

TrackMyJobApply Database

Mumin Jia

U73438924

Table of Contents

Introduction	3
TrackMyJobApply Use Cases	4
Business Rules for My Database	5
TrackMyJobApply Transaction.....	5
TrackMyJobApply History	10
TrackMyJobApply Question and Query	16
TrackMyJobApplication Summary.....	23

TrackMyJobApply Database



Introduction

In my opinion, it is necessary to develop an application that can track the status of job applications. With a highly competitive society, many people submit their resumes to hundreds of companies in order to find a job. In view of the current situation, candidates can learn about their application status mainly through two ways: receiving an email from the company that tells them whether to enter the next round of interviews, and logging in to the recruitment interface of each company to check the status. However, the time for applicants to submit resumes is different. Even if the application time is the same, the time for the company to reply or update the application status is different. Due to the large number of applied companies and the long period of job apply, applicants may not be able to fully remember the application status of all companies. It is especially important to develop a website that can check the status of all job applications to help applicants have real-time information. This website is called “TrackMyJobApply”.

Take me as an example. In order to be a summer intern in 2020, I began to submit resumes to various companies since last November. The positions to be applied are mainly business analysts and data analysts, as well as a few financial analysts. The company's fields are also diverse, including technology, finance, real estate, consulting and so on. A month later, I have forgotten which companies I applied for, let alone know the application status of each company. However, TrackMyJobApply's website helps applicants update their application status in real time. The website synchronizes the information after the applicant has applied for a job. After a while, I hope to check the status of previous applications. I just need to log in to

TrackMyJobApply's website, instead of logging in to the recruitment official website of each company or checking my own historical email.

TrackMyJobApply's website has a database that collects information about users applying for jobs. The database will store the job name, the application time, and the real-time status. At present, the database may not contain enough information. But as I learn more, I will further improve of the database.

TrackMyJobApply Use Cases

♣ *Creating an Account Use Case*

The first use case about this database is for users to create an account on the website.

1. The user first opens the website page.
2. The user clicks "Need an Account" or "Sign in"
3. When creating an account, users need to enter their name, phone number, email address and set a password.
4. Install Internet tracking software so that the system can automatically track pages when users apply for jobs.

♣ *Tracking Job Application Use Case*

In the second use case, the website tracks the job application of users and extract important information into the database.

1. Open network tracking software.
2. Click on the website of job that you want to apply for.
3. The website automatically tracks the pages where users apply for jobs and extracts important data.

♣ *Checking Application Use Case*

The core purpose of the website to integrate job application information is to provide convenience for users to check the status later.

1. User opens website and logs into created account.
2. The user enters a keyword or selects the range of information.
3. Click on "search".
4. The website will display all historical application records that the user needs.
5. If the user wants to focus on some application status, click "Favorite".
6. If the user wants to share the application link to others, click "Share".

Business Rules for My Database

- Each apply is associated with one account.
Each account may be associated with many applications.
- Each apply must be associated with a job.
A job is associated with one to many applications.
- Each apply must be associated with one company.
A company must be associated with many applications.
- Each job is associated with one company.
One company must have one or many job vacancies.
- An account is a free account or a paid account.
- A job is an internship, full-time job, and part-time job.
- Each job must have one address. One address may have many jobs.

TrackMyJobApply Transaction

I will create four stored procedures for my database.

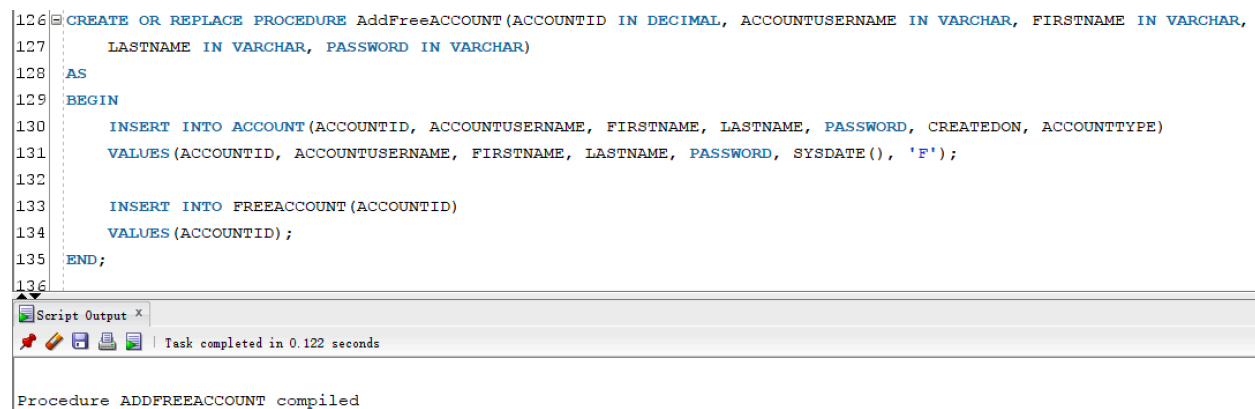
- ♣ Create a free account
- ♣ Add information about technology company
- ♣ Add address
- ♣ Add job

Use Case #1. Creating an Account

5. The user first opens the website page.
6. The user clicks "Need an Account" or "Sign in"
7. When creating an account, users need to enter their name, phone number, email address and set a password.
8. Install Internet tracking software so that the system can automatically track pages when users apply for jobs.

For this use case, I will implement a transaction that **creates a free account** by using Oracle.

Here is a screenshot of my stored procedure definition.

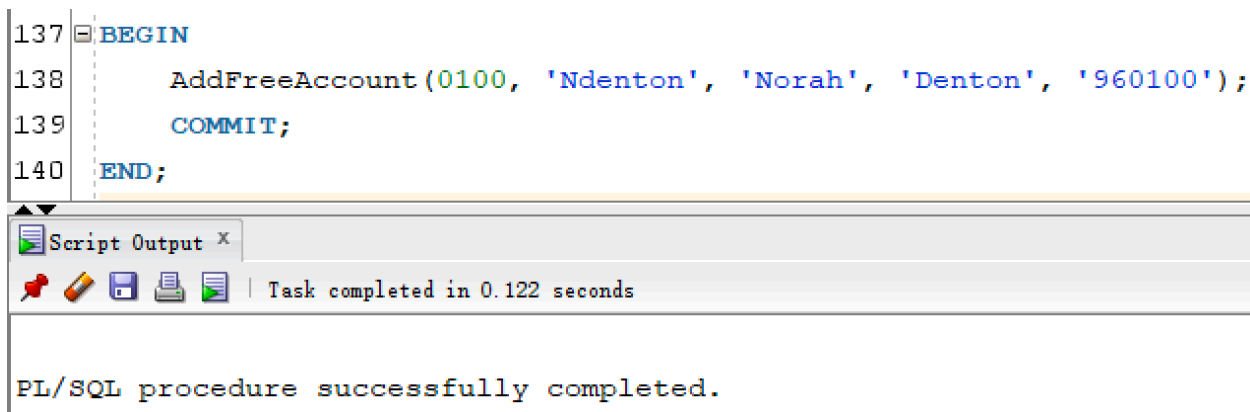


```
126 CREATE OR REPLACE PROCEDURE AddFreeACCOUNT (ACCOUNTID IN DECIMAL, ACCOUNTUSERNAME IN VARCHAR, FIRSTNAME IN VARCHAR,
127 LASTNAME IN VARCHAR, PASSWORD IN VARCHAR)
128 AS
129 BEGIN
130     INSERT INTO ACCOUNT (ACCOUNTID, ACCOUNTUSERNAME, FIRSTNAME, LASTNAME, PASSWORD, CREATEDON, ACCOUNTTYPE)
131     VALUES (ACCOUNTID, ACCOUNTUSERNAME, FIRSTNAME, LASTNAME, PASSWORD, SYSDATE(), 'F');
132
133     INSERT INTO FREEACCOUNT (ACCOUNTID)
134     VALUES (ACCOUNTID);
135 END;
136
```

