**Researches about theory (R)**

**2\_R. Describe the most common configuration of data repositories in the real world and corporate environment. Concepts such as Operational or Transactional systems (OLTP), Data Warehouse DW, Data Marts, Analytical and statistical systems (OLAP), etc. Try to draw a conceptual picture of how all these components may work together and how the flow of data and information is processed to extract useful knowledge from raw data.**

The **Online transaction processing** (OLTP) systems facilitate the transactions processes. We can define two meanings of the term transaction:

* Transaction in Database Management System means the atomic process to apply changes to data.
* Transaction in Financial world means the actual process to move money from a source to a target.

This type of systems is specifically advanced to process insert and update processes. One example of these is the ATM where you can send/receive money. Obviously, the requisites for these systems are availability, speed and high concurrency performance. The OLTP system design requires:

* Rollback segments: These are the portions of DBMS who record the actions of transactions and provide the ability to go back and restore the system to its previous state.
* Clusters: These are groups of tables in a DBMS, and they help the systems to make the JOIN operations faster.
* Discrete transactions: This technology make the changes atomic and defers all the changes until the transaction is committed. So, it stores changes in a different environment and apply the changes when the transaction is confirmed.
* Block size: The block size of these systems must be a multiple of the OS where the system is hosted to avoid multiple unnecessary and low-performance input/output actions.
* Buffer cache size: These types of systems must maximize the use of caching technology due to high level of concurrent requests.

This type of system is contrasted to **Online Analytical Processing** (OLAP) because these systems process much more complex queries but in a smaller volume. OLTP uses all the CRUD operations (Create, Read, Update and Delete) instead OLAP systems typically uses read only. Typical applications of OLAP include business reporting for sales, marketing, management reporting, business process management (BPM), budgeting and forecasting, financial reporting and similar areas, with new applications emerging, such as agriculture. There are three main types of OLAP systems:

* Multidimensional OLAP (MOLAP): MOLAP (multi-dimensional online analytical processing) is the classic form of OLAP and is sometimes referred to as just OLAP. MOLAP stores this data in an optimized multi-dimensional array storage, rather than in a relational database.
* Relational OLAP (ROLAP): ROLAP works directly with relational databases and does not require pre-computation. The base data and the dimension tables are stored as relational tables and new tables are created to hold the aggregated information.
* Hybrid OLAP (HOLAP): HOLAP database will use relational tables to hold the larger quantities of detailed data and use specialized storage for at least some aspects of the smaller quantities of more-aggregate or less-detailed data. HOLAP addresses the shortcomings of MOLAP and ROLAP by combining the capabilities of both approaches.

A **Data Warehouse** (DW) is a centralized repository of data that can be analyzed to make more informed decisions. DBMS and other systems send information to data warehouses usually on a regular basis. Business analysts, data engineers, data scientists access data to take decisions and study the flow of the information. The data and the related analysis have become critical factors to ensure the competitiveness of corporates. Reports, dashboards and analytics tools are indispensable for extracting insights from data, monitoring business performance and supporting decision making.

Diagram

Description automatically generated

Benefits of DW (taken from Wikipedia): A data warehouse maintains a copy of information from the source transaction systems. This architectural complexity provides the opportunity to:

* Integrate data from multiple sources into a single database and data model. More congregation of data to single database so a single query engine can be used to present data in an ODS.
* Mitigate the problem of database isolation level lock contention in transaction processing systems caused by attempts to run large, long-running analysis queries in transaction processing databases.
* Maintain data history, even if the source transaction systems do not.
* Integrate data from multiple source systems, enabling a central view across the enterprise. This benefit is always valuable, but particularly so when the organization has grown by merger.
* Improve data quality, by providing consistent codes and descriptions, flagging or even fixing bad data.
* Make decision–support queries easier to write.
* Organize and disambiguate repetitive data.

Other important well-deserved mentions are **Data marts**, which are repositories containing well-ordered and classified data for a specific topic, as we can see from the previous image (purchasing, sales, inventory).

3\_R. Show how we can obtain an online algo for the arithmetic mean and explain the various possible reasons why it is preferable to the "naive" algo based on the definition.

What is the arithmetic mean?  
In mathematics and statistics, the arithmetic mean, or simply the mean or the average (when the context is clear), is the sum of a collection of numbers divided by the count of numbers in the collection. The collection is often a set of results of an experiment or an observational study, or frequently a set of results from a survey.   
The naive algorithm, so the first coming in mind is, obviously, to sum all the values and then divide by the number of the items.  
So, for instance, if I have this dataset:

|  |  |
| --- | --- |
| Person | Age |
| Edoardo | 23 |
| Francesca | 45 |
| Manuele | 1 |
| Chiara | 22 |
| Mario | 56 |
| Francesco | 98 |
| Cristian | 33 |
| Zaira | 77 |
| Marco | 16 |
| Gianna | 19 |

To calculate the arithmetic mean, I must sum all the ages and then divide by 10. So: 390/10 = 39. We know now the arithmetic mean of this set of values is 39.  
We calculated this number without difficulty, but can we have some obstacles in some weird circumstances?  
**Yes**. First, as we have seen in the previous section, we could have an infinite set of values. So, for example, we could have to collect data every n milliseconds (timeseries data, for instance financial/geo-seismic data) and it's practically impossible to start this type of algorithm because we never will have a finite set of values.  
There are also others problems with floating point values and very huge values where the naive approach doesn't work. These types of problems are discussed in the Researches about application section. By the way, a better solution to calculate the arithmetic mean is using the Knuth formula:knuth formulaor the Kahan algorithm:

function KahanSum(input)

var sum = 0.0 // Prepare the accumulator.

