IES Gran Capitán

Módulo: Base de Datos

Ciclo Formativo de Grado Superior “Desarrollo de aplicaciones Web”

Cuaderno de Prácticas

Fecha entrega: 10/06/15

Autor: Pedro J. Ramos

# Calendario de trabajo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Práctica | 1 | 1 | Hora | 02-jun |
| Práctica | 2 | 1 | Hora | 03-jun |
| Práctica | 3 | 1 | Hora | 03-jun |
| Práctica | 4 | 4 | Horas | 08-jun |
| Práctica | 5 | 1 | Hora | 05-jun |
| Práctica | 6 | 2 | Horas | 05-jun |
| Práctica | 7 | 4 | Horas | 07-jun |
| Práctica | 8 | 4 | Horas | 06-jun |
| Práctica | 9 | 4 | Horas | 04-jun |
| Práctica | 10 | 3 | Horas | 09-jun |

# Practica 1

1. **Indica las diferencias existentes entre las funciones de manipulación y de descripción.**

En la función de descripción el diseñador especifica los elementos de datos que integran la base de datos, su estructura y las relaciones que existen entre ellos, las reglas de integridad semántica, etc. Además de las características de tipo físico y las vistas lógicas de los usuarios. Ésta función se lleva a cabo mediante el Lenguaje de Descripción/Definición de Datos (LDD) propio del SGBD. Durante la función de manipulación los usuarios pueden recuperar la información, es decir, consultar la base de datos, o bien actualizarla porque se han producido cambios en la base de datos. Para ello es preciso cargar los datos en las estructuras creadas durante la función de descripción para poder disponer de la base de datos completamente. Se realiza mediante el Lenguaje de Manipulación de Datos (LMD).

1. **¿Qué tipos de usuarios interaccionan con una base de datos?**

Usuarios normales, usuarios programadores y usuarios administradores.

1. **Indica que es un lenguaje huésped y un lenguaje anfitrión.**

El lenguaje huésped es el LMD de bajo nivel, cuya función es el de la manipulación física de los datos. Llamado así porque suele estar alojado en algún otro lenguaje de programación de propósito general. El lenguaje anfitrión es un lenguaje principal a partir del cual se desarrolla la actividad necesaria con la base de datos. Es independiente.

1. **La gestión del espacio de almacenamiento, ¿a qué nivel de la arquitectura ANSI/SPARC pertenece?**

Nivel Interno.

1. **Dibuja un diagrama de la arquitectura de sistemas de bases de datos (ANSI/SPARC).**

****

1. **Indica las principales funciones realizadas por el SGDB.**

El SGBD debe cumplir las siguientes funciones: Función de descripción o de definición. Debe permitir al administrador de la BD especificar los elementos de datos que la integran, su estructura, las relaciones que existen entre ellos, las reglas de integridad semántica, los controles a efectuar antes de autorizar el acceso a la BD, etc. Se lleva a cabo mediante el Lenguaje de Definición de Datos y debe suministrar los medios para definir las tres estructuras de datos, especificando las características de los datos a cada uno de estos niveles. Función de manipulación. Permite a los usuarios de la BD manipular los datos de ésta, siguiendo las especificaciones pertinentes y las normas de seguridad que el DBA haya establecido. Se lleva a cabo mediante el Lenguaje de Manipulación de Datos. Función de utilización. Comprende todas las interfaces que los diferentes usuarios requieran para interactuar con la BD y proporciona un conjunto de procedimientos para el administrador, como el Lenguaje de Control de Datos, funciones de servicio y otras relacionadas con la seguridad física y protección frente accesos no autorizados.

1. **Explica la diferencia entre la independencia física y lógica de los datos.**

La independencia lógica: es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación.

La independencia física: es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual

1. **¿Qué es el diccionario de datos?**

Un diccionario de datos contiene las características lógicas de los datos que se van a utilizar en el sistema que estamos programando, incluyendo nombre, descripción, alias, contenido y organización. Estos diccionarios se desarrollan durante el análisis de flujo de datos y ayuda a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño del proyecto.

1. **Diferencias entre el LDD y LMD de un sistema gestor de base de datos.**

El LDD, Lenguaje de Definición de Datos, tiene la función de describir y definir todos los esquemas que participen en la BD, es decir, la descripción de los objetos que vamos a representar. En cambio, el LMD, Lenguaje de Manipulación de Datos, permite a los usuarios acceder o manipular los datos.

1. **Indica los componentes principales de un sistema gestor de base de datos.**

Control de autorización. Comprueba que el usuario tiene los permisos necesarios para llevar a cabo la operación que solicita. Procesador de comandos. Control de la integridad. Comprueba que la operación a realizar satisface todas las restricciones de integridad necesarias. Optimizador de consultas. Determina la estrategia óptima para la ejecución de las consultas. Gestos de transacciones. Realiza el procesamiento de las transacciones. Planificador. Asegura que las operaciones que se realizan concurrentemente sobre la BD tienen lugar sin conflictos. Gestos de buffers. Responsable de transferir los datos entre la memoria principal y los dispositivos de almacenamiento secundario.

