TABLE OF CONTENTS

1 Step 1: Formulating the business case

Name: *** **MEMORANDUM**

Data Analyst
Health-analytics.ai
00100 Helsinki

Date: 31.05.2021

Mr. XYZ

Senior Manager
Health-analytics.ai
00100 Helsinki

REGARDING NECESSITY OF DEVELOPING A DATABSE MANAGEMENT SYSTEM

Dear Mr. XYZ,

We, currently, are using a traditional file processing system (the software we use to manage and organize the files in a storage medium) in our company which is time-consuming and not that efficient as data redundancy, inconsistency, lack of data integration, isolation, data loss, poor data control, etc. are associated with it. Further to this, security and dependency on application programs often have been issues that we must address. Apart from this, for our growing consultancy business, current way of storing detailed project data locally in Excel seems not sufficient to deeply understand our clients and analyze our consultants' performances. Our growing collection of data requires more sophisticated automated process for our desired efficiency and due to the shortcoming of our current structure we often fail to automate the tracking of clients and projects.

In this case, I would like you to reconsider our current structure. In fact, I would like to highly recommend for a Database Management System/DBMS approach along with data warehousing which will help us overcome all the issues we have associated with our current structure. The DBMS is well-structured, and it has also more flexibility than our current system. This system allows smooth concurrent access to the same data. In case of any data loss, it is possible to have a full recovery of the lost data as this system keeps the backup. With its facilitated variety of techniques of data storing and retrieving, we can store and organize different types of data in this system and perform faster data extraction, modification, and data processing conveniently. In addition, data warehousing will allow us to load and store our historic data in such a meaningful way that can be later used to transform those data into our information assets. Thus, the total structure will help us in our whole ETL process (Extract, Transform and Load) and improving performance in terms of data analysis and overall decision-making process.

However, to create the database, we will need the data types that represent all possible values, improve data integrity, support data manipulations, and minimize storage space. In this case, our possible set of data types include VARCHAR2, CHAR, CLOB, NUMBER, DATE, BLOB. To implement our database solution, we can consider generic systems development life cycle / SDLC approach. Hence, we need to plan our enterprise modeling and conceptual data modeling. In this step, we will analyze our current data processing and general business functions and their database needs. Thus, we will have a preliminary understanding of our business situation. This will also help us discover any possible opportunities. Once, the planning phase is done, we can then analyze our conceptual data modeling where we will determine our requirements and check the feasibility of the project. We aim to have our conceptual schema in this phase. Then, our third phase will be designing the logical database where we will transform the conceptual schema into logical schema. This is where the actual creation of the system will take place. There will be a physical schema to describe how data from our logical schema are stored. Finally, we will reach to our implementation phase where we will install the database and convert all the data from our old systems. To meet our changing business condition, we will have routine maintenance as well. This will also help us remove contaminated data and fix any possible errors.

Thank you in advance for your consideration. I look forward to hearing from you soon.

Yours sincerely,
******

2 Step 2: Conceptual design

2.1 Task 1: Assessment of the provided information and further improvement scope

The four provided entities- Consultants, Customer, Location and Service seem to be sufficient for now. However, considering the main business objectives, I would like to add and modify some attributes.

