STAT 311
Modern Database Systems

Term Project:
Human Resources
Database

Berfin Akdemir - 2614410
Buse Erdem - 2561231
Nida Karataş - 2561330
Zeynep Sevim Kanar - 2561306

METU
Fall, 2024

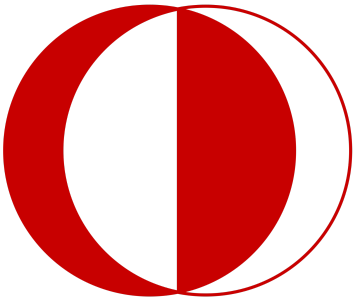


Table of Contents

Introduction

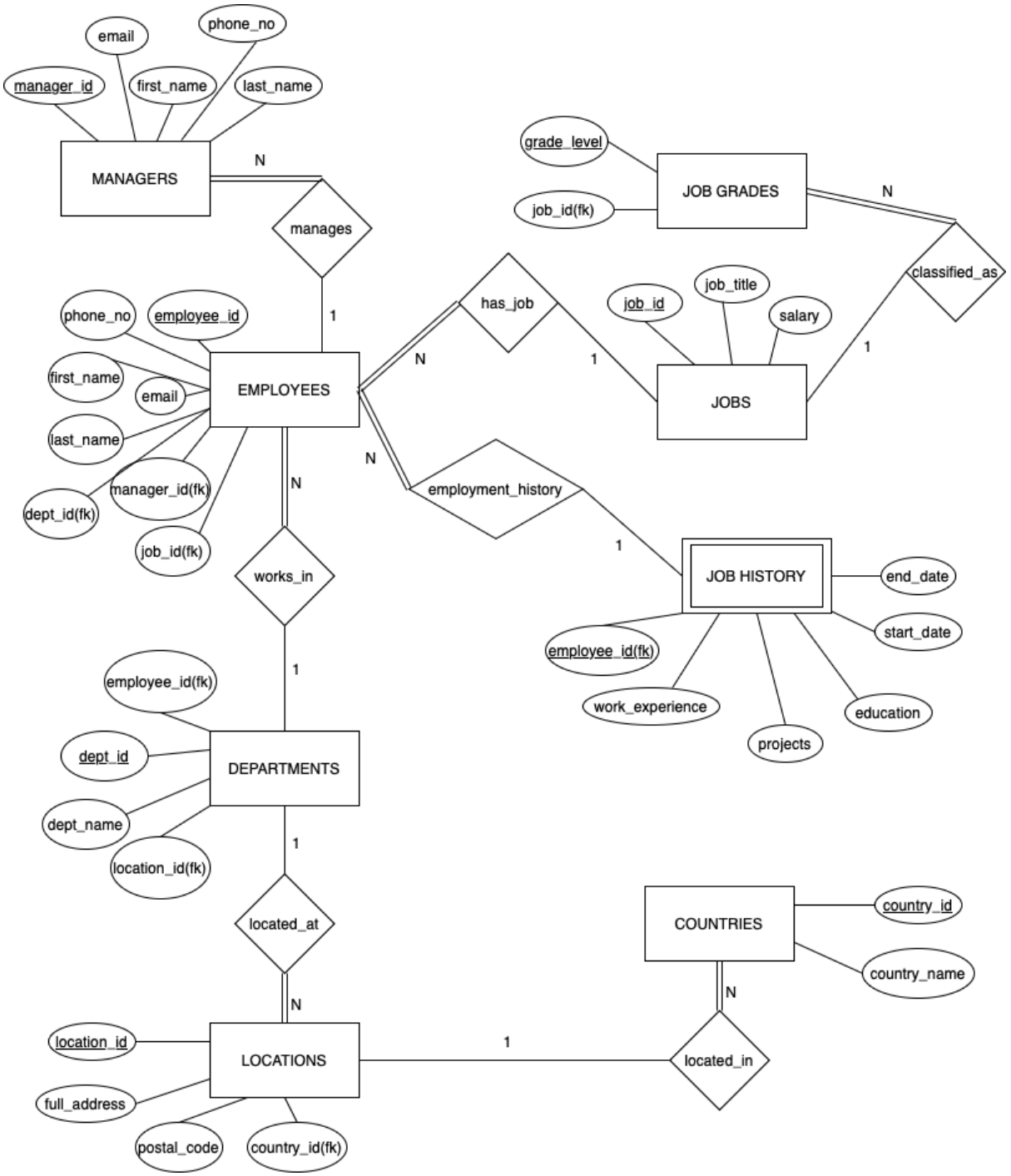
In today's dynamic and competitive business environment, Human Resources (HR) management is a cornerstone of organizational success, playing a critical role in fostering operational efficiency, enhancing employee satisfaction, and driving strategic decision-making. As organizations grow and workforce dynamics evolve, the complexity of managing employee data and HR processes has increased significantly. This shift underscores the indispensability of sophisticated HR Database Management Systems (HRDMS), which enable organizations to centralize HR operations, ensure data accuracy, and derive actionable insights from comprehensive employee information.

