Eeva Haataja

Displaying N-depth Parent Hierarchy in sObject Field

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Technology

Thesis

1 May 2017

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Eeva Haataja Displaying N-depth Parent Hierarchy in sObject Field 32 pages + 1 appendix 1 May 2017
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor(s)	Harri Airaksinen, Principal Lecturer

The goal of the project was to find a solution to display identification numbers of parent record hierarchy in Salesforce Custom Object text field. The requirement was that the depth of record hierarchy is not fixed and the number of characters used in field is the maximum character number of the longest available text field.

Project development environment was Salesforce, which is the leading CRM-software. Development was managed in Developer Edition sandbox and data model followed Salesforce model. Configuring object to track n-depth hierarchy is not common, therefore, the aim of this study was to do a research on issues related to n-depth Object hierarchy.

Firstly, each possible event for record was designed to a chain of events. After the first phase, restriction in Salesforce environment was discovered based on what problems the project might face. The discovery phase excluded approaches, but also brought up restrictions that cannot be avoided and thus, need to be noted when developing.

The third phase was to build a Process for each chain of event. Development for Autolaunched Flow followed, since Process needs to launch a Flow to complete record update. A clear and reliable solution was sought when developing a Flow. The last part of the solution development was Apex Trigger for event when the record should be deleted. The purpose of the Trigger was to delete the connection between record and its child records and then update the child records. A previously built Process was utilized when updating child records.

As a result of this study, a solution which meets the core criteria of the project was found. However, some risks were detected, which denotes that the solution might not apply to a larger scale. The solution discovered can be optimized and improved, though. The project was made for a software start-up company, which is focused on developing Salesforce applications.

Keywords	Salesforce, sObject, CRM, Apex Trigger, Process Builder, Autolaunched Flow

Tekijä Otsikko Sivumäärä Aika	Eeva Haataja Rajattoman syvyisen hierarkian esittäminen sObjectin kentällä 32 sivua + 1 liite 1.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Yliopettaja Harri Airaksinen

Insinöörityön tavoitteena oli löytää ratkaisu esittää itseensä viittaavan Salesforce-objektin tekstikentällä datahierarkiassa ylempien tallenteiden identiteettinumerot tekstimuodossa. Vaatimuksena oli, että hierarkian tulee olla rajaamaton ja rivin merkkien määrän suurin sallittu.

Toteutusympäristönä oli Salesforce, joka on johtava asiakkuudenhallintajärjestelmä (customer relationship management, CRM). Kehitysympäristönä toimi Salesforcen Developer Edition, ja datamallinnus on Salesforcen mallin mukainen. Salesforcen ympäristön objektin konfigurointi rajaamattoman syvyiseen hierarkiaan ei ole tavanomaista, joten tämän insinöörityön tarkoituksena oli selvittää, kuinka ratkaista siihen liittyvät käyttöongelmat.

Ensin projektissa suunniteltiin jokaiselle objektiin liittyvälle tapahtumalle, kuten luomiselle, poistamiselle ja päivittämiselle, vaatimusketju. Seuraavana vaiheena kartoitettiin ympäristön rajoittavat ominaisuudet, jotka voivat haitata ratkaisun löytymistä. Kartoituksen tavoitteena oli sulkea pois lähestymistapoja, mutta se toi myös esille huomioon otettavia rajoituksia, joita ei voitu välttää. Muun muassa tietokannan hakuihin ja toimintoihin liittyi paljon Salesforcen asettamia rajoituksia. Kartoituksen jälkeen kehitettiin Salesforcen Process Builder -työkalulla tapahtumaketju jokaiselle objektin tapahtumalle paitsi tallenteen poistamiselle. Seuraavana oli vuorossa automaattisesti käynnistyvän kulun kehitys sille, kuinka tallenne päivitetään ajantasalle. Kulkukehityksessä pyrittiin selkeään ja luotettavaan tulokseen. Viimeisenä vaiheena oli luoda Apex-käynnistin tallenteen poiston tapahtumaketjulle. Apex-käynnistimen tehtävänä oli poistaa yhteys alemman kerroksen tallenteiden ja poistettavan tallenteen väliltä ennen tallenteen poistamista.

Insinöörityö tuloksena syntyi toimiva ratkaisu, joka vastasi projektin päätavoitetta, eli objektin tekstikentälle pystyi luomaan tallenteen ylempien tallenteiden identiteettinumerot. Ratkaisussa huomattiin kuitenkin riskejä, kun tallenteiden määrä oli suuri tai hierakia oli rakennettu syväksi. Löydettyä ratkaisua on mahdollista vielä kehittää paremmaksi riskien vähentämiseksi sekä tapahtumaketjun kulun nopeuttamiseksi. Insinöörityön oli tilannut ohjelmistoalan start-up-yritys, joka on painottunut Salesforce-sovelluksen kehitykseen.

Avainsanat	Salesforce, CRM, asiakkuuksienhallintajärjestelmä, sObject, Apex-käynnistin, Process Builder

Contents

Abbreviations

CRM	Customer Relation Management.
ID	Identification.
PaaS	Platform as a Service
SaaS	Software as a Service
Org	Environment organization in Salesforce.
SOQL	Salesforce Object Query Language.
API	Application Programming Interface
DML	Data Manipulation Language

1 Introduction

The topic of the project was to set up a custom field to display a concatenated string of parent record IDs, where the depth of the record hierarchy is not limited. This project emphasised scalability and reliability of data. The goal was to be able to create an update and delete records while maintaining correct hierarchy display. Before starting the project, no scalable solution was known, since most straightforward approaches had major scalability issues and restrictions. Moreover, there was no guarantee that configuring custom field to show parent IDs in n-depth was even possible.

The project is made for young start up, Flowhaven Oy. Company was founded in early 2016. Flowhaven provides brand productization platform. Since Flowhaven is a Salesforce partner and naturally the product is utilizing Salesforce environment. This project serves as a technical enhancement needed by the company. A field that contains concatenated string of parent IDs can be utilized in multiple areas. In case of project success, it enables more scalability to the usage of record hierarchies. However, the hypothesis is that no scalable solution can be found since Salesforce has restrictions regarding n-depth record hierarchies.

The current solution to display hierarchy IDs, is to use formula field to fixed depth. The purpose of the project is to find out if this field can display n-depth hierarchy by configuring. Related concerns are, whether the populated data is reliable and accurate. Another concern is that the solution is scalable and can be optimized later. If no proper solution cannot be found, the project needs to answer why this is the case and whether there is a possibility it can be done by using another method.

2 CRM Systems

2.1 Defining CRM

CRM comes from Customer Relationship Management. The definition of the term can vary. Adrian Payne defines CRM as a strategic management of customer relations. CRM can include technology needed for management, such as CRM system. In this definition CRM is more than customer management but not a relationship marketing as shown in Figure 1. [1, p. 22-23]

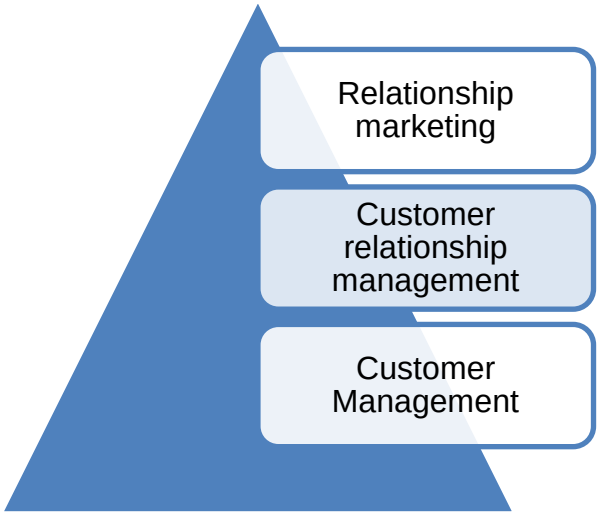

Figure 1. Modified from Handbook of CRM. [1, p. 22]

