ASSIGNMENT

DATABASE DESIGN & DEVELOPMENT

ISCG6423

Submitted by: Vijay Kumar
Submitted on: 5/11/23

Table of Contents

Task 1 - Introduction:

The aim of this project is to provide a robust database management system for the Looking Glass Recruitment. The company operates as a human resources agency. The current informal paper-based system frequently encounters issues when it comes to aligning a candidate's skills and qualifications with those needed for a specific job opening, as well as maintaining precise records of an employer's vacancies.

With a growing number of job seekers (referred to as candidates within the agency) availing the services of Looking Glass Recruitment, there is a need for a database application to streamline the processes of monitoring employers' vacancies and candidates' particulars. The primary objective of this new application is to establish a sturdy system that ensures employers receive accurate candidate information and candidates are directed towards suitable vacancies. It is imperative that this new application enhances the efficiency of Looking Glass Recruitment's services, leading to heightened satisfaction levels for both employers and candidates, and a substantial increase in its employer clientele.

The database must be designed and optimized according to the business requirements listed in table below:

Application Requirements	
Form Requirements	Reports Required
• Enter, modify, or delete candidates. • Enter, modify, or delete vacancies. • Add or remove skills or qualifications to or from a candidate. • Add or remove skills or qualifications to or from a vacancy. • Apply a candidate to a vacancy. • Mark a vacancy as filled.	• All filled vacancies. • All unfilled vacancies. • All unfilled vacancies with candidates interested. • Vacancies by category • Vacancies by employer • All candidates

The two most important processes are:

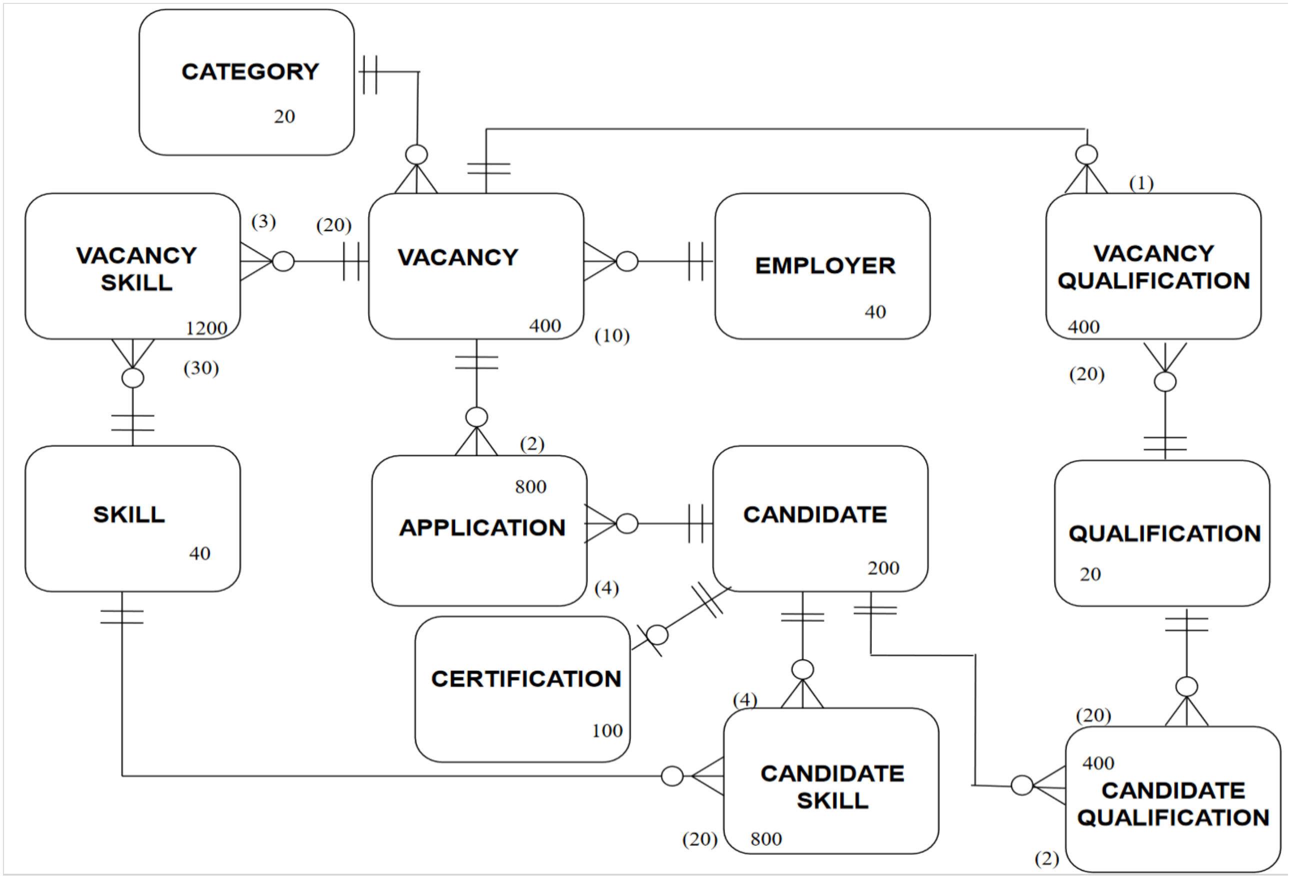
- All filled vacancies

  On average 5 times a day

- All unfilled vacancies with candidates interested.