Our project seeks to address these challenges by designing and implementing a cutting-edge HR Database Management System using MySQL. MySQL was selected for its robust capabilities in relational database management, scalability, and ability to handle complex queries and large datasets. The system is meticulously designed to store, organize, and manage diverse employee data, ranging from personal information and job histories to educational backgrounds, locations and job grades. By employing normalization techniques and adhering to best practices in database design, the system minimizes data redundancy, ensures referential integrity, and optimizes storage efficiency.

Beyond its technical foundation, the system is tailored to simplify routine HR operations. It streamlines tasks such as maintaining detailed employee profiles, managing promotions, and observing previously assigned projects. Moreover, its user-friendly structure ensures ease of access while robust security measures safeguard sensitive employee data.

This project not only exemplifies the practical application of database design principles but also demonstrates the potential of technology to transform HR management. By enabling data-driven decision-making and improving operational efficiency, the developed HRDMS aligns with contemporary organizational needs, paving the way for more efficient and strategic HR practices.

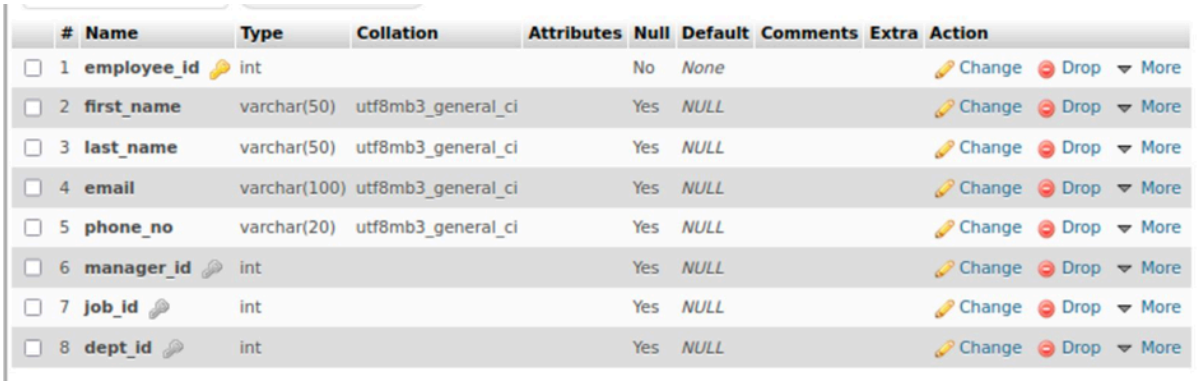
ER DIAGRAM


email
phone_no
manager_id
first_name
last_name
MANAGERS
N
manages
1
grade_level
job_id(fk)
JOB GRADES
N
classified_as
1
has_job
N
1
job_id
job_title
salary
JOBS
phone_no
employee_id
first_name
email
last_name
manager_id(fk)
dept_id(fk)
job_id(fk)
EMPLOYEES
N
N
employment_history
1
JOB HISTORY
end_date
start_date
employee_id(fk)
work_experience
projects
education
works_in
1
employee_id(fk)
dept_id
dept_name
location_id(fk)
DEPARTMENTS
1
located_at
N
COUNTRIES
country_id
country_name
N
location_id
full_address
postal_code
country_id(fk)
LOCATIONS
1
located_in

Schema Design

The schema and their table properties were developed using MySQL (and phpMyAdmin) in our virtualized Ubuntu environment. Below are the table-by-table design properties:

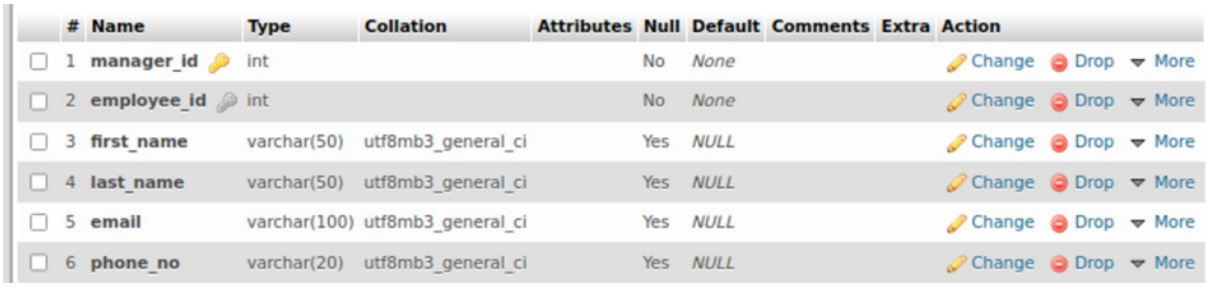
Employee Schema



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	employee_id	int			No	None			Change Drop More
2	first_name	varchar(50)	utf8mb3_general_ci		Yes	NULL			Change Drop More
3	last_name	varchar(50)	utf8mb3_general_ci		Yes	NULL			Change Drop More
4	email	varchar(100)	utf8mb3_general_ci		Yes	NULL			Change Drop More
5	phone_no	varchar(20)	utf8mb3_general_ci		Yes	NULL			Change Drop More
6	manager_id	int			Yes	NULL			Change Drop More
7	job_id	int			Yes	NULL			Change Drop More
8	dept_id	int			Yes	NULL			Change Drop More