The definition is made to fit its purpose for the company. CRM can also be defined as a business strategy, which takes into account the life-cycle of customer relationship. It can also be a tool that helps to market to customers in correct time and channel. The third explanation could be just simply marketing that is data-driven. There are plenty of other definitions that have small nuance different. [1, p. 19]

Salesforce defines CRM more as a technical aspect as that is their asset but the definition is also aligning with the same definition as Payne's. For Salesforce, CRM is a strategy to manage customer relationships by using technology. Technology can be an Excel sheet when CRM is in small scale but actual customer relationship management will happen with proper management tools. [2]

2.2 Features in a good CRM system

Adrian Payne states that there are three different types of CRM, which will be combined in a successful CRM system. The first one is operational CRM. It is specified to automate processes in sales, marketing and customer service. The second one is analytical CRM. It uses data to give out information that might been missed, and does calculation about which customer is more profitable, among other things. It can even predict customer needs. The third one is collaborative CRM, which means communication between customers and employees of the company. Together analytical CRM can use data, which has been inserted in operational CRM and operational CRM can use collaborative CRM to automate messaging, or even help in trouble situations.

A good CRM system is not only about the features it carries. It requires daily active users and relevant data to process. If only few employees use the CRM system, the effect is quite small. The goal of CRM system is to use it efficiently and utilize data imported by all users.

CRM Software is always an investment for a company. The value of investment for CRM Software is difficult to calculate. It depends on how much time is saved by using the operational automation and collaborative channels. Also, the value depends on whether the sales have increased when relying on analysed data and whether the quality of customer data is maintained. Also, customer satisfaction quality is one aspect that matters.

Nowadays, mobility is needed. Installing a software to a local computer and reach the data only when accessing work on a computer in an office is no longer efficient. Proper CRM solution enables cloud computing, so that the system can be accessed and used through multiple devices, even on mobile. [2] It is no surprise that the CRM software with most active users have a web browser based entry to the data and mobile applications for major mobile operating systems [3].

2.3 CRM Software

Building an own CRM system can be costly. Therefore, there is a market for CRM software because not only Salesforce is providing tool for customer relationship management. They have plenty of competitors. Currently Salesforce is growing and taking a big

leading position in CRM solutions. Capterra has made a ranking list of CRM software. Ranking is based on the number of customers, number of users and social presence. In social presence Facebook likes, LinkedIn followers and Twitter follower were counted on top of reviews in Capterra site. Results are from July 2016. [3]

Table 1. Modified form Capterra chart, version July 2016. [3]

Software	**Customers**	**Users**	**Rank in Capterra**
Salesforce	150 000	7 200 000	1
Zoho	80 000	20 000 000	2
Microsoft Dynamics	40 000	4 250 000	4
SAP	10 000	8 000 000	6

Capterra has ranked Salesforce as number one CRM software. Salesforce had 150 000 customers and 7 200 000 active monthly users in 2016. Zoho, which is in second place in Capterra charts, is also a web-based software like Salesforce. Zoho had less customers, only 80 000, but 20 000 000 active users, which is a lot more than Salesforce. Microsoft Dynamics is fourth. It had 110 000 less customers than Salesforce and only 4 250 000 active users. SAP, which is a well known ERP software, has a CRM side as well. SAP is only ranked as sixth in Capterra charts. SAP had only 10 000 customers but 800 000 more active users than Salesforce.

Zoho and Salesforce both have mobile applications. Where Zoho has multiple types of applications, Salesforce has focused on Salesforce1 application. The volume of the usage can be seen even in application level. Salesforce1 has more than a million active monthly users [4]. Alone in Google Play Salesforce1 application is in 1 000 000 to 5 000 000 downloads group. Zoho CRM application has significantly less, since it is in 100 000 to 500 000 downloads group. [5, 6]

2.4 Why Salesforce

The latest released Fiscal Year results look quite bad for Salesforce. Income from operations is barely 2 % of revenues in 2016 fiscal year which ended January 31st. For fiscal year 2017 which ended January 31st, the percentage is even lower, less than one percentage. However, Salesforce has invested 251 827 000 000 dollars on development

from previous year. That means Salesforce can afford huge investments to improve their service. Also, they have innovations coming up.

Table 2. Data collected from Salesforce Fiscal year report. Numbers are in thousands and currency is US dollar. [7]

	Fiscal Year 2017	**Fiscal Year 2016**
Total Revenue	8,391,984	6,667,216
Marketing cost	3,918,027	3,239,824
Development investment	1,208,127	946,300
Operating cost	967,563	748,238

Einstein, Artificial Intelligence, is one of Salesforce's newest innovations. Tony Prophet, Chief of Equality in Salesforce, said in Amsterdam Salesforce World Tour 2017 that Einstein is making Salesforce the smartest CRM [4]. Einstein is not a smart chatbot, but it is an AI that predicts and calculates the data that is given. It is improving the operational CRM side and analytical side of CRM. Building own artificial intelligence is costly and can explain the increasing investments in development.

Einstein is mostly working in background, Lightning on the other hand is more visible. Lightning is another development project in Salesforce. It renews the way Salesforce pages are created, structured and loaded. Salesforce classic is no doubt an old fashioned user interface, so with Lightning the visual appearance is brought to today's standards. Lightning pages are responsive and fitting all size of displays. Lightning is not directly improving any of the three types of CRM but it is more like a brand uplifting. Lightning is making page loading faster, so overall performance is improved. These two projects, Einstein and Lightning, are now both ready for all customers to use, so the next years will show if the investment was profitable.

On administrator and development side, Salesforce has certification exams and active forums. Expertise is widely shared online and through Salesforce events where community is kept alive. Moreover, Salesforce sandboxes offer a customizable platform where to develop additional features to Salesforce org or to AppExchange for others to use. Downside of Salesforce are the limitations which apply to development and usage. This causes complicated configurations. In turn, limitations in an environment are the key why Salesforce apps are running securely.

Salesforce CRM Software does meet the criteria of features in a good CRM system. That does not directly mean that Salesforce is a good CRM software. It can grow expensive since pricing is done by user per month and price gap between editions is enormous. Moreover, some features need add-ons, which can be expensive.

3 Salesforce environment

3.1 Salesforce success platform

Figure 2 shows how Salesforce has built its platform. The structure is layered into five different levels: Multitenant infrastructure, Data services, Artificial Intelligence Platform services, Development Platform and Salesforce Applications. These five levels are called the five success platforms in Salesforce.

