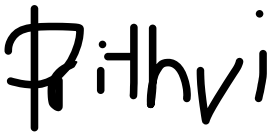
CS549 Coursework

by:

Group number		18	
Student ID	**Student email**	**Student name**	**Student's signature**
202487076	**mohan.loganathan.2024@uni.strath.ac.uk**	**Mohan Raj Loganathan**	
202480548	**harshan.rethinlu-selvakumar.2023@uni.strath.ac.uk**	**Harshan Rethinavelu Selvakumar**	
202479802	**prithviraj.ramesh.2023@uni.strath.ac.uk**	**Prithviraj Ramesh**	
202468855	**manoj.dharmaraj.2023@uni.strath.ac.uk**	**Manoj Kumar Dharmaraj**	

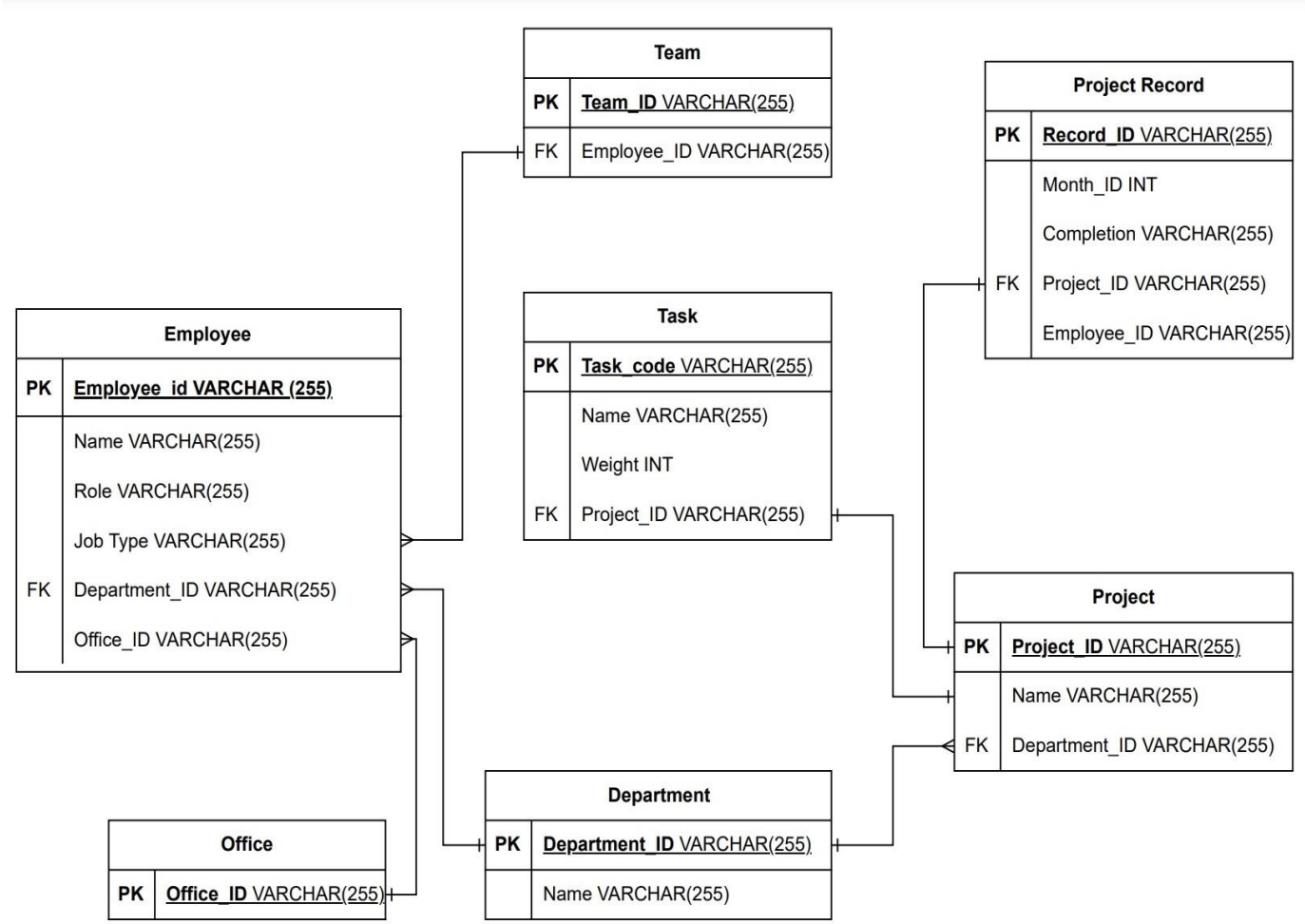
Department of
Computer and Information Sciences

March, 2024

Contents

1. Relational database

1.1 Entity relationship diagram



1.2 SQL statements to implement ERD

Department Table

CREATE TABLE Department(

```
    Department_ID VARCHAR(255) PRIMARY KEY,
    Name VARCHAR(255)
);
```

Office Table

```
CREATE TABLE Office(
    Office_ID VARCHAR(255) PRIMARY KEY,
    Location VARCHAR(255)
);
```

Employee Table

```
CREATE TABLE Employee(
    Employee_ID VARCHAR(255) PRIMARY KEY,
    Name VARCHAR(255),
    Role VARCHAR(255),
    Job_Type VARCHAR(255),
    Department_ID VARCHAR(255),
    Office_ID VARCHAR(255),
    FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID),
    FOREIGN KEY (Office_ID) REFERENCES Office(Office_ID)
);
```

Project Table

```
CREATE TABLE Project(
    Project_ID VARCHAR(255) PRIMARY KEY,
    Name VARCHAR(255),
    Department_ID VARCHAR(255),
    FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
);
```

Task Table

```
CREATE TABLE Task(
```

```
  Task_Code VARCHAR(255) PRIMARY KEY,
  Name VARCHAR(255),
  Weight INT,
  Project_ID VARCHAR(255),
  FOREIGN KEY (Project_ID) REFERENCES Project(Project_ID)
);
```

Project Record Table

```
CREATE TABLE Project_Record(
  Record_ID VARCHAR(255),
  Month_ID INT,
  Completion_ VARCHAR(255),
  Project_ID VARCHAR(255),
  Employee_ID VARCHAR(255),
  FOREIGN KEY (Project_ID) REFERENCES Project(Project_ID),
  FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID)
);
```

Team Table

```
CREATE TABLE Team(
  Team_ID VARCHAR(255)
);
CREATE TABLE Team_Members(
  Team_ID VARCHAR(255),
  Employee_ID VARCHAR(255),
  PRIMARY KEY (Team_ID, Employee_ID),
  FOREIGN KEY (Team_ID) REFERENCES Team(Team_ID),
  FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID));
```

1.3 SQL statements to insert data

Department Table

```
INSERT INTO `Department`(`Department_ID`, `Name`) VALUES ('Dept01','Group18 Technical'),('Dept02','Group18 Management'),('Dept03','Group19 Technical'),('Dept04','Group19 Management'),('Dept05','Group20 Technical');
```

