# UNIVERSIDAD ESTATAL A DISTANCIA
# DIRECCIÓN DE EXTENSIÓN UNIVERSITARIA
# ÁREA DE COMUNICACIÓN Y TECNOLOGÍA
# CURSOS EN LÍNEA

Curso:
Bases de Datos - 00826

Tema:
**Proyecto Programado 2**
**Creación de Sentencias SQL en MySQL server**

Estudiante:
Carlos Garita Campos

Profesor:
Luis Vindas Espinoza

Cuatrimestre III
29 de octubre del 2019

## Tabla de contenido

## Introducción

La estructuración de las bases de datos y el establecimiento del modelo conceptual y físico son de las etapas más importantes en el diseño de los sistemas de almacenamiento, a fin de garantizar el correcto funcionamiento e integridad de la información, de forma tal que pueda ser consultada en el momento que se requiera.

La normalización de la base de datos y el correcto estudio del proceso para el cual se está diseñando la misma, ayudan a disminuir las redundancias de información, los tiempos de consulta, la cantidad de espacio requerido para almacenamiento por parte de la base de datos, así como los errores de almacenamiento y consulta, por lo cual es necesario que el desarrollador de la base de datos analice correctamente ambos aspectos durante la etapa del modelo conceptual.

La manipulación y la consulta de bases de datos son procesos de mucho cuidado, que deben ser realizados por profesionales en el tema. Para lo anterior, existen múltiples herramientas tanto para la manipulación (como el TRUNCATE, DELETE, INSERT, etc.), como para la consulta de la información (como el SELECT, INNER JOIN, LEFT JOIN, ORDER BY, GROUP BY, etc), por lo cual, es importante conocer los detalles de funcionamiento de cada una para su correcta aplicación.

El presente documento plantea el modelo conceptual y físico de una base de datos relacional para el call center de la compañía Contacta2, así como también, los Scripts para la eliminación e inserción de registros en las tablas de la base de datos, y dos consultas para obtener información específica de los registros almacenados.

## Desarrollo

### I Parte. Descripción del problema

La empresa Contacta2 está implementando un Call Center y necesita un diseño de base de datos para satisfacer sus necesidades del negocio, que básicamente se resume en las siguientes fuentes de datos: casos, tipos de casos, sub-casos, estados y agentes.

Los requerimientos básicos de la base de datos son los siguientes:

1) El call center atiende llamadas, las cuales a su vez se les identifica como "casos". Nos interesa mantener todo lo relacionado a cada caso como por ejemplo: código de caso, la fecha de creación del caso, el agente que lo tiene asignado, el tipo de caso, la descripción general del caso, su estado respectivo y la fecha de cuando se cerró el caso.
2) Cada caso puede generar a su vez "sub-casos", si aplica, nos interesa saber cuántos sub casos posee un "Caso". Con respecto a la información de los sub-casos nos interesa saber lo siguiente: la descripción del sub-caso, su estado y la fecha de cuando se completó el sub caso.
3) Lista de posibles "estados" de cada caso y sub-caso, estos posibles estados pueden ser: pendiente, en progreso, en pausa, completado y cancelado.
4) Cada caso se clasifica en "Tipos de Casos" para una mayor facilidad de agrupar casos según la solicitud del cliente, estos posibles tipos de casos pueden ser los siguientes: Cancelación de cuenta, creación de cuenta, y modificación de cuenta.
5) También nos interesa llevar un maestro de "agentes", y tener a mano la información más importante de todos los agentes como por ejemplo: código de agente, nombre, usuario de red, extensión, estado (activo/inactivo), fecha de nacimiento, salario y puesto.

**II Parte. Diseño del diagrama relacional del modelo.**

De acuerdo con lo expuesto por la descripción del problema, la base de datos requerida constaría como mínimo de 3 entidades que se llamarían:

- Casos
- SubCasos
- Agentes

No obstante, analizando más a profundidad la forma en la que operaría el call center según se describe en la sección anterior y los atributos que poseería cada una de las entidades, sería importante hacer una entidad adicional que sea la que maneje la información de los puestos y salarios correspondientes a cada puesto, trabajando bajo el supuesto de que solo existe un salario para cada puesto, la cual puede ser la forma de trabajo de muchas empresas del sector privado. Esto ayudaría a reducir el consumo de espacio en la base de datos, debido a que se reduciría la cantidad de datos repetidos de tipo varchar.