Within this schema, each employee's unique identification number (ID) is stored in employee_id. It is designated as a primary key and to ensure that each employee has an ID number, employee_id cannot be NULL. first_name is used to store the employee's name. Because the varchar(50) data type is being used, text data up to 50 characters long can be entered. last_name is used to store the employee's last name. email is used to store the employee's email address. The employee's phone number is stored in phone_no. manager_id stores the identification number (ID) of the manager to which the employee reports. The int data type is used, it accepts only integer values. It can be NULL, meaning not every employee has to have a manager. manager_id is used as a foreign key to ensure the integrity and consistency of data in the database. job_id stores the employee's job position. job_id is associated with jobs_id in the Jobs schema as a foreign key. dept_id stores the identification number of the department to which the employee is affiliated.

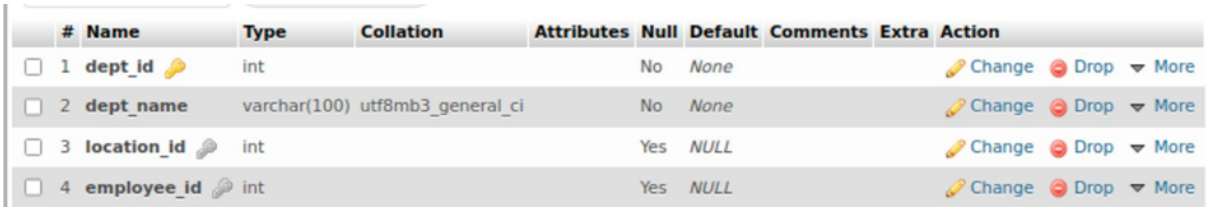
Managers Schema



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	manager_id	int			No	None			Change Drop More
2	employee_id	int			No	None			Change Drop More
3	first_name	varchar(50)	utf8mb3_general_ci		Yes	NULL			Change Drop More
4	last_name	varchar(50)	utf8mb3_general_ci		Yes	NULL			Change Drop More
5	email	varchar(100)	utf8mb3_general_ci		Yes	NULL			Change Drop More
6	phone_no	varchar(20)	utf8mb3_general_ci		Yes	NULL			Change Drop More

In this schema, manager_id contains the unique identification number (ID) for each manager. It is designated as a primary key. manager_id cannot be NULL in order to guarantee that every manager has an ID number. employee_id is used as a foreign key to ensure the integrity and consistency of data in the database. The employee_id value references an employee in the employees table. first_name is used to store the manager's name. last_name is used to store the manager's last name. email is used to store the employee's email address. The employee's phone number is stored in phone_no.

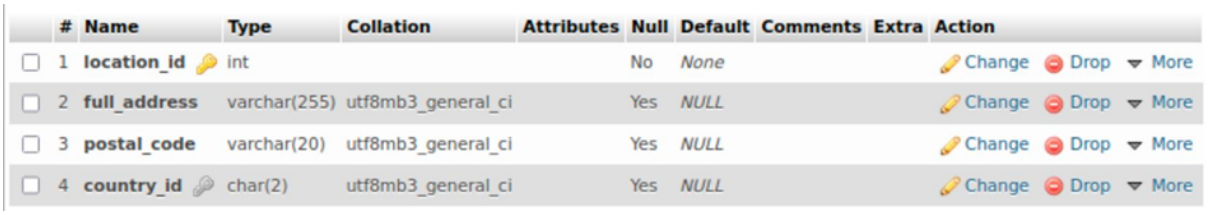
Departments Schema



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	dept_id	int			No	None			Change Drop More
2	dept_name	varchar(100)	utf8mb3_general_ci		No	None			Change Drop More
3	location_id	int			Yes	NULL			Change Drop More
4	employee_id	int			Yes	NULL			Change Drop More

dept_id is the Department ID, a unique identification number for each department. It is designated as a primary key, meaning that each department must have an ID and this ID number must be unique. dept_name stores the department name e.g. HR, Engineering, Sales. location_id is the identification number of the department's location. employee_id is used as a foreign key to indicate the employees of the department.

