Haaga-Helia
University of Applied Sciences

Timed Delay Data Management in Salesforce

Gearóid O' Ceallacháin

Bachelor's Thesis
Business Information Technology
2019

Haaga-Helia
University of Applied Sciences

Degree programme

Abstract

Date 26 May 2019

Author(s)
Gearóid O'Ceallacháin, A1600639

Degree programme
Degree Programme in Business Information Technology

Report/thesis title
Timed Delay Data Management in Salesforce

Number of pages and appendix.
39 + 17

This Bachelor's thesis in Business and Information Technology focuses on PaaS (Platform-as-a-service). In more specific, Time delay data management in Salesforce from a technical-economical point of view. In the context of PaaS, the aim of this study is to explore if the governor limits in Salesforce can be broken for the benefit of the users. Or could Time Delay Data management provide even more efficient solution for this purpose. The research question in this thesis is: is it possible to bend or break the governor limits in Salesforce to be more efficient? It is answered by analysing the following aspects.

I. Can one use batches and triggers to break or bend the governor limits?
II. Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?

This development case for a company 'x' was chosen based on a questionnaire (sent to 30 employees), followed by discussions with the same employees. Based on the answers, it was identified how could they be more efficient in their working methods, for example, by saving time in testing and debugging in Salesforce. Following that, it was tested if the governor limits can be broken by using synchronous and asynchronous commands in conjunction with batches and triggers.

As a result, it is discovered that the governor limits are means to keep order. Moreover, they are means to keep clients safe with the Salesforce product. As a self-guiding rail to near perfect code, governor limits maintain undisturbed service every time Salesforce is updated. As a conclusion, instead of trying to break the governor limit in Salesforce, it is more efficient to create the illusion of breaking the governor limits by using asynchronous methods.

Keywords PaaS, Salesforce, cloud, asynchronous, synchronous, batches, triggers.

Table of contents

Acronyms and Abbreviations

In this thesis Acronyms and abbreviations are used

Account	New customer, company, or consumer.
API	Application Programming interface
AWS	Amazon Web Services.
App	Application.
API	Application Programming interface.
Asset	Product or model the customer owns.
Camel case	Words are put together separated by capital letters
CRM	Customer Resource Management
CRUD	Create, Read, Update, Delete, Custom object only.
Child object	Related to the parent object. A sub-object
Custom Object	A new variable, which the customer needs, requires. Ends with __c This also adds to the customization of each build.
DML	Data Manipulation Language
GUI	Graphical user interface
Heroku	Based on AWS this connects the outside world with Salesforce.
IoT	Internet of Things.
ISVs	Independent Software vendors
MVC	Model View Controller
REST	Representational State Transfer
Salesforce	The world's #1 Customer Relationship platform
SLDS	Salesforce Lightning Design System.
SOAP	Simple Object Access protocol
sObject	Salesforce Object, Account, Opportunity, Event
SOSL	Salesforce search language
SOQL	Salesforce Object Query Language.

TP	Trailhead playground, a place where one can test before putting it in the cloud.
Triggers	Commands that happen when an action is performed.
Upsert	Combined verb for update or insert actions
URI	Uniform Resource Identifier,

(Salesforce Developers, 2019)

1 Introduction

The purpose of this bachelor's thesis in Business and Information Technology is to introduce and add code to a standard template Salesforce solution. Whilst companies are often writing code for themselves, the company 'x' of this study (consultant company using Salesforce) writes code for the client companies using apex through Salesforce. The application of a Salesforce approach in this thesis provide a completed solution for company 'x' needs: how to be more efficient in their working methods, for example, by saving time in testing and debugging in Salesforce.

The company 'x' needs are met by applying the method with Apex to the CRM (Customer relationship management) model together with Salesforce. Meaning, that company 'x' adds code on top of Salesforce with Apex in order to create Timed delay Data management. Timed Delay Data management in Salesforce narrows the possibilities of the automation. It does that both by out of the box (standard Salesforce platform), and also by customization that company 'x' offers to its clients who wish to integrate Salesforce into their company. But more complex code requires more customization in order to sit on top of Salesforce.

In addition, during company 'x' projects, code segments resurface and are recycled, whether intentional or not. Thus, they take code from one program (as the code has been tested and verified, known to work) and put into another program, that is already being tested and known error-free as well. This leaves less errors to be debugged at a later stage.

The aim of this study is to explore if the governor limits in Salesforce can be broken for the benefit of the users. Or could Time Delay Data management provide even more efficient answers to the company 'x' needs? This development case for a company 'x' was chosen based on a questionnaire (sent to 30 employees), followed by discussions with the same employees. Although the answers to the questionnaire did not point to any specific development need, the discussions with the company 'x' staff after the questionnaire did. Since they were mainly using

Salesforce in their work, they were often facing barriers created by governor limits. The answers to the questionnaire and discussions with the staff members also provided the needed information on what programs and pieces of code are recycled in the company ‘x’.

Overall, in this thesis is focused on testing if the governor limits can be broken by using synchronous and asynchronous commands in conjunction with batches and triggers. How synchronous and asynchronous methods along with triggers and batches can be used to gain efficiency over the governor limits?

1.1 Background: short history of cloud computing and governor limits in Salesforce

Cloud computing is still in the early stages of development. If using a computer analogy from the 80s: a 286-personal computer was fast for its time, but not when compared to the processing power of today’s world (Regalado, 2011).

The topic for this study is ‘Timed delay management in Salesforce’. Salesforce and its limits were created in 1996 to the cloud computing reality of that time (Regalado, 2011). Therefore, today it is useful to see if governor rules can be stretched or used beyond their limits. This would be needed today in order to work more efficiently, to pack more work into each command and to get more answers back with each query. Moreover, by using synchronous and Asynchronous methods (Apex Developer Guide, 2018), along with triggers and batches (Salesforce Developers, 2018) it is useful to investigate how it could be possible to process more information. Do we need to break the governor limits in order to make Salesforce to match the needs of today?

In Salesforce access to valuable customer information lies within the safe confines of the governor limits. In order to give an overview on governor limits, (Apex Developer Guide, 2013) it can be outlined that they are constraints put on the server to serve all users equally. In sum, the governor limits are created to give everyone the right to access data and to take part in the rich resources as well as services that Salesforce has (Ahmad & Janzewski, 2010).

Governor limits also prevent coders from writing bad code by putting a test marker that they must maintain. Governor limits are created by the consumer company themselves to maintain a higher quality/standard of code. This is also expected from them as service providers in this ever-changing world. However, quality goes both ways. The clean code is assured by its readability, meaning the clarity of the logic. Therefore, the code needs to be readable for others or understandable for the coders themselves later (Salesforce Developers, 2009), (Salesforce Developers, 2016).

1.2 Research question

The research problem can be outlined as following: is it possible to bend or break the governor limits in Salesforce to be more efficient? It is answered by analysing the following aspects.

I. Can one use batches and triggers to break or bend the governor limits?
II. Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?

The hypothesis in this thesis is that the governor limits are getting in the way of efficiency. This is because they create limits in our work and cause us to slow down when querying the database. Thus, the governor limits reduce the number of the possible queries executed at the same time since there are limitless number of users using it at the same time, accessing the same data. If it would be possible to bend or break the governor limits, we would get a lot more done per query than now, when the governor limits restrictions limit is in place. On the other hand, if the rate or flow of information going to and coming from the main servers is restricted, it would cause a level playing field for everyone.

When a code is recycled, it is used in several queries. Recycled code is a code that has been used before, in previous code modules. This way using the code becomes more efficient, as it can be reused without testing new code each time. This increases the efficiency. The premise is that code is recycled, and people are trying to attain more efficient code (Frakes & Terry, 1996).This helps in retrieving the relevant information requested from Salesforce. As a result, time is saved when a

code is recycled and also some the whole process of testing a code is skipped (Frakes & Terry, 1996).

2 Theoretical Framework

In this section, it is discussed about Salesforce.com or force.com triggers, and synchronous/asynchronous methods. These are all apart of older systems, such as SQL. At first, to outline how they were brought to Salesforce, a short overview to Salesforces' history is needed.

In 1999, Salesforce produced a platform called force.com (McCarthy, 2016). It was one of the first CRM applications in the cloud industry where the user could point and click to do what one wishes or make custom user interfaces. To add that unique feeling to the platform it also takes care of CRM applications, where everything is linked and interlinked. At the heart of all this is the multi-tenancy architecture, knowing that one is one of a million. Sharing these limited resources with the rest of the world (Weissman & Bobrowski, 2008).

2.1 Overview of force.com / Salesforce

Governor limits are making sure that everyone has an equal share of the cloud. As with many of the great cloud applications out there, Salesforce is encouraging independent software vendors (ISVs) to join their program without a cost to incorporate their new ideas into the Salesforce family (Hurwitz, et al., 2018). Salesforce users can use standard web development tools including HTML, AJAX, and Adobe Flex to design and write their pages.

Salesforce Mobile is an opensource connected to REST an OAuth these are used to build mobile apps along with SDK 2.0 which supports HTML 5 and hybrid and native mobile apps (Salesforce Developers, 2019).

Currently Salesforce has all aspects of cloud computing covered, from the security aspect to the application that it supplies. This brings the customer information available to the customer on the cloud and accessible to anyone (the customer chooses), creating a truly easy all accessible base of information that the customer created (Salesforce.com, 2017).

2.2 Batches in Salesforce

Being cloud-based, also used by many users at the same time, there is only a certain amount of data that can be processed simultaneously. By entering batches, the workload (for example, reading and sending emails to the entire company of 500) break down into bite-size manageable bites. Since everyone accesses the database individually, it is easy to forget that everyone in the company is accessing the same server. As/When the information is put in the form of batches, the information is not guaranteed to be executed in order (Salesforce Trailblazer Community, 2019), (Apex Developer Guide, 2018).

2.3 Synchronous front-end vs asynchronous backend

When it comes to priority and if some actions are deemed important enough to be done (and need to wait for the answer), synchronous can be used. However, it also has its limitations, such as governor limits. Thus, it takes time to execute an order, something as mundane as sending emails to everyone in the company and subsidiaries stating more than 10,000 employees. This can be done when the stack is at its lowest, when the query for information is low (after work hours for example). Since Salesforce is a PaaS with multiple companies (1000+) on the cloud, this could be anytime (Grogg, 2016).

Also, since asynchronous mode (giving a command and not expecting an answer straight away or executing when the server is quiet) is more popular with backend, it also allows one to set commands and do something else while that command is being executed. As Governor limits are different between synchronous and asynchronous batches, often doubling the amount of traffic allowed to be used asynchronous often used as a lesser priory, also used to do bulk commands such as mass emails or updating paid personnel. In short If the information that is required or the query is not as important or could be re-scheduled for a later date Asynchronous method could be used.

2.3.1 Batches in Salesforce

To explain batches, batches are codes written to process groups commands in Salesforce; for groups in the consumers system, since this is set as one command. Everything in that command gets updated at the same time (Apex Developer Guide, 2019).

Asynchronous and synchronous methods can be inside a batch, this would be several triggers and other components put together. To clarify, batches asynchronous and synchronous methods are not commands but a string(s) of algorithms. Batches are not commands outright, they are a group of commands to bulkify commands reducing the governor limits (Apex Developer Guide, 2019).

2.3.2 Structure of batch

In order to understand a batch in Salesforce, it is useful to outline their structure. Each batch has the following elements:

Start, first used to collect the data (records or objects) for processing. This is done with a query; these are simple commands. Loops would require special intervention and specialist custom code.
Execute, this is where the actual processing of the batch takes place and breaks up the workload into batches of 200 default.
Finish. This brings us to the end of the batch statement, to sum up, the process, giving a conclusion, such as an email or a dialogue box prompting the user to press ‘Ok’. (Apex Developer Guide, 2019).

