Table: USER

Attribute	Provided in variable	Can be NULL?	Notes
Username	user	N	New username can be provided in variable new_user for admin to update
PasswordHash	hash	N	string password is hashed, hash stored in variable 'hash'
Fname	fn	Y	
Lname	ln	Y	
IsActive	active	N	Default value of 1 == True

Creation

- User can be added by applicant when registering, or by an admin

```
INSERT INTO USER
VALUES (user, hash, fn, ln, active);
```

Updating

- Applicant can update password hash, first and last name
- Admin can also update an applicant's password

```
UPDATE USER
SET PasswordHash = hash
WHERE Username = user;

UPDATE USER
SET Fname = fn
WHERE Username = user;

UPDATE USER
SET Lname= ln
WHERE Username = user;
```

- Admin can additionally update username and active status (deactivate / reactivate user) for a specific user

```
UPDATE USER
SET Username = new_user
WHERE Username = user;

UPDATE USER
SET IsActive = active
```

WHERE Username = user;

Deleting

- Users can only be deleted by an admin

DELETE FROM USER
WHERE Username = user;

Table: APPLICANT

Attribute	Provided in variable	Can be NULL?	Notes
Username	user	N	FK, must reference existing user
Education	edu	Y	

Creation

- Applicant entry created when registering for an account; option to create entry with additional education details provided by applicant

INSERT INTO APPLICANT
VALUES (user, edu);

Updating

- Applicant can update education

UPDATE APPLICANT
SET Education = edu
WHERE Username = user;

Deleting

- Applicants can only be deleted by an admin

DELETE FROM APPLICANT
WHERE Username = user;

Table: APPLICANT_EXPERIENCE

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
Experience	exp	N	updated experience name can be provided in variable new_exp
ExperienceDesc	expd	Y	

Creation

- Applicant can add experience entries

```
INSERT INTO APPLICANT_EXPERIENCE
VALUES (user, exp, expd);
```

Updating

- Applicant can update experience name and description

```
UPDATE APPLICANT_EXPERIENCE
SET Experience = new_exp
WHERE ApplicantUsername = user
AND Experience = exp;

UPDATE APPLICANT_EXPERIENCE
SET ExperienceDesc = expd
WHERE ApplicantUsername = user
AND Experience = exp;
```

Deleting

- Applicants can delete specific experiences by name or all experiences at once

```
DELETE FROM APPLICANT_EXPERIENCE
WHERE ApplicantUsername = user
AND Experience = exp;

DELETE FROM APPLICANT_EXPERIENCE
WHERE ApplicantUsername = user;
```

Table: APPLICANT_PROJECT

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
Project	prj	N	updated project name can be provided in variable new_prj
ProjectDesc	prjd	Y	

Creation

- Applicant can add project entries

```
INSERT INTO APPLICANT_PROJECT
VALUES (user, prj, prjd);
```

Updating

- Applicant can update project name and description

```
UPDATE APPLICANT_PROJECT
SET Project = new_prj
WHERE ApplicantUsername = user
AND Project = prj;

UPDATE APPLICANT_PROJECT
SET ProjectDesc = prjd
WHERE ApplicantUsername = user
AND Project = prj;
```

Deleting

- Applicants can delete specific projects by name or all projects at once

```
DELETE FROM APPLICANT_PROJECT
WHERE ApplicantUsername = user
AND Project = prj;

DELETE FROM APPLICANT_PROJECT
WHERE ApplicantUsername = user;
```

Table: APPLICANT_CERTIFICATION

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
Certification	cert	N	updated certification name can be provided in variable new_cert

Creation

- Applicant can add certification entries

```
INSERT INTO APPLICANT_CERTIFICATION
VALUES (user, cert);
```

Updating

- Applicant can update certification name

```
UPDATE APPLICANT_CERTIFICATION
SET Certification = new_cert
WHERE ApplicantUsername = user
AND Certification = cert;
```

Deleting

- Applicants can delete specific certifications by name or all certifications at once

```
DELETE FROM APPLICANT_CERTIFICATION
WHERE ApplicantUsername = user
AND Certification = cert;

DELETE FROM APPLICANT_CERTIFICATION
WHERE ApplicantUsername = user;
```