One of key features is multitenant architecture. With multitenancy, Salesforce can flexibly split its resources to tenants. To avoid any tenant to dominate the resources, there are governor limitations, which are covered in chapter 4.3. Multitenancy is cost-efficient because a company who is using the CRM software does not need to maintain servers and operating updates of the service. [8]

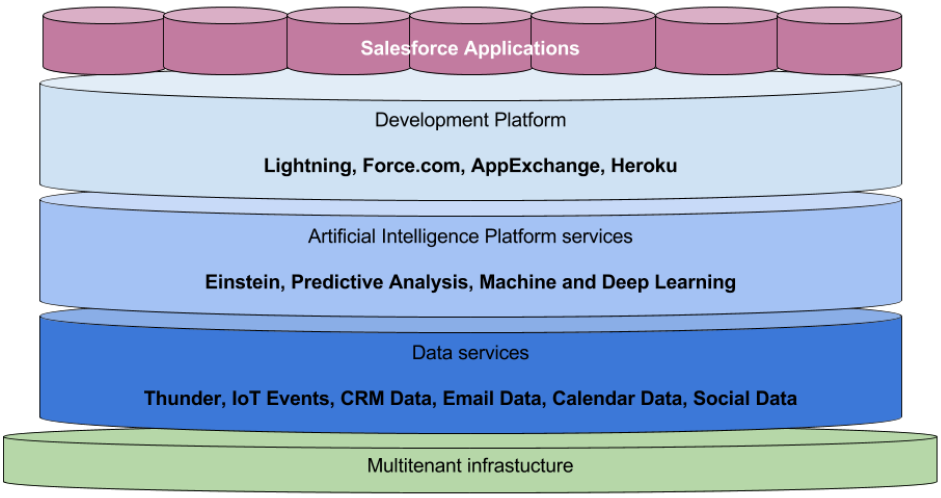

Figure 2. Salesforce success platform structure modified from Prophet's presentation. [4]

Thunder is an event processing engine. With Thunder, IoT Events can be scaled to wide extent in real time. Thunder contains CRM data, users email, calendar and social data. Thunder connects events which are happening in IoT Cloud in real time. When adding events to data points in Thunder, user can get personalized actions. Thunder is the base of the data services in Salesforce customer success platform. [9]

On top of Thunder there are artificial intelligence platform services. It consists of Einstein, predictive analysis and machine and deep learning. This layer is studying the data which is in Data service. Without this layer, end users would get a lot of information that is difficult to handle. Artificial intelligence platform analyses what is relevant and important information for the user and can even enrich the data to the end user. This means the user does not need to calculate the data points and meanings together since AI has done it already for the user.

Predictive analysis is an important matter. It learns patterns, habits and predicts behaviour so an enterprise can react or prepare for future events. Opportunities can be detected easily and marketing can be optimized when relying to predictive analysis. It can be time saving and increasing chances to have positive impact on customer relationships.

3.2 Salesforce Lightning

The third layer in customer success platform is the development layer, which this project is also focusing on. This layer contains Lighting, Force.com, AppExchange and Heroku. In this thesis Heroku will not be covered. As mentioned earlier in a previous chapter, Lightning has recently been added as a feature in Salesforce. It is going to replace Salesforce Classic. Salesforce Classic is the name of the platform which has been used as a standard in Salesforce. It looks like it is only about user interface but compared to Lightning, it deals with also performance. Lightning brings new items to Salesforce: Lightning experience, Lightning components, Lightning Connect and Lightning Design System. [10]

Lightning Experience is the user interface designed to look better and make work more efficient. A Lightning component can replace the usage of a Visualforce component,

which can be used on multiple pages. The difference is that a Lightning page loads components faster and displays component dynamically when loaded.

Lightning Connect is used when external application is needed. This could be the case when there is data in an enterprise resource planning (ERP) system. Salesforce Lightning Connect supports table integration from SAP, Microsoft Dynamics and more.

Lightning Design System is a pallet of CSS framework, icons and fonts. Also, Design tokens are included. CSS framework helps the developer to define the main UI components like headers and elements. Visual grid layouts and adjustments belongs under this framework. Icons ease the pain to visualize an action or button shorter manner than words**.** Salesforce provides a set of icons in PNG or SVG versions. The style of icons is naturally consistent with Lightning experience. [11]

3.3 Force.com

Force.com, which is mentioned in Development platform level, is a Platform as a Service. Platform as a Service is shortened to PaaS. As figure 3 shows, it is a service where application and data can be developed and customized. More specifically, PaaS provides a platform where example runtimes, middleware security, database and networking is already provided. In Force.com this means, when developing application, most of the application is already done. [8, 12]

Force.com even offers standard data structures and security settings so developing would be focused on the application itself. Building the application takes less time and connecting to database is securely done by Salesforce since Salesforce offers a developing platform.

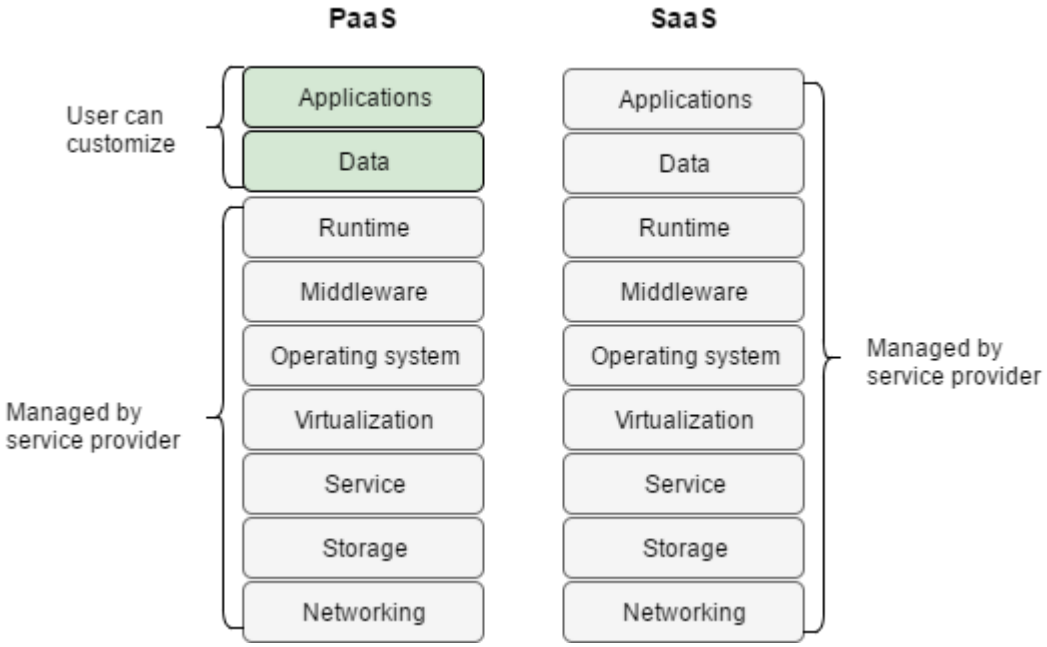

Figure 3. Difference between PaaS and SaaS [13].

Force.com has been also called as Software as a Service, which means that even applications are managed by Salesforce. This perspective is understandable, since Force.com is responsible for distributing software services on mobile and different devices. The term SaaS is mostly leaning to Salesforce.com side, which is the CRM Software name for Salesforce. Moreover, SaaS in Salesforce is Salesforce Customer Success Platform, which is above Development Platform. [14]

3.4 Salesforce Customer Success Platform

On top of the Salesforce success platform are Salesforce Applications, which are the Salesforce Customer Success Platform. Salesforce has developed tools for managing business in various areas. Each product has its own license and pricing table. The price is user and monthly based and varies between editions. Salesforce has the following products:

- Sales
- Service
- Marketing
- Community
- Analytics
- Apps

Products can be used in Salesforce Organization of the customer. Organization, usually referred as Org, in Salesforce means the tenant environment itself. Salesforce has different editions for orgs. For example, Developer Edition Orgs are free and anyone can create one. Depending on the edition there are limitations how much the user can do inside the org or how many active users can be created. [15]

Organization manages the application, communities and users who are engaged in environment. To customize org, admin can create own solutions by building own app. Another option is to install application from AppExchange. AppExchange is like any application store. Some applications are free and some have a price tag. Application packages are developed by different resources. AppExchange does not only offer applications, but also components and consulting services. Together these are called offerings.

Application in AppExchange are using Salesforce platform and they are standalone solutions. All applications in AppExchange are built by Salesforce Partners and gone have through security review. In other words, all applications are accepted by Salesforce. To pass this review, the code needs to be secure enough and implemented by following best practices. All Apex classes inside the package needs to have at least 75 % test coverage. Test coverage is met when Test Apex Classes are testing all methods and operations inside Apex classes by using test data and scenarios.