Employee Table

```
INSERT INTO `Employee`(`Employee_ID`, `Name`, `Role`, `Job_Type`, `Department_ID`, `Office_ID`) VALUES ('Emp01','Mohan','App Developer','Software','Dept01','Office01'),('Emp02','Harshan','App Developer','Software','Dept01','Office01'),('Emp03','Prithvi','App Developer','Software','Dept01','Office01'),('Emp04','Manoj','App Developer','Software','Dept01','Office01'),('Emp05','Valerio','App Developer','Software','Dept01','Office01');
```

Office Table

```
INSERT INTO `Office`(`Office_ID`, `Location`) VALUES ('Office01','Glasgow'),('Office02','Chennai'),('Office03','Banglore'),('Office04','London'),('Office05','New York')
```

Project Table

```
INSERT INTO `Project`(`Project_ID`, `Name`, `Department_ID`) VALUES ('Proj01','DIS App','Dept01'),('Proj02','ML App','Dept01'),('Proj03','BA App','Dept01'),('Proj04','DS App','Dept01'),('Proj05','Web Design App','Dept01');
```

Project Record Table

```
INSERT INTO `Project_Record`(`Record_ID`, `Month_ID`, `Completion_`, `Project_ID`, `Employee_ID`) VALUES ('Record01','04','50%','Proj01','Emp01'),('Record02','04','50%','Proj01','Emp02'),('Record03','04','505','Proj01','Emp03'),('Record42','05','100%','Proj01','Emp01'),('Record05','05','100%','Proj01','Emp02')
```

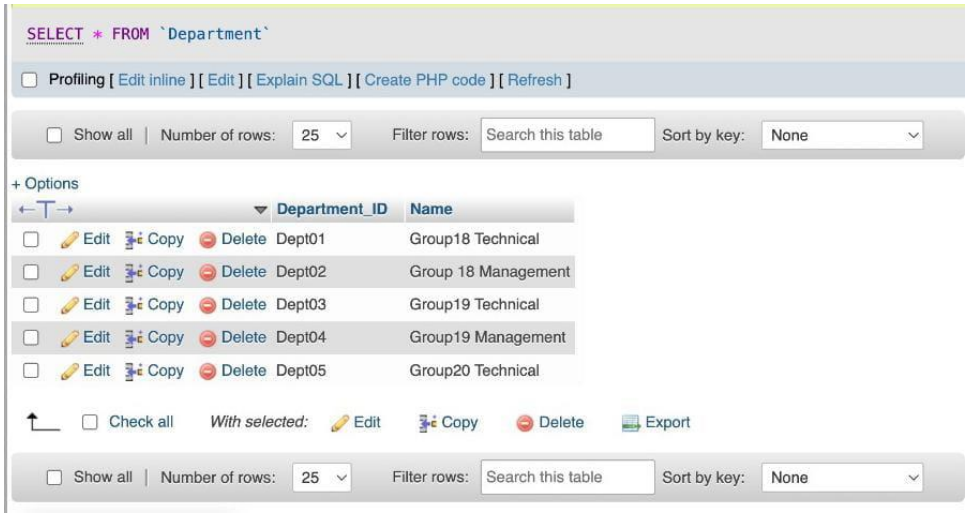
Task Table

```
INSERT INTO `Task`(`Task_Code`, `Name`, `Weight`, `Project_ID`) VALUES ('Task01','Front-End','30','Proj01'),('Task02','Back-End','30','Proj01'),('Task03','API','20','Proj01'),('Task04','User Testing','10','Proj01'),('Task05','Automated Testing','10','Proj01')
```

Team Table

```
INSERT INTO `Team`(`Team_ID`) VALUES ('Team01'),('Team02'),('Team03'),('Team04'),('Team05')
```

1.4 Tables with data shown.



```
SELECT * FROM `Department`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

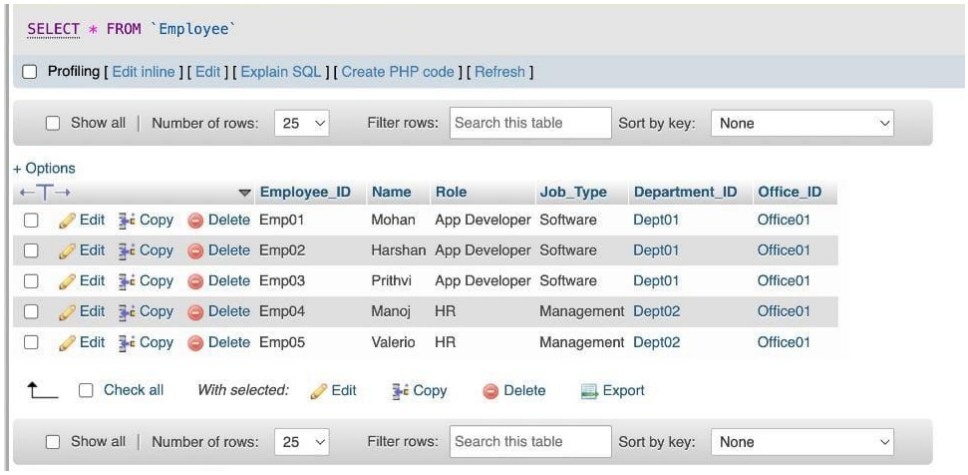
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Department_ID	Name
Edit Copy Delete	Dept01	Group18 Technical
Edit Copy Delete	Dept02	Group 18 Management
Edit Copy Delete	Dept03	Group19 Technical
Edit Copy Delete	Dept04	Group19 Management
Edit Copy Delete	Dept05	Group20 Technical

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None



```
SELECT * FROM `Employee`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

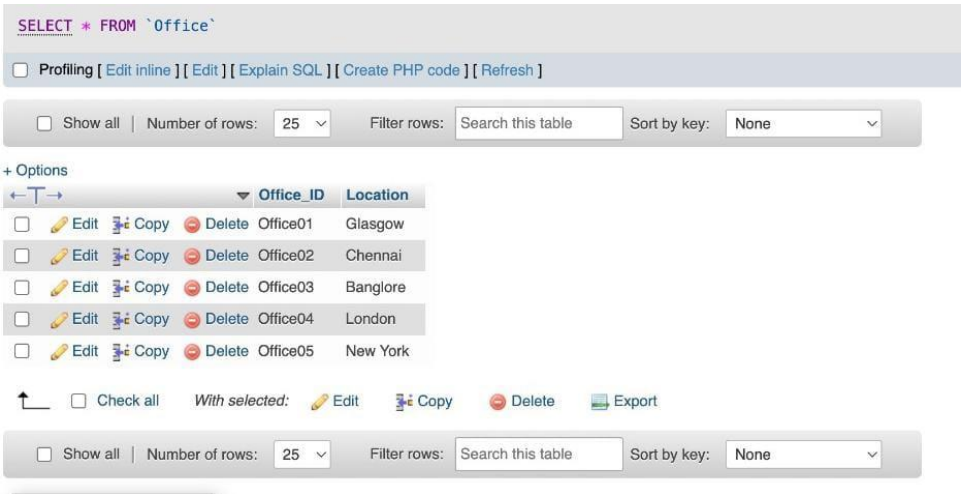
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Employee_ID	Name	Role	Job_Type	Department_ID	Office_ID
Edit Copy Delete	Emp01	Mohan	App Developer	Software	Dept01	Office01
Edit Copy Delete	Emp02	Harshan	App Developer	Software	Dept01	Office01
Edit Copy Delete	Emp03	Prithvi	App Developer	Software	Dept01	Office01
Edit Copy Delete	Emp04	Manoj	HR	Management	Dept02	Office01
Edit Copy Delete	Emp05	Valerio	HR	Management	Dept02	Office01