Table: APPLICANT_SKILL

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
Skill	skl	N	updated skill name can be provided in variable new_skl

Creation

- Applicant can add skill entries

```
INSERT INTO APPLICANT_SKILL
VALUES (user, skl);
```

Updating

- Applicant can update skill name

```
UPDATE APPLICANT_SKILL
SET Skill = new_skl
WHERE ApplicantUsername = user
AND Skill = skl;
```

Deleting

- Applicants can delete specific skills by name or all skills at once

```
DELETE FROM APPLICANT_SKILL
WHERE ApplicantUsername = user
AND Skill = skl;

DELETE FROM APPLICANT_SKILL
WHERE ApplicantUsername = user;
```

Table: APPLICANT_COMPETITION

Attribute	Provided in variable	Can be NULL?	Notes

ApplicantUsername	user	N	FK, must reference existing applicant
Competition	cmp	N	updated competition name can be provided in variable new_cmp

Creation

- Applicant can add competition entries

```
INSERT INTO APPLICANT_COMPETITION
VALUES (user, cmp);
```

Updating

- Applicant can update competition name

```
UPDATE APPLICANT_COMPETITION
SET Competition = new_cmp
WHERE ApplicantUsername = user
AND Competition = cmp;
```

Deleting

- Applicants can delete specific competitions by name or all competitions at once

```
DELETE FROM APPLICANT_COMPETITION
WHERE ApplicantUsername = user
AND Competition = cmp;

DELETE FROM APPLICANT_COMPETITION
WHERE ApplicantUsername = user;
```

Table: ADMIN

Attribute	Provided in variable	Can be NULL?	Notes
Username	user	N	FK, must reference existing user
DeveloperFlag	flag	N	Default value of 0 == False
DeveloperType	devt	Y	

Creation

- Admins can add other admins
- The first admin in the database must be hard-coded

```
INSERT INTO ADMIN
VALUES (user, flag, devt);
```

Updating

- Depending on their permission levels, admins can update their own DeveloperFlag and, if applicable, their DeveloperType

```
UPDATE ADMIN
SET DeveloperFlag = flag
WHERE Username = user;

UPDATE ADMIN
SET DeveloperType = devt
WHERE Username = user;
```

Deleting

- Admins with certain permissions can remove other admins

```
DELETE FROM ADMIN
WHERE Username = user;
```

Table: ADMIN_RESPONSIBILITY

Attribute	Provided in variable	Can be NULL?	Notes
AdminUsername	user	N	FK, must reference existing admin
Responsibility	resp	N	updated responsibility name can be provided in variable new_resp

Creation

- Admin can add responsibility entries

```
INSERT INTO ADMIN_RESPONSIBILITY
VALUES (user, resp);
```

Updating

- Admin can update responsibility name

```
UPDATE ADMIN_RESPONSIBILITY
SET Responsibility = new_resp
WHERE AdminUsername = user
AND Responsibility = resp;
```

Deleting

- Admins can delete specific responsibilities by name or all responsibilities at once

```
DELETE FROM ADMIN_RESPONSIBILITY
WHERE AdminUsername = user
AND Responsibility = resp;
```

```
DELETE FROM ADMIN_RESPONSIBILITY
WHERE AdminUsername = user;
```

Table: DEV_SPECIALIZATION

Attribute	Provided in variable	Can be NULL?	Notes
AdminUsername	user	N	FK, must reference existing admin
Specialization	spec	N	updated specialization name can be provided in variable new_spec

Creation

- Developer can add specialization entries

```
INSERT INTO DEV_SPECIALIZATION
VALUES (user, spec);
```

Updating

- Developer can update specialization name

```
UPDATE DEV_SPECIALIZATION
SET Specialization = new_spec
WHERE AdminUsername = user
AND Specialization = spec;
```

Deleting

- Developers can delete specific specializations by name or all specializations at once

```
DELETE FROM DEV_SPECIALIZATION
WHERE AdminUsername = user
AND Specialization = spec;

DELETE FROM DEV_SPECIALIZATION
WHERE AdminUsername = user;
```

