|  |
| --- |
| Business Template  **recruitment agency** |
| **Nabijonov Kamoliddin 22.11.2021** |

Contents

[1 Business Description 3](#_Toc62212630)

[1.1 Business background 3](#_Toc62212631)

[1.2 Problems. Current Situation 3](#_Toc62212632)

[1.3 Benefits from implementing a database. Project Vision 3](#_Toc62212633)

[2 Model description 3](#_Toc62212634)

[2.1 Definitions & Acronyms 3](#_Toc62212635)

[2.2 Logical Scheme 3](#_Toc62212636)

[2.3 Objects 3](#_Toc62212637)

# 

# **Business Description**

## **Business background**

We are required to design a new system for an imaginary recruitment agency. Such agencies work with a great number of candidates and organizations. It is hard for them to track the record of each candidate and his progress, analyze experience, and derive new conclusions to improve the business process. That brings many problems both for candidates, organizations, and agencies themselves. We need to design a proper database that will handle all issues and expected load.

Before designing our data model, it is essential to understand the business process for that system. First, let’s see the steps involved in the hiring process:

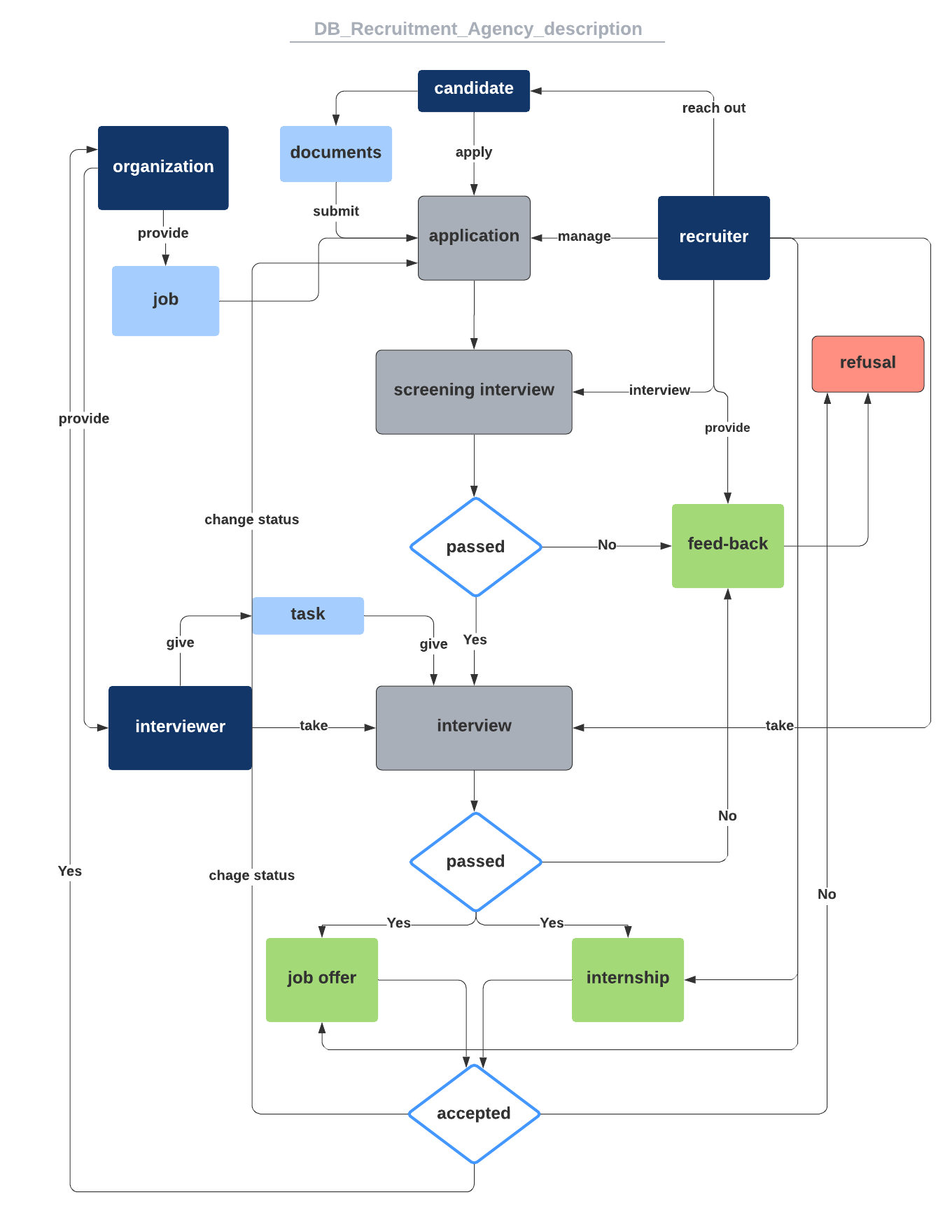
1. Company contact recruitment agencies and pass job lists that must be filled.
2. Agency assigns vacancies to employees(recruiters) who then start the recruitment process. This process usually takes multiple steps, such as screening interview, technical interview, behavioral interview, test tasks, tests, and so on until final decision and a job offer.
3. Candidates apply for the job.
4. Applicants are chosen from the list and invited to the first interview.
5. If successful, further interviews are scheduled. Often, client organizations provide specialists and tasks for this stage. If the candidates fail, they are provided with feedback and are denied.
6. Finale, results are calculated and each candidate is given a special score.
7. Applicants, further are evaluated based on this score and results from tests and interviews.
8. The hiring decision is made. The candidate will be provided with a job offer of denial.