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None



```
SELECT * FROM `Office`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Office_ID	Location
Edit Copy Delete	Office01	Glasgow
Edit Copy Delete	Office02	Chennai
Edit Copy Delete	Office03	Banglore
Edit Copy Delete	Office04	London
Edit Copy Delete	Office05	New York

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

```
SELECT * FROM `Project`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Project_ID	Name	Department_ID	Type	Category
Edit Copy Delete	Proj01	DIS App	Dept01	Mobile App	Educational Software
Edit Copy Delete	Proj02	ML App	Dept01	Mobile App	Educational Software
Edit Copy Delete	Proj03	BA App	Dept01	Mobile App	Educational Software
Edit Copy Delete	Proj04	DIS Website	Dept01	Web App	Educational Software
Edit Copy Delete	Proj05	ML Website	Dept01	Web App	Educational Software

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

```
SELECT * FROM `Project_Record` ORDER BY `Completion_` DESC
```

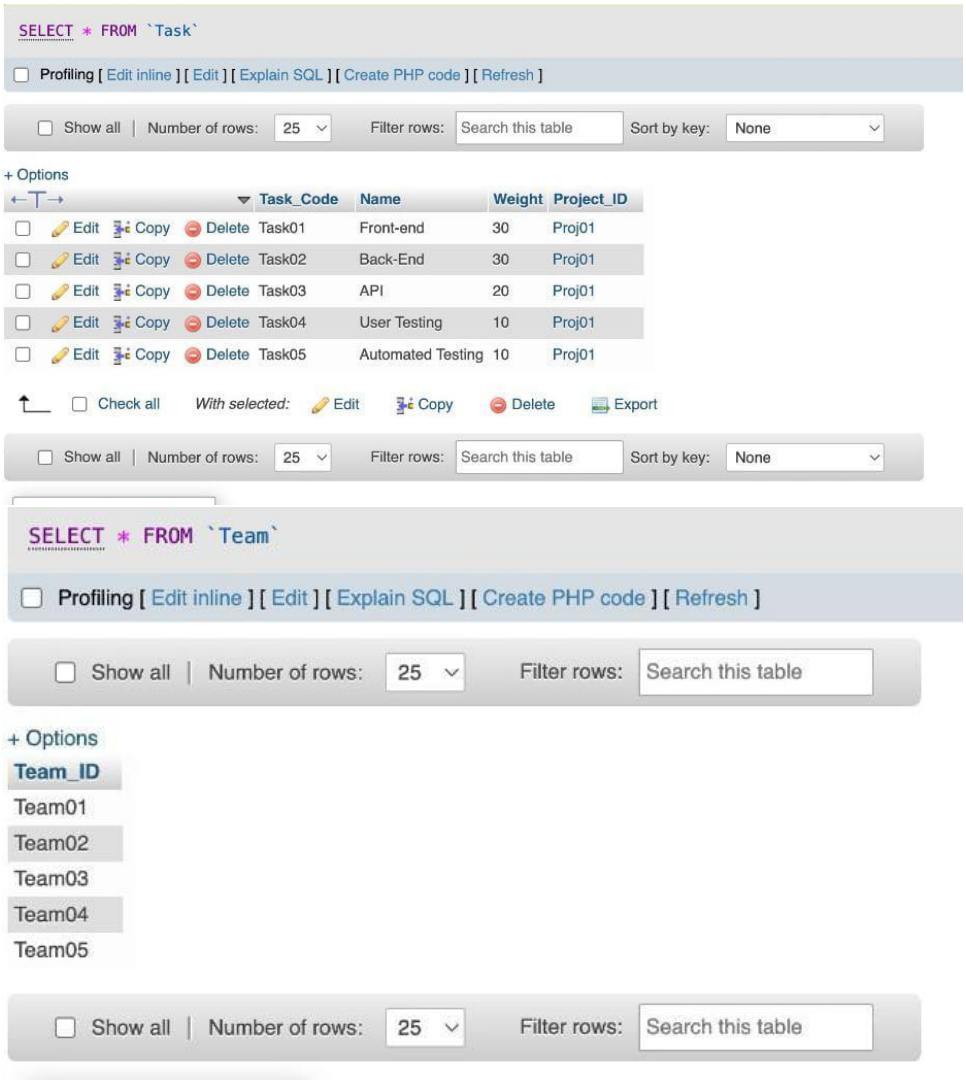
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

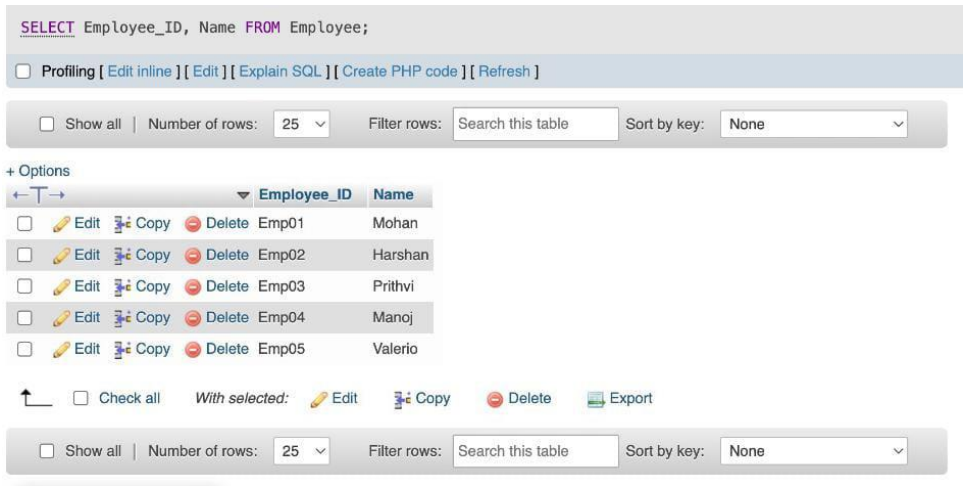
Record_ID	Month_ID	Completion_ 1	Project_ID	Employee_ID
Record01	4	Pending	Proj01	Emp01
Record02	4	Pending	Proj01	Emp02
Record03	4	Pending	Proj01	Emp03
Record42	5	Completed	Proj01	Emp01
Record05	5	Completed	Proj01	Emp02