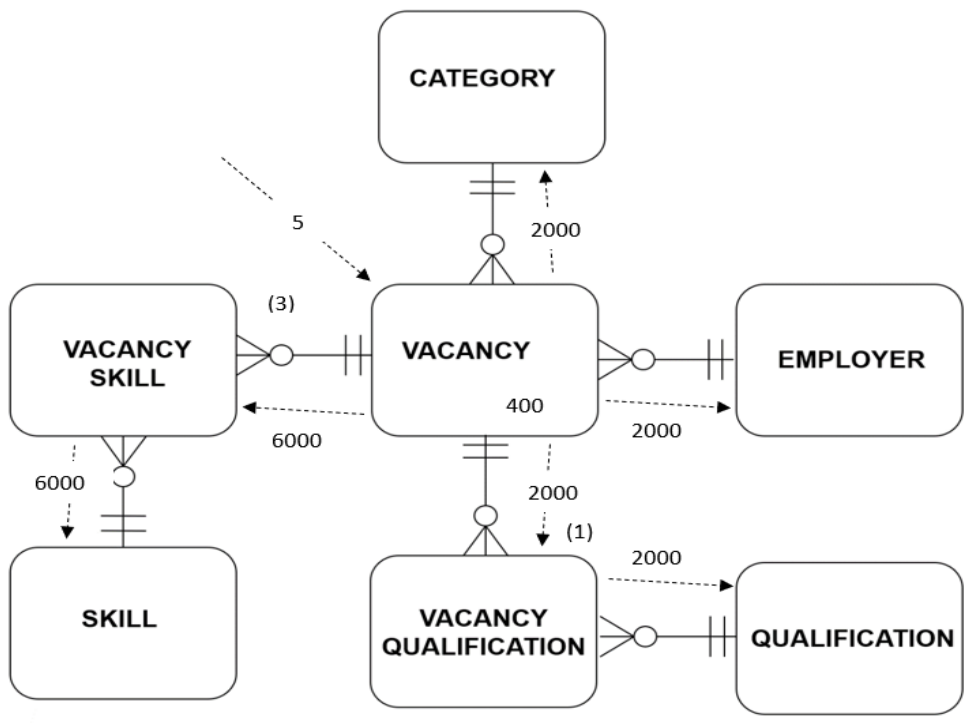
  On average 4 times a day

Task 2 - Data Volume Map: Looking Glass Recruitment

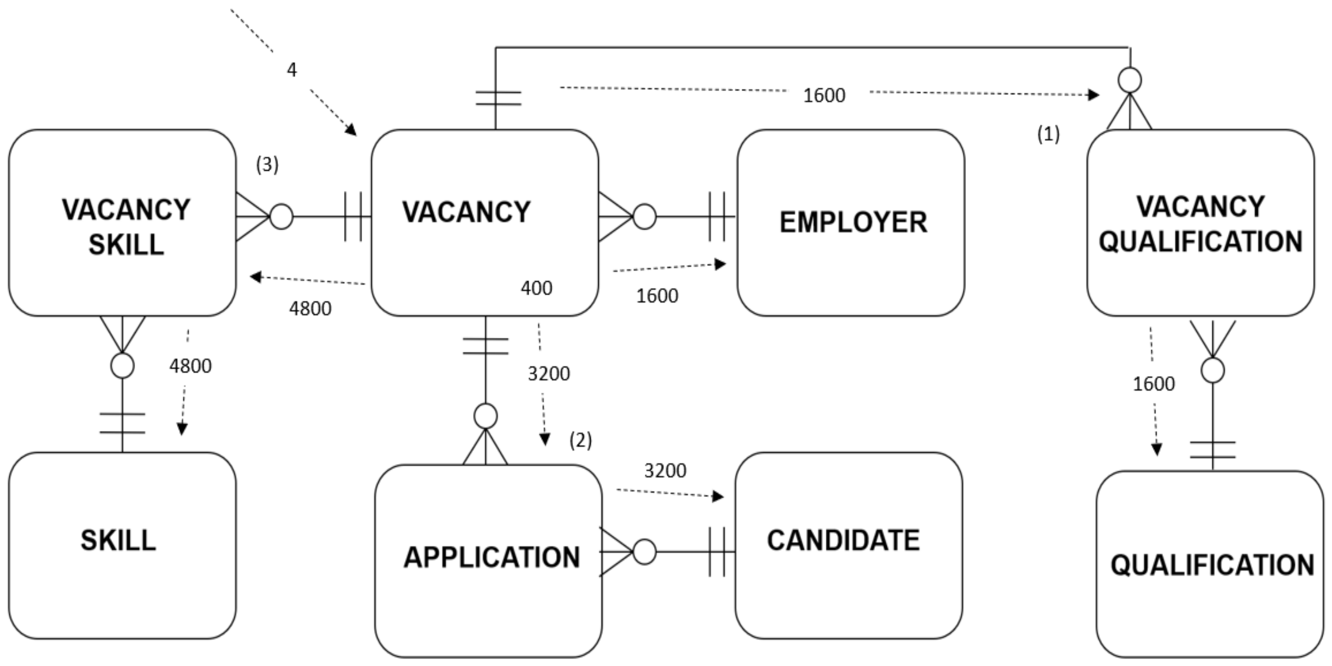


Task 3

3.1 - Data Usage Map: All Vacancies Filled



3.2 - Data Usage Map: All Unfilled Vacancies with Candidates Interested.



Task 4 - Business processes to physical design techniques matrix table using the business processes.

Data Entity types / Business functions	One-one Relationship	One-many Relationship(reference)	Associative Entity	Duplication	Horizontal Partitioning	Vertical Partitioning	Indexing
1.Enter, modify, or delete candidates.	Yes	No	No	No	No	Yes	Yes
2.Enter, modify, or delete vacancies.	No	Yes	No	Yes	No	Yes	Yes
3.Add or remove skills or qualifications to or from a candidate.	No	No	No	Yes	No	Yes	Yes
4.Add or remove skills or qualifications to or from a vacancy.	No	No	No	Yes	No	Yes	Yes
5.Apply a candidate to a vacancy.	No	No	No	Yes	Yes	Yes	Yes
6.Mark a vacancy as filled.	No	Yes	No	Yes	Yes	Yes	Yes
7.All filled vacancies.	No	Yes	Yes	Yes	Yes	Yes	No
8.All unfilled vacancies.	No	Yes	Yes	Yes	Yes	Yes	No
9.All unfilled vacancies with candidates interested.	No	No	Yes	Yes	Yes	Yes	No
10.Vacancies by Category.	Yes	No	No	Yes	No	Yes	Yes
11.Vacancies by Employer.	No	No	No	Yes	No	Yes	Yes
12.All Candidates.	Yes	No	No	Yes	No	Yes	Yes

Task 5

5.1 - Enter, modify, or delete candidates.

a) List candidate details (candidate ID, last name, first name, street address, suburb, phone number, and status) by first name within last name.
b) List certification details (notes and certification date) for a selected candidate.

Current SQL	a)	SELECT * FROM CANDIDATE
		ORDER BY LASTNAME, FIRSTNAME.

	b)	SELECT CERTIFICATIONDATE, NOTES FROM CERTIFICATION
		WHERE CANDIDATEID =4;

Indexing	a)	Candidate Last name and First name are frequently used among four business processes (1,3,5,12) to sort the dataset. Indexing on First name and last name can improve read performance. The SQL to this indexing is below: CREATE INDEX IDX_FIRSTNAME_LASTNAME ON CANDIDATE (LastName, FirstName);
Disadvantages		Indexing comes with the cost of increased storage and maintenance overhead. Whenever data is inserted, updated, or deleted in "Candidate" table (business process 1), the index needs to be maintained, which can slightly slow down these operations. Business process 1 is not an important process so indexing first name last name is considered.
Revised SQL		No change in SQL

5.2 - Enter, modify or delete vacancies.

a) List vacancy details (vacancy ID, description, type, status, hourly rate, employer name, and category name) by description.
b) List employers (employer ID, employer name) by employer name.
c) List categories (category ID, category name) by category name.

