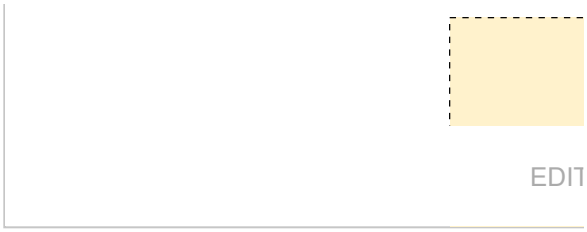
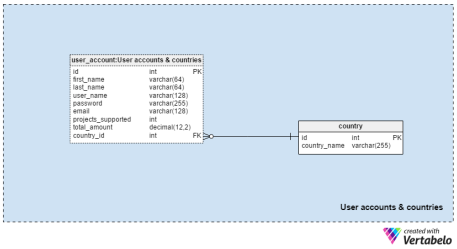


The data model consists of four main subject areas:

- `User accounts & countries`
- `Projects`
- `Project teams`
- `Investors`

We'll describe each subject area in the same order they are listed.

Section 1: User Accounts and Countries



The `User accounts & countries` section consists of only two tables, the **`country`** table and the **`user_account`** table. User account information appears in two other subject areas, so I'm describing this subject area first.

The **`country`** table is a dictionary containing a list of UNIQUE `country_name` values. We'll use these values only to relate users with countries.

The **`user_account`** table stores details about each user that creates an account on our platform. In this model, only investors must have accounts; project participants could have an account but it is not mandatory. For each user account, we'll store the following values:

- `first_name` and `last_name` – The first and the last name of the user.
- `user_name` – A UNIQUE username value.
- `password` – A password hash set by the user.

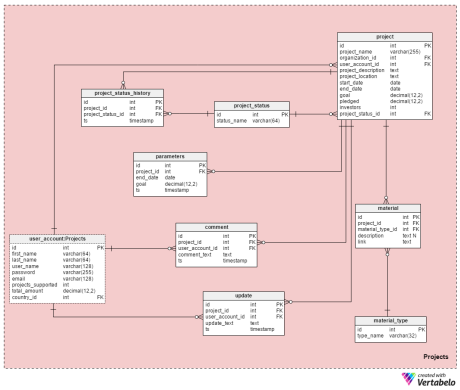
`email` – The email account used to sign in to our application.

`projects_supported` – The number of projects that user has supported so far. This data could also be calculated from other tables in the model, but I find it appropriate to store a redundant value here to increase performance. It's important that we update this attribute every time the user invests in some new project.

`total_amount` – The total amount the user has invested in all projects so far.

`country_id` – References the country where the user lives.

Section 2: Projects



The `Projects` subject area contains the most important tables in our model. Without projects, there is nothing for investors to be interested in.

A list of every project ever posted on our platform is stored in the **`project`** table. The attributes in this table are:

`project_name` – The project's name, as selected by the project initiator. This could be UNIQUE but I've chosen not to set it so. There is always the chance we will have two projects with the same name.

`organization_id` –References the **`organization`** table and denotes the organization that is behind this project.

`user_account_id` – References the account of the user who created this project.

`project_description` – A detailed description of the project including a summary of the idea, goals, the project's plan and roadmap, etc.

`project_location` – The physical location where either 1) the project will take place, or 2) the location of the organization, e.g. New York, USA.

`start_date` – The start date of the project or campaign.

`end_date` – The expected end date for the project or campaign. This attribute will hold only the current value. We might change end date of the project (e.g. in case we want to extend the campaign's duration). A history of all end dates for all projects is stored in the **`parameters`** table.

`goal` – The current financial goal of the project. This is the minimum amount of money we need to run the project successfully. Like `end_date`, this could change; we'll store its history in the **`parameters`** table.

`pledged` – The amount of funds currently pledged to the project.

`investors` – The current number of investors for this project.

`project_status_id` – References the **`project_status`** dictionary and holds the current status for this project.

The value that is usually displayed next to the project is the funded percentage. We won't store this value in our database, but we will calculate it on the screen using the formula *"pledged" / "goal"*.

Besides a basic description of the project, we'll need to store related materials like videos, written content, charts, infographics etc. We'll do this using two tables in our model: the **`material_type`** table and the **`material`** table.

The **`material_type`** dictionary contains only UNIQUE `type_name` values. These names will distinguish various types of materials and select how best to display them on our portal.

All materials related with all projects are stored in the **`material`** table. For each material, we'll store the ID of the related project, the material's type and description (if any), and a link to the location where that material is stored.

In the **`project`** table, we'll only store the ID of the current status of the project.

We have already mentioned project statuses. A list of all possible statuses is stored in the **`project_status`** dictionary. This table contains a list of UNIQUE `status_name` values. We can expect statuses like *"draft"*, *"campaign created"* (i.e. the draft has been finalized but the campaign has not started yet), *"campaign started"* (assigned automatically when we reach the `project`.`start_date`), *"campaign ended – successfully"*, *"campaign ended – unsuccessfully"*, *"campaign ended – cancelled"* and *"project started"*. Additional statuses could be added in order to describe the project's progress after the campaign is completed. For example, a business may want to track its delivery of services or goods to all investors. If the project is unsuccessful, we should be able to ensure donations are refunded, etc.

