

| Business Template  **SUBJECT AREAS** |
| --- |
| **Logo / Image** |

**Contents**

[1](#_1fob9te) Business Description 3

[1.1](#_3znysh7) Business background 3

[1.2](#_2et92p0) Problems. Current Situation 3

[1.3](#_tyjcwt) The benefits of implementing a database. Project Vision 3

[2](#_3dy6vkm) Model description 3

[2.1](#_1t3h5sf) Definitions & Acronyms 3

[2.2](#_4d34og8) Logical Scheme 3

[2.3](#_2s8eyo1) Objects 3

# 

# Business Description

## Business background

Dudiks Mountaineering Club is an adventure-focused organization dedicated to promoting outdoor exploration, physical fitness, and community bonding through mountaineering activities. Founded in 2024, our club has rapidly grown to become a cornerstone of the wideworld outdoor enthusiast community.

## Problems. Current Situation

Despite our success, we have faced difficulties with recording climbs of our club all over the world, and keeping track of the new members of it. At the moment we’re planning to open new subsidiaries in such countries as Germany, France, China, Japan and the USA. And to be successful we need to create an appropriate database which will help us manage business correctly.

## The Benefits of implementing a database. Project Vision

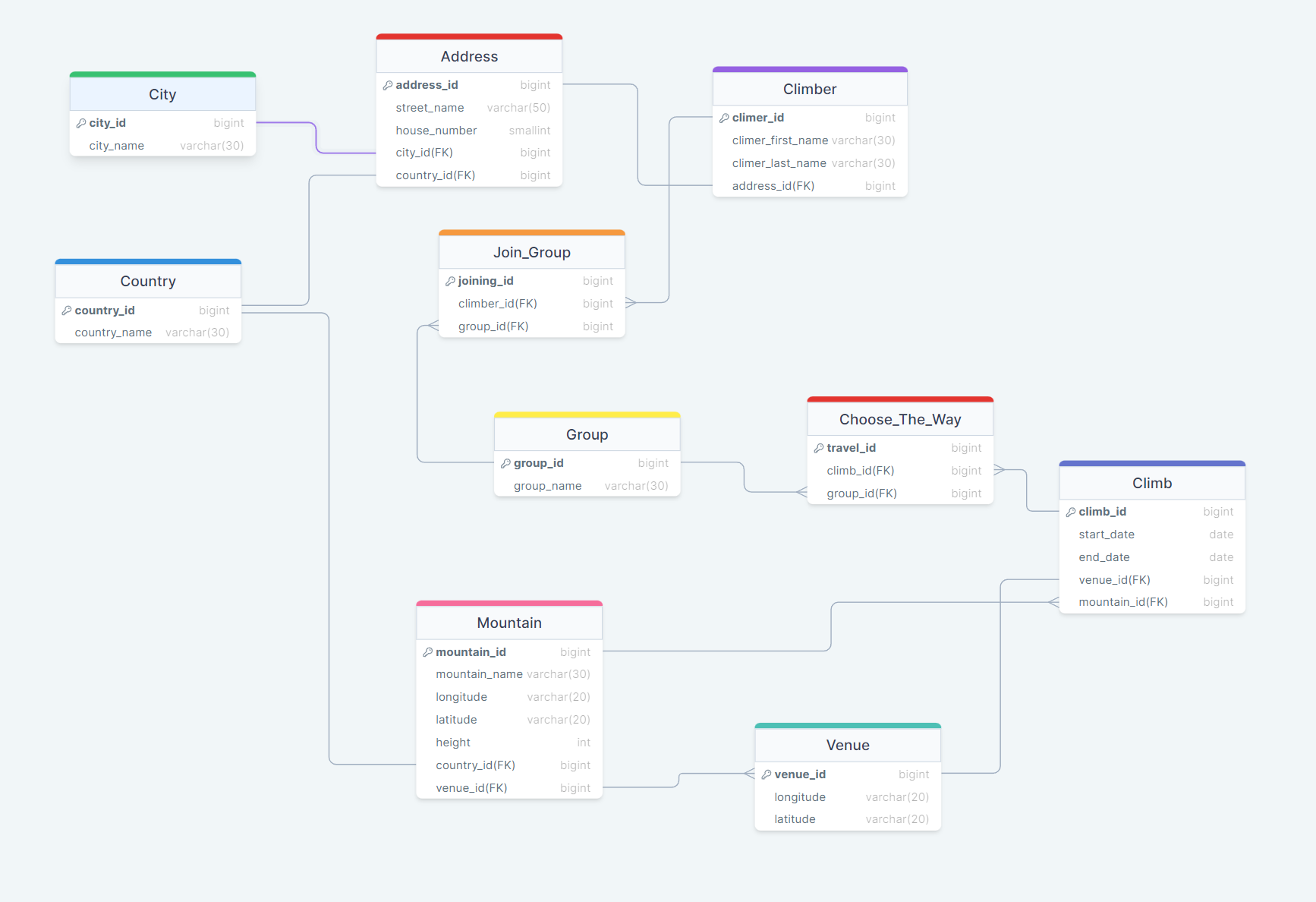
* Centralized Information Management: A database provides a centralized platform to store and manage information related to past, present, and future climbs. This includes details such as climb dates, locations, routes, participants, guides, equipment used, and trip reports.
* Improved Organization and Accessibility: With a database in place, all climb-related information is organized in a structured manner, making it easily accessible to club members, trip organizers, and administrators. This streamlines the process of planning, coordinating, and executing climbs.
* Enhanced Safety and Risk Management: By maintaining comprehensive records of climbs, including participant profiles, emergency contact information, and incident reports, the database facilitates better safety and risk management. Trip organizers can quickly access relevant information to ensure appropriate safety measures are in place for each climb.
* Historical Data Analysis: A database allows for the analysis of historical climb data, enabling the club to identify trends, patterns, and areas for improvement. Insights gained from analyzing past climbs can inform future trip planning, route selection, gear requirements, and safety protocols.
* Member Engagement and Participation: Access to climb data through the database encourages member engagement and participation in club activities. Members can easily find information about upcoming climbs, past experiences shared by fellow climbers, and opportunities to get involved in future expeditions.

