COMP 353 Main Project (iyc353_1)

Luigi Besani Urena (40030054)
Anoop Pukulakatt (40130695)
Abdul Sirawan (40074202)
Jeffrey Wilgus (29206345)

August 9, 2020

1 Design

Figure 1 shows the entity-relationship diagram for the Web Career Portal database, and Figure 2 shows the derived database schema.

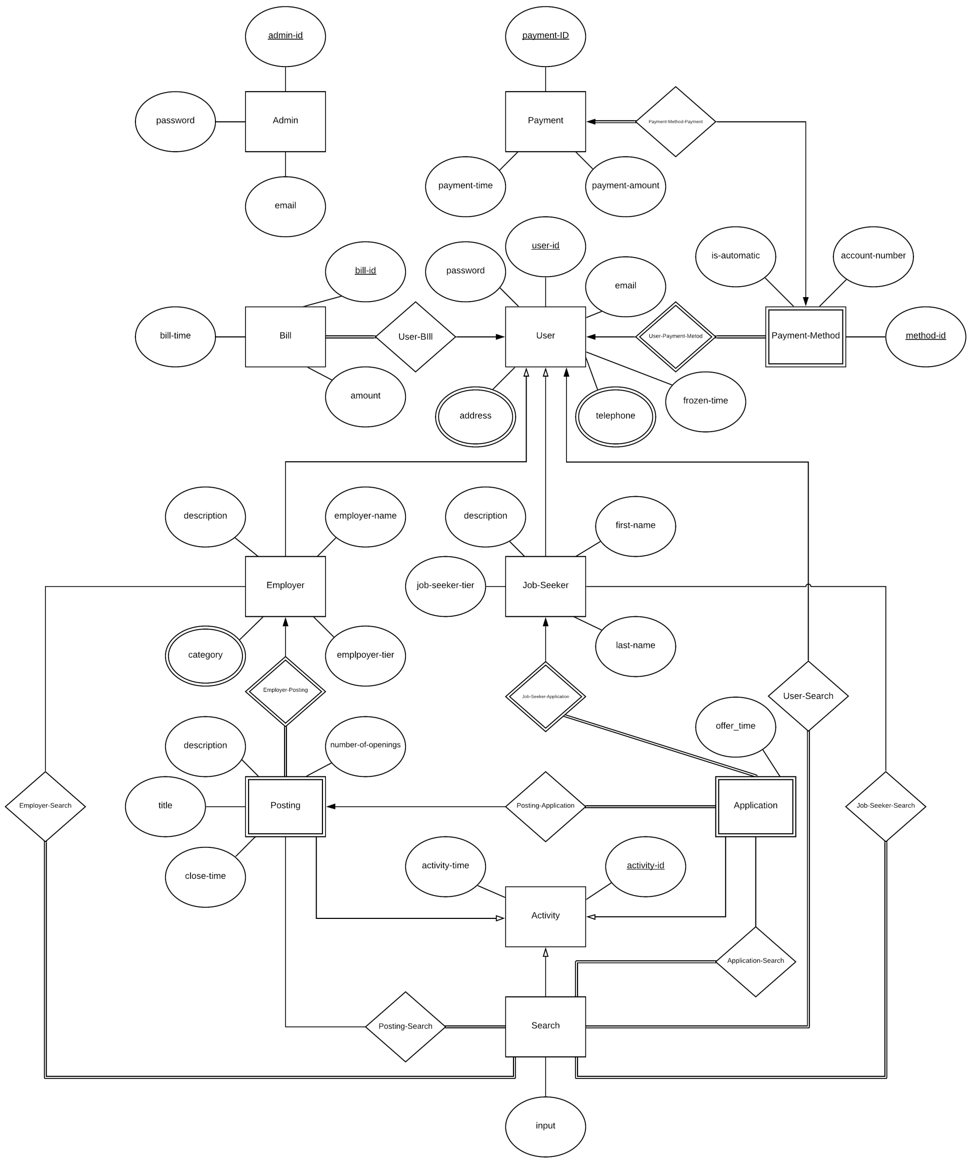

Figure 1: Web Career Portal ER Diagram

1.1 Foreign Key Constraints

Table 1 lists the foreign key constraints that hold on the relations from Figure 2.