De la misma forma que para el caso anterior, se crean entidades adicionales que manejen la información de los estados de los casos, tipos de caso y estados de los agentes, por lo cual el total de entidades (7 entidades) quedaría de la siguiente forma:

- Casos
- Estado_Casos
- Tipo_Casos
- SubCasos
- Agentes
- Estado_Agentes
- Puesto_Agentes

El diagrama relacional del modelo de la base de datos sería de la siguiente forma:

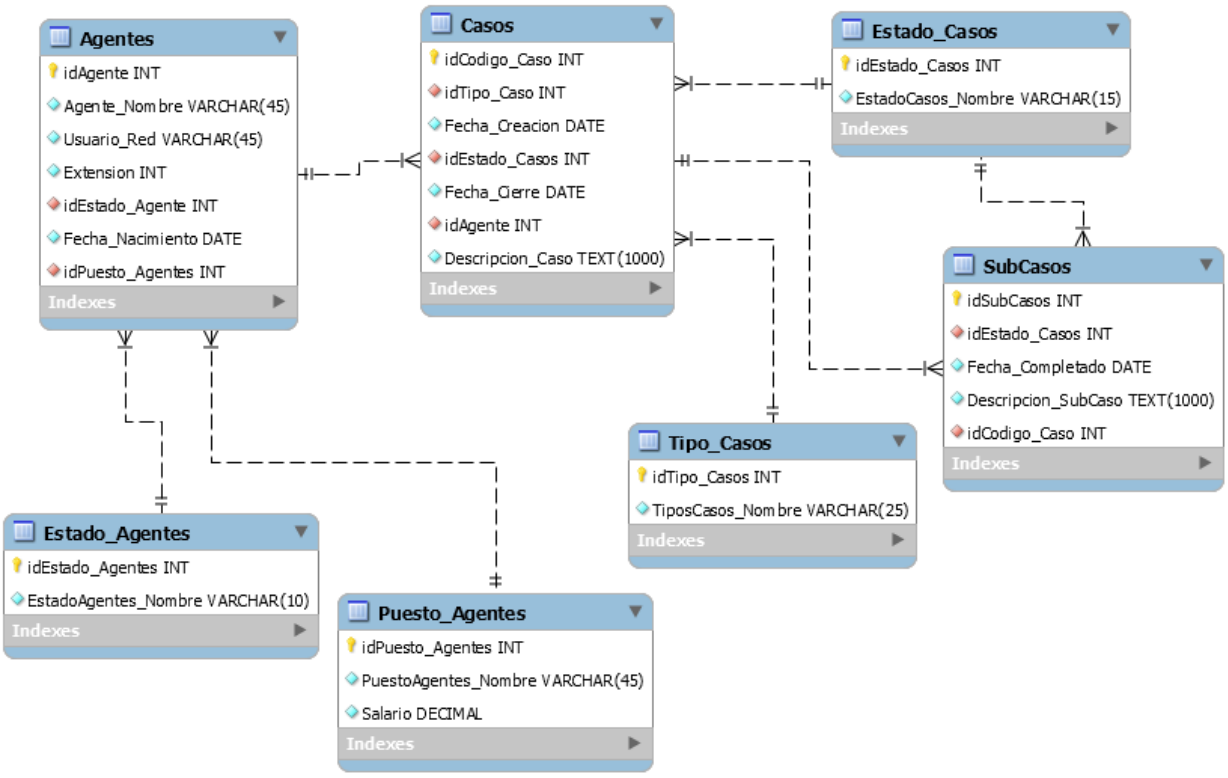


**Figura 1.** Diagrama relacional del modelo de la base de datos con 7 entidades en total realizado en MySQL.

**III Parte. Script de creación de tablas.**

Del diagrama anterior, se tendría que el Script correspondiente en MySQL sería:

-- MySQL Script generated by MySQL Workbench

-- Sun Oct  6 16:45:41 2019

-- 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,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema Call_Center

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema Call_Center

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `Call_Center` DEFAULT CHARACTER SET utf8 ;

USE `Call_Center` ;

-- -----------------------------------------------------

-- Table `Call_Center`.`Estado_Casos`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Call_Center`.`Estado_Casos` (