## **Problems.** **Current Situation**

* Data loss due to the absence of a management system
* Slow progress
* Manual tracking
* Absence of a base for future analytics and metrics
* High inaccuracy in the process

## **Benefits from implementing a database. Project Vision**

After successful integration, all the above-mentioned problems must be solved. As a result, all candidates, agencies, and organizations will benefit from such an autonomous system administered by HR experts. The delay between stages will be minimized and profits maximized due to the increased processing power. Data will remain secure and consistent and ready for future use. In addition, candidates and will be able to monitor their progress and results.



# **Model description**

## **Definitions & Acronyms**

**Candidate** – a potential employee of an organization under consideration.

**Document** – CV, Recommendation letters, Research papers, Cover letter.

**Skill** – a skill the candidates possess, like android, HTML, CSS, SQL, Python.

**Education** – institution and degree level of each candidate.

**Recruiter** – an employee of the recruiting agency who finds qualified candidates for a job opening and works to meet the demands of both the employer and the employee throughout the hiring process.

**Organization** – client company of the agency that is looking for employees and provides job positions.

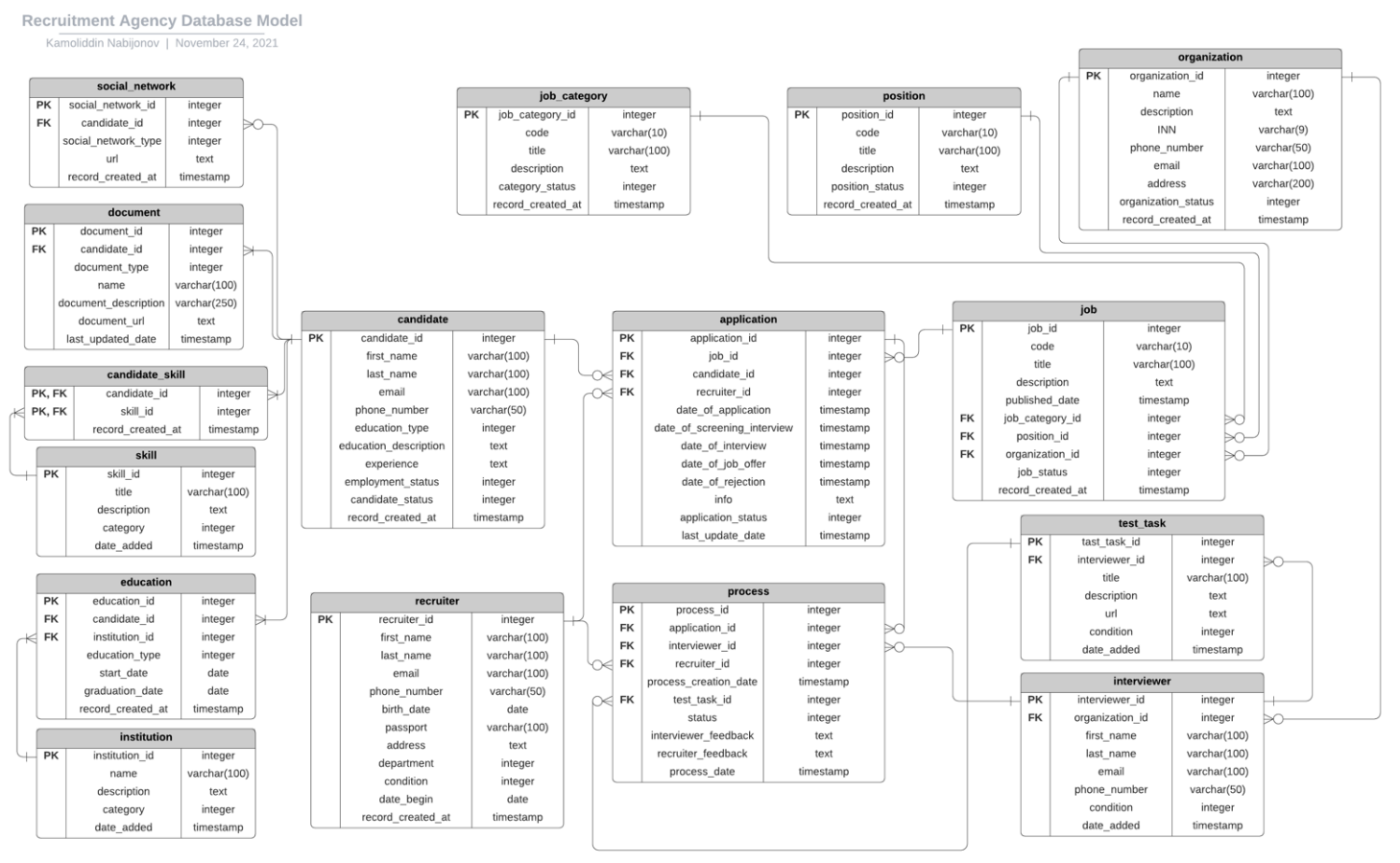
**Interviewer** – an employee of a client organization that will, if necessary, take a specific job-related interview from candidates.