1. **¿Qué es un modelo de datos?**

Un modelo de datos es un lenguaje orientado a describir una Base de Datos. Típicamente un modelo de datos permite describir:

* Las estructuras de datos de la base: El tipo de los datos que hay en la base y la forma en que se relacionan.
* Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
* Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

Otro enfoque es pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí.

1. **¿Qué son los lenguajes de cuarta generación? Pon ejemplos.**

Son lenguajes de programación de alto nivel, por lo que se alejan del idioma de la máquina y se acercan al del lenguaje humano más similar al idioma inglés que a un lenguaje de tercera generación (como Java o C++). En estos lenguajes en lugar de describir cómo deben obtenerse los resultados, se especifica el resultado a obtener. Ejemplos de estos lenguajes son los lenguajes de consulta de base de datos SQL, PL/SQL o el lenguaje NATURAL.

1. **Indica las principales ventajas de un sistema de bases de datos. ¿Existen algunas desventajas?**

Ventajas:

* Gestión y control centralizado de los datos.
* Reducción de redundancias.
* Compartición de datos.
* Integridad.
* Seguridad.
* Control de concurrencia.
* Independencia de datos, permitiendo cambios y crecimiento.

Desventajas:

* Coste de software y hardware: Además del coste de comprar el software, probablemente haya que ampliar o subir de versión el hardware (upgrade) para que el programa pueda almacenarse y funcione correctamente; además, si no, el sistema podría degradarse y los tiempos de respuesta subir drástica-mente por la sobrecarga que añade el SGBD al tener que implementar la seguridad, integridad y datos compartidos. Recordemos que estamos hablando de SGBD grandes, no por supuesto de los de ordenadores personales.
* Coste de migración de aplicaciones: Un coste adicional, también, es la migración de las aplicaciones para pasar a un entorno integrado.
* Las operaciones de respaldo y recuperación de los datos en caso de fallo son complejas, debido al acceso concurrente de múltiples usuarios.
* Criticidad del SGBD: La centralización también significa que los datos de la organización, están almacenados en un único lugar, en la base de datos, con lo que cualquier fallo de seguridad u operación del SGBD puede afectar a toda la organización (sistemas distribuidos y compartidos).

1. **¿Puede una clave foránea almacenar nulos en todos sus componentes?**

Si mientras esa clave foránea no forme parte de la clave principal de la tabla.

1. **¿Puede un mismo atributo ser clave ajena y primaria de una misma relación? Si la respuesta fuese afirmativa pon un ejemplo, y si fuese negativa explica las razones.**

Si en el caso de las relaciones recursivas donde una misma entidad se relaciona consigo misma por ejemplo Trabajador que es un jefe de un trabajador.

1. **¿Es cierto que toda relación con dos atributos está siempre en segunda forma normal?, ¿y en tercera?**

Si ya que ese atributo depende funcionalmente del otro que es clave principal y no puede haber transitividad (tercera) ya que no hay un tercer atributo.

1. **¿Puede estar una clave ajena y su correspondiente clave primaria definida sobre dominios diferentes? ¿En qué condiciones?**

Todos los atributos deben pertenecer al mismo dominio ya que si no violaría la regla de integridad referencial.

1. **¿Es cierto que en la transformación del esquema conceptual al relacional todo tipo de interrelación N:M se traduce en el fenómeno de propagación de clave? Justifica la respuesta.**

No, Se crea una tabla que tiene por clave primaria a las claves primarias de las tablas que relaciona. Si tiene atributos propios, habrá que observar si hay que “ampliar la clave”.

1. **¿Es cierto que en el modelo relacional los dominios representan propiedades de la relación? Justifica la respuesta.**

Los dominios representan valores posibles que puede tomar un atributo.

1. **Haz un esquema detallado del proceso de diseño de un sistema de información utilizando bases de datos.**

Problema mundo real 🡪 Análisis de Requisitos 🡪 Diseño Conceptual 🡪 Diseño Lógico 🡪 Diseño Físico

# practica 2

Requisitos:

• No utilizar herramientas de modelado.

Para el supuesto propuesto realiza:

• Diseño conceptual.

• Diseño lógico.

Supuesto.

La Consejería de Cultura de la Comunidad Gallega ha decidido guardar información referente al Camino de Santiago en una Base de Datos. La información que desea almacenar es la siguiente:

Teniendo en cuenta que la peregrinación a Santiago se puede realizar por distintos caminos (el camino francés, el aragonés, etc.), se quiere guardar información acerca de éstos. Esta información se refiere al nombre (que los identifica), el número de kilómetros totales y el tiempo estimado para la realización del camino.