var c = 0.0 // A running compensation for lost low-order bits.

for i = 1 to input.length do // The array input has elements indexed input[1] to input[input.length].

var y = input[i] - c // c is zero the first time around.

var t = sum + y // Alas, sum is big, y small, so low-order digits of y are lost.

c = (t - sum) - y // (t - sum) cancels the high-order part of y; subtracting y recovers negative (low part of y)

sum = t // Algebraically, c should always be zero. Beware overly-aggressive optimizing compilers!

next i // Next time around, the lost low part will be added to y in a fresh attempt.

return sum

**Applications / Practice (A)**

**2\_A. Create - in both languages C# and VB.NET - a demonstrative program which computes the online arithmetic mean (if it's a numeric variable) and your own algo to compute the distribution for a discrete variable and for a continuous variable (can use values simulated with RANDOM object).**

GITHUB!

**3\_A. Create an object providing a rectangular area which can be moved and resized using the mouse. This area will hold our future charts and graphics.**

TODO

**Researches about applications (RA))**

**1\_RA. Understand how the floating-point representation works and describe systematically (possibly using categories) all the possible problems that can happen. Try to classify the various issues and limitations (representation, comparison, rounding, propagation, approximation, loss of significance, cancellation, etc.) and provide simple examples for each of the categories you have identified (e.g.,** <https://floating-point-gui.de/basic/> ,  <https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html> , <http://indico.ictp.it/event/8344/session/50/contribution/207/material/slides/0.pdf> , <https://stackoverflow.com/questions/2100490/floating-point-inaccuracy-examples> , **etc.)**

In general, who don't have knowledge about computer programming and computer architecture could think as floating point numbers as just numbers with a comma (point, or separator in general). So, if I have to representate 16,56; this number can be stored in a computer like this: 16,56. So a number representing the integer number, a separator and a number representing the decimal part. This representation is quite uncomfortable. For example, if I have a computer working with 32-bit word, I can assign 20 bits for the integer part and the remaining 12 bits for the decimal part. With this method I can represent numbers going from -524.287 to 524.287 with only 4 digits for the decimal part, so the smallest number I can represent is 0,00025. This obviously it's very unuseful for scientific calculus.   
So, engineers thought about a new representation of floating point numbers in computers that works in this way: I have a triple of values <s,m,e>: the s is the sign of the number, the m is the mantissa and the e is the exponent. This: + 1656 x 10^-2 is a good way to represent the number 16,56. By the way, even if this representation is the real one used in computing now, it still has problems: let's analyze them. The problem of **scale**. Each FP number has an exponent which determines the overall “scale” of the number so you can represent either small values or really larges ones, though the number of digits you can devote for that is limited. Adding two numbers of different scale will sometimes result in the smaller one being “eaten” since there is no way to fit it into the larger scale. Conversions to integer are not intuitive: converting (63.0/9.0) to integer yields 7, but converting (0.63/0.09) may yield 6. This is because conversions generally truncate rather than round. Floor and ceiling functions may produce answers which are off by one from the intuitively expected value. Limited exponent range: results might overflow yielding infinity, or underflow yielding a subnormal number or zero. In these cases, precision will be lost. 64-bit integers may not fit into a double. Unless you know only 53 bits are used, the only suitable floating-point type is the 80-bit extended precision format (long double) on x86 processors. In many if not most cases numerical computations involve iteration rather than just the evaluation of a single formula. This can cause errors to accumulate, as the chance of an intermediate result not being exactly representable rises. Even if the target solution is an attractive fixed point of your iteration formula, numerical errors may catapult you out of its domain of attraction. In addition to the rounding errors introduced at every step in a computation, the values your processor computes for transcendental functions may be off by more than basic arithmetic. This is because optimal rounding to the nearest representable number requires distinguishing on which side of the middle between them (an infinitely narrow line) the true result lies. Transcendental functions are power series with infinitely many non-zero coefficients, so one cannot really know that for a given argument without computing an infinity of terms of the series.

**References**

[1][Data Science & Statistics: Levels of measurement](https://www.youtube.com/watch?v=eghn__C7JLQ)  
[2][Wikipedia - Level of measurement](https://en.wikipedia.org/wiki/Level_of_measurement)  
[3][Wikipedia - Online Transaction Processing](https://en.wikipedia.org/wiki/Online_transaction_processing)  
[4][Wikipedia - Online Analytical Processing](https://en.wikipedia.org/wiki/Online_analytical_processing)  
[5][Wikipedia - Data Warehouse](https://en.wikipedia.org/wiki/Data_warehouse)  
[6][Wikipedia - Arithmetic mean](https://en.wikipedia.org/wiki/Arithmetic_mean)  
[7][Best algorithms to compute the “online data stream” arithmetic mean](https://statisticsforcs.wordpress.com/2017/10/24/best-algorithms-to-compute-the-online-data-stream-arithmetic-mean/)  
[8][Wikipedia - Floating point arithmetic](https://en.wikipedia.org/wiki/Floating-point_arithmetic)  
[9][Progettazione di sistemi digitali - Rappresentazione dei numeri razionali](https://twiki.di.uniroma1.it/twiki/viewfile/Architetture1/AL/WebHome?rev=7;filename=05-razionali.pdf)  
[10][StackOverflow - Floating point inaccuracy examples](https://stackoverflow.com/questions/2100490/floating-point-inaccuracy-examples)  
[11][What you never wanted to know about floating point but will be forced to find out](https://www.volkerschatz.com/science/float.html)