# Project Title

Babysitting company project

# Project Description

The database store data about babysitters, families, and details of contracts between them.

Data about babysitters: name, date of birth, highest education degree, skills (painting, singing, playing guitar, stand-up comedy…), availability time (days of week, from-to time,), gender, occupation (babysitting can be the second job of the person), SSN, nationality, languages that the person can speak….

Data about families: full name of the family representative, address, phones, emails, marital status (married, divorced, single parent…)

Data about the child that needs babysitting: name, date of birth, age, gender, disability (if the child has any), languages spoken…

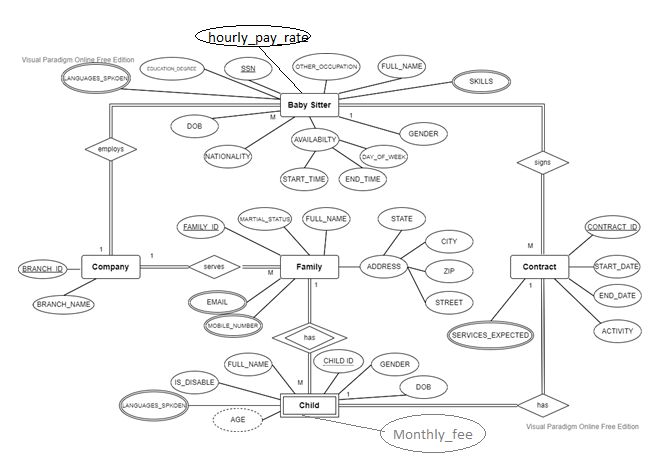
Data about contract: start and end date, the schedule of babysitting (time and activity done during that time), services expected from babysitter…

# Assumptions about Cardinality and Participations

Entities – Company, Babysitter (Employee), Families (Customers), Child, Contract.

* Company employs Babysitters (Employees)
* Company provide services to the Families (Customers)
* Families have children
* Company has a contract with the babysitters

# EER Modeling Diagram



# ER-Model Mapping to Database Relational Schema

* Company (branch\_id, branch\_name)

**1 to Many relationship & total participation between Company and Babysitter:**

Used foreign key in many side relation (baby\_sitter).

* Baby\_sitter (SSN, Full\_name, DOB, Nationality, Gender, start\_time, end\_time, day\_of\_week, other\_occupation, highest\_education\_degree, hourly\_pay\_rate **branch\_id**)
* Baby\_sitter\_language ( **SSN**, language\_spoken)
* Baby\_sitter\_skill ( **SSN**, skill )

**1 to Many relationship & total participation between Babysitter and Contract:**

Used foreign key in many side relation (Contract).

* Contract(contract\_id, start\_date, end\_date, activity, **SSN**)
* Contract\_services (**contract\_id**, service\_expected)

**1 to Many relationship & total participation between Company and Family:**

Used foreign key in many side relation (family).

* Family (family\_id, full\_name, marital\_status, street, city, state, zip, **branch\_id**)

* Family\_email (**family\_id**, email)

* Family\_mobile\_number (**family\_id**, mobile\_number)

**1 to Many relationship between Family with child as weak entity and total participation:** Used foreign key in many side of the relation (Child)

**One to One relationship between Contract and Children:** Used foreign key in one of the relation (Child).

* child (**family\_id**, child\_id, full\_name, Date\_Of\_Birth, gender, is\_disable, monthly\_fee **contract\_id**)

* Child\_languages\_spoken(**family\_id, child\_id**, language\_spoken)

# Normalization

1 NF –

We have already decomposed all the multi-valued attributes and composite attributes while designing relationship schema. So, all relations are normalized up to 1NF.

2 NF –

These are the relations where primary key contains multiple attributes –

Baby\_sitter\_languages, Baby\_sitter\_skills, contract\_services, Family\_email, Family\_Mobile\_number, child, Child\_languages\_spoken

We have verified and found that no non key attribute is functionally dependent on a part of the primary key. So, our relations are normalized up to 2NF.

3 NF –

We have verified all the relations for transitive dependencies. There is no relation that has a non key attribute functionally determined by another non key attribute or by a set of non key attributes. So, our relations are normalized up to 3NF.

BCNF –

We have verified all the relations for BCNF. We found that there is no functional dependency in any of them that a non prime attribute could determine a prime attribute.