Table: PERMISSION

Attribute	Provided in variable	Can be NULL?	Notes
PermissionLevel	perm	N	updated permission level can be provided in variable updated_perm

Creation

- Admin can add permission levels

INSERT INTO PERMISSION
VALUES (perm);

Updating

- Admin can update permission level (if for example permission levels need to be rearranged)

UPDATE PERMISSION
SET PermissionLevel = updated_perm
WHERE PermissionLevel = perm;

Deleting

- Admin can delete specific permission level, but never all permissions at once

DELETE FROM PERMISSION
WHERE PermissionLevel = perm;

Table: PERMISSION_ABILITY

Attribute	Provided in variable	Can be NULL?	Notes
PermissionLevel	perm	N	FK, must reference existing permission level
Ability	abl	N	updated ability name can be provided in variable new_abl

Creation

- Admin can add abilities for a specific permission level

INSERT INTO PERMISSION_ABILITY
VALUES (perm, abl);

Updating

- Admin can update ability name

UPDATE PERMISSION_ABILITY
SET Ability = new_abl
WHERE PermissionLevel = perm
AND Ability = abl;

Deleting

- Admins can only delete specific abilities by name, not all abilities for a permission level at once

DELETE FROM PERMISSION_ABILITY
WHERE PermissionLevel = perm
AND Ability = abl;

Table: ADMIN_HAS_PERM

Attribute	Provided in variable	Can be NULL?	Notes
AdminUsername	user	N	FK, must reference existing admin
PermLevel	lvl	N	FK, must reference existing permission level; updated permission level can be provided in variable new_lvl

Creation

- Admin able to assign a permission level to a specific admin

```
INSERT INTO ADMIN_HAS_PERM
VALUES (user, lvl);
```

Updating

- Admin can update a permission level for an admin

```
UPDATE ADMIN_HAS_PERM
SET PermLevel = new_lvl
WHERE AdminUsername = user
AND PermLevel = lvl;
```

Deleting

- Admins can only delete specific permission levels for an admin

```
DELETE FROM ADMIN_HAS_PERM
WHERE AdminUsername = user
AND PermLevel = lvl;
```

Table: INTERVIEW

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
InterviewID	id	N	is an auto-incremented primary key
Stage	stg	Y	
DateTime	dt	N	
ApplicationName	apn	N	FK, must reference existing

			application for the applicant

Creation

- Applicant can create a new interview, assigning all values (with Stage optionally left NULL) apart from InterviewID, as this is an auto-incremented primary key

```
INSERT INTO INTERVIEW(ApplicantUsername, Stage, DateTime, ApplicationName)
VALUES (user, stg, dt, apn);
```

Updating

- Applicant can update interview Stage, DateTime, and application being referenced

```
UPDATE INTERVIEW
SET Stage = stg
WHERE ApplicantUsername = user
AND InterviewID= id;

UPDATE INTERVIEW
SET DateTime = dt
WHERE ApplicantUsername = user
AND InterviewID= id;

UPDATE INTERVIEW
SET ApplicationName = apn
WHERE ApplicantUsername = user
AND InterviewID= id;
```

Deleting

- Applicants can delete specific interviews

```
DELETE FROM INTERVIEW
WHERE ApplicantUsername = user
AND InterviewID= id;
```

Table: DOCUMENT

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
DocFileName	doc	N	updated file name can be provided in variable new_doc
DocType	type	Y	

Description	desc	Y	

Creation

- Applicant can create a new document

```
INSERT INTO DOCUMENT
VALUES (user, doc, type, desc);
```

Updating

- Applicant can update document file name, type, and description

```
UPDATE DOCUMENT
SET DocFileName = new_doc
WHERE ApplicantUsername = user
AND DocFileName = doc;

UPDATE DOCUMENT
SET DocType = type
WHERE ApplicantUsername = user
AND DocFileName = doc;

UPDATE DOCUMENT
SET Description = desc
WHERE ApplicantUsername = user
AND DocFileName = doc;
```