Script Output x
Task completed in 0.122 seconds
Procedure ADDFREEACCOUNT compiled

I named the store procedure “AddFreeACCOUNT”, and give it parameters that correspond to the Account and FreeAccount tables. Since “CreatedOn” is always the current date, I do not need a parameter for that. I use the SYSDATE() function in Oracle. In addition, this procedure is always for a free account, I do not use a parameter for AccountType, but hardcode the character “F”.

Here is a screenshot of my stored procedure execution.



The screenshot shows a SQL script editor with the following code:

```
137 BEGIN
138     AddFreeAccount(0100, 'Ndenton', 'Norah', 'Denton', '960100');
139     COMMIT;
140 END;
```

Below the script editor is a 'Script Output' window. It contains the text: 'Task completed in 0.122 seconds' and 'PL/SQL procedure successfully completed.'

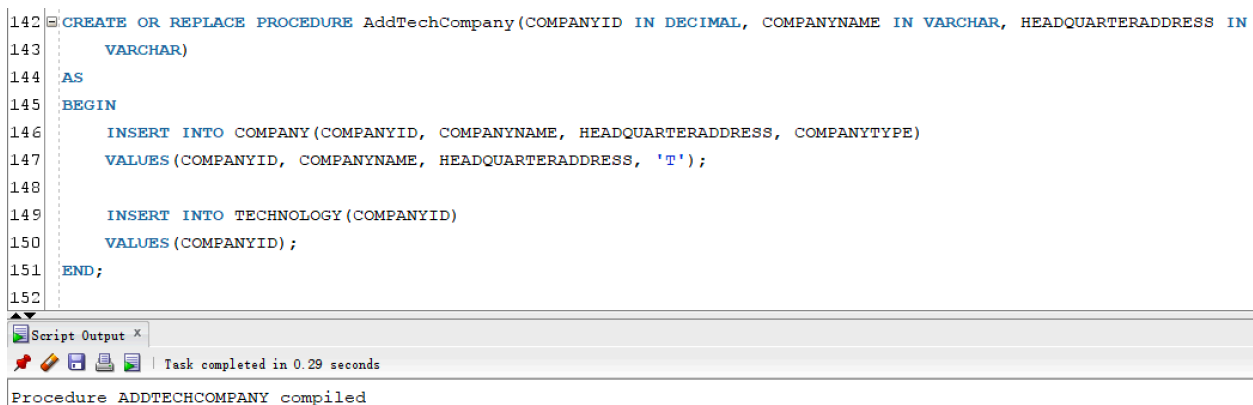
I add a person named Norah Denton with other information.

Use Case #2. Tracking Job Application Use Case

4. Open network tracking software.
5. Click on the website of job that you want to apply for.
6. The website automatically tracks the pages where users apply for jobs and extracts important data.

For this use case, I will implement a transaction that **adds information about technology company** by using Oracle.

Here is a screenshot of my stored procedure definition.



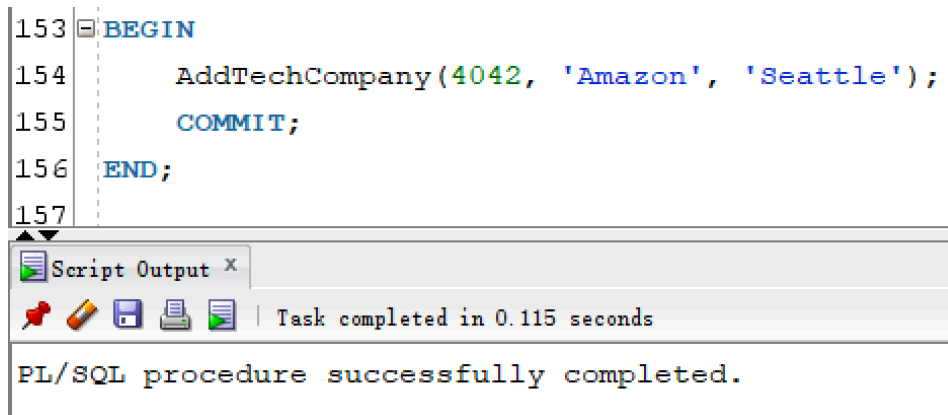
The screenshot shows a SQL script editor with the following code:

```
142 CREATE OR REPLACE PROCEDURE AddTechCompany(COMPANYID IN DECIMAL, COMPANYNAME IN VARCHAR, HEADQUARTERADDRESS IN
143     VARCHAR)
144 AS
145 BEGIN
146     INSERT INTO COMPANY(COMPANYID, COMPANYNAME, HEADQUARTERADDRESS, COMPANYTYPE)
147     VALUES (COMPANYID, COMPANYNAME, HEADQUARTERADDRESS, 'T');
148
149     INSERT INTO TECHNOLOGY(COMPANYID)
150     VALUES (COMPANYID);
151 END;
152
```

Below the script editor is a 'Script Output' window. It contains the text: 'Task completed in 0.29 seconds' and 'Procedure ADDTECHCOMPANY compiled'.

I name the stored procedure “AddTechCompany”, and give it parameters that correspond to the Company and Technology tables. Since this procedure is always for the technology company, I do not use a parameter for CompanyType, but hardcode the character “T”.

Here is a screenshot of my stored procedure execution.



```
153 BEGIN
154     AddTechCompany(4042, 'Amazon', 'Seattle');
155     COMMIT;
156 END;
157
```

Script Output x

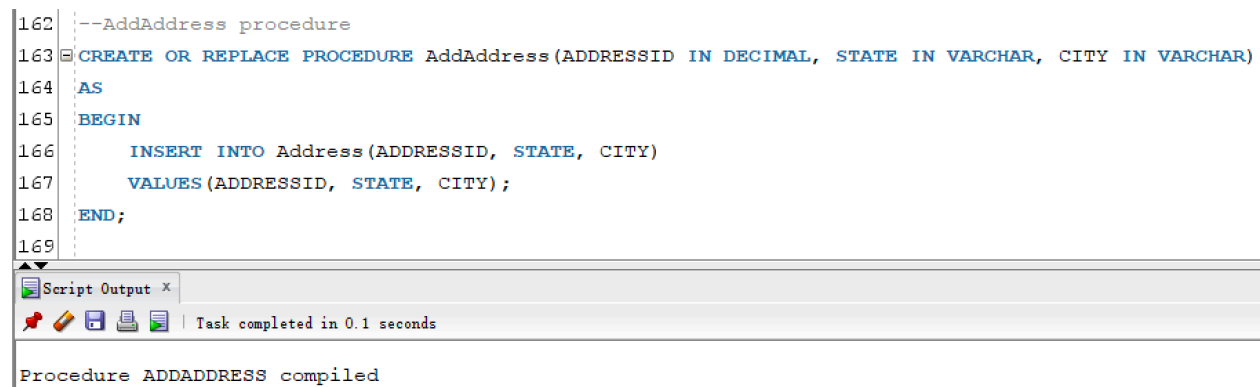
Task completed in 0.115 seconds

PL/SQL procedure successfully completed.