# Determining Data Types (Domain) and Constraints

You explain why you choose a certain data type for a field and why you apply certain constraints

***Company table –***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Branch\_Id | Integer –  Because we want a numeric value to easily identify each branch | Primary Key –  Each branch should have unique identifier |
| Branch\_name | Varchar (255) –  We used this data type to have a variable length for the name of the branch and we know the actual data will be way less than the capacity mentioned. | Not Null –  Every branch should have a valid name |

***Family table –***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Family\_Id | Integer –  Because we want a numeric value to easily identify each family | Primary Key –  Each family registered in the database should have an unique identifier |
| Full Name | Varchar (255) –  We used this data type to have a variable length for the name of the parent and we know the actual data will be way less than the capacity mentioned. | Not Null –  Name should be mandatory for the parent to contact and talk to them |
| Marital Status | Varchar (20) –  Possible values are – Married/ Single/ Other. So, we mentioned maximum length to be 20. |  |
| Street | Varchar (255) –  We used this data type to have a variable length for the name of the street. |  |
| City | Varchar (255) –  We used this data type to have a variable length for the name of the city and we know the actual data will be way less than the capacity mentioned. |  |
| State | Varchar (255) –  We used this data type to have a variable length for the name of the state and we know the actual data will be way less than the capacity mentioned. |  |
| Zip | Varchar (255) –  We used this data type to have a variable length for the name of the ZIP and we know the actual data will be way less than the capacity mentioned. |  |
| Branch\_ID | We are referencing this field as a foreign key | Foreign Key – As there is 1 to Many relationship & total participation between Company and Family we used foreign key branch id on the many side which is Family table  ON DELETE CASCADE –  we want to remove entire row values from family table if branch gets closed |

***Family Email table –***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Family\_Id | Integer –  Because we want a numeric value to easily identify each family | Foreign Key –  We want to identify which email Id belongs to which parent so we brought Family Id into Family email table.  ON DELETE CASCADE –  we want to remove entire row values from family email table if family gets removed. |
| Email | Varchar (255) –  We used this data type to have a variable length for the email address and we know the actual data will be less than the capacity mentioned. | Unique –  We know that email Id will be unique for each person  Primary Key –  Primary key is the combination of both family\_id and email because each parent can have multiple email addresses but the entry as a whole should not be null and should be unique |

***Family Mobile number table –***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Family\_Id | Integer –  Because we want a numeric value to easily identify each family | Foreign Key –  We want to identify which mobile number belongs to which parent so we brought Family Id into Family mobile number table.  ON DELETE CASCADE –  we want to remove entire row values from family mobile number table if family gets removed. |
| Mobile number | Varchar (10) –  We used this data type to have a variable length upto 10 characters because mobile number should not be more than that. | Unique –  We know that mobile number will be unique for each person  Primary Key –  Primary key is the combination of both family\_id and mobile number because each parent can have multiple mobile numbers but the entry as a whole should not be null and should be unique |

***Child table –***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Family\_Id | Integer –  Because we want a numeric value to easily identify each family | Foreign Key –  We want to identify which child belongs to which parent so we brought Family Id into Child table.  ON DELETE CASCADE –  we want to remove entire row values from child table if family gets removed. |
| Child\_Id | Integer –  Because we want a numeric value to easily identify each child | Primary Key –  Is the combination of family Id and child id because child is a weak entity and family can have multiple children. So, making the combination of those fields we can uniquely identify the child |
| Full Name | Varchar (255) –  We used this data type to have a variable length for the name of the child and we know the actual data will be way less than the capacity mentioned. | Not Null –  Name should be mandatory for the child to interact with them |
| Date of birth | Date –  We want date of birth to be stored in date format |  |
| Gender | Varchar(20) –  We expect gender values to be Male/Female/Others. So, we have mentioned maximum capacity as 20 |  |
| Is\_disable | Varchar(50) –  We expect values to be Yes/ No. we have mentioned maximum capacity as 50 because to allow any comments regarding this. | Not Null –  We wanted to make sure we know about the child’s disability while registering them into the database |
| Monthly\_fee | Decimal(38,2) –  We expect monthly fee to be double value |  |
| Contract Id | Integer –  Because we want a numeric value to easily identify each contract | Foreign Key –  We want to know with which contract is the child registered to. And also because we have one to one relationship with the contract and child.  ON DELETE SET NULL –  If the contract has ended, we don’t want to delete the child entry because there can be new contract with the other baby sitter. So, we set null and will update once a new baby sitter is available with the contract |

***Child languages spoken table:***

|  |  |  |
| --- | --- | --- |
| Field | Data type | Constraint |
| Family\_Id | Integer –  Because we want a numeric value to easily identify each family | Foreign Key –  We want family id to be foreign key to identify child  ON DELETE CASCADE –  we want to remove entire row value if family gets removed. |
| Child\_Id | Integer –  Because we want a numeric value to easily identify each child | Foreign Key –  We want child id to be foreign key to identify which child speaks which language  ON DELETE CASCADE –  We want to remove entire row value if child gets removed |
| Language\_spoken | Varchar(50) –  We should be able to store the language in this data type | Primary Key –  Primary key is the combination of (family\_id, child\_id) and language\_spoken. We made sure that the combination will always be unique. |

