Payroll Management System

Daniel Chang

SFSU ID: 921174056

Github: danieljchang

Checkpoint #	**Date Submitted**
Checkpoint 1	2/12/2023
Checkpoint 2	2/26/2023
Checkpoint 3	3/13/2023
Checkpoint 4	4/4/2023
Checkpoint 5	4/4/2023
Checkpoint 6	4/10/2023
Checkpoint 7	4/17/2023

Table of Contents

Project Description

- Payroll Database Management System

The payroll Database management system aims to improve the payment of employees in a more organized and timely manner. To ensure that whether it is entering hours, paid time off, sick days, vacation days, any form of hourly wages and/or salaries. This payroll management system will tackle the problems of managing its users and employees, creating fixating the task into something that will avoid any errors in calculations.

The advanced functionality of this payroll database management system will allow for functionality across all industries, it will manage to track hours as well as payment that is deserved to individuals. Those who work overtime will be rightfully paid their wages and will be automated to the exact amount. On top of ensuring that employees are being paid on time it also keeps track of the payment method of individuals, for example a direct deposit, check, electronic transfer etc. By collecting data on individual employees, we also get a general idea about who deserves a raise/promotion. The number of hours does not always correlate with the quality of work being done, but by understanding the effort of employees the employer will learn more about everyone. The ability to track the payment of individuals will also enforce government taxes. Since everything is recorded properly, it creates additional tax records and tax files on hand. It ensures that the company is working legally, as well as its employees. Lastly, but not limited to, it creates a detailed record of the financial status of a company in terms of its employees. Many companies will likely have a lot of flow of employees, this will give a record and a financial estimation on how much employees are paid and how much more they can spend on more employees. This also allows for when future jobs are created or people are needed for a different task, they have a budget that they can easily calculate due to the knowledge of how

much they already need to pay their employees. This will result in less evasion of paychecks and ensuring the company is responsible for the number of employees they have and the ability to pay each and every one of them. Many of the top products that already exist, include HR support, but lack recruitment. With my current product we have created a hiring advantage that could make pay anonymously but also exact amounts that new recruits could be making. Unlike Rippling, we can create transparency with its employees and employer, as well as keep the security of employees. We can create opportunities to attract job seekers that are looking for a company that they can trust. In addition, these payroll management systems are unaware of 401k's and 529 college savings. By having a payroll management system, not only do we track the users pay but also it allows for the company to match 401k and 529 college savings, instead of having to submit forms and files in order for it to be processed which only takes time, resulting in the company sitting on an individuals money. Lastly, like all payroll management systems that can be found online, are third party sources. Rippling, ADP, Gusto, etc, are all examples of third party systems, this means that we must pay outsiders to use their products instead of investing in our own payroll system, that would match the needs of the company itself rather than having to find something that matches perfectly to the desires of a companies future payroll management system. Many of these features are designed to ease the workload and lessen that natural error made by humans. By allowing for human resources to reduce the number of mistakes they could make which may affect many employees payroll, it becomes a source of trust that the employer and employees are expected to be on time and without error. It decreases the number of files that need to be kept, as well as the personal preferences of individuals across the company. It reduces the complexity of finding information, and bringing it all to one place for the best and most efficient method of organization.

Use Cases

Use Case 1:

Title: Payment status for an employee

Actors: Employee (Ryan)

Description: Ryan is a new salary worker that has just been hired by the company. He is living paycheck by paycheck and can't afford to have any errors in his paycheck or any delays. His last job also involved the company delaying payments and needs to a reliable time that he can expect his monthly payments. Going through the program and finding his status, he is able to find when and where he is expected to be paid, how much, and where the paycheck will be deposited. Once a month, he will receive an email saying his paycheck is either ready, or it has already been deposited directly into his bank account. By using the application he is able to check and confirm his paycheck status.

Use Case 2:

Title: Employee asking for PTO

Actors: Employee (Chris), Manager (Kyle)

Description: The employee Chris is trying to use his yearly PTO to take a few days off to travel. In order for him to request PTO, he is requesting directly to his manager, Kyle, and must wait for his response. Chris has forgotten how many days he has already taken off, but yet he still wants to take more time off. Chris has been a great employee and is completing more goals compared to all of his colleagues. Kyle can check Chris's status by searching for his accomplishments, his

performance reviews and how much time he has already taken off. Kyle will then determine whether to accept or reject his status based on his status. Chris continued to work hard for the next couple of weeks. After checking Chris's status, he has not taken any time off this year, in other words, has not reached his maximum number of days taken off. Kyle also checks for the days that Chris has requested to take off and ensures that it is not during an important week where multiple things are due or important meetings are expected. Kyle decides to approve Chris's PTO since his status has proved him to be a valuable member of the team and deserves a break.

Use Case 3:

Title: Employee is being Overworked

Actors: Employee (Tyler), Manager (Kyle)

Description: Tyler is currently working for the company and has been for the past 2 years. He has always enjoyed his time at the company and loves his work. He continues to pick up more and more projects. Tyler has a great performance rating, and has been a dedicated employee of the company. But because he has been doing so much he has requested a pay raise from his manager Kyle. Kyle searches up Tyler's status and finds out that Tyler is being underpaid for work he has been doing. Kyle also realizes that for the 2 years that Tyler has been at the company he has not received a raise. Kyle not only gives Tyler a raise, but he also ensures that Tyler is not being overworked since he has picked up so many projects.

