Press, 2011.

[14] H. Molina, et al., *Database systems – The complete book*, Pearson, 2008.

[15] S. Nyati, *Performance evaluation of unstructured NoSQL data over distributed framework,* In Advances in Computing, Communications and Informatics (ICACCI), 2013, pp. 1623 - 1627

[16] M. Reid, *InnoDB*, Packt Publishing, 2013.

[17] R. Rood, et al., *Oracle Advanced PL/SQL Developer Professional Guide*, Packt Publishers, 2012.