Future Planning

Part 1. Schema Evolutions

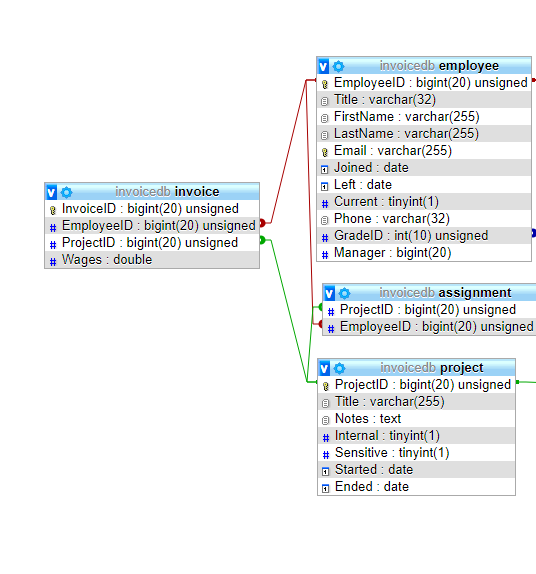
The most substantial change that I would make to the schema proposed is the system in which employees are paid, currently the only way in which their pay is considered is through the ‘payslip’ table in the ‘NetPay’ field. I feel this is an issue as there is no record of where that NetPay comes from, this could lead to issues with mistakes or the employee is paid the wrong amount there is no record of the correct amount of pay.

My proposal:

I propose the addition of an invoice table which is linked to the project and employee to show how much an employee was paid for working on a certain job

*Data Dictionary on next page*

ER Diagram in relation to invoice



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Database*** | WWIHR | ***Table Name*** | Invoice | ***(Composite)Key Field*** | Invoice\_ID |

**Related to:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table Name*** | ***Foreign Key*** | ***Table Name*** | ***Foreign Key*** |
| Project | ProjectID | Employee | EmployeeID |

|  |
| --- |
| **General table description: Shows the amount paid to an employee for a job** |

Minor Changes to schema:

* Add a loanID to `equipmentloan` table to allow for a loan to be tracked and searched for.
* Add a select statement before assigning an employee to a project to remove the employees that do not currently have a contract to make sure only employees with a current contract can be assigned a project.

Part 2. Security, performance, dependability

Preventing clashes with multiple users editing the database;

When a user is editing the database, there should be a write lock across all tables in that database, this should prevent double editing and then incorrect details being written to the database and improve the integrity of the database.

Only giving certain permissions to certain grades of employees;

The use of views can be used in the database to create virtual tables that only show data that a certain grade of employee can see. For example, a HR employee would be able to see the entire database, but an intern would only be able to see a very basic version of the database with all the sensitive information visible, this can be done through ‘GRANT’ in SQL by granting views to certain grades of employee, using the grade system to grant views also means when a new employee joins and is given a grade, they already have the correct permissions to view the database, this increases the security of the database as there is automated authorisation.

Using stored procedures to increase performance and security

There should be a stored procedure for everything that can be done in the database, for example the addition of an employee, editing employee details, adding a payslip, etc. this will help as employees will not need to use as much SQL code and should increase their performance. It will also allow us to grant certain procedures to each grade of employee, meaning that say an intern cannot delete an employee, this will increase security through authorisation.

SQL injection prevention within stored procedures

Each stored procedure should have all user input escaped to prevent the user from executing queries outside of what is intended, this will increase security as users will not be able to step outside of their authorisation.

Auditing changes

All Changes made to the database should be saved to a log table, there should be a log query for each trigger, saving the current users ID, the current time and the procedure they are executing. This will give accountability to each user and certain grants can be taken from that user if their permissions are abused.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Field Name*** | ***R*** | ***I*** | ***U*** | ***Data Type*** | ***Length*** | ***Format Rule(s)*** | ***Default Value*** | ***Description*** | ***Typical Data*** |
| InvoiceID | Y | Y | Y | BIGINT |  |  |  | Invoice number | 12345 |
| ProjectID | Y | Y | N | BIGINT |  |  |  | Project number | 12345 |
| EmployeeID | Y | Y | N | BIGINT |  |  |  | Employee number | 12345 |
| Wages | Y | N | N | Double |  |  |  | Amount Paid | 1200.45 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Key:** | **R=Required** | Y | (Yes – NOT NULL) | N | (No) |
|  | **I=Indexed** | Y | (Yes – indexed) | N | (Not indexed) |
|  | **U=Unique** | Y | Yes (Unique) | N | (Not unique) |