Current SQL	a)	SELECT V. VacancyID, V.Description,V.Type, V.Status,V.HourlyRate, E.EMPLOYERNAME, C.CATEGORYNAME FROM VACANCY V INNER JOIN EMPLOYER E ON V.EMPLOYERID = E.EMPLOYERID INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID ORDER BY DESCRIPTION;
	b)	SELECT EMPLOYERID, EMPLOYERNAME FROM EMPLOYER ORDER BY EMPLOYERNAME;
	c)	SELECT * FROM CATEGORY ORDER BY CATEGORYNAME;
Indexing	a)	In query "b" the data is sorted by employer name, in this case employer name can be indexed to make query to run faster. However, as per the description Employer numbers are likely to increase this makes it volatile, hence Indexing for employer name won't be carried out.
	b)	In query "c" the data is sorted by category name, in this case category name can be indexed to make query to run faster. Categories are unlikely to change often, this concludes to use indexing on category name. SQL below: CREATE INDEX IDX_CATEGORYNAME ON CATEGORY (CategoryName); This can be used when data is sorted by categoryname like in SQL c above.

Disadvantages	It consumes extra storage when indexing categoryname. Overheads on Insert, Update, and Delete Operations for category table.
Revised SQL	No change in SQL

5.3 - Add or remove skills or qualifications to or from a candidate.

a) List candidate details (candidate ID, last name, and first name) by first name within last name.
b) List skills (skill id and description) by description.
c) List skills (skill id, description, and years) by description for a selected candidate.
d) List qualifications (qualification id, level and description) by description.
e) List qualifications (qualification id, level and description) by description for a selected candidate.

Current SQL	a)	SELECT CANDIDATEID, LASTNAME, FIRSTNAME FROM CANDIDATE
		ORDER BY LASTNAME, FIRSTNAME;

	b)	SELECT * FROM SKILL
		ORDER BY DESCRIPTION;

	c)	SELECT CS.SKILLID, S.DESCRIPTION, CS.YEARS FROM CANDIDATESKILL CS
		INNER JOIN SKILL S ON CS.SKILLID=S.SKILLID
		WHERE CANDIDATEID = 2
		ORDER BY DESCRIPTION;

	d)	SELECT * FROM QUALIFICATION
		ORDER BY DESCRIPTION;

	e)	SELECT CQ.QUALIFICATIONID, Q.QUAL_LEVEL, Q.DESCRIPTION FROM CANDIDATEQUALIFICATION CQ
		INNER JOIN QUALIFICATION Q ON Q.QUALIFICATIONID = CQ.QUALIFICATIONID
		WHERE CQ.CANDIDATEID = 4;

Vertical Partitioning	a)	

		Vertical partitioning is available for candidate table, it will split into two tables called VP1_CANDIDATE((candidateid, firstname, lastname), VP2_CANDIDATE(streetaddress, suburb, phonenumber,status). In VP1 the columns are frequently used together in business process 3,5,6 and 9, to query the data. This will improve the performance of the query for business process 3,5,6 and 9 by reducing the amount data that needs to be read from disk. However, I decided not to carry out this partitioning as this is not an important business process and other optimization techniques are available.
Disadvantages		In terms of business process there will be an impact on performance for business process 1 and 12. These business processes require all the columns from the candidate table, query will be slow. Further partitioning can introduce data Inconsistency and integrity challenges.
Revised SQL		No Change in SQL

5.4 - Add or remove skills or qualifications to or from a candidate.

a) List vacancy details (vacancy ID, description, and employer name) for unfilled vacancies only by description.
b) List skills (skill id and description) by description.
c) List skills (skill id, description, and years) by description for a selected vacancy.
d) List qualifications (qualification id, level and description) by description.
e) List qualifications (qualification id, level and description) by description for a selected vacancy.

Current SQL	a)	SELECT
		   V.VACANCYID,
		   V.DESCRIPTION,
		   E.EMPLOYERNAME
		FROM
		   VACANCY V
		INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID
		WHERE STATUS = 'Unfilled'
		ORDER BY DESCRIPTION;

	b)	SELECT * FROM SKILL
		ORDER BY DESCRIPTION;

	c)	SELECT
		   VS.SKILLID,
		   S.DESCRIPTION,
		   VS.YEARS
		FROM
		   VACANCYSKILL VS
		INNER JOIN SKILL S ON S.SKILLID = VS.SKILLID
		WHERE VS.VACANCYID = 16
		ORDER BY S.DESCRIPTION;

	d)	SELECT * FROM qualification
		ORDER BY DESCRIPTION;

	e)	SELECT
		   VQ.QUALIFICATIONID,
		   Q.QUAL_LEVEL,
		   Q.DESCRIPTION
		FROM
		   VACANCYQUALIFICATION VQ
		INNER JOIN QUALIFICATION Q ON Q.QUALIFICATIONID =
		VQ.QUALIFICATIONID
		WHERE VQ.VACANCYID = 2
		ORDER BY DESCRIPTION;

Indexing	a)	Qualification and Skill tables records are unlikely to change, frequently used, and need to be ordered by description, this makes it appropriate to do Indexing on both tables for description table. SQL for indexing below: - CREATE INDEX SKILL_DESCRIPTION_ON SKILL (DESCRIPTION); - CREATE INDEX QUALIFICATION_DESCRIPTION_ON QUALIFICATION (DESCRIPTION); This Indexing can be used when sorting the skills and qualification by description as this is the case in this business process.
Disadvantages.		Indexing increases storage and maintenance overhead. Whenever data is inserted, updated, or deleted in skill and qualification table, the index needs to be maintained, which can slightly slow down these operations.
Revised SQL		No change

5.5 - Apply a candidate to a vacancy.

a) List vacancy details (vacancy ID, description, type, and employer name) for unfilled vacancies only by description.
b) List applications (candidate's last name and first name) for a selected vacancy by candidate's last name
c) List candidate details (candidate ID, last name, and first name) by first name within last name.
d) List skills (skill id, description, and years) by description for a selected vacancy.
e) List qualifications (qualification id, level and description) by description for a selected vacancy.
f) List skills (skill id, description, and years) by description for a selected candidate.
g) List qualifications (qualification id, level and description) by description for a selected candidate.