# Model description

## Definitions & Acronyms

* PK(Primary Key): A primary key is a special relational database table column(or combination of columns) designated to uniquely identify all table records. A primary key must contain unique values and cannot contain NULL values.
* FK( Foreign key): A foreign key is an attribute or a set of attributes in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a relationship between them.
* DATE: A data type in SQL used for storing dates as a single field.
* SMALLINT: The SMALLINT data type stores small whole numbers that range from –32,767 to 32,767. The maximum negative number, –32,768, is a reserved value and cannot be used. The SMALLINT value is stored as a signed binary integer. Integer columns typically store counts, quantities, and so on.
* BIGINT: The BIGINT data type is an integer value from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. BIGINT is SQL Server's largest integer data type. It uses 8 bytes of storage. BIGINT should be used when values can exceed the range of INT.
* INT: The INT data type is an integer value from -2,147,483,648 to 2,147,483,647. INT uses 4 bytes of storage. INT is the most commonly used integer data type in SQL Server.
* VARCHAR: VARCHAR is a datatype in SQL that holds characters of variable length. This data type stores character strings of up to 255 bytes in a variable-length field. The data can consist of letters, numbers, and symbols.

## Logical Scheme



## Objects

**Table 1**

Describes Climber themself, contains their name, id, and their address

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Climber | climber\_id | id of climber, PK | bigint |
| climber\_first\_name | first name of climber | varchar(30) |
| climber\_last\_name | last name of climber | varchar(30) |
| address\_id | id of the address of the climber, FK | bigint |

Have a one to one relationship with the Address Table, so we can store the address data separately. Also have a “one to many” connection with the Join\_Group table, because we cannot directly connect Group and Climber due to their “many to many” relationship.

Example with data

| climber\_id | climber\_first\_name | climber\_last\_name | address\_id |
| --- | --- | --- | --- |
| 1 | Vlad | Dudik | 1234 |

**Table 2**

Describes Address, contains its street name, house number, id, and its city and country id.

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Address | address\_id | id of address, PK | bigint |
| street\_name | name of the street | varchar(50) |
| house\_number | number of the house | smallint |
| city\_id | id of the city the address belongs to, FK | bigint |
| country\_id | id of the country the address belongs to, FK | bigint |

Have a “one to one” relationship with the Climber Table to provide each climber his address information. The same connections are with the City Table and Country Table, which were created because of the existing fact of the two cities with the same name in different countries.

Example with data

| address\_id | street\_name | house\_number | city\_id | country\_id |
| --- | --- | --- | --- | --- |
| 1234 | Matusevich | 86 | 6 | 10 |

**Table 3**

Describes Сity ​​glossary

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| City | city\_id | id of the city, PK | bigint |
| city\_name | name of the city | varchar(30) |

Have a one to one relationship with the Address Table. Main goal to store cities names and their unofficial names.

