Human Resource Management System

Assignment 1

Distributed System Design

Under guidance of-
Prof. Ananthanarayana V. S.

Submitted by-
Namrata Kadam (202IT008)

Introduction

Human resource management system(HRMS) includes following features -

- Recruitment process
- Performance management
- Workforce management
- Time and attendance
- Absence and leave management
- Learning and development
- Talent management

Each department maintains their respective information system. The recruitment department manages and coordinates the recruitment process. The performance and workforce management is handled by the performance management team. Similarly, the leave management system takes care of attendance and leave management. The training department manages learning, development as well as talent management.

Data sources

- Recruitment department
- Performance management system
- Leave management system
- Learning and talent management system

Actors

- Human resource manager
- Job applicants
- Employees
- Resource manager

Data flow and control flow

- Recruitment process (Data related to open positions and job applications)
  - Head HR and associate HR uses this information to coordinate recruitment process

- ➢ Interviewee and Interviewer both uses this data for checking candidate's skills, personal information and application status
- ➢ Project managers can look for required skilled candidates, also can fill the project requirements to hire people
- ➢ Resource management team can check for requirements and can contact the required skilled people present in resource pool, they can be allocated instead of new recruitment
- ➢ Open position data contains information like position id, designation, role, skill required, experience in years, work location
- ➢ Candidate application contains information like application id, position id (for which candidate is applying for), candidate name, mail id, skill, experience in years and application status

❖ Performance and workforce management (Data of employee performance)

- ➢ Reporting managers can use database to fill the performance assessment details like ratings and reviews
- ➢ Project manager, unit head and HR can use performance ratings for giving compensation, rewards and promotions
- ➢ Employees can update their skills
- ➢ Project manager, unit head and HR can take appropriate actions for employees with low performance
- ➢ Training department people can look for skilled employees and ask them to arrange trainings
- ➢ Performance system contains information like employee id, employee name, designation, project name, skill, performance ratings, performance assessment date and performance remark

❖ Attendance and leave management (attendance and leave related data)

- ➢ Reporting manager can look at subordinate's attendance in particular number of working hours while filling the performance assessment
- ➢ HR and unit head can check for unpaid leaves taken by employee if any while considering him for promotions
- ➢ Leaves system includes data like employee id, month, unpaid leave hours, extra hours worked
- ➢ Attendance system contains data like employee id, date, entry time, exit time, total working hours, effective working hours

❖ Learning and development (trainings attended, test scores and resource pool related data)

- ➢ Employees can check training details.
- ➢ Employees who attended training will go through the test after training , the test

scores will be stored.

- ➢ Training details contains training id, training name, prerequisites, topics covered, from date, to date
- ➢ Employee details who attended training contains training id, employee id, employee name, test score, grade
- ➢ Resource management will be part of this department. People in the resource pool (who are not allocated to any project) will be trained as per required skills.
- ➢ Resource management system will contain details of employees in resource pool (not allocated to any project) which includes employee id, employee name, mail id, skill, preferred location 1, preferred location 2

Constraint

1. Salary scale of lead and manager should be greater than the salary of a software engineer.

Queries

Simple queries

1. Check for open positions if any at a given location and for particular skill.
2. List the name of candidates who all applied for an open position for a given designation.
3. Select list of training attended by an employee.
4. Look for unpaid leave hours in a given month for an employee.

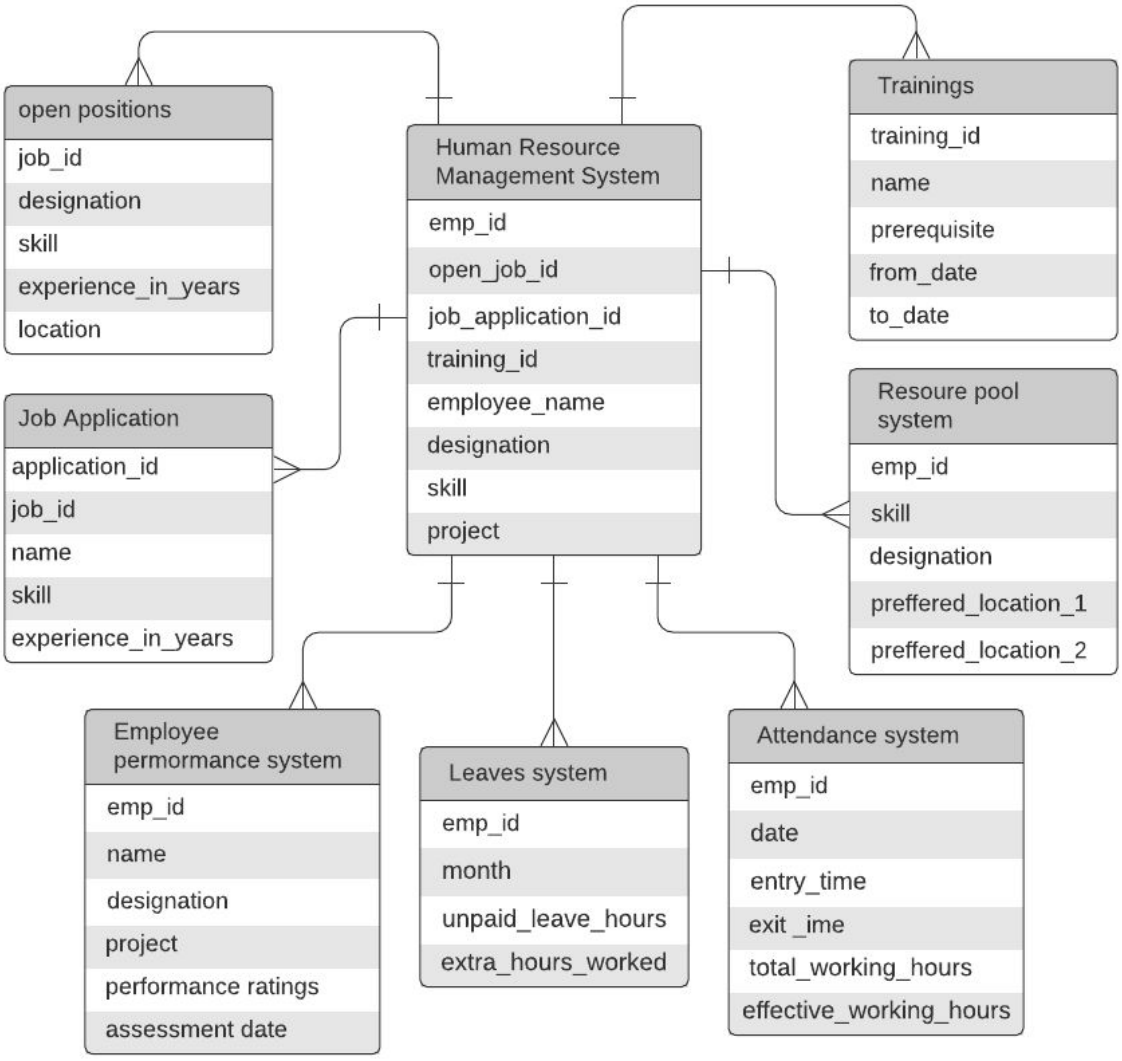
Complex queries

1. Select employee details whose performance is above average and attended required training with expected training test score.
2. Select employee details who is in a resource pool that is not allocated to any project and whose skill and preferred location matches with any open position.
3. Select employee details who has not taken any unpaid leaves in a given month and allocated to a given project.