Use Case 4:

Title: Working overtime as a wage worker

Actors: Wage employee (Will) layoff

Description: Will is a wage worker, and has been with this company for a few years. Will has been known to slack off while on the clock, and is not the best of employees. The economy has recently crashed, and the company is looking to lay off a few of their employees. By searching through their payroll management database, the company is able to locate the employees with the least number of projects completed, and the lowest performance rating. By searching for the department that is the least necessary, they decided to lay off the bottom 20% of Will's department. Sadly, that also includes, Will who know he has been doing almost nothing while on the clock, still expects to be paid the money that he deserves. The company will use its automatic payroll system to calculate the number of hours Will has worked and will send him a paycheck for his deserved hours.

Use Case 5:

Title: Switching departments

Actors: Employee (Charles)

Description: Charles recently wanted to switch deparmtnets in the company. He is currently in the research and development team at the current company. But he has been in this department for 5 years. He has decided that he wants to pursue a different topic and enter a different field at the company that he currently works at. Instead of doing research and development he wants to work on the hardware team to design and develop products. He has asked his manager if it is possible to switch departments. Charles's manager accepts his request but only temporarily. For the next couple weeks Charles will need to prove himself to his manager that he is capable and he has enough knowledge that allows him to work in that field. After a couple of weeks Charles

receives his monthly performance rating, and he does not have a high enough rating for him to switch, but it is still high enough for him to be able to switch eventually. As a suggestion, Charles's manager tells him to continue working in the R&D team but continue to work with the hardware team when he has time. This would not only improve Charles' skills, but also eventually allow him to fully switch into a department that he enjoys. Ultimately, this may affect his salary, because he is not as good at development compared to R&D.