Example with data

| city\_id | city\_name |
| --- | --- |
| 6 | Minsk |

**Table 4**

Describes Сountry ​​glossary

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Country | country\_id | id of the country, PK | bigint |
| country\_name | name of the country | varchar(30) |

Have a one to one relationship with the Address Table, and also have 1 to 1 connection with Mountain via City\_id, ensuring that each mountain is located in a particular country.

Example with data

| country\_id | country\_name |
| --- | --- |
| 10 | Belarus |

**Table 5**

Describes Mountain, contains its id, name, longitude, latitude, height, and its country id and venue id.

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Mountain | mountain\_id | id of mountain, PK | bigint |
| mountain\_name | name of the mountain | varchar(30) |
| longitude | measures the distance north or south of the equator of the mountain. | varchar(20) |
| latitude | measures distance east or west of the prime meridian of the mountain. | varchar(20) |
| height | height of the mountain in meters | int |
| country\_id(FK) | id of the country the mountain takes place, FK | bigint |
| venue\_id(FK) | id of the venue, FK | bigint |

Have a “one to one” relationship with the Country Table to ensure that it’s located in a particular one. Also have a “one to many” connection with the Climb Table, this is done to record each climb for a particular mountain, the design helps avoid data redundancy and maintain its integrity. And in addition to that, Mountain have a “one to many” relationship with the Venue Table, to ensure that there are many starting/gathering points of the climb and not only one.

Example with data

| mountain\_id | mountain\_name | longitude | latitude | height | country\_id | venue\_id |
| --- | --- | --- | --- | --- | --- | --- |
| 192 | Mont Blanc | “45.8326° N” | “6.8652° E” | 4,809 | 18 | 17 |

**Table 6**

Describes Venue themself, contains their id, longitude, latitude,

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Venue | venue\_id | id of climber, PK | bigint |
| longitude | measures the distance north or south of the equator of the venue. | varchar(20) |
| latitude | measures distance east or west of the prime meridian of the venue. | varchar(20) |

Have a many to one connection with Mountain, storing all possible gathering points for each mountain.

Example with data

| venue\_id | longitude | latitude |
| --- | --- | --- |
| 17 | “45.9237° N” | “6.8694° E” |

**Table 7**

Describes Joining to the Group, contains its id, climber id the id of the group they want to join.

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Join\_Group | joining\_id | id of climber, PK | bigint |
| climber\_id | id of the climber who wants to join the group, FK | bigint |
| group\_id | id of the group climber wants to join, FK | bigint |

Have a “many to one” relationship with tables Climber and Group, serving as a buffer table to make a many to many connection between Group and Climber.

Example with data

| joining\_id | climber\_id | group\_id |
| --- | --- | --- |
| 161 | 1 | 13 |

**Table 8**

Describes Group themself, contains their name, id, and their address.

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Group | group\_id | id of climber, PK | bigint |
| group\_name | name of the group | varchar(30) |

Have a “one to many” relationship with the Join\_Group Table in order to connect climbers with groups. Also have “one to many” connection with the Choose\_The\_Way Table to ensure “many to many” connection with Climb Table and store the information about groups who climbed a particular way.

Example with data

| group\_id | group\_name |
| --- | --- |
| 13 | “Indiana Jones” |

**Table 9**

Describes which Mountain the group will climb, contains travel id, group id, and climb id.

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Choose\_The\_Way | travel\_id | id of chosen way | bigint |
| climb\_id | id of the climb record, FK | bigint |
| group\_id | id of the group who will climb, FK | bigint |

Buffer table to ensure “many to many” connection between Group and Climb tables. Have “many to one” with tables Group and Climb

Example with data

| travel\_id | climb\_id | group\_id |
| --- | --- | --- |
| 17 | 10 | 13 |
|  |  |  |

**Table 10**

Describes a record of the climb, contains climb id, start and end date, venue id, and mountain id.

| Table Name | Field name | Field Description | Data Type |
| --- | --- | --- | --- |
| Climb | climb\_id | id of chosen way | bigint |
| start\_date | id of the climb record, FK | date |
| end\_date | id of the group who will climb, FK | date |
| venue\_id | id of the venue the climb will start from, FK | bigint |
| mountain\_id | id of the mountain the climb will take place on, FK | bigint |

Have a “one to many” connection to extend information about which groups take part in climbing. Also have one to one connection with the Venue table, because each climb starts only at one place. And have a many to one connection with mountain due to the existence of several climbs to one mountain.

Example with data

| climb\_id | start\_date | end\_date | venue\_id | mountain\_id |
| --- | --- | --- | --- | --- |
| 1080 | 18.04.2024 | 21.04.2024 | 17 | 192 |