Data sources maintained at different locations

Data source	Location	Description
Recruitment system	Site 1	Maintains information about open job positions and details of job applicants
Performance management system	Site 2	Maintains employee's performance ratings and remarks details
Leave management system	Site 3	Maintains employee's leave and attendance details
Learning and talent management system	Site 4	Maintains the details of trainings being held under learning department and details of employees who are not allocated to any project

EER model (conceptual model)



Global conceptual schema

Open positions

job_id	designation	role	skill	exp_in_yrs	location

Job Application

application_id	job_id	name	mail_id	skill	exp_in_yrs	status

Employee performance System

emp_id	name	designation	project	skill	rating	assessment_date

Leaves System

emp_id	month	unpaid_leave_hours	extra_hours_worked

Attendance System

emp_id	date	entry_time	exit_time	total_working_hours	effective_working_hours

Training

training_id	name	prerequisites	from_date	to_date

Trainee

emp_id	training_id	emp_name	test_score	grade

Resource Pool System

emp_id	name	mail_id	skill	designation	pref_location_1	pref_location_2

Human Resource management system

emp_id	open_job_id	job_application_id	training_id	emp_name	designation	skill	project

Normalization

Normalization is the process of organizing a database to reduce redundancy and improve data integrity. Normalization also simplifies the database design so that it achieves the optimal structure composed of atomic elements (i.e. elements that cannot be broken down into smaller parts).
Also referred to as database normalization or data normalization, normalization is an important part of relational database design, as it helps with the speed, accuracy, and efficiency of the database.
By normalizing a database, data can be arranged into tables and columns. It ensures that each table contains only related data. If data is not directly related, a new table can be created for that data.
For example, if we have a “Customers” table, we’d normally create a separate table for the products they can order (we could call this table “Products”). We’d create another table for customers’ orders (perhaps called “Orders”). And if each order could contain multiple items, we’d typically create yet another table to store each order item (perhaps called “OrderItems”). All these tables would be linked by their primary key, which allows us to find related data across all these tables (such as all orders by a given customer).
Normalization reduces redundancy and anomalies.

Redundancy : Redundancy means having repeated values in the same table, which leads to wastage of space, for example, for 100 employees working for the same department, we have to repeat the same department information for all these 100 employees in a table. We have huge storage capacity at lower cost nowadays. So, what is the issue here? Anomalies.
Anomalies : Normalization is the process of splitting relations into well structured relations that allow users to insert, delete, and update tuples without introducing database inconsistencies. Without normalization many problems can occur when trying to load an integrated conceptual model into the DBMS. These problems arise from relations that are generated directly from user views and are called anomalies.
Insertion Anomaly : An insertion anomaly is the inability to add data to the database due to absence of other data. For example, a Student_Group is defined so that null values are not allowed. If a new employee is hired but not immediately assigned to a Student_Group then this employee could not be entered into the database. This results in database inconsistencies due to omission.
Deletion Anomaly : A deletion anomaly is the unintended loss of data due to deletion of other data. For example, if the student group G1 was deleted from the table Student_Group, some departments would cease to exist. This results in database inconsistencies and is an example of how combining information that does not really belong together into one table can cause problems.
Update Anomaly : An update anomaly is a data inconsistency that results from data redundancy and a partial update. For example, each employee in a company has a department associated with them as well as the student group they participate in.

We can achieve normalization using following steps -

1. First normal Form (1NF)
2. Second normal form (2NF)
3. Third normal form (3NF)
4. Boyce-Codd Normal Form (BCNF)

First Normal Form (1 NF)

A relation is said to be in first normal form if it satisfies the following :

1. No multi-valued attribute
2. No composite attribute
3. Primary key identified

Here, the relationship is converted to either relation or a foreign key is used or relations are merged.

All the relations here are in first normal form because every attribute in all the relations is a single valued attribute.

Outcome of 1st normalization:

- Primary key has been identified in each table using closure property
- Composite attributes has been resolved
- Multi-valued attributes has been resolved

Open positions

job_id	designation	role	skill	exp_in_yrs	location

Job Application

application_id	job_id	name	mail_id	skill	exp_in_yrs	status

Employee performance System

emp_id	name	designation	project	skill	rating	assessment_date	remark

Leaves System

emp_id	month	unpaid_leave_hours	extra_hours_worked

Attendance System

emp_id	date	entry_time	exit_time	total_working_hours	effective_working_hours

Training

training_id	name	prerequisites	from_date	to_date

Trainee

emp_id	training_id	emp_name	test_score	grade

Resource Pool System

emp_id	name	mail_id	skill	designation	pref_location_1	pref_location_2

Human Resource management system

emp_id	open_job_id	job_application_id	training_id	emp_name	designation	skill	project

Second Normal Form (2NF)

The relation to be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has **No Partial Dependency,** i.e.**,** no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

Prime attribute (part of candidate key that determines anything else), also called partial dependency is identified, and eliminated. Because, 2nd NF is based on Full Functional dependency (key should determine all other attributes in a table).

In the relations defined for the human resource management system we do not see any relation where a proper subset of candidate key is determining non-prime attribute, hence no partial dependency is present.

job_id is a foreign key in relation ‘job application’ and primary key in relation ‘open position’ Similarly, training_id is foreign key in relation ‘trainee’ which refers to the primary key in relation ‘training’

Open positions

job_id	designation	role	skill	exp_in_yrs	location

Job Application

application_id	job_id	name	mail_id	skill	exp_in_yrs	status

Employee performance System

emp_id	name	designation	project	skill	rating	assessment_date

Leaves System

emp_id	month	unpaid_leave_hours	extra_hours_worked

Attendance System

emp_id	date	entry_time	exit_time	total_working_hours	effective_working_hours

Training

training_id	name	prerequisites	from_date	to_date

Trainee

emp_id	training_id	emp_name	test_score	grade

Resource Pool System

emp_id	name	mail_id	skill	designation	pref_location_1	pref_location_2

Human Resource management system

emp_id	open_job_id	job_application_id	training_id	emp_name	designation	skill	project

Third Normal Form (3NF)

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

A relation is in 3NF if at least one of the following condition holds in every non-trivial functional dependency X –> Y

X is a super key.

Y is a prime attribute (each element of Y is part of some candidate key).

Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency.

In the above created relations, for every X ⟶ Y, X is always the super key. Hence the relations are already in 3rd Normal Form.

