UNIVERSIDAD ESTATAL A DISTANCIA

VICERRECTORIA ACADEMICA

ESCUELA DE CIENCIAS EXACTAS YNAURALES

CARRERA INGENIERÍA INFORMÁTICA

Proyecto No1

MODALIDAD ESCOGIDA: PROYECTO

Pablo André Valenciano Blanco
1-1572-0043

Curso: Base de Datos [00826]

Grupo #2

Profesor: Steven Brenes Torres

CENTRO UNIVERSITARIO DE HEREDIA

PAC: 2023-3

HEREDIA, 2023-10-08

Contenido

Introducción 3
Marco Teórico 4
Conceptos Básicos 4
Diagrama de Entidad-Relación 4
Explicación de las entidades 5
Explicación de la cardinalidad 8
Reportes 9
Normalidades 9
Primera Forma Normal 9
Segunda Forma Normal 9
Tercera Forma Normal 10
Sentencias 10
Conclusiones 15
Bibliografía 16

Tabla de Figuras

Figura 1.....4

Introducción

El siguiente documento presenta la realización del primer proyecto del curso de Base de Datos, la cual evaluara el tema 1 del curso, el cual es constituido por: Introducción a la base de datos, entornos de bases de datos, diseño de bases y modelo relacional, los conceptos y aprendizaje del libro de Fundamentos de bases de datos (2014).

El proyecto tiene como objetivo crear el modelo conceptual y físico de una base de datos relacional, esta base de datos estará definida según los requerimientos que la empresa o contrato definan, estos requerimientos pueden ser flojos o a disposición del ingeniero a cargo o más estrictas y según restricciones definidas de parte del cliente, usualmente no hay una solución totalmente correcta y pueden variar temporal o espacialmente según el ambiente interno, lógica de las relaciones o externo, tales como limitaciones del servidor de las bases de datos.

Este proyecto iniciara con una breve introducción de los conceptos relacionados al tema de base de datos, se darán las instrucciones pedidas por la empresa InnovaTECH y continuaremos con las especificaciones de cómo se diseña el modelo relacional que cumpla los cumplimientos adecuados a la normalidad e indicando la cardinalidad de sus atributos entre cada una de las entidades y las relaciones que se establecen entre ellas.

Ya con el modelo relacional normalizado se incluirán las sentencias para la creación de las bases de datos, para cada atributo se indicará su tipo y los constraints, y se pasará a la par de este documento el con los comandos SQL.

Marco Teórico

Conceptos Básicos

Base de datos: Según Oracle.com: “Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.”

Modelo Entidad-Relación: Según Lucid Chart: “es un tipo de diagrama de flujo que ilustra cómo las "entidades", como personas, objetos o conceptos, se relacionan entre sí dentro de un sistema. Los diagramas ER se usan a menudo para diseñar o depurar bases de datos relacionales en los campos de ingeniería de software, sistemas de información empresarial, educación e investigación.”

MySqL: Según Pablo Londoño en su blog: “es un sistema de administración de bases de datos relacionales. Es un software de código abierto desarrollado por Oracle. Se considera como la base de datos de código abierto más utilizada en el mundo.”

Diagrama de Entidad-Relación

El diagrama entidad-relación que se diseñó bajo los requerimientos es el siguiente:

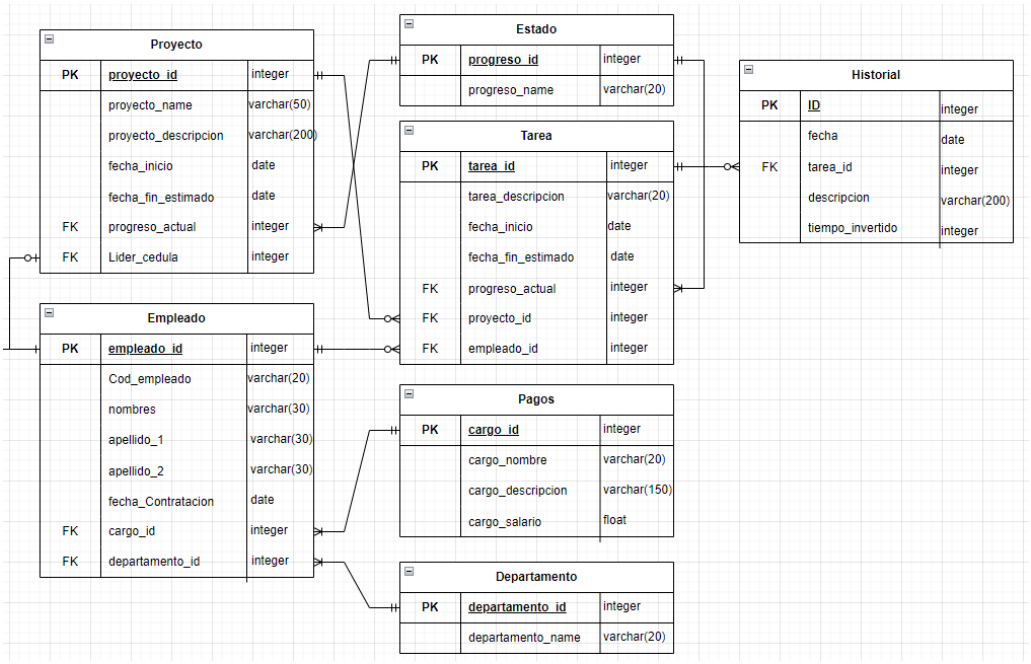

Figura 1. Diagrama Entidad-Relación Propuesto

Explicación de las entidades

La entidad principal para la base de datos propuesta son las tareas, ya que estas son el eje principal, debido a tres razones:

1. Los proyectos consisten de un grupo de tareas
2. Las tareas pueden ser asignadas a cada empleado.
3. El histórico de horas se hacen sobre las tareas y no sobre los proyectos.
4. El resto de identidades secundarias obligatorias, se conectan a esta de forma directa: proyecto, empleados e historial.

Como entidades secundarias importantes se tienen los proyectos, empleados e historial, estos son requerimientos del mismo proyecto, no todos se relacionan entre ellos, pero son importantes para el desarrollo de la base de daros, para el cumplimiento de los objetivos realizados por la empresa y los posibles reportes que en estas se generan.

De últimos tenemos las entidades agregadas, con ella se cumple lo mínimo de 5 entidades, para el desarrollo del proyecto, y ayudan también a que la empresa no sufra de redundancia y se cumplan las 3 normas de normalización. Las entidades de este tipo son: estado, pagos y departamento. Las 3 se crean con el objetivo de evitar problemas de escritura, ya que al inicio son variables de tipo varchar, pueden cometerse errores de digitación, por lo que es más sencillo digitarlo por medio de un número o identificador y que se restrinjan los posibles valores de este.

Ahora se explicaron los atributos para cada entidad:

1. Proyecto:
    a. proyecto_id: Identificador único de cada proyecto. Es un entero.
    b. proyecto_name: Nombre del proyecto. String de 50 caracteres
    c. proyecto_descripcion: Descripción del proyecto. String de 200 Caracteres
    d. fecha_inicio: Día en el cual inicia el proyecto. Tipo fecha
    e. fecha_fin_estimado: Fecha estimada del fin de proyecto. Tipo fecha

f. progreso_actual: Progreso modificable, solo posee 4 valores que se explican en la entidad Estado. Tipo entero.

g. Lider_cedula: Identificador del líder de proyecto, se decide poner en esta entidad, para evitar valores nulos en la entidad empleado. Tipo entero.

2. Empleado:

a. empleado_id: Numero único referente a la cedula del empleado en cuestión. Solo puede ser entero.

b. Cod_empleado: código único de cada empleado activo, asignado por la empresa, según disposiciones de la empresa se podrían dar repeticiones, al ingresar o despedir personal, esto debido a las restricciones dadas de la empresa, de otro modo para cumplir normalización se debería crear una entidad por aparte. Es un String de hasta 20 caracteres.

c. nombres: Nombre del empleado, es un string de una cantidad de 30 caracteres.

d. apellido_1: Primer apellido del empleado, es un string de una cantidad de 30 caracteres.

e. apellido_2: Segundo apellido del empleado, es un string de una cantidad de 30 caracteres.

f. fecha_contratacion: Es la fecha en la cual el empleado fue contratado.

g. cargo_id: Identificador del cargo, esto para el sistema de pagos. Es un entero el cual se usa para identificar. Es un entero.

h. departamento_id: Es el identificador del departamento el cual el empleado pertenece, es un entero.

