# Name of Your Company

H&M

Database of small companies

# Team

* Mark Wade Brakstad (MySQL Expert)
* Henrique Reis Neves (MS SQL Server Expert)

# Weekly Meeting Hours

We settle on Mondays at 10am. In case we need a second meeting, we will do it on a Friday of the same week at 4:30pm

# Project Description

This database will keep record of all small businesses, their services, the customers that receive services from these companies, the detail of deals between customers and the businesses, and reviews(comments) written for those businesses.

Businesses have name, several addresses (several branches), several phone numbers, list of services they provide, fees and costs for they services, the area they provide their services, number of customers they had so far…

Customers have name, phone number, address(s) (a customer may have several houses…)…

Reviews (comments) are composed of score (number of starts), comment, date…

Deals done between customers and businesses are recorded: date, fee/cost agreed, services provided, location that the services were provided, the name of the worker, the name of the company…

(this description is a draft description and must be completed gradually as the database is developed to reflect the complete description of the final database. The description should contain information about entities, attributes, relationships, cardinality ratios, and constraints…)

# Assumptions about Cardinality and Participations

You can write all the assumptions about Cardinality and Participations (total/partial) here.

* Business have 1 to many services
* Businesses have 1 to many employees
* Businesses do work on 1 to many contracts
* Employees work on one to many contracts
* Contracts are located at 1 to many addresses
* Customers have 1 to many Addresses
* Customers can give 0 to many Reviews
* There can be 0 to many reviews for businesses
* Services form 1 to many contracts
* Customers can buy 1 to many contracts

More will be added in the future as we start modeling and developing the database.

# EER Modeling Diagram

In the following drawing canvas, EER Modeling shapes have been provided. You can copy and replicate them (Ctrl+C to copy and Ctrl+V to paste. You can also select a shape, then press Ctrl button and drag and drop to copy a shape) and edit them to build your diagram.

Have

Businesses

Customers

Reviews

Give

For

0, N

0, N

0, N

1, N

Contract

Do work

Buy

1 , N

1 , N

1 , N

1 , N

Services

Have

1 , N

1 , N

Form

1 , N

1 , N

Employee

Have

1 , N

1 , N

Work on

1 , N

1 , N

Address

Located

1 , N

1 , N

# ER-Model Mapping to Database Relational Schema

Businesses (BusinessID, Addresses, Name, Cost & Fees, Services, Number of Costumers)

BusinessPhoneNumber (**BusinessID**, Phone number)

Customers (CustumerID, Name, Phone)

Employee (EmployeeID, Start Date, **BusinessID**)

Services (ServiceID, Cost & Fees, Type, **ContractID, BusinessID**)

Contracts (**BusinessID, CustomerID,** ContactID, Date, Agreed fee/cost)

ContractAddress (**CustomerID**, AddressID, **ContractID, BusinessID**)

Reviews (**CustumerID, BusinessID**, ReviewID, Comment, Score)

EmployeeWorksOnContract (**EmployeeID**, **ContractID,** **BusinessID, CustomerID**)

# Normalization

All relations must be normalized up to BCNF. You must explain why you believe every relation in your database in normalized.

Our relations in our database satisfy 1st normal form as each value is atomic. This is shown as each column has a unique name. The only attribute that could have violated this rule as shown in our ER-model would have been the multi-valued attribute(Phone number) which has been broken off into a separate table to satisfy 1st normal form.

Our relations in our database satisfy 2nd normal form as most attributes currently in our schema are either single primary keys, or if they are composite, the other attributes rely on all parts of the primary key making them have full dependency on them. In the future if more attributes are needed and they do not satisfy this normal form, we will break down the tables into separate tables with whatever attribute the new attribute depends on and this will satisfy 2nd normal form.

Our relations in our database satisfy 3rd normal form as there is no transitive dependency currently within our database attributes. Every attribute wholly relies on the primary key and has full dependency on it. If in the future if there are new attributes with transitive dependencies on attributes not part of the primary key, we will make a new parent table with both attributes and make the attribute that is being depended on the primary key of this new table and a foreign key in the original table, thus satisfying 3rd normal form.

Our relations in our database satisfy BCNF as none of the non-prime attributes could derive any of the prime attributes in their table. This is true as all of our keys are IDs of each table, meaning each attribute which is a part of the table will be derived off of their specific primary key. If you tried to derive the primary key using non-primary attributes you would not be able to.

# Determining Data Types (Domain) and Constraints

For the different data types we used in our sql scripts, we used int values in several different tables. The general rule we had when using the int data type is if we needed a number value but were not too particular about how many numbers were needed. We also used a decimal data type for the cost and fees value in our business table. This makes It easier to keep the values similar as these are all small businesses and their cost and fees will be relatively similar. We also used the date data type whenever we needed a date as our value in the table. We also similarly used both char and varchar where we found them applicable. In cases such as Comment where the user can write a comment of their choosing in length, we chose to leave the user with the decision of the length. But when dealing with values such as address, it makes sense to limit the number of characters usable to make it more realistic as not many addresses are too long. For constraints we used two throughout our scripts. We used not null in important cases such as BusinessName where the values are essential and should not be left blank, and we also used on delete cascade in the foreign keys so that If rows are deleted in a parent table, the subsequent child table rows are deleted as well.

# Creating Database and Tables - SQL DDL

You do not need to copy SQL commands here. Save your SQL commands in a script file and just mention the name of the file here. Make sure the script file is stored besides this document within the same folder.

# Inserting Values in Tables

You do not need to copy SQL commands here. Save your SQL commands in a script file and just mention the name of the file here. Make sure the script file is stored beside this document within the same folder.

# SQL Queries

You do not need to copy SQL commands here. Save your SQL commands in a script file and just mention the name of the file here. Make sure the script file is stored beside this document within the same folder.

# Views

You do not need to copy SQL commands here. Save your SQL commands in a script file and just mention the name of the file here. Make sure the script file is stored beside this document within the same folder.

Have