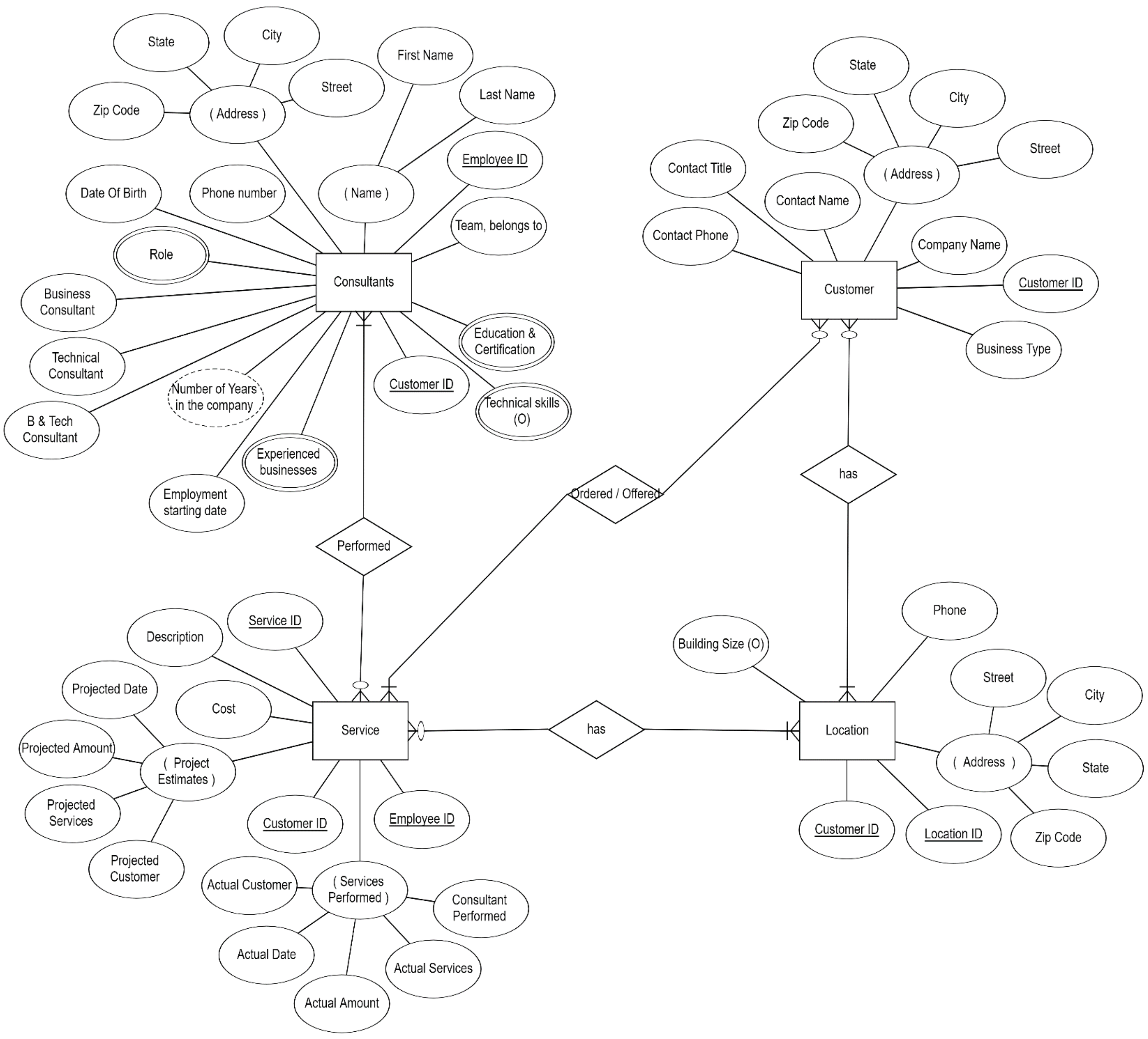
For example, it is important to know the team and team lead that the consultant belongs to. In this case, I recommend adding an attribute 'Team, belongs to' in the Consultants entity. Name of the consultants can be divided into two parts- 'First Name' and 'Last Name' and in this case, 'Name' will be a Composite attribute. Even though, the consultants' 'Role' is a Multivalued attribute, for creating a good team mix and efficient allocation of manpower, I would recommend to specify their exact skillset. In this case, adding three more attributes- 'Business Consultant', Technical Consultant, and 'B & Tech Consultant' would make more sense. In the same entity, the attribute 'Number of Years in the company' should be as Derived attribute which can be calculated by the 'Employment starting date'. Another attribute can be added in the Consultants entity to record their education and certifications. I recommend it to be a Multivalued attribute and that is 'Education and Certification'. I assume the Type of Business [or businesses] the consultant is experienced in is a Multivalued attribute as the consultants might have experiences in multiple business types; I named it as 'Experienced businesses'. The attribute 'Technical skills' is suggested as Multivalued and Optional. Finally, for referencing purpose, I recommend adding the 'Customer ID' as a Unique attribute here in this entity.

Under the 'Location' entity, the attribute 'Building Size' seems to me not that important. In that case, I would consider it as optional. For the additional business rules, rather than having new entities, I recommend those to be attributes under the 'Service' entity as we first do the estimation for the project and then the actual services are performed. In this case, both the 'Project Estimates' and the 'Services Performed' will be Composite attributes. However, I would like to recommend two more Unique attributes- 'Customer ID' and 'Employee ID' (referring Consultants) to be added in the 'Service' entity.

2.2 Task 2: ER diagram, Design choices documentation, entity relationship description

2.2.1 Drawing the ER Diagram

Based on the previous discussion and the given information the following ER diagram has been drawn using erdplus-



2.2.2 Design choices documentation and entity relationship description

The reason I recommend adding the Unique attributes to the ‘Service’ entity is, for example, if we think of joining the entity tables, these two attributes will help in referencing. Otherwise, the attributes we had in that entity were not sufficient for basing. The same reason holds true for the attribute ‘Customer ID’ in the ‘Consultants’ entity.