***BABY SITTER-***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| SSN | Integer –  Because we want a numeric value to easily identify each baby sitter | Each baby sitter registered in the database should have an unique identifier |
| DOB | Date –  We want date of birth to be stored in date format |  |
| Nationality | Varchar (50)  We expect length of nationality to be not more than 50 characters |  |
| Gender | Varchar(20) –  We expect gender values to be Male/Female/Others. So, we have mentioned maximum capacity as 20 |  |
| Available\_start\_time | Time-  We wanted to store time in this field. So, we used time datatype | Not Null –  We want to know the availability times for baby sitters before entering them into the database |
| Available\_end\_time | Time-  We wanted to store time in this field. So, we used time datatype | Not Null –  We want to know the availability times for baby sitters before entering them into the database |
| Available\_day\_of\_week | Varchar(20) –  We expect day values to be less than 20 characters. So, we used this datatype. | We want to know the availability day of week of baby sitters before entering them into the database |
| other\_occupation | Varchar (255) –  This data type can be used to store this information |  |
| Highest\_education\_degree | Varchar (255) –  This data type can be used to store this information |  |
| Hourly\_pay\_rate | Decimal(38,2) –  We expect hourly pay to be double value |  |
| Branch\_Id | Integer –  Because we want a numeric value to easily identify each branch | Foreign key –  We want to know in which branch does the baby sitter works in.  ON DELETE CASCADE –  If branch is closed we wanted to remove all the babysitters associated to that branch |
| Full\_name | Varchar (255) –  We used this data type to have a variable length for the name of the baby sitter and we know the actual data will be way less than the capacity mentioned. | Not Null –  Name should be mandatory for the baby sitter |

***Baby\_sitter\_language -***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| SSN | Integer –  Because we want a numeric value to easily identify each baby sitter | Foreign Key –  We want to know the information about language spoken by the baby sitter  ON DELETE CASCADE –  We want to remove the babysitter information in this table if baby sitter is dropped from the parent table |
| language\_spoken | VARCHAR(50)- we expect length of each language value wouldn’t be greater than 50 characters | Primary key with combination of SSN because each babysitter can speak multiple languages and the combination of both these fields is unique and shouldn’t be NULL |

***Baby\_sitter\_skills -***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| SSN | Integer –  Because we want a numeric value to easily identify each baby sitter | Foreign Key –  We want to know the information about skills for each babysitter  ON DELETE CASCADE –  We want to remove the babysitter information in this table if baby sitter is dropped from the parent table |
| skill | VARCHAR(255)-  This datatype can hold the skill information for baby sitter | Primary key with combination of SSN because each babysitter can have multiple skills and the combination of both these fields is unique and shouldn’t be NULL if values are entered in this table |

***Contract -***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Contract\_ID | Integer –  Because we want a numeric value to easily identify each contract | Primary Key –  Each contract should have an unique value |
| Start\_date | Date –  We want to store the date when the contract starts |  |
| End\_date | Date –  We want to store the date when the contract ends |  |
| Activity | Varchar(255) –  This capacity should be enough to enter all the activities which were performed as part of the contract |  |
| SSN | Integer –  Because we want a numeric value to easily identify each baby sitter` | Foreign Key –  We want to know the information about which baby sitter got involved in the contract.  ON DELETE SET NULL –  We want to set the babysitter information in this table to null if baby sitter is dropped from the parent table because then we could assign a different babysitter to the contract with the child |

***Contract Services -***

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Constraint |
| Contract Id | Integer –  Because we want a numeric value to easily identify each contract | Foreign Key –  We want to know the information about the contract and its services expected.  ON DELETE CASCADE –  If contract was removed from its parent table there is no need to store its services expected information. So, we drop it |
| Service\_expected | Varchar(255) –  This capacity should be able to hold all the information required to fill the services | Primary Key –  Combination of contract ID and service\_expected. Because, there could be multiple services expected as part of the contract but the combination of both the columns should be unique and not null. |

# Creating Database and Tables - SQL DDL

CREATE\_TABLE\_MYSQL\_SCRIPT

DROP\_TABLE\_MYSQL\_SCRIPT

CREATE\_TABLE\_MSSQL\_SCRIPT

DROP\_TABLE\_MSSQL\_SCRIPT

# Inserting Values in Tables

INSERT\_TABLE\_MYSQL\_SCRIPT

INSERT\_TABLE\_MSSQL\_SCRIPT

# SQL Queries

RETREIVAL\_MYSQL\_SCRIPT

# RETREIVAL\_MSSQL\_SCRIPT