Functional Database Requirements

1. User

    1.1. A user shall create only one account.

    1.2. A user shall be part of at least one department.

    1.3. A user shall have at least 0 or more hours.

    1.4. A user shall have at least one payment method.

    1.5. A user shall have only one employee number.

    1.6. A user shall have a unique calendar.

    1.7. A user shall have one tax record.

    1.8. A user shall have zero or one hourly wage.

    1.9. A user shall have zero or one salary.

    1.10. A user shall have one w2 form.

    1.11. A user shall have one I9 form.

    1.12. A user shall have many managers.

    1.13. A user shall have many projects.

    1.14. A user shall have visa information/status.

2. Employee

    2.1. An employee shall be a user.

    2.2. An employee shall have a wage.

    2.3. An employee shall be assigned to the HR team but not to the labor team, sw engineer, testing team.

    2.4. An employee shall only have a manager which is also an employee.

3. Manager

3.1. A manager shall be an employee, but an employee shall not have to be an employee.

3.2. A manager shall have a wage.

4. Account

4.1. An account shall be created by only one user.

4.2. An account shall many payment methods in order of priority.

4.3. An account shall many layers or security

4.4. An account shall one unique employee number.

4.5. An account shall have only one connected email address

5. Role

5.1. A user shall have at least one role.

5.2. A user shall permissions dependent on the role

5.3. A role will have many positions.

5.4. A role will have many departments.

5.5. A role will have many employees.

6. Benefits

6.1. A user shall have only one 401(k) plan.

6.2. A user shall have only one health insurance plan.

6.3. Benefits shall have at least one plan.

7. Calendar

7.1. A calendar shall have automatic scheduled reports.

7.2. A calendar shall have updated holidays.

7.3. A calendar shall have specified dates for work dates.

7.4. A calendar shall have a specified date for paychecks.

7.5. A calendar shall have automated reminders.

8. Expenses

8.1. Expenses shall have many automated reminders.

8.2. Expenses shall have one reimbursement.

8.3. Expenses shall have many digital receipts.

8.4. Expenses shall be approved by at least one manager.

9. Taxes

9.1. Taxes shall have many official tax values.

9.2. Taxes shall have one state tax.

9.3. Taxes shall have one federal tax.

9.4. Taxes shall have one local tax.

9.5. Taxes shall have many tax reports.

9.6. Taxes shall have one verification of tax payment.

9.7. Taxes shall have one tax credit.

9.8. Taxes shall have many W2 forms.

9.9. Taxes shall have many withholds on payroll taxes from earnings.

9.10. Taxes shall have many voluntary deductions.

9.11. Taxes shall have many wage garnishments.

9.12. Taxes shall have many earning codes in compliance with IRS guidelines.

10. Time

10.1. Times shall have employee scheduling.

10.2. Times shall have one timesheet entries.

10.3. Times shall track many overtime hours.

10.4. Times shall have at least one method of time tracking.

10.5. Times shall have many dates in the format, mm/dd/yyyy.

10.6. Times shall have many hours

11. Payroll

11.1. Payroll shall have many pay groups.

11.2. Payroll shall have a tax calculator.

11.3. Payroll shall have many salary structures.

11.4. Payroll shall have medical withholdings.

11.5. Payroll shall have tax withholdings.

Non-Functional Requirements

1. System Requirements

1.1. System shall work with IOS, Windows, or Linux

1.2. System shall work with 64-bit computing

1.3. System shall work with 32 bit computing

2. Performance

2.1. Queries shall return data in at least 10ms.

2.2. The database system shall be extremely flexible, and easy to modify.

2.3. The database system shall have accurate computations for wages.

2.4. The costs shall be cheaper than manual work.

2.5. The database system shall be easy to use.

2.6. Throughput is fast.

3. Storage

3.1. The database system shall assign 10mb of memory per table.

3.2. The database system should support persistent storage.

3.3. The database system should be able to support an increase in size.

3.4. The database system shall be dynamically programmed.

4. Security

4.1. Only encrypted passwords shall be supported by the database system.

4.2. All the values inserted into the database shall be consistent with the datatypes.

4.3. The database shall automatically backup every day at 11:59pm.

4.4. The database shall record all transactions in a logs file.

4.5. The database shall keep individual hours as well as projects or any personal information a secret.

4.6. The database shall only provide data to those with high enough roles.

5. Environmental

5.1. Ease of access

5.2. Usability

5.3. The program will be online, no paper, no pen, no pencil.

6. Legal

6.1. The database shall ensure tax credibility and ensure that employees maintain the integrity of the company.

6.2. The credentials and officiality of hours submitted.

6.3. Hours worked is proportional to work done/projects finished.

7. Content

7.1. The database system shall have the ability to section off information that is needed for individual employees.

7.2. The database system shall have private information for the user, and public information of the company.

7.3. The database system shall allow scheduling with other employees.

8. Compatibility

8.1. Each table shall be assigned 2mb of memory.

8.2. Each table shall have specific data types.

8.3. Each table shall be specific to protect a user's information.

8.4. The ability to long in from different devices.

8.5. One device at a time.

9. Organizational

9.1. The database system shall have a find or search function to find specified keys.

9.2. The organization of employees and their roles

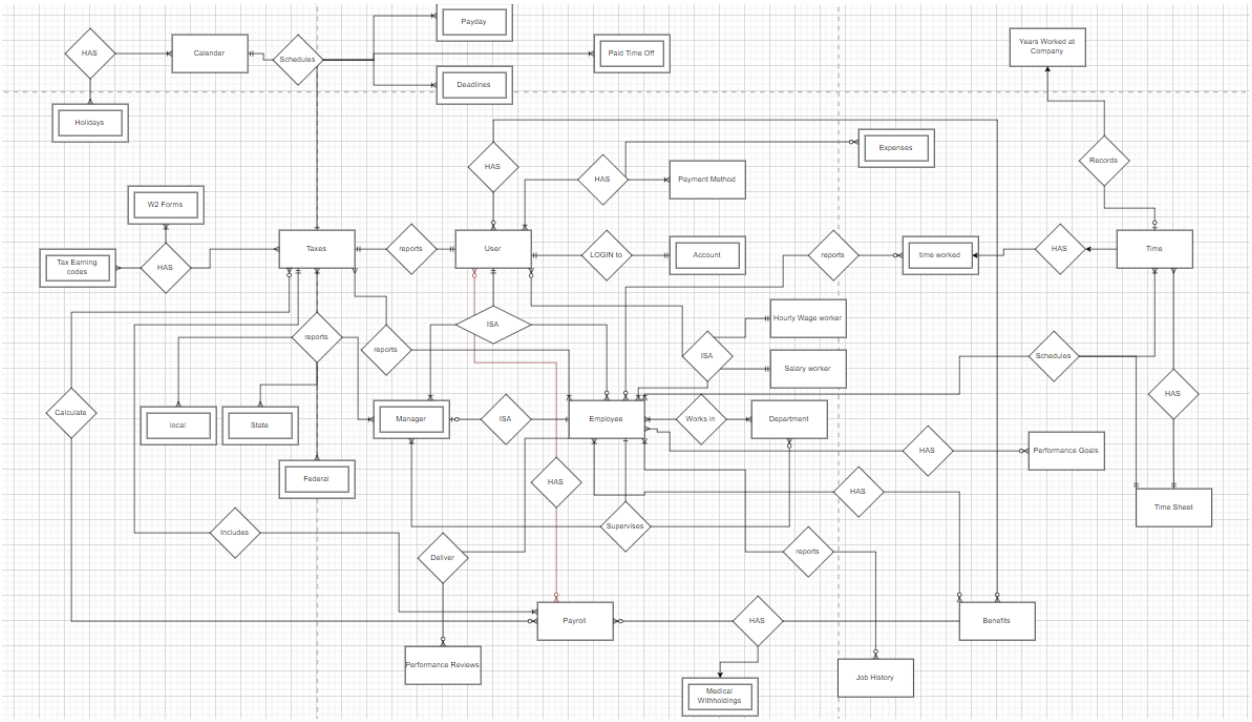
10. Scalability

10.1. The database system shall have proper bandwidth to support large companies.

10.2. The database system shall have a management system that groups departments.

10.3. The database system shall be dynamic.

Entity Relationship Diagram



Entity and Attributes Description

1. user: Strong

- id (INT, Primary Key)
- login_time (TIMESTAMP)
- create_time (TIMESTAMP)
- username (VARCHAR(45), Not Null)
- email (VARCHAR(45), Not Null)
- password (VARCHAR(45), Not Null)