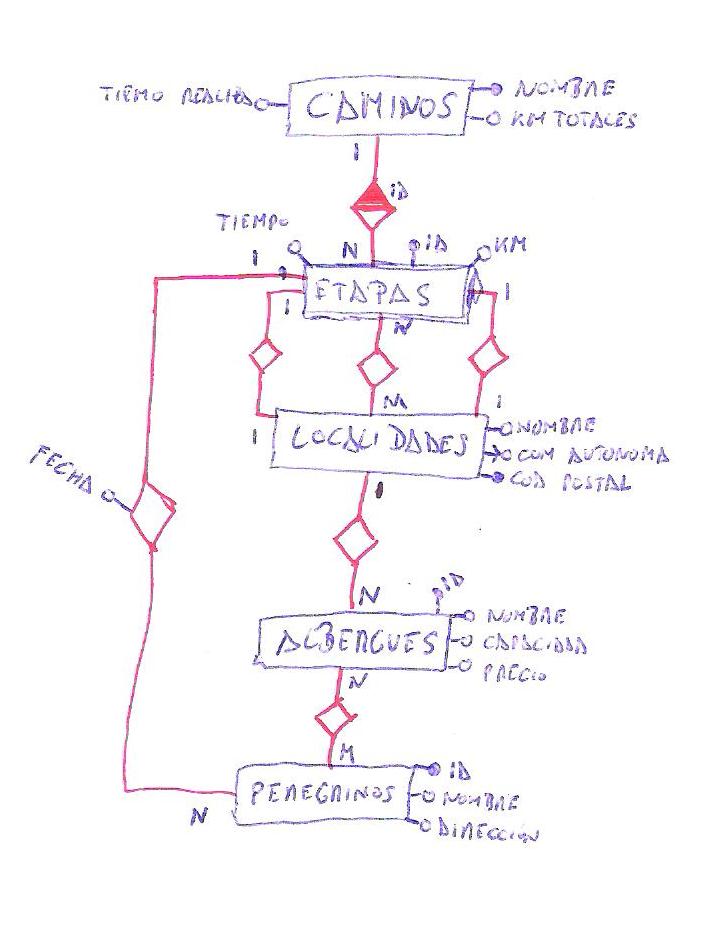
Cada camino se compone de distintas etapas que se identifican por un número correlativo dentro de cada camino, y para cada una de ellas se desea saber el número de kilómetros, el tiempo estimado y las distintas localidades por las que pasa. Además se quiere recoger la localidad de salida y de llegada de la correspondiente etapa.

Se recogerán las distintas localidades por la que pasa cada camino. La información que se recogerá de cada localidad será: nombre de la misma, Comunidad Autónoma a la que pertenece y código postal. Se debe tener en cuenta que pueden existir localidades comunes a distintos caminos.

Se desea guardar información sobre los albergues para peregrinos que existen en algunas de las localidades que pertenecen al camino. Esta información consta de: nombre de albergue, capacidad y precio (si lo tuvieran)

Por último, se quiere registrar los peregrinos que realizan el camino. Para llevar este control cada uno de ellos lleva un carnet que consta de un número de identificación, el nombre completo del peregrino, su dirección y las localidades por las que ha ido pasando a lo largo del recorrido junto con el día que llegaron a dicha localidad.

MODELO ENTIDAD-INTERELACIÓN



MODELO RELACIONAL (**PK-Negrita,** *FK-Cursiva)*

Caminos (**Nombre,** kmTotales, TiempoRealización

Etapas (**ID,** KMTotales, TiempoRealización, ***NombreCamino,*** *CodPostalLocalidadInicio, CodPostalLocalidadFinal****)***

Localidades (**CodPostal,** Nombre, ComunidadAutónoma

Albergues (**ID,** Nombre, Capacidad, Precio, *CodPostalLocalidad)*

Peregrinos (**ID,** nombre, dirección, *IDEtapas, FechaPaso)*

Etapas-Localidades (**IDEtapa, CodPostalLocalidad)**

Albergues-Peregrinos (**IDAlbergue, IDPeregrino)**

# PRACTICA 3

Requisitos:

• No utilizar herramientas de modelado.

Para el supuesto propuesto realiza:

• Diseño conceptual.

• Diseño lógico.

Supuesto

Se pretende llevar a cabo un control sobre la energía eléctrica que se produce y consume en un determinado país. Se parte de las siguientes hipótesis.

Existen productores básicos de electricidad que se identifican por un nombre, de los cuales interesa su producción media, producción máxima y fecha de entrada en funcionamiento. Estos productores básicos pertenecen a algunas de las siguientes categorías: Central Hidroeléctrica, Central Solar, Central Nuclear o Central Térmica. De una central hidroeléctrica o presa interesa saber su ocupación, capacidad máxima y número de turbinas. De una central solar interesa saber la superficie total de paneles solares, la media anual de horas de sol y el tipo (fotovoltaica o termodinámica). De una central nuclear, interesa saber el número de reactores que posee, el volumen de plutonio consumido y el de residuos nucleares que produce. De una central térmica, interesa saber el número de hornos que posee, el volumen de carbón consumido y el volumen de su emisión de gases.

Por motivos de seguridad nacional interesa controlar el plutonio de que se provee una central nuclear. Este control se refiere a la cantidad de plutonio que compra a cada uno de sus posibles suministradores (nombre y país) y que porta un determinado transportista (nombre y matrícula). Ha de tenerse en cuenta que un mismo suministrador puede vender plutonio a distintas centrales nucleares y que cada porte (un único porte por compra) puede realizarlo un transportista diferente. Cada día, los productores entregan la energía producida a una o varias estaciones primarias, las cuales pueden recibir diariamente una cantidad distinta de energía de cada uno de esos productores. Los productores entregan siempre el total de su producción. Las estaciones primarias se identifican por su nombre y tienen un número de transformadores de baja a alta tensión y son cabecera de una o varias redes de distribución.