**Task** – a test given by the interviewer and that must be completed by the candidate.

**Application**- the process of matching candidates and jobs provided. Managed by the recruiter.

**Process** – in our case, processes like an interview, considerations, scoring, testing candidates.

## **Logical Scheme**



## **Objects**

**candidate** - table to store data of all candidates of the agency.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| candidate | candidate\_id | ID of a candidate, autoincrement, PK | integer |
| first\_name | First name of a client | varchar(100) |
|  | last\_name | Last name of a client | varchar(100) |
| email | Email of a client | varchar(100) |
| phone\_number | Phone number of a client | varchar(50) |
| education\_type | Education level.  For example: 1 – high school, 2 – BS, 3 – MS | integer |
| education\_description | Textual description of a degree | text |
| experience | Textual description of past and current experience | text |
| employment\_status | Current Employment status  For example:  0 – unemployed 1 – intern 2 – employed | integer |
| candidate\_status | Currrent candidate status  For example: 0 - not reviewed 1 - under review 2 – denied | integer |
| record\_created\_at | Date and Time when record was created | datetime |

Candidate entity is split into 5 tables. First, is the candidate table described above. Nearly all candidates have social networks such as Facebook, LinkedIn, and so on. So I created a separate table (social\_network) to store this data with a **1:M** relationship.

Also, all candidates must have at least one document. For example, a CV, attached to them while creating the profile. For this, we have a table **document** that is related to the candidate table with a **1:M** relationship.

It is easy for recruiter to find candidates with some specific skills. For this purpose, we have separate table **skill** where we store all possible skills a candidate might have and **candidate\_skill** table to match skills and candidates. One specific skill might belons to many candidates and one candidate might have many different skills.

The same with education. One candidate might have several educations. We have **institution** table to store data about collage, universities, schools and etc. and table **education** to match candidates to the institution they have finished with a special column **education\_type** indicating education level.

**social\_network** - table to store the information about all social networks that clients might have. Candidate may or may not indicate his or her social network accounts. Candidate may have many accounts on different platforms. Column social\_network\_type indicates which platform (GitHub, Facebook and etc).

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| social\_network | social\_network\_id | ID of a social network, autoincrement, PK | integer |
| candidate\_id | ID of a candidate the network belongs to, FK | integer |
|  | social\_network\_type | Type of social network  For example: 1 – GitHub 2 – Facebook | integer |
| url | Link to candidate’s home page | text |
| record\_created\_at | Date and time when record was added into table | datetime |

Related to **candidate** table with a **1:M** cardinality. Table stores all links for each candidate. References **candidate** table using **candidate\_id** columns.