3. Estado:

a. progreso_id: Llave primaria que establece las posibilidades del estado: en proceso, finalizada, sin iniciar o cancelada y asignadas a un entero.

b. progreso_name: Las posibilidades previamente mencionadas.

4. Tarea:
    a. tarea_id: identificador único de cada tarea. Es un entero.
    b. fecha_inicio: Día en el cual da inicio la tarea. Es de tipo fecha.
    c. fecha_fin_estimado: Fecha estimada a terminar la tarea
    d. progreso_actual: Progreso modificable, solo posee 4 valores que se explican en la entidad Estado. Tipo entero.
    e. proyecto_id: proyecto al cual pertenece la tarea. Tipo entero.
    f. empleado_id: empleado a cargo de realizar la tarea. Tipo entero.
5. Pagos:
    a. cargo_id: Llave primaria y se refiere al rol de cada empleado. Es un entero.
    b. cargo_nombre: Nombre del rol. Es un string de 20 caracteres.
    c. cargo_descripcion: descripción de las tareas de ese rol. Es un string de 150 caracteres.
    d. cargo_salario: Salario para cada uno de los roles. Es un numero flotante.
6. Departamento:
    a. departamento_id: Código para cada uno de los departamentos. Es un entero único por departamento.
    b. departamento_name: Nombre del departamento, Es un string de extensión hasta 20 de caracteres.
7. Historial:
    a. ID: Numero incremental, todos los históricos, nunca deben borrarse. Es un entero, que se incrementa cada vez que se agrega.

b. fecha: Fecha en la cual se actualiza. Es de tipo fecha.

c. tarea_id: Identificación de la tarea en la cual se está actualizando. Es de tipo entero.

d. descripción: Descripción de la actualización hecha. Es de string con una cantidad máxima caracteres de 200.

e. tiempo_invertido: Es el tiempo en horas, en las que se tuvo trabajo ese día la tarea.

Explicación de la cardinalidad

1) Lider_cedula [Proyecto] <-> empleado_id [Empleado]: Para cada proyecto debe haber un líder, el cual debe pertenecer a la lista de empleados. 1 a 1.

2) progreso_actual [Proyecto] <-> progreso_id [Estado]: Por medio del código de progreso se define en qué estado se encuentra. N a 1.

3) proyecto_id [Proyecto] <-> proyecto_id [Tarea]: Todas las tareas deben pertenecer a un proyecto, pero muchas tareas pueden poseer al mismo proyecto. 1 a N.

4) empleado_id [Empleado] <-> empleado_id [Tarea]: En cada tarea puede haber 0 o 1 empleado, y un mismo empleado puede realizar distintas tareas. 1 a N.

5) progreso_actual [Empleado] <-> progreso_id [Estado]: Descripción del progreso actual de la tarea según el estado en el cual este se encuentra, según el código proporcionado. N a 1.

6) tarea_id [Tarea] <-> tarea_id [Historial]: Revisión de los históricos hechos por los empleados, para cada una de las tareas. Como no puede haber históricos, como puede haber uno como pueden a haber muchos registros a esa tarea en el historial. 1 a N.

7) cargo_id [Empleado] <-> cargo_id [Pagos]: Relación que describe el tipo de empleado y lo relaciona con la tabla de pagos, para hacer cálculos de planilla. Varios empleados pueden ser del mismo tipo de cargo. N a 1.

8) departamento_id [Empleado] <-> departamento_id [Departamento]: Define a cuál

departamento pertenece a cada, en la cual muchos empleados pueden pertenecer a un mismo departamento, pero cada empleado solo puede pertenecer a un departamento. N a 1.

Reportes

La empresa desea entre sus requisitos generar 3 reportes descritos en el proyecto.