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

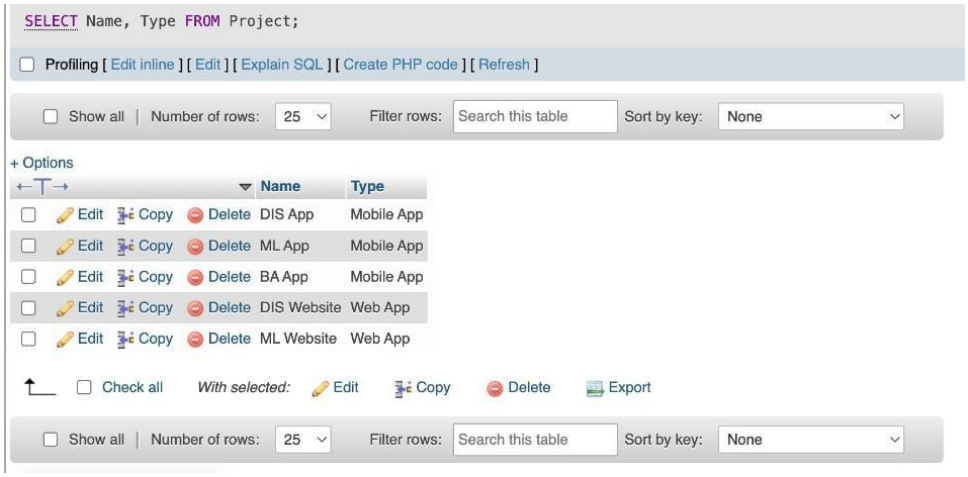


1.5 Five SQL queries with results

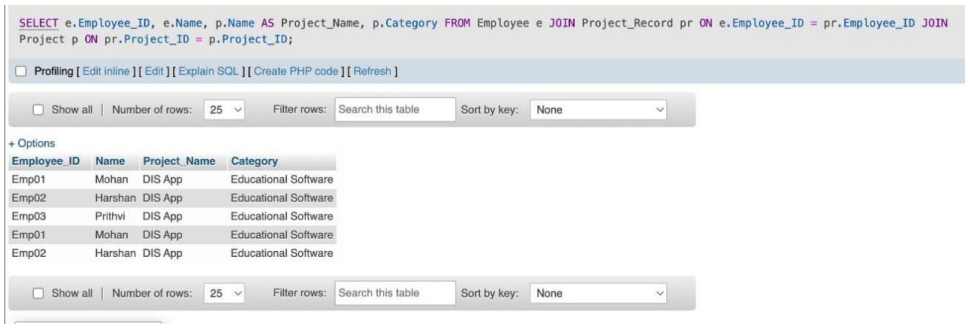
1. List the ID and name of all employees



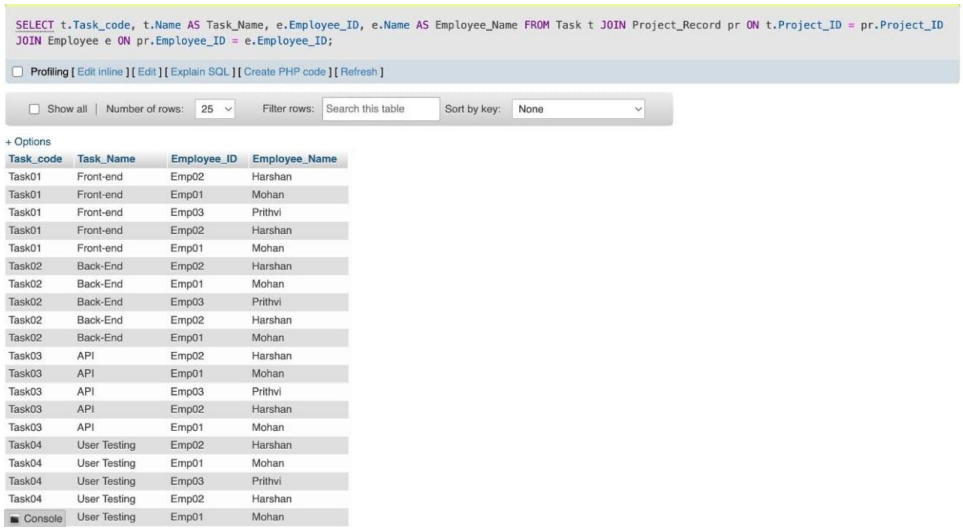
2. List the name of all projects together with their type