Deleting

- Applicants can delete specific documents

```
DELETE FROM DOCUMENT
WHERE ApplicantUsername = user
AND DocFileName = doc;
```

Table: APPLICATION

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
AName	name	N	updated application name can be provided in variable new_name
Notes	note	Y	
DateSubmitted	dsub	Y	

Status	st	Y	

Creation

- Applicant can create a new application

```
INSERT INTO APPLICATION
VALUES (user, name, note, dsub, st);
```

Updating

- Applicant can update application name, notes, date submitted, status

```
UPDATE APPLICATION
SET AName = new_name
WHERE ApplicantUsername = user
AND AName = name;

UPDATE APPLICATION
SET Notes = note
WHERE ApplicantUsername = user
AND AName = name;

UPDATE APPLICATION
SET DateSubmitted = dsub
WHERE ApplicantUsername = user
AND AName = name;

UPDATE APPLICATION
SET Status = st
WHERE ApplicantUsername = user
AND AName = name;
```

Deleting

- Applicants can delete specific applications

```
DELETE FROM APPLICATION
WHERE ApplicantUsername = user
AND AName = name;
```

Table: APPL_RELEVANT_URL

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
AName	name	N	FK, must reference existing application

RelevantURL	url	N	updated URL can be provided in variable new_url

Creation

- Applicant can add new URLs to an application

```
INSERT INTO APPL_RELEVANT_URL
VALUES (user, name, url);
```

Updating

- Applicant can update an existing URL

```
UPDATE APPL_RELEVANT_URL
SET RelevantURL = new_url
WHERE ApplicantUsername = user
AND AName = name
AND RelevantURL = url;
```

Deleting

- Applicants can delete specific URLs from an application or all URLs for an application at once

```
DELETE FROM APPL_RELEVANT_URL
WHERE ApplicantUsername = user
AND AName = name
AND RelevantURL = url;

DELETE FROM APPL_RELEVANT_URL
WHERE ApplicantUsername = user
AND AName = name;
```

Table: APPL_CATEGORY

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
AName	name	N	FK, must reference existing application
Category	ctg	N	updated category name can be provided in variable new_ctg

Creation

- Applicant can assign new categories to an application

```
INSERT INTO APPL_CATEGORY
```

VALUES (user, name, ctg);

Updating

- Applicant can update an existing category

```
UPDATE APPL_CATEGORY
SET Category = new_ctg
WHERE ApplicantUsername = user
AND AName = name
AND Category = ctg;
```

Deleting

- Applicants can delete specific categories from an application or all categories for an application at once

```
DELETE FROM APPL_CATEGORY
WHERE ApplicantUsername = user
AND AName = name
AND Category = ctg;

DELETE FROM APPL_CATEGORY
WHERE ApplicantUsername = user
AND AName = name;
```

Table: COMPANY

Attribute	Provided in variable	Can be NULL?	Notes
CompanyName	comp	N	updated company name can be provided in variable new_comp
Industry	ind	Y	
HomePageURL	url	Y	
Description	desc	Y	

Creation

- Applicant can add a new company if it does not exist

```
INSERT INTO COMPANY
VALUES (comp, ind, url, desc);
```

Updating

- Applicant can update company name, industry, homepage url, and description

```
UPDATE COMPANY
SET CompanyName = new_comp
```

```
WHERE CompanyName = comp;

UPDATE COMPANY
SET Industry = ind
WHERE CompanyName = comp;

UPDATE COMPANY
SET HomePageURL = url
WHERE CompanyName = comp;

UPDATE COMPANY
SET Description = desc
WHERE CompanyName = comp;
```

Deleting

- **Only admins** can fully delete companies, as there are multiple foreign key references to CompanyName throughout the database.

```
DELETE FROM COMPANY
WHERE CompanyName = comp;
```

Table: JOB

Attribute	Provided in variable	Can be NULL?	Notes
CompName	comp	N	FK, must reference existing company
PositionName	pos	N	updated position name can be provided in variable new_pos
Description	desc	Y	
JobPostFile	file	Y	
PositionType	type	Y	
ApplicationDeadline	deadline	N	
Salary	slr	Y	