1) Generar reportes que muestren el estado actual de cada proyecto y las tareas asociadas agrupados por su estado: en proceso, finalizado, sin iniciar o cancelado.
    a. El reporte del estado de cada proyecto se puede generar con la cardinalidad de numero 2, y el estado de las tareas asociadas a cada proyecto, se encuentra con la cardinalidad del número 2 y relacionándola con la cardinalidad 5.
2) Visualizar el historial de una tarea en el orden cronológico.
    a. Desde la entidad tarea, se utiliza la cardinalidad 6, y de ahí se pueden ver todos los históricos de esa tarea y ordenar acorde a la fecha.
3) Identificar a los empleados que están asignados a cada tarea y al líder de cada proyecto.
    a. Con las entidades empleados y tarea, se pueden relacionar usando la cardinalidad 4, para validar cada tarea con el empleado a cargo.
    b. De la tabla proyectos, se pueden ver todos los proyectos y por medio de la cardinalidad 1, se obtiene la información de cada líder.

Normalidades

Primera Forma Normal

No hay agrupación de datos similares, se pudo haber incumplido al agrupar todas las tareas de un mismo en la tabla proyecto, o el histórico de las tareas en la tabla de tareas, cosa que no se hizo y se cumple la primera forma normal.

Segunda Forma Normal

Esta es la que se pone en duda, con el tema de código empleado ya que si es única deberá crearse una tabla más, para no dejar que el resto de las variables desentendería de dos posibles llaves primarias. Se supondrá que el código de

trabajo, no es único y si puede utilizar en algún otro trabajador, sea que uno fue despedido y el otro activo, en caso de tener una cantidad de códigos es limitada, se aconseja no manejar dos claves primarias en la misma tabla.

Tercera Forma Normal

La tercera forma normal se cumple, al separar las entidades que dependen únicamente de ellas y no dependen de las situaciones que le ocurran a los empleados, haciendo una correcta separación de estas.