3. List the ID and name of all employees together with the project name and category they work on.



4. List the ID and name of all tasks with the name and ID of the employee who works on the task.



5. List the ID and name of all employees with the office number they are based in.

```
SELECT e.Employee_ID, e.Name, o.Office_ID FROM Employee e JOIN Office o ON e.Office_ID = o.Office_ID;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

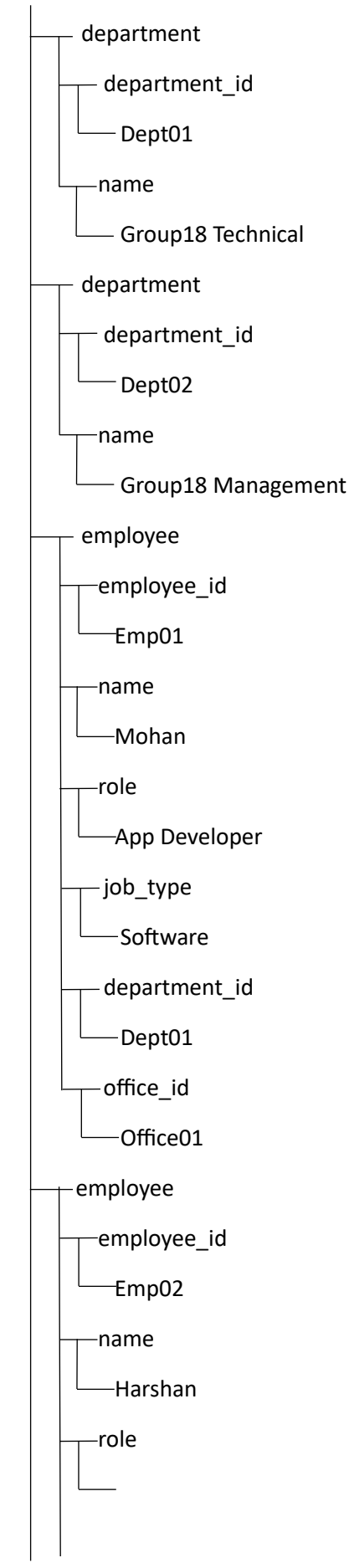
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

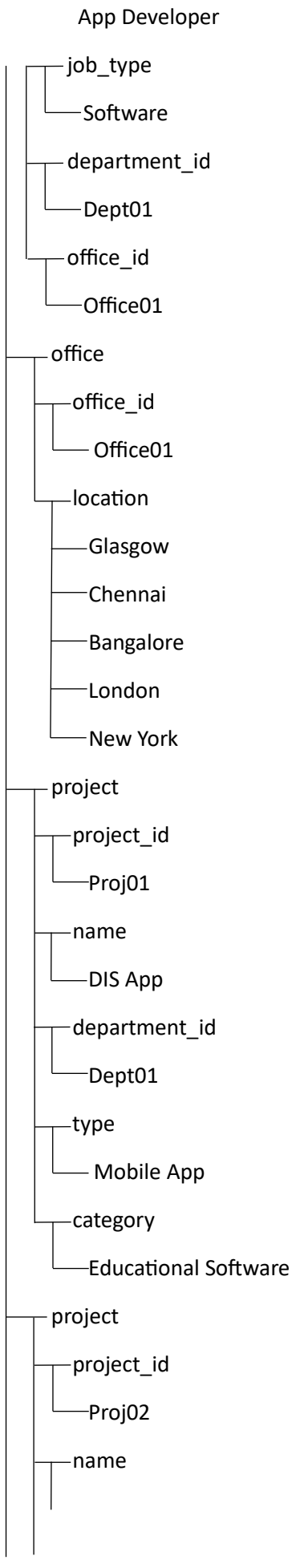
+ Options

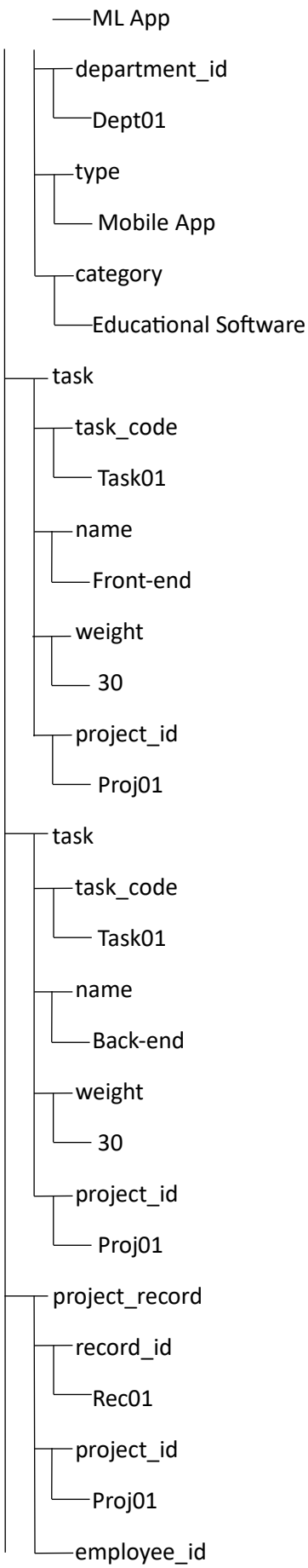
Employee_ID	Name	Office_ID
Emp01	Mohan	Office01
Emp02	Harshan	Office01
Emp03	Prithvi	Office01
Emp04	Manoj	Office01
Emp05	Valerio	Office01

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

2. XML Tree Diagram:




App Developer
job_type
Software
department_id
Dept01
office_id
Office01
office
office_id
Office01
location
Glasgow
Chennai
Bangalore
London
New York
project
project_id
Proj01
name
DIS App
department_id
Dept01
type
Mobile App
category
Educational Software
project
project_id
Proj02
name


ML App
department_id
Dept01
type
Mobile App
category
Educational Software
task
task_code
Task01
name
Front-end
weight
30
project_id
Proj01
task
task_code
Task01
name
Back-end
weight
30
project_id
Proj01
project_record
record_id
Rec01
project_id
Proj01
employee_id

1.1 2.2 XML DTD with explanation

1.2 2.3 Five XQuery queries with results

List the ID and name of all employees.

```
for $emp in //employee
return
 <employee>
  <id>{ $emp/employee_id/text() }</id>
  <name>{ $emp/name/text() }</name>
 </employee>
```

Output:

Emp01

Mohan

Emp02

Harshan

Emp03

Prithvi

Emp04

Manoj

Emp05

Valerio

List the name of all projects along with their type.

```
for $proj in //project
return
 <project>
  <name>{data($proj/name)}</name>
  <type>{data($proj/type)}</type>
 </project>
```

Output:

DIS App

Mobile App

ML App

Mobile App

BA App

Mobile App

DIS Website

Web App

ML Website

Web App

List the ID and name of all employees together with the project name and project category they work on.

```
for $emp in //employee,
   $rec in //project_record[employee_id = $emp/employee_id],
   $proj in //project[project_id = $rec/project_id]
return
 <employee_project>
   <employee_id>{data($emp/employee_id)}</employee_id>
   <employee_name>{data($emp/name)}</employee_name>
   <project_name>{data($proj/name)}</project_name>
   <project_category>{data($proj/category)}</project_category>
 </employee_project>
```

Output:

Emp01

Mohan

DIS App

Educational Software

Emp01

Mohan

DIS App

Educational Software

Emp02

Harshan

DIS App

Educational Software

Emp02

Harshan

DIS App

Educational Software

Emp03

Prithvi

DIS App

Educational Software

List the ID and name of all tasks with the name and ID of the employee who works on the task.

```
for $task in //task,
   $rec in //project_record[project_id = $task/project_id],
   $emp in //employee[employee_id = $rec/employee_id]
return
 <task_employee>
   <task_id>{data($task/task_code)}</task_id>
   <task_name>{data($task/name)}</task_name>
   <employee_id>{data($emp/employee_id)}</employee_id>
   <employee_name>{data($emp/name)}</employee_name>
 </task_employee>
```

Output:

Task01

Front-end

Emp01

Mohan

Task01

Front-end

Emp02

Harshan

Task01

Front-end

Emp03

Prithvi

Task01

Front-end

Emp01

Mohan

Task01

Front-end

Emp02

Harshan

Task02

Back-End

Emp01

Mohan

Task02

Back-End

Emp02

Harshan

Task02

Back-End

Prithvi

Task02

Back-End

Emp01

Mohan

Task02

Back-End

Emp02

Harshan

Task03

API

Emp01

Mohan

Task03

API

Emp02

Harshan

Task03

API

Emp03

Prithvi

Task03

API

Emp01

Mohan

Task03

API

Emp02

Harshan

Task04

User Testing

Emp01

Mohan

Task04

User Testing

Emp02

Harshan

Task04

User Testing

Emp03

Prithvi

Task04

User Testing

Emp01

Mohan

Task04

User Testing

Emp02

Harshan

List the ID and name of all employees with the office number they are based in.