2.3.3 Batch basic code

To write a batch, one must execute to the Database. Batchable a sign that one is writing a batch file. Also, in order to the batch to be complete, it needs to contain three subheadings **Start, Execute, Finish** (Apex Developer Guide, 2019).

The Querylocator a SOQL command for simple triggers, this, in turn, will sit upon the Platform of Salesforce, adding a more complex code, the other iterable side is

used. Governor limits are observed. Also, using any external objects, the trigger is used (Salesforce Trailblazer community, 2019).

Now that all the information is gathered, it is time to execute, referencing the Database. In Query locator (or if more complex the iterable<sObject>) these are usually expected in the same order, from the Start section, but not guaranteed.
Post Processing completed in the Finish section; this is executed after the batch is done (Salesforce Trailhead, 2019).

```
global class MyBatchClass implements Database.Batchable<sObject> {

   global (Database.QueryLocator | Iterable<sObject>) start (Database.Batchable-
Context bc) {
      // collect the batches of records or objects to be passed to execute
   }

   global void execute (Database.BatchableContext bc, List<P> records) {
      // process each batch of records
   }

   global void finish (Database.BatchableContext bc) {
      // execute any post-processing operations
   }
```

Code Excerpt 1 (Salesforce Trailhead, 2019).

Since this batch could process 10,000 records running the batch 50 times. Each time the governor limits are reset. (see code Excerpt1)

2.3.4 Building on basic batch code

In an international company, new employees unless stated, will not put in the country prefix in front of their telephone numbers. Batches would fill in the missing information placing the country code in front of the telephone number and the country domains in the email. This would be a backend type of command to be executed. Doing this change in batches, several customers at time, or commands at a time would save time and governor limit processing.

```
global class Accountupdate implements Database.Batchable<sObject>
{

                global Database.QueryLocator start(Database.BatchableContext BC)
```

```
            {
        String query = 'SELECT Id,Name,Phone FROM Account ';
        return Database.getQueryLocator(query);
    }

    global void execute(Database.BatchableContext BC, List<Account> scope)
    {
        for ( Account a : prefix)
        {
          a.Phone='+358';
        }
        update prefix;
    }
    global void finish (Database.BatchableContext BC)
    {
    }
}
```

Code Excerpt 2 Code for basic use (Salesforce Developers, 2015).

2.3.5 Batch usage, when and why it is used

Limitation of batch results and batch execution segmentation (batch to be split into many slices). One would use a batch as a reminder. Batches are often used to eliminate the mundane but also for equally important tasks that need to be done, for example, sending a “thank-you” email or email of confirmation when the client would book or reserve a ticket. (Salesforce Trailhead, 2019)

2.4 Triggers

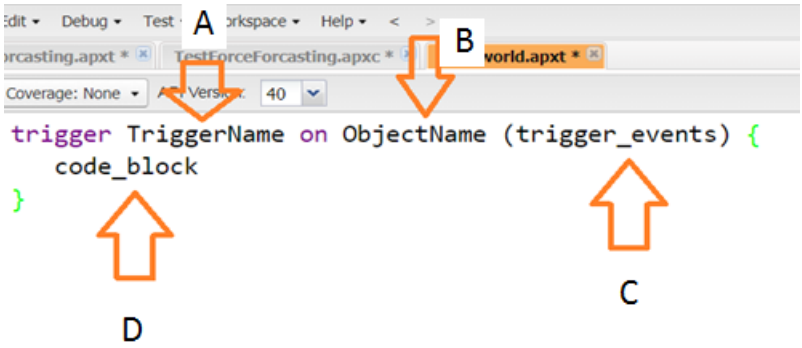
Triggers are programs in Salesforce designed by developers to execute certain logic when the trigger is true. These are custom actions, which can execute before or after changes to Salesforce records. (if the answer is yes, execute answer no, don’t execute) (Salesforce Developer Guide, 2019).

Example code

```
public class Yogurt {
```

```
public static void applyDiscount (yogurt__c [] yogurts) {
    for (Yogurt__c y: yogurts) {
        y.price__c *=0.85;          }
                                                    }
    }
```

Code Excerpt 3 Basic Trigger (code created from salesforce trailhead trail course). I put curly brackets under each pair to denote pairs. Explanation: this sample code would give a discount of 85% to all yoghurts. (Code Excerpt 3)



Code Excerpt 4 Basic Trigger, (Salesforce Trailblazer Community, 2019)

Every trigger starts with 'trigger' then, denoting on what it does.

(A) Trigger name, which should be as descriptive as possible, both one's own piece of mind (the writer) and for one's co-workers. (Nguyen, 2018)

(B) ObjectName – this is where the trigger is going to fire, only in this Object. A sObject is usually an account, subject, or opportunities, (Salesforce Objects) sObjects that are standard (reserved) within Salesforce, and these are usually not custom objects, as they do not update, as updates happen. In order to make this thesis coherent and to focus on Timed Delay Data management in Salesforce, custom objects are not discussed in this work. (Salesforce Trailhead, 2019)

(C) Trigger events - the whole trigger system will not work without code or the trigger, to this end a single use case scenario. These are Basic commands such as insert, update, delete, merge upsert (both update and insert) and undelete (Salesforce Developers, 2019) (Salesforce Trailblazer Community, 2019).
(D) Code block, this is where the instructions of the Trigger are placed, commands issued automation occurs within Salesforce. Along with the instructions from the object name where it is executed. Before executing this trigger, it must be safe to use. Assuring the safe use, a test class is run, before moving it into production (Salesforce Trailhead, 2019).

2.4.1 Comparing Triggers and Batches

The main difference between triggers and batches is that triggers are synchronous, in relation to the commands that they execute after them. Triggers can also fire asynchronous events (Salesforce Developers, 2018).

Batches are asynchronous. In practice, once a batch is run or is running one must wait for the result. All the resources are taken while the said process is being executed (Apex Developer Guide, 2019). Batches, as the name implies, executes blocks of accounts (in groups of 200) to execute commands. Asynchronous, denoted with the @future syntax signifying that it will be done in the asynchronously, by using a separate thread and queued. This does not require an immediate interaction; one can work on something else. The result is usually the completion of a mundane task such as deleting old files, sending emails.

Triggers are only executed when the conditions are true. For example, setting triggers to fire when 100 emails have been processed, to save queries would be trigger Email Trigger on Bulk Email (trigger_send) code block (thanks, your feedback is important to us, we have 24 hr. service online at dummywebsite.madeup.com

2.5 Governor limits

At this point a more detailed look into Governor limits is needed. Since bandwidth, is a premium. Sending ambiguous statements to the cloud to collect data for a minuscule number of candidates is a waste of time and resources (See appendix 2 Governance Rules, Figure 12,13.).

When the synchronous and asynchronous times are compared, the time needed for asynchronous is almost double. This is because these can be done when the server is at its quietest time and do not require the user to wait for the result. For batches, these are targeted towards synchronous commands as opposed to Asynchronous commands. Asynchronous is also used for another reason. As online services vary from place to place and information is vital, updating all the time takes energy and bandwidth, when outside large cities and suburbs will tend to get a lot weaker. Asynchronous data can be used when there is bandwidth to synchronize your data when in contact with a hot-spot. These can be joined with batches to create an asynchronous file sharing service when in reach of a better signal.

2.5.1 Breaking governor limits

Overall, it is needed to get as much information as possible through as fast as possible. However, one must be careful in order not to exceed the governor limits.

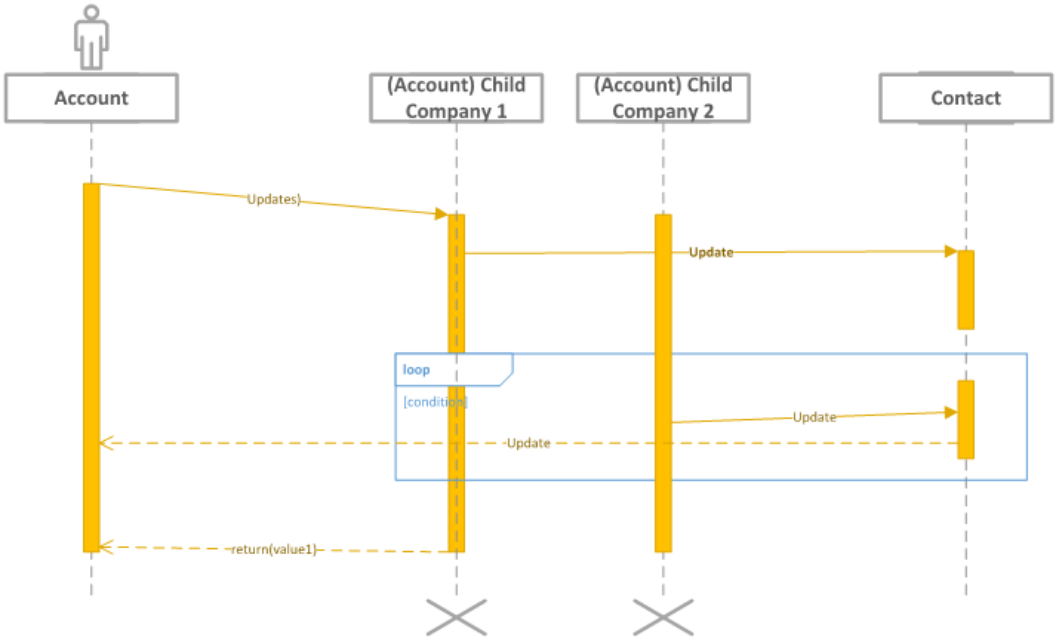


Figure 1 Contract updating account.

1) Updates the Account.
2) The child account(s) get updated.
3) The contract then gets updated.

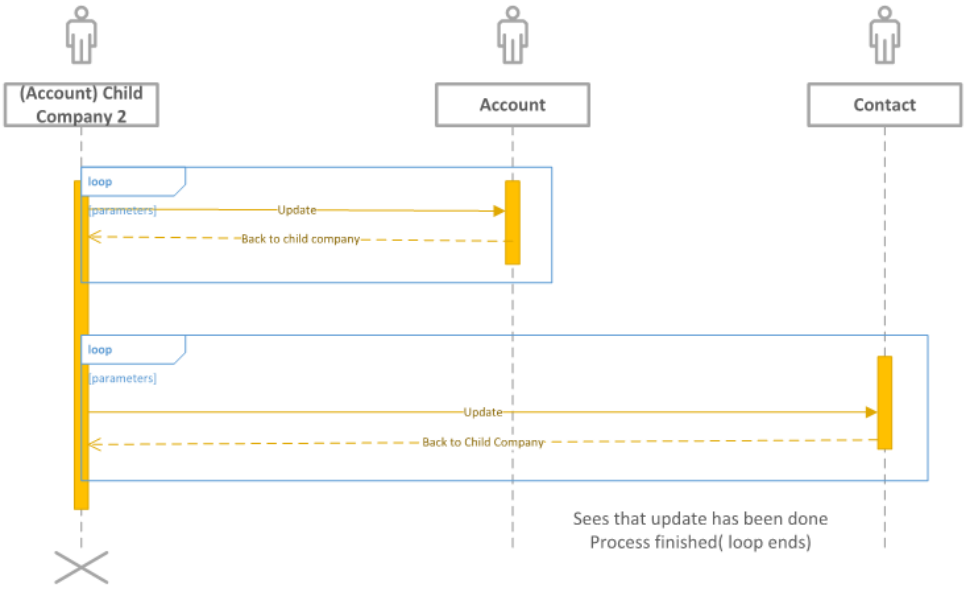


Figure 2 Updating simultainously.