  `idEstado_Casos` INT NOT NULL AUTO_INCREMENT,

  `EstadoCasos_Nombre` VARCHAR(15) NOT NULL,
```

```
  PRIMARY KEY (`idEstado_Casos`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Call_Center`.`Estado_Agentes`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Call_Center`.`Estado_Agentes` (

  `idEstado_Agentes` INT NOT NULL AUTO_INCREMENT,

  `EstadoAgentes_Nombre` VARCHAR(10) NOT NULL,

  PRIMARY KEY (`idEstado_Agentes`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Call_Center`.`Puesto_Agentes`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Call_Center`.`Puesto_Agentes` (

  `idPuesto_Agentes` INT NOT NULL AUTO_INCREMENT,

  `PuestoAgentes_Nombre` VARCHAR(45) NOT NULL,

  `Salario` DECIMAL NOT NULL,

  PRIMARY KEY (`idPuesto_Agentes`))

ENGINE = InnoDB;

-- -----------------------------------------------------
```

```
-- Table `Call_Center`.`Agentes`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Call_Center`.`Agentes` (

  `idAgente` INT NOT NULL AUTO_INCREMENT,

  `Agente_Nombre` VARCHAR(45) NOT NULL,

  `Usuario_Red` VARCHAR(45) NOT NULL,

  `Extension` INT NOT NULL,

  `idEstado_Agente` INT NOT NULL,

  `Fecha_Nacimiento` DATE NOT NULL,

  `idPuesto_Agentes` INT NOT NULL,

  PRIMARY KEY (`idAgente`),

  UNIQUE INDEX `Extension_UNIQUE` (`Extension` ASC) VISIBLE,

  UNIQUE INDEX `Usuario_Red_UNIQUE` (`Usuario_Red` ASC) VISIBLE,

  INDEX `Estado_Agentes_idx` (`idEstado_Agente` ASC) VISIBLE,

  INDEX `Puesto_Agentes_idx` (`idPuesto_Agentes` ASC) VISIBLE,

  CONSTRAINT `Estado_Agentes`

    FOREIGN KEY (`idEstado_Agente`)

    REFERENCES `Call_Center`.`Estado_Agentes` (`idEstado_Agentes`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `Puesto_Agentes`
    FOREIGN KEY (`idPuesto_Agentes`)
    REFERENCES `Call_Center`.`Puesto_Agentes` (`idPuesto_Agentes`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Call_Center`.`Tipo_Casos`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Call_Center`.`Tipo_Casos` (
  `idTipo_Casos` INT NOT NULL AUTO_INCREMENT,
  `TiposCasos_Nombre` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`idTipo_Casos`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Call_Center`.`Casos`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Call_Center`.`Casos` (
  `idCodigo_Caso` INT NOT NULL AUTO_INCREMENT,
  `idTipo_Caso` INT NOT NULL,
```

```
`Fecha_Creacion` DATE NOT NULL,

`idEstado_Casos` INT NOT NULL,

`Fecha_Cierre` DATE NOT NULL,

`idAgente` INT NOT NULL,

`Descripcion_Caso` TEXT(1000) NOT NULL,

PRIMARY KEY (`idCodigo_Caso`),

INDEX `Estado_Casos_idx` (`idEstado_Casos` ASC) VISIBLE,

INDEX `Agentes_idx` (`idAgente` ASC) VISIBLE,

INDEX `Tipo_Casos_idx` (`idTipo_Caso` ASC) VISIBLE,

CONSTRAINT `Estado_Casos`

  FOREIGN KEY (`idEstado_Casos`)

  REFERENCES `Call_Center`.`Estado_Casos` (`idEstado_Casos`)

  ON DELETE NO ACTION

  ON UPDATE NO ACTION,

CONSTRAINT `Agentes`

  FOREIGN KEY (`idAgente`)

  REFERENCES `Call_Center`.`Agentes` (`idAgente`)