Furthermore, Consultants might have Zero to Many Customers, whilst a Customer has (must) at least One Consultant or even more. In other words, Consultants perspective is optional that can be Zero to Many. On the other hand, from Customer perspective the relationship is a Mandatory One or Many. As, the relationship of the Consultants and Customers are captured through the Service entity, we do not need to have a direct relationship between these two entities.

However, a Customer must have at least One Location and the number of Location can exceed that, whilst each Location might have Zero to Many Customers. Therefore, this ‘has’ relationship is Mandatory One or Many from Customer perspective. On the contrary, it is an Optional relationship from the Location perspective which might be Zero or even Many.

The relationship between Location and Service is Mandatory One or May from Service perspective while it is Optional (Zero to Many) from Location perspective. Each location may have Zero or even Many Service call while the Service will be performed at least at One Location or even Many.

To say about the relationship between Consultants and Service, each Service will be performed by at least One or even Many Consultants and that is the reason that from Service point of view it is a Mandatory relationship which can be One or Many. In contrast, not all Consultants might perform the Service while even a Consultant might perform more than one Service, and therefore from Consultants perspective it is an Optional relationship- Zero to Many.

From Customer perspective, the relationship between Customer and Service is Mandatory (One or Many) as because, to include Customer in our database the respective Customer Must place an Order. On the other hand, from Service perspective, it is an Optional relationship, meaning, we can offer our service to Zero to Many Customers.

3 Step 3: Logical design

3.1 Task 1: Relational and Non-relational Database- NoSQL in question

Both the Relational and Non-relational database systems offer some advantages. It depends on how much data we are dealing with.

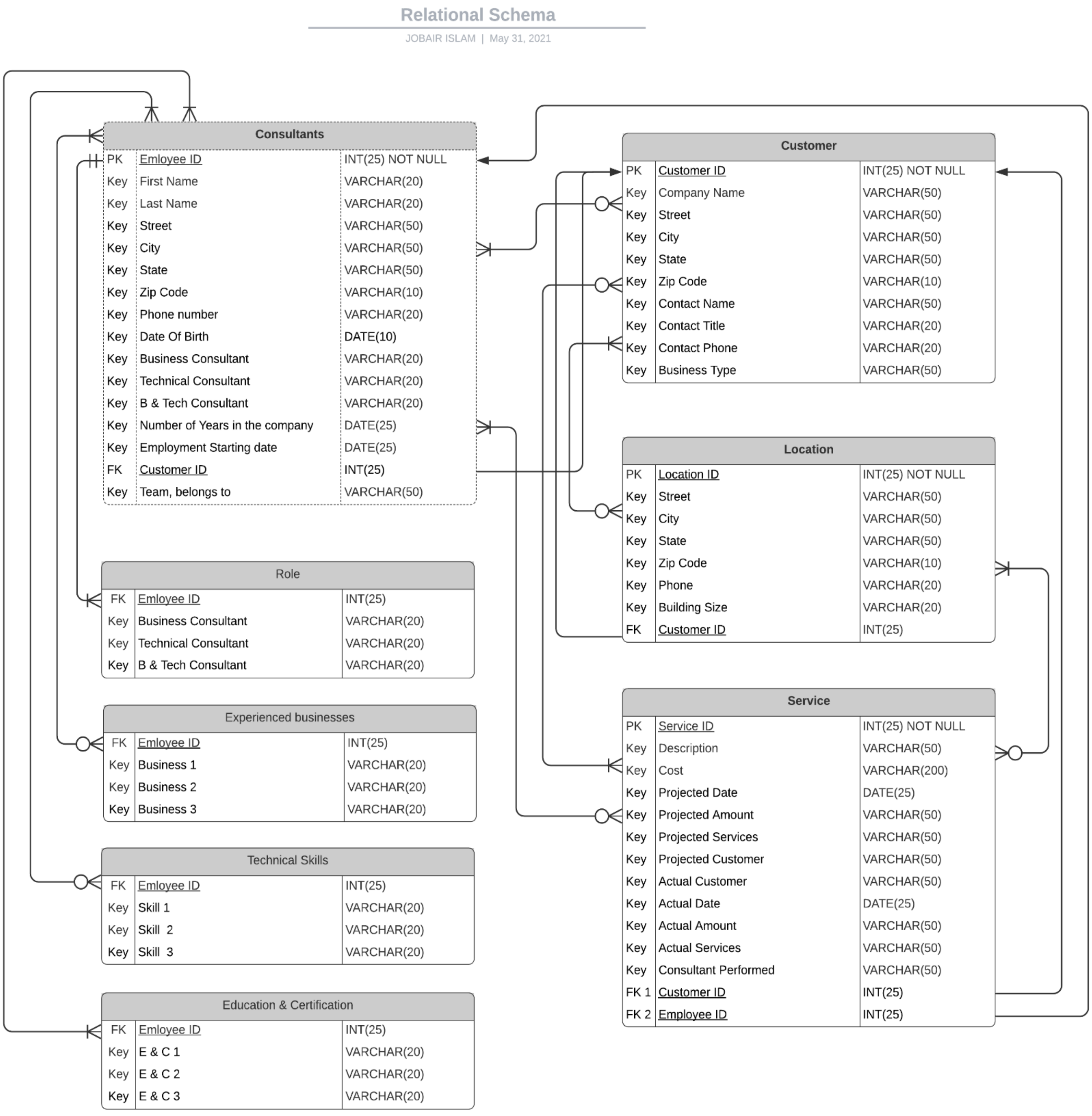
Nonetheless, based on our current amount of data and even considering the near future, Relational database will be the best option to go for. Data is stored in an organized way with columns, rows, and tables in Relational database. Due to its well-structured nature and consistency, it gives an ease in navigation and relationships of the data points can easily be defined. Its easy query for information retrieval, accuracy and reliability will provide us advantages in our business. On top of these, it is flexible, it has a normalized approach to data storage and data organization. We can easily set different access level for the users based on the data sensitivity and confidentiality and therefore, it is safe (Ilyukha, (n.d)).