Boyce-Codd Normal Form(BCNF)

A relation is in BCNF iff, X is superkey for every functional dependency (FD) X ⟶ Y in a given

relation.
Every 3NF is not BCNF but if a table is in BCNF then it is already in 3NF.
BCNF says every LHS of FD's must be the key of one of the tables. That is, every prime attribute should determine all other attributes in a table. Therefore, the given relations satisfy BCNF

Outcome of normalization:

Open positions

job_id	designation	role	skill	exp_in_yrs	location

Job Application

application_id	job_id	name	mail_id	skill	exp_in_yrs	status

Employee performance System

emp_id	name	designation	project	skill	rating	assessment_date	remark

Leaves System

emp_id	month	unpaid_leave_hours	extra_hours_worked

Attendance System

emp_id	date	entry_time	exit_time	total_working_hours	effective_working_hours

Training

training_id	name	prerequisites	from_date	to_date

Trainee

emp_id	training_id	emp_name	test_score	grade

Resource Pool System

emp_id	name	mail_id	skill	designation	pref_location_1	pref_location_2

Human Resource management system

emp_id	open_job_id	job_application_id	training_id	emp_name	designation	skill	project

Global Schema

Open positions

Attribute Name	**Attribute Type(Size in bytes)**
job_id	Char (20)
designation	Char (20)
role	Char (20)
skill	Char (20)
exp_in_yrs	Int(2)
location	Char (20)

Job Application

Attribute Name	**Attribute Type(Size in bytes)**
application_id	Char (20)
job_id	Char (20)
name	Char (20)
mail_id	Char (20)
skill	Char (20)
exp_in_yrs	float(4)
status	Char (20)

Employee performance system

Attribute Name	**Attribute Type(Size in bytes)**
emp_id	Char (20)
name	Char (20)
designation	Char (20)

skill	Char (20)
project	Char(20)
rating	float(4)
assessment_date	Date

Leaves System

Attribute Name	**Attribute Type(Size in bytes)**
emp_id	Char (20)
month	Char (10)
unpaid_leave_hours	Int(2)
extra_hours_worked	Int(2)

Attendance System

Attribute Name	**Attribute Type(Size in bytes)**
emp_id	Char (20)
date	Date
entry_time	Timestamp
exit_time	Timestamp
total_working_hours	Int (2)
effective_working_hours	Int (2)

Training

Attribute Name	**Attribute Type(Size in bytes)**
training_id	Char (20)
name	Char (50)
prerequisite	Char (50)
from_date	Date

to_date	Date

Trainee

Attribute Name	**Attribute Type(Size in bytes)**
emp_id	Char (20)
training_id	Char (20)
emp_name	Char (50)
test_score	Float (4)
grade	Char (5)

Resource Pool System

Attribute Name	**Attribute Type(Size in bytes)**
emp_id	Char (20)
name	Char (20)
mail_id	Char (20)
skill	Char (20)
designation	Char (10)
pref_location_1	Char (20)
pref_location_2	Char (20)

Human Resource management system

Attribute Name	**Attribute Type(Size in bytes)**
emp_id	Char (20)
open_job_id	Char (20)
job_application_id	Char (20)
training_id	Char (20)

emp_name	Char (20)
designation	Char (20)
skill	Char (20)
project	Char (20)

Fragmentation

Database tables are usually decomposed into smaller fragments for following reasons :

- When storage exhausted out
- For parallel processing
- For load balancing
- To improve query response time
- For better local processing
- Availability

Decomposed fragments are placed into some other site to facilitate query and optimize other quality of services. These fragments permit a number of transactions concurrently. Taking a copy of a relation and maintaining it in another site is called replication. One can combine fragmentation and replication for better service provision.

There are two kinds of fragmentation:

- Horizontal fragmentation
- Vertical fragmentation

They must satisfy the following properties :

- Completeness : All rows or columns must be present in at least one site
- Reconstruction : While reconstructing the relation, there should not be any inconsistency or loss of data
- Disjointness : Row or column must be present in at most one site, else it will lead to inconsistent data Fragmentation takes place in a relation based on the query and its frequency.

Fragmentation takes place in a relation based on the query and its frequency. The predicates

used in the query servers are an important statistical input for fragments.

Following are the lists of queries depicting the transactions in Payroll Management System :

1. **Select employee details whose performance is above average and attended required training with expected training test score.**
   SELECT emp_id, emp_name from Human_resource_management_System
   where emp_id in (SELECT emp_id from Employee_performance_system where rating > 3
   AND emp_id in (SELECT emp_id from Trainee where training_id='JV102' and
   test_score>70))
2. **Select employee details who is in a resource pool that is not allocated to any project and whose skill and preferred location matches with any open position.**
   SELECT emp_id, name from Resource_Pool_System
   WHERE (skill,pref_location_1) in (SELECT skill, location from Open_Positions) OR
   (skill,pref_location_2) in (SELECT skill, location from Open_Positions)
3. **Select employee details who has not taken any unpaid leaves in a given month and allocated to a given project.**
   SELECT emp_id, name from Human_resource_management_system
   WHERE project = 'project1' AND emp_id in (select emp_id from Leaves_Syestm WHERE
   unpaid_leave_hours = 0)
4. **Check for open positions if any at a given location and for particular skill.**
   SELECT job_id, role, designation from Open_positions WHERE skill='Java' AND
   location='Pune'
5. **List the name of candidates who all applied for an open position for a given designation.**
   SELECT name from Job_Application WHERE job_id in (SELECT job_id from Open_Positions
   where designation='team lead')
6. **Select list of training attended by an employee.**
   SELECT name from trainings where training_id in (SELECT training_id from Trainee WHERE
   emp_id='201')
7. **Look for unpaid leave hours in a given month for an employee.**
   SELECT unpaid_leaves_hours from Leaves_System where month='May2020' AND
   emp_id='201'

Horizontal Fragmentation

Horizontal fragmentation partitions the relation along its tuples of the relations. Every fragment will have the same number of attributes. There are two ways of doing it. Primary and derived horizontal fragmentation.
The relation R is fragmented horizontally using-

1. SImple predicate
   Example: location = ’Bangalore’
2. Minterm predicate
   Example: designation = ‘team lead’ AND performance_rating > 4

But, it is usually done using the predicate defined on the queries.
Derived horizontal fragmentation is possible between two tables having common attributes. It is implemented using semi joins.
In the above listed queries, the most frequently used queries are 4th and 5th ones as mostly employees and managers will check for open positions and people who applied for those positions respectively. Therefore, based on the relation and attributes, the predicates are:

1. **Job_id=value, skill=value, designation=value from Open_Positions**
   Example: Job_id=’dev123’, skill=’java’, designation=’junior developer’
2. **Application_id=value, skill=value, exp_in_yrs= value from Job_Application**
   Example: Application_id=’ap123’, skill=’data analytics’, exp_in_yrs= ‘4.5’
3. **training_id=value, name=vale, from_date=value, to_date=vale from Trainings**
   Example: training_id=’DB102’, name=’DBMS’, from_date=2020-12-01, to_date=’2020-12-06’
4. **emp_id=value, name=value, rating>value from Employee_Performance_System**
   Example: emp_id=’PS201’, name=’abc’, rating > 3
5. **emp_id=value, unpaid_leave_hours=value, Month=value from Leaves_System**
   Example: emp_id=’PS123’, unpaid_leave_hours=’90’, Month=’Oct2020’

We can horizontally fragment relations based on above predicates.
We can fragment Human Resource Management system relation based on Skill values:
Fragment 1: skill = java
Fragment 2: skill = data analytics
We can fragment Employee Performance System relation based on project value
Fragment 1: project = ‘project1’
Fragment 2: project = ‘project2’

Vertical Fragmentation

The vertical fragmentation of a relation R produces subschemas R1, R2, R2,...Rn. Each of which contains a subset of attributes, and only one fragment has a candidate key. To satisfy reconstruction,we need to use a joining attribute common between the sub schema. There are two methods to perform vertical fragmentation:

1. Grouping (bottom up): done by combining every two attributes at a time and takes a longtime if the number of attributes are over 100 to get desired fragments.
2. Splitting (top down) : given all attributes together is taken as a fragment and split them

as many fragments as you want to get. This is much quicker than the first method.