1.2 Propagation of Modifications

The deletion of a tuple in a relation containing an attribute for which a foreign key constraint exists on another relation for the most part results in the deletion of any referring tuples in the constrained relation.

admin(admin-id, email, password)
user(user-id, email, password, frozen-time)
address(user-id, street-number, street-name, city, state, country, postal-code, type)
telephone(user-id, phone-number, type)
payment-method(user-id, method-id, account number, is-automatic)
bill(bill-id, user-id, bill-amount, bill-time)
payment(payment-id, user-id, method-id, payment-amount, payment-time)
employer(employer-id, employer-name, description, employer-tier)
employer-category(employer-id, category)
job-seeker(job-seeker-id, first-name, last-name, description, job-seeker-tier)
posting(employer-id, posting-id, posting-time, close-time, title, description, number-of-openings)
application(job-seeker-id, application-id, employer-id, posting-id, application-time, offer-time)
search(search-id, user-id, input, search-time)
employer-search(employer-id, search-id)
job-seeker-search(job-seeker-id, search-id)
posting-search(employer-id, posting-id, search-id)
application-search(job-seeker-id, application-id, search-id)

Figure 2: Web Career Portal Database Schema

However, the *payment* relation merely nullifies referenced *user-id*s and (payment) *method-id*s should a valid referent cease to exist in either the *user* or *payment-method* relations. This way, pertinent information contained in that relation is not destroyed if and when a user decides to delete their account or remove a payment method from their account. For similar reasons, the deletion of a tuple in the *user* relation does not result in the removal from the *search* relation any tuple referencing that user.

Keep in mind that these are not the only measures we might take to prevent the loss of valuable information. A protocol for deleting an account may have the corresponding tuple in the *user* relation take a null value on its *password* attribute (or perhaps a salted value that indicates deactivation so that the original password may be recovered at a later time if necessary) thus obviating the need to propagate the potentially destructive effects of deleting the tuple outright.

Care should be taken to prevent the deletion of a tuple from the *user* relation for which an outstanding balance exists (see §1.8 for a discussion on how to make such a determination). However, given the difficulty in enforcing this constraint directly on the database, it seems best to implement the logic at the front end of the system (i.e. check first for an outstanding balance, then permit query to be sent only if there is none).

1.3 Domain Constraints

The following constraints hold on the values that the specified attributes may take.

- All attributes containing the suffix "id" are integers on the interval $(0, \infty)$.
- *employer-tier* and *job-seeker-tier* are integers on the intervals $[0, 1]$ and $[0, 2]$ respectively.
- All attributes containing the suffix "time" are of the datetime data type.
- All attributes containing the suffix "amount" are of a numeric data type with seven significant digits and two digits of precision after its decimal point.
- All other attributes are arbitrary character strings.

1.4 Default Values

Most attributes do not have a defined default value. However, temporal attributes for the most part default to the current value of the system clock. Exceptionally the $frozen\text{-}time$, *close-time*, and $offer\text{-}time$ attributes of the *user*, *posting*, and *application* relations respectively default to null. $frozen\text{-}time$ indicates the moment at which an account went into arrears by virtue of keeping a negative balance for a prescribed period of time

Constrained Relation	**Constrained Attribute**	**Referenced Relation**	**Referenced Attribute**
address	*user-id*	*user*	*user-id*
telephone	*user-id*	*user*	*user-id*
payment-method	*user-id*	*user*	*user-id*
bill	*user-id*	*user*	*user-id*
payment	*user-id*	*payment-method*	*user-id*
payment	*method-id*	*payment-method*	*method-id*
employer	*employer-id*	*user*	*user-id*
employer-category	*employer-id*	*employer*	*employer-id*
job-seeker	*job-seeker-id*	*user*	*user-id*
posting	*employer-id*	*employer*	*employer-id*
application	*job-seeker-id*	*job-seeker*	*job-seeker-id*
application	*employer-id, posting-id*	*posting*	*employer-id, posting-id*
search	*user-id*	*user*	*user-id*
employer-search	*employer-id*	*employer*	*employer-id*
employer-search	*search-id*	*search*	*search-id*
job-seeker-search	*job-seeker-id*	*job-seeker*	*job-seeker-id*
job-seeker-search	*search-id*	*search*	*search-id*
posting-search	*employer-id, posting-id*	*posting*	*employer-id, posting-id*
posting-search	*search-id*	*search*	*search-id*
application-search	*job-seeker-id, application-id*	*application*	*job-seeker-id, application-id*
application-search	*search-id*	*search*	*search-id*

Table 1: Foreign Key Constraints on Web Career Portal Database Relations

after the most recently issued bill. It is reset to null every time a user brings their balance to a non-negative value. The other two attributes should generally only be modified once an employer has closed a posting and made an offer to an applicant respectively. The default value of the *is-automatic* attribute on the *payment-method* relation is false.