Location Schema



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	location_id	int			No	None			Change Drop More
2	full_address	varchar(255)	utf8mb3_general_ci		Yes	NULL			Change Drop More
3	postal_code	varchar(20)	utf8mb3_general_ci		Yes	NULL			Change Drop More
4	country_id	char(2)	utf8mb3_general_ci		Yes	NULL			Change Drop More

In this schema, location_id is a unique identification number of the locations. It is designated as a primary key, meaning that each location must have an ID and this ID number must be unique. full_address contains the full and detailed address information of the locations. This is used to specify in detail where the company's offices are located. postal_code stores the postal code of the region where the location is located. country_id indicates which country the location is in, it refers to the country identification number in the Countries table.

Countries Schema

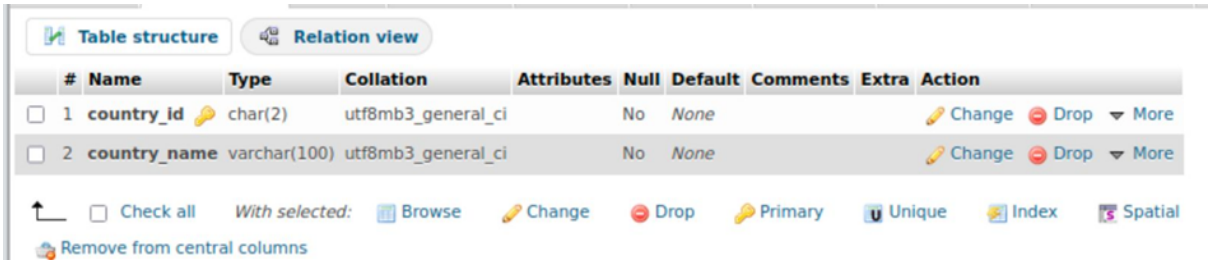


Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	country_id	char(2)	utf8mb3_general_ci		No	None			Change Drop More
2	country_name	varchar(100)	utf8mb3_general_ci		No	None			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Spatial

Remove from central columns

The country identifier for this nation is referred to as country_id. It serves as the Primary key and is unique to every nation, meaning it cannot contain NULL values. Its data type is Char(2). For example, TR represents Turkey, while FR represents France. country_name describes which country the country code in the country_id column corresponds to. country_name contains the country name as text for easier understanding by users.

Jobs Schema

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	job_id	int			No	None			Change Drop More
2	job_title	varchar(100)	utf8mb3_general_ci		No	None			Change Drop More
3	salary	decimal(10,2)			Yes	NULL			Change Drop More

job_id represents the job title (task ID) associated with the employee. It serves as the primary key. This field contains a unique identification number (ID) for every job position, ensuring that each one has a distinct ID. It cannot be NULL, which means that every job position must have an associated ID number. job_title stores job titles such as Software Engineer, Hr Specialist, Sales Manager. It defines the position of an employee at work. salary stores the salaries of employees.

Job Grades

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	grade_level	varchar(10)	utf8mb3_general_ci		No	None			Change Drop More
2	job_id	int			Yes	NULL			Change Drop More

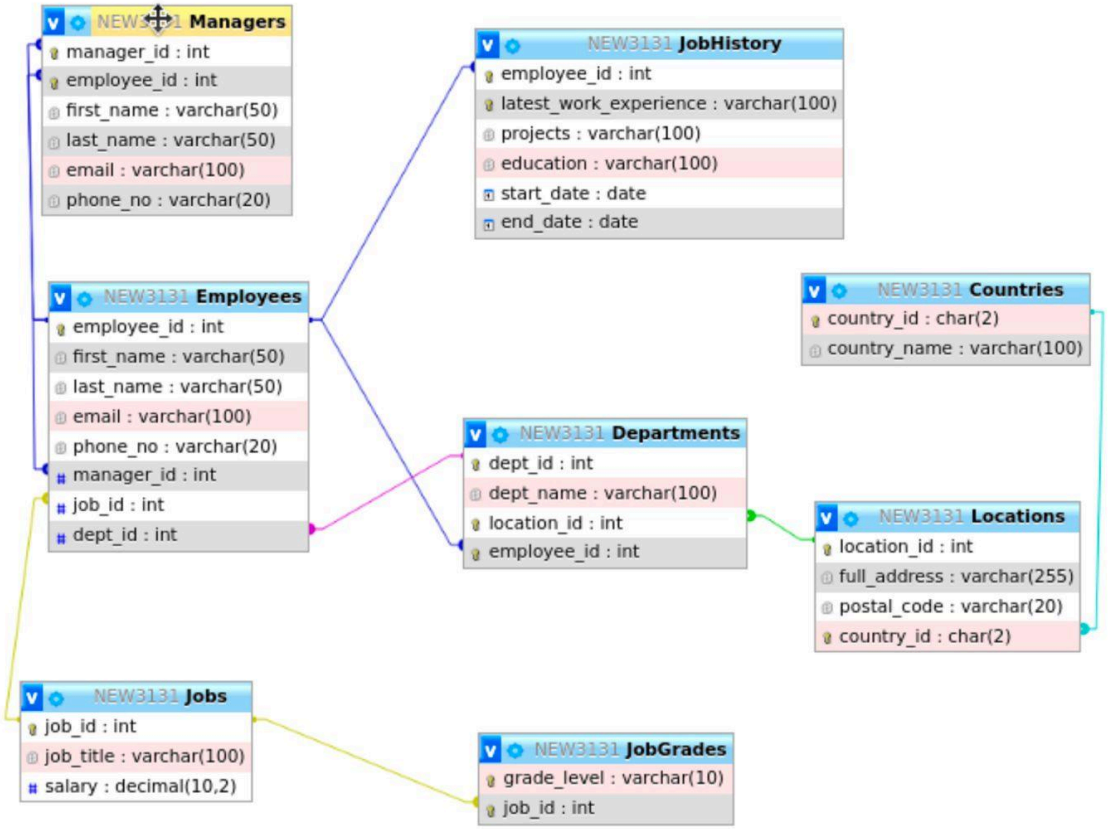
The grade_level refers to the employee's grade level (accepting values such as A, B, C). It serves as the primary key, necessitating that each grade_level value is distinct. Additionally, it cannot be NULL. The job_id represents a unique identifier for a certain position and functions as a foreign key.