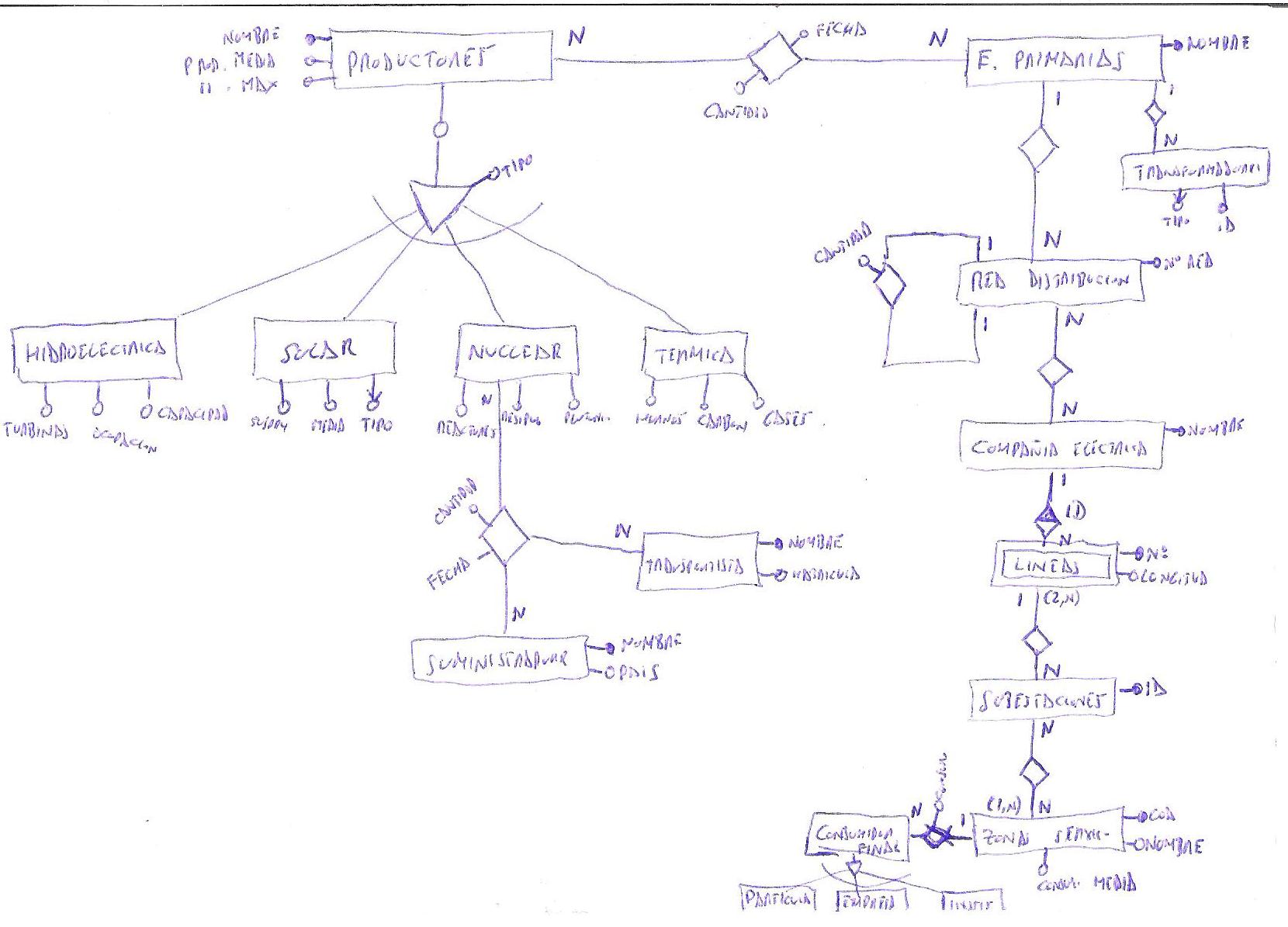
Una red de distribución se identifica por un número de red y sólo puede tener una estación primaria como cabecera. La propiedad de una red puede ser compartida por varias compañías eléctricas. A cada compañía eléctrica se le identifica por su nombre.

La energía sobrante en una de las redes puede enviarse a otra red. Se registra el volumen total de energía intercambiada entre dos redes.

Una red está compuesta por una seria de líneas, cada línea se identifica por un número secuencial dentro del número de red y tiene una determinada longitud. La menor de las líneas posibles abastecerá al menos a dos subestaciones. Una subestación es abastecida sólo por una línea y distribuye a una o varias zonas de servicio. A estos efectos, las provincias (código y nombre), se encuentran divididas en tales zonas de servicio, aunque no puede haber zonas de servicio que pertenezcan a más de una provincia. Cada zona de servicio puede ser atendida por más de una subestación.

En cada zona de servicio se desea registrar el consumo medio y el número de consumidores finales de cada una de las siguientes categorías: particulares, empresas e instituciones.