1.5 Nullity Constraints

Table 2 lists the attributes from the relations listed in Figure 2 that may not hold null values. Primary and foreign keys are not listed for the sake of brevity but also may not be null.

Relation	**Attribute**
admin	*email, password*
user	*email, password*
payment	*account-number*
bill	*bill-amount, bill-time*
payment	*payment-amount, payment-time*
employer	*employer-name, employer-tier*
job-seeker	*first-name, last-name, job-seeker-tier*
posting	*posting-time, title, description*
search	*input, search-time*

Table 2: Non-null Attributes

The constraint on the *password* attribute on the *user* relation may be relaxed should the scheme described in §1.2, in which account deletion is signified by nullified passwords rather than the physical removal of user tuples, is implemented.

1.6 Temporal Constraints

The following constraints hold on the specified attributes such that a total ordering is imposed on the tuples from the implicated relations.

- If a tuple p from the *payment* relation holds a value $p.bill\text{-}id$ where $p.bill\text{-}id = b.bill\text{-}id$ for some tuple b from the *bill* relation, then $b.bill\text{-}time < p.payment\text{-}time$.
- If a tuple a from the *application* relation holds values $a.employer\text{-}id$ and $a.posting\text{-}id$ where $a.employer\text{-}id = p.employer\text{-}id$ and $a.posting\text{-}id = p.posting\text{-}id$ for some tuple p from the *posting* relation, then $p.posting\text{-}time < a.application\text{-}time$, $a.application\text{-}time < p.close\text{-}time$, and $a.offer\text{-}time < p.close\text{-}time$.
- For a tuple p from the *posting* relation, $p.posting\text{-}time < p.close\text{-}time$.
- For a tuple a from the *application* relation, $a.application\text{-}time < a.offer\text{-}time$.

It may prove inefficient to implement these constraints using the facilities provided by the DBMS directly, and so we shall limit the interface to the database in such a way as to prevent violation of the constraints without explicitly enforcing them.

1.7 Functional Dependencies and Normalization

A thorough examination of the previously listed relations reveals that, for the most part, all functional dependencies are adequately expressed in primary key constraints. That is, for all non-trivial functional dependencies $\alpha \to \beta$ that hold on a relation r in the schema shown in Figure 2, α is a key for that relation. Therefore, each relation is in BCNF save for two exceptions.

The pair of attributes ($email$, $password$) on *user* uniquely identifies an account in the same way that *user-id* does. Formally, $\{email,\ password\} \to user\text{-}id$. However, since *user-id* is a key for the user relation, the relation is in 3NF. Similar reasoning may be applied to the *admin* relation. Since all relations are in either BCNF or 3NF, the schema is in 3NF.

1.8 Other Considerations

User passwords, and the account numbers associated with their various payment methods should be properly hashed before transmission and subsequent storage in the database, but will most likely not be due to time and scope constraints.

The fact that a specific user is an employer or job-seeker may be inferred by the presence of their *user-id* in the *employer* or *job-seeker* relation respectively. A user's outstanding balance, and therefore their account status, may be obtained by comparing the sum of all payment amounts associated with their *user-id* with the sum of all bill amounts associated with their *user-id*.

The *bill-amount* attribute on the *bill* relation introduces some redundancy into the system because its value for a given set of tuples could be determined by joining the *user* and *employer* and the *user* and *job-seeker* relations, and dispatching on the tier attributes of the resulting relations. The join of $users$ and $employers$ is expressed below.

$$\Pi_{user\text{-}id,employer\text{-}tier}(user \bowtie_{user\text{-}id=employer\text{-}id} employer)$$

A similar expression for $job\text{-}seekers$ is easily formulated. Given the complexity of this computation (at least conceptually) we will accept the redundancy

Notice the Admin entity's general lack of participation in relationships with other entities. This indicates that an admin does not contribute directly to the evolution of the information contained in the system, but rather observes it and may make changes without necessarily respecting the rules imposed on other participants.

The status of an application can be determined by inspecting the values $a.application\text{-}time$ and $a.offer\text{-}time$ on a tuple a from the *application* relation and the value $p.close\text{-}time$ on a tuple p from the posting relation, where $a.employer\text{-}id = p.employer\text{-}id$ and $a.posting\text{-}id = p.posting\text{-}id$. Table 3 lists the possible combinations.