Job History

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	employee_id	int			No	None			Change Drop More
2	latest_work_experience	varchar(100)	utf8mb3_general_ci		No	None			Change Drop More
3	projects	varchar(100)	utf8mb3_general_ci		Yes	NULL			Change Drop More
4	education	varchar(100)	utf8mb3_general_ci		Yes	NULL			Change Drop More
5	start_date	date			Yes	NULL			Change Drop More
6	end_date	date			Yes	NULL			Change Drop More

This schema is designed to oversee the work experience and education of employees. The employee_id is associated with the owner entity (Employee) that uniquely records each employee's information. A record cannot be added to the Job History table unless the employee_id exists in the Employee table, thereby maintaining data integrity. The latest_work_experience captures the employee's most recent work experience. Together with employee_id, it creates a Composite Primary Key and cannot have a NULL value. The projects field documents the projects in which the employee took part. The education field contains the employee's educational background. The start_date indicates when the employee began the job, while the end_date shows when they left the position. If the employee remains in that job, this field will be NULL.

We can summarize all the relations between the schemas as follows;



- The manager_id value (which identifies who is the manager of an employee) as a foreign key in the Employees table should reference the current manager_id value in the Managers table. Since a manager can be responsible for more than one employee, there is a 1-N relationship.
- The dept_id value (indicates which department the employee belongs to) as a foreign key in the Employees table should reference the current dept_id value in the Departments table. Since there can be more than one employee in a department, this relationship is a 1-N relationship.
- The location_id (indicates in which location a department is located) as a foreign key in the Departments table should reference the current location_id value in the Locations table. This relationship is 1-N as there may be more than one department of the company in a location.
- The country_id (indicates which country each location is in) as a foreign key in the Locations table should reference the current country_id value in Countries table. Since there can be more than one location in a country, this relationship is also a 1-N relationship.
- The job_id (shows what job the employee does) as a foreign key in the Employees table should reference the current job_id value in Jobs table.Since there can be more than one employee in a job position, this relationship is also a 1-N relationship.
- The employee_id in the Job History table is connected to the employee_id in the Employee table through a Foreign Key (FK). In the Job History table, the combination of employee_id and latest_work_experience makes up a Composite Key. A 1-N relationship exists since an employee may have multiple work history records.

- The job_id in the Job Grades table is associated with the job_id in the Jobs table through a Foreign Key (FK). In the Job Grades table, job_id and grade_level are combined to create a Composite Primary Key. There exists a 1-N relationship since a single job position can have multiple salary levels.

Populated Tables

We use test records to populate the table, which are also updated during development. The finalized versions of these tables, along with their populated records and tuples, are provided in this section.