Creation

- Applicant can create a new job if it does not exist

```
INSERT INTO JOB
VALUES (comp, pos, desc, file, type, deadline, slr);
```

Updating

- Applicant can update job name, description, job post file, position type, application deadline, and salary

```
UPDATE JOB
SET PositionName = new_pos
WHERE CompName = comp
AND PositionName = pos;

UPDATE JOB
SET Description = desc
WHERE CompName = comp
AND PositionName = pos;

UPDATE JOB
SET JobPostFile = file
WHERE CompName = comp
AND PositionName = pos;

UPDATE JOB
SET PositionType = type
WHERE CompName = comp
AND PositionName = pos;

UPDATE JOB
SET ApplicationDeadline = deadline
WHERE CompName = comp
AND PositionName = pos;

UPDATE JOB
SET Salary = slr
WHERE CompName = comp
AND PositionName = pos;
```

Deleting

- **Only admins** can fully delete jobs, as there are multiple foreign key references to PositionName throughout the database.

```
DELETE FROM JOB
WHERE CompName = comp
AND PositionName = pos;
```

Table: JOB_QUALIFICATION

Attribute	Provided in variable	Can be NULL?	Notes

CompanyName	comp	N	FK, must reference existing company
PositionName	pos	N	FK, must reference existing position
Qualification	qual	N	updated qualification can be provided in variable new_qual

Creation

- Applicant can add new qualifications to a job

```
INSERT INTO JOB_QUALIFICATION
VALUES (comp, pos, qual);
```

Updating

- Applicant can update an existing qualification

```
UPDATE JOB_QUALIFICATION
SET Qualification = new_qual
WHERE CompanyName = comp
AND PositionName = pos
AND Qualification = qual;
```

Deleting

- Applicants can delete specific qualifications for a job, but <u>not</u> all qualifications at once

```
DELETE FROM JOB_QUALIFICATION
WHERE CompanyName = comp
AND PositionName = pos
AND Qualification = qual;
```

<u>Table: JOB_RESPONSIBILITY</u>

Attribute	Provided in variable	Can be NULL?	Notes
CompanyName	comp	N	FK, must reference existing company
PositionName	pos	N	FK, must reference existing position
Responsibility	resp	N	updated responsibility can be provided in variable new_resp

Creation

- Applicant can add new responsibilities to a job

INSERT INTO JOB_RESPONSIBILITY
VALUES (comp, pos, resp);

Updating

- Applicant can update an existing responsibility

UPDATE JOB_RESPONSIBILITY
SET Responsibility = new_resp
WHERE CompanyName = comp
AND PositionName = pos
AND Responsibility = resp;

Deleting

- Applicants can delete specific responsibilities for a job, but not all responsibilities at once

DELETE FROM JOB_RESPONSIBILITY
WHERE CompanyName = comp
AND PositionName = pos
AND Responsibility = resp;

Table: OFFER

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
OfferFileName	offern	N	updated offer file name can be provided in variable new_offern
ResponseDeadline	deadline	N	
Compensation	compens	Y	
StartDate	start	Y	
Notes	note	Y	
CompName	comp	N	FK, must reference existing company
PosName	pos	N	FK, must reference existing job

Creation

- Applicant can create a new offer

INSERT INTO OFFER
VALUES (user, offern, deadline, compens, start, note, comp, pos);

Updating

- Applicant can update offer name, response deadline, compensation, start date, notes, and job that the offer refers to

```
UPDATE OFFER
SET OfferFileName = new_offern
WHERE ApplicantUsername = user
AND OfferFileName = offern;

UPDATE OFFER
SET ResponseDeadline = deadline
WHERE ApplicantUsername = user
AND OfferFileName = offern;

UPDATE OFFER
SET Compensation = compens
WHERE ApplicantUsername = user
AND OfferFileName = offern;

UPDATE OFFER
SET StartDate = start
WHERE ApplicantUsername = user
AND OfferFileName = offern;

UPDATE OFFER
SET Notes = note
WHERE ApplicantUsername = user
AND OfferFileName = offern;

UPDATE OFFER
SET CompName = comp, PosName = pos
WHERE ApplicantUsername = user
AND OfferFileName = offern;
```