The list of vertical fragments are: based on the vertical fragmentation calculations, there are no possibilities and benefits in making vertical fragmentation.

Relation 1: Open positions

Attribute Usage Matrix:

Query No	job_id	designation	role	skill	exp_in_yrs	location
Q1	0	0	0	0	0	0
Q2	0	0	0	1	0	1
Q3	0	0	0	0	0	0
Q4	1	1	1	1	0	1
Q5	1	1	0	0	0	0
Q6	0	0	0	0	0	0
Q7	0	0	0	0	0	0

Query Access frequency matrix

Sites	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	job_id	designation	role	skill	exp_in_yrs	location
job_id	60	60	20	20	0	20

designation	60	60	20	0	0	0
role	20	20	20	20	0	0
skill	20	0	20	95	0	95
exp_in_yrs	0	0	0	0	0	0
location	20	0	0	95	0	95

For role:

CONT(A0, role,job_id) = 0+2*3200-0 =6400

CONT(job_id,role,designation) =2*3200+2*2800-2*7600= -3200

CONT(designation,role,skill) = 2*2800+2*2700-2*1600 = 7800

Here, CONT(designation,role,skill) is more

Attributes: job_id, designation,role

For skill:

CONT(A0, skill,job_id) = 0+2*5400-0 = 10800

CONT(job_id, skill, designation) = 2*(5400+1600-7600)= -1200

CONT(designation, skill, role) = 2*(1600+2700-2800) = 3000

CONT(role,skill,exp_in_yrs) = 2*(2700+0-0)=5400

CONT(A0, skill,job_id) is more

Attributes: skill,job_id,designation,role

For exp_in_yrs:

CONT(A0, exp_in_yrs,skill) =0+0-0 = 0

CONT(skill,exp_in_yrs,job_id)=0+0-5400*2 = -10800

CONT(job_id,exp_in_yrs,designation)= 0+0-2*7600 = -15200

CONT(designation,exp_in_yrs,role)=0+0-2*2800 = -5600

CONT(role,exp_in_yrs,location) = 0+0-2*2300 = - 4600

CONT(A0, exp_in_yrs,skill) is more

Attributes: exp_in_yrs,skill,job_id,designation,role

For location:

CONT(A0, location, exp_in_yrs) = 0

CONT(exp_in_yrs, location, skill)= 0+2*18450-0 =36900

CONT(skill, location, job_id) = 2*(18450+5000-5400) = 36100

CONT(job_id, location, designation) = 2*(5000+1200-7600)= -2800

CONT(designation, location, role)=2*(1200+2300-2800)= 1400

CONT(role, location, A7)=2*(2300+0-0)= 4600

CONT(exp_in_yrs, location, skill) is more

Attributes: exp_in_yrs, location, skill,job_id,designation,role

Partitions: {exp_in_yrs}{location, skill,job_id,designation,role}

TQ={} BQ={q2,q4,q5} OQ={}

z=0*135-0 = 0

Partitions:{exp_in_yrs,location}{ skill,job_id,designation,role}

z=0*0-135^2 = -18225

Partitions:{exp_in_yrs,location, skill}{job_id,designation,role}

z= 75*40-400 = 2600

Partitions:{exp_in_yrs,location, skill,job_id}{designation,role}

z=75*0-3600= -3600

Partitions:{exp_in_yrs,location, skill,job_id,designation}{role}

z=0*115-400 = -400

Maximum value is for **Partitions:{exp_in_yrs,location, skill}{job_id,designation,role}**

So, fragments will be f1= **{exp_in_yrs,location, skill}** and f2=**{job_id,designation,role}**

Relation 2: Job Application

Attribute Usage Matrix:

Query No	application_id	job_id	name	mail_id	skill	exp_in_yrs	status
Q1	0	0	0	0	0	0	0
Q2	0	0	0	0	0	0	0

Q3	0	0	0	0	0	0	0
Q4	0	0	0	0	0	0	0
Q5	0	1	1	0	0	0	0
Q6	0	0	0	0	0	0	0
Q7	0	0	0	0	0	0	0

Query Access frequency matrix

Sites	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	application_id	**job_id**	**name**	**mail_id**	**skill**	**exp_in_yrs**	**status**
application_id	0	0	0	0	0	0	0
job_id	0	40	40	0	0	0	0
name	0	40	40	0	0	0	0
mail_id	0	0	0	0	0	0	0
skill	0	0	0	0	0	0	0
exp_in_yrs	0	0	0	0	0	0	0

status	0	0	0	0	0	0	0

For name:

CONT(A0, name,application_id) = 0+0-0 =0

CONT(application_id,name,job_id) = 0+2*(3200)-0 = 6400

CONT(job_id,name,mail_id) = 2*(3200)+0-0 = 6400

Here, CONT(application_id,name,job_id) and CONT(job_id,name,mail_id) is more

We can choose any of these, we will go with CONT(job_id,name,mail_id)

Attributes: application_id, job_id, name

For mail_id:

CONT(A0, mail_id,application_id) = 0+0-0 = 0

CONT(application_id, mail_id, job_id) = 0

CONT(job_id, mail_id, name) = 0+0-6400= -6400

CONT(name,mail_id,skill) = 0

Maximum value is 0 ,for CONT(A0, mail_id,application_id), CONT(application_id, mail_id, job_id) and CONT(name,mail_id,skill)

Attributes: application_id, job_id, name, mail_id

For skill:

CONT(A0, skill,application_id) =0+0-0 = 0

CONT(application_id,skill,job_id) = 0

CONT(job_id,skill,name)= 0+0-2*3200 = -6400

CONT(name,skill,mail_id)=0+0-0 = 0

CONT(mail_id,skill,status) = 0

CONT(mail_id,skill,status) is more

Attributes: application_id, job_id, name, mail_id,skill

For status:

CONT(A0, status, application_id) = 0

CONT(application_id, status, job_id)= 0

CONT(job_id, status, name) = -6400

CONT(name, status, mail_id) = 0

CONT(mail_id, status, skill)=0

CONT(skill, status, A7)= 0

We can go with CONT(skill, status, A7) since value zero is maximum

Attributes: application_id, job_id, name, mail_id,skill, status

Partitions: {application_id}{job_id, name, mail_id,skill, status}

TQ={} BQ={q5} OQ={}

z=0*40-0 = 0

Partitions:{application_id,job_id}{ name, mail_id,skill, status}

z=0*0-40^2 = -1600

Partitions:{application_id,job_id,name}{ mail_id,skill, status}

z= 40*0-0 =0

Partitions:{application_id,job_id,name,mail_id}{skill, status}

z=40*0-0= 0

Partitions:{application_id,job_id,name,mail_id,skill}{ status}

z=40*0-0 = 0

Fragmentation not required as none of the z value is positive and greater than 0.