Employees Table

+ Options

	employee_id	first_name	last_name	email	phone_no	manager_id	job_id	dept_id
Edit Copy Delete	1	Alice	Smith	alice.newemail@example.com	123-456-7890	*NULL*	1	2
Edit Copy Delete	2	Bob	Brown	bob.brown@example.com	123-456-7891	1	2	1
Edit Copy Delete	3	Carol	Davis	carol.davis@example.com	123-456-7892	1	3	3
Edit Copy Delete	4	David	Evans	david.evans@example.com	123-456-7893	2	1	2
Edit Copy Delete	5	Eve	Foster	eve.foster@example.com	123-456-7894	3	3	3
Edit Copy Delete	6	Frank	Green	frank.green@example.com	123-456-7895	*NULL*	1	1
Edit Copy Delete	7	Grace	Hall	grace.hall@example.com	123-456-7896	2	2	2
Edit Copy Delete	8	Hank	Ivy	hank.ivy@example.com	123-456-7897	3	3	3
Edit Copy Delete	9	Ivy	Jones	ivy.jones@example.com	123-456-7898	1	1	1
Edit Copy Delete	10	Jack	King	jack.king@example.com	123-456-7899	*NULL*	2	2
Edit Copy Delete	11	Kara	Lee	kara.lee@example.com	123-456-7900	2	3	3
Edit Copy Delete	12	Liam	Morris	liam.morris@example.com	123-456-7901	3	1	1
Edit Copy Delete	13	Mia	Nelson	mia.nelson@example.com	123-456-7902	1	2	2
Edit Copy Delete	14	Nina	Owens	nina.owens@example.com	123-456-7903	2	3	3
Edit Copy Delete	15	Oscar	Perez	oscar.perez@example.com	123-456-7904	3	1	1
Edit Copy Delete	16	Paula	Quinn	paula.quinn@example.com	123-456-7905	*NULL*	2	2
Edit Copy Delete	17	Quinn	Reed	quinn.reed@example.com	123-456-7906	1	3	3
Edit Copy Delete	18	Rachel	Stone	rachel.stone@example.com	123-456-7907	2	1	1
Edit Copy Delete	19	Steve	Taylor	steve.taylor@example.com	123-456-7908	3	2	2
Edit Copy Delete	20	Tina	Underwood	tina.underwood@example.com	123-456-7909	1	3	3
Edit Copy Delete	21	Uma	Vance	uma.vance@example.com	123-456-7910	*NULL*	1	1
Edit Copy Delete	22	Victor	Wells	victor.wells@example.com	123-456-7911	2	2	2
Edit Copy Delete	23	Wendy	Xavier	wendy.xavier@example.com	123-456-7912	3	3	3
Edit Copy Delete	24	Xander	Young	xander.young@example.com	123-456-7913	1	1	1
Edit Copy Delete	25	Yara	Zimmer	yara.zimmer@example.com	123-456-7914	2	2	2

Departments Table

+ Options

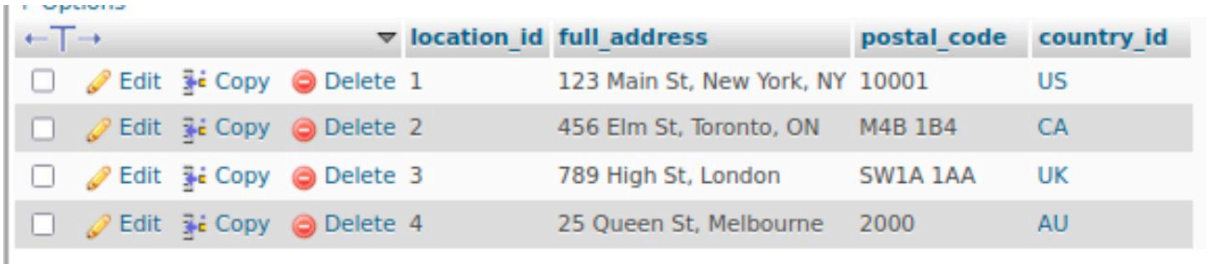
	dept_id	dept_name	location_id	employee_id
Edit Copy Delete	1	HR	2	*NULL*
Edit Copy Delete	2	Engineering	2	*NULL*
Edit Copy Delete	3	Sales	3	*NULL*
Edit Copy Delete	4	Marketing	4	*NULL*

Managers Table

+ Options

	manager_id	employee_id	first_name	last_name	email	phone_no
Edit Copy Delete	1	31	Buse	Erdem	buse.erdem@example.com	503-456-7890
Edit Copy Delete	2	32	Berfin	Akdemir	berfin.akdemir@example.com	513-456-7891
Edit Copy Delete	3	33	Nida	Karatas	nida.karatas@example.com	523-456-7892
Edit Copy Delete	4	34	Zeynep	Kanar	zeynep.kanar@example.com	533-456-7893

Locations Table



	location_id	full_address	postal_code	country_id
Edit Copy Delete	1	123 Main St, New York, NY	10001	US
Edit Copy Delete	2	456 Elm St, Toronto, ON	M4B 1B4	CA
Edit Copy Delete	3	789 High St, London	SW1A 1AA	UK
Edit Copy Delete	4	25 Queen St, Melbourne	2000	AU

Countries Table

+ Options

	country_id	country_name
Edit Copy Delete	AU	Australia
Edit Copy Delete	CA	Canada
Edit Copy Delete	UK	United Kingdom
Edit Copy Delete	US	United States

Jobs Table

	job_id	job_title	salary
Edit Copy Delete	1	Software Engineer	95000.00
Edit Copy Delete	2	HR Specialist	60000.00
Edit Copy Delete	3	Sales Manager	75000.00
Edit Copy Delete	4	Marketing Analyst	65000.00