Application Time	Offer Time	Close Time	Status
null	null	null	application withdrawn
null	null	non-null	application withdrawn
max-datetime	non-null	null	offer rejected
max-datetime	non-null	non-null	offer rejected
anything else	null	null	application in review
anything else	null	non-null	application rejected
min-datetime	non-null	null	offer accepted
min-datetime	non-null	non-null	offer accepted

Table 3: Possible Application Statuses

2 Data Definition

Listing 1 provides the SQL statements that translate the database design discussed in the previous section into a concrete system.

```sql
CREATE TABLE admin (
    admin_id BIGINT AUTO_INCREMENT,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    PRIMARY KEY (admin_id)
);

CREATE TABLE users (
    user_id BIGINT AUTO_INCREMENT,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    frozen_time DATETIME DEFAULT NULL,
    PRIMARY KEY (user_id)
);

CREATE TABLE address (
    user_id BIGINT,
    street_number INT,
    street_name VARCHAR(127),
    city VARCHAR(127),
    state VARCHAR(127),
    country VARCHAR(127),
    postal_code VARCHAR(127),
    designation VARCHAR(127),
    PRIMARY KEY (user_id, street_number, street_name, city, state, country),
    FOREIGN KEY (user_id)
        REFERENCES users (user_id)
        ON DELETE CASCADE
);

CREATE TABLE telephone (
    user_id BIGINT,
    phone_number VARCHAR(255),
    designation VARCHAR(255),
    PRIMARY KEY (user_id, phone_number),
    FOREIGN KEY (user_id)
        REFERENCES users (user_id)
        ON DELETE CASCADE
);

CREATE TABLE payment_method (
    user_id BIGINT,
    method_id BIGINT,
    account_number VARCHAR(255) NOT NULL,
    is_automatic BOOLEAN DEFAULT FALSE,
    PRIMARY KEY (user_id, method_id),
    FOREIGN KEY (user_id)
        REFERENCES users (user_id)
        ON DELETE CASCADE
);

CREATE TABLE bill (
    bill_id BIGINT AUTO_INCREMENT,
    user_id BIGINT,
    bill_amount DECIMAL(5, 2) NOT NULL,
    bill_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (bill_id),
    FOREIGN KEY (user_id)
        REFERENCES users (user_id)
);

CREATE TABLE payment (
    payment_id BIGINT AUTO_INCREMENT,
    user_id BIGINT,
    method_id BIGINT,
    payment_amount DECIMAL(5, 2) NOT NULL,
    payment_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (payment_id),
    FOREIGN KEY (user_id, method_id)
        REFERENCES payment_method (user_id, method_id)
        ON DELETE SET NULL,
);

CREATE TABLE employer (
    employer_id BIGINT,
    employer_name VARCHAR(255) NOT NULL,
    description VARCHAR(2047),
    employer_tier INT CHECK (employer_tier > -1 AND employer_tier < 2) NOT NULL,
    PRIMARY KEY (employer_id),
    FOREIGN KEY (employer_id)
        REFERENCES users (user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE employer_category (
    employer_id BIGINT,
    category VARCHAR(255),
    PRIMARY KEY (employer_id, category),
    FOREIGN KEY (employer_ID)
        REFERENCES employer (employer_id)
        ON DELETE CASCADE
);

CREATE TABLE job_seeker (
    job_seeker_id BIGINT,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    description VARCHAR(2047),
    job_seeker_tier INT CHECK (job_seeker_tier > -1 AND job_seeker_tier < 3) NOT NULL,
    PRIMARY KEY (job_seeker_id),
    FOREIGN KEY (job_seeker_id)
        REFERENCES users (user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE posting (
    employer_id BIGINT,
    posting_id BIGINT,
    posting_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    close_time DATETIME DEFAULT NULL,
    title VARCHAR(255) NOT NULL,
    description VARCHAR(2047) NOT NULL,
    number_of_openings INT,
    PRIMARY KEY (employer_id, posting_id),
    FOREIGN KEY (employer_id)
        REFERENCES employer (employer_id)
        ON DELETE CASCADE
);

CREATE TABLE application (
    job_seeker_id BIGINT,
    application_id BIGINT,
    employer_id BIGINT,
    posting_id BIGINT,
    application_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    offer_time DATETIME DEFAULT NULL,
    PRIMARY KEY (job_seeker_id, application_id),
    FOREIGN KEY (job_seeker_id)
        REFERENCES job_seeker (job_seeker_id)
        ON DELETE CASCADE,
    FOREIGN KEY (employer_id, posting_id)
        REFERENCES posting (employer_id, posting_id)
        ON DELETE CASCADE
);

CREATE TABLE search (
    search_id BIGINT AUTO_INCREMENT,
    user_id BIGINT,
    input VARCHAR(255) NOT NULL,
    search_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (search_id),
    FOREIGN KEY (user_id)
        REFERENCES users(user_id)
        ON DELETE SET NULL
);

CREATE TABLE employer_search (
    employer_id BIGINT,
    search_id BIGINT,
    PRIMARY KEY (employer_id, search_id),
    FOREIGN KEY (employer_id)
        REFERENCES employer (employer_id)
        ON DELETE CASCADE,
    FOREIGN KEY (search_id)
        REFERENCES search (search_id)
);

CREATE TABLE job_seeker_search (
    job_seeker_id BIGINT,
    search_id BIGINT,
    PRIMARY KEY (job_seeker_id, search_id),
    FOREIGN KEY (job_seeker_id)
        REFERENCES job_seeker (job_seeker_id)
        ON DELETE CASCADE,
    FOREIGN KEY (search_id)
        REFERENCES search (search_id)
);

CREATE TABLE posting_search (
    employer_id BIGINT,
    posting_id BIGINT,
    search_id BIGINT,
    PRIMARY KEY (employer_id, posting_id, search_id),
    FOREIGN KEY (employer_id, posting_id)
        REFERENCES posting (employer_id, posting_id)
        ON DELETE CASCADE,
    FOREIGN KEY (search_id)
        REFERENCES search (search_id)
);
```

```sql
CREATE TABLE application_search (
    job_seeker_id BIGINT,
    application_id BIGINT,
    search_id BIGINT,
    PRIMARY KEY (job_seeker_id, application_id, search_id),
    FOREIGN KEY (job_seeker_id, application_id)
        REFERENCES application (job_seeker_id, application_id)
        ON DELETE CASCADE,
    FOREIGN KEY (search_id)
        REFERENCES search (search_id)
);
```

