April Polubiec
COMP2024
MySQL DB assignment
29/11/2023

Summary

This database represents vacant positions within departments of a company which require certain skills as well as candidates who have interviewed for them. It consists of 8 tables: Candidates, Departments, Positions, Interviews, Skills, Offers, CandidateSkills and PositionSkills, two of which are association tables: CandidateSkills and PositionSkills. Positions are available in various departments and each require certain skills. Interviews represent an interview with a candidate for a position and if the candidate receives an offer for a position, that will be stored in the Offers table.

Assumptions, decisions and additions

The table below describes the decisions made in designing each table with regards to data types/columns, unique/primary keys, foreign keys and policies. This includes any assumptions about the usage of this database.

Table	Typing & Columns	Unique / Primary key	Foreign Keys	Policies
Candidates	- Only first name and last name are required to be not null. This allows users to insert a candidate into the database even if they are awaiting contact details. - Phone number is stored as a *varchar(45)* instead of a numeric. Storing the number as a numeric could lead to unintended effects on the data when moving it around to different interfaces or softwares (eg: leading 0s being dropped, programming language converting to a decimal float, etc). In addition, using a varchar reduces complexity in handling the various formats a number could be in (eg: entered in free-text form, scraped from sources on the web, various country codes). We will assume that the phone numbers in this database will be read by a human who can interpret the number appropriately. If any automation does need to be run on these numbers in the future requiring a consistent format, we will have to implement the necessary cleaning and formatting on the backend. If we expect a high volume of international candidates, we can add a country code column but for the timebeing we will assume a mostly homogeneous country of residence.	- Unique key on (first_name, surname, phone, street_address_1, street_address_2, phone). This is to allow flexibility of duplication in candidates with the same name, same phone number (what if two candidates use the same household phone number, siblings, spouses, etc?), same addresses (what if we are in a small town and a candidate moves into an apartment that another candidate once lived in? What if roommates apply?). However, no two candidates can have the exact same name, address, and phone number - it is technically possible in a large enough database but highly unlikely. It is more likely that a user erroneously enters duplicate candidate data than a duplicate existing in reality, so we will add the constraint.	N/a	N/a

Table	Typing & Columns	Unique / Primary key	Foreign Keys	Policies
Departments	- Similar decision as above in regards to phone number typing	- Unique key on (name, address). This allows us to have multiple addresses for the same department (for example, we can have an HR department in a Dublin branch as well as a Galway branch) - Phone number is not unique - departments may share a phone line (for example, IT and Engineering department might both share a general “Tech” phone number).	N/a	N/a
Positions		- Unique key on (position_id, department_id). Position names only need to be unique within a department - the same position can exist in multiple departments (for example, Office Manager in the Legal Department and in the Finance Department). Even if there are multiple vacancies for a position, it should only have one entry into the database - if there were an entry for each vacancy, how would we know which entry to associate with an interview or an offer? The column “openings” indicates how many vacancies there are for the position.	- Departments, 1:n; one department can have many positions but one position can only belong to one department.	- **DELETE policy is set NULL for FK to Departments** - if a department gets deleted, we still want to maintain a record of the positions that have existed and been interviewed for. - **UPDATE policy is set to CASCADE for FK to Departments** - we want the position to have the most up-to-date information on the Department.
Skills	- We use a unique ID column in addition to “Name”. By referencing skills by their ID, we can easily change skill names without worrying about propagating the changes. One user can change the name of the skill and another user can still query it by ID without knowing what the new name is.	- Skill name is unique - ID is unique	N/a	N/a