2. manager: Strong

- id (INT, Primary Key)
- name (VARCHAR(45), Not Null)

- manager_level (INT ZEROFILL, Not Null)

3. Employee Strong

- id (INT, Primary Key)
- name (VARCHAR(90), Not Null)
- major (VARCHAR(45), Not Null)
- employee_level (INT ZEROFILL, Not Null)
- is_wage (TINYINT, Not Null, Default: 0)
- last_bonus (TIMESTAMP)

4. time_worked: Strong

- amt_time (INT, Not Null)
- employee_id (INT, Not Null, Foreign Key referencing Employee.id)

5. payment_method Weak

- other (VARCHAR(45), Not Null, Default: 'none')
- direct_deposit (TINYINT, Not Null, Default: 0)
- checks (TINYINT, Not Null, Default: 0)
- employee_id (INT, Not Null, Foreign Key referencing Employee.id)

6. paid_time_off: Weak

- other (VARCHAR(45), Not Null, Default: 'none')
- direct_deposit (TINYINT, Not Null, Default: 0)
- checks (TINYINT, Not Null, Default: 0)
- employee_id (INT, Not Null, Foreign Key referencing Employee.id)

7. payroll: Strong

- id (INT, Not Null)

- employee_id (INT, Not Null, Foreign Key referencing Employee.id)

8. department: Strong

- id (INT, Primary Key)
- funding (INT ZEROFILL, Not Null)
- num_employees (INT ZEROFILL, Not Null)
- funding_saved (INT ZEROFILL, Null)
- department_name (VARCHAR(45), Not Null, Default: 'none')

9. taxes: Strong

- id (INT, Primary Key)
- funding (INT ZEROFILL, Not Null)
- num_employees (INT ZEROFILL, Not Null)
- funding_saved (INT ZEROFILL, Null)
- department_name (VARCHAR(45), Not Null, Default: 'none')

10. W2: Weak

- amount_earned (INT ZEROFILL, Not Null)
- amount_withheld (INT ZEROFILL, Not Null)
- taxes_id (INT, Not Null, Foreign Key referencing Taxes.id)

11. expenses: Strong

- id (INT, Primary Key)
- amount (INT ZEROFILL, Not Null)
- spent_on (VARCHAR(45), Not Null)

12. performance_goals: Weak

- id (INT, Primary Key)

- amount (INT ZEROFILL, Not Null)
- spent_on (VARCHAR(45), Not Null)

13. salary_worker: Strong

- id (INT, Primary Key)
- amount (INT ZEROFILL, Not Null)
- spent_on (VARCHAR(45), Not Null)

14. hourly_worker: Strong

- id (INT, Primary Key)
- amount (INT ZEROFILL, Not Null)
- spent_on (VARCHAR(45), Not Null)

15. project: Strong

- id (INT, Not Null)
- name (VARCHAR(45), Not Null)
- department_id (INT, Not Null, Foreign Key referencing Department.id)

16. deadlines: Strong

- id (INT, Not Null)
- deadline (DATETIME, Not Null)
- late (TINYINT, Not Null)
- project_id (INT, Not Null, Foreign Key referencing Project.id)

17. calendar: Strong

- id (INT, Not Null)
- deadline (DATETIME, Not Null)
- late (TINYINT, Not Null)

- project_id (INT, Not Null, Foreign Key referencing Project.id)

18. holidays: Weak

- id (INT, Not Null)
- holiday_name (VARCHAR(45), Not Null)
- day_off_in_us (DATE, Not Null)
- day_off_other (DATE, Not Null)
- calendar_id (INT, Not Null, Foreign Key referencing Calendar.id)

19. payday: Weak

- day_to_pay (DATE, Not Null)
- employee_paid (TINYINT, Not Null)
- calendar_id (INT, Not Null, Foreign Key referencing Calendar.id)

20. performance_reviews: Strong

- day_to_pay (DATE, Not Null)
- employee_paid (TINYINT, Not Null)
- calendar_id (INT, Not Null, Foreign Key referencing Calendar.id)

21. time_stayed: Weak

- start_date (DATE, Not Null)
- time_on_project (INT, Not Null)
- employee_id (INT, Not Null, Foreign Key referencing Employee.id)
- calendar_id (INT, Not Null, Foreign Key referencing Calendar.id)
- calendar_deadlines_id (INT, Not Null, Foreign Key referencing Calendar.deadlines_id)
- calendar_deadlines_project_id (INT, Not Null, Foreign Key referencing Calendar.deadlines_project_id)