Current SQL	a)	SELECT V.VACANCYID, V.DESCRIPTION, E.EMPLOYERNAME FROM VACANCY V INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID WHERE STATUS = 'Unfilled' ORDER BY DESCRIPTION;

	b)	SELECT
		C.LASTNAME,
		C.FIRSTNAME
		FROM CANDIDATE C
		INNER JOIN APPLICATION A ON C.CANDIDATEID = A.CANDIDATEID
		WHERE A.VACANCYID = 8
		ORDER BY C.LASTNAME;

	c)	SELECT CANDIDATEID, LASTNAME, FIRSTNAME FROM CANDIDATE
		ORDER BY LASTNAME, FIRSTNAME;

	d)	SELECT VS.SKILLID, S.DESCRIPTION, VS.YEARS FROM VACANCYSKILL VS INNER JOIN SKILL S ON S.SKILLID = VS.SKILLID WHERE VS.VACANCYID = 16 ORDER BY S.DESCRIPTION;

	e)	SELECT
		    VQ.QUALIFICATIONID,
		    Q.QUAL_LEVEL,
		    Q.DESCRIPTION
		FROM
		    VACANCYQUALIFICATION VQ
		INNER JOIN QUALIFICATION Q ON Q.QUALIFICATIONID = VQ.QUALIFICATIONID
		WHERE VQ.VACANCYID = 2
		ORDER BY DESCRIPTION;

	f)	SELECT CS.SKILLID, S.DESCRIPTION, CS.YEARS FROM CANDIDATESKILL CS
		INNER JOIN SKILL S ON CS.SKILLID=S.SKILLID
		WHERE CANDIDATEID = 2
		ORDER BY DESCRIPTION;

	g)	SELECT CQ.QUALIFICATIONID, Q.QUAL_LEVEL, Q.DESCRIPTION FROM CANDIDATEQUALIFICATION CQ
		INNER JOIN QUALIFICATION Q ON Q.QUALIFICATIONID = CQ.QUALIFICATIONID
		WHERE CQ.CANDIDATEID = 4;

Duplication	a)	Data can be duplicated into VACANCYSKILL and CANDIDATESKILL table by adding Description column. This would reduce the extra join required in terms of query (d) and (f). Hence improvement in query performance. I choose to carry out duplication only into VACANCYSKILL table as this increases the performance of this and important business processes 7 and 9. This can be used in query d) as well as important business process 7 and 9. This would eliminate extra join and simplify the query. **VACANCYSKILL** (<u>VacancyID</u>*, SkillID*, Years, Skill_Description)
	b)	Data can be duplicated into VACANCYQUALIFICATION and CANDIDATEQUALIFICATION table by adding Description and Level column. This would reduce the extra join required in terms of query (e) and (g). Hence improvement in query performance. I choose to carry out duplication only into VACANCYQUALIFICATION as this increases the performance of this and important business processes 7 and 9. This can be used in query e) as well as important business process 7 and 9. This would eliminate extra join and simplify the query.

		VACANCYQUALICATION (<u>VacancyID*, QualificationID</u>*, Qual_Level, Qual_Description)
Disadvantages		Duplicated data can lead to data integrity problems. If the same information is stored in multiple places, there's a risk of inconsistency or discrepancies if not managed carefully. Duplication increases the storage requirements. This can be significant if large amounts of data are duplicated.

Revised SQL	d)	SELECT
		SKILLID,
		DESCRIPTION,
		YEARS
		FROM
		VACANCYSKILL
		WHERE VACANCYID = 16
		ORDER BY DESCRIPTION;

	e)	SELECT
		  QUALIFICATIONID,
		  QUAL_LEVEL,
		  DESCRIPTION
		FROM
		  VACANCYQUALIFICATION VQ
		  WHERE VQ.VACANCYID = 2
		   ORDER BY DESCRIPTION;

5.6 - Mark a vacancy as filled.

a) List vacancy details (vacancy ID, description, type, hourly rate and employer name) for unfilled vacancies only by description.
b) List applications (status, candidate's last name and first name) for a selected vacancy by candidate's last name

Current SQL	a)	SELECT
		  V.VACANCYID,
		  V.DESCRIPTION,
		  V.TYPE,
		  V.STATUS,
		  V.HOURLYRATE,
		  E.EMPLOYERNAME
		FROM

		VACANCY V
		INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID
		WHERE STATUS = 'Unfilled'
		ORDER BY DESCRIPTION;

	b)	SELECT
		C.LASTNAME,
		C.FIRSTNAME
		FROM CANDIDATE C
		INNER JOIN APPLICATION A ON C.CANDIDATEID = A.CANDIDATEID
		WHERE A.VACANCYID = 8
		ORDER BY C.LASTNAME;

Vertical Partitioning	a)	In query a) only certain column of the vacancy table is used, there is possibility of vertical partitioning. The partition will look like below: VP1_VACANCY(VACANCYID, DESCRIPTION, TYPE STATUS, HOURLYRATE) VP2_VACANCY(VACANCYID, EMPLOYERID, CATEGORYID) VP1 can be used in query a) and this would improve performance of this query. Since this is not an important business process and it would bring negative impact on performance to important business process, So I decide not to use it.
Disadvantages		It would bring data redundancy and increased maintenance which may outweigh the performance benefits and effects performance of important business processes.
Revised SQL		No Change in SQL

5.7 - All filled vacancies.

a) For each filled vacancy list the vacancy id, description, type, status, category name, employer name, employer street address, employer suburb, descriptions of skills, and descriptions of qualifications.

Current SQL	a)	SELECT V.VACANCYID, V.Description, V.TYPE, V.STATUS, C.CATEGORYNAME,

		E.EMPLOYERNAME, E.STREETADDRESS, E.SUBURB, S.Description AS SKILL_QUALIFICATION FROM VACANCY V INNER JOIN VACANCYSKILL VS ON V.VACANCYID = VS.VACANCYID INNER JOIN SKILL S ON VS.SKILLID = S.SKILLID INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID WHERE V.STATUS = 'Filled' UNION SELECT V.VACANCYID, V.Description, V.TYPE, V.STATUS, C.CATEGORYNAME, E.EMPLOYERNAME, E.STREETADDRESS, E.SUBURB, Q.Description AS SKILL_QUALIFICATION FROM VACANCY V INNER JOIN VACANCYQUALIFICATION VQ ON V.VACANCYID = VQ.VACANCYID INNER JOIN QUALIFICATION Q ON VQ.QUALIFICATIONID = Q.QUALIFICATIONID INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID WHERE V.STATUS = 'Filled'
De-normalization opportunity	a)	There is an opportunity to de-normalize category table by combining into vacancy table. This would eliminate the join requirements to retrieve category name from category table, thus enhancing the performance of this query as well queries used in other business processes 2,8 and 10.
Disadvantages		Since “CATEGROYNAME” will exist in vacancy table this leads to data redundancy. Data manipulation is slowed. When a column is regularly updated, the speed of updates slows down. Impact on Business process 2.c wouldn’t exist.
Revised SQL		SELECT

```
                V.VACANCYID,
                V.Description,
                V.TYPE,
                V.STATUS,
                V.CATEGORYNAME,
                E.EMPLOYERNAME,
                E.STREETADDRESS,
                E.SUBURB,
                VS.Description AS SKILL_QUALIFICATION
        FROM VACANCY V
        INNER JOIN VACANCYSKILL VS ON V.VACANCYID = VS.VACANCYID
        INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID
        WHERE V.STATUS = 'Filled'
        UNION
        SELECT
                V.VACANCYID,
                V.Description,
                V.TYPE,
                V.STATUS,
                V.CATEGORYNAME,
                E.EMPLOYERNAME,
                E.STREETADDRESS,
                E.SUBURB,
                VQ.Description AS SKILL_QUALIFICATION
        FROM VACANCY V
        INNER JOIN VACANCYQUALIFICATION VQ ON V.VACANCYID =
VQ.VACANCYID
        INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID
        WHERE V.STATUS = 'Filled'
```

Please note: Query above also reflect changes because of duplication technique applied in Business process 5.