Relation 3: Employee performance System

Attribute Usage Matrix:

Query No	emp_id	name	designation	project	skill	rating	assessment_date
Q1	1	0	0	0	0	1	0
Q2	0	0	0	0	0	0	0
Q3	0	0	0	0	0	0	0
Q4	0	0	0	0	0	0	0
Q5	0	0	0	0	0	0	0
Q6	0	0	0	0	0	0	0

Q7	0	0	0	0	0	0	0

Query Access frequency matrix

Sites	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	emp_id	**name**	**designation**	**project**	**skill**	**rating**	**assessment_date**
emp_id	55	0	0	0	0	55	0
name	0	0	0	0	0	0	0
designation	0	0	0	0	0	0	0
project	0	0	0	0	0	0	0
skill	0	0	0	0	0	0	0
rating	55	0	0	0	0	55	0
assessment_date	0	0	0	0	0	0	0

For designation:

CONT(A0, designation,emp_id) = 0+0-0 =0

CONT(emp_id,designation,name) = 0

CONT(name,designation,project) = 0

We can choose any of these, we will go with CONT(name,designation,project)

Attributes: emp_id, name, designation

For project:

CONT(A0, project,emp_id) = 0+0-0 = 0

CONT(emp_id, project, name) = 0

CONT(name, project, designation) = 0

CONT(designation,project,skill) = 0

We can choose any of these, we will go with CONT(designation,project,skill)

Attributes: emp_id, name, designation, project

For skill:

No need to calculate for skill, because all values will be zero and the attribute sequence will remain the same as what happened above for the 'project' attribute.

Attributes: emp_id, name, designation, project,skill

For rating:

CONT(A0, rating, emp_id) = 0+2*6050-0 = 12100

CONT(emp_id, rating, name)= 12100+0-0= 12100

CONT(name, rating, designation) = 0

CONT(designation, rating, project) = 0

CONT(project, rating, skill)=0

CONT(skill, rating, assessment_date)= 0

We can choose CONT(A0, rating, emp_id) or CONT(emp_id, rating, name)

Attributes: rating,emp_id, name, designation, project,skill

For assessment_date:

CONT(A0, assessment_date,rating) = 0+0-0 = 0

CONT(rating, assessment_date,emp_id)= 0+0-12100= -12100

CONT(emp_id, assessment_date, name) = 0

CONT(name, assessment_date, designation) = 0

CONT(designation, assessment_date, project)=0

CONT(project, assessment_date, skill)= 0

CONT(skill, assessment_date, A8)= 0

We can choose CONT(skill, assessment_date, A8)

Attributes: rating,emp_id, name, designation, project,skill, assessment_date

Partitions: {rating}{emp_id, name, designation, project,skill, assessment_date}

TQ={} BQ={} OQ={q1}

z=0*0-110^2 = -1210

Partitions:{rating,emp_id}{name, designation, project,skill, assessment_date}

z=110*0-0 = 0

Partitions:{rating,emp_id, name}{ designation, project,skill, assessment_date}

z= 110*0-0 =0

Fragmentation not required as none of the z value is positive and greater than 0.

Relation 4: Leaves System

Attribute Usage Matrix:

Query No	emp_id	month	unpaid_leave_hours	extra_hours_worked
Q1	0	0	0	0
Q2	0	0	0	0
Q3	1	0	1	0
Q4	0	0	0	0
Q5	0	0	0	0
Q6	0	0	0	0
Q7	1	1	1	0

Query Access frequency matrix

Sites	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Recruitment system	0	50	0	20	40	0	0

Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	emp_id	**month**	**unpaid_leave_hours**	**extra_hours_worked**
emp_id	106	30	106	0
month	30	30	30	0
unpaid_leave_hours	106	30	106	0
extra_hours_worked	0	0	0	0

For unpaid_leave_hours:

CONT(A0, unpaid_leave_hours,emp_id) = 0+23372*2-0 =46744

CONT(emp_id,unpaid_leave_hours,month) = 46744+14520-14520 = 46744

CONT(month,unpaid_leave_hours,extra_hours_worked) = 14520+0-0 = 14520

We can choose a CONT(emp_id,unpaid_leave_hours,month)

Attributes: emp_id,unpaid_leave_hours,month

For extra_hours_worked:

CONT(A0, extra_hours_worked,emp_id) = 0

CONT(emp_id, extra_hours_worked, unpaid_leave_hours) = -46744

CONT(unpaid_leave_hours, extra_hours_worked, month) = -14520

CONT(month,extra_hours_worked,A5) = 0

We will go with CONT(month,extra_hours_worked,A5)

Attributes: emp_id,unpaid_leave_hours,month, extra_hours_worked

Partitions: {emp_id}{unpaid_leave_hours,month, extra_hours_worked}

TQ={} BQ={} OQ={q3,q7}

z=0*0-(106)^2 = -11236

Partitions:{emp_id,unpaid_leave_hours}{month, extra_hours_worked}

z=76*0-30^2 = -900

Partitions:{emp_id,unpaid_leave_hours,month}{extra_hours_worked}

z= 106*0-0 =0

Fragmentation not required as none of the z value is positive and greater than 0.

Relation 5: Attendance System

Attribute Usage Matrix:

Query No	**emp_id**	**date**	entry_time	exit_time	total_working_hours	effective_working_hours
Q1	0	0	0	0	0	0
Q2	0	0	0	0	0	0
Q3	0	0	0	0	0	0
Q4	0	0	0	0	0	0
Q5	0	0	0	0	0	0
Q6	0	0	0	0	0	0
Q7	0	0	0	0	0	0

No need to calculate fragmentation on this relation since all values in the affinity matrix are zero.

Relation 6: Training
Attribute Usage Matrix:

Query No	**training_id**	**name**	**prerequisites**	**from_date**	**to_date**
Q1	1	0	0	0	0
Q2	0	0	0	0	0
Q3	0	0	0	0	0

Q4	0	0	0	0	0
Q5	0	0	0	0	0
Q6	1	1	0	0	0
Q7	0	0	0	0	0

Query Access frequency matrix

Sites	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	training_id	**name**	**prerequisites**	**from_date**	**to_date**
training_id	90	35	0	0	0
name	35	35	0	0	0
prerequisites	0	0	0	0	0
from_date	0	0	0	0	0
to_date	0	0	0	0	0

For prerequisites:

CONT(A0, prerequisites,training_id) = 0+0-0 =0

CONT(training_id,prerequisites,name) = 0+0-8050 = -8050 < 0

CONT(name,prerequisites,from_date) = 0

We can choose a CONT(name,prerequisites,from_date)

Attributes: training_id,name,prerequisites

For from_date:

CONT(A0, from_date,training_id) = 0

CONT(training_id, from_date, name) = -8050

CONT(name, from_date, prerequisites) = 0

CONT(prerequisites,from_date,to_date) = 0

We will go with CONT(prerequisites,from_date,to_date)

Attributes: training_id,name,prerequisites, from_date

Similarly, for to_date, we will get zeroes and negative values only. So, no need to calculate for the attribute to_date.