Listing 1: Data Definition Statements

3 Data Manipulation

Listing 2 provides the SQL statements that populate the database with initial data. Other data may be added through the web interface.

```sql
INSERT INTO admin (admin_id, email, password)
    VALUES (1, 'pcommander0@miitbeian.gov.cn', 'SkooM8d9mDD'),
        (2, 'kruckert1@shareasale.com', 'BPbl5J'),
        (3, 'mrippen2@domainmarket.com', 'jE08z4QRL'),
        (4, 'jscase3@comcast.net', 'NkClyuvak'),
        (5, 'asnasdell4@flavors.me', 'hiIAyYF');

INSERT INTO users (user_id, email, password)
    VALUES (1, 'agohier0@booking.com', 'xpnTTJ8'),
        (2, 'bswaden1@youku.com', 'snzG67SUR'),
        (3, 'cmathewes2@ibm.com', 'HEvXuBuQl3Z'),
        (4, 'bforten3@sciencedaily.com', 'VYzYI8jyC'),
        (5, 'ibeardow4@adobe.com', '7J0upL'),
        (6, 'ohuntingford5@wufoo.com', 'vIZzqNaWyW'),
        (7, 'rmckechnie6@zdnet.com', 'iYhnr62Fn'),
        (8, 'cbutlerbowdon7@google.cn', 'a6tZQabIL'),
        (9, 'msoaper8@paginegialle.it', 'DDuqpotcX'),
        (10, 'felijah9@elegantthemes.com', 'nmsgpuUo43b');

INSERT INTO address (user_id, street_number, street_name, city, state, country, postal_code, designation)
    VALUES (1, '27527', 'Shoshone', 'Atlanta', 'Georgia', 'United States', '30351', 'work'),
        (2, '0', 'Waxwing', 'Rockford', 'Illinois', 'United States', '61105', 'home'),
        (3, '7838', 'Evergreen', 'Phoenix', 'Arizona', 'United States', '85025', 'home'),
        (4, '9', 'Killdeer', 'Ladysmith', 'British Columbia', 'Canada', NULL, 'home'),
        (5, '88', 'Towne', 'Seattle', 'Washington', 'United States', '98104', 'work'),
        (6, '79108', 'Vermont', 'Barrie', 'Ontario', 'Canada', 'L9J', 'home'),
        (7, '966', 'Steensland', 'Saint-Sauveur', 'Québec', 'Canada', 'J0R', NULL),
        (8, '01754', 'Hanson', 'Marystown', 'Newfoundland and Labrador', 'Canada', 'L2N', NULL),
        (9, '1695', 'Lakeland', 'Albuquerque', 'New Mexico', 'United States', '87105', 'work'),
        (10, '52440', 'Merry', 'Arlington', 'Texas', 'United States', '76011', 'home');

INSERT INTO telephone (user_id, phone_number, designation)
    VALUES (1, '5623560622', 'work'),
        (2, '4473604365', 'home'),
        (3, '3695527831', 'mobile'),
        (4, '4344699876', 'home'),
        (5, '5369351277', 'work'),
        (6, '5087655329', 'mobile'),
        (7, '4374569631', 'work'),
        (8, '2618996475', 'home'),
        (9, '3605600615', 'mobile'),
        (10, '6072920866', 'work');

INSERT INTO payment_method (user_id, method_id, account_number)
    VALUES (1, 1, '34-175-9681'),
        (2, 1, '22-867-5650'),
        (3, 1, '22-796-6933'),
        (4, 1, '38-616-9923'),
        (5, 1, '57-972-6811'),
        (6, 1, '21-487-4424'),
        (7, 1, '26-721-5240'),
        (8, 1, '16-976-3616'),
        (9, 1, '50-151-9909'),
        (10, 1, '55-342-6162');

INSERT INTO bill (bill_id, user_id, bill_amount)
    VALUES (1, 1, 50),
        (2, 2, 100),
        (3, 3, 10),
        (4, 4, 20),
        (5, 5, 100),
        (6, 6, 20),
        (8, 8, 10),
        (9, 9, 50),
        (10, 10, 50);

INSERT INTO payment (payment_id, user_id, bill_id, payment_amount)
    VALUES (1, 4, 4, 20),
        (2, 5, 5, 60.8),
        (3, 1, 1, 22.08),
        (4, 8, 8, 5.41),
        (5, 6, 6, 19.91),
        (6, 2, 2, 100),
        (7, 9, 9, 50),
        (8, 10, 10, 49.99);
```