5.8 - All unfilled vacancies.

a) For each unfilled vacancy list the vacancy id, description, type, status, category name, employer name, employer street address, employer suburb, descriptions of skills, and descriptions of qualifications.

Current SQL	a)	`SELECT V.VACANCYID, V.Description, V.TYPE, V.STATUS, C.CATEGORYNAME, E.EMPLOYERNAME, E.STREETADDRESS, E.SUBURB, S.Description AS SKILL_QUALIFICATION FROM VACANCY V INNER JOIN VACANCYSKILL VS ON V.VACANCYID = VS.VACANCYID INNER JOIN SKILL S ON VS.SKILLID = S.SKILLID INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID WHERE V.STATUS = 'Unfilled' UNION SELECT V.VACANCYID, V.Description, V.TYPE, V.STATUS, C.CATEGORYNAME, E.EMPLOYERNAME, E.STREETADDRESS, E.SUBURB, Q.Description AS SKILL_QUALIFICATION FROM VACANCY V INNER JOIN VACANCYQUALIFICATION VQ ON V.VACANCYID = VQ.VACANCYID INNER JOIN QUALIFICATION Q ON VQ.QUALIFICATIONID = Q.QUALIFICATIONID INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID WHERE V.STATUS = 'Unfilled'`
Horizontal Partitioning	a)	Vacancy table “Status” attribute is either “filled” or “unfilled”. When querying, where clause is used. In this case it is reasonable to perform Horizontal partitioning on Vacancy table. This will make the query run faster for both important business processes (7,9) and other business processes as well (5,6,8). The vacancy table will split up horizontally, VACANCY_FILLED and VACANCY_UNFILLED, based on the value of status field.

Current SQL a):

```
SELECT
        V.VACANCYID,
        V.Description,
        V.TYPE,
        V.STATUS,
        C.CATEGORYNAME,
        E.EMPLOYERNAME,
        E.STREETADDRESS,
        E.SUBURB,
        S.Description AS SKILL_QUALIFICATION
FROM VACANCY V
INNER JOIN VACANCYSKILL VS ON V.VACANCYID = VS.VACANCYID
INNER JOIN SKILL S ON VS.SKILLID = S.SKILLID
INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID
INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID
WHERE V.STATUS = 'Unfilled'
UNION
SELECT
        V.VACANCYID,
        V.Description,
        V.TYPE,
        V.STATUS,
        C.CATEGORYNAME,
        E.EMPLOYERNAME,
        E.STREETADDRESS,
        E.SUBURB,
        Q.Description AS SKILL_QUALIFICATION
FROM VACANCY V
INNER JOIN VACANCYQUALIFICATION VQ ON V.VACANCYID = VQ.VACANCYID
INNER JOIN QUALIFICATION Q ON VQ.QUALIFICATIONID =
Q.QUALIFICATIONID
INNER JOIN EMPLOYER E ON E.EMPLOYERID = V.EMPLOYERID
INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID
WHERE V.STATUS = 'Unfilled'
```

		I choose to perform this technique; this will make the business process faster as the data is already filtered and less rows to query through.
Disadvantages		Managing multiple partitions can add complexity to the database design and maintenance process. Horizontal partitioning can lead to increased storage requirements.
Revised SQL		No change in terms of Horizontal partitioning applied above.

5.9 - All unfilled vacancies with candidates interested.

a) For each unfilled vacancy list the vacancy id, description, type, status, employer name, descriptions and years of skills, descriptions of qualifications and full names of the candidates.

Current SQL	a)	
		```
		SELECT
		   V.VACANCYID,
		   V.DESCRIPTION,
		   V.TYPE,
		   V.STATUS,
		   E.EMPLOYERNAME,
		   S.DESCRIPTION AS SKILL_DESCRIPTION,
		   VS.YEARS,
		   Q.DESCRIPTION AS QUALIFICATION_DESCRIPTION,
		   C.LASTNAME,
		   C.FIRSTNAME
		FROM VACANCY V
		INNER JOIN VACANCYSKILL VS ON VS.VACANCYID = V.VACANCYID
		INNER JOIN SKILL S ON S.SKILLID = VS.SKILLID
		INNER JOIN VACANCYQUALIFICATION VQ ON VQ.VACANCYID =
		V.VACANCYID
		INNER JOIN QUALIFICATION Q ON VQ.QUALIFICATIONID =
		Q.QUALIFICATIONID
		INNER JOIN EMPLOYER E ON V.EMPLOYERID = E.EMPLOYERID
		INNER JOIN APPLICATION A ON A.VACANCYID = V.VACANCYID
		INNER JOIN CANDIDATE C ON A.CANDIDATEID = C.CANDIDATEID
		WHERE V.STATUS = 'Unfilled'
		```

Duplication	a)	Duplication of Candidate’s First and last name will be considered. First and last name will be duplicated into Application table. Standard relation notion for Application table will look like below:

		APPLICATION (CandidateID*, VacancyID*, Status, C_FirstName, C_LastName) This will eliminate the join required to retrieve data from Candidate table (FirstName, LastName). Thus, increase query performance for this process.
Disadvantages		Since candidate First Name and last name will exist in two tables (Application and Candidate) this leads extra consumption in disk space and data redundancy. Maintenance will increase, updating and deleting data needs to happen in both tables.