**document** - table to store the document data that the candidate might have. Each candidate must at least have one document like CV, Research paper or Cover letter. This table is used to store the records that indicate the documents the candidate has. Document is bind to candidate using **1:M** relationship.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| document | document\_id | ID of a document, autoincrement, PK | integer |
| candidate\_id | ID of a candidate the document belongs to, FK | integer |
| name | The name of the document | varchar(100) |
|  | document\_description | Description of the document | varchar(250) |
| document\_url | URL address of the document location (For example google drive link) | text |
| last\_updated\_date | Date and Time when the record was last updated | datetime |

Related to **candidate** table with a **1:M** cardinality referencing **candidate\_id**. One candidate might have many documents and each document belongs to only 1 candidate.

**skill** - table to store all skills that the candidate might have. Each candidate must at least have one skill like SQL, Python and etc. This table is used to store the records that indicate the skills the candidate has. Skill is bind to **candidate** using **M:M** relationship and **candidate\_skill** as linking table.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| skill | skill\_id | ID of a skill, autoincrement, PK | integer |
| title | Title of the skill (Python, Accounting) | varchar(100) |
| description | Textual, detailed description of the skill. Includes requirements and the definition. | text |
|  | category | Category to which the skill belongs to (IT, Finance) | integer |
| date\_added | Date and Time when the record was added into the table | datetime |

Related to **candidate** table with a **M:M** cardinality using **candidate\_skill** linking table. One candidate might have many skills and each skill belongs to many candidates.

**candidate\_skill** - table used to link the **skills** to **candidates**. Acts as a linking table in **M:M** relationship and stores both **candidate\_id** and **skill\_id**.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| document | candidate\_id | ID of a candidate the skill belongs to, autoincrement, PK, FK | integer |
| skill\_id | ID of a skill the candidate has, PK, FK | integer |
| record\_created\_at | Date and Time when the record was added into the table | datetime |

Related to **candidate** and **skill** tables with the **1:M** cardinality referencing **candidate\_id** and **skill\_id**. One candidate might have many skills and each skill belongs to only many candidates.

**institution** - table to store all education institutions that the candidate might studied at. Each candidate must at least have school education. For this reason, cardinality is at least 1 or Many. **Institution** is bind to **candidate** using **M:M** relationship and **education** as linking table.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| institution | institution\_id | ID of a institution, autoincrement, PK | integer |
| name | The name of a institution | varchar(100) |
| description | Textual, detailed description of the institution | text |
|  | category | Category to which the institution belongs to (School, University, Collage) | integer |
| date\_added | Date and Time when the record was added into the table | datetime |

Related to **candidate** table with a **M:M** cardinality using **education** linking table.

**education** - table used to store education of each candidate. Each candidate has at least finished one institution and each institution may have many graduates. Acts as a linking table between **candidate** and **institution**. Stores important dates such as graduation and start dates. Also indicates (**education\_type**) what degree the candidate has from each institution he or she has attended (BS, MS, PhD).

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| education | education\_id | ID of a education record, autoincrement, PK | integer |
| candidate\_id | ID of a candidate, FK | integer |
| institution\_id | ID of a institution the candidate studied at, FK | integer |
|  | education\_type | Degree the candidate acquired (PhD, MS, MBA, BS) | integer |
| start\_date | Date when the candidate started particular institution | date |
| graduation\_date | Date when the candidate finished particular institution | date |
| record\_created\_at | Date and time when record was added into the table | datetime |

Table references **candidate** table with **1:M** relationship using **candidate\_id** column.

Table references **institution** table with **1:M** relationship using **institution\_id** column.