Attributes: training_id,name,prerequisites, from_date, to_date

Partitions: {training_id}{name,prerequisites, from_date, to_date}

TQ={q1} BQ={} OQ={q6}

z=90*0-(35)^2 < 0 => negative value

Partitions:{training_id,name}{prerequisites, from_date, to_date}

z=125*0-0= 0

Partitions:{training_id,name,prerequisites}{ from_date, to_date}

z= 0

similarly , z value will be 0 for remaining partitions, Since, training_id and name will be in the same partition.

Hence, Fragmentation not required as none of the z values is positive and greater than 0.

Relation 7: Trainee

Attribute Usage Matrix:

Query No	emp_id	training_id	emp_name	test_score	grade
Q1	1	1	0	1	0
Q2	0	0	0	0	0

Q3	0	0	0	0	0
Q4	0	0	0	0	0
Q5	0	0	0	0	0
Q6	1	1	0	0	0
Q7	0	0	0	0	0

Query Access frequency matrix

Sites	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	emp_id	**training_id**	**emp_name**	**test_score**	**grade**
emp_id	90	90	0	55	0
training_id	90	90	0	55	0
emp_name	0	0	0	0	0
test_score	55	55	0	55	0
grade	0	0	0	0	0

For emp_name:

CONT(A0, emp_name,emp_id) = 0

CONT(emp_id,emp_name,training_id) = 0+0-38450 = -38450 < 0

CONT(training_id,emp_name,test_score) = 0+0-0 = 0

We can choose a CONT(training_id,emp_name,test_score)

Attributes: emp_id,training_id,emp_name

For test_score:

CONT(A0, test_score,emp_id) = 0+25850-0=25850

CONT(emp_id, test_score, training_id) = 25850+25850-38450 = -38450<0

CONT(training_id, test_score, emp_name) = 25850+0-0=25850

CONT(emp_name,test_score,grade) = 0

CONT(training_id, test_score, emp_name) is more

Attributes: emp_id,training_id, test_score, emp_name

For grade:

CONT(A0, grade,emp_id) = 0

CONT(emp_id, grade, training_id) = 0+0-38450 = -38450<0

CONT(training_id, grade, test_score) = 0+0-38450=-38450

CONT(test_score, grade, emp_name) = 0+0-0=-0

CONT(emp_name,grade,A6) = 0

CONT(emp_name,grade,A6) is more

Attributes: emp_id,training_id, test_score, emp_name,grade

Partitions: {emp_id}{training_id, test_score, emp_name,grade}

TQ={} BQ={} OQ={q1,q6}

z=0*0-(90)^2 = -8100

Partitions:{emp_id,training_id}{test_score, emp_name,grade}

z=35*0-55^2 = -3025

Partitions:{emp_id,training_id,test_score}{ emp_name,grade}

z= 125*0-0 =0

Similarly for all remaining partitions z value will be 0 (125*0-0).

Fragmentation not required as none of the z value is not positive and greater than 0.

Relation 8: Resource Pool System

Attribute Usage Matrix:

Query No	emp_id	name	mail_id	skill	designation	pref_location_1	pref_location_2
Q1	0	0	0	0	0	0	0
Q2	1	1	0	1	0	1	1
Q3	0	0	0	0	0	0	0
Q4	0	0	0	0	0	0	0
Q5	0	0	0	0	0	0	0
Q6	0	0	0	0	0	0	0
Q7	0	0	0	0	0	0	0

Query Access frequency matrix

Sites	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	emp_id	name	mail_id	skill	designation	pref_location_1	pref_location_2
emp_id	75	75	0	75	0	75	75
name	75	75	0	75	0	75	75
mail_id	0	0	0	0	0	0	0
skill	75	75	0	75	0	75	75

designation	0	0	0	0	0	0	0
pref_location_1	75	75	0	75	0	75	75
pref_location_2	75	75	0	75	0	75	75

For mail_id:

CONT(A0, mail_id,emp_id) = 0

CONT(emp_id,mail_id,name) = -28125

CONT(name,mail_id,skill) = -28125

CONT(A0, mail_id,emp_id) is more

Attributes: mail_id,emp_id,name

For skill:

CONT(A0, skill,mail_id) = 0

CONT(mail_id, skill, emp_id) = 28125

CONT(emp_id, skill, name) = 28125+28125-28125 = 28125

CONT(name,skill,designation) = 28125+0-0 = 28125

CONT(name,skill,designation) is more

Attributes: mail_id,emp_id,name,skill

For designation:

CONT(A0, designation,mail_id) = 0

CONT(mail_id, designation, emp_id) = -28125<0

CONT(emp_id, designation, name) = -28125 < 0

CONT(name,designation,skill) = -28125 <0

CONT(skill,designation,pref_location_1) = -28125<0

CONT(A0, designation,mail_id) is more

Attributes: designation,mail_id,emp_id,name,skill

For pref_location_1:

CONT(A0, pref_location_1,designation) = 0

CONT(designation, pref_location_1,mail_id) = 28125

CONT(mail_id, pref_location_1, emp_id) = 28125

CONT(emp_id, pref_location_1, name) = 28125+28125-28125=28125

CONT(name,pref_location_1,skill) = 28125

CONT(skill,pref_location_1,pref_location_2) = 28125

We can choose CONT(skill,pref_location_1,pref_location_2)

Attributes: designation,mail_id,emp_id,name,skill, pref_location_1

SImilarly we will get 0 and 28125 values for pref_location_2 attribute. So, we can have following attribute sequence

Attributes: designation,mail_id,emp_id,name,skill, pref_location_1, pref_location_2

Partitions: {designation}{mail_id,emp_id,name,skill, pref_location_1, pref_location_2}

TQ={} BQ={q2} OQ={}

z=0*75-0 = 0

Partitions:{designation,mail_id}{emp_id,name,skill, pref_location_1, pref_location_2}

z=0*75-0 = 0

Partitions:{designation,mail_id,emp_id}{name,skill, pref_location_1, pref_location_2}

z=0*0-75^2 = -5625

Similarly for remaining partitions we will get negative values as attributes accessed by query are in both partitions.

Fragmentation not required as none of the z value is not positive and greater than 0.