In contrast to the Relational database, the Non-relational database (NoSQL / Not Only SQL) is non tabular, meaning, it is not that structured or confined even though columns and rows are used to enter data types and its values and keys are used to identify objects. This is why data of this database is called unstructured data. However, different types of data, such as- videos, photos, etc. can be stored in this database. Sometimes, a pre-defined schema is not suitable for fitting critical data and in that case this Non-relational database offers more flexibility. The general rule of thumb is- the bigger the data set, the more likely a NoSQL database is a better fit (White, 2020). Its quick updatability with assigned keys gives it agility where switching between the tables is not required. It is considered as an Open-source as most of NoSQL functional solutions are available for free. One more advantage of this database is it does not require any SQL knowledge to execute queries.

As we often need to work with complex queries and reports and we have a high transaction application, until our amount of data to be large enough for next couples of years the Relational database will be the best option for us. On the other hand, I am happy to see our management's concern about NoSQL. I believe, for our growing consulting service, in future, we can consider developing the NoSQL database as well to deal with all our unstructured data.

3.2 Task 2: Mapping the developed ER diagram to relational schema

The previously developed ER diagram to relational schema has been mapped below using lucidchart-



Consultants

PK	Emloyee ID	INT(25) NOT NULL
Key	First Name	VARCHAR(20)
Key	Last Name	VARCHAR(20)
Key	Street	VARCHAR(50)
Key	City	VARCHAR(50)
Key	State	VARCHAR(50)
Key	Zip Code	VARCHAR(10)
Key	Phone number	VARCHAR(20)
Key	Date Of Birth	DATE(10)
Key	Business Consultant	VARCHAR(20)
Key	Technical Consultant	VARCHAR(20)
Key	B & Tech Consultant	VARCHAR(20)
Key	Number of Years in the company	DATE(25)
Key	Employment Starting date	DATE(25)
FK	Customer ID	INT(25)
Key	Team, belongs to	VARCHAR(50)

Customer

PK	Customer ID	INT(25) NOT NULL
Key	Company Name	VARCHAR(50)
Key	Street	VARCHAR(50)
Key	City	VARCHAR(50)
Key	State	VARCHAR(50)
Key	Zip Code	VARCHAR(10)
Key	Contact Name	VARCHAR(50)
Key	Contact Title	VARCHAR(20)
Key	Contact Phone	VARCHAR(20)
Key	Business Type	VARCHAR(50)

Location

PK	Location ID	INT(25) NOT NULL
Key	Street	VARCHAR(50)
Key	City	VARCHAR(50)
Key	State	VARCHAR(50)
Key	Zip Code	VARCHAR(10)
Key	Phone	VARCHAR(20)
Key	Building Size	VARCHAR(20)
FK	Customer ID	INT(25)