Job Grades Table

	grade_level	job_id
Edit Copy Delete	A	1
Edit Copy Delete	B	2
Edit Copy Delete	C	3

Job History Table

	employee_id	latest_work_experience	projects	education	start_date	end_date
Edit Copy Delete	1	5 years Software Engineering	Project Delta	BSc in Computer Science	2015-06-01	2020-06-01
Edit Copy Delete	2	3 years HR Experience	Project Beta	MBA in HR	2017-06-01	2020-06-01
Edit Copy Delete	3	7 years Sales	Project Gamma	BBA	2013-06-01	2020-06-01

Insert Operation

The **INSERT** operation is used to add new data into a table in a database. It allows you to insert a new row with specific values for each column. Let's see its usage in our data.

-- Insert new employee

INSERT INTO Employees (employee_id, first_name, last_name, email, phone_no, manager_id, job_id, dept_id)

VALUES (36, 'Olivia', 'Parker', 'olivia.parker@example.com', '987-654-3210', 1, 2, 1);

After:

Edit Copy Delete	31	Buse	Erdem	buse.erdem@example.com	513-456-7890	1	1	2
Edit Copy Delete	32	Berfin	Akdemir	berfin.akdemir@example.com	523-456-7891	2	2	1
Edit Copy Delete	33	Nida	Karatas	nida.karatas@example.com	523-456-7892	3	4	4
Edit Copy Delete	34	Zeynep	Kanar	zeynep.kanar@example.com	543-456-7893	4	1	2
Edit Copy Delete	35	Ian	Walker	ian.walker@example.com	123-456-7895	*NULL*	4	4
Edit Copy Delete	36	Olivia	Parker	olivia.parker@example.com	987-654-3210	1	2	1

INSERT INTO Jobs (job_id, job_title, salary)

VALUES (5, 'Data Scientist', 95000.00);
After:

Cascade on Delete and Update and Referential Integrity

Cascade options in databases are used to maintain **referential integrity**, which ensures that relationships between tables remain consistent when changes are made. These options help prevent orphaned records or broken links between related tables.

```
CREATE TABLE Countries (
    country_id CHAR(2) PRIMARY KEY,
    country_name VARCHAR(100) NOT NULL
);
CREATE TABLE Locations (
    location_id INT PRIMARY KEY,
    full_address VARCHAR(255),
    postal_code VARCHAR(20),
    country_id CHAR(2),
    FOREIGN KEY (country_id) REFERENCES Countries(country_id)
);
CREATE TABLE Departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100) NOT NULL,
    location_id INT,
    employee_id INT,
    FOREIGN KEY (location_id) REFERENCES Locations(location_id)
);
CREATE TABLE Jobs (
    job_id INT PRIMARY KEY,
    job_title VARCHAR(100) NOT NULL,
    salary DECIMAL(10, 2)
);
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
```

```
    email VARCHAR(100),
    phone_no VARCHAR(20),
    manager_id INT,
    job_id INT,
    dept_id INT,
    FOREIGN KEY (manager_id) REFERENCES Employees(employee_id),
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id)
);
CREATE TABLE JobHistory (
    employee_id INT,
    latest_work_experience VARCHAR(100),
    projects VARCHAR(100),
    education VARCHAR(100),
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (employee_id, latest_work_experience),
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
CREATE TABLE Managers (
    manager_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_no VARCHAR(20),
    FOREIGN KEY (manager_id) REFERENCES Employees(employee_id)
);
CREATE TABLE JobGrades (
    grade_level VARCHAR(10) PRIMARY KEY,
    job_id INT,
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id)
);
ALTER TABLE Departments

ADD CONSTRAINT fk_employee_id FOREIGN KEY (employee_id) REFERENCES
Employees(employee_id);

ALTER TABLE Employees

ADD CONSTRAINT fk_dept_id FOREIGN KEY (dept_id) REFERENCES Departments(dept_id);

ALTER TABLE Managers

ADD CONSTRAINT fk_employee FOREIGN KEY (employee_id) REFERENCES
Employees(employee_id) ON DELETE CASCADE
```

In the provided code, this is applied in the Managers table, where the employee_id column references the Employees table. If an employee is deleted from the Employees table, any corresponding record in the Managers table is also automatically deleted. For example, if an employee

with employee_id = 1 is removed, the manager linked to this ID in the Managers table will also be deleted. This feature simplifies data management by automating cleanup processes and ensuring consistency across related tables.

The **DELETE** operation is used to remove rows from a table in a database. It allows you to delete specific rows based on a condition or even all rows in a table. For example:
-- Delete a specific employee

DELETE FROM Employees

WHERE employee_id = 34;

After:
We deleted an employee who is already manager. Hence it deleted also from managers table.