Revised SQL		SELECT
		    V.VACANCYID,
		    V.DESCRIPTION,
		    V.TYPE,
		    V.STATUS,
		    E.EMPLOYERNAME,
		    VS.DESCRIPTION AS SKILL_DESCRIPTION,
		    VS.YEARS,
		    VQ.DESCRIPTION AS QUALIFICATION_DESCRIPTION,
		    A.LASTNAME,
		    A.FIRSTNAME
		FROM VACANCY V
		INNER JOIN VACANCYSKILL VS ON VS.VACANCYID = V.VACANCYID
		INNER JOIN VACANCYQUALIFICATION VQ ON VQ.VACANCYID = V.VACANCYID
		INNER JOIN EMPLOYER E ON V.EMPLOYERID = E.EMPLOYERID
		INNER JOIN APPLICATION A ON A.VACANCYID = V.VACANCYID
		WHERE V.STATUS = 'Unfilled'
		
		Please note: Query above also reflect changes because of duplication technique applied in Business process 5.

5.10 - Vacancies by Category.

a) For each vacancy list the vacancy id, description, hourly rate, and category name. Sort by category name.

Current SQL	a)	SELECT
		  V.VACANCYID,
		  V.DESCRIPTION,
		  V.HOURLYRATE,
		  C.CATEGORYNAME
		FROM VACANCY V
		INNER JOIN CATEGORY C ON C.CATEGORYID = V.CATEGORYID
		ORDER BY C.CATEGORYNAME

Duplication	a)	There is an opportunity to duplicate “CATEGORYNAME” attribute into vacancy table. A new column will be added into the vacancy table with the name “CATEGORYNAME”. This would eliminate the join requirements to retrieve category name from category table, thus enhancing the performance of this query as well queries used in other business processes 2,8 and 10. I choose not to carry out this duplication. As disadvantages may outweigh the benefits.
Disadvantages		Since “CATEGROYNAME” will exist in two tables (Categories and Vacancy) this leads extra consumption in disk space and data redundancy. Maintenance will increase, updating and deleting data needs to happen in both tables.
Revised SQL		No change in SQL.

5.11 - Vacancies by Employer.

f) For each vacancy list the vacancy id, type, status, and employer name. Sort by employer name.

Current SQL	a)	SELECT
		  V.VACANCYID,
		  V.TYPE,
		  V.STATUS,

		E.EMPLOYERNAME
		FROM VACANCY V
		INNER JOIN EMPLOYER E ON V.EMPLOYERID = E.EMPLOYERID
		ORDER BY EMPLOYERNAME

Duplication	a)	There is an opportunity to duplicate the “EMPLOYERNAME” into vacancy and enhance the performance of the above query. Considering this is not as important business process, duplication won’t be carried out.
Disadvantages		This duplication will lead to extra consumption of disk space and data redundancy. Maintenance will increase, updating and deleting data needs to happen in both tables.
Revised SQL		No change in SQL since not duplicating.

5.12 - All Candidates.

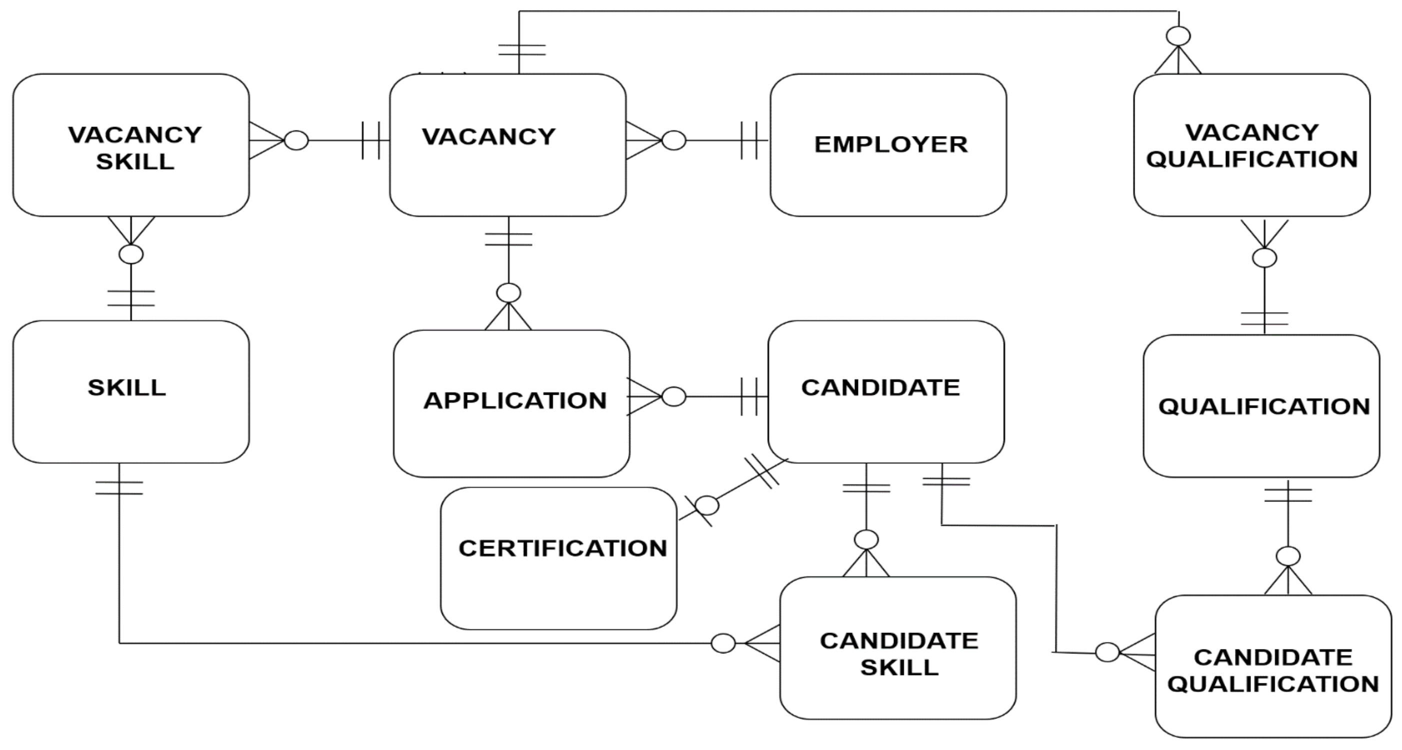
a) For each candidate list the candidate id, last name, first name, street address, suburb, phone number, status, and certification date (if applicable). Sort by first name within last name.

Current SQL	a)	SELECT C.CANDIDATEID, C.LASTNAME, C.FIRSTNAME, C.STREETADDRESS, C.SUBURB, C.PHONENUMBER, C.STATUS, CD.CERTIFICATIONDATE FROM CANDIDATE C LEFT JOIN CERTIFICATION CD ON C.CANDIDATEID = CD.CANDIDATEID ORDER BY C.LASTNAME, C.FIRSTNAME
De-normalization opportunity	a)	There is one to one relationship between Candidate and Certification table due to this de-normalization opportunity exists. After de-normalizing, Certification table attributes will be added in the Candidate table.

