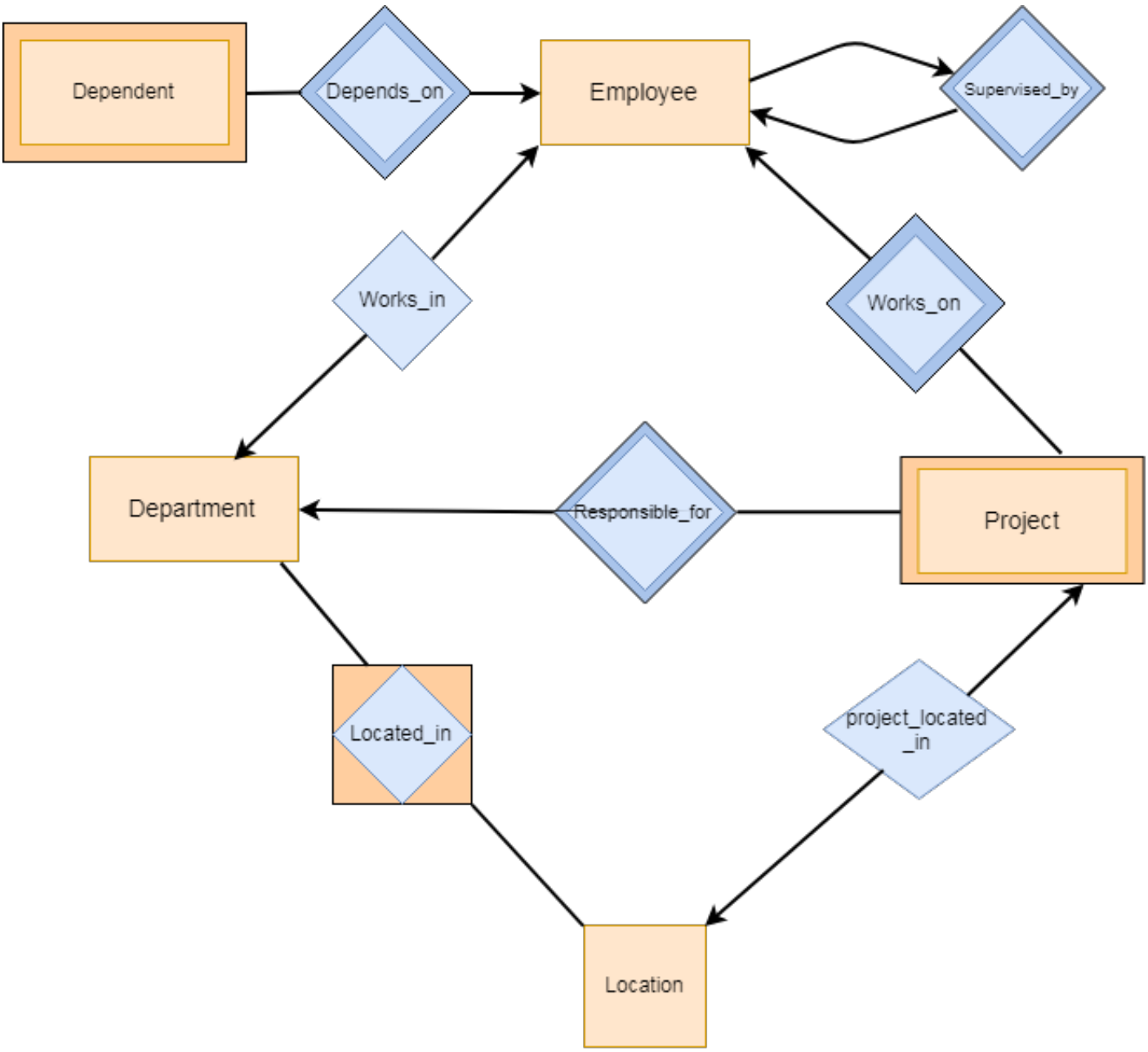
Comp353 Project Report

Kai Nicoll-Griffith[40012407], Stephen Prizio[40001739],
Giovanni Gebran[40018637], Nizar Belhassan[27519443]

Team kzc353_4

April 10, 2018

1 Entity Relationship Diagram



2 Reasonable Assumptions

2.1 general cases

An assumption is made that all identification numbers are unsigned integers. An identification key will never have a sign so the database restricts this.

2.2 department table

In the case of the 'department' table, both the manager_id and manager_start_date are given the opportunity to be null since it is not always true that a 'department' needs a manager. Small groups could potentially self manage if that is the policy of the company.