1) The Account Child company gets updated
2) This updates the contract and the Account (at the same time)
3) The Contract and Account get updated, goes back to the Child Company (sees that the update has been done, the process finished). (loop ends)

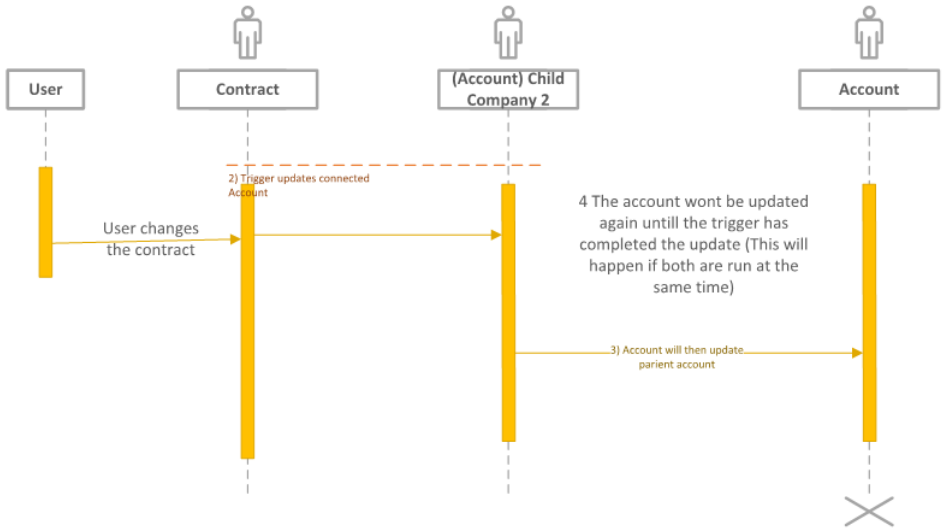


Figure 3 Using a trigger.

A Trigger:

1) User changes the contract
2) The Trigger in the contract will update the Connected Account
3) The Account will then update the parent Account
4) The account won't be updated again until the trigger has completed the update (This will happen if both are run at the same time)

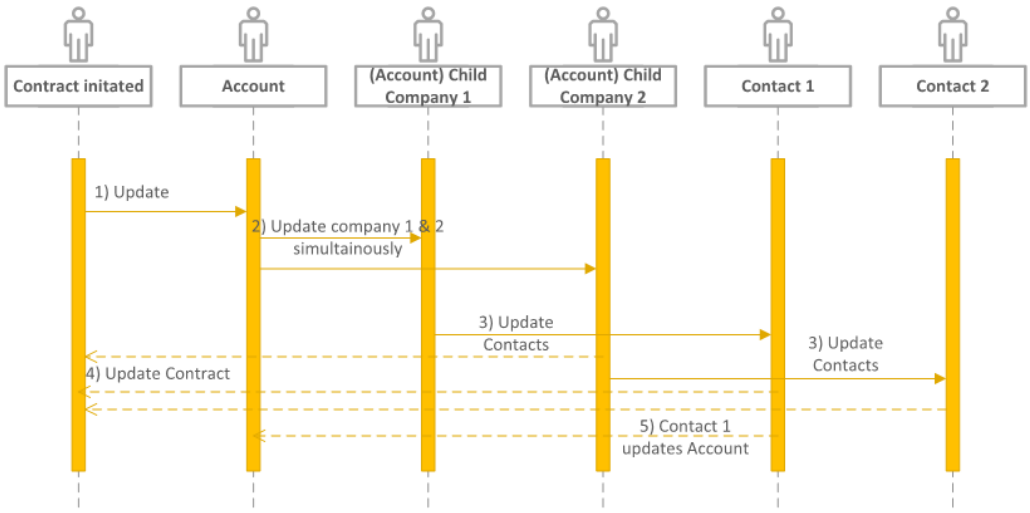


Figure 4 Endless loop of updating.

Triggers update before or after whenever something is inserted, updated, deleted merged upserted or undeleted.

1) The contract in the middle starts the snowball effect by updating the Account
2) Which updates the two child accounts (company 1 and 2)
3) Which updates their contacts
4) Which update the Contract (in the middle) and the contact from company 1 updates the Account.
5) The contract (who initiated this) is updated once (and now again)

2.5.2 Governance and its limitations considered

Governance ensures that all the requested information is dealt with in an equal manner. Since queries are required to address the information in an orderly fashion, the user can only pass a certain amount of commands at once. Before the governor limits have been exceeded.

But there is a way of using your data limit in a productive and an efficient way. Loops can call edit and replace information which done inefficiently would be four calls but using a loop this can be done in one. Thus, in order to be more efficient it is wise to use bulk triggers inside of the loop, or with the loop to keep the query level below the governor limit. Doing that can save our limit and execute upwards of 100 calls at the same time. Storing Actions are used when there is latency on the network 3G or unreliable coverage. They can also be used when wanted to limit a server trip. Then all data that is stale will be refreshed from the server (FOSCO, 2019).

In the process of writing this thesis, I also took part in a training program of Salesforce. The following example (below) is one task that I completed as part of the training program. It demonstrates Manipulate Records with DML, where I had to add extra clients to their database. It was also requested to debug code to check the heap size (Salesforce Trailblazer community, 2017).

```
//Check the heap size before running dml
System.debug('Before running dml, heap size is '+ Limits.getHeapSize());
// create a list of contacts

List<Contact> conList = new List<Contact> {
      new Contact (FirstName='Joe',LastName='Smith',Department='Finance'),
      new Contact(FirstName='Kathy',LastName='Smith',Department='Technology'),
      new Contact(FirstName='Caroline',LastName='Roth',Department='Finance'),
      new Contact()};
```

```
// Bulk insert all contacts with one DML call
Database.SaveResult[] srList = Database.insert(conList, false);

// Iterate through each returned result
for (Database.SaveResult sr: srList) {
    if (sr. isSuccess()) {
        // Operation was successful, so get the ID of the record that was processed
        System.debug('Successfully inserted contact. Contact ID: ' + sr.getId());
    } else {
        // Operation failed, so get all errors
        for(Database.Error err : sr.getErrors()) {
            System.debug('The following error has occurred.');
            System.debug(err.getStatusCode() + ': ' + err.getMessage());
    //Check to see the stack after the code has been run.
 System.debug('Contact fields that affected this error: ' + err.getFields());
                    }
    }
}
```

Code Excerpt 5 Manipulate Records with DML (Apex Developer Guide, 2019), (Salesforce Trail-head, 2019).

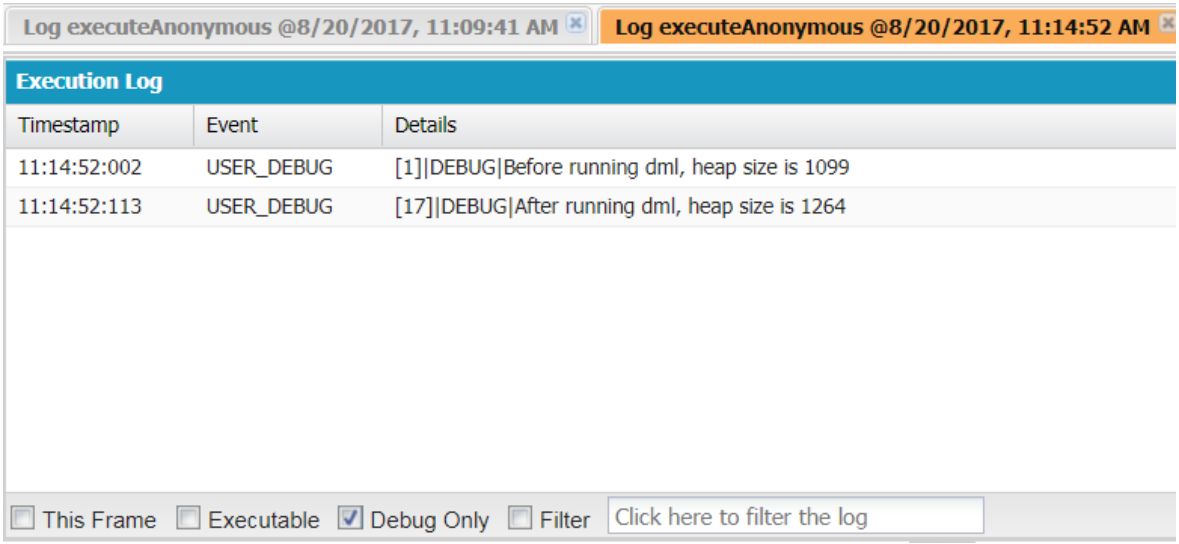


Figure 5 Execution of Triggers.

As outlined in the example above (Figure 5 Execution of Triggers), there are different heap sizes before and after the account was processed. This is a feature that must be monitored, as Salesforce is a multitenant platform where many people access the same data simultaneously. Editing the settings of one person in an efficient use of computational power. This would be a safe way of never exceeding the governor limits. One could put this into a Trigger, to do this automatically.

Execution Log

Timestamp	Event	Details
11:20:24:000	LIMIT_USAGE_...	Number of query rows: 0 out of 50000
11:20:24:000	LIMIT_USAGE_...	Number of SOSL queries: 0 out of 20
11:20:24:000	LIMIT_USAGE_...	Number of DML statements: 1 out of 150
11:20:24:000	LIMIT_USAGE_...	Number of DML rows: 1 out of 10000
11:20:24:000	LIMIT_USAGE_...	Maximum CPU time: 0 out of 10000
11:20:24:000	LIMIT_USAGE_...	Maximum heap size: 0 out of 6000000
11:20:24:000	LIMIT_USAGE_...	Number of callouts: 0 out of 100
11:20:24:000	LIMIT_USAGE_...	Number of Email Invocations: 0 out of 10
11:20:24:000	LIMIT_USAGE_...	Number of future calls: 0 out of 50
11:20:24:000	LIMIT_USAGE_...	Number of queueable jobs added to the queue: 0 out of 50
11:20:24:000	LIMIT_USAGE_...	Number of Mobile Apex push calls: 0 out of 10
11:20:24:000	LIMIT_USAGE_...	
11:20:24:018	CUMULATIVE_L...	

Figure 6 Limit usage.

In the example above, I demonstrated the number *Limit_usage* or number of *rows* that I was able to execute within the governor limits. At that time, those numbers demonstrating the possibilities inside the governor limits seemed quite large and almost impossible to exceed. However, it is good to remember that Salesforce is a multitenant platform, where the users are accessing the database that as well other companies are accessing at the same time. In sum: the multiple amount of user accessing the information at the same time increases the possible number of *Limit_usage* or *rows*, and thus the possibility to go over the governor limits.

3 Reconsidering the purpose of the limitations in Force.com

In chapter 1.2 I introduced the research question (*is it possible to bend or break the governor limits in Salesforce to be more efficient?*).

After studying the governors limits more, I can still state that it would be possible to break it. However, as demonstrated in *Figure 7 Endless loop of updating,* breaking the governor limits would also put one in an infinite loop. Meaning, the user would succeed in breaking the limit, but also cause an unwanted result as one's work would go to a standstill. This, in turn, will cause everything to be regenerated back to the original state. Meaning, no information will be saved, and everything is at the same stage as before breaking the governor limits in the first place. To demonstrate this, I show figure 4, where one of the many examples that would cause a governor limit breakage is shown.

The research question in this thesis is: is it possible to bend or break the governor limits in Salesforce to be more efficient? It is answered by analysing the following aspects.

I. Can one use batches and triggers to break or bend the governor limits?
II. Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?

How to become more efficient in Salesforce from my research, which involves the above-mentioned triggers batches synchronous and asynchronous methods.
Using triggers to automate everything, anything that is predicable, this includes responses, pay day letters, etc. all which take time if done manually, use triggers or batches of triggers this also can be done asynchronously, (when the server is quieter)
Sending bulk emails that are not top priority can be also used asynchronously,
Using company 'x' as a model I am writing best practices to be more efficient in working life by using the above-mentioned commands and batch of commands in a more economical way.