Table	Typing & Columns	Unique / Primary key	Foreign Keys	Policies
Interviews	- Storing interview_date as a datetime so we can keep track of what time the interview was at and have multiple within one day.	- Primary key on (interview_date, candidate_id, position_id). A candidate can be interviewed multiple times for the same role; it can even be multiple times on the same day since we are using datetime (eg: one at 1pm and another at 3pm). The candidate can also be interviewed for different positions.	- Candidates, 1:n; a candidate can have many interviews, but an interview can only have one candidate. - Positions, 1:n; a position can have many interviews but an interview can only be for one position.	- **DELETE policy is restrict on position_id.** Interview data is not very meaningful without knowing the position the interview was for. We want to keep a record of the positions that we have interviewed for and users should be discouraged from deleting them, especially since they may open again in the future. Instead of deleting positions, we can use the "openings" column in the Positions table to keep track of whether it is active or not. - **DELETE policy is set null for candidate_id** - its possible that we will want to wipe a candidate completely from the system (and certain privacy regulations may even require this). However, we do still want to keep a record of interviews that have occurred for a particular position for analytics reasons. So, we will just set candidate id to null. - **UPDATE policy is CASCADE on position_id and candidate_id.** We want the most up to date data.
Offers		- Use a unique identifier for each offer. This allows the flexibility of candidates being offered different positions on the same date or getting offered a position more than once (eg: they re-apply years later). Also, multiple candidates can be offered the same position (because what if one rejects?). The alternative would be to have a unique key on (offer_date, position_id, candidate_id) but then we would not be able to set candidate_id to null for our DELETE policy.	- Candidates, 1:n; a candidate can have many offers, but an offer can only be made to one candidate. - Positions, 1:n; a position can have many offers but an offer can only be to one person.	- **DELETE policy is restrict on position_id.** Offer data is not meaningful without knowing the position the offer was for. We want to keep a record of the offers we have made and for which positions and users should be discouraged from deleting them, especially since they may open again in the future - **DELETE policy is SET NULL on candidate_id.** Even if a candidate is deleted from our system, we want to keep track of the offers that have gone out for each position. - **UPDATE policy is CASCADE on position_id and candidate_id.** We want the most up to date data.

Table	Typing & Columns	Unique / Primary key	Foreign Keys	Policies
CandidateSkills		- Unique key on (candidate_id, skill_id)	- Candidates, 1:n; one Candidate can have many CandidateSkills but a CandidateSkill only applies to one candidate. - Skills, 1:n; one Skill can have many CandidateSkills but a CandidateSkill only applies to one skill.	- **DELETE policy is set to CASCADE for FK to Candidates.** CandidateSkill data is meaningless if the Candidate does not exist. - **DELETE policy is set to CASCADE for FK to Skills.** CandidateSkill data is meaningless if the Skill does not exist. - **UPDATE policy is set to CASCADE for FK to Candidates.** We want the most up to date candidate data. - **UPDATE policy is set to CASCADE for FK to Skills.** We want the most up to date skills data.
PositionSkills		- Unique key on (position_id, skill_id)	- Skills, 1:n; one Skill can have many PositionSkills - Positions, 1:n; one Position can have many PositionSkills	- **UPDATE policy is set to CASCADE for FK to Skills.** We want to have up to date skills data. - **DELETE policy is set to CASCADE for FK to Skills.** If a skill no longer exists, the PositionSkill entry is meaningless - it now only represents a position and nothing more. - **UPDATE policy is set to CASCADE for Positions.** We want to have up to date positions data. - **DELETE policy is set to CASCADE for FK to Positions.** If a position is deleted, we no longer care about any skills that were associated with it.