MODELO ENTIDAD-INTERRELACION



MODELO RELACIONAL (**PK-Negrita,** *FK-Cursiva)*

Hidroeléctrica (***Nombre,*** Turbinas, Ocupación, Capacidad)

Solar (***Nombre,*** Superficie, Media, Tipo)

Nuclear (***Nombre,*** Reactores, Residuos, Plutonio)

Térmica (***Nombre,*** Horno, Carbón, Gases)

Transportista (**Nombre,** Matrícula)

Suministrador (**Nombre,** País)

Productores (**Nombre,** ProducMedia, ProducMax, Tipo)

E. Primarias (**Nombre)**

Transformadores (**ID,** Tipo, *NombreEPRimaria)*

Red Distribución (**NºRed,** *NºRedCede)*

Compañía eléctrica (**Nombre)**

Líneas (**Nº,** Longitud, ***NombreCompañiaEléctrica***)

Subestaciones (**ID,** *NºLinea, NombreCompañiaElectrica)*

ZonasdeServicio (**Cod,** nombre, consumo medio)

ConsumidorFinal (**Nombre,** *CodZOnaServicio)*

Nuclear-Transportista-Suministrador(***NombreProductor, NombreTrasnsportista, NombreSuministrador,*** cantidad, fecha)

Productores-EPrimarias (***NombreProductor, NombreEPrimaria,*** Fecha, Cantidad)

RedDistribución-CompañiaElectrica (***NºRed, NombreCompañiaElectrica***)

Subestaciones-ZonasServicio (***IDSubEstación, CodZonaServicio***)

# Practica 4

Requisitos:

• No utilizar herramientas de modelado.

• Consultar datos en:

http://es.wikipedia.org/wiki/Anexo:Campeones\_de\_torneos\_de\_Grand\_Slam\_

%28individual\_masculino%29#Campeones\_por\_a.C3.B1o

Para el supuesto propuesto realiza:

• Diseño conceptual.

• Diseño lógico.

• Script SQL de creación del esquema.

• Script SQL de introducción de datos.

Supuesto

La Federación Internacional de Tenis nos contrata para diseñar una base de datos para dar soporte a los torneos de Grand Slam.

El sistema debe memorizar todos los encuentros que se han desarrollado desde que existe el torneo, así como las siguientes características de estos. Descripción: El Grand Slam se compone de cuatro torneos anuales que se celebran en Gran Bretaña, Estados Unidos, Francia y Australia. En cada país se pueden desarrollar en distintos lugares (p. ej., en EE.UU. puede desarrollarse en Forest Hill o en Flashing Meadows). Cada partido tiene asociado un premio de consolación para el perdedor que dependerá de la fase en que se encuentre el torneo (p. ej., el perdedor de octavos de final puede ganar 5.000 dólares). El ganador de la final recibirá el premio correspondiente al torneo. Cada torneo tiene cinco modalidades:

Individual masculino, individual femenino, dobles masculino, dobles femenino y dobles mixtos. También hay que tener en cuenta la nacionalidad de un jugador, de forma que éste puede ser apátrida o tener varias nacionalidades.

Resultados a considerar: El sistema debe dar respuesta a las siguientes preguntas:

1. Dado un año y un torneo, composición y resultado de los partidos.

2. Lista de árbitros que participaron en el torneo.

3. Ganancias percibidas en premios por un jugador a lo largo del torneo.

4. Lista de entrenadores que han entrenado a un jugador a lo largo del torneo y fechas en las que lo hizo.

Ejemplos de acceso a la base de datos.

1. Connors ganó a Gerulaitis en Roland Garros en 1979 en cuartos de final en individuales masculinos por 6-3 4-6/7-5 6-0.