Listing 2: Data Insertion Statements

Listing 3 provides the SQL statements that produce the data to be displayed to various users and admins of the Web Career Portal.

```
-- i ---------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

-- create employer, given name and tier
INSERT INTO employer (employer_name, employer_tier)
    Values (<name>, <tier>);

-------------------------------------------------------------------------------------------------------------------------

-- delete employer, given id
UPDATE users
SET password = password * -1 -- cascades to employer; categories, postings, and applications cannot be resolved
WHERE user_id = <id>;

-------------------------------------------------------------------------------------------------------------------------

-- edit employer, given id, key, and value
UPDATE employer
SET <key> = <value>
WHERE employer_id = <id>;

-------------------------------------------------------------------------------------------------------------------------

-- report employer information, given id
SELECT employer_name, description, email
FROM employer, users
WHERE employer_id = <id>;
    AND employer_id = user_id

--report employer address(es) and phone number(s) (see utilities)

-------------------------------------------------------------------------------------------------------------------------

-- ii --------------------------------------------------------------------------------------------------------------------

-- create category, given category and employer id
INSERT INTO employer_category (employer_id, category)
    VALUES (<id>, <category>);

-------------------------------------------------------------------------------------------------------------------------

-- delete category, given id and category
DELETE FROM employer_category
WHERE employer_id = <id>
    AND category = <category>

-------------------------------------------------------------------------------------------------------------------------

-- edit category, given id and old/new category
UPDATE employer_category
SET category = <new_category>
WHERE employer_id = <id>
    AND category = <old category>;

-------------------------------------------------------------------------------------------------------------------------

-- report category, given id
SELECT category
FROM employer_category
WHERE employer_id = <id>;

-------------------------------------------------------------------------------------------------------------------------

-- iii -------------------------------------------------------------------------------------------------------------------

-- post a new job, given employer id, posting id, title description, and number of openings
INSERT INTO posting (employer_id, posting_id, title, description, number_of_openings)
    VALUES (<employer id>, <posting id>, <title>, <description>, <num openings>);

-------------------------------------------------------------------------------------------------------------------------

-- iv --------------------------------------------------------------------------------------------------------------------

-- provide a job offer given job seeker id, and application id
UPDATE application
SET offer_time = CURRENT_TIMESTAMP
WHERE job_seeker_id = <job seeker id>
    AND application_id = <application id>;

-------------------------------------------------------------------------------------------------------------------------

-- v ---------------------------------------------------------------------------------------------------------------------

-- report posting information, given employer id and posting id
SELECT title, description, posting_time
FROM posting
WHERE employer_id = <employer id>
    AND posting_id = <posting id>;

-- report applicant information given employer id and posting id
SELECT first_name, last_name, application_time, offer_time, close_time
FROM job_seeker, application, posting
WHERE employer_id = <employer_id>
    AND posting_id = <posting_id>
    AND application.employer_id = posting.employer_id
    AND application.posting_id = posting.posting_id;

--------------------------------------------------------------------------------------------------------------------------

-- vi --------------------------------------------------------------------------------------------------------------------

-- report posted jobs, given employer id, origin time, and bound time
SELECT title, description, posting_time, number_of_openings,
    COUNT( SELECT job_seeker_id, application_id
        FROM application) AS number_of_applicants,
    COUNT(SELECT job_seeker_id, application_id
        FROM application
        WHERE application_time IS '1000-01-01 00:00:00' -- sentinel for accepted offer
            AND offer_time IS NOT NULL) AS number_of_accepted_offers
FROM posting, application
WHERE posting.employer_id = <employer id>