3.1 When and for what purpose Lightning and apex code can be used?

Lightning and Apex code are used together, as one. All the modifications can be done with a point and click method through lightning. Salesforce is updated quarterly. Then the finetuning will be done in Apex, the Apex code which sits upon the Salesforce platform. Apex is used to fine tune the customization of an app in order to make it unique for the customer. For example, using triggers to execute when a new customer is introduced into the system, or just thanking them for visiting the page that was created. In the future Apex must be proofed enough to withstand the quarterly updates that occurs in Salesforce.

3.2 DML / Query

DML (Data manipulation language) is an API that is an application. It plugs one directly into the services or grants access to certain parts of the server. An API allows independent software's to share information.

DML is written either in bulk or in single operations. The list of sObjects includes some sObjects that are not supported or do not support DML operations. Salesforce provides a ready-made list of these not supported SObjects (Apex Developer Guide, 2018) Also, in order to the be possible to edit the sObject, it must have the corresponding property set to true.

It is good to note that process Instance cannot be updated, deleted or created. Doing bulk DML operations avoids hitting the governor limits, (Apex Developer Guide, 2017) as DML limit is 150 statements per Apex transaction. Doing bulk Operations would count as one DML statement for all, instead of one for each. Before manipulation, the record data is created in memory as a sObject. This way it will be counted as one, like a file with many files inside DML for writing SoQl reading. Also, in Salesforce, there is an all or nothing operation (optAllOrNone). All the records must be updated successfully for the rule to be a success or none of the files will be updated. Performance, Memory usage, Process time (Apex Devloper Guide, 2017).

3.3 Considering bulky code

DML and SOQL as means to increase efficiency in the Salesforce / as best practices bulky code makes sure that code can handle more than one record at once. This can also be useful when thinking about the efficiency in the Salesforce.
In order to avoid loops, DML and in SOQL can be used (Apex Developer Guide, 2019). As database operations, they executed once per iteration of the loop. These can easily add up to reach the governor limits. Also, they move any query outside the loop.

3.4 Considering Error Handling in Salesforce

As stated, (Apex Developer Guide, 2019) an exception is an error in code, as it changes the direction of the natural flow. When an error or unusual turn in the code is encountered, it is usually a human error. It can happen in cases such as having incorrectly assigning nothing to a variable, when it expects to see a text or Boolean statement.

The following three conditions can cause Apex to raise, or throw, an exception:

- The most common examples are cases where: the code expects a value or text.
- a missing link in code (updating something that does not exist)
- querying something that doesn't exist

In all these three instances it is tried something that the language deems impossible. Thus, an exception is thrown. Moreover, Apex has many kinds of exceptions (Apex Developer Guide, 2019). But since they are all subclasses from a generic exception class, they can be dealt with a similar way. All the exceptions support standard methods for accessing the error message and the exception type:

- Error handling in Apex in the Developer console and debug logs.
- Debug log can record, database, system processes and errors. That occurs when executing tests
    - They contain information on:
    - Database changes and Queries loading below average apex triggers
    - HTTP callout issues and latency

- Resources by Apex
- Automated workflow processes
- There are filters to the debug log or log Levels ranging from none to Finest, the most common would be an error, warn, and info and Debug. The last three Fine, Finer, and Finest could overwhelm one with too much senseless information. Also, take longer to run.
- Complex Apex problems: As with multiliteracy it all adds up, running

4 Summary of the analytical model for company ‘x’ demo

In this part, I am outlining a canvas model of a basic product. I am using it in my company ‘x’ demo because it contains all essential elements to get every project started. It does that by recycling code that they use in every project at the start. The idea behind this is that most if not all various way of adding loops, asynchronous synchronous as well as batches and triggers in all formats have already been written. Therefore, it is just a case of recycling various bits of information that has already been used in previous projects to get the wanted results. This is also efficient way of re-coding as they already exist and have been tested.

4.1 The learning process before executing company ‘x’ demo

The hypothesis in this thesis is that the governor limits are getting in the way of efficiency. It was my understanding that they obstructed the flow of information that could be used. Also, it was considered that that governor limits were there to just to protect the server on the backend from being overloaded.

Assuming that the amount of information that a company could use was based on how much it paid for using the services in Salesforce. which bought me to the assumption that profit was the underlying factor – and hence the governor limits were created. Originally, I wanted to prove this case.

However, getting familiar with relevant readings, talking and interviewing people, I concluded in testing, I had to face my own preconceived idea that these rules were monitory grounded. That an American company was just interested in making profit by setting up governor limits in the service. Studying and learning about I.T.

security changed my perception on the security side, and I realised that the governor limits is more a security issue than a profit-making issue.

Together with this realization I also started to reconsider the aspect the Salesforces' infancy, associating the speed with the first computers, came to mind.

The Questionnaire that I came up with was to enable me to collect the relevant information. The Evidence that I needed to prove my case. The answers given were of poor value. As from a horde of 30 only five returned the answers, two of them were incomplete. But, in further interactions with the company employees, learning about their working habits, what programs they used, how often, code recycling efficiency. Gained more information I used the questionnaire as a step stone to further understanding.

5 Company 'x' Demo

Salesforce CRM's vast functional areas cannot be seen from a single or even a dozen use case samples. To this end, a single use case scenario has been selected that will allow the virtual journey of the usage of integrating Salesforce and its triggers, synchronous and asynchronous methods, displaying the ease and time-saving rewards.

Use Case: The Demo will create a contact and if the contact is not associated with an existing account then a trigger will an approval flow to request access for the user's manager.
To confirm acceptance that this user and/or his account (lead dependent) is confirmed. (Salesforce Trailhead, 2019)This link is a part of the learning process in Salesforce.
Flow: The approval process will send a custom email template that will have a concatenated link and once clicked, the batch/trigger will show a manual flow that asks the details of the missing account information. Next, the approval which will use a process builder that will call an automated flow that will create an account and send a confirmation to account owner, or original contact.

As stated in the chapter 2.1, I also wanted to analyse if one could use batches and triggers to break or bend the governor limits?

Using batches would of course be possible. But it would have the same effect as above: everything would regenerate to a time before the breakage.

Triggers different from batches, as explained earlier. To sum up, triggers are used when a certain command or sequence of commands are executed. When a customer is buying opera tickets, automatic response of 'thank you' message, and a receipt would be a trigger. As I wanted to analyse if triggers can be used to break or bend the governor limits, the answer is that they can be used to break it. This is demonstrated in the Figure 4 above. Using triggers is one of the many ways to break the governor limits. But, as stated before, breaks the limit puts one in an infinite loop.

Therefore, using batches or triggers is not a suitable solution on their own to get efficiency in Salesforce. It may/will cause breaking the governor limits, which is causing inefficiency as it takes us to back to the starting point.
I also wanted to analyse ii) Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?

As discussed in the previous chapters, using asynchronous methods to exceed the synchronous governor limit is one way of giving the illusion of being more efficient. This is because it is possible to use two tracks to execute commands (the two tracks are essentially one). When thinking about the efficiency aspect, it is important to remember that asynchronous does not need to be monitored. They will be executed after/when all the synchronous commands are executed or later, for example if a timing variable is used on that day. Differing from this, the synchronous side will execute instantaneously.

At this point it is useful also to reconsider my original hypothesis for this thesis (*the governor limits are getting in the way of efficiency*). The hypothesis that I will actually test in the company 'x' demo is that is it possible to be more efficient it the limits of governor limits, by

I also wanted to analyse ii) Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?

As discussed in the previous chapters, using asynchronous methods to exceed the synchronous governor limit is one way of giving the illusion of being more efficient. This is because it is possible to use two tracks to execute commands (the two tracks are essentially one). When thinking about the efficiency aspect, it is important to remember that asynchronous does not need to be monitored. They will be executed after/when all the synchronous commands are executed or later, for example if a timing variable is used on that day. Differing from this, the synchronous side will execute instantaneously.

5.1 Force.com Limitation considerations Empirical part

5.2 Force.com Governance and limitation considerations

Governance, DML, loops, SQL count, field maintenance check (isupdateable etc.) community usage (don't overdo on calls and queries a community will timeout for example.

Since Salesforce is a PaaS, there are multiple users in this platform. It can help to picture this governance rule like a speed limit on a motorway, along with police monitoring stations to make sure that those rules are followed. Failure to do so will be dealt with restrictions to the platform. Needing to address all the information in an orderly fashion, passing a certain amount of information, buikifying the code. Enabling safe updates without reaching the governor limits.

There is a way of using your data limit in a productive efficient way, using loops to call, edit and replace information which can be done inefficiently. Here four calls are being wasted on a loop whereas this can be done in one. Saving our limit, also using bulk triggers that can execute upwards of 100 calls at a time, would then create more efficient time savings. Storing Actions, are used when there is latency on the network 3G or unreliable coverage, also limiting a server trip, all data that is stale will be refreshed from the server.

In the example below, how to Manipulate Records with DML, a part of the training program, I had to add extra clients to the database. If this was in a company of global size. Recording new prospects would be impossible due to the number of applicants and unwieldy changes from a day to day basis, database management would have to be automated, checking the heap size. Restoring and updating the database. The best case would be using triggers and batches to contain the over-flow of information the rapidly shifting information of personnel being hired and fired by the company.

```
//Check the heap size before running dml
System.debug('Before running dml, heap size is ' + Limits.getHeapSize());
// create a list of contacts
```

```
List<Contact> conList = new List<Contact> {
        new Contact(FirstName='Joe', LastName='Smith',Department='Finance'),
        new Contact(FirstName='Kathy', LastName='Smith',Department='Technology'),
        new Contact(FirstName='Caroline', LastName='Roth',Department='Finance'),
        new Contact()};

// Bulk insert all contacts with one DML call
Database.SaveResult[] srList = Database.insert(conList, false);

// Iterate through each returned result
for (Database.SaveResult sr : srList) {
    if (sr.isSuccess()) {
        // Operation was successful, so get the ID of the record that was processed
        System.debug('Successfully inserted contact. Contact ID: ' + sr.getId());
    } else {
        // Operation failed, so get all errors
        for(Database.Error err : sr.getErrors()) {
            System.debug('The following error has occurred.');
            System.debug(err.getStatusCode() + ': ' + err.getMessage());
    //Check to see the stack after the code as been run.
 System.debug('Contact fields that affected this error: ' + err.getFields());
                    }
    }
}
```

Code Excerpt 6 Updating contact list

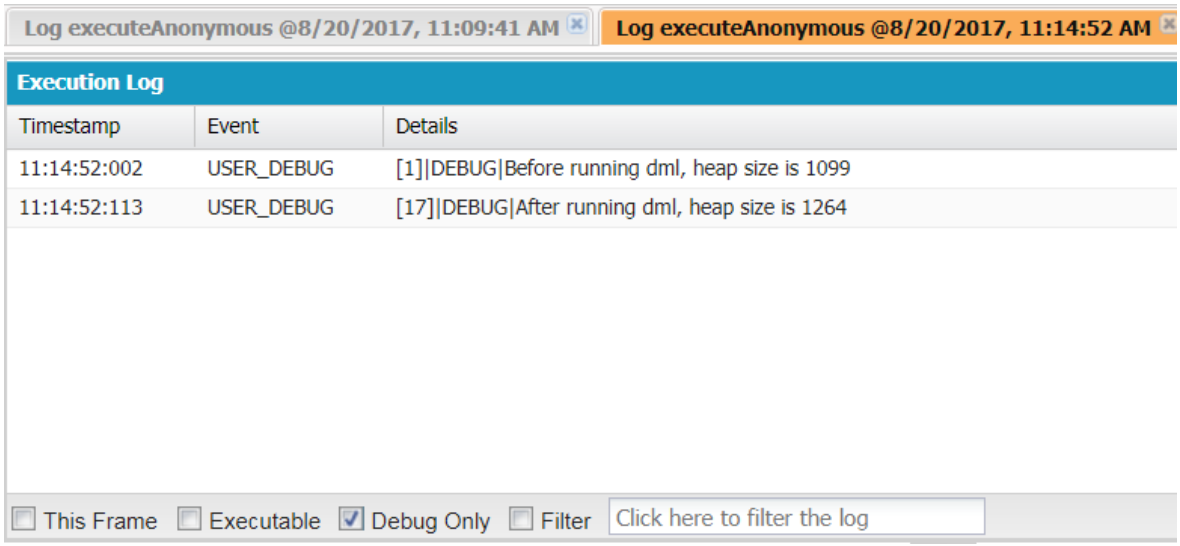


Figure 8 heap size (Trailblazer Community, 2019)

As one can see there are different heap sizes before and after the account was processed, This seems very small or, if one is new to Salesforce and to data usages in general, one does not really care what those numbers are if the code works. But this is just one person being added. If this would be (if done incorrectly multiplied by 10,000, the figure would break the governor limit. Thus, would not be run. One's program would be rolled back (to prevent breaking the cloud).