**recruiter** - table to store data of all recruiters of the agency. Because recruiters are actual employees of the agency, we need to store personal details like passport, address and so on. Recruiters control nearly all processes with hiring process. They are references in both transaction tables **application** and **process**.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| recruiter | recruiter\_id | ID of a recruiter, autoincrement, PK | integer |
| first\_name | First name of a recruiter | varchar(100) |
|  | last\_name | Last name of a recruiter | varchar(100) |
| email | Email of a recruiter | varchar(100) |
| phone\_number | Phone number of a recruiter | varchar(50) |
| birth\_date | Birth date of a recruiter | date |
| passport | Passport data | varchar(100) |
| address | Textual description of address where employee lives | text |
| department | Current working department  For example:  0 – IT 1 – Finance 2 – Security | integer |
| condition | Current employee status  For example: 0 – active 1 – inactive | integer |
| date\_begin | Date when the recruiter joined the agency | date |
| record\_created\_at | Date and Time when record was created | datetime |

One recruiter might have many applications under control and might control many processes like interview, offer, screening. For this reason this table is connected with **1:M** cardinality with application table and **1:M** cardinality to process table too.

**organization** - table to store data of all client organizations of the agency. Often organizations require candidates to go through domain-related interviews to check their expertise. It is hard for recruitment agencies to manage each field separately. So organizations provide some of their employees to take such interviews and in addition they also provide test tasks for candidates to solve. That is why we split this entity into to others like **test\_task** and **interviewer** with **1:M** relationship accordingly.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| organization | organization\_id | ID of a organization, autoincrement, PK | integer |
| name | The name of a organization | varchar(100) |
|  | description | Textual description of a organization | text |
| INN | INN of a organization | varchar(9) |
| phone\_number | Working contact number of a organization | varchar(50) |
| email | Email of a organization | varchar(100) |
| address | Short textual description of address where the organization is located | varchar(200) |
| organization\_status | Current status of organization  For example:  1 – active 2 – inactive 3 – currently not hiring 4 – hiring | integer |
| record\_created\_at | Date and Time when record was created | datetime |

Organization table connected with **test\_task** and **interviewer** tables with 1:M cardinality. One organization may provide more than one task and employee for the process. Each task and interviewer belongs to one organization.

**Interviewer** - table to store data of all employees provided by organization for interview process. Each interviewer manages many processes. Domain specific interviews for each candidate if required. For this reason, this table is references by process table to find out responsible employee for each going process.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| interviewer | interviewer\_id | ID of a interviewer, autoincrement, PK | integer |
| organization\_id | ID of a organization the interviewer works for, FK | integer |
|  | first\_name | First name of a interviewer | varchar(100) |
| last\_name | Last name of a interviewer | varchar(100) |
| email | Email of a interviewer | varchar(100) |
| phone\_number | Phone number of a interviewer | varchar(50) |
| condition | Current condition of a interviewer  For example: 0 – active 1 – inactive | integer |
| date\_added | Date and Time when a record was added into the table | datetime |

Table references **organization** table with **M:1** cardinality and referenced by **process** table with **1:M** relationship.

**test\_task** - table to store data of all tasks provided by interviewer for the hiring process. Each task might be used in many processes (task might be assigned to many candidates) if domain specific interviews for each candidate is required. For this reason, this table is references by process table to find out what task each candidate was assigned.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| test\_task | tast\_task\_id | ID of a task, PK | integer |
| Interviewer\_id | ID of a interviewer the task belongs to, FK | integer |
|  | title | Name of the task (document) | varchar(100) |
| description | Textual description and notes of a task | text |
| url | Link to the document (Google drive, Local host and etc) | text |
| condition | Current status of the task  For example:  1 – active 2 – inactive | integer |
| date\_added | Date and time a task was added into the table | datetime |

Table references **organization** table with **M:1** cardinality and referenced by **process** table with **1:M** relationship.

**job** - table to store data of all available jobs. Each job belongs to organization and each organization may have many jobs. For this reason, table is connected to **organization** table with **1:M** relationship. Table also indicates category the job belongs to and the position it includes. As a result, table connected to two others: **position** and **job\_category**.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| job | job\_id | ID of a job, autoincrement, PK | integer |
| code | Special code that identifies the job  For example:  SEW2021S3P - Software Engineer WEB 2021 Senior 3 Part-time | varchar(10) |
|  | title | The title of the job  Example: Software engineer internal tooling | varchar(100) |
| description | Textual description of a job indicating all requirements, mode, salary and etc | text |
| published\_date | The date when the job was posted | datetime |
| job\_category\_id | ID of category the job belongs to, FK | integer |
| position\_id | ID of indicated position, FK | integer |
| organization\_id | ID of organization the job was posted from, FK | integer |
| job\_status | Current status of a job  For example: 1 – active 2 – inactive | integer |
| record\_created\_at | Date and Time when the job was posted | datetime |

Table connected references **job\_category** and **position** tables with **1:M** cardinality. Referenced in application process to link the **candidate** to **job** with **M:M** relationship using linking table **application**.