I add a company named Amazon and its ID is 4042. Amazon’s headquarter is located in Seattle. Obviously, Amazon is a technology company. I nested the stored procedure call between transaction control statements to ensure the transaction is committed.

Moreover, I will **add address information** for this use case by using Oracle.

Here is a screenshot of my stored procedure definition.



```
162 --AddAddress procedure
163 CREATE OR REPLACE PROCEDURE AddAddress (ADDRESSID IN DECIMAL, STATE IN VARCHAR, CITY IN VARCHAR)
164 AS
165 BEGIN
166     INSERT INTO Address (ADDRESSID, STATE, CITY)
167     VALUES (ADDRESSID, STATE, CITY);
168 END;
169
```

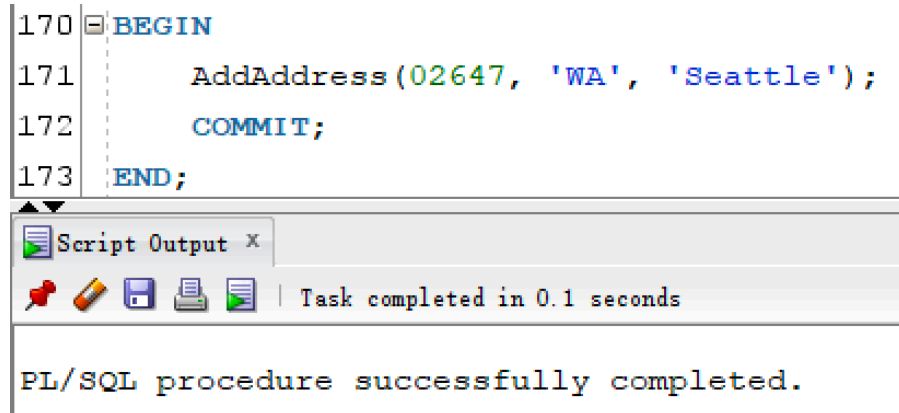
Script Output x

Task completed in 0.1 seconds

Procedure ADDADDRESS compiled

I named the store procedure “AddAddress” , and give it parameters that correspond to the Address table.

Here is a screenshot of my stored procedure execution.



```
170 BEGIN
171     AddAddress(02647, 'WA', 'Seattle');
172     COMMIT;
173 END;
```

Script Output x

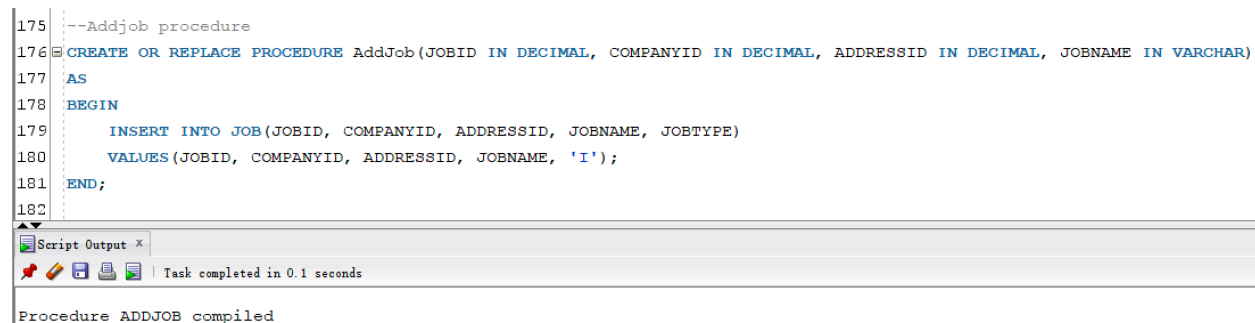
Task completed in 0.1 seconds

PL/SQL procedure successfully completed.

I added an address. The Address ID is 02467. Seattle is located in Washington State. I nested the stored procedure call between transaction control statements to ensure the transaction is committed.

After address and company information, I will also **add job information**.

Here is a screenshot of my stored procedure definition.



```
175 --Addjob procedure
176 CREATE OR REPLACE PROCEDURE AddJob(JOBID IN DECIMAL, COMPANYID IN DECIMAL, ADDRESSID IN DECIMAL, JOBNAME IN VARCHAR)
177 AS
178 BEGIN
179     INSERT INTO JOB(JOBID, COMPANYID, ADDRESSID, JOBNAME, JOBTYP)
180     VALUES(JOBID, COMPANYID, ADDRESSID, JOBNAME, 'I');
181 END;
182
```

Script Output x

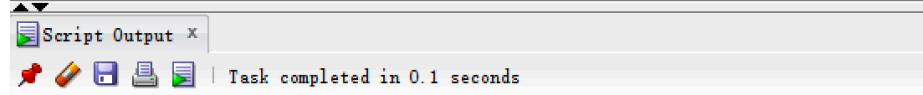
Task completed in 0.1 seconds

Procedure ADDJOB compiled

I named the store procedure “AddJob” , and give it parameters that correspond to the Job table. Since this procedure is always for the internship, I do not use a parameter for JobType, but hardcode the character “I”.

Here is a screenshot of my stored procedure execution.

```
183 BEGIN
184     AddJob(23890145, 4042, 02647, 'Software Engineer');
185     COMMIT;
186 END;
187
```



Script Output x
Task completed in 0.1 seconds

Procedure ADDAPPLICATION compiled

I added a job. This job belongs to Amazon and is located in Seattle. As a result, the Company ID is the ID of Amazon and the Address ID is the ID of Seattle. I nested the stored procedure call between transaction control statements to ensure the transaction is committed.