Applications can be unmanaged or managed packages. An unmanaged package is not hiding the code, so the org which installs it, can modify it. Application provider cannot

update an unmanaged package unlike managed packages. Managed packages also protect the code by hiding it. Since it cannot be edited, upgrades to existing package are possible. [16]

An organization can install as many offerings as it wants, but inside the organization license limitations. There is a limitation how many custom objects an organization can have and custom objects of an installed package are counted. Apps and tabs of a managed package are not counted against license limitations. [17]

4 Development in Salesforce

If AppExchange does not have a solution for Org, it can develop their own solution. Anyone can develop own applications in Salesforce. Salesforce has its own object-oriented programming language called Apex. Apex is similar to Java programming language. It is directly compatible to Force.com components, such as database. No database passwords or connectors are needed since Force.com copes with the security.

4.1 Custom objects

Objects in Force.com database are called sObject. Objects are either standard or custom objects. sObjects are Salesforce objects, which follow Salesforce's own logical data model. Salesforce offers plenty of standard objects and fields, which are commonly used for example, in Contract and Product objects. This project is using a custom object, since content is not specified and creating independent custom object allows different kind of structures. [18]

Custom object does not mean it is totally customizable, it also has mandatory standard fields such as ID, Created by and most importantly Name field. Name field can be an automated number or text field of 80 characters. Standard fields are common fields used in all types of objects and therefore, they are not restricting the customization. [19]

Both standard and custom objects can have a selection of custom fields to be configured. There are multiple types of custom fields to use. Relevant custom fields considering this project are:

- Checkbox
- Formula
- Lookup Relationship
- Master-Detail Relationship
- Roll-up Summary
- Text
- Text Area
- Text Area (Long)
- Text Area (Rich)

Checkbox is typical Boolean Checkbox, in a way that it can have only two types of values, true of false. Checkbox can be set to null value in Apex Class but that equals to false value. In this thesis project, checkbox is used to check if the record has been updated or needs an update.

There are multiple types of Text fields and the major difference is the maximum length allowed. Text Area (Rich) allows even to have text styling code. The shortest text field type has maximum limit of 255 characters. This field can be used in SOQL queries to filter results like in Code Example 1. Filtering of text field can be done by using comparison type LIKE. LIKE supports percentage character and underscore wildcards. Percentage wildcard indicates that zero or more characters can match. Underscore wildcard on other hand, can match only one character. In code example 1, query return only Sample Object records which Sample Text field starts with Sampl. Comparison type is case-insensitive.

```
SELECT
          Id,
          SampleText__c
FROM
          SampleObject__c
WHERE
          SampleText__c  LIKE 'Sampl%'
```

Code example 1. SOQL query of Sample Object records.

Lookup relationship and master-detail relationship are similar, but also different. If an object has a master-detail relationship, it is a required field to create a record. Security rules are extended from master object to detail object. Lookup is a in a way a neutral link to another object and a not required field unless otherwise stated. [20]

As Figure 4 shows, if an object is a master side of the master-detail relationship, it can use Roll-Up Summary Field. Roll-Up can summarize set of detail records into one field. For example, Roll-Up can count all related detail records into integer value to tell how many detail records there are. Other options are to calculate maximum, minimum value of the detail records or sum values up. Lookup relationships are not valid for Roll-Up Summary Field and detail records need to be directly related to the master record. [21]

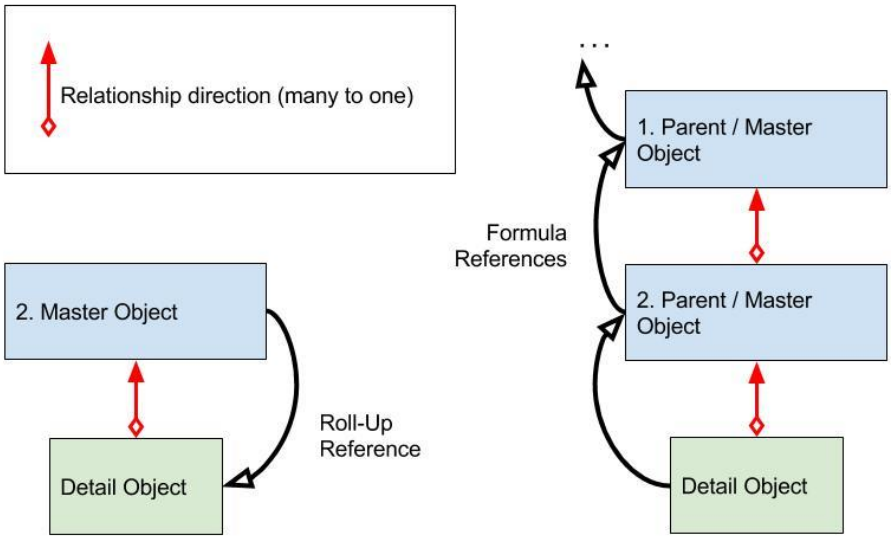

Figure 4. Difference between roll-up summary and formula field.

In most cases a formula field can be implemented when Roll-Up cannot be used. Formula can refer to another direction, to master relationship form detail or to parent lookup object as in figure 4. Figure 4 illustrates how Roll-Up refers to detail records but a formula is going do another direction, towards parent record. Many detail records can refer to one master record. Also, Lookups are one to many relationships. [22]

Formula can generate different types of values, for example checkbox, text or number. Formula has its own syntax to use different type of values. Formulas can be used to display manipulated values of other fields or calculated results of related fields. Formula field can refer even to fields inside a related lookup, or even a lookup of a lookup.

```
CASE( Postal_type__c,
          'Fast Delivery',  Sending_Product__r.Price__c + 80,
          'Courier', Sending_Product__r.Price__c + 60,
          Sending_Product__r.Price__c )
```

Code example 2. Formula to calculate total price for different cases.

Formula syntax has operations for example for math, text and logical operations. Code example 2 is presenting CASE formula which is a logical operation. It calculates the total price of the order, including product price and postal fee, and returns number value. The two first cases are for Postal Type text value. The last row is for cases which are not matching predefined cases, in this example Fast Delivery or Courier.

A natural way to display a field which is populated in parent corresponding field, is a formula field type where result value would be text. However, since each record field is built by a parent record it is not that straightforward because of restrictions. A formula field cannot refer to itself, not even indirectly. Therefore, it is not possible to manipulate field records using a related record's corresponding field.

A formula field can contain a maximum of 3 900 characters. It contains spaces, returns and comments. It is long and can reach many levels in n-depth hierarchy. However, as long text is not supported, the longest text formula can create, is 255 characters long. 255 characters is enough up to 13 record IDs, which are 18 characters long and have "greater than" – sign after each ID number.

4.2 Salesforce Object Query Language

Apex uses SOQL and SOSL instead of SQL. These languages are used to retrieve data from Force.com database. SOQL stands for Salesforce Object Query Language and it is used when the object is known. SOSL, Salesforce Object Search Language is for cases when a specific object is not defined in query. [23]

```
SELECT
          Id,
          Name,
          CreatedBy.Name
          Status__c,
          Parent__r.Name,
          Parent__r.RelationSample__r.RiskStatus__c,
          (SELECT
                    Id,
                    Name,
                    description__c
          FROM childObjects)
FROM
          SampleObject__c
WHERE
          status__c!='Terminated'
          OR
          EndDate__c > TODAY()
ORDER BY
          Name ASC
LIMIT
          100
```

Code Example 3. Example of a SOQL query.