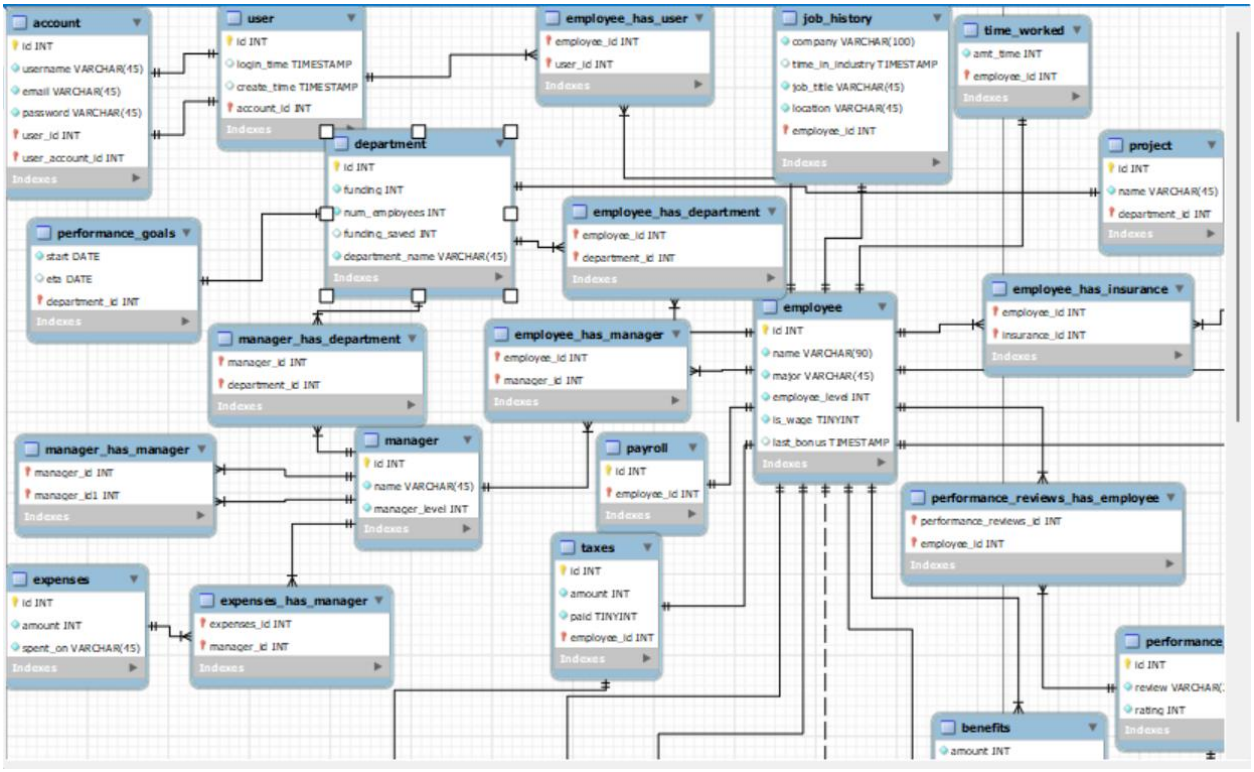
22. job_history: Weak

- company (VARCHAR(45), Not Null)
- time_in_industry (INT, Not Null)
- job_title (VARCHAR(45), Not Null)
- location (VARCHAR(45), Not Null)
- employee_id (INT, Not Null, Foreign Key referencing Employee.id)

23. time_sheet: Strong

- id (INT, Primary Key)
- time_start (DATETIME, Not Null)
- time_end (DATETIME, Not Null)
- employee_id (INT, Not Null, Foreign Key referencing Employee.id)

Entity Establishment Relationship Diagram (EER)