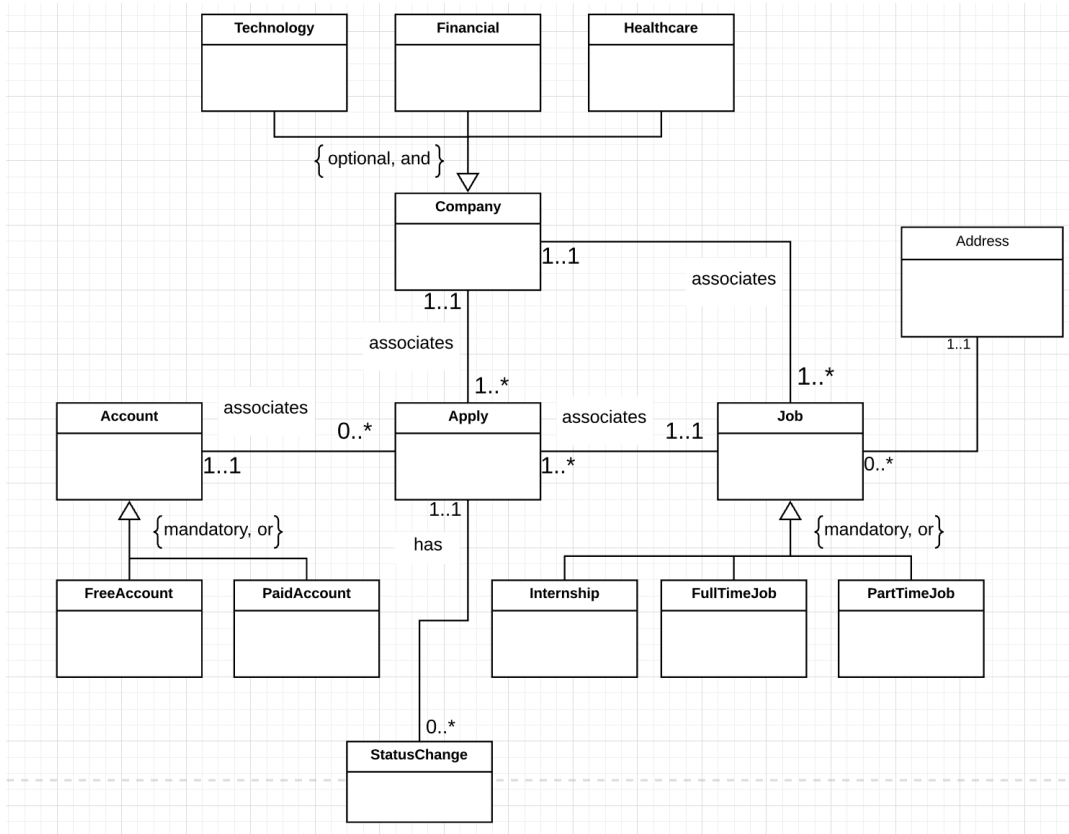
TrackMyJobApply History

In reviewing my DBMS physical ERD, one piece of data that would obviously benefit from a historical record application status in the Apply Table. Such a history would help me understand the progress of the job application and arrange a reasonable time for preparing for the interview later. Viewing the progress of job applications is also the biggest purpose of creating this database. I will capture history for my project with following aspects.

1. Updated conceptual ERD
2. Updated DBMS physical ERD
3. Table Creation
4. Trigger Creation and explanation
5. Ensure an application existed
6. Update status a couple of times
7. Verify the StatusChange Table

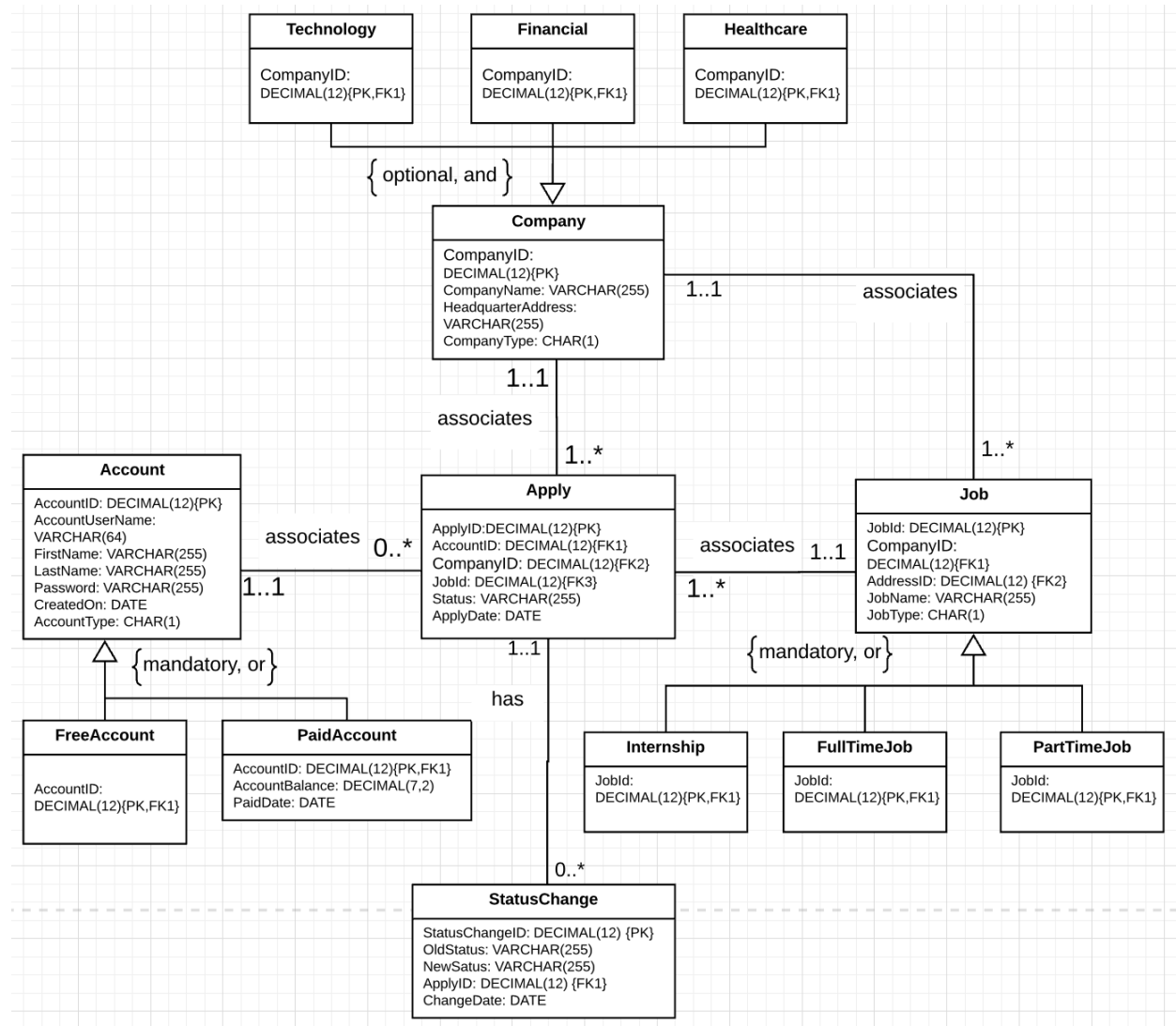
♣ Updated conceptual ERD

My new structural database rule is: Each application may have many status changes; each change is just for one application.



I added the StatusChange entity and related it to Apply.

♣ Updated DBMS physical ERD



The StatusChange entity is present and linked to Apply. Below are the attributes I added and why.