Role

FK	Emloyee ID	INT(25)
Key	Business Consultant	VARCHAR(20)
Key	Technical Consultant	VARCHAR(20)
Key	B & Tech Consultant	VARCHAR(20)

Experienced businesses

FK	Emloyee ID	INT(25)
Key	Business 1	VARCHAR(20)
Key	Business 2	VARCHAR(20)
Key	Business 3	VARCHAR(20)

Service

PK	Service ID	INT(25) NOT NULL
Key	Description	VARCHAR(50)
Key	Cost	VARCHAR(200)
Key	Projected Date	DATE(25)
Key	Projected Amount	VARCHAR(50)
Key	Projected Services	VARCHAR(50)
Key	Projected Customer	VARCHAR(50)
Key	Actual Customer	VARCHAR(50)
Key	Actual Date	DATE(25)
Key	Actual Amount	VARCHAR(50)
Key	Actual Services	VARCHAR(50)
Key	Consultant Performed	VARCHAR(50)
FK 1	Customer ID	INT(25)
FK 2	Employee ID	INT(25)

Technical Skills

FK	Emloyee ID	INT(25)
Key	Skill 1	VARCHAR(20)
Key	Skill 2	VARCHAR(20)
Key	Skill 3	VARCHAR(20)

Education & Certification

FK	Emloyee ID	INT(25)
Key	E & C 1	VARCHAR(20)
Key	E & C 2	VARCHAR(20)
Key	E & C 3	VARCHAR(20)

NB: The Multivalued attributes have been shown as new entities in the map. ‘Employee ID’ has been used as a foreign key for all those new entities.

3.3 Task 3: Assessment of the model based on its current normal form and further transformation

3.3.1 Model assessment

Database normalization is a technique used to organize data. It is a systematic approach to decompose tables for data redundancy elimination. With this multi-step process data is put into tabular form while duplicated data is removed from the relational tables. The purposes of this approach are to eliminate data redundancy and to ensure that the data dependencies make enough logical sense. The idea is having resources /data in the database with proper application. Therefore, basically removing data anomalies, such as- updation, insertion, deletion, is the main purpose of the database normalization approach. There are six different steps in normalization- first normal form/1NF, second normal form/2NF, third normal form/3NF, Boyce-Codd normal form/BCNF, fourth normal form/4NF, and fifth normal form/5NF.

Based on the characteristics of the first normal form, we can say that our relational schema is now in first normal form. For example, while mapping the ER diagram to relational schema, we have tackled the problem of atomicity as we have treated the composite and multivalued attributes. Therefore, there are not any multiple values in any columns. In addition, we also have defined the primary key as a unique identifier of each row in the relation. Thus, we have already achieved the first normal form.

3.3.2 Transforming to second normal Form

As we know, a model to be set in second normal form it must fulfill two conditions- firstly, it must be in first normal form and secondly, it should not contain any kind of partial functional dependency. Our first condition is met already in the previous step. Now, we need to treat the model for the second condition fulfilment.

A partial dependency refers to the determination of non-prime attribute by proper subset of a candidate key. Attributes that are not a part of the candidate key are called non-prime attributes. For example, in the model, the Location table has the ‘Customer ID’ attribute as the proper subset of the candidate key but that does not determine other non-prime attributes, meaning, Street, City, State, Zip Code, Phone, and Building Size do not rely on this, instead, these are related to the ‘Location ID’.

To convert our model into second normal form, we can split the Location table in two parts- one with the attributes ‘Customer ID’ and ‘Location ID’, another with the attributes ‘Location ID’ and those non-prime attributes. Now in the first part, the column Location ID is dependent on ‘Customer ID’ while in the second table, the non-prime attributes are fully dependent on only the ‘Location ID’. Thus, we have removed the partial functional dependency that we had before, and we have achieved the second normal form of that Location table.