2. El señor Wilkinson arbitro ese partido.

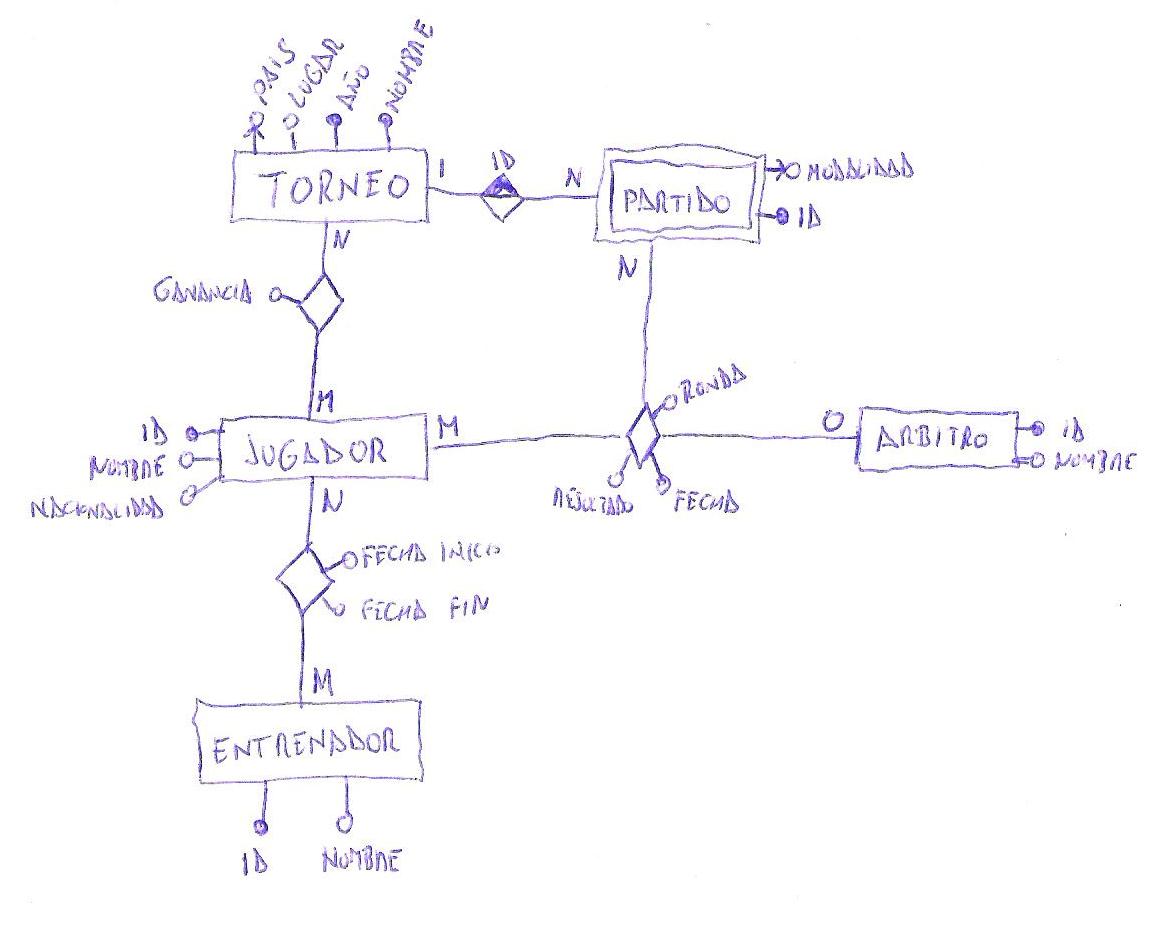
3. Alemania ha ganado dos veces las individuales masculinas de Wimbledon.

4. Borg ha ganado 2.000.000 de dólares a lo largo de su participación en el Grand Slam.

5. El ganador de Roland Garros de 1987 ganó 20.000 dólares.

6. Noah ha jugado cuatro veces en dobles mixtos con Mandlikova.

MODELO ENTIDAD-INTERRELACION



MODELO RELACIONAL (**PK-Negrita,** *FK-Cursiva)*

Torneo (**Nombre, Año,** País, Lugar)

Partido (**ID,** Modalidad, *Año, Nombre)*

Jugador (**ID,** Nombre, Nacionalidad)

Arbitro (**ID,** Nombre)

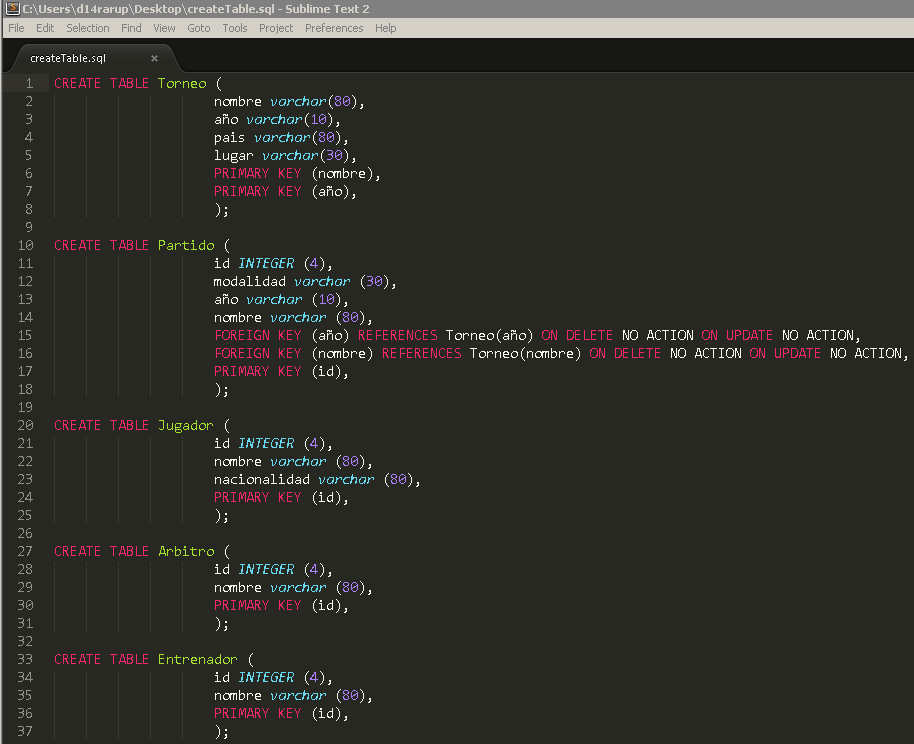
Entrenador (**ID,** Nombre)