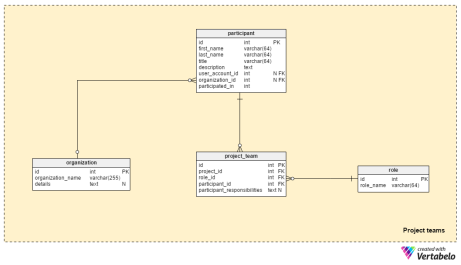
All statuses that were ever assigned to any project are stored in the **`project_status_history`** table. Each time when we insert or change the value of the `project`.`project_status_id` attribute, we'll insert a record into this table. It stores the ID of the related project, the ID of the related status, and the timestamp when that status was assigned.

Every crowdfunding campaign has many parameters that might change along the way. We'll start with just two of them, the `end_date` of the campaign and its `goal`. These values could change according to the current campaign status – if we have reached our planned `goal`, we could set a new goal or extend the campaign's duration. When we make these changes, we should be able to track the history of all such changes. That is what the **`parameters`** table is for. When we add or change the `end_date` or the `goal` in the **`project`** table, we'll also insert a new row into the **`parameters`** table. This will store the ID of the related project, the previous `end_date` and `goal` values, and the timestamp the change happened.

The remaining two tables in this subject area, **`comment`** and **`update`**, relate to comments. Both have almost the same structure. They store the related project and user IDs, the actual `comment_text` or `update_text`, and the timestamp when that comment was inserted. The main reason to have two tables in the model is to separate user/investor comments from project initiator comments. The **`comment`** table holds

only unofficial comments, like investor's questions and remarks. Project initiators will post their comments (i.e. project updates) and these comments are stored in the `update` table. Note that project initiators who are also registered users could also write "regular" comments.

Section 3: Project Teams



This subject area also contains the tables that describe the project. All of these tables are closely related with project participants and teams.

Behind every project is an organization. That will usually be an institution, a company, or a group of people, but it could also be an individual. We'll store all such organizations in the `organization` table. The only mandatory attribute is the `organization_name` , while the `details` attribute can be used to describe organization in detail.

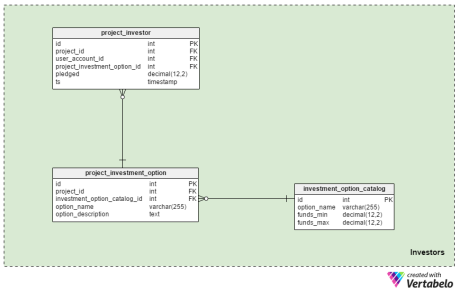
A list of all project participants is stored in the `participant` table. For each participant, we'll record a:

- `first_name` and `last_name` – The first and the last name of the participant.
- `title` – The participant's title.
- `description` – A text description of participant's experience and roles so far.
- `user_account_id` – The ID of the related user account, if the participant is also a user of the crowdfunding platform. In that case, the values stored in the `first_name` and `last_name` attributes are the same in both tables. (Remember, a user account is not mandatory for participants.)
- `organization_id` – The ID of the related organization, if any.
- `participated_in` – The number of projects this user has participated in.

The `role` dictionary contains a list of each UNIQUE `role_name` that could be assigned to team members.

Knowledge and experience in running crowdfunding campaigns can be crucial to making a project happen, so it's important to have a strong team. The `project_team` table stores all team members related with all of our projects. For each team member, we'll store IDs for the related project, role, and participant. We'll also store a detailed description of their project responsibilities.

Section 4: Investors



The last subject area deals with more project stakeholders and their actions. In this case, though, the stakeholders are the `Investors` – the registered app users who will fund or donate to projects.

We can expect that each project will have some default investment options. Usually, users choose between a few predefined amounts, e.g. 10 USD, 25 EUR, etc. Each amount will have a different benefit for the project investor. For example, if you were to invest in a game development project, you'd be able to download a copy of the game for $10; for $25, you'd get the game plus some additional power ups; for $50 the project initiators would mention you as a contributor, and for $1,000 you'd get to name a character or object in the game.

All predefined investment options are stored in the `investement_option_catalog` . The `option_name` value can hold only UNIQUE values while the `funds_min` and the `funds_max` define the range of funds that could be invested. When the initiators set all donation amounts in advance, these attributes will share the same values.

The `project_investement_option` stores the predefined investment options chosen by the

initiators. For each record in this table, we'll store IDs for the related project and the related investment option. The `option_name` is the name that will appear on the screen. If the project initiator wants, they could keep the same option name as the one stored in the `investment_option_catalog`.`option_name` attribute. A detailed description of that option is stored in the `option_description` attribute. This is the place where we'll describe investor benefits for the option.

The last table in our model is the **`project_investor`** table. It relates investors and projects. For each investment, we'll store the ID of the related project and the related investor as well the chosen investment option. We'll additionally store the amount of funds `pledged`. This is important, especially in cases when the investor can set their own donation from a predefined range of values. And finally we come to the timestamp attribute, which denotes when an investment happened.

Today we have discussed a model that could be used to store the data for a crowdfunding platform. I have described the essentials, but there is a lot we could add to this model. For example, what if a campaign is not successful? How can users track a project after the campaign is completed?

I look forward to seeing your ideas in the comment section.