Sentencias

```
-- MySQL Script generated by MySQL Workbench
-- Sat Oct 14 09:13:12 2023
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,N
O_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Pagos`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pagos` (
```

```
  `cargo_id` INT NOT NULL,
  `cargo_nombre` VARCHAR(20) NOT NULL,
  `cargo_descripcion` VARCHAR(150) NOT NULL,
  `cargo_salario` FLOAT NOT NULL,
  PRIMARY KEY (`cargo_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Depatamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Depatamento` (
  `departamento_id` INT NOT NULL,
  `departamento_name` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`departamento_id`),
  UNIQUE INDEX `departamento_name_UNIQUE` (`departamento_name` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Empleado`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Empleado` (
  `empleado_id` INT NOT NULL,
  `Cod_empleado` VARCHAR(20) NOT NULL,
  `nombres` VARCHAR(30) NOT NULL,
  `apellido_1` VARCHAR(30) NOT NULL,
  `apellido_2` VARCHAR(30) NULL,
  `fecha_Contratacion` DATE NOT NULL,
  `cargo_id` INT NOT NULL,
  `departamento_id` INT NOT NULL,
  PRIMARY KEY (`empleado_id`),
  INDEX `cargo_id_idx` (`cargo_id` ASC) VISIBLE,
  INDEX `departamento_id_idx` (`departamento_id` ASC) VISIBLE,
```

```
  CONSTRAINT `cargo_id`
    FOREIGN KEY (`cargo_id`)
    REFERENCES `mydb`.`Pagos` (`cargo_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `departamento_id`
    FOREIGN KEY (`departamento_id`)
    REFERENCES `mydb`.`Depatamento` (`departamento_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Estado`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Estado` (
  `progreso_id` INT NOT NULL,
  `progreso_name` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`progreso_id`),
  UNIQUE INDEX `progreso_name_UNIQUE` (`progreso_name` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Proyecto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Proyecto` (
  `proyecto_id` INT NOT NULL,
  `proyecto_name` VARCHAR(50) NOT NULL,
  `proyecto_descripcio` VARCHAR(200) NOT NULL,
  `fecha_inicio` DATE NOT NULL,
  `fecha_fin_estimado` DATE NOT NULL,
  `progreso_actual` INT NOT NULL,
```

```
  `Lider_cedula` INT NOT NULL,
  PRIMARY KEY (`proyecto_id`),
  INDEX `Liderazgo_idx` (`Lider_cedula` ASC) VISIBLE,
  INDEX `Estado_actual_idx` (`progreso_actual` ASC) VISIBLE,
  CONSTRAINT `Liderazgo`
    FOREIGN KEY (`Lider_cedula`)
    REFERENCES `mydb`.`Empleado` (`empleado_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Estado_actual`
    FOREIGN KEY (`progreso_actual`)
    REFERENCES `mydb`.`Estado` (`progreso_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Tarea`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Tarea` (
  `tarea_id` INT NOT NULL,
  `tarea_descripcion` VARCHAR(20) NOT NULL,
  `fechaDeInicio` DATE NOT NULL,
  `fecha_fin_estimado` DATE NOT NULL,
  `progreso_actual` INT NOT NULL,
  `proyecto_id` INT NOT NULL,
  `empleado_id` INT NOT NULL,
  PRIMARY KEY (`tarea_id`),
  INDEX `proyecto_id_idx` (`proyecto_id` ASC) VISIBLE,
  INDEX `empleado_id_idx` (`empleado_id` ASC) VISIBLE,
  INDEX `progreso_idx` (`progreso_actual` ASC) VISIBLE,
  CONSTRAINT `proyecto_id`
    FOREIGN KEY (`proyecto_id`)
```

```
    REFERENCES `mydb`.`Proyecto` (`proyecto_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `empleado_id`
    FOREIGN KEY (`empleado_id`)
    REFERENCES `mydb`.`Empleado` (`empleado_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `progreso`
    FOREIGN KEY (`progreso_actual`)
    REFERENCES `mydb`.`Estado` (`progreso_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Historial`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Historial` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `fecha` DATE NOT NULL,
  `tarea_id` INT NOT NULL,
  `descripcion` VARCHAR(200) NULL,
  `tiempo_invertido` INT ZEROFILL NULL,
  PRIMARY KEY (`ID`),
  INDEX `tarea_id_idx` (`tarea_id` ASC) VISIBLE,
  CONSTRAINT `tarea_id`
    FOREIGN KEY (`tarea_id`)
    REFERENCES `mydb`.`Tarea` (`tarea_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Conclusiones

El trabajo consistió en una introducción de como las bases de datos, se construyen por los atributos, y según las necesidades se agrupan en entidades de la forma más óptima y correcta para entablar las relaciones entre estos grupos de información llamado entidades o tablas y relacionarlas por el modelo entidad-relación.

El ingeniero no de la forma más óptima, usualmente busca hacer código mientras desarrolla el planteo, de cómo la aplicación trabaje en aplicaciones sencillas es lo más común y no hay mucho problema, pero en lo que es base de datos, es casi obligación de parte de alguien que desea realizar una buena base de datos, debe existir un planteo adecuado, previo a la introducción de la inserción de los conocimientos a la herramienta por lo que debe hacer uso de otras herramientas de diseño, no de scripts los cuales se debe previamente aprobar para luego empezar a insertar dicha información al código como tal.

Parte crucial del diseño de las bases de datos, es definir los constraints y los tipos de cada uno de los atributos, y considerar aquellas necesidades que se ocupen cumplir, como pueden ser los reportes y la velocidad en que estas se generan puede ser más útil un diseño que otro, recordando que las bases de datos son desde Megabytes hasta Gigabytes e inclusive Terabytes y el procesamiento se puede retrasar por la ruta. Un ejemplo de esto, la base de datos actual nuestra es muy rápido en obtener la información de las tareas que pertenecen a un proyecto, pero es muy lenta procesando los históricos de las tareas hechas por cada uno de los empleados en un departamento” X” y haciendo ciertos cambios existe la posibilidad que sea más rápido haciéndolo de otro modo.

Por último, la importancia de la normalidad con el objetivo de eliminar toda redundancia es necesaria para cumplir lo mínimo esperado para una base de datos, y la importancia de la cardinalidad indica las relación X a X entre cada uno de las entidades y los atributos comunes como inferirlos.

Bibliografía

Lucidchart (2020). Qué es un diagrama entidad-relación https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

Londoño, P. (2023). Qué es MySQL, para qué sirve y características principales. Extraido de: https://blog.hubspot.es/website/que-es-mysql

Oracle (2023). Que es una base de datos. Extraido de: (https://www.oracle.com/mx/database/what-is-database/

Silberchatz. A. (2014). Fundamentos de Bases de datos. 6° Edición. McGraw-Hill. Madrid, España.