2.3 employee table

To ensure there will always be relevant 'employee' data, there are no optional or null possible parameters possible within the 'employee' table. It is assumed that a company needs to keep accurate track of everyone within it and null values would encourage poor data management practice of the company. A salary(a 5,2 decimal datatype) is given to each employee in dollars per hour to make certain queries easier to process. Due to legislation, gender attribute is defined by one ambiguous character. An 'employee' must work for a single 'department'.

2.4 project table

It is assumed that a 'project' can not be assigned to multiple 'departments'. Also a 'project' has a varchar phase attribute which keeps track of the progress of each individual project within the COMPANY database.

2.5 dependent table

The 'dependent' table holds vital information that has potential legal importance so none of these fields may be null. A dependent is linked to an 'employee' by a foreign key holding employee_id and has the multiplicity of one to many. An 'employee' may have many 'dependents'.

2.6 location table

In order to specify where a 'project' or 'department' is situated, a 'location' table keeps track of all of the possible locations where departments and projects operate. An entity table will therefore use a relation table holding an unsigned location_id to specify where the department or project is located in both address and an optional name. The name is assumed to be used for employee convenience to identify a location while a mandatory address is used for more direct positioning and referencing(as would be used by a post office). The name is a varchar, while the address is medium text since it is assumed that the address could be as specific as country down to room number and limitations on varchar size could be problematic.

2.7 supervised_by table

The 'supervised_by table' defines a role of being a subordinate to someone and helps to give information about the status of an employee in the business hierarchy. Supervision does not imply that an employee is a manager and it could be that an employee both supervises and manages a 'department'.

It is assumed that this relation is solely used to show the hierarchy of employees within the company. To recognize the ‘employee‘ who is supervised, each employee is given a single supervisor_id with a 1:1 multiplicity. Our assumption is that an employee should only be supervised by one person or none at all therefore employee _id is a primary key enforcing uniqueness while supervisor_id is a default null value, where null implies an ‘employee‘ is unsupervised.

2.8 depends_on relation

The weak relation ‘depends_on‘ creates the assumption an ‘employee‘ can have many ‘dependants‘ in a 1:many relationship.

2.9 works_on relation

The weak relation ‘works_on‘ creates the assumption that an ‘employee‘ can work on many ‘projects‘ in a 1:many relationship

2.10 works_in relation

The strong relation ‘works_in‘ creates the assumption that an ‘employee‘ can only work in one ‘department‘ in a 1:1 relationship.

2.11 responsible_for relation

The weak relation ‘responsible_for‘ creates the assumption that a ‘department‘ can be responsible for many ‘projects‘ in a 1:many relationship

2.12 project_located_in relation

The strong relation ‘project_located_in‘ creates the assumption that a project has to be tied to one location in a 1:1 relationship.

2.13 department_located_in relation

The associative entity ‘project_located_in‘ creates the assumption that a ‘department‘ can be positioned in many ‘locations‘ while at the same time a ‘location‘ can be assigned to many ‘departments‘ in a many:many relationship.

3 ER to Relation conversion

4 Normalization steps and assumptions

5 Implemented Functionalities

5.1 Database design

In the COMPANY database There are three primary categories of entity from which more complex entities are defined. These are:

1. departments,
2. employees,
3. projects,

Each of these tables specifies information that defines the three main entities in the database. These three main entity sets are also enhanced by the entity sets of:

1. dependent
2. location

And also the role relation:

1. supervised_by

Which specifies an employees role against other employees as a supervisor.
While the entity-relation diagram specifies multiple that multiple possible relations can be made, in order to reduce the complexity of the design(and therefore the queries) only the following relations are used

1. works_on
2. responsible_for
3. located_in

These three relations were deemed most important and the other relations seen on the E/R diagram have been omitted.

5.2 Language and tools

The application makes use of the PHP 5.5.9 language due to it's reliable and simple functions for connecting with a MySQL database. In order to more easily input queries on the database and build a modern looking front end system, Lavarel has been used to make development easier which adds additional functionality to and shortcuts to front-end design.

5.3 Query Functionalities

21 Queries allow the system to select, update and add to the company database. These are in the form of .php filenames found in the source code folder.