SOQL must have at least a SELECT -statement and a FROM -statement. As the code example shows, query can contain many more statements. In SELECT statement all API Names of the fields, that are needed to fetch from database, are presented. Fields from lookups or even their lookups fields can be referred. Lookup or master-detail relationship child records can be fetched only by using nested query. Nested query automatically

filters the results to be relevant to the parent record. WHERE -statement can be used if results need filtering. The code example filters according to status or end date.

The result amount can be limited if needed and order can be ascending or descending according to any field mentioned in SELECT -statement. In code example 3, the query will give List of Sample Custom Object with maximum of 100 records, excluding child records in inner query. Results are in a list in ascending order by Name of the record.

Custom objects have __c – suffix in their API Names. Parent fields are referred by first mentioning the API name of the field where lookup to parent takes place. Instead of using custom object normal suffix after API name, __r suffix is used. Letter R means relationship. However, the target field is using C-letter after API Name. Standard fields do not use any underscore suffixes, like in the example code CreatedBy.Name is written.

Example code 3 has a nested query inside parentheses. Nested queries have some limitations. Nested queries can be made only to children records. Therefore, accessing fields in children records is difficult and might be done only by making separate SOQL calls. If children records need to be fetched, it can be better to start the query from children, since getting parent data is easier than children data.

SOQL calls can be made inside for-loop section, however, this is not a good practice and will be discussed in code security review. Moreover, having SOQL in for-loop is a risk, since action will fail in case governor limitations for SOQL are hit.

4.3 Governor limitations

As Salesforce is a multitenancy platform, transactions in the Org have limitations. This makes sure that no Org will monopolize the resources. One transaction is one chain of operations in one unit. For example, Apex code that creates a record and fires a trigger which updates another record is one transaction. Triggers, Processes, Flows and Apex Classes are all included in Apex limitations. [24]

Governor core limitations are limiting one transaction. Therefore, it is a good practice to split enormous operation units into smaller ones or optimize the code. For example, following items are limited:

- SOQL queries called
- Number of Records retrieved
- Number of DML statements

One transaction can call SOQL queries synchronously 100 times in total. In case where n-depth hierarchy is retrieved, this limit is soon reached if bad quality coding is practiced. Queries inside a for-loop are not recommended and it will be noted in security scans. Queries can retrieve only 50 000 records in total per one transaction.

Only 150 DML statements are allowed in one transaction. Such as, delete, insert and update are counted against DML limitations. 150 is a low number when big number of records are being manipulated. Therefore, statements need to be bulkified to avoid hitting the limits. One statement can update multiple records if they are in one list of object.

4.4 Apex Trigger

Apex code can be tied to sObjects by using Apex Triggers. Apex Triggers are fired before or after a change is detected in records of the related sObject. Triggers are needed when data in the record affects another record or needs validation. Example code 4 is made for Sample custom object specific which is triggered only after Sample record has been updated. Example trigger manipulates child records if a new status of the triggered record is terminated. [25]

```
Trigger sampleTrigger on Sample__c (after update){
          Set<Id> recordIds = new Set<Id>
          for(Sample__c record:Trigger.new){
                    if(record.status__c == 'Terminated'){
                              recordIds.add(record.Id);
                    }
          }

          List<ChildObj__c> childrenRecords = [
                    SELECT
                              Id,
                              childStatus__c
                    FROM
                              ChildObj__c
                    WHERE
                              parentId IN :(recordIds)];

          for(ChildObj__c child:childrenRecords){
                    child.childStatus__c = 'Cancelled';
          }
          update childrenRecords;
}
```

Code Example 4. Apex Trigger example for Sample custom object.

Triggers can be fired by mass action, for example mass update. Therefore, as you can see in example code 4 in row 3, the records which fired the Trigger are collected before doing the query in line 10. In the example code 4, only records which status has been changed to Terminated are collected.

Trigger can use previous values of the record or new values. Old values can be referred to with an old-variable and new values with a new-variable. Old-variable can be used only to update and delete triggers. New-variable can be used to insert, update and un-delete triggers. [26]

Some assignments can be implemented with Formula and Trigger exclusively. For example, this status change can be done in formula, but in that case, the field is Read Only so the user cannot update the field directly. Trigger can only manipulate fields which are not Read only fields. For example, user can later change the childStatus__c to something else, which can be either a security risk or a wanted feature. Depending on the use case formula or trigger is selected to do the job.

Trigger has some more advantages compared to formula field. A Trigger can use child values, unlike formula. Since Trigger uses Apex language it is in a way more flexible than Formula which has a restricted amount of methods. Moreover, Apex Trigger can refer to a static method in any public Apex Class in the org.

Using Trigger to update the record which triggered the Trigger can be complicated. After an update, it is not possible since it will cause an infinite loop. Moreover, Trigger cannot even be structured in this way, since an error will prevent saving. The only solution is to add the value into the new record details before update.

In case the trigger is supposed to update n-depth child values which are referring to parent value, updating comes more difficult. Since the trigger is triggered before updating, there is no guarantee that the children will get the right value from parent. Before update, SOQL queries are retrieving data that is currently valid, which in this case is old data.

4.5 Process Builder

Process Builder is a workflow tool which has a business like approach. The process builder is visual, and therefore, no coding is needed. The purpose of the tool is to automate business workflow that is usually done manually, such as sending emails or invoke a process. With process builder it is also possible to create or update a record. [27]

An automated process can be launched only through record changes or another process. This means processes are tied to record events. A record which has started the process goes first through the criteria steps. The other parts of the criteria are important since the process goes from top to bottom and only the first match is noted. In case a record meets a criterion, in the process it will execute the actions which are made for specifically for that criteria. If no criteria are met, no action is executed.

Processes can be activated and inactivated. It is important not to have overlapping processes active the same time since the order of the processes cannot be guaranteed. Activation can be done even at version level. A new version is created each time a process is edited.

Process builder is in a way a linear approach, meaning that it does not bend to complicated workflows. Therefore, a process can call Apex classes or launch an autolaunched flow if more steps are needed to accomplish the result.

When designing the processes, and combining it to workflows it is important not to make the workflow executed in the same cases as the processes. It cannot be predicted which one launces first. The same goes for triggers. Therefore, it is good practice not to have overlapping flows and processes or triggers in the org. [28]

4.6 Visual Workflow

Visual workflows are more complex to implement than process builders. However, it offers more flexibility. There are three types of flows: Flow, Autolaunched Flow and User Provisioning Flow. Normal Flows always requires user interaction. Autolaunched Flow is the opposite, since user interaction is not allowed. User Provisioning Flow is for third-party services, which are not covered in this project. [29]

User interaction is required when Flow uses screens, steps or choices. When creating autolaunched flow, these elements cannot be used. Best practises recommend to use steps when designing the flow. However, this might cause issues when converting the flow to autolaunched flow. Process builder might not recognize the autolaunched flow since there is trace a that it has been a normal flow.

Flows have also versioning, but it is not mandatory in similar way as processes. A developer can save the edit as a same version or as a new version. Naturally only one version can be active in one Flow. Multiple flows can be active at the same time.

When Autolaunched flow is launched from Process, there are fields where variables needed for the flow are inserted. Values can be dynamically assigned from the related record. Autolaunched flow can be called from Apex Class by using Flow URL. Variable needed for the flow to execute are added to the link. Code example 5 describes the structure of Flow URL. The first flow is declared and after the second slash API Name of the called flow is expressed. After question mark the variables are mentioned. Value is assigned after variable name and “is equal” -sign. Other variables can be sent by adding “and”-sign between the variable assignments.

```
/flow/SampleFlow?recordId=a025800000CPQg5&newStatus=Terminated
```

Code Example 5. Example of a flow link. [30]

5 Project and its results

5.1 Project problem definition

The goal of this project was to create an automated process that updates specific sObject fields individually. A custom field contains parent objects’ ID numbers in order, separated by “greater than” -sign and the last ID would be its own ID. Figure 5 is the sample of a record, which is in fourth level in the hierarchy tree.