Relation 9: Human Resource management system

Attribute Usage Matrix:

Query No	emp_id	open_job_id	job_application_id	training_id	emp_name	designation	skill	project
Q1	1	0	0	0	1	0	0	0

Q2	0	0	0	0	0	0	0	0
Q3	1	0	0	0	1	0	0	1
Q4	0	0	0	0	0	0	0	0
Q5	0	0	0	0	0	0	0	0
Q6	0	0	0	0	0	0	0	0
Q7	0	0	0	0	0	0	0	0

Query Access frequency matrix

Sites	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
Recruitment system	0	50	0	20	40	0	0
Performance management system	10	0	0	0	0	0	0
Leave management system	0	0	44	0	0	0	30
Learning and talent management system	25	25	0	0	0	35	0
Human resource management system	20	0	32	0	0	0	0

Attribute affinity matrix

	emp_id	**open_job_id**	**job_application_id**	**training_id**	**emp_name**	**designation**	**skill**	**project**
emp_id	131	0	0	0	131	0	0	76
open_job_id	0	0	0	0	0	0	0	0
job_application_id	0	0	0	0	0	0	0	0
training_id	0	0	0	0	0	0	0	0

emp_name	131	0	0	0	131	0	0	76
designation	0	0	0	0	0	0	0	0
skill	0	0	0	0	0	0	0	0
project	76	0	0	0	76	0	0	76

For job_application_id:

CONT(A0, job_application_id,emp_id) = 0

CONT(emp_id,job_application_id,open_job_id) = 0

CONT(open_job_id,job_application_id,training_id) = 0

We can choose a CONT(open_job_id,job_application_id,training_id)

Attributes: emp_id,open_job_id,job_application_id

For training_id:

CONT(A0, training_id,emp_id) = 0

CONT(emp_id, training_id, open_job_id) = 0

CONT(open_job_id, training_id, job_application_id) = 0

CONT(job_application_id,training_id,emp_name) = 0

We will go with CONT(job_application_id,training_id,emp_name)

Attributes: emp_id,open_job_id,job_application_id,training_id

For emp_name:

CONT(A0, emp_name,emp_id) = 40098

CONT(emp_id, emp_name, open_job_id) = 40098

CONT(open_job_id, emp_name, job_application_id) = 0

CONT(job_application_id,emp_name,training_id) = 0

CONT(training_id,emp_name,designation) = 0

CONT(emp_id, emp_name, open_job_id) is more

Attributes: emp_id,emp_name,open_job_id,job_application_id,training_id

For designation:

CONT(A0, designation,emp_id) = 0

CONT(emp_id, designation, emp_name) = -40098 < 0

CONT(emp_name, designation, open_job_id) = 0

CONT(open_job_id, designation, job_application_id) = 0

CONT(job_application_id,designation,training_id) = 0

CONT(training_id,designation,skill) = 0

We can choose CONT(training_id,designation,skill)

Attributes: emp_id,emp_name,open_job_id,job_application_id,training_id,designation

For skill attribute, calculation will be similar like designation. So we can keep sequence as follows

Attributes: emp_id,emp_name,open_job_id,job_application_id,training_id,designation,skill

For project:

CONT(A0, project,emp_id) = 0+51376-0 =51376

CONT(emp_id, project, emp_name) = 51376+51376-40098 =62654

CONT(emp_name, project, open_job_id) = 51376

CONT(open_job_id, project, job_application_id) = 0

CONT(job_application_id,project,training_id) = 0

CONT(training_id,project,designation) = 0

CONT(designation,project,skill) = 0

CONT(skill,project,A9) = 0

CONT(emp_id, project, emp_name) is maximum

Attributes:

emp_id,project,emp_name,open_job_id,job_application_id,training_id,designation,skill

Partitions:

{emp_id}{project,emp_name,open_job_id,job_application_id,training_id,designation, skill}

TQ={} BQ={} OQ={q3,q1}

z=0*0-(131)^2 = -17161

Partitions:

{emp_id,project}{emp_name,open_job_id,job_application_id,training_id,designation, skill}

z=0*0-(131)^2 = -17161

Partitions:{emp_id,project,emp_name}{open_job_id,job_application_id,training_id, designation,skill}

z= 131*0-0 =0

Similarly for remaining partitions z value will be zero as only one partition contains attributes which are accessed by queries.

Fragmentation not required as none of the z value is positive and greater than 0.

Physical design

This physical design talks about how these fragments are stored in secondary memory. Based on the global schema defined in a section above, the size of all the attributes of all relations remains the same.

Assumptions

1. Fixed length records are considered for all relations.
2. The delimiter for each field is length of the field
3. Total number of records in respective relations (provided in below table)
4. Block size is 1024 bytes.
5. Record doesn't span over multiple blocks (this can be achieved by taking floor function during calculating number of records per block to restrict single record doesn't span over blocks).
6. Block pointer (Bp) size is 4 bytes
7. Average Seek Time (S) is 20 ms irrespective of any site
8. Average Disk rotation time (Latency) Time (L) is 10 ms irrespective of any site.
9. Block transfer rate (Tr) is 0.5 ms irrespective of any site

Fragment	Relation	No. of	Record size	Blocking	No.of blocks

		records(N)	**in bytes (S)**	**factor(F= block size/S)**	**(N/F)**
1	Open Positions	100	20	51	2
2	Open Positions	100	20	51	2
3	Job Application	300	15	68	5
4	Employee Performance System	150	10	102	1
5	Employee Performance System	100	30	34	3
6	Leave System	250	40	25	10
7	Attendance System	250	15	68	4
8	Training	50	14	73	1
9	Trainee	100	10	102	1
10	Resource Pool System	200	35	29	7
11	Human Resource Management System	300	55	18	17
12	Human Resource Management System	500	20	51	10

Considering the assumption we can calculate easily the size of a single record (tuple) of every relation with the help of Global Schema. The above table gives the number of records in each relation, size of each record, blocking factor for a particular block of that relation and number of blocks required to store the entire relation. Having records on secondary storage, if you want to access them faster, then you need indexing. If a database is frequently queried and it is too large then it is supposed to have an index to increase performance. There are various indexes used in databases.

Here, we consider the following indexing scheme:

- Primary Index
- Clustered Index
- Secondary index.

Based on the query, we decide what type of indexing file. The below table provides the details of fragment, relation name, the attribute on which the index file is built and the type of index file built. For fragments from horizontal procedure, cluster index would be beneficial.

Fragments 4, 8 and 9 are fit into one block. Therefore, using indexing is not really needed.

Fragment	**Relation**	**Index Type**	**Indexing attribute(s)**	**Is key?**
1	Open Positions	Cluster	job_id	yes
2	Open Positions	Cluster	job_id	yes
3	Job Application	Primary	application_id	no
4	Employee Performance System	NA	NA	No
5	Employee Performance System	Cluster	emp_id	yes

6	Leave System	Cluster	emp_id	yes
7	Attendance Syster	Primary	emp_id	yes
8	Training	NA	NA	No
9	Trainee	NA	NA	No
10	Resource Pool System	Primary	emp_id	Yes
11	Human Resource Management System	Cluster	emp_id	yes
12	Human Resource Management System	Cluster	emp_id	yes

Using above information we can now calculate number of block accesses required with indexing
Index size per record = block pointer size (4) + indexing attribute size
No. of index records per block = Block size(1024) / Index size per record

Fragment	Relation	No. of records	No. of data blocks	Index size per record (R)	No. of index records per block	No. of index blocks	No. of block access without indexing	No. of block access with indexing
1	Open Positions	100	2	4+10	73	2	2	2
2	Open Positions	100	2	4+10	73	2	2	2
3	Job Application	300	5	4+ 6	102	4	5	3
4	Employee Performance	150	1	NA	NA	NA	1	1

	System							
5	Employee Performance System	100	3	4+4	128	1	3	1
6	Leave Syster	250	10	4+4	128	2	10	2
7	Attendance System	250	4	4+4	128	2	4	2
8	Training	50	1	NA	NA	NA	1	1
9	Trainee	100	1	NA	NA	NA	1	1
10	Resource Pool System	200	7	4+4	128	2	7	2
11	Human Resource Managemer System	300	17	4+4	128	3	17	3
12	Human Resource Managemer System	500	10	4+4	128	4	10	3

Indexing the data file definitely reduces the number of block accesses needed to find a particular record from the data file. The complete statistics are shown in the above table.