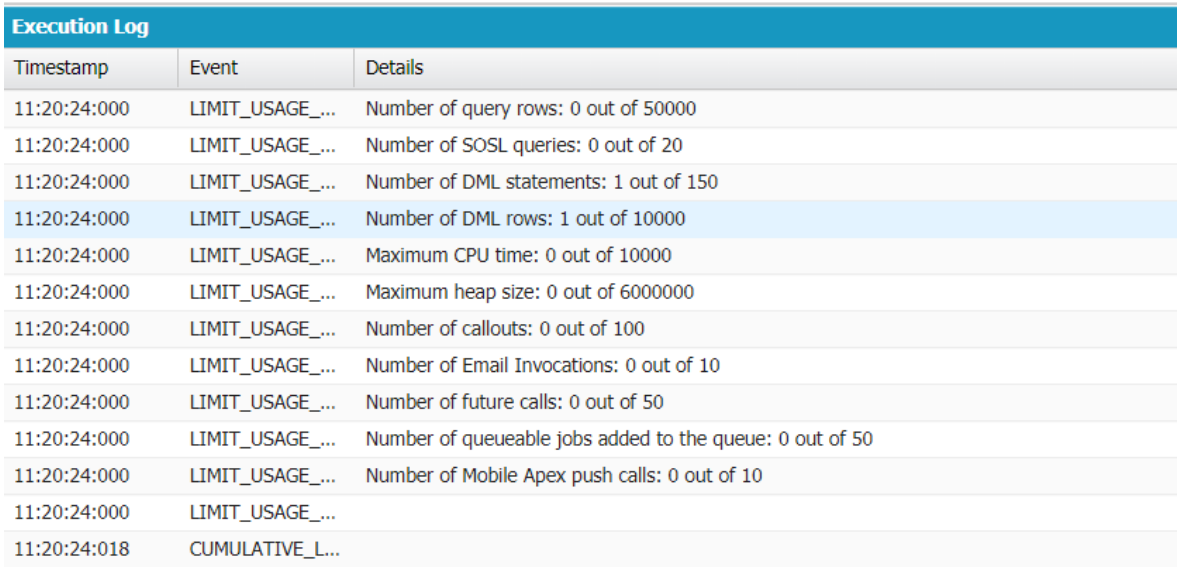


Figure 9 Answers to stack (Stack Exchange, 2019)

Next, I used the number Limit_usage or number of rows. Now, at this moment, these numbers seem quite large and almost impossible to break or to use up. Nevertheless, as stated before, one is not alone, thousands of people or hundreds of companies. This is where care is required with data handling. Here we are trying to acknowledge that it is not efficient to put one person at a time into the database.

5.3 DML / Query

DML (Data Manipulation Language) is an API that is an application, which, plugs one directly into the services or another by granting it access to certain parts of the server. An API allows two strange or different software to share information.
DML is written either in bulk or in a single operation, on a list of sObjects, there are some sObjects that are not supported or do not support DML operations, This is again beyond the scope of this thesis but to note, that there is a list. (Apex Developer Guide, 2017). In addition, in order to be edited, the sObject must have the corresponding property set to true. Aside: process Instance cannot be updated, deleted, or created. Doing bulk DML operations avoids hitting the governor limits (DML limit is 150 statements per Apex transaction). Doing bulk Operations would count as one DML statement for all, instead of one for each. Before manipulation, the record data is created in memory as a sObject (counted as one, like a file with many files inside) DML for writing SoQl reading. Also, in Salesforce, there is an (optAllOrNone) all or nothing operation (the brackets here signify a command) all or none command. All the records must be updated successfully for the rule to be a success. Otherwise, none of the files will be updated
(Apex Developer Guide, 2019).

5.4 Building better queries

Code Bulkification are used as code when it is wanted to use a code to query multiple queries as opposed to one or two. Bulkify code exists to make sure that code can handle more than one record at once. This is also be useful and efficient. Bulkify code, means that one can run correctly process more than one record at a time (Salesforce Developers, 2009). As this is a multitenant platform this is used and is designed to be used by the masses, one has to think in the bigger picture, think companies and not people. As when one is changing or updating it is a waste

to just update one figure when one can wait and update all figures in the same query. This leads to more efficient programming and query usage.

Avoiding For loops, in DML and in SOQL as database operations executed once per iteration of the loop. These can easily add up to reach the governor limits. Move any query outside the loop. As the loop is being executed the query is also being used. To have the query outside the loop will retain the usage until the loop has been completed, thus if the loop runs 100's of times the query will be one as opposed to 100.

5.5 Exceptions

An Exception is an error in code; it changes the direction of the natural flow. When it encounters or takes an unusual turn in the code, this is usually a human error, such as having incorrectly assigning nothing to a variable, when it expects to see a text or Boolean statement (Apex Developer Guide, 2019). (Apex Developer Guide, 2017)

So, what kinds of conditions can cause Apex to raise, or throw, an exception? Below are listed the most common examples. In all these instances, one is trying something that the language deems impossible, thus an exception is thrown. Apex has (Apex Developer Guide, 2017) that is many kinds of exceptions. But since they are all subclasses from a generic exception class, they are very similar to deal with. All the exceptions support standard methods for accessing the error message and the exception type.

- Error handling in Apex in the Developer console and debug logs.
- Debug log can record, database, system processes and errors. That occurs when executing tests
    - They contain information on:
    - Database changes and Queries loading below average apex triggers
    - HTTP callout issues and latency
    - Resources by Apex
    - Automated workflow processes

- There are filters to the debug log or log Levels ranging from none to Finest, the most common would be an error, warn, and info and Debug. The last three Fine, Finer, and Finest could overwhelm one with too much senseless information. Also, take longer to run.
- Complex Apex problems: As with multiliteracy it all adds up, running programs, running batches and Apex code(s) backing up systems, all take-up time
- Network Connectivity would be the first place to look if there was a problem, Latency and backup problems would influence performance

(Salesforce Trailblazer Community, 2017)
(Apex Developer Guide, 2019)

6 Company X Demo – summary of results

To this end a single use case scenario has been selected that will allow the virtual journey of the usage of integrating Salesforce and its triggers, processes & flows, batch processes alongside the needed apex and lightning component (UI display dependent) displaying the ease and time-saving rewards.
Use Case: The Demo will create a contact and if the contact is not associated with an existing account then a trigger will an approval flow to request access for user's manager to confirm acceptance that this user and/or his account (lead dependent) is confirmed.
Flow: The approval process will send a custom email template that will have a concatenated link and once clicked will show a manual flow that asks the details of the missing account information and after that the approval which will use a process builder that will call an automated flow that will create an account and send confirmation to account owner that being the original contact.
Starting off: with a meeting with company X want a proof of concept and environment set up various accounts (being a medium sized sales company going from Microsoft excel, word PowerPoint Slack. Required for information that will house all our communications emails, drafts. notes etc. coordinated. Salesforce seems to be the best option. Also, the sales department is the first to be optimized. then, in

time the rest of the company. This is then easily done as Salesforce has many assets that can be integrated since this is all Salesforce everything is seamless.

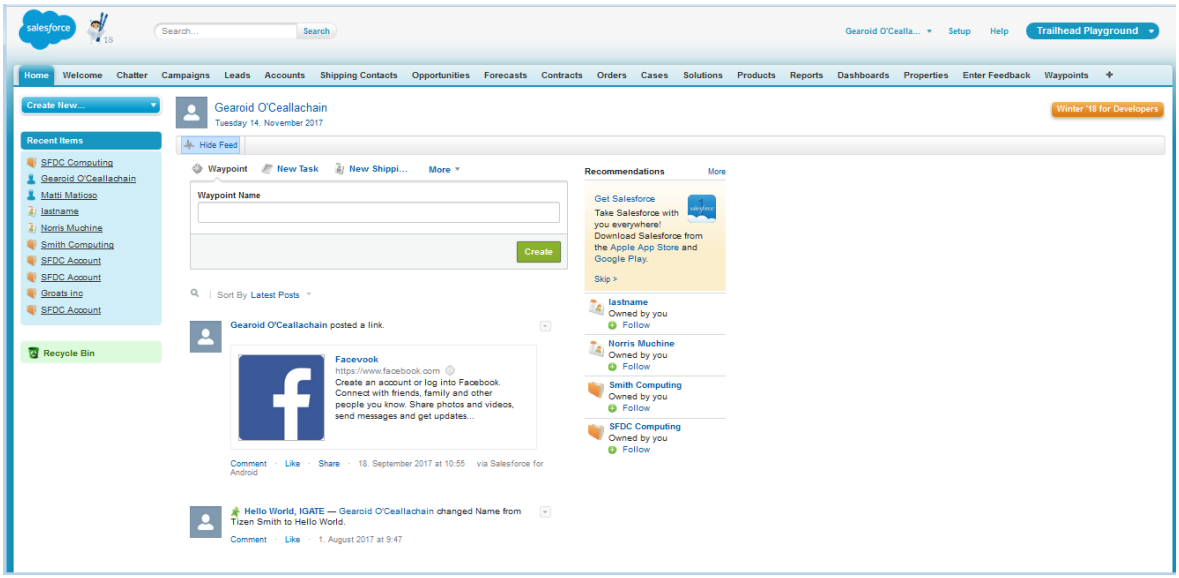


Figure 10 Unique landing page (Ledgeview partners, 2015)

The picture above is the best way to explain Salesforce. It proves how big, or detailed Salesforce is. Being a user of PaaS requires you to understand what is being managed by Salesforce as a service, and what you must manage. This is used mainly for applications that are written by the developer(s) for companies allowing them to build upon the operating system supplied by the service, thus the service involved servicing the updates storage and infrastructure. There are differences between various services between (see below) starting off from, on-premises, where none of the cloud computing is offered as a service. Going to the other side of the scale of Software as a Service where everything is being taken care of.

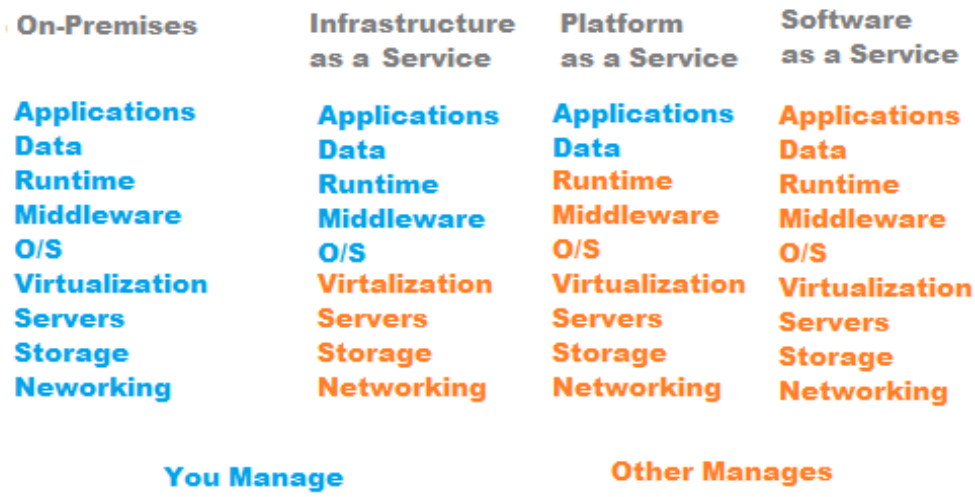


Figure 11 Differences between PaaS and private system.