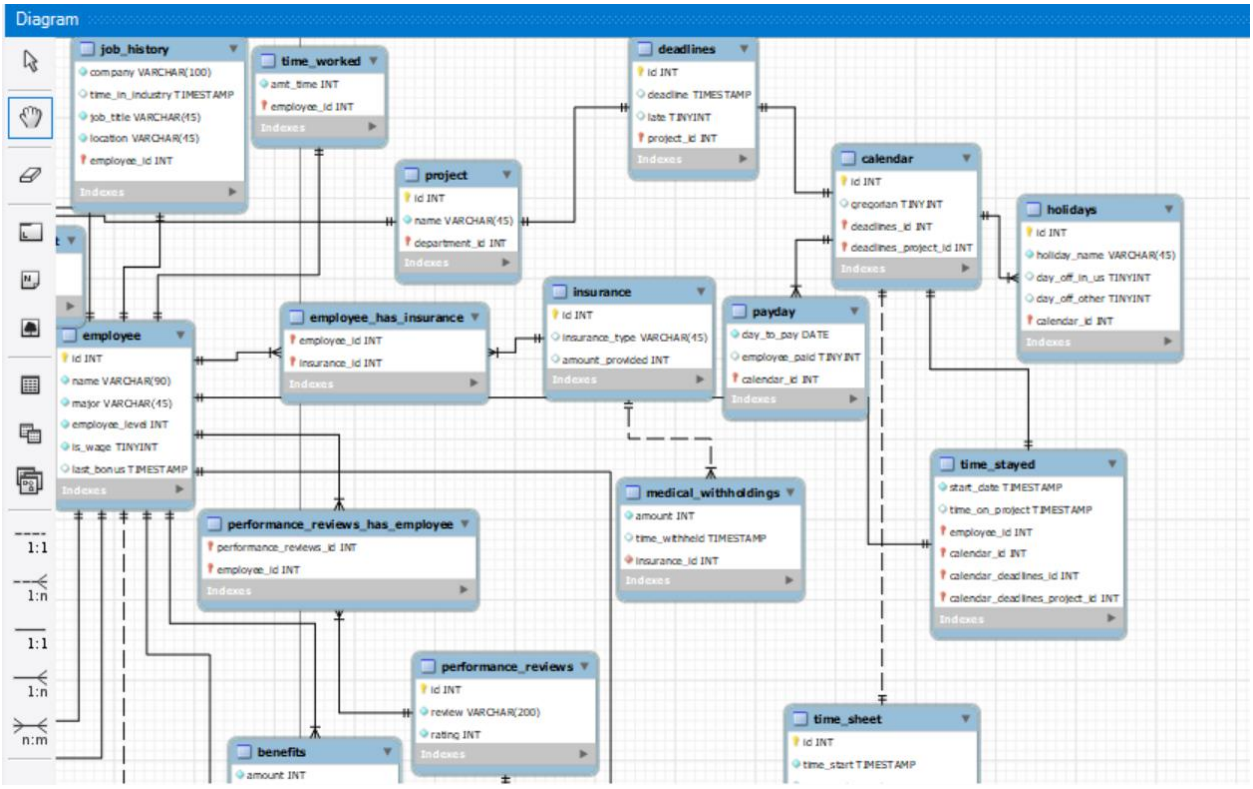
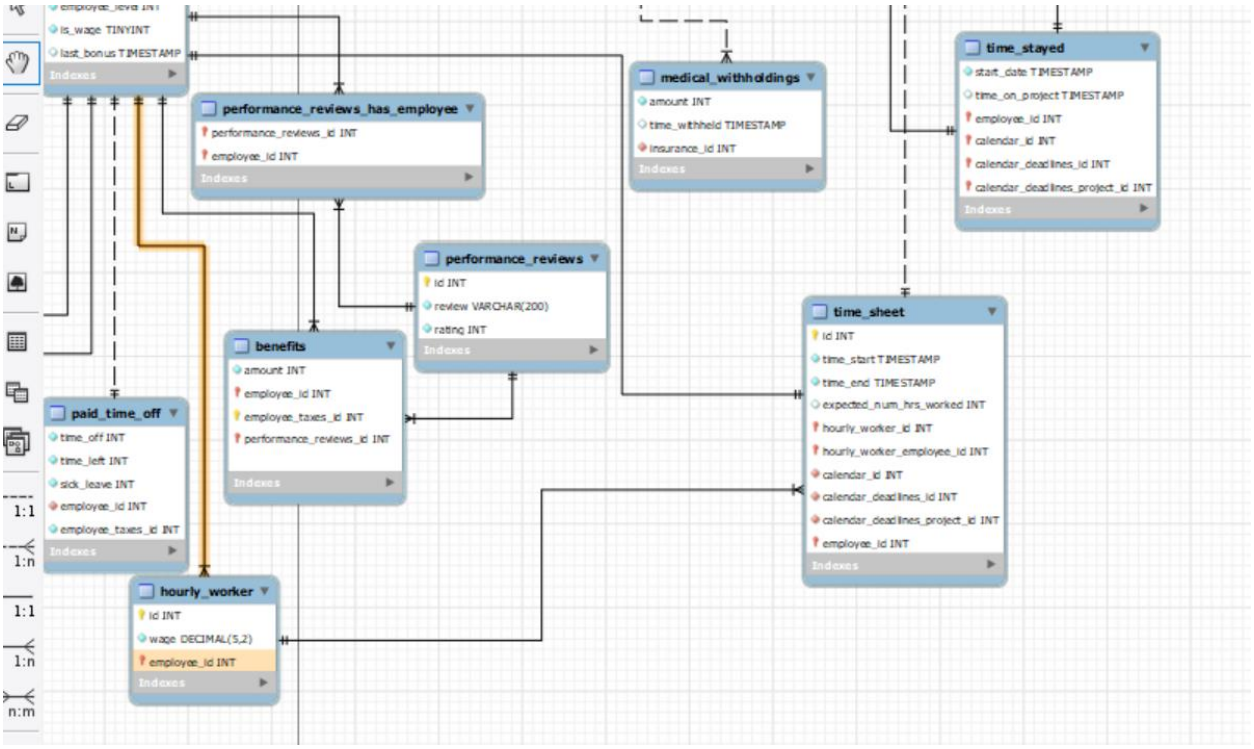
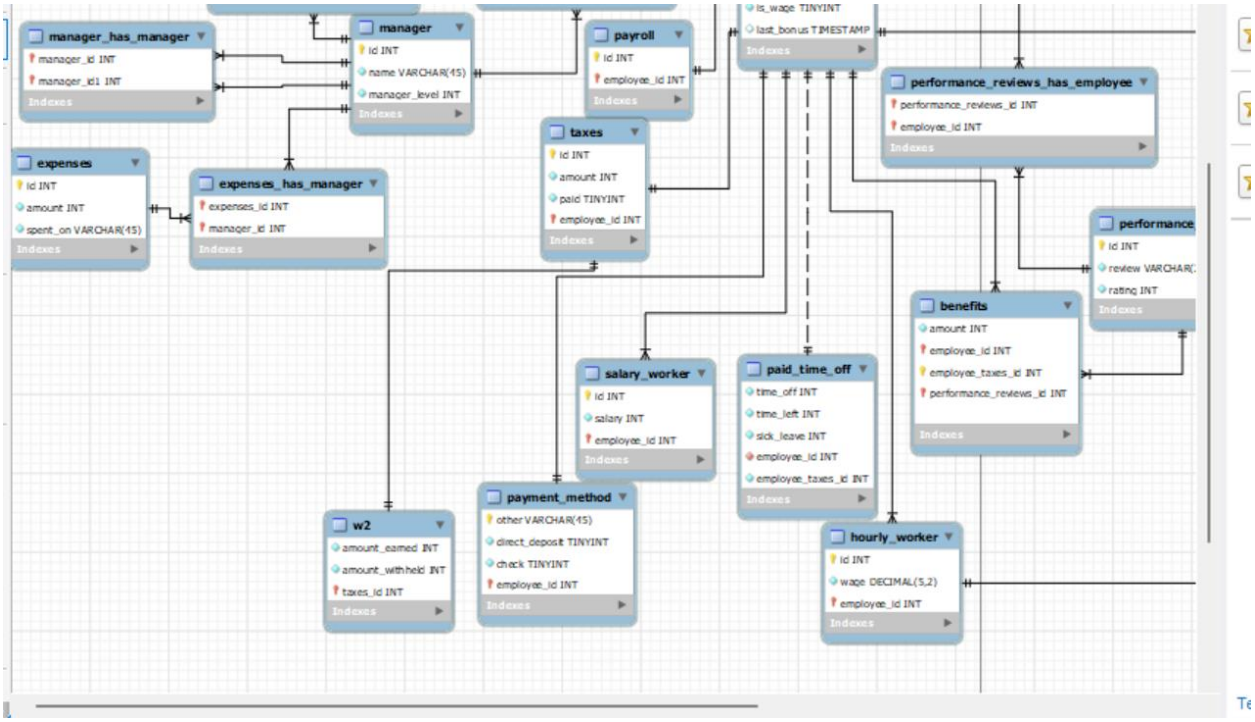