Access Time to local query: Next we will calculate the time taken to query each relation considering the relation is locally present. Even though in our sample SQL's are just read still provided the Local (keyword local because it's not yet distributed)

Query Time and Local Update Time formulae :

i. Local Query Time = (Seek Time + Latency + Block Transfer Time) * N

ii. Local Update Time = (Seek Time + Latency + Block Transfer Time) * N * 2 Where,

- N is number of disk block access, which depends on the relation (we already calculated this above and will consider indexed logic no. of block access)
- *2 is included in the Update time, since the data block has to be fetched into memory from the disk, updated and then written back to the disk

Access Time to Remote query:

Let us consider the distance between sites. Assume that each site is located at some distance say 100kms from the other site and the speed of the transmission media connecting the sites is 10^7 meters/second. Propagation delay between the sites is computed as below.

Propagation Delay = (Distance between sites)/(Speed of Transmission media)

= 100 * 10^3/ 2 *10^6 which will be = 0.05s = 50 ms

Let us assume bandwidth of the network as 1MBps and data is exchanged between sites in form of packets. Package size is assumed to be 1500bytes. Transmission Time for a packet is given by,

Packet Transmission Time = (Size of packet) / Bandwidth = (1500 B) / (10^6 B/s) = 0.0015s , approximately equal to 2ms

Remote Query Time = Local Query Time + 2 * Propagation Delay + Packet Transmission Time

Remote Update Time = Local Update Time + 2 * Propagation Delay

Packet Transmission Time is included in Remote Query Time because; the result of the query will contain some data which is not negligible. But this data depends on the query, so it is assumed to be one packet, on an average. Using the formulations made above, the below table can be constructed.

Fragment	**Relation**	**No. of Records**	**No. of data blocks**	**No. of block access with indexing**	**Local Query Time (ms) = (S + L + Tr) * N**	**Remote Query Time (ms)**
1	Open Positions	100	2	2	61	163
2	Open	100	2	2	61	163

	Positions					
3	Job Application	300	5	3	91.5	193.5
4	Employee Performance System	150	1	1	30.5	132.5
5	Employee Performance System	100	3	1	30.5	132.5
6	Leave System	250	10	2	61	163
7	Attendance System	250	4	2	61	163
8	Training	50	1	1	30.5	132.5
9	Trainee	100	1	1	30.5	132.5
10	Resource Poo System	200	7	2	61	163
11	Human Resource Management System	300	17	3	91.5	193.5
12	Human Resource Management System	500	10	3	91.5	193.5

Total storage required

The disk storage required to store data = No. of blocks * size of each block

We assumed size of each block to be 1024 Bytes (1KB)

Disk storage = 67 * 1 KB = 67 KB

Work space area

Work space area is the buffer size required while calculation query results to store temporary results retrieved from one query and that to be used in another query.

In the query list mentioned above do not contain any such query which requires a buffer while calculating the result. So, no work space area required.

Allocation and replication

For allocating the fragments to sites we considered the Redundant All Beneficial Sites method.

Transaction table is given below: consider there are four sites: S1, S2, S3, S4

Query	Originating site	Frequency	Fragment access
Q1	S1	200	F1 - 4R
Q1	S2	100	F2 - 3R
Q1	S3	300	F3 - 4R
Q1	S4	400	F6 - 5R
Q2	S1	150	F7 - 2R
Q2	S2	50	F8 - 3R
Q2	S3	200	F11 -5R
Q2	S4	500	F5 - 7R
Q3	S1	150	F4-5R
Q3	S2	500	F11- 6R
Q3	S3	300	F9 - 3R
Q3	S4	50	F10 - 4R
Q4	S1	600	F6 - 4R
Q4	S2	100	F6 - 5R
Q4	S3	250	F7 - 2R

Q4	S4	150	F8 - 3R
Q5	S1	300	F9 - 7R
Q5	S2	400	F1 - 4R
Q5	S3	500	F2 - 3R
Q5	S4	600	F3 - 4R
Q6	S1	250	F6 - 5R
Q6	S2	100	F7 - 2R
Q6	S3	300	F8 - 3R
Q6	S4	200	F11 -5R
Q7	S1	700	F5 - 7R
Q7	S2	150	F4-5R
Q7	S3	50	F3 - 4R
Q7	S4	100	F2 - 2R

Since our sample queries are related only read (not update) will calculate only Benefit Computation (not Cost computation) of placing a fragment at a particular site. Let us proceed to Benefit Computation. Benefit computation is based on read queries. The benefit of placing each fragment at each site is given in the below table.

Fragments	Originating Sites	Query	No. of Reads * Frequency * (Remote Time – Local Time)	Benefit (ms)
F1	S1	Q1	4*200*102	81600
F1	S2	Q5	4*400*102	163200

F1	S3	none	none	0
F1	S4	none	none	0
F2	S1	none	none	0
F2	S2	Q1	3*100*102	30600
F2	S3	Q5	3*500*102	153000
F2	S4	Q7	2*100*102	20400
F3	S1	none	0	0
F3	S2	none	0	0
F3	S3	Q1,Q7	4*300*102+ 4*50*102	142800
F3	S4	Q5	4*600*102	244800
F4	S1	Q3	5*150*102	76500
F4	S2	Q7	5*150*102	76500
F4	S3	none	none	0
F4	S4	none	none	0
F5	S1	Q7	7*700*102	499800
F5	S2	none	none	0
F5	S3	none	none	0
F5	S4	Q2	7*500*102	357000
F6	S1	Q4,Q6	4*600*102 +5*250*102	372300
F6	S2	Q4	5*100*102	51000
F6	S3	Q1	5*400*102	204000
F6	S4	none	none	0
F7	S1	Q2	2*150*102	30600

F7	S2	Q6	2*100*102	20400
F7	S3	Q4	2*250*102	51000
F7	S4	none	none	0
F8	S1	none	none	0
F8	S2	Q2	3*50*102	15300
F8	S3	Q6	3*300*102	91800
F8	S4	Q4	3*150*102	45900
F9	S1	Q5	7*300*102	214200
F9	S2	none	none	0
F9	S3	Q3	3*300*102	91800
F9	S4	none	none	0
F10	S1	none	none	0
F10	S2	none	none	0
F10	S3	none	none	0
F10	S4	Q3	4*50*102	20400
F11	S1	none	none	0
F11	S2	Q3	6*500*102	306000
F11	S3	Q2	5*200*102	102000
F11	S4	Q6	5*200*102	102000

From the above table we can put fragment at site where benefit is maximum-

1. F1 is at S2
2. F2 is at S3
3. F3 is at S4

4. F4 is at S1
5. F5 is at S1
6. F6 is at S3
7. F7 is at S3
8. F8 is at S3
9. F9 is at S1
10. F10 is at S4
11. F11 is at S2