```
for $emp in //employee,
   $office in //office[office_id = $emp/office_id]
return
 <employee_office>
   <employee_id>{data($emp/employee_id)}</employee_id>
   <employee_name>{data($emp/name)}</employee_name>
   <office_id>{data($office/office_id)}</office_id>
 </employee_office>
```

Output:

Emp01

Mohan

Office01

Emp02

Harshan

Office01

Emp03

Prithvi

Office01

Emp04

Manoj

Office01

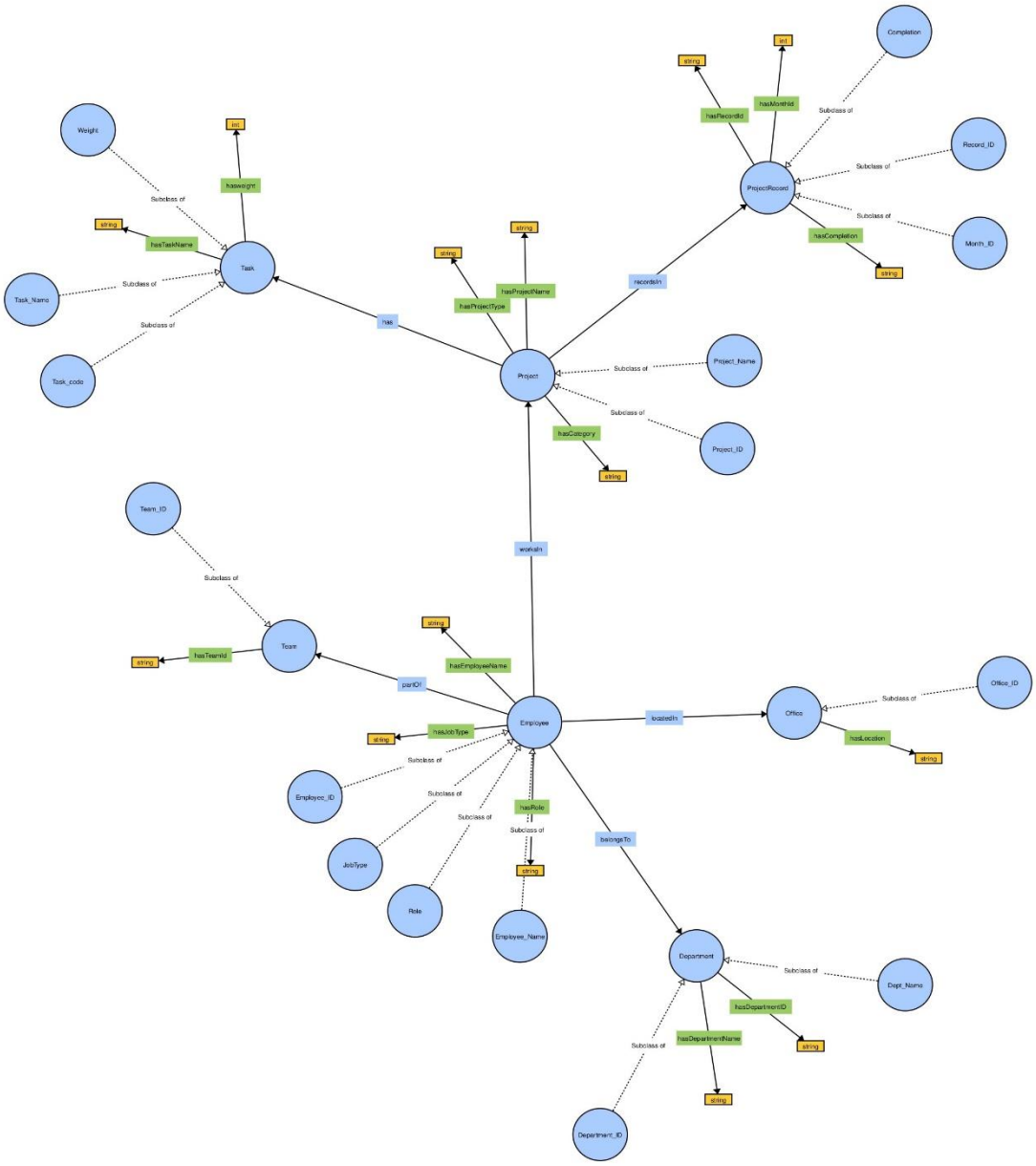
Emp05

Valerio

Office01

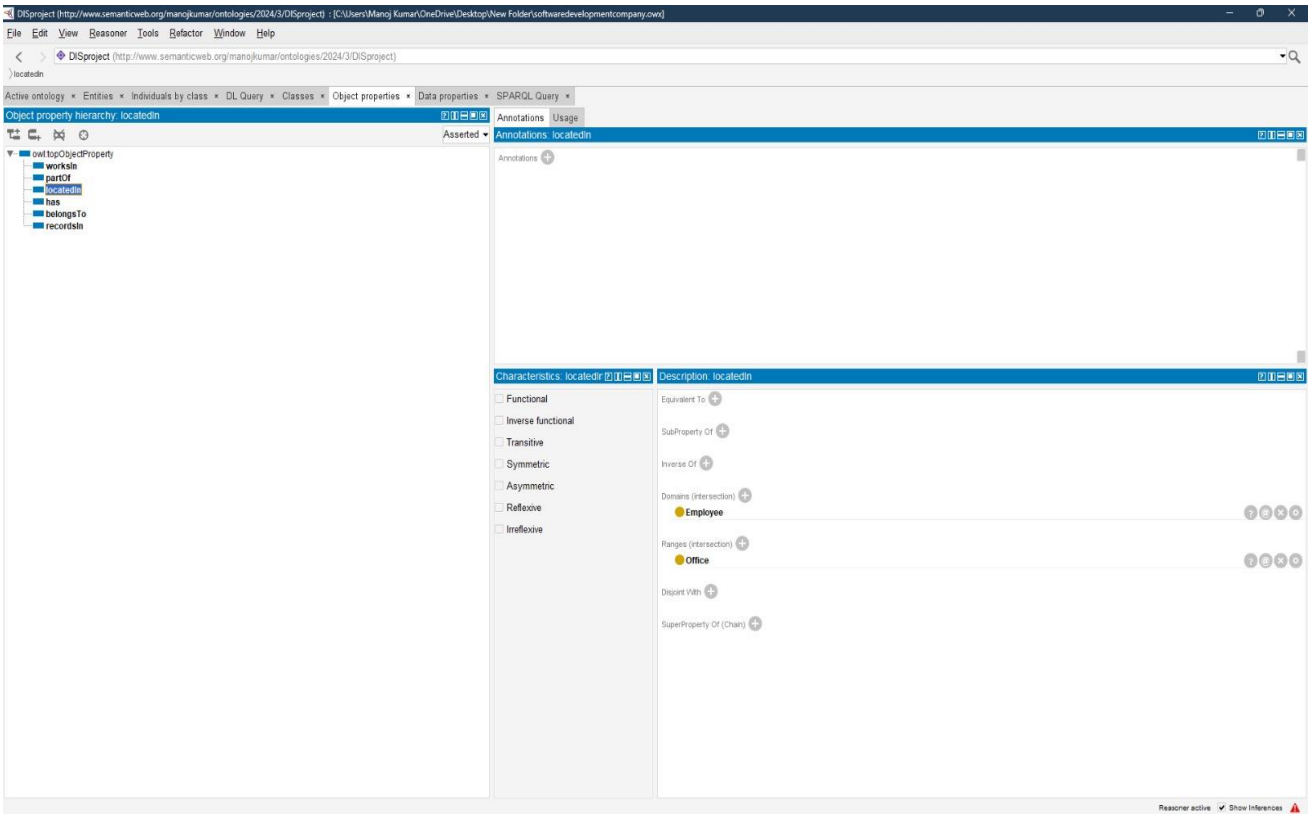
3 Ontology

3.1 Ontology diagram with explanation



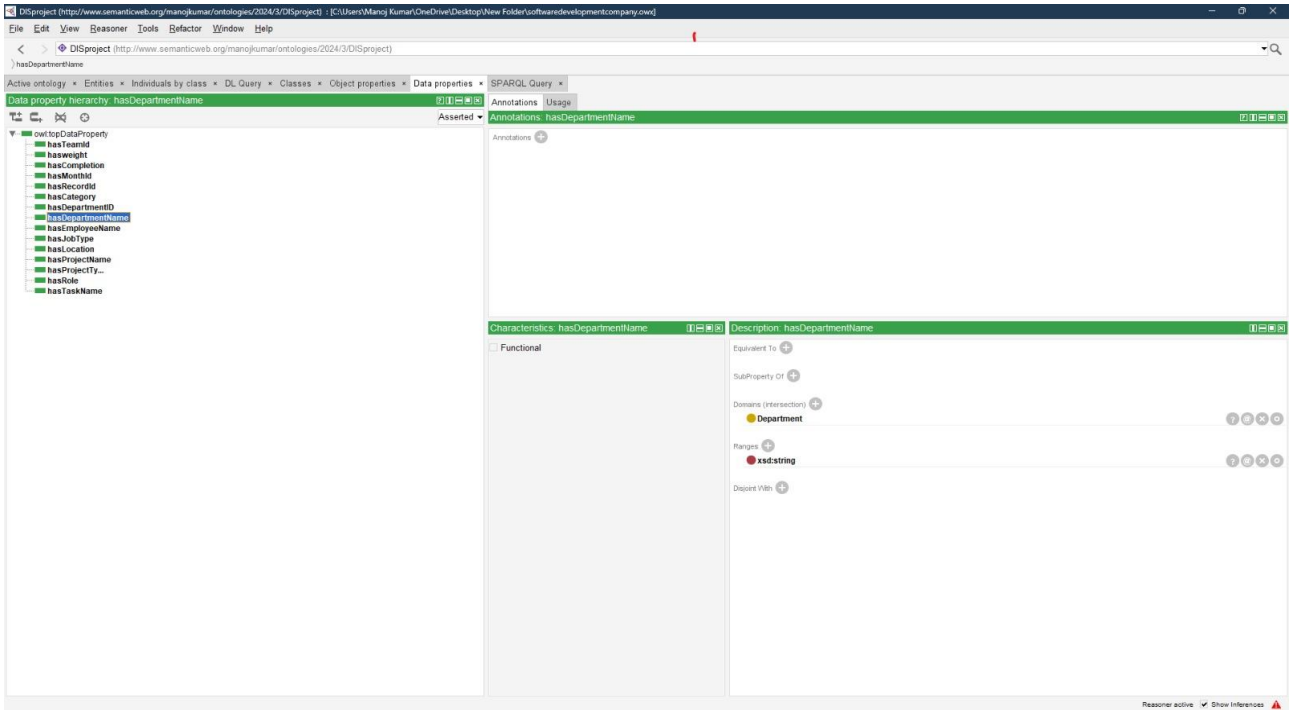
The above ontology is developed to define the connections between different elements in the domain such as department, employee, projects, tasks, office, etc. The ontology specifies, concepts, links, and limitations making it easier for the user to understand and use data.

3.2 Object properties with explanation



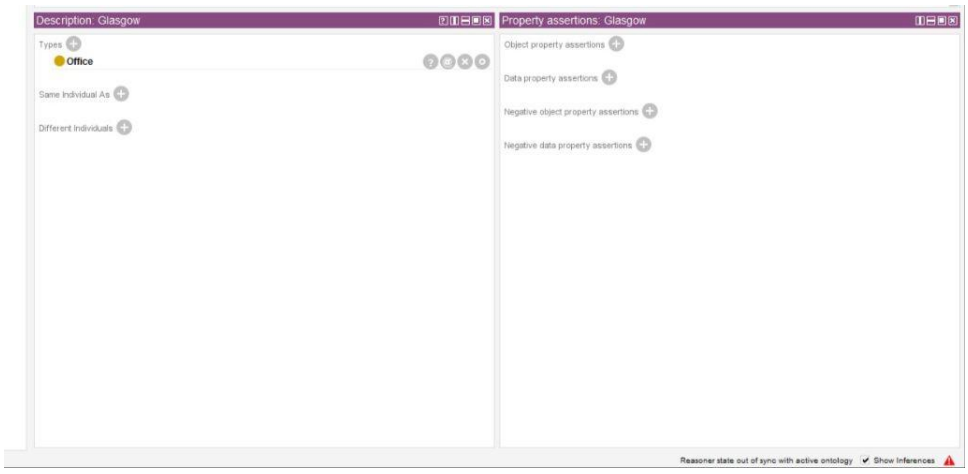
The primary focus is on the locatedIn property, which connects individuals of one class to another and specifies where an entity is located within the senario. locatedIn is specified as a feature, which means that each individual can be located in one office at a time, in line with real world limits that employee cannot be in two offices at the same time. In such a way different object properties function based on their requirements.

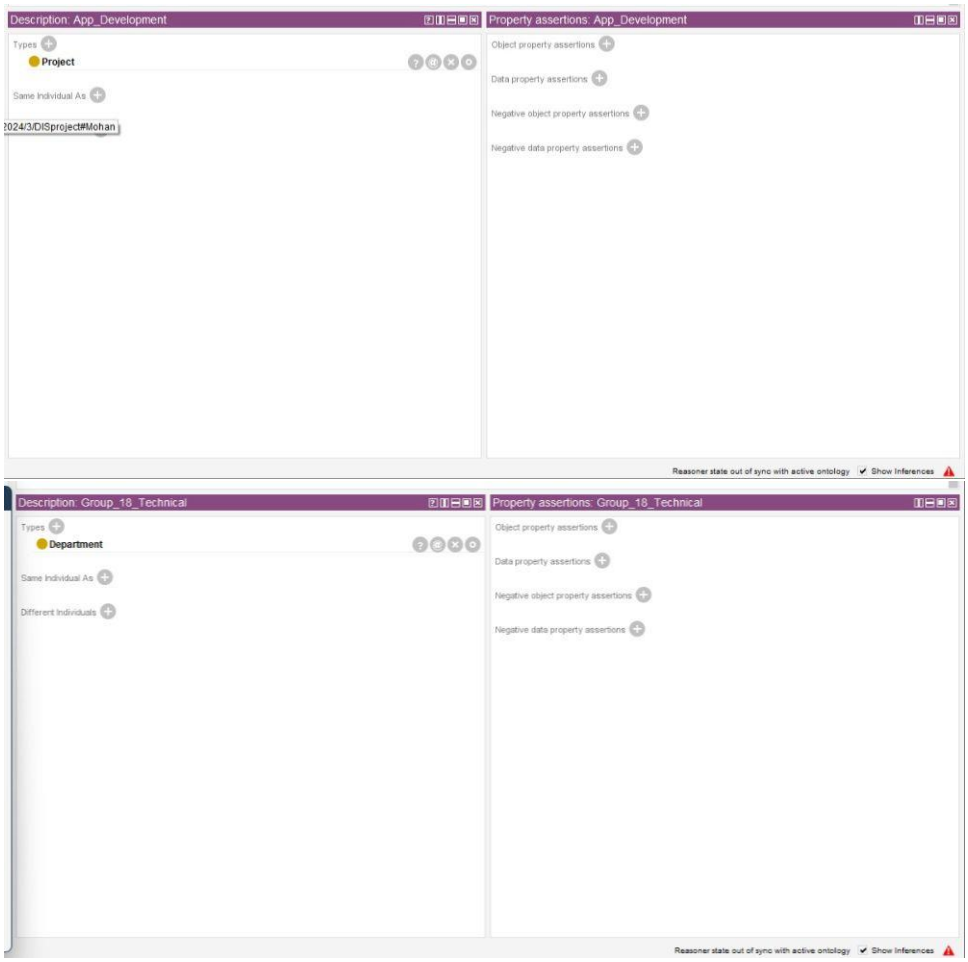
3.3 Data properties with explanation

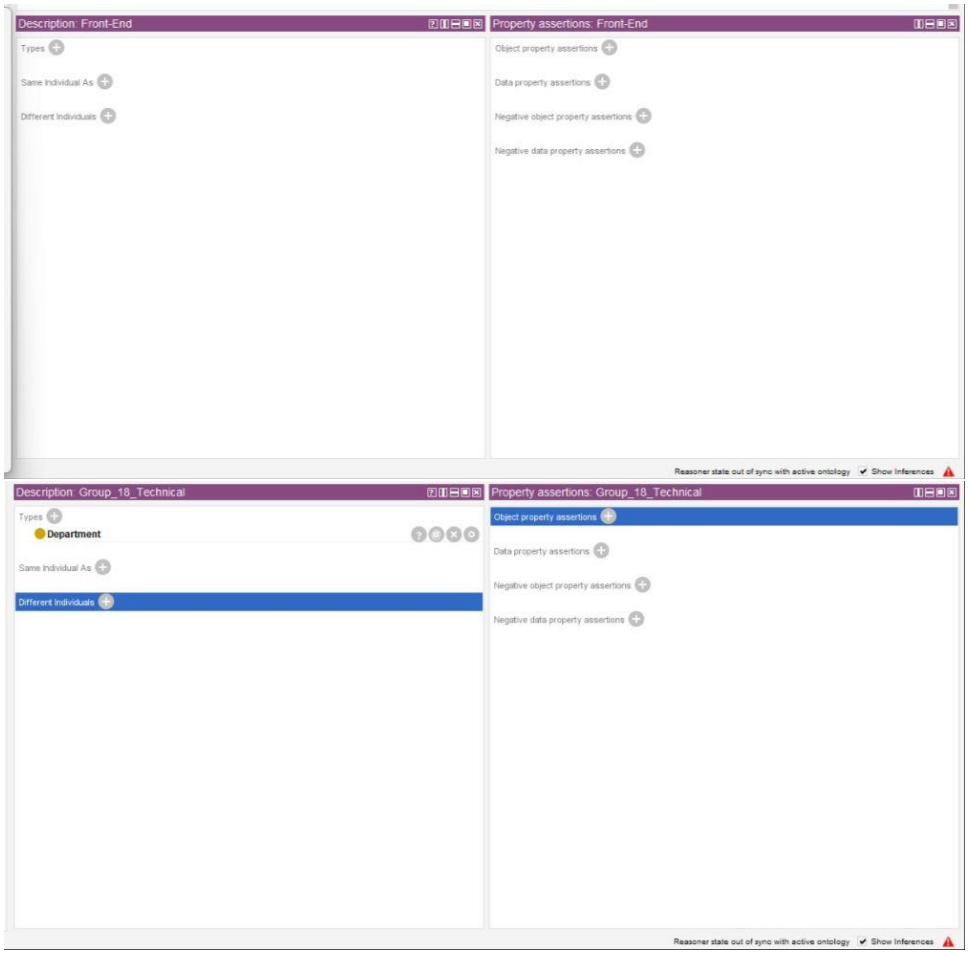


The ontology's data properties such as hasDepartmentName, hasEmployeeName, hasJobType, hasLocation provide information about departments, role, job, project, and tasks inside the company. These qualities contribute to demonstrate the distinct between storing, integrating, and exchanging data in information system using multiple