Diagram
job_history
company VARCHAR(100)
time_in_industry TIMESTAMP
job_title VARCHAR(45)
location VARCHAR(45)
employee_id INT
time_worked
amt_time INT
employee_id INT
deadlines
id INT
deadline TIMESTAMP
late TINYINT
project_id INT
project
id INT
name VARCHAR(45)
department_id INT
calendar
id INT
gregorian TINYINT
deadlines_id INT
deadlines_project_id INT
holidays
id INT
holiday_name VARCHAR(45)
day_off_in_us TINYINT
day_off_other TINYINT
calendar_id INT
employee
id INT
name VARCHAR(90)
major VARCHAR(45)
employee_level INT
is_wage TINYINT
last_bonus TIMESTAMP
employee_has_insurance
employee_id INT
insurance_id INT
insurance
id INT
insurance_type VARCHAR(45)
amount_provided INT
payday
day_to_pay DATE
employee_paid TINYINT
calendar_id INT
time_stayed
start_date TIMESTAMP
time_on_project TIMESTAMP
employee_id INT
calendar_id INT
calendar_deadlines_id INT
calendar_deadlines_project_id INT
medical_withholdings
amount INT
time_withheld TIMESTAMP
insurance_id INT
performance_reviews_has_employee
performance_reviews_id INT
employee_id INT
performance_reviews
id INT
review VARCHAR(200)
rating INT
time_sheet
id INT
time_start TIMESTAMP
benefits
amount INT
1:1
1:n
1:1
1:n
n:m


is_wage TINYINT
last_bonus TIMESTAMP
medical_withholdings
amount INT
time_withheld TIMESTAMP
insurance_id INT
time_stayed
start_date TIMESTAMP
time_on_project TIMESTAMP
employee_id INT
calendar_id INT
calendar_deadlines_id INT
calendar_deadlines_project_id INT
performance_reviews_has_employee
performance_reviews_id INT
employee_id INT
performance_reviews
id INT
review VARCHAR(200)
rating INT
time_sheet
id INT
time_start TIMESTAMP
time_end TIMESTAMP
expected_num_hrs_worked INT
hourly_worker_id INT
hourly_worker_employee_id INT
calendar_id INT
calendar_deadlines_id INT
calendar_deadlines_project_id INT
employee_id INT
benefits
amount INT
employee_id INT
employee_taxes_id INT
performance_reviews_id INT
paid_time_off
time_off INT
time_left INT
sick_leave INT
employee_id INT
employee_taxes_id INT
hourly_worker
id INT
wage DECIMAL(5,2)
employee_id INT
1:1
1:n
1:1
1:n
n:m