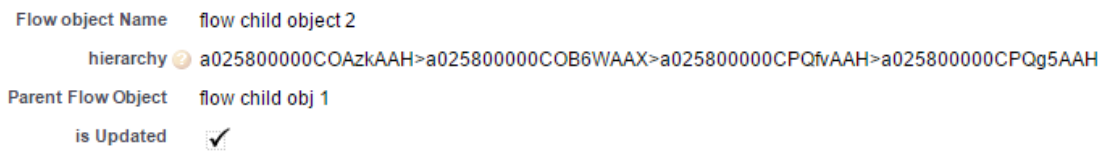

Figure 5. Example of a record.

Flow Object has three custom fields required for this project. The most important one is a lookup field which is referring to itself. It must be a lookup relationship, since a master-detail relationship cannot be made for an object itself. With a lookup relationship, records

can be connected to each other and hierarchy is possible. Another custom field is Long Area Text field which has maximum amount of characters available. We cannot use standard Text field since it is limited to the maximum of 255 characters. 255 characters can contain the maximum of 13 level hierarchy if IDs are displayed in 18 characters. If IDs are displayed in 15 characters, hierarchy level can be up to 16.

To keep track of updated records, there is a checkbox field named is Updated. When it is true, the record should be up to date. If it is false, the record needs to be updated. These fields help identifying records which are pending for update.

In case of insert, a record can have a parent but it is not necessary. When there is no parent, a hierarchy field contains only its own ID number. In a situation when a record has a parent object, it will inherit its parents' hierarchy field and add "greater than" -sign as well as its own ID number to last. The same method can be implemented for displaying record names instead of ID numbers. However, to maintain accuracy, this project is using ID numbers to track hierarchy. Record names might change and be duplicates, but ID numbers are static and unique.

In case a parent record has been changed, the hierarchy field and its child hierarchy fields needs to be updated to new hierarchy. Update needs to run until there are no more children found in the hierarchy. This part is the most difficult one and can easily hit governor limits.

In case a record deletion there are at least two ways to update child records. One is to clear a child-parent field and another one is to replace the field to the parent of the deleted record. This project is using the first mentioned logic.

This project can be implemented for other purposes as well. Similar methods can be used to count the depth of the current record in hierarchy. Instead of adding identification numbers, it would add number values based on what is in parents' respective field.

With this ID hierarchy field, Apex code can split the string to set of ID values and query records that contain the following value. ID hierarchy fields cannot be filtered if they are a Long Text Area, children cannot be fetched by searching the records with the same beginning in hierarchy field. However, this project is focusing on getting parent values over child values.

5.2 Project solution approach

During this project one approach came out reasonably easy to configure. It was also n-depth scalable but it might hit the SOQL limitations quite fast. It is a mixture of a Process builder, automated Flow and Apex trigger. Process builder, shown in appendix 1, is going through the possible scenarios when hierarchy field needs editing. It then calls Flow that generates the new hierarchy text. In the end, it also alerts related child records that they need an update by updating “is Updated” -field to false. These children go through the Process builder process as well. This process will loop until all children related to are left updated. Trigger is only needed to clear out the relation between to-be-deleted record and its children which triggers Process builder.

When a record of Flow Object is being created or updated, it will start a process which can be called a Hierarchy Process. Hierarchy Process is set up always to start in cases when Flow Object type of record is being created or any its fields have been changed. This means DML calls insert, upsert and update will start the process for the Flow Object record.

Process builder has four scenarios listed. It goes from top to bottom as you can see the flow in attachment 1. The first one has a case which checks if the record “is Updated” -field equals false and the record has a parent. Then it will call a linked automated flow. Autolaunced flow was named as a Hierarchy Flow. Hierarchy Process will give two values to Hierarchy Flow: ID of the record which started the Hierarchy flow and the hierarchy of its parent.

If the first condition is not matching the record, there is a second scenario. This scenario is checking if the record field “is Updated” equals false, but does not have a parent record. In this second case, Hierarchy Process will send only the ID of the record which started the process to the autolaunched flow. The flow is the same Hierarchy Flow, but because the hierarchy string is not sent, Hierarchy Flow will act differently. These first two cases cannot be combined, even though they are launching the same flow. The reason is that if Hierarchy Process tries to assign null value to Hierarchy Flow, it will cause an error. An error would be thrown every time the parent field is null and no record can be created.

If the first two criteria are not met, the record has been manually edited and “is Update” -field is not set changed true. In these cases, Hierarchy Flow checks if parent exist has it changed. In Apex Trigger this would be comparing New value and Old value. If a parent is changed and the record still has a parent, Hierarchy Process will launch Hierarchy Flow. Process will send the ID of the record and hierarchy string of its parent.

Fourth and last criteria is for cases when a parent record has been deleted or record does not have a parent anymore. Hierarchy Flow checks if the parent has changed and the parent field must now be null. If conditions are met, Hierarchy process will send to Hierarchy Flow only the ID of the record which started the process.

In a case when none of the criteria is met, the process is stopped. Notice that the record will only match a maximum of one criteria in the process. The order of the cases can affect the results, but in this project, it had no significant influence. The only influence is the performance difference, when the most common case is placed first.

Hierarchy Flow requires at least the record ID to go through the whole flow. Hierarchy string is conditional for the flow to start, but naturally it is required if the parent record exists. As you can see from figure 6, the first item to check is if parent exists. “Has parent” -result is met if hierarchy string is not null. If the string is null, no parent exists. If the record has a parent, the flow is updating the hierarchy string by adding a first “greater than” -sign and the record ID to the end of hierarchy string. If the record has no parent it will start the hierarchy by adding only its own ID to hierarchy string.

A created or an updated hierarchy string is added to record when record is updated in the next step. During the record update, also the “is Updated” -field is updated to be true. Updating the record is one DML statement and it is counted against governor limitations.