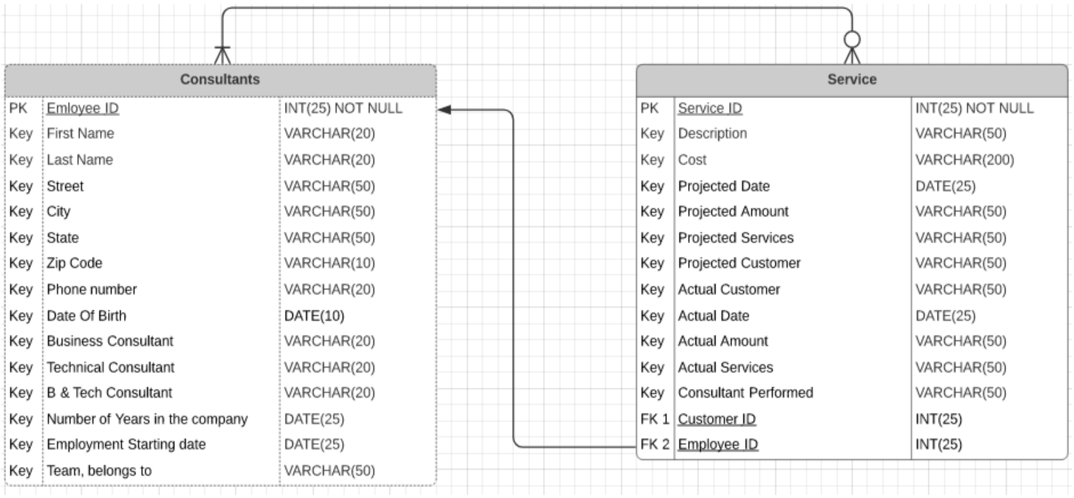
However, normalization for other tables of the model is out of scope. Therefore, it can be said that our whole model is now converted to second normal form.

4 Step 4: Physical database design and performance

4.1 Task 1: Denormalization benefits

First of all, it is important to be clear that denormalization does not necessarily mean not doing the normalization, rather a normalized database is prerequisite of denormalization. It is a mechanism by which data from multiple tables are combined into a single table.

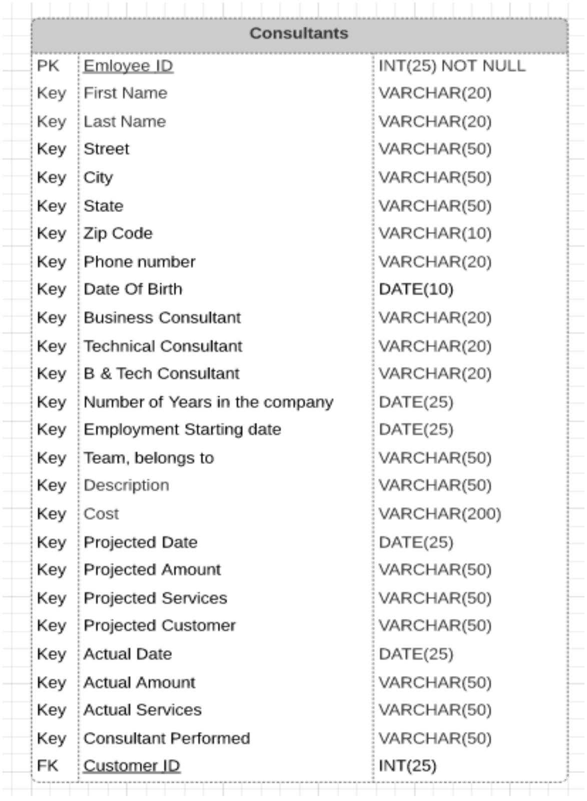
To improve overall database performance, such as- query performance enhancement, relatively more convenient management of the database, acceleration of reporting, etc. the database denormalization technique is used. It aims at giving a boost to the performance by minimizing the running time of the queries. In this optimization technique we allow redundant data to one or more tables to avoid costly and complex joining. Thus, in some cases, denormalization reduces the number of tables in the database. It helps quick access to stored data and retrieve information with simpler queries. For example, in our model, two entities with many-to-many relationship (Consultants and Service) can be denormalized. Currently normalization form of both the entities looks like the following-



Consultants		
PK	Emloyee ID	INT(25) NOT NULL
Key	First Name	VARCHAR(20)
Key	Last Name	VARCHAR(20)
Key	Street	VARCHAR(50)
Key	City	VARCHAR(50)
Key	State	VARCHAR(50)
Key	Zip Code	VARCHAR(10)
Key	Phone number	VARCHAR(20)
Key	Date Of Birth	DATE(10)
Key	Business Consultant	VARCHAR(20)
Key	Technical Consultant	VARCHAR(20)
Key	B & Tech Consultant	VARCHAR(20)
Key	Number of Years in the company	DATE(25)
Key	Employment Starting date	DATE(25)
Key	Team, belongs to	VARCHAR(50)