    AND application.employer_id = <employer id>
    AND posting.posting_id = application.posting_id
    AND posting_time BETWEEN <origin> AND <bound>
GROUP BY title, description, posting_time, number_of_applicants;

-----------------------------------------------------------------------------------------------------------------------

-- vii ----------------------------------------------------------------------------------------------------------------

-- create job-seeker, given first name, last name, and tier
INSERT INTO job_seeker (first_name, last_name, job_seeker_tier)
    VALUES (<fname>, <lname>, <tier>);

-----------------------------------------------------------------------------------------------------------------------

-- delete job-seeker, given id
UPDATE users
SET password = password * -1 -- cascades to job-seeker; applications cannot be resolved
WHERE user_id = <id>

-----------------------------------------------------------------------------------------------------------------------

-- edit job-seeker, given id, key and value
UPDATE job_seeker
SET <key> = <value>
WHERE job_seeker_id = <id>;

-----------------------------------------------------------------------------------------------------------------------

-- report job-seeker information, given id
SELECT first_name, last_name, description, email
FROM job_seeker, users
WHERE job_seeker_id = users_id
    AND job_seeker_id = <id>;

-- report job-seeker address(es) and phone number(s) (see utilities)

-----------------------------------------------------------------------------------------------------------------------

-- viii ---------------------------------------------------------------------------------------------------------------

-- search for jobs assuming input is employer name
SELECT posting_time, close_time, title, description, number_of_openings
FROM posting, employer
WHERE employer_name = <input>
    AND posting.employer_id = employer.employer_id;

-- search for jobs assuming input is employer category
SELECT posting_time, close_time, title, description, number_of_openings
FROM posting, employer_category
WHERE category = <input>
    AND posting.employer_id = employer_category.employer_id;

-- search for jobs assuming input is a substring of posting title or description
SELECT posting_time, close_time, title, description, number_of_openings
FROM posting
WHERE title LIKE '%{input}%' OR description LIKE '%{input}%'
    AND employer_id IN ( -- to exclude postings from deleted employers
        SELECT employer_id
        FROM employer);

-- add record to search table, given id and input
INSERT INTO search (user_id, input)
    VALUES (<id>, <input>);

-- add previous search results (denoted here A, B, C) to posting-search table, given search id
INSERT INTO posting_search (employer_id, posting_id, search_id)
    SELECT employer_id, posting_id, <search_id>
    FROM <A> UNION <B> UNION <C>; -- distinct would be more expensive and obliterate result frequency information

-----------------------------------------------------------------------------------------------------------------------

-- ix -----------------------------------------------------------------------------------------------------------------

-- apply for a job, given job-seeker id, application id, employer id, and posting id
INSERT INTO application (job_seeker_id, application_id, employer_id, posting_id)
    VALUES (<job-seeker id>, <application id>, <employer id>, <posting id>);

-----------------------------------------------------------------------------------------------------------------------

-- x ------------------------------------------------------------------------------------------------------------------

-- assuming offer time is non-null on given application

-- accept a job offer, given job-seeker id and application id
UPDATE application
SET application_time = '1000-01-01 00:00:00' -- 'When can you start?' 'YESTERDAY!'
WHERE job_seeker_id = <job-seeker id>
    AND application_id = <application id>;

-- reject a job offer, given job-seeker if and application id
UPDATE application
SET application_time = '9999-12-31 23:59:59' -- 'When can you start?' 'When hell freezes over.'
WHERE job_seeker_id = <job-seeker id>
    AND application_id = <application id>;

-----------------------------------------------------------------------------------------------------------------------

-- xi -----------------------------------------------------------------------------------------------------------------
```

```sql
-- assuming offer time is null on given application

-- withdraw application, given job-seeker id and application id
UPDATE application
SET application_time = NULL
WHERE job_seeker_id = <job-seeker id>
    AND application_id = <application id>;

------------------------------------------------------------------------------------------------------------------------

