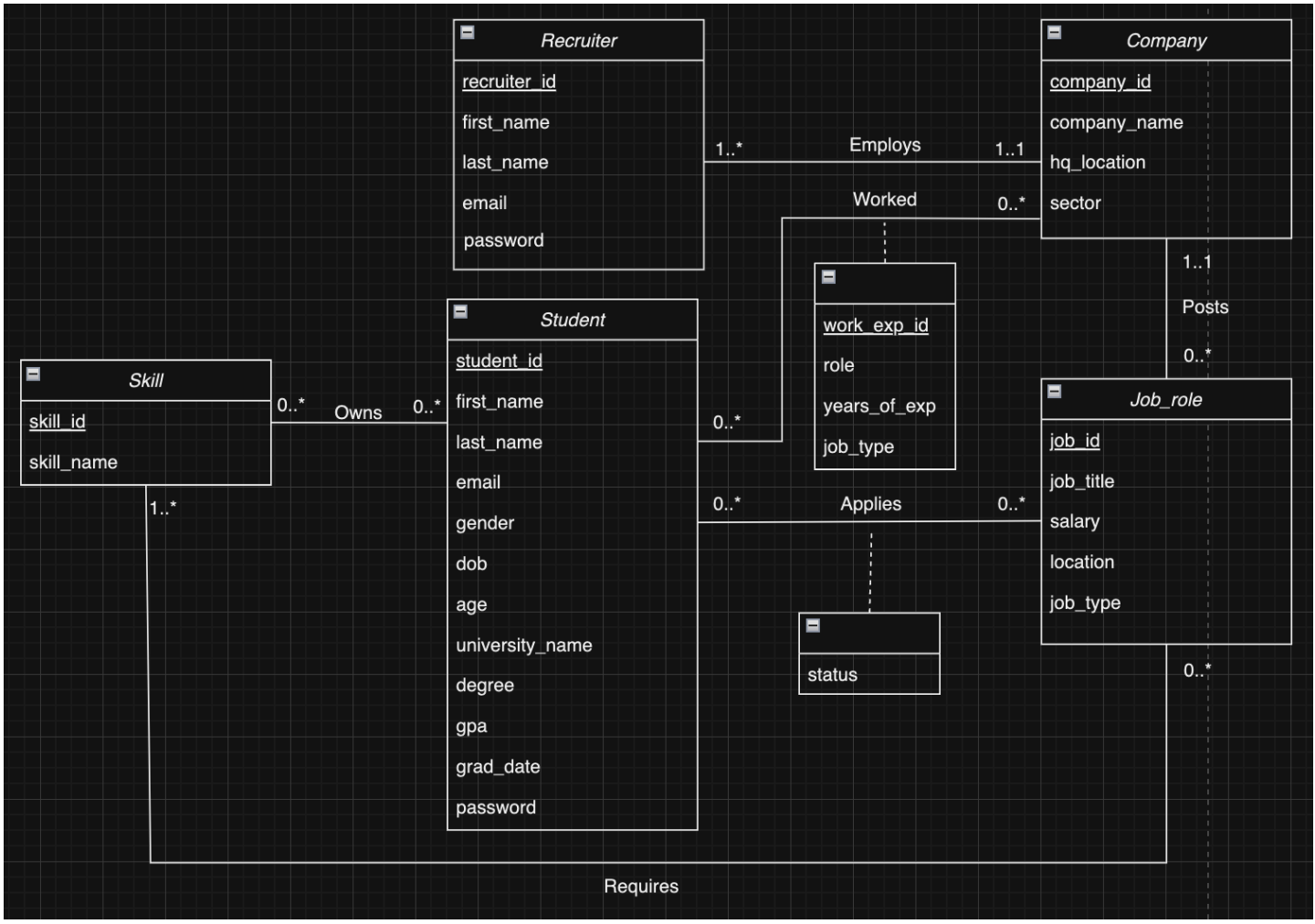
HireIt

Stage 2: Database Design
Team No: 081
Team Name: ACID

UML Diagram:



Assumptions:

- We assume that the Company table will have the list of all the companies that are recruiting through our platform. This also includes companies that a student may have worked for previously. For example, if a student has worked for Amazon previously and is currently applying for Microsoft, we assume that both these companies exist in the Company table.
- We assume that each job role posting will require a minimum of one skill.