  ON DELETE NO ACTION

  ON UPDATE NO ACTION,

CONSTRAINT `Tipo_Casos`
```

```
    FOREIGN KEY (`idTipo_Caso`)
    REFERENCES `Call_Center`.`Tipo_Casos` (`idTipo_Casos`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Call_Center`.`SubCasos`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Call_Center`.`SubCasos` (
  `idSubCasos` INT NOT NULL AUTO_INCREMENT,
  `idEstado_Casos` INT NOT NULL,
  `Fecha_Completado` DATE NOT NULL,
  `Descripcion_SubCaso` TEXT(1000) NOT NULL,
  `idCodigo_Caso` INT NOT NULL,
  PRIMARY KEY (`idSubCasos`),
  INDEX `Estado_SubCasos_idx` (`idEstado_Casos` ASC) VISIBLE,
  INDEX `Codigo_Caso_idx` (`idCodigo_Caso` ASC) VISIBLE,
  CONSTRAINT `Estado_SubCasos`
    FOREIGN KEY (`idEstado_Casos`)
    REFERENCES `Call_Center`.`Estado_Casos` (`idEstado_Casos`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Codigo_Caso`
    FOREIGN KEY (`idCodigo_Caso`)
    REFERENCES `Call_Center`.`Casos` (`idCodigo_Caso`)
    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;
```

**IV Parte. Script para borrado de todos los registros de cada tabla.**

Para la eliminación de todos los registros contenidos en cada tabla, se tendrían los siguientes comandos en MySQL (se elige TRUNCATE en lugar de DELETE debido a que de esta forma se eliminan todos los registros de cada tabla de una sola vez y se reinicia la numeración de cada PRIMARY_KEY de cada tabla):

```
SET FOREIGN_KEY_CHECKS = 0;
TRUNCATE casos;
SET FOREIGN_KEY_CHECKS = 1;
```

```
SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE subcasos;

SET FOREIGN_KEY_CHECKS = 1;


SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE agentes;

SET FOREIGN_KEY_CHECKS = 1;


SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE estado_agentes;

SET FOREIGN_KEY_CHECKS = 1;


SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE estado_casos;

SET FOREIGN_KEY_CHECKS = 1;


SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE puesto_agentes;

SET FOREIGN_KEY_CHECKS = 1;
```

```
SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE tipo_casos;

SET FOREIGN_KEY_CHECKS = 1;
```

**V Parte. Script para inserción de nuevos registros en cada tabla.**

Para la inserción de nuevos registros en cada tabla, se tendrían los siguientes comandos en MySQL:

```
INSERT INTO estado_agentes (idEstado_Agentes, EstadoAgentes_Nombre) VALUES

(1,'Activo'),

(2,'Inactivo');

INSERT INTO estado_casos (idEstado_Casos, EstadoCasos_Nombre) VALUES

(1,'Pendiente'),

(2,'En Progreso'),

(3,'En Pausa'),

(4,'Completado'),

(5,'Cancelado');

INSERT INTO puesto_agentes (idPuesto_Agentes, PuestoAgentes_Nombre, Salario) VALUES

(1,'P1', 200000),
```

(2,'P2', 300000),

(3,'P3', 400000),

(4,'P4', 500000),

(5,'P5', 600000);

INSERT INTO tipo_casos (idTipo_Casos, TiposCasos_Nombre) VALUES

(1,'Cancelación de cuenta'),

(2,'Creación de cuenta'),

(3,'Modificación de cuenta');

INSERT INTO agentes (idAgente, Agente_Nombre, Usuario_Red, Extension, idEstado_Agente, Fecha_Nacimiento, idPuesto_Agentes) VALUES

(1,'Carlos','cgc123',1327,1,'1985-01-02',3),

(2,'María','mna123',1329,1,'1990-06-20',4);

INSERT INTO casos (idCodigo_Caso, idTipo_Caso, Fecha_Creacion, idEstado_Casos, Fecha_Cierre, idAgente, Descripcion_Caso) VALUES

(1,2,'2019-01-02',5,'2019-01-20',1,'Queja'),

(2,3,'2019-03-19',1,'2019-03-30',1,'Consulta'),

(3,1,'2019-05-15',3,'2019-05-22',2,'Consulta');

INSERT INTO subcasos (idSubCasos, idEstado_Casos, Fecha_Completado, Descripcion_SubCaso, idCodigo_Caso) VALUES

(1,1,'2019-10-27', 'Queja',3),

(2,1,'2019-10-26', 'Consulta',3),

(3,3,'2019-10-25', 'Queja',2),

(4,5,'2019-10-24', 'Queja',1),

(5,4,'2019-10-23', 'Consulta',1),

(6,1,'2019-10-22', 'Queja',1),

(7,2,'2019-10-21', 'Consulta',1);

**VI Parte. Script de consulta para obtener todos los casos, ordenados por la descripción del estado del caso.**

Para obtener todos los casos ordenados por la descripción del estado del caso, se utilizan los siguientes comandos en MySQL:

```
SELECT
    idCodigo_Caso AS 'Código Caso',
    Fecha_Creacion AS 'Fecha Creación de Caso',
    casos.idAgente AS 'Código Agente',
    Agente_Nombre AS 'Nombre de Agente',
    Descripcion_Caso AS 'Descripción General de Caso',
```

```
    estado_casos.idEstado_Casos AS 'Código Estado de Caso',
    EstadoCasos_Nombre AS 'Descripción Estado de Caso',
    casos.idTipo_Caso AS 'Código Tipo de Caso',
    TiposCasos_Nombre AS 'Descripción Tipo de Caso',
    Fecha_Cierre AS 'Fecha Cierre de Caso'
FROM casos
INNER JOIN agentes ON casos.idAgente = agentes.idAgente
INNER JOIN estado_casos ON casos.idEstado_Casos = estado_casos.idEstado_Casos
INNER JOIN tipo_casos ON casos.idTipo_Caso = tipo_casos.idTipo_Casos
order by estado_casos.idEstado_Casos asc;
```

**VII Parte. Script de consulta para obtener todos los casos ordenados por la descripción del tipo de caso, y que cuenta la cantidad de sub-casos que cada caso posee.**

Para obtener todos los casos ordenados por la descripción del tipo de caso, y que se cuente la cantidad de sub-casos que cada caso posee, se utilizan los siguientes comandos en MySQL:

```
SELECT
    casos.idTipo_Caso AS 'Código Tipo de Caso',
    TiposCasos_Nombre AS 'Descripción Tipo de Caso',
    casos.idCodigo_Caso AS 'Código Caso',
    Descripcion_Caso AS 'Descripción Caso',
```

```
    estado_casos.idEstado_Casos AS 'Código Estado de Caso',
    EstadoCasos_Nombre AS 'Descripción Estado de Caso',
   (SELECT count(idSubCasos) FROM subcasos
    where casos.idCodigo_Caso = subcasos.idCodigo_Caso) AS 'Cantidad Subcasos'
FROM casos
INNER JOIN tipo_casos ON casos.idTipo_Caso = tipo_casos.idTipo_Casos
INNER JOIN subcasos ON casos.idCodigo_Caso = subcasos.idCodigo_Caso
INNER JOIN estado_casos ON casos.idEstado_Casos = estado_casos.idEstado_Casos
group by tipo_casos.TiposCasos_Nombre order by tipo_casos.TiposCasos_Nombre asc;
```

## Conclusiones

1. Las entidades principales de la base de datos son 3 (Casos, SubCasos y Agentes), sin embargo, es importante contar con 7 entidades en total como mínimo, a fin de disminuir el uso del almacenamiento al disminuir la repetición de datos tipo varchar o text.
2. El estudio de la forma en la que se lleva a cabo el proceso para el cual se está diseñando la base de datos, así como la respectiva normalización de la información, es crucial, para el buen funcionamiento del sistema de almacenamiento, entre otros factores importantes.
3. Para la eliminación de todos los registros de una tabla de una sola vez, es conveniente utilizar TRUNCATE, ya que elimina todos los registros sin importar la cantidad y reinicia el conteo de la PRIMARY_KEY cuanto se tiene programada para autoincremento.
4. Para la obtención del conteo de Subcasos por cada caso, es necesaria la utilización de un SELECT anidado que realice el conteo de Subcasos para cada caso y muestre el resultado como una columna adicional del SELECT principal.

5. El INNER JOIN permite obtener registros de diferentes tablas en una sola consulta, cuando se tienen columnas con registros coincidentes entre tablas.

## Recomendaciones

1. Utilizar TRUNCATE en lugar de DELETE cuando se quiere eliminar todos los registros de una tabla, ya que permite eliminar todos los registros en un solo comando sin importar la cantidad de registros que tenga la tabla.
2. Utilizar cuando se quieren obtener datos de varias tablas que tienen columnas con registros coincidentes entre tablas.
3. Utilizar el comando “AS” para renombrar las columnas en las consultas, a fin de que sea más fácil de leer para el usuario de la consulta.

## Bibliografía

1. Silberschatz A., Korth H., Sudarshan S. (2014). Fundamentos de base de datos. Editorial McGraw Hill, 6a edición, México.