Deleting

- Applicants can delete specific offers

```
DELETE FROM OFFER
WHERE ApplicantUsername = user
AND OfferFileName = offern;
```

Table: CONTACT

Attribute	Provided in variable	Can be NULL?	Notes

ContactID	id	N	is an auto-generated primary key
LinkedInURL	url	Y	
Fname	fn	N	
Lname	ln	N	

Creation

- Applicant can create a new contact, if they do not already exist based on first and last name, assigning all values (with LinkedInURL optionally left NULL) apart from ContactID, as this is an auto-generated primary key

```
INSERT INTO CONTACT(LinkedInURL, Fname, Lname)
VALUES (url, fn, ln);
```

Updating

- Applicant can update contact LinkedInURL and first and last names

```
UPDATE CONTACT
SET LinkedInURL = url
WHERE ContactID = id;

UPDATE CONTACT
SET Fname = fn
WHERE ContactID = id;

UPDATE CONTACT
SET Lname = ln
WHERE ContactID = id;
```

Deleting

- **Only admins** can fully delete contacts, as there are multiple foreign key references to ContactID throughout the database.

```
DELETE FROM CONTACT
WHERE ContactID = id;
```

Table: CONTACT_EMAIL

Attribute	Provided in variable	Can be NULL?	Notes
ContactID	id	N	FK, must reference existing contact
Email	em	N	updated email can be provided in variable

			new_em

Creation

- Applicant can add new emails to a contact

```
INSERT INTO CONTACT_EMAIL
VALUES (id, em);
```

Updating

- Applicant can update an existing email

```
UPDATE CONTACT_EMAIL
SET Email = new_em
WHERE ContactID = id
AND Email = em;
```

Deleting

- Applicants can delete specific emails for a contact, but <u>not</u> all emails at once

```
DELETE FROM CONTACT_EMAIL
WHERE ContactID = id
AND Email = em;
```

<u>Table: CONTACT_PHONE</u>

Attribute	Provided in variable	Can be NULL?	Notes
ContactID	id	N	FK, must reference existing contact
Phone	ph	N	updated phone can be provided in variable new_ph

Creation

- Applicant can add new phones to a contact

```
INSERT INTO CONTACT_PHONE
VALUES (id, ph);
```

Updating

- Applicant can update an existing phone

```
UPDATE CONTACT_PHONE
SET Phone = new_ph
WHERE ContactID = id
AND Phone = ph;
```

Deleting

- Applicants can delete specific phones for a contact, but <u>not</u> all phones at once

```
DELETE FROM CONTACT_PHONE
WHERE ContactID = id
AND Phone = ph;
```

Table: REFERRAL

Attribute	Provided in variable	Can be NULL?	Notes
ContID	contact	N	FK, must reference existing contact; updated contact can be provided in variable new_contact
ReferralID	refID	N	is an auto-incremented primary key
Date	date	Y	
Notes	note	Y	
ApplicantUsername	user	N	FK, must reference existing applicant
CompName	comp	N	FK, must reference existing company
PosName	pos	N	FK, must reference existing job

- When manipulating referrals, applicants should provide a contact's first name (variable fname) and last name (variable lname) to retrieve the contact's ContactID, using the following query:

```
SELECT ContactID
FROM CONTACT
WHERE Fname = fname
AND Lname = lname;
```

Creation

- Applicant can create a new referral, assigning all values (with Date and Notes optionally left NULL) apart from ReferralID, as this is an auto-incremented primary key

```
INSERT INTO REFERRAL(ContID, Date, Notes, ApplicantUsername, CompName, PosName)
VALUES (contact, date, note, user, comp, pos);
```

Updating

- Applicant can update the contact who issued the referral, as well as the referral date, notes, and job that the referral refers to

```
UPDATE REFERRAL
SET ContID = new_contact
WHERE ReferralID = refID
AND ContID = contact;

UPDATE REFERRAL
SET Date = date
WHERE ReferralID = refID
AND ContID = contact;

UPDATE REFERRAL
SET Notes = note
WHERE ReferralID = refID
AND ContID = contact;

UPDATE REFERRAL
SET CompName = comp, PosName = pos
WHERE ReferralID = refID
AND ContID = contact;
```