Options

	employee_id	first_name	last_name	email	phone_no	manager_id	job_id	dept_id
Edit Copy Delete	26	Zane	Adler	zane.adler@example.com	123-456-7915	3	3	3
Edit Copy Delete	27	Amy	Benson	amy.benson@example.com	123-456-7916	*NULL*	1	1
Edit Copy Delete	28	Brian	Carter	brian.carter@example.com	123-456-7917	2	2	2
Edit Copy Delete	29	Clara	Dunn	clara.dunn@example.com	123-456-7918	3	3	3
Edit Copy Delete	30	Derek	Ellis	derek.ellis@example.com	123-456-7919	1	1	1
Edit Copy Delete	31	Buse	Erdem	buse.erdem@example.com	513-456-7890	1	1	2
Edit Copy Delete	32	Berfin	Akdemir	berfin.akdemir@example.com	111-222-3333	2	2	1
Edit Copy Delete	33	Nida	Karatas	nida.karatas@example.com	523-456-7892	3	4	4
Edit Copy Delete	35	Ian	Walker	ian.walker@example.com	123-456-7895	*NULL*	4	4
Edit Copy Delete	36	Olivia	Parker	olivia.parker@example.com	987-654-3210	1	2	1

Options

	manager_id	employee_id	first_name	last_name	email	phone_no
Edit Copy Delete	1	31	Buse	Erdem	buse.erdem@example.com	503-456-7890
Edit Copy Delete	2	32	Berfin	Akdemir	berfin.akdemir@example.com	513-456-7891
Edit Copy Delete	3	33	Nida	Karatas	nida.karatas@example.com	523-456-7892

DELETE FROM JobHistory

WHERE employee_id = 2 AND latest_work_experience = '3 years HR Experience';

After:

Options

	employee_id	latest_work_experience	projects	education	start_date	end_date
Edit Copy Delete	1	5 years Software Engineering	Project Delta	BSc in Computer Science	2015-06-01	2020-06-01
Edit Copy Delete	3	7 years Sales	Project Gamma	BBA	2013-06-01	2020-06-01

The **UPDATE** operation is used to modify existing records in a table. You can update specific columns for rows that meet a condition.

UPDATE Employees

SET phone_no = '111-222-3333'

WHERE employee_id = 32;

After:

	employee_id	first_name	last_name	email	phone_no	manager_id	job_id	dept_id
Edit Copy Delete	26	Zane	Adler	zane.adler@example.com	123-456-7915	3	3	3
Edit Copy Delete	27	Amy	Benson	amy.benson@example.com	123-456-7916	*NULL*	1	1
Edit Copy Delete	28	Brian	Carter	brian.carter@example.com	123-456-7917	2	2	2
Edit Copy Delete	29	Clara	Dunn	clara.dunn@example.com	123-456-7918	3	3	3
Edit Copy Delete	30	Derek	Ellis	derek.ellis@example.com	123-456-7919	1	1	1
Edit Copy Delete	31	Buse	Erdem	buse.erdem@example.com	513-456-7890	1	1	2
Edit Copy Delete	32	Berfin	Akdemir	berfin.akdemir@example.com	111-222-3333	2	2	1
Edit Copy Delete	33	Nida	Karatas	nida.karatas@example.com	523-456-7892	3	4	4
Edit Copy Delete	34	Zeynep	Kanar	zeynep.kanar@example.com	543-456-7893	4	1	2
Edit Copy Delete	35	Ian	Walker	ian.walker@example.com	123-456-7895	*NULL*	4	4

UPDATE Jobs

SET job_title = 'Senior Data Scientist'

WHERE job_id = 5;

After:

	job_id	job_title	salary
Edit Copy Delete	1	Software Engineer	95000.00
Edit Copy Delete	2	HR Specialist	60000.00
Edit Copy Delete	3	Sales Manager	75000.00
Edit Copy Delete	4	Marketing Analyst	65000.00
Edit Copy Delete	5	Senior Data Scientist	95000.00

Conclusion

In conclusion, the project focused on developing a Human Resources Database Management System, tailored to manage HR-specific data and processes efficiently. We successfully transformed our abstractly designed relational database, initially constructed using an ER diagram, into a functional computerized system through ER-to-Relational Mapping. Based on our design framework, we created 8 tables and established 7 relationships of various types, including 1-to-1 and 1-to-N, ensuring comprehensive data representation and interconnectivity.

To maintain referential integrity, primary keys, foreign keys, and unique attributes were meticulously defined, and necessary constraints were implemented. The database was further normalized to eliminate potential anomalies, reduce data redundancy, and optimize storage efficiency, aligning with best practices in database design.

The system was developed using MySQL, leveraging its robust capabilities for relational database management and its compatibility with complex queries and large datasets. Several critical queries were executed to rigorously test the database's functionality and performance, confirming its ability to handle HR-specific tasks effectively.

Overall, the project demonstrates the successful application of database design principles in creating a fully operational HR Database Management System that is both efficient and ready for deployment in real-world scenarios.