Part 1:

1. Just solving this task I did learn something new. This is why I didn’t know what the name of a Lambda Architecture is. Intuitively I used few years ago something similar. In case I don’t know something or I want to step into something new I do following:
   1. Search and evaluate information and their sources
   2. Compare different sources to match content is same, similar or different
   3. To became some first experience build up a small prototype and test the main important use case which might important. (This I did not for lambda architecture, because it is more a model than a technology.)
2. A lambda architecture describes a model to handle data with the same structure which is stored in an existing database on the hand and data which is new, that means created or read in real time from a different source than the existing database.

It is divided into 3 layers (see <http://lambda-architecture.net/img/la-overview_small.png>):

Note: all data received are forwarded to batch and speed layer

* 1. batch layer is covering the existing database with in the historical data
     + it should be used as immutable and append-only database
     + it also might pre calculate some data based on the incoming data to prepare batch views in serving layer
  2. serving layer
     + does support low-latency and ad-hoc access to the data, by indexing batch views
  3. speed layer
     + does support real-time views with the new data only.

It solves the problem, that access to the historical data have different latencies and most often worse latencies than accessing live streams. Also the historical data views will be available later than data arrive in the live stream. It is a scalable, generic and fault-tolerant data processing architecture.

From the user/app perspective the data is every time available if requested with very small latencies.

Part 3:

To decide what is the best solution for a data ware house architecture depends on:

1. what data is handled (stream, real-time, historical data) all together or just one of them
2. what time is defined is required to get the data from database to user
3. in what frequency the data is being updated, how fast the data must be forwarded to the user
4. does the schema differs between different source
5. do we have more than one technology
6. does the data must preprocessed any how

The most often use case I currently expect is: the user want to have an app which displays combined similar (same schema) data with historical content and live content. That’s why a ware house based on the lambda architecture is a modern data warehouse architecture.

In cases it is not that time critical or no preprocessing is necessary the speed layer might skipped to safe resources and costs. Than we will get an architecture which is modern enough to solve requirements.

For additional readings and decisions, will be helpful “Data Warehouse Design: Modern Principles and Methodologies“By Matteo Golfarelli, Stefano Rizzi or for a short summary <https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwi2zJjrndPZAhWEyKQKHeBnDPkQFghCMAA&url=https%3A%2F%2Fwww.sqlsaturday.com%2FSessionDownload.aspx%3Fsuid%3D16617&usg=AOvVaw2sCaJqLJilmL0Y0y3g5WtI> .

Part 2:

Mysql create statements:

Small modification, since e.g. auto\_increment vor varchar is not useful.

**DROP** **TABLE** **IF** **EXISTS** ` session\_events `;

**CREATE** **TABLE** `session\_events` (

`id` **INT**(11) **NOT** **NULL** **AUTO\_INCREMENT**,

`user\_id` **VARCHAR**(50) **NOT** **NULL** **DEFAULT** '0',

`session\_id` **VARCHAR**(256) **NOT** **NULL** **DEFAULT** '0',

`event\_name` **ENUM**('E1','E2','E3','E4') **NOT** **NULL** **DEFAULT** 'E1',

`event\_duration` **INT**(11) **NOT** **NULL** **DEFAULT** '0',

`at` **TIMESTAMP** **NOT** **NULL** **DEFAULT** **CURRENT\_TIMESTAMP**,

**PRIMARY** **KEY** (`id`)

)

**ENGINE**=**InnoDB**

;

**DROP** **TABLE** **IF** **EXISTS** `screen\_flows`;

**CREATE** **TABLE** `screen\_flows` (

`session\_id` **VARCHAR**(255) **NOT** **NULL**,

`at` **TIMESTAMP** **NOT** **NULL** **DEFAULT** **CURRENT\_TIMESTAMP** **ON** **UPDATE** **CURRENT\_TIMESTAMP**,

`name\_1` **ENUM**('E1','E2','E3','E4') **NOT** **NULL**,

`duration\_1` **INT**(10) **UNSIGNED** **NOT** **NULL** **DEFAULT** '0',

`event\_count\_1` **INT**(10) **UNSIGNED** **NOT** **NULL** **DEFAULT** '0',

`name\_2` **ENUM**('E1','E2','E3','E4') **NOT** **NULL**,

`duration\_2` **INT**(10) **UNSIGNED** **NOT** **NULL** **DEFAULT** '0',

`event\_count\_2` **INT**(10) **UNSIGNED** **NOT** **NULL** **DEFAULT** '0',

`name\_3` **ENUM**('E1','E2','E3','E4') **NOT** **NULL**,

`duration\_3` **INT**(10) **UNSIGNED** **NOT** **NULL** **DEFAULT** '0',

`event\_count\_3` **INT**(10) **UNSIGNED** **NOT** **NULL** **DEFAULT** '0',

**PRIMARY** **KEY** (`session\_id`)

)

**ENGINE**=**InnoDB**

;