7 Conclusion

The research question in this thesis is: is it possible to bend or break the governor limits in Salesforce to be more efficient? It is answered by analysing the following aspects.

I. Can one use batches and triggers to break or bend the governor limits?
II. Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?

Batches and triggers on their own can break the governor limits as I have proven in the above figure 5 (without code) but with governor limits the action afterwards, is everything just gets reset. This is to help Salesforce to maintain continued service. So, everything keeps on working.

As a result, it is discovered that the governor limits are means to keep order. As there are several ways of doing anything one wants in Salesforce provided one does not need go over the governor limits or at least not do it all the time. Moreover, they are means to keep clients safe with the Salesforce product. As a self-guiding rail to near perfect code, governor limits maintain undisturbed service every time Salesforce is updated.

Answering part two of the question,
Can one use asynchronous and synchronous methods, including batches and triggers, to create the illusion of getting more information from the governor limits?
The Answer is yes: to code efficiently, to divide the tasks into priorities
The lesser for asynchronous: and anything above would demand synchronous using batches and triggers plus batches to cut the work load for the user.
Triggers could be used for the mundane tasks that require little options either true or false statements. This would then build on further statements cutting down the workload. This would create efficiency in the work place.

Therefore, the original hypothesis of the thesis (the governor limits are getting in the way of efficiency) was proved inaccurate. Salesforce is a service where the consumers have been given the possibility to make it their own, as Salesforce gives us a platform that works the same way for everyone. It gets updated at regular intervals, but the users need to make it work for them. With Apex, the users can customize Salesforce as they wish, for example, to execute programs, Keeping it compliant with updated software. This is where the governor limits come in. They help us keep our standard high.

I thought that governance limits would get in the way of work and cause a problem by putting in too many breaks in the workflow. However, during the process I have noticed that unlike other on-premises platforms that can be updated by the I.T.-department or by the private individual, Salesforce has no involvement or needs any local IT configuration (Burns, 2015).

Governance limits are more than just speed restrictions on the Platform. They are there to ensure that everyone is in the safe confines of the service that is being delivered, that everyone has a fair share (and no more) than everyone else. They are there to ensure that everyone does not let the standard of their code diminish, by having a standard (governor limit) these will always be above what the average.

To sum up: it is easier and more efficient to adhere to the governance rules that you respect. Knowing that they are there for the ease of data flow. This it gives us space and freedom to do what we were set to accomplish without restriction. Starting this Thesis, I thought that the governance rules were created to restrict our movements, to save on backend workload, that we will be able to cross the limit and be more productive.

8 Discussion

PaaS is the way that all computer will be running in the future, as it will make it easier to upgrade everything at once. Relying on each computer to do its basic upgrade leaves too many security holes. Upgrading one master copy and distributing it via PaaS will solve this problem. Then the entire hard drive can be used for just storing information that needs to be used locally.

Salesforce is one of the biggest PaaS software companies out there, they give near perfect service. Salesforce encourages each company to create their own governor limits (Burns, 2015) starting small and working up from there. Since I.T. is not needed in Salesforce everything needs to be kept in check. But with a sandbox that one can do anything with, there is also a danger that one will drown. One can learn how to do it yourself, but it might not be cost effective. It may take a newcomer to Salesforce half a day to enter a customer and even then, it may not be done right. From an experienced user it would take only five minutes to execute simple task like this and it would be guaranteed to work. Therefore, although with skill comes a cost, it is useful to have guidance from someone who is well versed in Salesforce.

To stay on that track and to move forward at the speed that we set for ourselves. The benefits include innovation, each company giving each consumer access to create at their own pace, testing out in a sandbox, making sure that all the errors are fixed, Being compliant to the Salesforce platform, also means that if one company has software that works, it will work on your system too, without out any editing, this works for all aspects of I.T. from security, drive access and data visibility.

Salesforce Allows a real bond between I.T. and business both meet, defining roles creating challenges and opportunities implementing enhancements, by viewing analysing problems and solving them. Thus, creating a backlog (programs that are not up to standard) and implementation, the programs that are going online. Using Salesforce eliminates the need for operating systems, infrastructure or updates. PaaS allows and encourages any business to create applications without the worry of scalability (Watts, 2017).

Like all things in the industry, the best practices and implementation methods have a short refresh cycle, and what was standard at the time, starting this thesis is slightly different currently. A good example are triggers in Salesforce, that do not have any business logic (a.k.a. code present but a call to a utility class). Main reason for this shift was multiple triggers on the same object with multiple functionalities which consolidating in utility class is best practice. This is a very good example of governance in general as well as company's constant product delivery assurance.

8.1 Problems encountered

The first thing that made me realize that I was over my head was the number of Acronyms and abbreviations that existed in Salesforce. It seemed like a new (and is) language to me. Also, as Salesforce is an American Company, shortening of words come naturally. Learning the acronyms and language was just one of many hurdles that I had to overcome; it seemed the more I learned the less I knew. During the process one question on Salesforce just opened a door to several other questions.

Logging into salesforce for the first time and subsequent times, I expected all webpages to be different, to be unique to each other, this was not the case. Which caused confusion. Salesforce, one must look at the URL in order to find out which passcode to use.

Also learning how to use Salesforce required me to log into a Salesforce training program called Trailhead. Through that I was learning how to use Salesforce and

in order to complete the training program, I had to complete some onboarding tasks to get myself acquainted with the whole salesforce experience. During that I had to set up a profile with a name automatically assigned to you along with numbers. These sites you cannot enter but are public as it is the authors profile. Therefore, by following the same profile, (Trailhead, 2017) anyone can have the same access.

My thesis supervisor in company ‘x’ did help, but it was difficult to know the right questions to ask at the beginning. Perhaps it would have helped to have an A.I. supervisors (online artificial intelligence) to sift all the relevant information without overloading the supervisor with irrelevant requests. But on the other hand, direct contact with the company and especially with the supervisor in the company ‘x’, allowing me to formulate my research question according their needs and to search for a more detailed answer.

As mentioned earlier, collecting information via questionnaire turned out limited answers. But it gave me a view to how people work, what programs they use every day and in every project. This was one of the reasons why this thesis was written, to ascertain on the core programs and core code that could be recycled time after time, to re-create any project without having to start from the start. Once one could have the essential tools and code in front of them, they would be ready to explore the next project, and have it finished without problems. This would give a head start in every project a solid foundation on which to build.

Trying to jump ahead of the course by starting my thesis without going through the proper procedure was a very large mistake on my behalf, now showing on the writing skills that one sees here. But on saying that, it was a mistake that I feel if I didn’t do now, I would sure to do it at a later date.

At the more practical level I was facing come difficulties with references. As I was studying and saving my references in Mendeley and in Zotero, during the final round of editing Mendeley stopped working, I had to depend on word references, causing a massive delay in my thesis process.

I was also collecting cookies on my computer automatically. As each website collects cookies, also being an ordinary event in the lifetime of people who browse the web, research data. Salesforce being a PaaS also updates its information four times a year. Therefore, the webpages that I used earlier were up to date earlier, were now out of date. They were replaced with newer information but had the same webpage.

8.2 Further research

As I was answering the research question, it opened the view for other questions that needed answering. So, in answering the research question I had to leave some questions unanswered to stay on track with this thesis.

Since the questionnaire did not yield the information that I required to having the answered desired and thus, creating a database of regular commands used, in an above average use. This could save the company ‘x’s time and effort. The only thing needed is to pool the resources together and create their own database in order to bring the code together.

Creating a chart to formulate the efficiency curve between recycling code and not recycling code, or the percentage code recycled (Frakes & Terry, 1996)

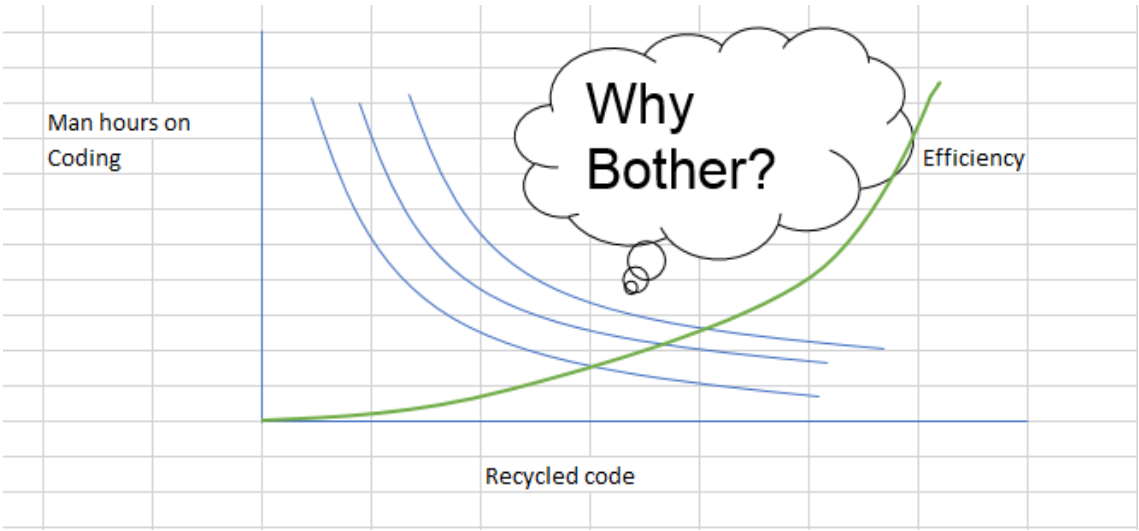


Figure 12 Efficiency Curve

As all code that have been written have been tested, there is no need to write new code. It is just a matter of recycling parts of the code to create your new program. Just connecting the code together. (Figure 11 Efficiency curve) The recycled code itself would be in a database ‘barebone tested code yard’ requiring the coder to pick and snap code fragments together to create the program required. It would only require of testing the near 100% tested code.

Thanks to my thesis coordinator Juha Pispa who asked the question ‘which one are you trying to figure out?’ (graph to show what he meant).

9 References

Apex Developer Guide, 2013. *Execution Governors and Limits.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm
[Accessed 5 April 2017].

Regalado, A., 2011. *Who Coined 'Cloud Computing'?.* [Online]
Available at: https://www.technologyreview.com/s/425970/who-coined-cloud-computing/
[Accessed 08 May 2019].

Salesforce Trailblazer community, 2017. *Build better apex scripts to manage heap limits.* [Online]
Available at: https://help.salesforce.com/articleView?id=000170956&type=1
[Accessed 08 May 2019].

Ahmad, R. & Janzewski, L., 2010. *Triangulation Theory: An approach to mitigate Governance Risk in Clouds.* Aukland, Management, Business School, University of Aukland.

Apex Developer Guide, 2017. *Apex DML Operations.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.206.0.apexcode.meta/apexcode/apex_dml_section.htm
[Accessed 07 May 2019].

Apex Developer Guide, 2017. *Data Manipulation Language.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_dml.htm
[Accessed 07 May 2019].

Apex Developer Guide, 2017. *Exceptions in Apex.* [Online]
Available at: https://developer.salesforce.com/page/An_Introduction_to_Exception_Handling
[Accessed 07 May 2019].

Apex Developer Guide, 2017. *SOQL For Loops.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_loops_for_SOQL.htm
[Accessed 08 May 2019].

Apex Developer Guide, 2017. *Understanding Testing in Apex.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing_intro.htm?search_text=Understanding%20Testing%20in%20Apex
[Accessed 08 May 2019].