-- xii ------------------------------------------------------------------------------------------------------------------

-- obliterate nullable job-seeker information, given job-seeker id
UPDATE job_seeker
SET description = NULL
WHERE job_seeker_id = <job-seeker id>;

-- delete job-seeker address(es), given job-seeker id
DELETE FROM address
WHERE user_id = <job-seeker id>;

-- delete job-seeker phone number(s) given job-seeker id
DELETE FROM telephone
WHERE user_id = <job-seeker id>;

------------------------------------------------------------------------------------------------------------------------

-- xiii

-- report of applications, given job-seeker id, origin time, and bound time
SELECT title, description, application_time, offer_time, close_time
FROM posting, application
WHERE job_seeker_id = <job-seeker id>
    AND posting.employer_id = application.employer_id
    AND posting.posting_id = posting.posting_id
    AND application_time BETWEEN <origin> AND <bound>;

------------------------------------------------------------------------------------------------------------------------

-- xiv & xv -------------------------------------------------------------------------------------------------------------

-- add a payment method, given user id, method id
INSERT INTO payment_method (user_id, method_id)
    VALUES (<id>, <method id>);

-- if automatic payments are selected
UPDATE payment_method
SET is_automatic = TRUE
WHERE user_id = <id>
    AND method_id = <method id>;

------------------------------------------------------------------------------------------------------------------------

-- delete a payment method, given user id and method id
DELETE FROM payment_method
WHERE user_id = <id>
    AND method_id = <method id>;

------------------------------------------------------------------------------------------------------------------------

-- edit payment method account information, given user id, method id, and new account number
UPDATE payment_method
SET account_number = <new account>
WHERE user_id = <id>
    AND method_id = <method id>;

-- edit payment method is-automatic, given user id, method id, and new value
UPDATE payment_method
SET is_automatic = <new value>
WHERE user_id = <id>
    AND method_id = <method id>;

------------------------------------------------------------------------------------------------------------------------

-- xvi ------------------------------------------------------------------------------------------------------------------

-- make a manual payment, given user id, method id, and amount
INSERT INTO payment (user_id, method_id, payment_amount)
    VALUES (<id>, <method id>, <amount>);

-- if balance for user id is > -1
UPDATE users
SET frozen_time = NULL
WHERE user_id = <id>;

------------------------------------------------------------------------------------------------------------------------

-- xvii -----------------------------------------------------------------------------------------------------------------

-- report employer administrative information
SELECT employer_name, email, frozen_time AS status, balance
FROM employer, users, balance
WHERE employer_id = users.user_id
    AND users.user_id = balance.user_id;

-- report employer categories
SELECT *
FROM employer_category;

-- report job-seeker administrative information
SELECT first_name, last_name, email, frozen_time AS status, balance
FROM job_seeker, users, balance
WHERE job_seeker_id = users.user_id
    AND users.user_id = balance.user_id;

------------------------------------------------------------------------------------------------------------------------
```

```
-- xviii -------------------------------------------------------------------------------------------------------------

-- report employers with negative balances
SELECT employer_name, email, balance, frozen_time
FROM employer, users, balance
WHERE balance < 0
    AND employer_id = users.user_id
    AND users.user_id = balance.user_id;

-- report employees with negative balances
SELECT first_name, last_name, email, balance, frozen_time
FROM job_seeker, users, balance
WHERE balance < 0
    AND job_seeker_id = users.user_id
    AND users.user_id = balance.user_id;

----------------------------------------------------------------------------------------------------------------------

-- utilities ---------------------------------------------------------------------------------------------------------

-- report address(es), given id
SELECT street_number, street_name, city, state, country, postal_code, designation
FROM address
WHERE user_id = <id>;

-- report phone number(s), given id
SELECT phone_number, designation
FROM phone
WHERE user_id = <id>;

-- virtual balances table
CREATE VIEW user_balance
    SELECT payment.user_id, SUM(payment_amount) - SUM(bill_amount) AS balance
    FROM payment, bill
    WHERE payment.user_id = bill.user_id;
```

Listing 3: Queries on the Database