**job\_category** - table to store data of all possible categories, jobs might belong to. For example: full-time, contract-based, remote and etc. Used in jobs table to indicate the category.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| job\_category | job\_category\_id | ID of a category, autoincrement, PK | integer |
| code | Special code that describes the category  Example: FSCH2021 –  Finance Supply Chain 2021 | varchar(10) |
|  | title | Title of a category | varchar(100) |
| description | Textual description of a category | text |
| category\_status | Current status of category  For example: 1 – active 2 – inactive | integer |
| record\_created\_at | Date and Time when the category was added into the table | datetime |

Table referenced by **job** table using **job\_category\_id** as foreign key and **1:M** cardinality relationship

**position** - table to store data of all positions to which jobs may belong to. For example: Senior Developer, Android developer, Cloud architect and so on. Referenced by the **job** table with **1:M** relationship.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| position | position\_id | ID of a position, autoincrement, PK | integer |
| code | Special code that describes the position | varchar(10) |
|  | title | Title of a position | varchar(100) |
| description | Textual description of a position | text |
| position\_status | Current status of a position  For example: 1 – active 2 – inactive | integer |
| record\_created\_at | Date and Time when the position was added into the table | datetime |

**application** - one of main tables in database that stores all applications by linking jobs to candidates. Managed by recruiters in all stages. Table contains all information of application from status to important dates when each process finished. Connected to job, candidate, recruiter tables and acts as linking-table for many-to-many relationship. All necessary data in stored in this table. Table has status column to indicate current status of applicant.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| application | application\_id | ID of an applicant, autoincrement, PK | integer |
| job\_id | ID of the job the candidate applied for, FK | integer |
|  | candidate\_id | ID of a candidate, FK | integer |
| recruiter\_id | ID of recruiter that controls the process, FK | integer |
| date\_of\_application | The date when candidate applied for given job | datetime |
| date\_of\_screening\_interview | Date and Time when screening interview was passed | datetime |
| date\_of\_interview | Date and Time when special interview was passed with the organization representative | datetime |
| date\_of\_job\_offer | Date and Time when the candidate was provided with a job offer | datetime |
| date\_of\_rejection | Date and Time when the candidate got rejected | datetime |
| info | Additional notes from recruiter that belongs to application and provides reason for given status | text |
| application\_status | Status of application  For example: 1 – under consideration 2 – rejected 3 – job offer 4 – candidate was hired 5 – application was canceled | integer |
| last\_update\_date | Date and Time when status of the record (application) was updated | datetime |

Table references **job** table with **1:M** relationship using **job\_id** column.

Table references **candidate** table with **1:M** relationship using **candidate\_id** column.

Table references **recruiter** table with **1:M** relationship using **recruiter\_id** column.

**process** - each application has different stages starting from screening to interview and final offer. To manage this, **process** table was created to store all this data and manage processes. Managed by both recruiter and interviewer. All interviews, screening talks, test tasks are stored in this table. Acts as linking table between application, recruiter, task and interviewer. One application has many processes and each process linked to individual application.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| process | process\_id | ID of the process, autoincrement, PK | integer |
| application\_id | ID of the application that current process belongs to, FK | integer |
|  | interviewer\_id | ID of the interviewer that is responsible for current stage, FK | integer |
| recruiter\_id | ID of the recruiter that the current process managed by, FK | integer |
| process\_creation\_date | Date and Time when the process was activated | datetime |
| test\_task\_id | ID of the task that is assigned in the process, FK | integer |
| status | Current status of the process  For example:  1 – active 2 – finished 3 – canceled 4 – scheduled 5 – rescheduled | integer |
| interviewer\_feedback | Feed-back related to current process provided by the interviewer. | text |
| recruiter\_feedback | Feed-back related to current process provided by the recruiter | text |
| process\_date | Date and Time the process is scheduled for | datetime |

Table references **application** table with **1:M** relationship using **application\_id** column.

Table references **interviewer** table with **1:M** relationship using **interviewer\_id** column.

Table references **recruiter** table with **1:M** relationship using **recruiter\_id** column.

Table references **test\_task** table with **1:M** relationship using **test\_task\_id** column.