- We assume that the number of years of work experience(years_of_exp) is a whole number. Ie, We consider years of experience in terms of years and not months.
- The dob attribute in the student table determines the value of age

Description of each relationship and its cardinality

- **Employs**:
  This is a one-many relationship table between the company and recruiter table.
  1 company can have 1 or more employees
  1 recruiter will be employed by one company

- **Worked:**
  This is a many-many relationship table between the company and student table. This relationship table also contains attributes for job role (role), years of experience (years_of_exp) and work experience ID (work_exp_id).
  A student could have worked at 0 or many of the companies in the past
  A company could have been the workplace for 0 or many students

- **Posts:**
  This is a one-many relationship table between the company and job_role table.
  1 company can have 0 or more job role postings
  A job role posting will be will posted by 1 company

- **Applies:**
  This is a many-many relationship table between the student and job_role table. This relationship table also has an attribute to track the status of the job application (status).
  A student can apply to 0 or many job roles
  A job role can be applied by 0 or many students

- **Requires:**
  This is a one-many relationship table between the job_role and skill table.
  1 job role will require 1 or many skills
  A skills maybe required by 0 or many job roles

- **Owns:**
  This is a many-many relationship table between the student and skills table.
  A student can own 1 or many skills
  A skill maybe owned by 1 or many students

Relational Schema:

- **Student** (student_id:INT NOT NULL [PK], first_name:VARCHAR(255) , last_name:VARCHAR(255), email:VARCHAR(255), gender:VARCHAR(255),dob:DATETIME, age: INT, university_name:VARCHAR(255), degree:VARCHAR(255), gpa:DECIMAL(1,2), grad_date: DATETIME, password: VARCHAR(20))

- **Recruiter** (recruiter_id: INT NOT NULL [PK] , first_name: VARCHAR(255), last_name: VARCHAR(255), email: VARCHAR(255), password: VARCHAR(20))

- **Company** (company_id: INT NOT NULL [PK] L, company_name: VARCHAR(255), hq_location: VARCHAR(255), sector: VARCHAR(255))

- **Job_role** (job_id: INT [PK], job_title: VARCHAR(255), salary: INT, location: VARCHAR(255), job_type: VARCHAR(255))

- **Skill** (skill_id: INT NOT NULL [PK], skill_name: VARCHAR(255))

- **Applies** (status: VARCHAR(255), student_id:INT [FK to Student.student_id], recruiter_id:INT [FK to Recruiter.recruiter_id], student_id,recruiter_id:INT,INT [PK])

- **Employs** (company_id:INT [FK to Company.company_id], recruiter_id:INT [FK to Recruiter.recruiter_id], company_id,recruiter_id:INT,INT [PK])
- **Worked** (company_id:INT [FK to Company.company_id], student_id :INT [FK to Student.student_id], work_exp_id:INT [PK], role: VARCHAR(255), years_of_exp: INT, job_type: VARCHAR(255))

- **Posts** (company_id:INT [FK to Company.company_id], job_id:INT [FK to Job_role.job_id, company_id,job_id:INT,INT [PK])

- **Requires** (student_id:INT [FK to Student.student_id, job_id:INT [FK to Job_role.job_id], job_id:INT, student_id,job_id: INT,INT [PK])

- **Owns** (student_id:INT [FK to Student.student_id], skill_id:INT [FK to Skill.skill_id], student_id,skill_id: INT,INT [PK])

Functional Dependencies:

Recruiter

- recruiter_id -> first_name, last_name, email, password
- email -> recruiter_id, first_name, last_name, password

Company

- company_id -> company_name, hq_location, sector

Job_role

- job_id -> job_title, salary, location, job_type

Student

- student_id -> first_name, last_name, email, gender, dob, age, university_name, degree, gpa, grad_date, password
- email -> first_name, last_name, student_id , gender, dob, age, university_name, degree, gpa, grad_date, password
- student_id, dob -> age
- email_id, dob -> age

Skill

- skill_id -> skill_name

Applies

- student_id, job_id -> status

Worked

- work_exp_id -> student_id, company_id, role, job_type, years_of_exp

Normalization:

1. **Recruiter Relation:**
    - recruiter_id -> A
    - first_name -> B
    - last_name -> C
    - email -> D
    - password -> E

    A -> BCDE

D -> ABCE

a) Making RHS of every FD as singleton
   A ->B
   A ->C
   A ->D
   A ->E
   D ->A
   D ->B
   D ->C
   D ->E

b) There are no redundant attributes in the LHS
c) Removing unnecessary FD's
   A ->D
   D ->B
   D ->C
   D ->E
   D ->A

   Finally,
   A ->D
   D ->ABCE

Final FD's:
- recruiter_id -> email
- email -> recruiter_id, first_name, last_name, password

Candidate Key - {recruited_id, email}

In the above relation, the LHS of the minimal basis FD is a super key. Therefore the relation is in both BCNF and 3NF