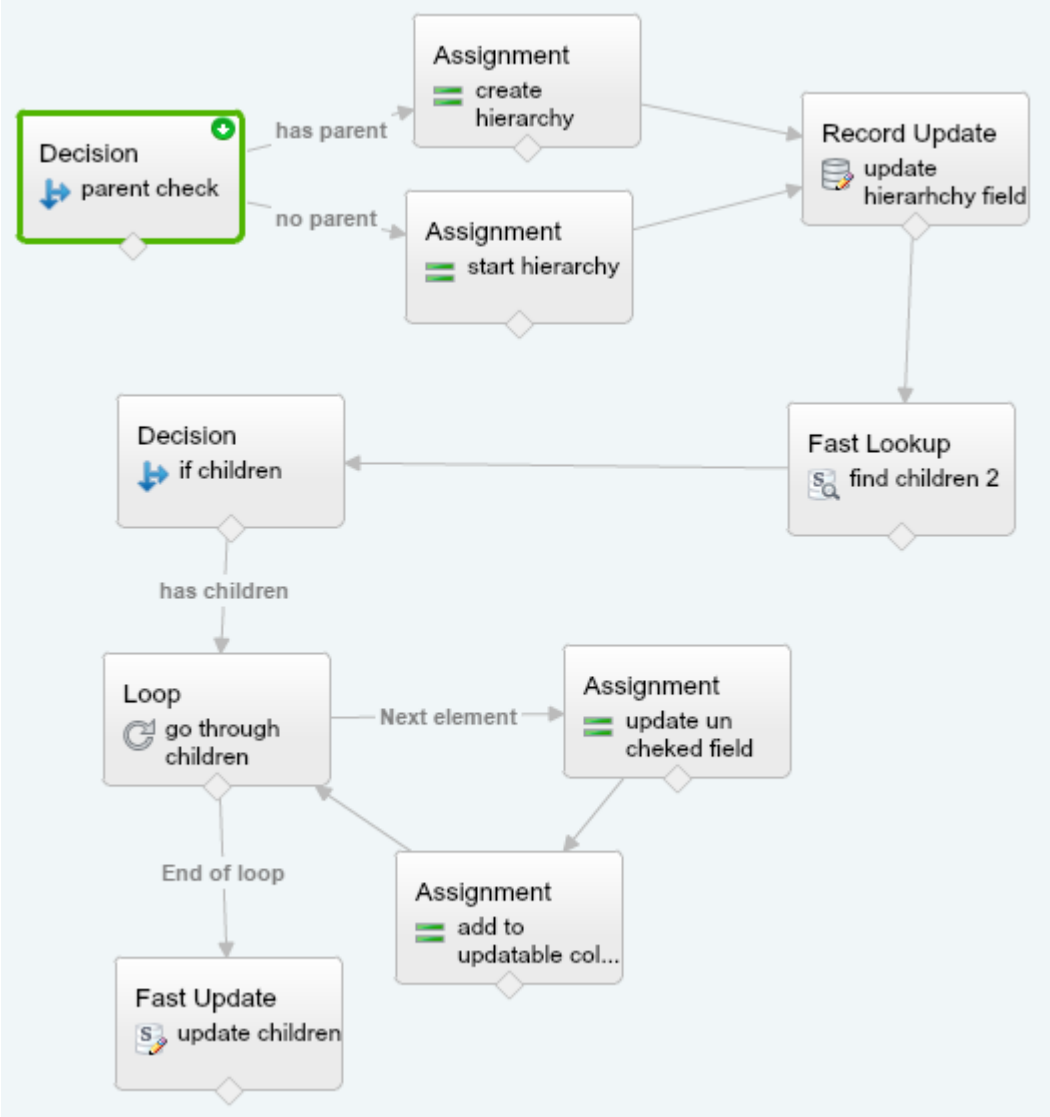

Figure 6. View of the hierarchy flow.

After record update, Hierarchy flow moves to fast lookup the group of records. Fast lookup will fetch children of the record which started the current flow. Fast Lookup is one SOQL call and is counted as one against governor limitations. Fast lookup collects results to sObject collection which is Flow Object type.

Decision will follow fast lookup. Decision checks if children exist. If sObject collection is empty, flow will end, since there is no assigned next step for the scenario. If sObject collection is not empty, it means the record has child records or a child record. This scenario will lead to looping step.

In the next step, the flow will loop through the child records and changes the value of "is Updated" -field to false. To avoid conflicts it is important to collect the children into another sObject collection of Flow object type. When testing the Hierarchy Flow, there was an issue when changing value in the looping record. Therefore, after updating the value to record level loop is adding the record to another sObject collection.

When a loop has iterated through all children and has added them one by one to other sObject collections it has one more step. Last step is to Fast Update the records in sObject collection. Children could be updated inside the loop by using Record Update, but that is consuming DML statements. With Fast Update all records can be updated against governor limitation by using only one DML statement. Bulkifying the DML statement is the key factor to make the flow possible.

After the Hierarchy Flow ends it might start another Hierarchy Process. This happens in the cases children are found and is Updated field is updated to false. Child records are meeting the first criteria in Hierarchy Process which launces the Hierarchy Flow again. This is looping until no children are left to update.

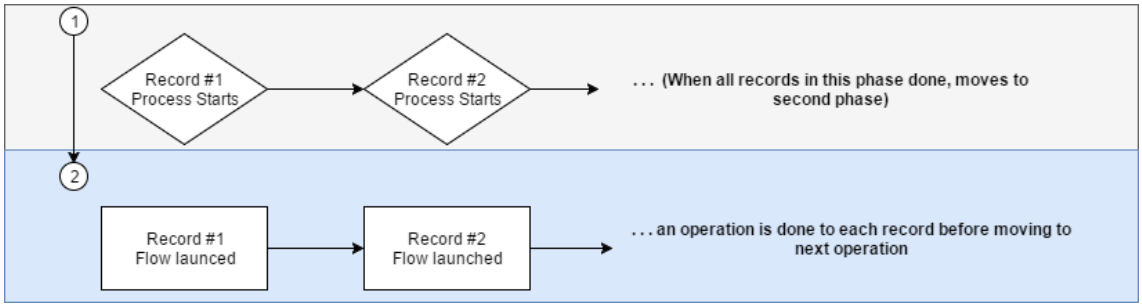

Figure 7. The order of operations in Process to Flow.

Figure 7 illustrates how all records which have started the process will do the steps in process before launching a flow. In the autolaunced flow an operation is done to each record before moving to next operation. Flows have been bulkified for a long time but only since Salesforce Winter '16 update processes have also been bulkified [31]. Child

processes and flows are running alongside after the update. This means the hierarchy tree is being updated level by level in sync.

In case of record deletion, Hierarchy Process will not be called first. That requires an Apex Trigger to start updating hierarchies. When a record is going to be deleted, it is important to handle child records first to ensure hierarchy will not break because of deletion. A Flow Object has an Apex Trigger updateChildrenBeforeDeletion tied and active. Trigger is only triggered before record is deleted as can be seen in Code example 6.

```
trigger updateChildrenBeforeDeletion on Flow_object__c (before
delete) {
          //catch all deleted to set
          Set<Id> delIds = new Set<Id>();
          for (Flow_object__c delObj : Trigger.Old){
                    delIds.add(delObj.Id);
          }
          //query children records
          List<Flow_object__c> children =
                    [SELECT id
                    FROM Flow_object__c
                    WHERE Parent_Flow_Object__c IN :delIds];
          //change the updated value
          for(Flow_object__c child : children){
                    child.Parent_Flow_Object__c = null;
                    child.is_Updated__c = false;
          }
          //bulk update children records
          update children;
}
```

Code example 6. Apex Trigger for Flow Object for before record delete event.

Trigger which is specified as before delete, collects all to-be-deleted records, meaning that single and bulkified deletions are supported. After collecting record IDs to ID Set, it calls the database. SOQL is used since Object is known and filtering ID values are known. Query is getting children of soon-to-be-deleted records. Only first level of children are found by this query.

All queried records are collected to a list of Flow Object. Inside a for-loop, records are being iterated through it. Parent Flow field is set to null and “is Updated” -field is changed to false for each record in the list. After for-loop, records are mass updated to database. Mass update is consuming only one DML statement. Once children records are updated bulkified Hierarchy Flow starts for each of the records. Process and flow will loop as in any other case. After all children in all levels are updated, the record which triggered the Apex Trigger can be deleted. If errors occur in any part of the process or flow, parent record will not be deleted as all database changes are rolled back. Accuracy of the database can be maintained only if Before Delete Event is used over After Delete.

5.3 Project results

Populating hierarchy in various hierarchy depths was successful. Accuracy was maintained. However, there are concerns about the scalability of the solution. It might meet the governor limitations if there are many records. Unlimited professional environment, however, can use the solution which was discovered in the project. Governor limitation do not apply in unlimited professional edition and therefore, looping queries is not a problem.

Looping process and flow are not the best practice, especially when there are queries and ML statements inside. However, one approach tried to minimize the looping, at least avoiding going back to Hierarchy Process. This approach was to update children in n-depth in one flow. There was Fast lookup inside a loop, but it would have been called only after hierarchy level is looped through. In other words, if hierarchy is six level deep Fast update would have been called only six times. However, this flow was overly complicated and mixed up the ID hierarchies and sometimes it even missed some records. This alternative approach was not as reliable as the flow which starts another process.