		This will eliminate the join when querying this business process and the query will perform faster. For All Candidate's business process only need to query one table (CANDIDATE) instead two. I decided not to carry out de-normalization opportunity described above because this is not the important business process, and this will potentially slow down the other important business process (9) by increasing the size of the Candidate table vertically.
Disadvantages		Data manipulation/maintenance is harder. Risk of anomalies.
Revised SQL		No Change.

Task 6

6.1 - Final Physical Design



6.2 – Standard relation notation and On cascade delete.

EMPLOYER (EmployerID, EmployerName, StreetAddress, Suburb, Email, PhoneNumber)

CANDIDATE (CandidateID, FirstName, LastName, StreetAddress, Suburb, PhoneNumber, Status)

SKILL (SkillID, Description)

QUALIFICATION (QualificationID, Description, Level)

VACANCY (VacancyID, Description, Type, HourlyRate, Status, EmployerID*, CategoryName)

EMPLOYERID: ON DELETE RESTRICT

APPLICATION (CandidateID*, VacancyID*, Status, C_FirstName, C_LastName)

CANDIDATEID: ON DELETE RESTRICT

VACANCYID: ON DELETE RESTRICT

VACANCYSKILL (VacancyID*, SkillID*, Years, Skill_Description)

SKILLID: ON DELETE RESTRICT

VACANCYID: ON DELETE RESTRICT

CANDIDATESKILL (CandidateID*, SkillID*, Years)

CANDIDATEID: ON DELETE CASCADE

VACANCYQUALICATION (VacancyID*, QualificationID*, Qual_Level, Qual_Description)

QUALIFICATIONID: ON DELETE RESTRICT

CANDIDATEQUALICATION (CandidateID*, QualificationID*)

QUALIFICATIONID: ON DELETE CASCADE

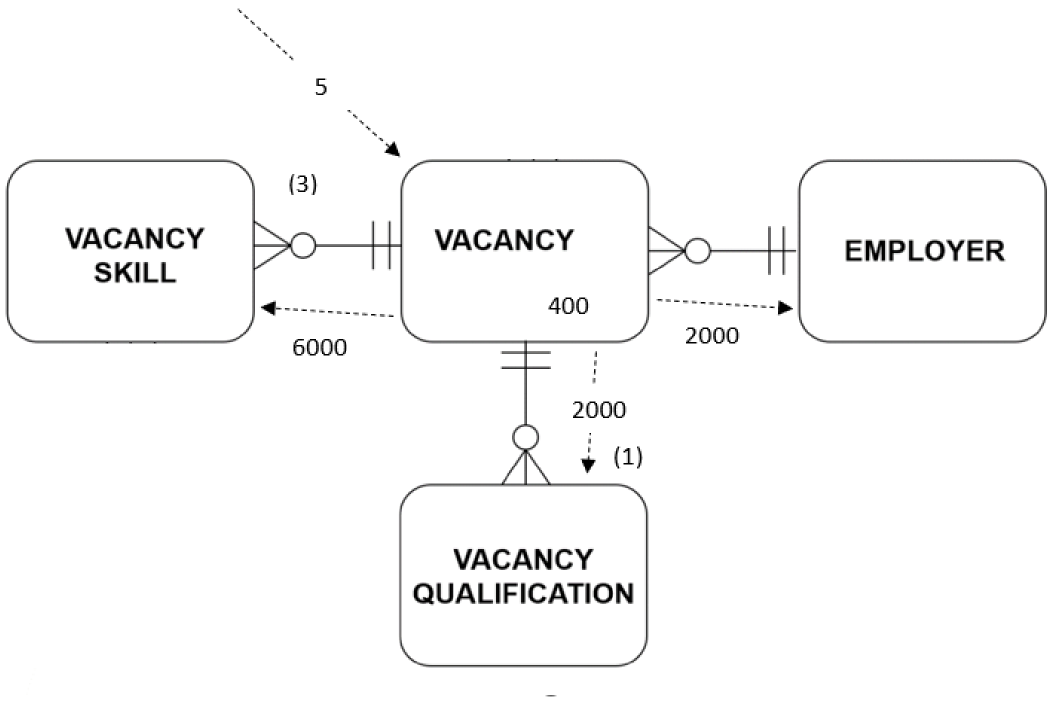
VACANCYID: ON DELETE RESTRICT

CERTIFICATION (CandidateID*, CertificationDate, Notes)

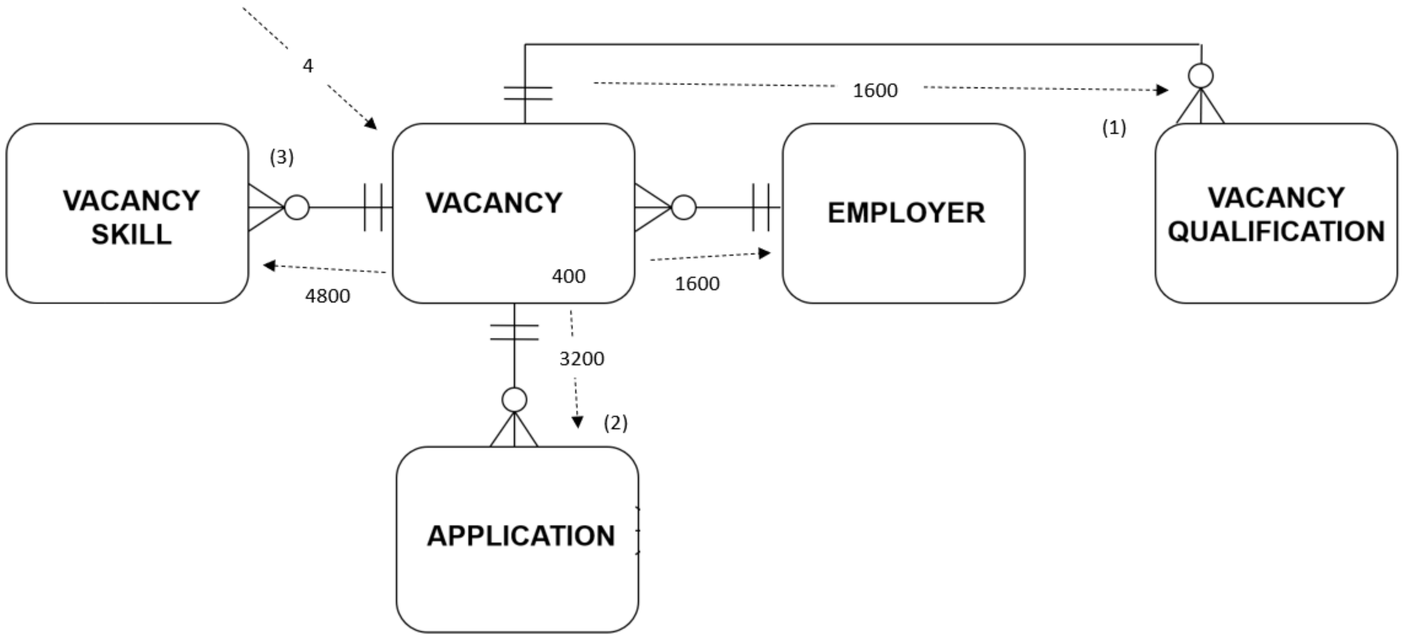
CANDIDATEID: ON DELETE CASCADE

Task 7 Redevelop the two data usage maps.