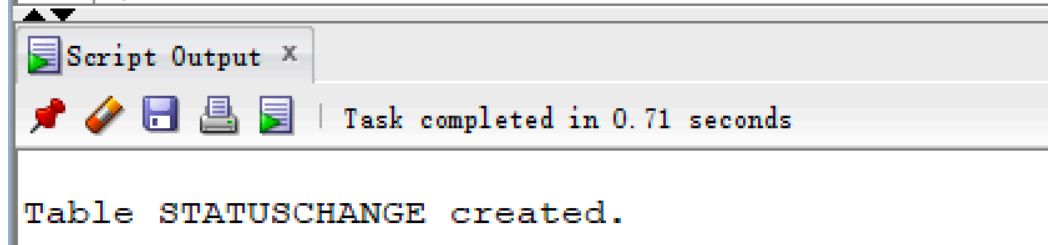
Attribute	Description
StatusChangeID	This is the primary key of the history table. It is DECIMAL(12) to allow for many values.
OldStatus	This is the status of the application before the change. The datatype mirrors the Status datatype in the Apply Table.
NewStatus	This is the status of the application after the change. The datatype mirrors the Status datatype in the Apply Table.
ApplyID	This is a foreign key to the Apply Table, a reference to the account that had the change in status.

ChangeDate	This is the date the status changed occurred, with a DATE datatype.
-------------------	---

♣ Table creation

Here is a screenshot of my table creation, which has all of the same attributes and datatypes as indicated in the DBMS physical ERD.

```
202  --Create StatusChange Table
203  CREATE TABLE STATUSCHANGE (
204  STATUSCHANGEID DECIMAL(12) NOT NULL,
205  OLDSTATUS VARCHAR(255) NOT NULL,
206  NEWSTATUS VARCHAR(255) NOT NULL,
207  APPLYID DECIMAL(12) NOT NULL,
208  CHANGEDATE DATE NOT NULL,
209  PRIMARY KEY (STATUSCHANGEID),
210  FOREIGN KEY (APPLYID) REFERENCES APPLY);
211
```



The screenshot shows a database script execution window. The script defines a table named STATUSCHANGE with columns STATUSCHANGEID, OLDSTATUS, NEWSTATUS, APPLYID, and CHANGEDATE. It includes a primary key on STATUSCHANGEID and a foreign key on APPLYID. The script is executed, and the output shows 'Table STATUSCHANGE created.' The window also displays a status bar indicating 'Task completed in 0.71 seconds'.

♣ Trigger creation and Explanation

Here is a screenshot of my trigger creation which will maintain the StatusChange Table.

```

212 --Create Status Change Trigger
213 CREATE OR REPLACE TRIGGER StatusChangeTrigger
214 BEFORE UPDATE OF Status ON APPLY
215 FOR EACH ROW
216 BEGIN
217     INSERT INTO statuschange (STATUSCHANGEID, OLDSTATUS, NEWSTATUS, APPLYID, CHANGEDATE)
218     VALUES (NVL((SELECT MAX(StatusChangeID)+1 FROM StatusChange), 1),
219             :OLD.STATUS,
220             :NEW.STATUS,
221             :New.ApplyID,
222             trunc(sysdate));
223 END;

```

Script Output x

Task completed in 0.863 seconds

Trigger STATUSCHANGETRIGGER compiled

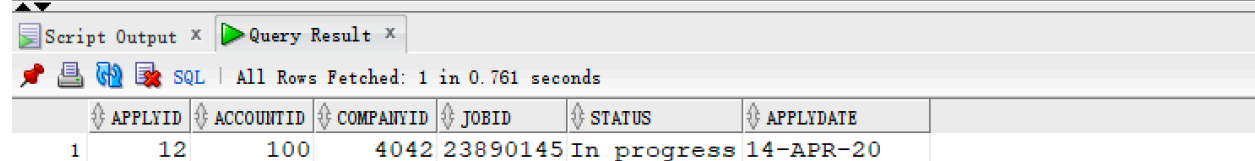
Let's work through it line by line.

CODE	DESCRIPTION
CREATE OR REPLACE TRIGGER StatusChangeTrigger BEFORE UPDATE OF Status ON APPLY	This names the trigger "StatusChangeTrigger" and links it to the apply table. Further, it specifically indicates that the trigger will run before an update on Apply Table, and it only runs if the Status column is updated.
FOR EACH ROW BEGIN	This indicates that the trigger should run for every row that is updated, which necessary to get access to the specific status that changed.
INSERT INTO statuschange(STATUSCHANGEID, OLDSTATUS, NEWSTATUS, APPLYID, CHANGEDATE) VALUES(NVL((SELECT MAX(StatusChangeID)+1 FROM StatusChange), 1), :OLD.STATUS, :NEW.STATUS, :New.ApplyID, trunc(sysdate));	This is the insert statement that records the status change by adding row into the StatusChange table, the statusChanegID column is generated by getting the maximum StatusChangeID + 1 if one exists, or 1 otherwise. The old and new status are accessed through the :NEW and :OLD pseudo tables provided in PL/SQL triggers. The ApplyID IS EXTRACTED FROM :NEW pseudo table. The built-in variable sysdate obtains the current date, and the built-in function trunc() removes the time.
END;	This ends the trigger definition.

♣ Ensure an application existed

To test out that the trigger works, I created a row in the Apply table with a ApplyID of 12, and an initial status of “In Progress”, resulting in this Apply row.

```
225 INSERT INTO APPLY (APPLYID, ACCOUNTID, COMPANYID, JOBID, STATUS, APPLYDATE)
226 VALUES (12, 100, 4042, 23890145, 'In progress', CAST('14-APRIL-2020' AS DATE));
227
228 SELECT*
229 FROM APPLY;
```

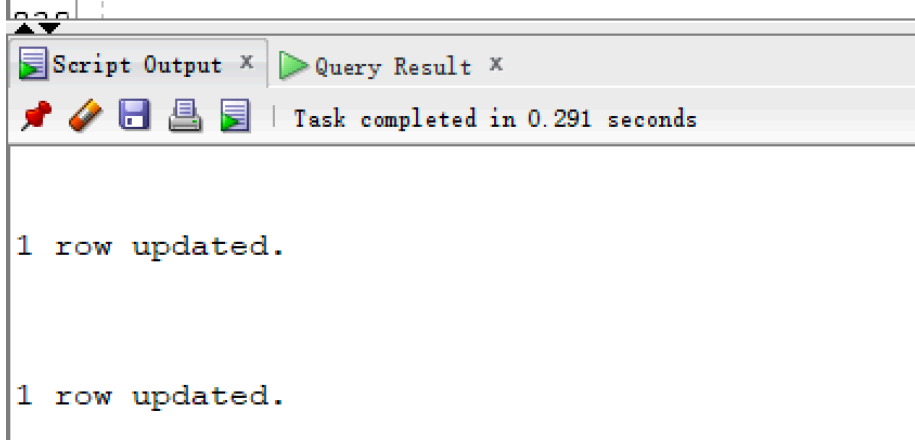


APPLYID	ACCOUNTID	COMPANYID	JOBID	STATUS	APPLYDATE
12	100	4042	23890145	In progress	14-APR-20

♣ Update status a couple of times

I then change the status of this application two times, as demonstrated below.

```
231 UPDATE APPLY
232 SET STATUS = 'Interview'
233 WHERE APPLYID = 12;
234
235 UPDATE APPLY
236 SET STATUS = 'No longer consideration'
237 WHERE APPLYID = 12;
238
```



```
Script Output x Query Result x
Task completed in 0.291 seconds

1 row updated.

1 row updated.
```