5.3.1 delete_department.php

DELETE FROM department WHERE department.id ='$id'

5.3.2 get_all_projects_for_department.php

SELECT responsible_for.project_id AS Project_ID, (SELECT project.name FROM project WHERE responsible_for.project_id=project.id) AS Project_Name FROM responsible_for WHERE department_id = $department_id

5.3.3 get_employee_dependents.php

SELECT dependent.id AS Dependent_ID, dependent.first_name, dependent.last_name FROM dependent, employee WHERE dependent.employee_id='$employee_id' AND employee.id=dependent.employee_id

5.3.4 get_employee_involved_in_least_num_of_projects.php

SELECT works_on.employee_id, employee.first_name, employee.last_name FROM works_on JOIN employee on employee.id=works_on.employee_id Group by employee_id Order by COUNT(project_id) ASC LIMIT 1

5.3.5 get_employee_involved_in_most_num_of_projects.php

SELECT works_on.employee_id, employee.first_name, employee.last_name FROM works_on JOIN employee on employee.id=works_on.employee_id Group by employee_id Order by COUNT(project_id) Desc LIMIT 1

5.3.6 get_employee_supervisor.php

SELECT role.supervisor_id AS Supervisor_ID, (SELECT first_name FROM employee WHERE role.supervisor_id=
) AS First_Name, (SELECT last_name FROM employee WHERE role.supervisor_id=employee.id) AS
Last_Name FROM role, employee WHERE role.employee_id='$employee_id' AND employee.id=role.employee_id

5.3.7 get_employees_who_work_on_a_project.php

SELECT employee.id AS Manager_ID,employee.first_name, employee.last_name FROM department
JOIN responsible_for ON department.id=responsible_for.department_id JOIN employee ON employee.id=departme
JOIN works_on ON works_on.project_id=responsible_for.project_id AND works_on.employee_id=department.manag

5.3.8 get_hours_worked_employee.php

SELECT hours_worked FROM works_on, employee WHERE employee.id = '$employee_id' AND project_id = '$project_id' AND employee.id=works_on.employee_id

5.3.9 get_how_much_employee_gets.php

SELECT employee.salary FROM employee WHERE employee.id=$employee_id

5.3.10 get_project_location.php

SELECT project.location_id AS Location_ID, (SELECT location.name FROM location WHERE project.location_i
AS Location_name, (SELECT location.address FROM location WHERE project.location_id=location.id)
AS Address FROM project WHERE project.id =$project_id

5.3.11 get_total_hours_worked_for_project.php

SELECT SUM(works_on.hours_worked) AS total_hours FROM works_on WHERE works_on.project_id
=$project_id

5.3.12 get_total_pay_for_each_project.php

SELECT works_on.hours_worked, works_on.employee_id, employee.salary From works_on, employee Where works_on.project_id=2 AND employee.id=works_on.employee_id

5.3.13 insert_department.php

INSERT INTO department (id, name, manager_id, manager_start_date) VALUES ('$id','$name','$manager_id','$m

5.3.14 insert_dependent.php

INSERT INTO dependent (id, first_name, last_name, sin, date_of_birth, gender, employee_id) VALUES ('$id', '$first_name', '$last_name', '$sin', '$date_of_birth','$gender', '$employee_id')

5.3.15 insert_employee.php

INSERT INTO employee (id, first_name, last_name, sin, date_of_birth, address, phone, salary, gender, department_id) VALUES ('$id', '$first_name', '$last_name', '$sin', '$date_of_birth', '$address', '$phone', '$salary', '$gender', '$department_id')

5.3.16 insert_located_in.php

INSERT INTO located_in (location_id, department_id) VALUES ('$location_id', '$department_id')

5.3.17 insert_location.php

INSERT INTO location (id, name, address) VALUES ('$id', '$name','$address')

5.3.18 insert_project.php

INSERT INTO project (id, name, location_id, phase) VALUES ('$id', '$name','$location_id', '$phase')

5.3.19 insert_responsible_for.php

INSERT INTO responsible_for (project_id, department_id) VALUES ('$project_id', '$department_id')

5.3.20 insert_role.php

INSERT INTO role (employee_id, supervisor_id) VALUES ('$employee_id', '$supervisor_id')

5.3.21 insert_works_on.php

INSERT INTO works_on (project_id, employee_id, hours_worked) VALUES ('$project_id','$employee_id', '$hours_worked')

6 contributions

6.1 Giovanni Gebran

- Database Design

6.2 Nizar Belhassan

- Database Design
- Majority of Queries

6.3 Kai Nicoll-Griffith

- Database Design
- Database Attribute Refinements
- Report setup and latex entry
- Report: ER Diagram
- Report: Constraints and assumptions
- Report: Functionalities

6.4 Stephen Prizio

- Database Design
- Front end Lazarel design
- SQL sample data and database
- Minority of Queries