Constraints Description

Table	FK	On Delete	On Update	Comment
Account	user	ON CASCADE	ON CASCADE	If a user is deleted, then the account should be deleted with it.
User	employee	SET NULL	ON CASCADE	If employee is deleted, the user should not be deleted.
User	employee	SET NULL	ON CASCADE	If employee is deleted then the user is will still exist within the account.
Department	Project	SET NULL	ON CASCADE	Should not delete all projects

				and all departments.
Department	Manager	Set NULL	ON CASCADE	If the manager is deleted, the department will still exist.
Employee	Department	SET NULL	ON CASCDE	If the department is deleted then everything is deleted, due to no department existing but the employees still exist
Employee	Salary Worker	ON CASCDE	ON CASCADE	Removes only this specific salary worker. But should also remove the account connected
Employee	Wage Worker	ON CASCADE	ON CASCADE	Removes only the specific worker. But should also remove the account connected
Benefits	Employee	ON CASCADE	ON CASCADE	If the employee is deleted, then the benefits should also be deleted.
Employee	Taxes	ON CASCADE	ON CASCDE	If taxes are deleted then employee should not exist because employees MUST pay taxes.
Expenses	Department	SET NULL	ON CASCADE	If the department is deleted, then the expenses should still exist since it is money

				previously spent and data that must be recorded.
Expenses	Manager	SET NULL	ON CASCADE	If the manager is deleted, then the expenses should still exist, because a department will still have expenses even without a manager.
Taxes	W2	SET NULL	ON CASCADE	If W2 is deleted, taxes will still exist as other ways.
Taxes	Tax Earning Codes	SET NULL	ON CASCADE	Should still exist even after being deleted, because taxes come from income.
Employee	Job History	SET NULL	ON CASCADE	If a user does not have a job history. Then the employee will have no job history but that doesn't change the abilities of the employee.
Employee	Payroll	ON CASCADE	ON CASCADE	If payroll is deleted then no employee is being paid and thus no employees are working.
Employee	TimeStayed	ON CASCADE	ON CASCADE	If time timestayed is no longer existing then the employee no

				longer works at the company.
Calendar	Payday	SET NULL	ON CASCADE	If payday is deleted, then the calendar still checks for holidays and other days/time
Calendar	Deadlines	SET NULL	ON CASCADE	The deadlines will be deleted, that means the projects will not have specified dates to be finished. But the calendar itself will not be deleted
Project	Deadlines	SET NULL	ON CASCADE	Without deadlines projects will no longer have an expected finish time, but the projects will continue to exist and need to be worked on or can be worked on later.

Business Requirements

1. “For every employee, the user shall see the number of hours worked greater than <X> only for employees that took more than <N> sick days from year <V> to <Z>. Both inclusive”
2. “For every employee, the user shall see the number of hours worked on a given project where <Y> the project id. And the number of hours the employee is expected to work on the project.”
3. "For every employee, the user shall find all employees on the department <x>”
4. “For every employee, the user shall find all employees, who have not met their expected goals, and the expected salary the employee is being paid.”
5. “For every employee, the user shall find all employees that have completed project <z> within the time frame <X> to <Y> (both inclusive), and the if they are being paid more than <A>.”
6. “For every employee, the user shall be able to find all managers that have <X> employees working under them and how many <Y> projects were completed under the department with id <C>.”
7. For every employee, the user shall find the project with the closest deadline they are assigned to within <X> department, and <y> number of projects”
8. “For every employee, the user shall find all the amount they are going to be paid for a given time period with the salary that they have. Where <x> is the minimum salary selected“
9. “For every employee, the user shall be able to find the expected amount they are being paid given they work for <X> hours given the employees current wage.”

10. “For every employee, the user shall be able to find the past wages of employees in department <Y> searched by the number of <X> years the employee has worked at the company.”
11. “For every employee, the user shall be able to see the current wages of employees if they meet the given requirements based on position and they work in department <X>. (If the user is a manager, then they can check the current wages of the people that work under them, but not people in higher positions or in different departments) <Y> Where it is the specific employee and is working under manager <T>”
12. “For every employee, the user shall find the benefits that they meet the minimum requirements for given their performance rating and the benefits they have taken previously.”
13. “For every employee, the user shall find the <X> insurance plan that the employee can receive given that they have stayed <Y> years at the company, and that they have not changed their insurance plan within the last year.”
14. “For every manager, the user shall find when project <X> was started, and the budget that is given initially and how much is left give the current date of the search.”
15. “For every manager, the user shall find the job history of every employee in department <X> and the projects that the employee has worked on.”
16. “For every employee, the user shall find the employees that have taken <Z> hours paid time off, and when project <X> that the employee participates in is due, given that they have not already taken off <Y> hours from work given sick leave.”

17. “For every manager, the user shall find the employees that have job history of <X> years working in department <Y>, given that they have completed <Z> projects related to the field <S>.”
18. "For every benefit, the number of hours shall be greater than <X> and they must have completed <n> projects to claim the benefits."
19. “For every payroll, the employee will find how much they are expected to be paid in a given time period from department <x>”
20. "For every insurance benefit, the manager shall find when the employee last changed their insurance plan."
21. "For every hourly worker, the time sheet will display all users, that have more than <x> hours and a higher than <y> wage that has a performance rating greater than <z>."