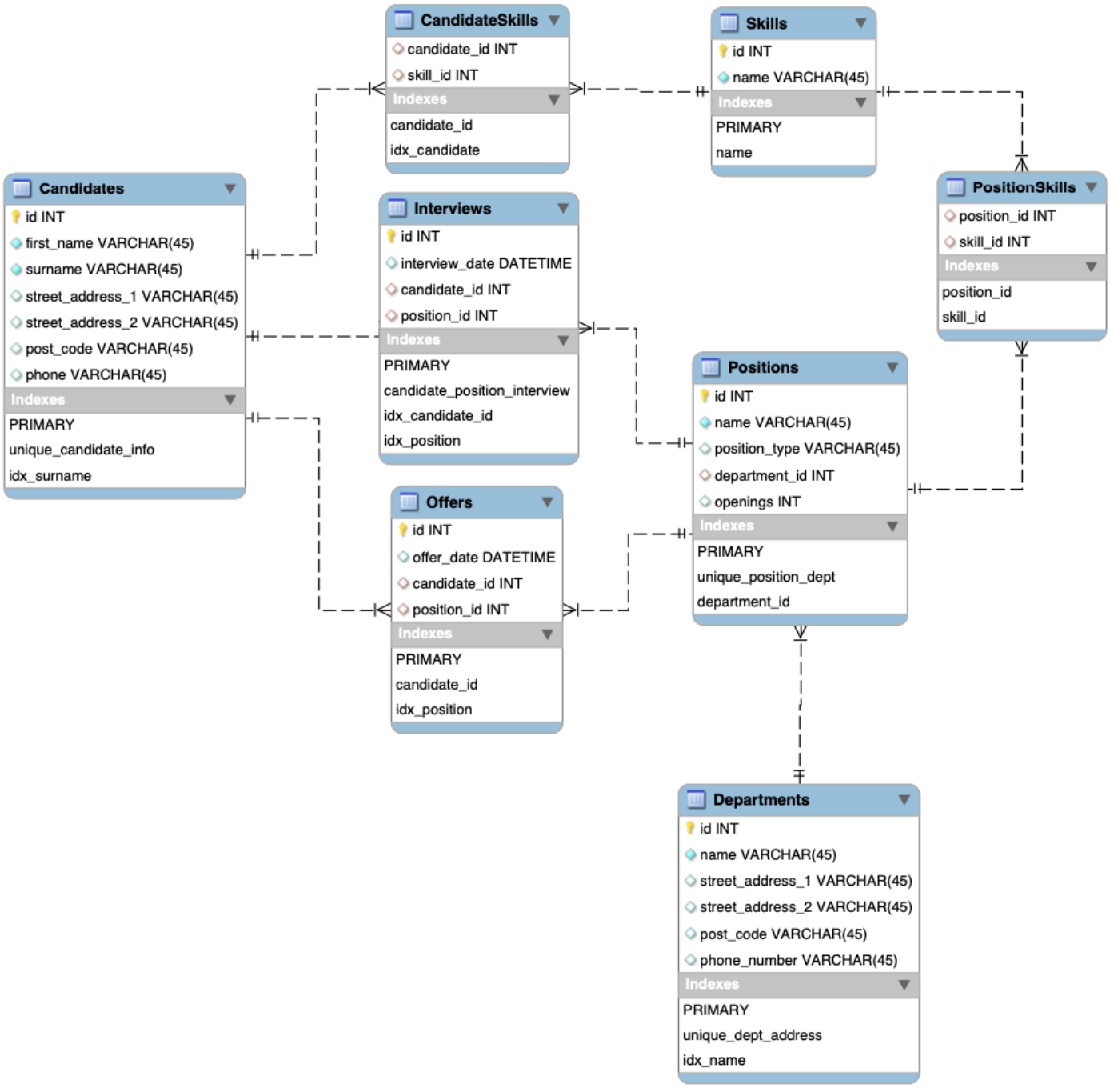
Indexes

Where possible, indexes were added to tables based on their most frequent queries in order to improve performance. These include:

- *surname* on **Candidates** table
- *name* of department in **Departments** table
- *position_id* in **Interviews** table
- *skill_id* in **CandidateSkills** table
- *position_id* in **Offers** table

Operating system

This database was built on MacOS Ventura 13.5.2, MySQLWorkbench 8.0.


CandidateSkills
candidate_id INT
skill_id INT
Indexes
candidate_id
idx_candidate
Skills
id INT
name VARCHAR(45)
Indexes
PRIMARY
name
Candidates
id INT
first_name VARCHAR(45)
surname VARCHAR(45)
street_address_1 VARCHAR(45)
street_address_2 VARCHAR(45)
post_code VARCHAR(45)
phone VARCHAR(45)
Indexes
PRIMARY
unique_candidate_info
idx_surname
Interviews
id INT
interview_date DATETIME
candidate_id INT
position_id INT
Indexes
PRIMARY
candidate_position_interview
idx_candidate_id
idx_position
PositionSkills
position_id INT
skill_id INT
Indexes
position_id
skill_id
Positions
id INT
name VARCHAR(45)
position_type VARCHAR(45)
department_id INT
openings INT
Indexes
PRIMARY
unique_position_dept
department_id
Offers
id INT
offer_date DATETIME
candidate_id INT
position_id INT
Indexes
PRIMARY
candidate_id
idx_position
Departments
id INT
name VARCHAR(45)
street_address_1 VARCHAR(45)
street_address_2 VARCHAR(45)
post_code VARCHAR(45)
phone_number VARCHAR(45)
Indexes
PRIMARY
unique_dept_address
idx_name