Service		
PK	Service ID	INT(25) NOT NULL
Key	Description	VARCHAR(50)
Key	Cost	VARCHAR(200)
Key	Projected Date	DATE(25)
Key	Projected Amount	VARCHAR(50)
Key	Projected Services	VARCHAR(50)
Key	Projected Customer	VARCHAR(50)
Key	Actual Customer	VARCHAR(50)
Key	Actual Date	DATE(25)
Key	Actual Amount	VARCHAR(50)
Key	Actual Services	VARCHAR(50)
Key	Consultant Performed	VARCHAR(50)
FK 1	Customer ID	INT(25)
FK 2	Employee ID	INT(25)

If we want to analyze how the experience of our consultants affect the overall service including taken time, quality of service, etc. we need to perform a join operation first which often results to a slower read. Apart from this, it also takes heavy quarrying that can overwhelm and crash hardware. These issues are not in line with our business goals. In this case, we may consider the denormalization of these two tables which will look like the following-



Consultants		
PK	Emloyee ID	INT(25) NOT NULL
Key	First Name	VARCHAR(20)
Key	Last Name	VARCHAR(20)
Key	Street	VARCHAR(50)
Key	City	VARCHAR(50)
Key	State	VARCHAR(50)
Key	Zip Code	VARCHAR(10)
Key	Phone number	VARCHAR(20)
Key	Date Of Birth	DATE(10)
Key	Business Consultant	VARCHAR(20)
Key	Technical Consultant	VARCHAR(20)
Key	B & Tech Consultant	VARCHAR(20)
Key	Number of Years in the company	DATE(25)
Key	Employment Starting date	DATE(25)
Key	Team, belongs to	VARCHAR(50)
Key	Description	VARCHAR(50)
Key	Cost	VARCHAR(200)
Key	Projected Date	DATE(25)
Key	Projected Amount	VARCHAR(50)
Key	Projected Services	VARCHAR(50)
Key	Projected Customer	VARCHAR(50)
Key	Actual Date	DATE(25)
Key	Actual Amount	VARCHAR(50)
Key	Actual Services	VARCHAR(50)
Key	Consultant Performed	VARCHAR(50)
FK	Customer ID	INT(25)

As, here we already have the needed data in this table, we will not need to perform any other join operation which will give us an advantage with easy and fast access to the data and thus we can retrieve our desired information easily.

Further to this, denormalization can also be introduced in case of Location and Service entities which are in many-to-many relationship. To analyze, for instance, in which location we tend to have more service calls or which location takes more service time we need to perform a join operation with our current normal form which problematic in terms of time and risk with hardware failure or slowing down the system. On the other hand, if we do the denormalization, we can easily have access to those data with a single table.

Thus, denormalization serves our business goals in a more meaningful way.

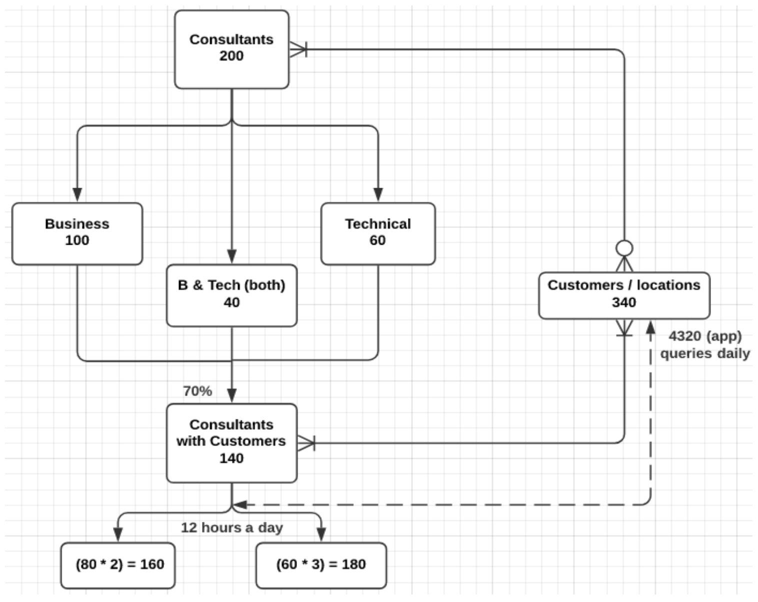
4.2 Task 2: Basic usage analysis

When we think of large-scale database implementation, there is an importance of continuous monitoring for any significant changes in data volume and frequency of use statistics. For maintaining a good understanding of the data volumes and usage patterns, throughout the life cycle of the database, it is necessary to have the inputs from the designing phase (Hoffer, Ramesh & Topi, 2016).