The approach presented in this study can be optimized and developed to be faster. Especially, when Salesforce platform improves in data processing there can be more opportunities and solutions to populate hierarchy to a string. Previous update, Winter ‘16 version already improved the process in Process Builder since it got an update to bulkify processes.

Comparing fixed maximum depth in formula field to n-depth solution, formula seems to be more reliable, faster and less risky. N-depth solution can provide deeper hierarchy but when hierarchy is likely to not cross 15 levels, formula is the solution. It is also a good question whether the hierarchy should be that deep. Moreover, formula text field gives more functionalities to use in SOQL queries.

6 Conclusion

As this project faced continuous restrictions, it seems it is too difficult to display the hierarchy to configure the one single field. Some vulnerabilities were discovered, and the major one was uncertainty in scalability. There was not enough information about the bulkifying improvement of Process Builder and its impact to project results. Moreover, since Salesforce debug logs could not give information about how close the implementation was to hit governor limitations, solid information of the scalability could not be calculated. However, the solution did meet the core requirements which were set for this project.

After finishing the project, I met Jack van Dijk in Salesforce World Tour in Amsterdam on 9th of March 2017. He is a Salesforce Cloud Architect in Salseforce. Mr. van Dijk agreed that this project might have some issues when configuring the field. He pointed out another approach to display the hierarchy. Lightning component could be successful because it shows up to date visualization of the hierarchy instead of custom field. Since the project was to configure a custom field, Lightning component was not considered. Moreover, Lightning was still at beta stage when the project was started. Additionally, Lighting component would not have solved the scalability limit in other purposes where the custom field would have been utilized, for example in Apex Classes. [32]

References

1 Payne, Adrian. Handbook of CRM: Achieving Excellence in Customer Management. Great Britain: Butterworth-Heinemann; 2008.

2 What is CRM [online]. Salesforce.
URL: https://www.salesforce.com/eu/crm/what-is-crm.jsp. Accessed 11 April 2017.

3 Hollar, Katie. Top CRM Software [online]. Capterra. 7 July 2016
URL: http://www.capterra.com/customer-relationship-management-software/#infographic. Accessed 11 April 2017.

4 Prophet, Tony. 2017. Chief Equality Officer, Salesforce, United States. Presentation, 9 March 2017.

5 Salesforce1 [online]. Google Play.
URL: https://play.google.com/store/apps/details?id=com.salesforce.chatter&hl=fi. Accessed 13 April 2017.

6 Zoho CRM [online]. Google Play.
URL: https://play.google.com/store/apps/details?id=com.zoho.crm&hl=fi. Accessed 13 April 2017.

7 Cummings, J., Farber, D. 2017. Salesforce Announces Fiscal 2017 Fourth Quarter and Full Year Results [online]. Salesforce Investor. 38 February 2017.
URL: http://investor.salesforce.com/about-us/investor/investor-news/investor-news-details/2017/Salesforce-Announces-Fiscal-2017-Fourth-Quarter-and-Full-Year-Results/default.aspx. Accessed 11 April 2017.

8 The Force.com Multitenant Architecture [online]. Salesforce Developers.
URL: https://developer.salesforce.com/page/Multi_Tenant_Architecture. Accessed 11 April 2017.

9 Products – Thunder [online]. Salesforce.
URL: https://www.salesforce.com/products/platform/products/thunder/. Accessed 11 April 2017.

10 Lightning Design System: Understanding Key Principles behind the Design System [online]. Salesforce Trailhead.
URL: <https://trailhead.salesforce.com/modules/lightning_design_system/units/lightning-design-system1. Accessed 11 April 2017.

11 Icons [online]. Lightning Design Systems.
URL: https://www.lightningdesignsystem.com/icons/. Accessed 11 April 2017.

12 PaaS Multi Tenancy [online]. Oracle.
URL: http://www.oracle.com/technetwork/topics/cloud/paas-multi-tenancy-092593.html. Accessed 11 April 2017.

13 Haby Azure team. What's the difference between different cloud services like IaaS, PaaS and SaaS? [online]. Hanu Software. 16 august 2012. URL: http://hanusoftware.com/ whats-the-difference-between-different-cloud-services-like-iaas-paas-and-saas/. Accessed 13 April 2017.

14 Butler, Brandon. PaaS Primer: What is platform as a service and why does it matter? [online]. Network World. 11 February 2013. URL: http://www.networkworld.com/article/2163430/cloud-computing/ paas-primer--what-is-platform-as-a-service-and-why-does-it-matter-.html. Accessed 11 April 2017.

15 Products [online]. Salesforce. URL: https://www.salesforce.com/products/. Accessed 11 April 2017.

16 Understanding Packages [online]. Salesforce Help. URL: https://help.salesforce.com/articleView?id=sharing_apps.htm&type=0. Accessed 12 April 2017.

17 AppExchange – FAQ [online]. Salesforce. URL: https://www.salesforce.com/solutions/appexchange/faq/. Accessed 11 April 2017.

18 sObject Types [online]. Salesforce Developers. URL: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/ apexcode/langCon_apex_SObjects.htm. Accessed 11 April 2017.

19 Custom Field Types [online]. Salesforce Help. URL: https://help.salesforce.com/articleView?id=custom_field_types.htm& language=en&type=0. Accessed 11 April 2017.

20 Relationships Among Objects [online]. Salesforce Developers. URL: https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/ relationships_among_objects.htm. Accessed 12 April 2017.

21 Roll-Up Summary Field [online]. Salesforce Help. URL: https://help.salesforce.com/articleView?id=fields_about_roll_up_summary_fields .htm&type=0. Accessed 12 April 2017.

22 An Introduction to Formulas [online]. Salesforce Developers. May 2016. URL: https://developer.salesforce.com/page/An_Introduction_to_Formulas. Accessed 12 April 2017.

23 SOQL and SOSL Queries [online]. Salesforce Developers. URL: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/ apexcode/langCon_apex_SOQL.htm. Accessed 12 April 2017.

24 Execution Governor and Limits [online]. Salesforce Developers. URL: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/ apexcode/apex_gov_limits.htm. Accessed 11 April 2017.

25 Triggers [online]. Salesforce Developers. URL: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/ apexcode/apex_triggers.htm. Accessed 11 April 2017.

26 Trigger Context Variables [online]. Salesforce Developers.
URL: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_context_variables.htm. Accessed 11 April 2017.

27 Process Automation: Automate Basic Business Processes with Process Builder [online]. Salesforce Trailhead.
URL: https://trailhead.salesforce.com/modules/business_process_automation/units/process_builder. Accessed 11 April 2017.

28 Process limits [online]. Salesforce Help.
URL: https://help.salesforce.com/articleView?id=process_limits.htm&language=en_US&type=0. Accessed 11 April 2017.

29 Flow Types [online]. Salesforce Developers.
URL: https://developer.salesforce.com/docs/atlas.en-us.salesforce_vpm_guide.meta/salesforce_vpm_guide/vpm_admin_flow_type.htm. Accessed 11 April 2017.

30 Set Flow Variable from a Flow URL [online]. Salesforce Developers.
URL: https://developer.salesforce.com/docs/atlas.en-us.salesforce_vpm_guide.meta/salesforce_vpm_guide/vpm_url_setvar.htm. Accessed 11 April 2017.

31 Reduced Chances of Hitting SOQL Limits in Processes [online]. Salesforce Release Notes.
URL: https://help.salesforce.com/articleView?id=000230637&type=1. Accessed 13 April 2017.

32 van Dijk, Jack. 2017. Salesforce Cloud Architect, Salesforce, Netherlands. Conversation, 9 March 2017.

Process Builder View