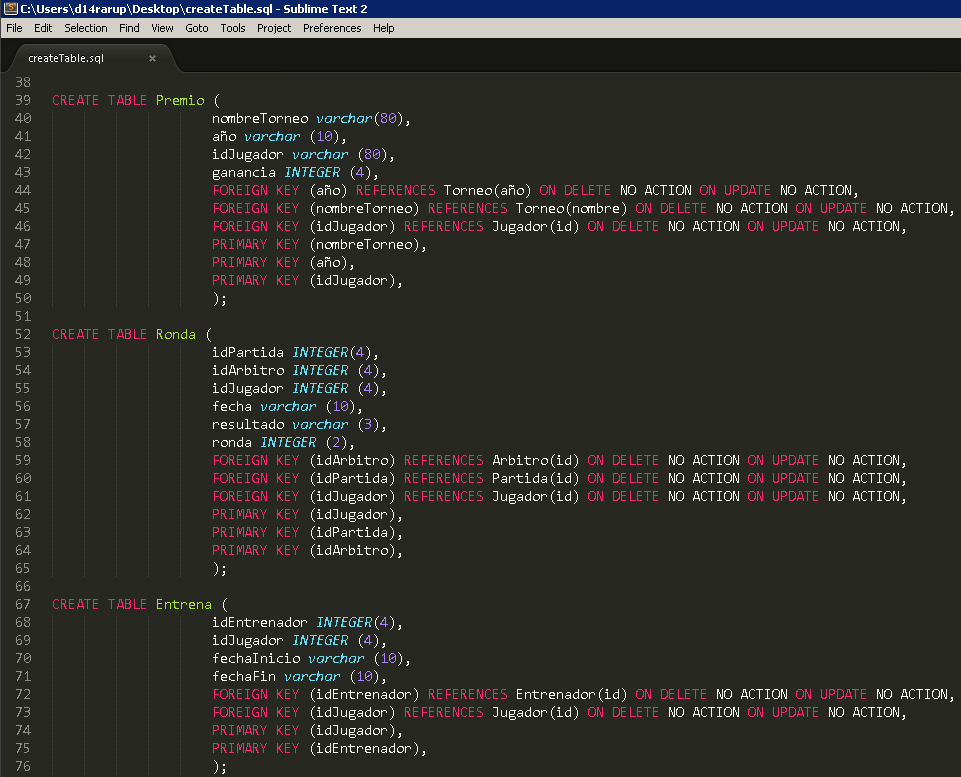
Premios (Torneo-Jugador) (***NombreTorneo, Año, IdJugador,*** Ganancia)

Ronda (Jugador-Partido-Arbitro) (***IDJugador, IDArbitro, IDPartido,*** Fecha, Resultado, Ronda)

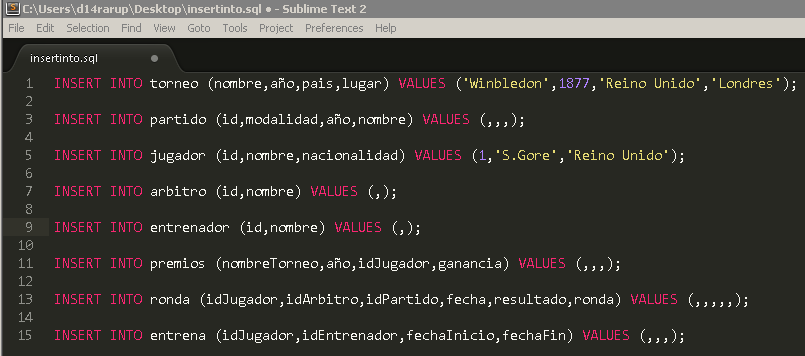
Entrena (Jugador-Entrenador) (***IDJugador, IDEntrenador,*** FechaInicio, FechaFin)

Script SQL de creación del esquema





Script SQL de introducción de datos



# Practica 5

Requisitos:

• No utilizar herramientas de modelado.

Para el supuesto propuesto realiza:

• Diseño conceptual.

• Diseño lógico.

Supuesto.

Se desea diseñar un sistema de control de vuelos adaptado a las siguientes reglas de gestión.

a) De cada aeropuerto se conoce su código, nombre, ciudad y país.

b) En cada aeropuerto pueden tomar tierra diversos modelos de aviones (el modelo de un avión determina su capacidad, es decir, el número de plazas.

c) En cada aeropuerto existe una colección de programas de vuelo. En cada programa de vuelo se indica el número de vuelo, línea aérea y días de la semana en que existe dicho vuelo.

d) Cada programa de vuelo despega de un aeropuerto y aterriza en otro.

e) Los números de vuelo son únicos para todo el mundo.

f) En cada aeropuerto hay múltiples aterrizajes y despegues. Todos los aeropuertos contemplados están en activo, es decir, tienen algún aterrizaje y algún despegue.