Apex Developer Guide, 2018. sObjects That Don't Support DML Operations. In: A. D. Guide, ed. *Salesforce Apex Developer Guide.* San Francisco: Salesforce, p. 137.

Apex Developer Guide, 2018. *Using Batch Apex.* [Online]
Available at: https://resources.docs.salesforce.com/212/latest/en-us/sfdc/pdf/salesforce_apex_language_reference.pdf

Apex Developer Guide, 2019. *Asynchronous Apex.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_async_overview.htm
[Accessed 07 May 2019].

Apex Developer Guide, 2019. *Debug Log.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_debugging_debug_log.htm
[Accessed 13 May 2019].

Apex Developer Guide, 2019. *DMLOptions Class.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_methods_system_database_dmloptions.htm
[Accessed 12 May 2019].

Apex Developer Guide, 2019. *Exception Class and Built-In Exceptions.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_exception_methods.htm
[Accessed 13 May 2019].

Apex Developer Guide, 2019. *Exceptions in Apex.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_exception_definition.htm
[Accessed 13 May 2019].

Apex Developer Guide, 2019. *Limit Methods.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_methods_system_limits.htm
[Accessed 13 May 2109].

Apex Developer Guide, 2019. *SOQL for loops.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_loops_for_SOQL.htm
[Accessed 13 May 2019].

Apex Developer Guide, 2019. *Using Batch apex.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_batch_interface.htm
[Accessed 7 May 2019].

Apex Developer Guide, 2017. *SaveResult Class.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_methods_system_database_saveresult.htm
[Accessed 08 May 2019].

Burns, J., 2015. *Introduction to Governance whitepaper.* [Online]
Available at: https://help.salesforce.com/servlet/servlet.FileDownload?file=015300000037bACAAY
[Accessed 14 January 2019].

FOSCO, 2019. *What is Synchronous Transmission and Asynchronous Transmission?* [Online]
Available at: https://www.fiberoptics4sale.com/blogs/archive-posts/95042950-what-is-synchronous-transmission-and-asynchronous-transmission
[Accessed 8 May 2019].

Frakes, W. & Terry, C., 1996. *http://www.dcc.ufmg.br/dcc/.* [Online]
Available at: https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/papers/acm96frakes.pdf
[Accessed 08 May 2019].

Grogg, S., 2016. *Quora.* [Online]
Available at: https://www.quora.com/If-the-front-end-of-my-web-application-is-synchronous-does-it-help-to-have-an-asynchronous-microservices-back-end
[Accessed 7 May 2019].

Hurwitz, J., Bloor, R., Kaufman, M. & Halper, F., 2017. *Working with Salesforce.com's Force.com Platform in Cloud Computing.* [Online]
Available at: http://www.dummies.com/programming/cloud-computing/working-with-salesforce-coms-force-com-platform-in-cloud-computing/
[Accessed 08 May 2019].

Ledgeview partners, 2015. *Customize the Salesforce Home Page To Fit Your Needs.* [Online]
Available at: http://ledgeviewpartners.com/blog/customize-the-salesforce-home-page-to-fit-your-needs/
[Accessed 4 April 2018].

Margret Rouse, S. B., 2017. *Platform as a Service (PaaS.* [Online]
Available at: http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS
[Accessed 4 March 2018].

McCarthy, B., 2016. *A Brief History Of Salesforce.com,* London: EMPUA,UK.

Nguyen, L., 2018. *Hackernoon.* [Online]
Available at: https://hackernoon.com/how-to-write-clean-code-d557d998bb08
[Accessed 23 May 2019].

Salesforce Developer Guide, 2019. *Triggers.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers.htm
[Accessed 7 May 2019].

Salesforce Developers, 2019. *Mobile SDK Development Guide.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.mobile_sdk.meta/mobile_sdk/html5_intro.htm#!
[Accessed 09 May 2019].

Salesforce Developers, 2009. *Best practice: Bulkify your code.* [Online]
Available at: https://developer.salesforce.com/page/Best_Practice%3A_Bulkify_Your_Code
[Accessed 08 May 2019].

Salesforce Developers, 2016. *Apex Code best practices.* [Online]
Available at: https://developer.salesforce.com/page/Apex_Code_Best_Practices
[Accessed 10 May 2019].

Salesforce Developers, 2018. *Batch Apex Trigger.* [Online]
Available at: https://developer.salesforce.com/forums/?id=9060G000000XcD4QAK
[Accessed 23 April 2018].

Salesforce Developers, 2019. *Apex Developer guide - Triggers- Implementation Considerations.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers.htm
[Accessed 7 May 2019].

Salesforce Developers, 2019. *Salesforce Style Guide for Documentation and User Interface Text.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.salesforce_pubs_style_guide.meta/salesforce_pubs_style_guide/style_abbreviations.htm
[Accessed 13 May 2019].

Salesforce Developers, 2109. *Salesforce Developer Limits and Allocations Quick Reference.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_platform_apexgov.htm
[Accessed 13 May 2019].

Salesforce Discussion Forms, 2017. *Batch apex Sample code.* [Online]
Available at: https://developer.salesforce.com/forums/?id=906F0000000BOduIAG
[Accessed 04 June 2017].

Salesforce trailblazer Community, 2017. *Compare Lightning Experience and Salesforce Classic.* [Online]
Available at: https://help.salesforce.com/articleView?id=lex_aloha_comparison.htm&type=0
[Accessed 08 May 2019].

Salesforce Trailblazer Community, 2017. *Debug Logs.* [Online]
Available at: https://help.salesforce.com/articleView?id=code_debug_log.htm&type=5
[Accessed 13 May 2019].

Salesforce Trailblazer Community, 2019. *Define Apex Triggers.* [Online]
Available at: https://help.salesforce.com/articleView?id=code_define_trigger.htm&type=5
[Accessed 7 May 2019].

Salesforce Trailhead, 2019. *Batch Apex Syntax.* [Online]
Available at: https://trailhead.salesforce.com/en/content/learn/modules/asynchronous_apex/async_apex_batch
[Accessed 13 May 2019].

Salesforce Trailhead, 2017. *Salesforce & Heroku Integration.* [Online]
Available at: https://developer.salesforce.com/page/Extreme_Salesforce_Data:_Distributed_Application_Partitioning_with_Force.com_Canvas_and_Heroku
[Accessed 07 May 2019].

Salesforce Trailhead, 2019. *Automate Simple Business Processes with Process Builder.* [Online]
Available at: https://trailhead.salesforce.com/en/modules/business_process_automation/units/process_builder"
[Accessed 13 May 2019].

Salesforce Trailhead, 2019. *Example: Inserting Records with Partial Success.* [Online]
Available at: https://trailhead.salesforce.com/content/learn/modules/apex_database/apex_database_dml
[Accessed 13 May 2019].

Salesforce Trailhead, 2019. *Get Started with Apex Triggers.* [Online]
Available at: https://trailhead.salesforce.com/en/content/learn/modules/apex_triggers/apex_triggers_intro
[Accessed 7 May 2019].

Salesforce Trailhead, 2019. *Test New Releases in a Sandbox.* [Online]
Available at: https://trailhead.salesforce.com/en/content/learn/modules/financial-services-cloud-release-readiness/test-new-releases-in-a-sandbox
[Accessed 7 May 2019].

Salesforce Trailhead, 2019. *Use Batch Apex.* [Online]
Available at: https://trailhead.salesforce.com/en/content/learn/modules/asynchronous_apex/async_apex_batch
[Accessed 7 May 2019].

Salesforce.com, 2008. *Multitenant Architecture.* [Online]
Available at: http://cloud.pubs.dbs.uni-leipzig.de/sites/cloud.pubs.dbs.uni-leipzig.de/files/p889-weissman-1.pdf
[Accessed 4 May 2019].

Salesforce.com, 2008. whitepaper. In: Salesforce, ed. *WHITEPAPER the Force.com Multitenant Architecture Understanding the Design of Salesforce.com's Internet Application Development Platform.* San Francisco: Salesforce, p. 16.

Salesforce.com, 2017. *Salesforce Knowledge.* [Online]
Available at: https://help.salesforce.com/articleView?id=knowledge_whatis.htm&type=5
[Accessed 07 May 2019].

Salesforce, 2017. *Use Batch Apex.* [Online]
Available at: https://trailhead.salesforce.com/en/modules/asynchronous_apex/units/async_apex_batch
[Accessed 07 May 2019].

Stack Exchange, 2019. *How to best analyze limit problems in the Developer Console.* [Online]
Available at: https://salesforce.stackexchange.com/questions/57039/how-to-best-analyze-limit-problems-in-the-developer-console
[Accessed 05 May 2019].

Trailblazer Community, 2019. *Execution Log.* [Online]
Available at: https://help.salesforce.com/articleView?id=code_dev_console_view_system_log.htm&type=5
[Accessed 04 May 2019].

Trailhead, 2017. *Trailhead basics Get Started with Trailhead.* [Online]
Available at: https://trailhead.salesforce.com/modules/trailhead_basics/units/get-started-with-trailhead?trailmix_creator_id=005500000060MntAAE&trailmix_id=start-learning-today-with-trailhead
[Accessed 04 May 2017].

Watts, S., 2017. *SaaS vs PaaS vs IaaS: What's the Difference and How to Choose.* [Online]
Available at: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/
[Accessed 23 January 2019].

William, F. C. T., 1996. *https://homepages.dcc.ufmg.br/~figueiredo/.* [Online]
Available at: https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/papers/acm96frakes.pdf
[Accessed 8 May 2019].

Yeasmin, T., 2017. *Clouds and platforms.* [Online]
Available at: https://www.slideshare.net/TaniaYeasminPreity/introduction-to-salesforce-ppt-64301532
[Accessed 08 May 2019].

Figures and Code Excerpts

Bibliography

99, S., 2013. *What are governor limits.* [Online]
Available at: http://www.sfdc99.com/2013/11/17/what-are-governor-limits/
[Accessed April 2017].

Ali, 2017. *Web Services Architecture – When to Use SOAP vs REST.* [Online]
Available at: http://comtech2.com/web-services-architecture-when-to-use-soap-vs-rest2/
[Accessed April 2017].

Apex Developer Guide, 2017. *Understanding Testing in Apex.* [Online]
Available at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing_intro.htm?search_text=Understanding%20Testing%20in%20Apex
[Accessed 08 May 2019].

apps, n., 2017. *https://www.newgenapps.com.* [Online]
Available at: https://www.newgenapps.com/blog/game-face-on-using-gamification-to-build-a-top-app

B. Joseph Pine, J. H. G., 2011. *The Experience economy.* Boston: Harvard business review press.

Businessinsider.com, 2017. *https://recruiterbox.com/blog/gamification-in-recruiting.* [Online]
Available at: https://recruiterbox.com/blog/gamification-in-recruiting/businessinsider.com

Cloud Concept, 2017. *Cloud Concept First Lightning Ready Partner in Middle East.* [Online]
Available at: http://cloudconceptgroup.com/first-and-only-lightning-ready-partner-in-middle-east/
[Accessed 11 April 2018].

Csikszentmihalyi, M., 2008. Flow. In: *Flow: The Psychology of Optimal Experience.* New York: HarperCollins Publishers, p. 196. [Accessed 15 April 2017]

Douglas, G., 2017. *Difference Between REST and SOAP.* [Online]
Available at: http://www.differencebetween.net/technology/internet/difference-between-rest-and-soap/

Gupta, R., 2015. *Getting Started with Process Builder – Part 5 (Launch a Flow Automatically).* [Online]
Available at: https://automationchampion.com/tag/flows-action/ [Accessed 2018 April 2].

Jagani, S., 2016. *Lightning Experience vs. Salesforce Classic.* [Online]
Available at: http://www.alliancetek.com/Blog/post/2016/10/18/Lightning-Experience-vs-Salesforce-Classic.aspx [Accessed 07 May 2019].