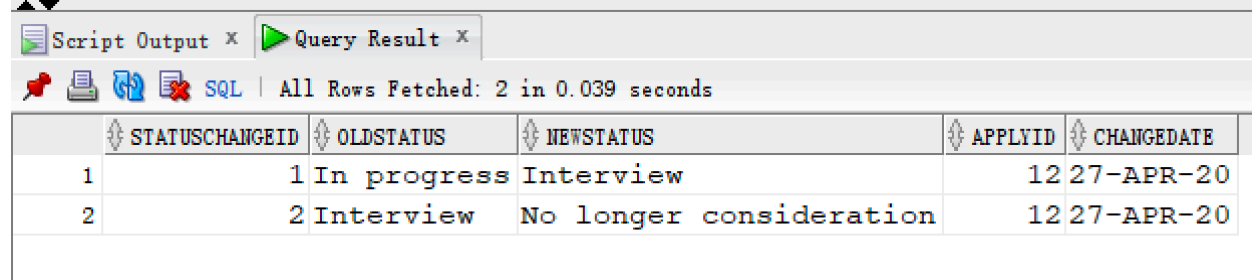
♣ Verify the StatusChange Table

Finally, I verify that my trigger worked as expected by selecting from StatusChange table, illustrated below.

```

239 | SELECT *
240 | FROM STATUSCHANGE;

```



	STATUSCHANGEID	OLDSTATUS	NEWSTATUS	APPLYID	CHANGEDATE
1	1	In progress	Interview	12	27-APR-20
2	2	Interview	No longer consideration	12	27-APR-20

The results demonstrate that the status went from “In progress” to “Interview”, then to “No longer consideration”, all for the application with ApplyID 12. The old and new balances are now tracked with a trigger and a history table.

TrackMyJobApply Question and Query

Question #1.

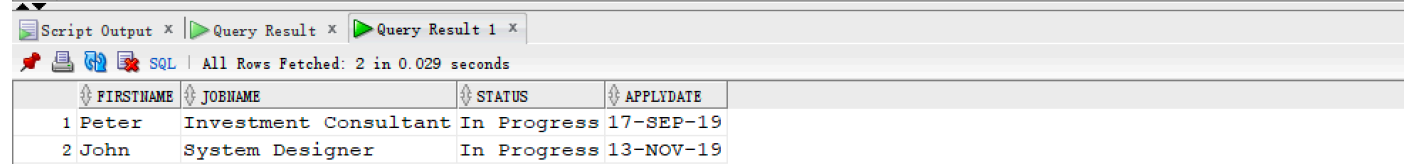
How many applications were submitted more than three months ago, but the status is still in progress? We also need to view the applicants' first name and the job name.

I explain why this question is useful. The answer can be used to learn about the application status. Depending upon the number of applicants, company sometimes could not contact with each applicant. They only contact with applicants whose qualifications match needs for the role. In other words, if there is no response from the company three months after submitting the application, it is

likely that we do not match the position. In view of this, applicants can apply for more jobs instead of waiting for replies of these job applications.

Here is a screenshot of the query I use.

```
361 --Question#1.
362 --This query answers this question:
363 --How many applications were submitted more than three months ago, but the status is still in progress?
364 --We also need to view the applicants' first name and the job name.
365 SELECT ACCOUNT.FIRSTNAME,
366        JOB.JOBNAME,
367        APPLY.STATUS,
368        APPLY.APPLYDATE
369 FROM APPLY
370 JOIN ACCOUNT ON ACCOUNT.ACCOUNTID = APPLY.ACCOUNTID
371 JOIN JOB ON JOB.JOBID = APPLY.JOBID
372 WHERE APPLY.APPLYID IN
373        (SELECT APPLY.APPLYID
374         FROM APPLY
375         WHERE STATUS = 'In Progress' AND APPLYDATE < CAST('27-JAN-2020' AS DATE));
376
```



The screenshot shows a SQL query execution interface. The query text is displayed in a text area, and the results are shown in a table below. The table has four columns: FIRSTNAME, JOBNAME, STATUS, and APPLYDATE. There are two rows of data: one for Peter, Investment Consultant, In Progress, 17-SEP-19, and one for John, System Designer, In Progress, 13-NOV-19. The interface also shows a status bar indicating 'All Rows Fetched: 2 in 0.029 seconds'.

FIRSTNAME	JOBNAME	STATUS	APPLYDATE
1 Peter	Investment Consultant	In Progress	17-SEP-19
2 John	System Designer	In Progress	13-NOV-19

To get the result, I join the Apply to the Account and Job table, limit the results with “In Progress” status. I also limit the Apply date more than three months ago. The result is that two applications are still with “In Progress” status more than three years.

To help prove that the query is working properly, I show the full contents of the Apply table with Job and Account Table.

```

377 SELECT ACCOUNT.FIRSTNAME,
378        JOB.JOBNAME,
379        APPLY.STATUS,
380        APPLY.APPLYDATE
381 FROM APPLY
382 JOIN ACCOUNT ON ACCOUNT.ACCOUNTID = APPLY.ACCOUNTID
383 JOIN JOB ON JOB.JOBID = APPLY.JOBID;
384

```

Script Output x Query Result x Query Result 1 x

SQL | All Rows Fetched: 9 in 0.083 seconds

	FIRSTNAME	JOBNAME	STATUS	APPLYDATE
1	Norah	Software Engineer	No longer consideration	14-APR-20
2	Mary	Software Engineer	Interview	04-APR-20
3	Mary	Data Scientist	No longer consideration	22-FEB-20
4	Hilary	Data Scientist	In Progress	23-MAR-20
5	Hilary	Data Scientist	In Progress	26-MAR-20
6	Stanton	Software Engineer	In Progress	25-APR-20
7	John	System Designer	In Progress	13-NOV-19
8	John	Data Analyst	Interview	02-JAN-20
9	Peter	Investment Consultant	In Progress	17-SEP-19

Upon inspection, you see that there are 9 applications in my database. Only five of them were with “In Progress” status. Then, we need to find applications with “In Progress” more than three months. However, Hilary applied for two jobs in March 2020, so it is excluded. Stanton applied one job in April 2020, so it is also excluded. John applied one job on Jan 20th, 2020, so it is also excluded. Last, you can only two applications are included. One is that John applied for System Designer job on November 13th, 2019. The other is that Peter applied for Investment Consultant job on September 17th, 2019. So as is demonstrated, the query appears to be returning the correct results based upon the question.

Question #2.

How many applicants are there for each job? We also need to see the number of applicants for each job, the job name and the company name.

