Normal Forms Discussion

Highest Normal Form: BCNF

We initially tried to identify all the individual components of our project scope and make them entities; *"User"* (*"Student_Alum"* or *"Recruiter"*), *"Job"*, *"Job_Application"*, *"Employer"* and *"Resource."* To us, these were individual concepts that, given relationships between one another, would satisfy our functional requirements. Anything else would be aggregating concepts together, or going beyond the scope of the project. Additionally, we decided on using generated IDs for most of the entities making it easy to have only the primary key be what uniquely identifies each tuple, and not solely relying on things that exist in the real world. Despite this simplistic approach which technically satisfies higher normal forms, we quickly encountered violations relating to 1NF and its no-multivalue requirement. For example, an employer may have multiple jobs posted, a student may have multiple degrees, and a job may require multiple skills.

In order to satisfy 1NF's conditions we had to make a few changes. We added the *"Education"* entity making it possible for a User_ID to be referenced as a foreign key by multiple tuples. We also made it so a recruiter and employer do not have multivalued attributes relating to posted jobs. Instead, our final design includes the Employer_ID and recruiter User_ID as foreign keys in the job entity. This enables a use case where you could gather information about a poster of jobs. Additionally, we decided against a uniquely generated Job_Application_ID, as we realized that Student_Alum's should technically only be able to apply to a Job once, therefore the pairing of a Job_ID and User_ID was enough to uniquely identify a job application. Given this realization, it also made sense to convert it into a relationship. Finally, we knew we could not have skills contained in a single string due to it being multivalued, so we added a *"Skill"* entity with associated relationships between it and resource, job, and student/alum.

The only gripe we had in the end was the complexity that involves the skill entity. We initially wanted to simplify the approach by using simple string matching algorithms, but thinking this technical ahead of schedule made it so we were not in normal form. Still, we wonder if we will change our approach for future parts to remove some complexity we find may not be necessary.

*Some entities/relationships are left out for their trivial answers.

1NF: All attributes must be atomic

Entity/Relationship	Result

USER	Examples like location, contact, and name are properly split up.
STUDENT_ALUM	Preferences are properly split up into individual meaningful components. Resume and experience are “BLOBs” to allow for implementation interpretation, but still represent individual components.
RECRUITER	N/A (No new attributes from USER).
EMPLOYER	All single values.
EDUCATION	Split into meaningful individual components. Is_Graduated is derived and not stored from Graduation_Date. Does not violate.
RESOURCE	Similar to resume and experience: documents are BLOBs but still single valued.
SKILL	All single values.
JOB	Location, salary min-max, and other information is properly split up into individual components.
JOB_APPLICATION	All single values.

2NF: Non-key attributes depend only on the primary key

Entity/Relationship	Result
USER	The primary key is User_ID, and all attributes rely.
STUDENT_ALUM	The primary key is still User_ID.
RECRUITER	The primary key is still User_ID.
EMPLOYER	The primary key is Employer_ID, and all attributes rely.
EDUCATION	The primary key is Education_ID, and all attributes rely. Is_Graduated is derived and not stored from Graduation_Date.
RESOURCE	The primary key is Resource_ID, and all attributes rely.
SKILL	The primary key is Skill_ID, and all attributes

	rely. *Tuples may also be uniquely identified with Skill_Name for now.
JOB	The primary key is Job_ID, and all attributes rely.
JOB_APPLICATION	The primary key is Job_ID and User_ID, and all attributes rely.

3NF: Remove all transitive dependencies

Entity/Relationship	Result
USER	All non-key attributes are dependent on the primary key ONLY, despite any real-life correlation (ie: email and phone number, or location).
STUDENT_ALUM	N/A
RECRUITER	N/A
EMPLOYER	All non-key attributes are dependent on the primary key ONLY, despite any real-life correlation (ie: name of the company and its location).
EDUCATION	All non-key attributes are dependent on the primary key ONLY, despite any real-life correlation (ie: the degrees that a university offers).
RESOURCE	All non-key attributes are dependent on the primary key ONLY, despite any real-life correlation.
SKILL	All non-key attributes are dependent on the primary key ONLY, despite any real-life correlation.
JOB	All non-key attributes are dependent on the primary key ONLY, despite any real-life correlation. Derived Is_Expired is excluded.
JOB_APPLICATION	All non-key attributes are dependent on primary key ONLY.

BCNF: For every functional dependency X->Y, X is the superkey

Entity/Relationship	Result
USER	User_ID is superkey
STUDENT_ALUM	Same as USER
RECRUITER	Same as USER
EMPLOYER	Employer_ID is superkey
EDUCATION	Education_ID is superkey
RESOURCE	Resource_ID is superkey
SKILL	Skill_ID is superkey
JOB	Job_ID is superkey
JOB_APPLICATION	(Job_ID, User_ID) is superkey