Overflow, S., 2017. *SOAP Vs REST.* [Online]
Available at: https://stackoverflow.com/questions/19884295/soap-vs-rest-differences

Pombriant, D., 2010. *Well, i am a systems guy....* [Online]
Available at: https://www.enterpriseirregulars.com/30561/well-i-am-a-systems-guy%E2%80%A6/

Rouse, M., 2014. *SOAP (Simple Object Access Protocol).* [Online]
Available at: http://searchmicroservices.techtarget.com/definition/SOAP-Simple-Object-Access-Protocol

Rouse, M., Bigelow.S, 2017. *Platform as a Service (PaaS.* [Online]
Available at: http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS

[Accessed 4 March 2018].

Salesforce, 2017. *Automate Basic Business Processes with Process Builder.* [Online]
Available at: https://trailhead.salesforce.com/en/modules/business_process_automation/units/process_builder

[Accessed 05 April 2018].

Salesforce Developers, 2012. *How to integrate Salesforce with external systems.* [Online]
Available at: https://developer.salesforce.com/forums/?id=906F00000008rFZIAY [Accessed 06 April 2018].

Salesforce Developers, 20017. *Salesforce Tools and Toolkits- Development Tools.* [Online]
Available at: https://developer.salesforce.com/page/Tools [Accessed 07 May 2019].

Salesforce Developers, 2017. *What is the difference between "Salesforce" and "Salesforce platform" licences?.* [Online]
Available at: https://salesforce.stackexchange.com/questions/8962/what-is-the-difference-between-the-salesforce-and-salesforce-platform-licens

[Accessed 23 April 2018].

Salesforce.com, Inc., 2017. *Salesforce Knowledge.* [Online]
Available at: https://help.salesforce.com/articleView?id=knowledge_whatis.htm&type=5

[Accessed 07 May 2019].

Salesforce Lightning platform, 2017. *Salesforce Lightning platform.* [Online]
Available at: https://developer.salesforce.com/lightning [Accessed 08 May 2019].

Salesforce Trailhead, 2017. *Salesforce & Heroku Integration.* [Online]
Available at: https://developer.salesforce.com/page/Extreme_Salesforce_Data:_Distributed_Application_Partitioning_with_Force.com_Canvas_and_Heroku [Accessed 07 May 2019].

Salesforce Trailblazer Community, 2017. *Compare Lightning Experience and Salesforce Classic.* [Online]
Available at: https://help.salesforce.com/articleView?id=lex_aloha_comparison.htm&type=0

[Accessed 08 May 2019].

Salesforce Trailhead, 2017. *Trails: follow guided learning paths.* [Online]
Available at: https://trailhead.salesforce.com/trails#role=role-dev [Accessed 08 May 2019].

Salesforce Trailhead, 2017. *Understand the salesforce architecture.* [Online]
Available at: https://trailhead.salesforce.com/en/modules/starting_force_com/units/starting_understanding_arch [Accessed 08 May 2019].

Salesforce Trailblazer Community, 2017. *Which Automation Tool Do I Use?* [Online]
Available at: https://help.salesforce.com/articleView?id=process_which_tool.htm&type=0&language=en_US
[Accessed 08 May 2019].

Salesforce Trailblazer Community, 2019. *Simultaneous execution of workflow rules and workflow actions.* [Online]
Available at: https://help.salesforce.com/articleView?id=000212758&type=1 [Accessed 7 May 2019].

Salesforce Trailblazer community, 2019. *Use Third-Party Data to Update and Add Records to Salesforce.* [Online]
Available at: https://help.salesforce.com/articleView?id=data_integration_update_and_import_records.htm&type=5 [Accessed 8 May 2019].

Wiggins, A., 2017. *The twelve-Factor App.* [Online]
Available at: https://12factor.net/ [Accessed 7 May 2019].

Yeasmin, T., 2017. *Clouds and platforms.* [Online]
Available at: https://www.slideshare.net/TaniaYeasminPreity/introduction-to-salesforce-ppt-64301532 [Accessed 08 May 2019].

Appendices

Appendix 1. Questionnaire for Company X.

Biit and Salesforce

Hi, My name is Gearóid O'Ceallacháin I'm doing a thesis In Biit. I am sending out this questionnaire to find out if any of the projects executed so far have any thing in common, thus building a model project. Then adding the essential properties to create a unique plan for the client.

1. **Name (may put anonymous)**

2. **What do you do in Biit?**

3. **How long have you been in the company?**

4. **How is your day structured?**

5. **What is the life cycle in customer on sales once a new client as been acquired?**

6. **What is your approach when obtaining a new client?**

7. **From a Development point of view, how does a project start - middle -end?**

8. **Is there any parts/ processes that one can use form the previous project(s) to expedite your progress in the present project?**

9. **As a consultant/developer in a salesforce project, what tools enable you to enhance your performance and what tools do you feel can aid in a better salesforce implementation for client needs? PM tools and development tools**

10. **Do projects seem to be all the same? or is there a variety that keeps you interested?**

11. **Is there a way of using the same tools in a different project? Is there some aspects to the each project that are the same?**

12. **If one could change anything in Salesforce or any part of the salesforce PaaS what would it be? What would you change?**

13. **Any other thoughts?**

Hi, my name is Gearóid O'Ceallacháin I'm doing a thesis In ~~Biit~~ company X. I am sending out this questionnaire to find out if any of the projects executed so far have anything in common, thus building a model project. Then adding the essential properties to create a unique plan for the client.

Company X and Salesforce

Name (may put anonymous)
Tsailing Wong
Olli Huhtala
Miikka Henäsmäki

What do you do in company X?
Senior Salesforce developer
Consultant
Senior Consultant
Business consultant

How long have you been in the company?

One year
3years
1.5 years
1,5 years

How is your day structured?

Check emails, fix bugs, work on features, respond to requests (client and/or ~~Biit~~ internal), some documentation, read articles, and lunch somewhere in between.

Check the daily/weekly project plans and personal calendar and then perform

Customers first, me second.

What is your approach when obtaining a new client?

Not applicable to my position.

I don't understand the scope of the question

Ask what they struggle with.

Focus on customer business: Company ‘X’s best sales approach is to build the solution together with the customer. Also, diminishing the sales phase and start doing things together with the customer as early as possible

From a Development point of view, how does a project start - middle -end?

Discussion with the client, create a demo or proof of concept (PoC) as needed, setup environment, create profiles and accounts, start development, and conduct an ongoing review and test meetings. End with production deployment and ongoing maintenance and bug-fixing.

I don't understand the question.

Starts good but normally ends badly because of bad estimates or no estimates at all.

Are there any parts/ processes that one can use form the previous project(s) to expedite your progress in the present project?

Sure. Apex trigger/batch/scheduler/services framework, documentation templates, CSS templates and examples, diligent logging of tips and tricks from previous projects.

Yes

As a consultant/developer in a Salesforce project, what tools enable you to enhance your performance and what tools do you feel can aid in a better Salesforce implementation for client needs? PM tools and development tools

Eclipse IDE for coding, Salesforce Data Loader for data migration, MavensMate for code download, and Google Office for documentation.

A good tool for backlog and task management is a necessity.

Task management: Jira, Asana, Trello Efforts + Estimates: Google sheet Slack as communication.

For my job (also not a Salesforce developer) the most important tool is a collaboration channel (discussions, communication, documentation). Now it’s all about teamwork as don't have any good fundamental tools.

Do projects seem to be all the same? or is there a variety that keeps you interested?

There's always something different and something new to learn. Someone always has a new way of solving an old issue. It helps to work in teams with different people to see the problem-solving methodology of others. Salesforce is constantly deprecating old tools and inventing new ones, so it is good to keep updated and trained on them.

Well, every project seems to have some common stuff, but mostly the businesses are different, so it always requires new thoughts. Also, customers might need a similar kind of solution, but they need to be implemented at a different level for each customer. For example, one customer can do with a simplistic solution whereas another needs a lot more features around the same solution. It requires both business and SF-technical understanding to create a suitable solution.

New technologies, new features, project management, doing things smart

Projects are unique, but many customers have the same problems.

Is there a way of using the same tools in a different project? Are there some aspects to each project that are the same?

Yep, I use Eclipse, Data Loader, and MavensMate in all projects.

Same tools yes but using the same SF components might not be that easy. However, having elemental tools to start. Perhaps then building up those elemental packages rather than creating from scratch every time.
Use Cases Work Estimates Slack

If one could change anything in Salesforce or any part of the Salesforce PaaS what would it be? What would you change?

Better source control and deployment tools. Also, more flexible/customizable forecasting tool. Better documentation and support for third-party tools that they have bought and integrated, for example, CPQ. Better Apex API documentation. Currently, I have to resort to the Eclipse IDE database tool to identify the object ID prefix and API names, especially for junction objects for the approval process, history and sharing objects, the various content management objects, etc.

Chatter is bad (compared to Slack). Wikimedia type documentation tool would be nice.

Any other thoughts?

getting only four responses at the start and some of the answers not being answered or not understood doesn't give a lot of information but the information that it does give is, that there are core programs being used and there is a possibility of a core model for every project.

Appendix 2. Governance rules

Description	Synchronous Limit	Asynchronous Limit
Total number of SOQL queries issued[1]	100	200
Total number of records retrieved by SOQL queries	50,000	
Total number of records retrieved by `Database.getQueryLocator`	10,000	
Total number of SOSL queries issued	20	
Total number of records retrieved by a single SOSL query	2,000	
Total number of DML statements issued[2]	150	
Total number of records processed as a result of DML statements, `Approval.process`, or `database.emptyRecycleBin`	10,000	
Total stack depth for any Apex invocation that recursively fires triggers due to `insert`, `update`, or `delete` statements[3]	16	
Total number of callouts (HTTP requests or Web services calls) in a transaction	100	
Maximum cumulative timeout for all callouts (HTTP requests or Web services calls) in a transaction	120 seconds	
Maximum number of methods with the `future` annotation allowed per Apex invocation	50	
Maximum number of Apex jobs added to the queue with `System.enqueueJob`	50	
Total number of `sendEmail` methods allowed	10	
Total heap size[4]	6 MB	12 MB
Maximum CPU time on the Salesforce servers[5]	10,000 milliseconds	60,000 milliseconds
Maximum execution time for each Apex transaction	10 minutes	
Maximum number of push notification method calls allowed per Apex transaction	10	
Maximum number of push notifications that can be sent in each push notification method call	2,000	

Figure 13 Apex Governor Limits (Salesforce Developers, 2109)

Description	Limit
The maximum number of asynchronous Apex method executions (batch Apex, future methods, Queueable Apex, and scheduled Apex) per a 24-hour period[1]	250,000 or the number of user licenses in your org multiplied by 200, whichever is greater
Number of synchronous concurrent transactions for long-running transactions that last longer than 5 seconds for each org.[2]	10
Maximum number of Apex classes scheduled concurrently	100. In Developer Edition orgs the limit is 5.
Maximum number of batch Apex jobs in the Apex flex queue that are in `Holding` status	100
Maximum number of batch Apex jobs queued or active concurrently[3]	5
Maximum number of batch Apex job `start` method concurrent executions[4]	1
Maximum number of batch jobs that can be submitted in a running test	5
Maximum number of test classes that can be queued per 24-hour period (production orgs other than Developer Edition)[5]	The greater of 500 or 10 multiplied by the number of test classes in the org
Maximum number of test classes that can be queued per 24-hour period (sandbox and Developer Edition orgs)[5]	The greater of 500 or 20 multiplied by the number of test classes in the org
Maximum number of query cursors open concurrently per user[6]	50
Maximum number of query cursors open concurrently per user for the Batch Apex `start` method	15
Maximum number of query cursors open concurrently per user for the Batch Apex `execute` and `finish` methods	5

Figure 14 Apex Governor limits Cont. (Salesforce Developers, 2109)