2. Company Relation:

- company_id -> A
- company_name-> B
- hq_location-> C
- sector -> D

A -> BCD

a. Making RHS of every FD as singleton

A ->B
A ->C
A ->D

b. There is no redundant attributes in the LHS

c. Removing unnecessary FD's

A ->B
A ->C
A ->D

Finally,
A->BCD

Final FD:

- company_id-> company_name, hq_location, sector
- Candidate Key - {company_id}

In the above relation, the LHS of the minimal basis FD is a super key. Therefore the relation is in both BCNF and 3NF

3. Job Relation:

- job_id -> A
- job_title-> B
- salary-> C
- location-> D
- job_type -> E

A -> BCDE

a. Making RHS of every FD as singleton

A ->B
A ->C
A ->D
A -> E

b. There is no redundant attributes in the LHS

c. Removing unnecessary FD's

A ->B
A ->C

A ->D

Finally,
A -> BCDE

Final FD:

- job_id -> job_title, salary, location, job_type
- Candidate Key - {job_id}

In the above relation, the LHS of the minimal basis FD is a super key. Therefore the relation is in both BCNF and 3NF

4. Skill Relation:

- skill_id -> A
- skill_name-> B

A -> B

a. Making RHS of every FD as singleton
A ->B

b. There is no redundant attributes in the LHS
c. Removing unnecessary FD's
A ->B

Final FD:

- skill_id -> skill_name

Candidate Key - {skill_id }
In the above relation, the LHS of the minimal basis FD is a super key. Therefore the relation is in both BCNF and 3NF

5. Student Relation

- student_id -> A
- first_name-> B
- last_name-> C
- email-> D
- gender -> E
- dob->F

- age->G
- university_name->H
- degree->I
- gpa->I
- grad_date->J
- password->K

A -> BCDEFGHIJK
D -> ABCEFGHIJK
AF->G
DF->G

a. Making RHS of every FD as singleton

A ->B
A ->C
A ->D
A ->E
A ->F
A ->G
A ->H
A ->I
A ->J
A ->K
D ->B
D ->C
D ->A
D ->E
D ->F
D ->G
D ->H
D ->I
D ->J
D ->K
AF -> G
DF- > G

b. Redundant attributes in the LHS
AF -> G is redundant because A->G
Therefore it can be AF->G can be reduced to A->G

DF -> G is redundant because D -> G
Therefore DF-> G can be reduced to D -> G

c. Removing unnecessary FD's

A ->D
D -> A
D -> B
D -> C
D -> E
D -> F
D -> G
D -> H
D -> I
D -> J
D -> K

Finally,
A -> D
D -> ABCEFGHIJK

Final FD:

- student_id -> email
- email -> first_name, last_name, student_id , gender, dob, age, university_name, degree, gpa, grad_date, password

Candidate Keys - {student_id, email}
In the above relation, the LHS of all the minimal basis FD is a super key.
Therefore the relation is in both BCNF and 3NF

6. Applies Relation

- student_id->A
- job_id -> B
- Status -> C

AB -> C

a. Making RHS of every FD as singleton

AB -> C

b. There is no redundant attributes in the LHS
c. Removing unnecessary FD's

AB -> C

Final FD:

- Student_id, job_id -> Status

Candidate Key - {student_id}
In the above relation, the LHS of the minimal basis FD is a super key. Therefore the relation is in both BCNF and 3NF

7. Worked Relation

- work_exp_id -> A
- Student_id -> B
- Company_id -> C
- Role -> D
- job_type -> E
- Years_of_exp -> F

A -> BCDEF

a) Making RHS of every FD as singleton
   A ->B
   A ->C
   A ->D
   A ->E
   A ->F
b) There is no redundant attributes in the LHS

c) Removing unnecessary FD's
   A ->B
   A ->C
   A ->D
   A ->E
   A ->F

   Finally,
   A -> BCDEF

Final FD's:

- work_exp_id -> student_id, company_id, role, job_type, years_of_exp

Candidate Key - {work_exp_id}
In the above relation, the LHS of the minimal basis FD is a super key. Therefore the relation is in both BCNF and 3NF

Normalized FD's:

- recruiter_id -> email
- email -> recruiter_id, first_name, last_name, password
- company_id-> company_name, hq_location, sector
- job_id -> job_title, salary, location, job_type
- skill_id -> skill_name
- student_id -> email
- email -> first_name, last_name, student_id , gender, dob, age, university_name, degree, gpa, grad_date, password
- Student_id, job_id -> Status
- work_exp_id -> student_id, company_id, role, job_type, years_of_exp

3NF vs BCNF:

We have meticulously designed our database to consist of relations that are both 3NF and BCNF. We reduced redundancy in the FDs by identifying their corresponding minimal basis.