We have 200 consultants and 100 customers. We can divide our consultants in 03 parts-

a. 100 consultants are business experts,
b. 60 consultants are technical experts and
c. rest 40 consultants are equally expert in both the areas.

Along with this, each of our customers operate in multiple locations (3-4), therefore, from location perspective the numbers of our customers are around 340. It is also possible that some of the consultants will not have any customers while some may have more than one (2-3) customers and we have limited office hours and scope of queries. Hence, we assume that, uninterruptedly 70% of the consultants from each three parts run queries 12 hours a day, which means 70 consultants from part a, 42 consultants from part b and 28 consultants from part c will run queries for 12 hours (on average) daily basis for all 340 customers. We also assume that, among those total 140 serving or active consultants, 80 consultants have 2 customers each while rest 60 consultants have 3. Then the total queries will be around (80 * 2 * 12) + (60 * 3 * 12) = 4320 per day.



Further note:

It is not that important to have the precise numbers, but the estimation of the usage patterns is necessary. However, attributes to the entities must be stored efficiently to avoid a problematic system. In addition to that, for all three types of consultants (expertise wise), we may consider separate tables for the purpose of efficient allocation of manpower according to customer needs. Moreover, denormalization will be a visionary step for efficient use of our database.

5 Step 5: Data Warehousing and data integration

5.1 Task 1: Benefits of EDW over Independent Data Marts

In our company, we have some independent data marts in separate departments, such as-finance, sales, HR, etc. Even though this structure is cost effective, it is not sufficient for our company wise data analysis due to its limited data set. For our expanding business, it would be wise to opt for Enterprise Data Warehousing/EDW.

While data mart is now focused on specific functional area, EDW will support all (or most) of the enterprise. If we look at the typical memory size, the difference will be clear, that data marts have limited memory (less than 100 GB) while EDW can be larger (1 TB+). While our current structure is associated with only summarized data, the EDW can hold each type of data, be it raw, summarized or metadata. It can track and modify marketing campaigns which will allow us to reach our target customers. Even though, it initially seems to be costly, in a long run it will reduce cost with business intelligence and increase our ROI. Apart from this, EDW will give us an extra edge to maintain security, consistency, and quality data. It also contains historical data which will enable our analytics team to examine data trends. On top of that, this will help us in our overall strategic decision making for the entire organization (Salesforce, n.d.).

However, even though real-time data warehousing has some benefits, there are some technical aspects and risks associated with it, for example, system downtime. Another issue is the open source framework- Hadoop is not compatible. Further to this, real-time data warehousing will require completely a new approach due to its huge volume of instant data production and as we are not used to handling data in such rapid rate, it will be troublesome approach for us. Therefore, I do not think we will require any real-time data warehousing as we hardly have any transactional database. Instead, we can have a ‘Near real-time’ approach with increased frequency in data loading, for example, everyday basis and if needed we can also upgrade this frequency to twice a day or even hourly basis (Langseth, (n.d.), Pal, 2018).

5.2 Task 2: Regarding manual entry

In the Service entity, there are some attributes that need manual entry, such as- ‘Description’, ‘Projected Services’ and ‘Actual Services’. Further to this, the attribute ‘Business Type’ in the Customer entity also may require manual entry. Finally, due to our consultants’ ongoing skill developments, we might manually enter the skills and certification to their respective attributes in the Consultants entity at some point as well.

References

Hoffer, A. J., Ramesh, V. & Topi, H. (2016). Modern Database Management. Pearson Education, Inc.

Ilyukha, V. (n.d.). Differences Between Relational and Non-Relational Database. Retrieved on 24 May 2021 from https://jelvix.com/blog/relational-vs-non-relational-database

Langseth, J. (n.d.). Real-Time Data Warehousing: Challenges and Solutions. Retrieved on 31 May 2021 from http://dssresources.com/papers/features/langseth/langseth02082004.html

Pal, K. (2018). Weighing the Pros and Cons of Real-Time Big Data Analytics. *Technology Trends.* Retrieved from https://www.techopedia.com/2/31245/technology-trends/big-data/weighing-the-pros-and-cons-of-real-time-big-data-analytics

White, A. (2020). What's the Difference? Relational vs Non-Relational Databases. Retrieved from https://www.izenda.com/relational-vs-non-relational-databases/

Salesforce.(n.d). Advantages of Implementing an Enterprise Data Warehouse. ENTERPRISE DATA WAREHOUSE. Retrieved from https://www.salesforce.com/products/crm-analytics/resources/enterprise-data-warehouse/