Deleting

- Applicants can delete specific referrals

```
DELETE FROM REFERRAL
WHERE ReferralID = refID
AND ContID = contact;
```

Table: WORKS_AT

Attribute	Provided in variable	Can be NULL?	Notes
CompName	comp	N	FK, must reference existing company
ContID	cont	N	FK, must reference existing contact
Role	role	Y	

- When manipulating entries, applicants should provide a contact's first name (variable fname) and last name (variable lname) to retrieve the contact's ContactID, using the following query:

```
SELECT ContactID
FROM CONTACT
WHERE Fname = fname
AND Lname = lname;
```

Creation

- Applicant can add new WORKS_AT entries for a specific contact and company

```
INSERT INTO WORKS_AT
VALUES (comp, cont, role);
```

Updating

- Applicant can update an existing role for a contact at a company

```
UPDATE WORKS_AT
SET Role = role
WHERE CompName = comp
AND ContID = cont;
```

Deleting

- Applicants can delete specific roles for a contact, but not all roles at once

```
DELETE FROM WORKS_AT
WHERE CompName = comp
AND ContID = cont;
```

Table: KNOWS

Attribute	Provided in variable	Can be NULL?	Notes
ContID	cont	N	FK, must reference existing contact
ApplicantUsername	user	N	FK, must reference existing applicant
Relationship	relation	Y	
Notes	note	Y	
LastContactDate	date	Y	

- When manipulating entries, applicants should provide a contact's first name (variable fname) and last name (variable lname) to retrieve the contact's ContactID, using the following query:

```
SELECT ContactID
FROM CONTACT
WHERE Fname = fname
AND Lname = lname;
```

Creation

- Applicant can add entries for contacts that they know

```
INSERT INTO KNOWS
```

VALUES (cont, user, relation, note, date);

Updating

- Applicant can update the relationship, notes, and last contact date for a specific contact that they know

```
UPDATE KNOWS
SET Relationship = relation
WHERE ApplicantUsername = user
AND ContID = cont;

UPDATE KNOWS
SET Notes = note
WHERE ApplicantUsername = user
AND ContID = cont;

UPDATE KNOWS
SET LastContactDate = date
WHERE ApplicantUsername = user
AND ContID = cont;
```

Deleting

- Applicants can delete specific entries for contacts that they know, but <u>not</u> all entries at once

```
DELETE FROM KNOWS
WHERE ApplicantUsername = user
AND ContID = cont;
```

<u>Table: ATTENDS</u>

Attribute	Provided in variable	Can be NULL?	Notes
ContID	cont	N	FK, must reference existing contact
ApplicantUsername	user	N	FK, must reference existing applicant
InterID	inter	N	FK, must reference existing interview

- When manipulating entries, applicants should provide a contact's first name (variable fname) and last name (variable lname) to retrieve the contact's ContactID, using the following query:

```
SELECT ContactID
```

```
FROM CONTACT
WHERE Fname = fname
AND Lname = lname;
```

- Applicants should also provide the date and time of the specific interview (variable dayTime) to retrieve the interview's InterID, using the following query:

```
SELECT InterviewID
FROM INTERVIEW
WHERE ApplicantUsername = user
AND DateTIme = daytime;
```

Creation

- Applicant can add entries for contacts that will attend specific interviews

```
INSERT INTO ATTENDS
VALUES (cont, user, inter);
```

Updating

- Since no additional information is tracked about contacts attending interviews, applicants are only able to create and delete entries; thus, no updating statements are possible

Deleting

- Applicants can delete specific contacts attending an interview, or all contacts attending an interview

```
DELETE FROM ATTENDS
WHERE ApplicantUsername = user
AND ContID = cont
AND InterID = inter;

DELETE FROM ATTENDS
WHERE ApplicantUsername = user
AND InterID = inter;
```

Table: MENTIONS

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
InterID	inter	N	FK, must reference existing interview
CompName	comp	N	FK, must reference existing company