7.1 - All Vacancies Filled



7.2 - All Unfilled Vacancies with Candidates Interested.



Task 8 - Physical design data dictionary.

Employer

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
EmployerID	The identifier of Employer.	NUMBER(5,0)	1-9999	Yes	-	Yes	Auto-Number	Auto
EmployerName	The name of the Employer	VARCHAR2(30)	Example: 'Data Logic'	-	-	Yes	-	-
StreetAddress	The street address where Employer resides.	VARCHAR2(40)	Example: '82 Bell Road'	-	-	Yes	-	-
Suburb	The suburb where the Employer resides.	VARCHAR2(20)	Example: 'Parnell'	-	-	Yes	-	-
EmailAddress	The Employer's email address.	VARCHAR2(30)	Example: 'hr@rts.co.nz'	-	-	Yes	-	-
PhoneNumber	The Employer's Phone number	VARCHAR2(12)	Example: '212 3452'	-	-	Yes	-	-

Candidate

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
CandidateID	The identifier of the Candidate	NUMBER(5,0)	1-9999	Yes	-	Yes	Auto-Number	Auto
LastName	The last name of the Candidate	VARCHAR2(20)	Example: 'Ellen'		-	Yes	-	Index 1
FirstName	The first name of the Candidate	VARCHAR2(20)	Example: 'John'	-	-	Yes	-	Index 2
StreetAddress	The street Address where Candidate resides	VARCHAR2(30)	Example: '1 King Street'	-	-	Yes	-	-
Suburb	The suburb where the Candidate resides	VARCHAR2(20)	Example: 'Avondale'	-	-	Yes	-	-
PhoneNumber	The Candidate; s phone number	VARCHAR2(12)	Example: '123 444'	-	-	Yes	-	-
Status	The Candidate's employment status	VARCHAR2(10)	Example: 'Employed or Unemployed'	-	-	Yes	-	-

Skill

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
SkillID	The identifier of Skill	NUMBER(5,0)	1-999	Yes	-	Yes	Auto-Number	Auto
Description	The description of skill.	VARCHAR(30)	Example: 'MS Access'	-	-	Yes	-	Index 3

Qualification

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
QaulificationID	The identifier of Qualification	NUMBER(3.0)	1-999	Yes	-	Yes	Auto-number	Auto
Description	The description of Qualification	VARCHAR(30)	Example: 'Bachelor of Art'	-	-	Yes	-	Index 4
Qual_Level	The level of the Qualification	NUMBER(2,0)	Between 1 and 12, Example: '7'	-	-	Yes	-	-

Vacancy

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
VacancyID	The identifier of the Vacancy	NUMBER(5,0)	1-99999	Yes	-	Yes	Auto-Number	Auto
Description	The description of Vacancy	VARCHAR(30)	Example: 'Programmer'	-	-	Yes	-	-
HourlyRate	Hourly rate of the Vacancy	NUMBER(6,2)	1-999999	-	-	Yes	-	-
Status	The status of the Vacancy	VARCHAR(30)	Filled or Unfilled	-	-	Yes	-	-
Type	The type of the Vacancy	VARCHAR(30)	Part-time or Full-time	-	-	Yes	-	-
EmployerID	The identifier of the Employer	NUMBER(5,0)	1-99999	-	Yes(Employer)	Yes	-	-
CategoryName	The category name of Vacancy	VARCHAR(30)	Example: 'Software Developer'	-	-	Yes	-	-

Note: The Vacancy table will be split up horizontally into 2 partitions, VACANCY_FILLED AND VACANCY_FILLED, based on the value of the status field.

Application

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
CandidateID	The identifier of Candidate	NUMBER5,0)	1-99999	Yes	Yes (Candidate)	Yes	-	Auto
VacancyID	The identifier of Vacancy	NUMBER(5,0)	1-99999	Yes	Yes (Vacancy)	Yes	-	Auto
Status	The Status of the Application	VARCHAR2(8)	Pending or successful or unsuccessful	-	-	Yes	-	
C_FirstName	The First Name of the Candidate	VARCHAR2(20)	Example: 'Simmons'	-	-	Yes	-	Index 2
C_LastName	The Last Name of the Candidate	VARCHAR2(20)	Example: 'Joe'	-	-	Yes	-	Index 1

VacancySkill

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
VacancyID	The identifier of the Vacancy	Number(5,0)	1-99999	Yes	Yes (Vacancy)	Yes	-	Auto
SkillID	The Identifier of the Skill	Number(5,0)	1-99999	Yes	Yes (Skill)	Yes	-	Auto
Years	Number of years in skill reauired.	Number(2,0)	1-10	-	-	Yes	-	-
Skill_Description	Description of the skill.	VARCHAR(30)	Example: 'MS Access'	-	-	Yes	-	Index 3

VacancyQualification

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
VacancyID	The identifier of Vacancy	Number(5,0)	1-99999	Yes	Yes (Vacancy)	Yes	-	Auto
QaulificationID	The identifier of Qualification	NUMBER(3.0)	1-999	Yes	Yes (Qualification)	Yes	-	Auto
Description	The description of Qualification	VARCHAR(30)	Example: 'Bachelor of Art'	-	-	Yes	-	Index 4
Qual_Level	The level of the Qualification	NUMBER(2,0)	Between 1 and 12, Example: '7'	-	-	Yes	-	-

CandidateQualification

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
CandidateID	The identifier of Candidate	NUMBER5,0)	1-99999	Yes	Yes (Candidate)	Yes	-	Auto
QaulificationID	The identifier of Qualification	NUMBER(3.0)	1-999	Yes	Yes (Qualification)	Yes	-	Auto

Certification

Name	Description	Data Type & Size	Domain (all ranges are inclusive), example or format	Primary key	Foreign Key	Required	Default value	Index
CandidateID	The identifier of Candidate	NUMBER5,0)	1-99999	Yes	Yes (Candidate)	Yes	-	Auto
CertificationDate	Date of the Certification issued.	DATE	DD-MM-YY	-	-	Yes	-	-
Notes	Notes associated with Certification.	VARCHAR(40)	Example: 'Programming'	-	-	Yes	-	-