data models. The hasDepartment function is used to associate each department with its corporation with its name, and also representing the organisation's structure. Other attributes such as hasEmployeeName, hasJobType, are used to associate employees with their individual names and job roles, thereby meeting the requirement to provide Employee ID's and names.

3.4 Instances




Description: App_Development
Types
Project
Same Individual As
2024/3/DISproject#Mohan
Property assertions: App_Development
Object property assertions
Data property assertions
Negative object property assertions
Negative data property assertions
Reasoner state out of sync with active ontology
Show Inferences
Description: Group_18_Technical
Types
Department
Same Individual As
Different Individuals
Property assertions: Group_18_Technical
Object property assertions
Data property assertions
Negative object property assertions
Negative data property assertions
Reasoner state out of sync with active ontology
Show Inferences



3.5 Five SPARQL queries with results

List the ID and name of all employees.

SELECT ?employeeID ? EmployeeName

WHERE {

?employee a :Employee .

?employee :hasEmployeeID ?employeeID

?employee :hasEmployeeName ?employeeName .

}

List the name of all projects together with their type.

SELECT ?projectName ?ProjectType

WHERE {

?project a:Project .

?project :hasProjectName ?projectName .

?project :hasProjectType ?projectType .

}

List the ID and name of all employees together with the project name and project category they work on.

SELECT ?employeeID ?employeeName ?projectName ?projectCategory

WHERE {

?employee a: Employee .

?employee :hasEmployeeID ?employeeID .

?employee :hasEmployeeName ?employeeName.

?project a:Project .

?project :hasProjectName ?projectName .

?project: hasProjectCategory ?projectCategory .