I explain why this question is useful. The result of this query could help us find the popularity of each job by counting the number of applicants for each job. For the most popular work, applicants should improve themselves more, fully demonstrate their strength, and also have a good attitude. If you are not admitted, the applicant should not be frustrated.

```

385 --Question2.
386 --How many applicants are there for each job?
387 --We also need to see the number of applicants for each job,
388 --the job name and the company name.
389 SELECT
390     JOB.JOBNAME,
391     COMPANY.COMPANYNAME,
392     COUNT(*) AS NUM
393 FROM APPLY
394 JOIN JOB ON JOB.JOBID = APPLY.JOBID
395 JOIN COMPANY ON COMPANY.COMPANYID = APPLY.COMPANYID
396 GROUP BY JOB.JOBNAME, COMPANY.COMPANYNAME
397 ORDER BY NUM;

```

Script Output x

Query Result x

SQL | All Rows Fetched: 5 in 0.658 seconds

	JOBNAME	COMPANYNAME	NUM
1	Data Analyst	Apple	1
2	Investment Consultant	PWC	1
3	System Designer	Uber Technology	1
4	Data Scientist	Apple	3
5	Software Engineer	Amazon	3

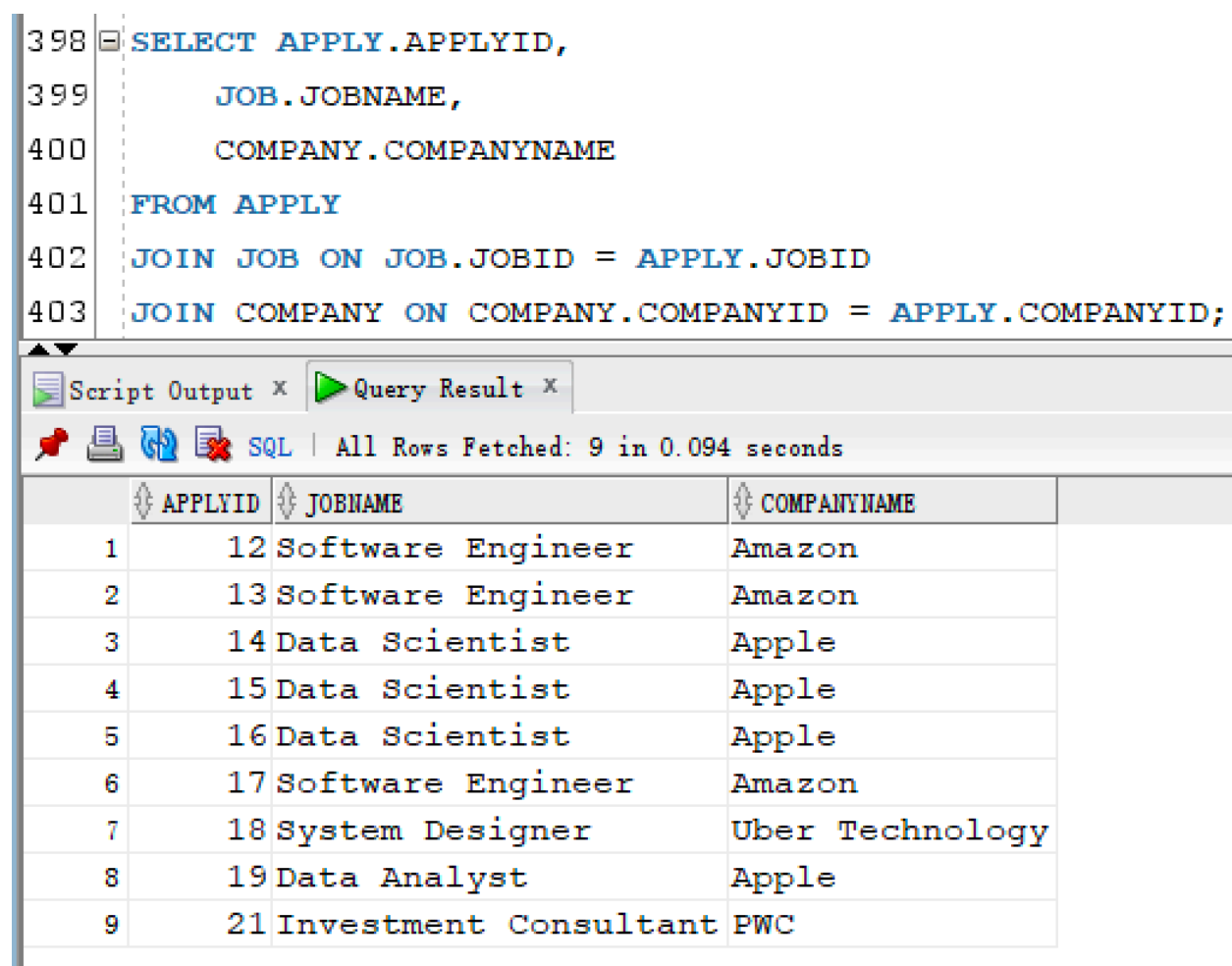
To get the result, I join the Apply to the Job and Company tables. I also order the results by the column “NUM”, which means we could see the ranking of jobs with increasing number of applicants. The result of query is that more people apply for the job Data Scientist and job Software Engineer. Therefore, applicants must have more advantages in order to be admitted.

To help prove that the query is working properly, I show the full contents of the Apply table with Job and Company Table.

```

398 SELECT APPLY.APPLYID,
399     JOB.JOBNAME,
400     COMPANY.COMPANYNAME
401 FROM APPLY
402 JOIN JOB ON JOB.JOBID = APPLY.JOBID
403 JOIN COMPANY ON COMPANY.COMPANYID = APPLY.COMPANYID;

```



	APPLYID	JOBNAME	COMPANYNAME
1	12	Software Engineer	Amazon
2	13	Software Engineer	Amazon
3	14	Data Scientist	Apple
4	15	Data Scientist	Apple
5	16	Data Scientist	Apple
6	17	Software Engineer	Amazon
7	18	System Designer	Uber Technology
8	19	Data Analyst	Apple
9	21	Investment Consultant	PWC

Upon inspection, you see that there are 9 applications in my database. There are three applications for jobs in Amazon company and four applications for jobs in Apple company. The number of applications for each job is the same with the

result of query that I mentioned before. The number of people who applied for PWC company is only one. There is only one person who applied for the job in the Uber Technology. So as is demonstrated, the query appears to be returning the correct results based upon the question.

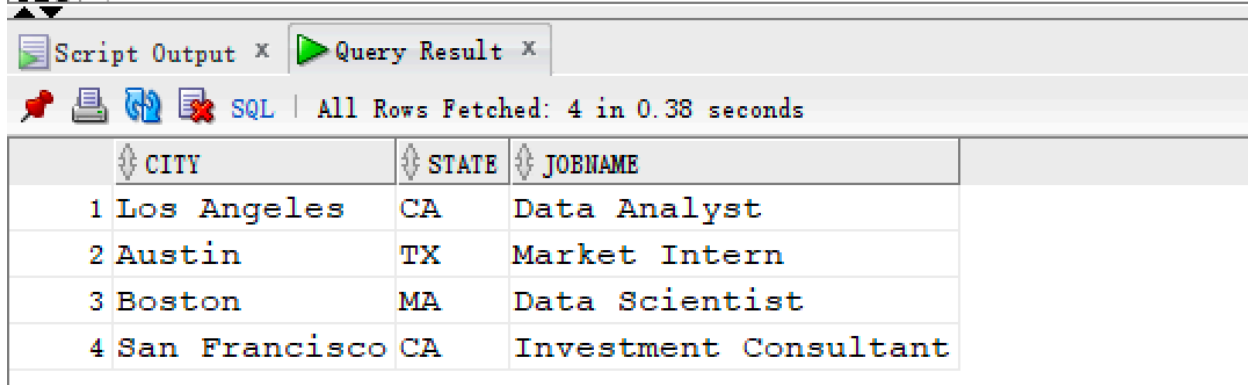
Question #3.