PosName	pos	N	FK, must reference existing job

- When manipulating entries, applicants should provide the date and time of the specific interview (variable dayTime) to retrieve the interview's InterID, using the following query:

```
SELECT InterviewID
FROM INTERVIEW
WHERE ApplicantUsername = user
AND DateTIme = daytime;
```

Creation

- Applicant can add entries for positions that are mentioned in specific interviews

```
INSERT INTO MENTIONS
VALUES (user, inter, comp, pos);
```

Updating

- Since no additional information is tracked about jobs being mentioned in interviews, applicants are only able to create and delete entries; thus, no updating statements are possible

Deleting

- Applicants can delete specific jobs mentioned in an interview, but <u>not</u> all jobs mentioned at once, as an interview must be related to and thus mention at least one job

```
DELETE FROM MENTIONS
WHERE ApplicantUsername = user
AND InterID = inter
AND CompName = comp
AND PosName = pos;
```

<u>Table: SUBMIT_WITH</u>

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
DocFile	doc	N	FK, must reference existing document
ApplicationName	apl	N	FK, must reference existing application

Creation

- Applicant can add entries for documents submitted with specific applications

INSERT INTO SUBMIT_WITH
VALUES (user, doc, apl);

Updating

- Since no additional information is tracked about documents being submitted with an application, applicants are only able to create and delete entries; thus, no updating statements are possible

Deleting

- Applicants can delete entries for specific documents mentioned in specific applications, all documents mentioned in a specific application, or all applications that mention a specific document

```
DELETE FROM SUBMIT_WITH
WHERE ApplicantUsername = user
AND DocFile = doc
AND ApplicationName = apl;

DELETE FROM SUBMIT_WITH
WHERE ApplicantUsername = user
AND ApplicationName = apl;

DELETE FROM SUBMIT_WITH
WHERE ApplicantUsername = user
AND DocFile = doc;
```

Table: CORRESPONDS_TO

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
ApplicationName	apl	N	FK, must reference existing application
CompName	comp	N	FK, must reference existing company
PosName	pos	N	FK, must reference existing job
JobPostURL	url	Y	

Creation

- Applicant can add entries for jobs that correspond to specific applications

```
INSERT INTO CORRESPONDS_TO
```

VALUES (user, apl, comp, pos, url);

Updating

- Applicant can update the job post URL for a specific job mentioned in a specific application, or for a specific job mentioned in all the applicant's applications

```
UPDATE CORRESPONDS_TO
SET JobPostURL = url
WHERE ApplicantUsername = user
AND ApplicationName = apl
AND CompName = comp
AND PosName = pos;

UPDATE CORRESPONDS_TO
SET JobPostURL = url
WHERE ApplicantUsername = user
AND CompName = comp
AND PosName = pos;
```

Deleting

- Applicants can delete specific jobs corresponding to an application, but <u>not</u> all jobs mentioned at once, as an application must correspond to and thus mention at least one job

```
DELETE FROM CORRESPONDS_TO
WHERE ApplicantUsername = user
AND ApplicationName = apl
AND CompName = comp
AND PosName = pos;
```

<u>Table: TRACKS</u>

Attribute	Provided in variable	Can be NULL?	Notes
ApplicantUsername	user	N	FK, must reference existing applicant
CompName	comp	N	FK, must reference existing company
PosName	pos	N	FK, must reference existing job
Notes	note	Y	
DateToApply	date	Y	

Creation

- Applicant can add specific jobs that they want to track

```
INSERT INTO TRACKS
VALUES (user, comp, pos, note, date);
```

Updating

- Applicant can update notes and date to apply for a specific job that they track

```
UPDATE TRACKS
SET Notes = note
WHERE ApplicantUsername = user
AND CompName = comp
AND PosName = pos;

UPDATE TRACKS
SET DateToApply = date
WHERE ApplicantUsername = user
AND CompName = comp
AND PosName = pos;
```

Deleting

- Applicants can delete entries for specific jobs that they track

```
DELETE FROM TRACKS
WHERE ApplicantUsername = user
AND CompName = comp
AND PosName = pos;
```