?employee :worksOn ?project .

}

List the ID and name of all tasks together with the name and ID of the employee who works on this task.

SELECT ?taskID ?taskName ?employeeName

WHERE {

?task a :Task .

?task :hasTaskID ?taskID .

?employee a :Employee .

?employee :hasEmployeeID ?emplyeeID .

?employee :hasEmployeeName ?employeeName .

?employee :worksOnTask ?task .

}

List the ID and name of all employees together with the office number that they are based in.

SELECT ?employeeID ?employeeName ?officeNumber

WHERE {

?employee a :Employee .

?employee :hasEmployeeID ?employeeID .

?employee :hasEmployeeName ?employeeName .

```
?office a :Office .

?office :hasOfficeNumber ?officeNumber .

?employee :isLocatedIn ?office .

}
```

We were facing continuous error with protege sparql query, so we are not able to run the sparql.

4. Discussion about extra knowledge based on the developed ontology.

By concluding this coursework, the developed ontology enables the extraction of extra knowledge not readily apparent in relational or XML data models. It can reduce implicit relationships, such as identifying potential project tasks for employees based on their roles. For example, a software engineers' involvement in multiple projects could imply a readiness for a leadership role, an insight gained via inferencing across various classes and properties. This inferencing capability allows us to predict workload distributions and identify key personnel who may be critical for project completion. Furthermore, the ontology can suggest which employees might be critical for project completion or filling out knowledge gaps between the teams. It contributes to a dynamic understanding of the organisations structure and its operational capabilities, providing a more comprehensive view of the company’s workflow and personnel dynamics.