Which city only has one job that can be applied for? We also need to see the city name and the job name.

I explain why this question is useful. The question is which city only has one job that can be applied for. According to the information I added before, some cities only has one job that can be applied for. Due to the limited information I added, I will formulate the question here as to which cities have only one job being recruited. In fact, there must be more than one job in every city. We can expand this question to which cities do not have more than 2,000 jobs being recruited.

What is the use of understanding how many jobs are recruited in each city? We can know which cities have larger or smaller demand for what jobs. This can help us clarify which city to develop further in the future. For example, there are more jobs related to finance in New York and more jobs related to technology in Seattle. If you are a finance student, why not consider future development on the East Coast? After all, there is more demand for financial talent.

```
406 --Question#3.
407 --Which city only has one job that can be applied for?
408 --We also need to see the city name and the job name.
409 SELECT ADDRESS.CITY,
410         ADDRESS.STATE,
411         JOB.JOBNAME
412 FROM ADDRESS
413 JOIN JOB ON JOB.ADDRESSID = ADDRESS.ADDRESSID
414 WHERE ADDRESS.ADDRESSID IN
415     (SELECT ADDRESS.ADDRESSID
416     FROM ADDRESS
417     JOIN JOB ON JOB.ADDRESSID = ADDRESS.ADDRESSID
418     GROUP BY ADDRESS.ADDRESSID
419     HAVING COUNT(JOB.ADDRESSID) = 1);
420
```

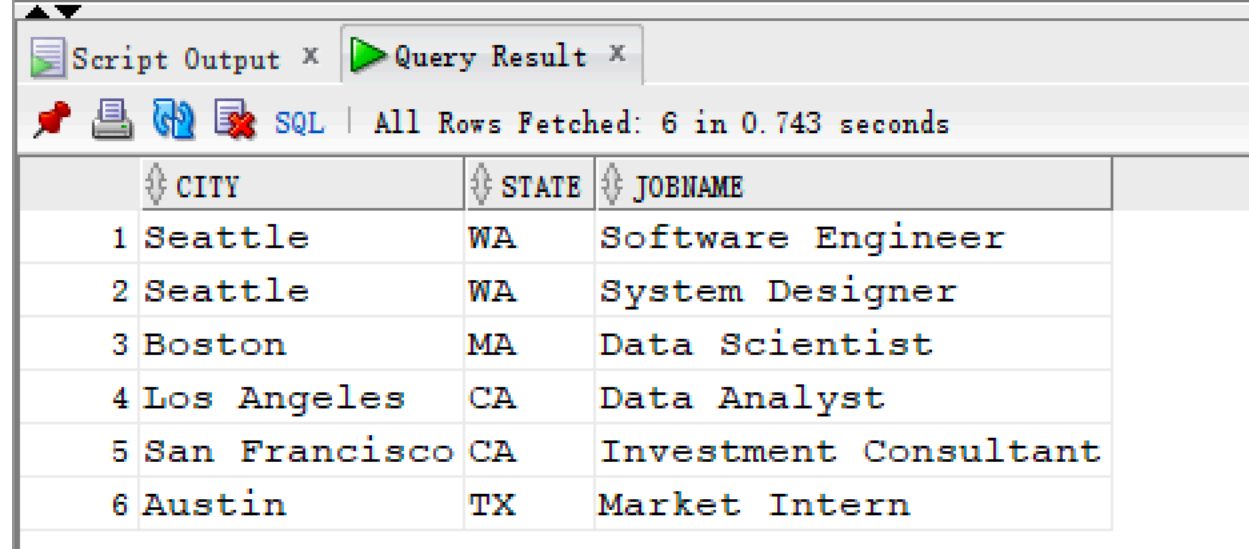


The screenshot shows a SQL query execution interface. At the top, there's a script editor with a query. Below it, a toolbar contains icons for saving, printing, and refreshing, along with a status bar indicating 'All Rows Fetched: 4 in 0.38 seconds'. The main area displays a table with 4 rows and 4 columns: CITY, STATE, and JOBNAME. The first column is an implicit index.

	CITY	STATE	JOBNAME
1	Los Angeles	CA	Data Analyst
2	Austin	TX	Market Intern
3	Boston	MA	Data Scientist
4	San Francisco	CA	Investment Consultant

To help prove that the query is working properly, I show the full contents of the Job and Address Table.

```
424 SELECT ADDRESS.CITY,  
425         ADDRESS.STATE,  
426         JOB.JOBNAME  
427 FROM ADDRESS  
428 JOIN JOB ON JOB.ADDRESSID = ADDRESS.ADDRESSID;
```



	CITY	STATE	JOBNAME
1	Seattle	WA	Software Engineer
2	Seattle	WA	System Designer
3	Boston	MA	Data Scientist
4	Los Angeles	CA	Data Analyst
5	San Francisco	CA	Investment Consultant
6	Austin	TX	Market Intern

With the exception of Seattle, every city has only one job. The Software Engineer job and the System Designer job are both located in Seattle. Other four cities (Boston, Los Angeles, San Francisco, and Austin) only has one job that can be applied for. So as is demonstrated, the query appears to be returning the correct results based upon the question.

TrackMyJobApplication Summary

Having a database to record the status of job applications is useful. After creating the account and install internet tracking software, the users apply for jobs of many companies and the relevant data is stored in the database.

TrackMyJobApply's database can provide a single interface for all applications, which is very convenient for users to check status of job applications.

As for, you can see an updated conceptual ERD and an updated DBMS physical ERD. They contain the important entities of Account, Apply, Job, and Company, as well as relationships between them. The design contains a hierarchy of Account/Free Account and Account/Paid Account to reflect two types of accounts people create. The design also contains a hierarchy of Job/Internship, Job/Full Time, and Job/Part Time to reflect three kinds of job that could be applied for. A hierarchy of Company is also contained in my database. The DBMS physical ERD contains the same entities and relationships, uses the best practice of synthetic keys, and contain the important attributes needed by the database to support the application.

The SQL script that contains all table creations that follows the specification from the DBMS physical ERD exactly. Some questions useful to TrackMyJobApplication have been identified, and implemented with SQL queries. A history of status has been created as well as a query which tracks changes to status for a specific month.

In conclusion, TrackMyJobApplication is such a useful database. I really hope to make this a real Android or iPhone application. I can't help applying this database to practice to help more people who are looking for jobs.