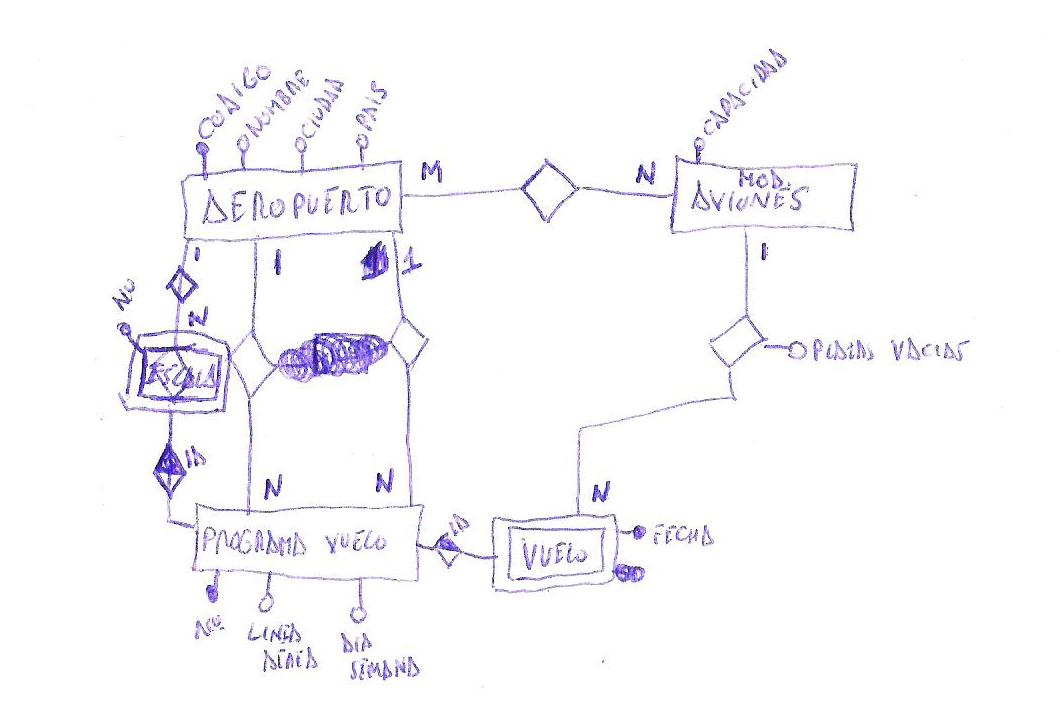
g) Cada vuelo realizado pertenece a un cierto programa de vuelo. Para cada vuelo se quiere conocer su fecha, plazas vacías y el modelo de avión utilizado.

h) Algunos programas de vuelo incorporan escalas técnicas intermedias entre los aeropuertos de salida y de llegada. Se entiende por escala técnica a un aterrizaje y despegue consecutivos sin altas o bajas de pasajeros.

i) De cada vuelo se quieren conocer las escalas técnicas ordenadas asignándole a cada una un número de orden. Por ejemplo, el programa de vuelo 555 de Iberia con vuelos los lunes y jueves despega de BarajasMadrid-España y aterriza en Caudell-Sydney-Australia teniendo las siguientes escalas técnicas: 1- Los P radiños-Sao Paulo-Brasil, 2-Eºl Emperador-Santiago-Chile y 3-Saint Kitts-Auckland-Nueva Zelanda.

¿Qué cambios se producirán en el caso anterior si en las escalas pudiesen bajar o subir pasajeros? Explicar cómo se podría representar esta nueva situación.

MODELO ENTIDAD-INTERRELACION



MODELO RELACIONAL (**PK-Negrita,** *FK-Cursiva)*

Aeropuerto (**CodAeropuerto,** nombre, ciudad, país)

Programa Vuelo (**NºProgVuelo,** línea aérea, día semana, *CodAeropuertoSalida, CodAeropuertoLlegada)*

ModAviones (**IDModAvion,** cantidad\_pasajeros)

Vuelo (**Fecha, NºProgVuelo,** *IDModAvion, PlazasVacias)*

Escala (**NºEscala, NºProgVuelo,** *CodAeropuerto)*

Aeropuerto-ModAviones (***CodAeropuerto, IDModAvion)***

# practica 6

Normaliza las siguientes relaciones.

RELACIÓN 1.

Información relativa a los préstamos que realizan las editoriales a los profesores de primaria de los colegios para su evaluación en alguna de las asignaturas/habilidades que imparten.

A un profesor no se le puede prestar más de un libro de la misma editorial en el mismo día y a un profesor no se le puede prestar más de una vez un mismo libro

PRÉSTAMO\_LIBROS (Colegio, ProfesorInfantil, Asignatura\_Habilidades, Aula, Curso, Libro, Editorial, Fecha\_Préstamo)

T1 (profesor, **colegio**)

T2 (***profesor***, fechaPrestamo, ***libro***)

T3 (**libro**, editorial)

T4 (**profesor**, curso, asignatura, aula)

RELACIÓN 2.

Información relativa a la utilización de los recursos de la universidad por parte de los profesores en periodos de tiempo:

UTILIZACIÓN (DNI\_Prof, Nombre\_Prof, Despacho\_Prof, Recurso, Ubicación, Fecha\_inicio, Fecha\_fin)

T1 (***DniProfesor***, ***recurso***, **fechaInicio**, **fechaFin**)

T2 (**DniProfesor**, nombreProfesor, descripcionProfesor)

T3 (**recurso**, descripción)

RELACIÓN 3.

Información relativa a la asignación de los turnos de trabajo de los empleados de los distintos centros de una cadena de tiendas de moda.

ASIGNACIÓN (DNI, Nombre, Código\_Tienda, Dirección\_Tienda, Fecha, Turno)

T1 (***DNI***, ***codigoTienda***, fecha, turno)

T2 (**DNI**, nombre)

T3 (**codigoTienda**, direccionTienda)

RELACIÓN 4.

Información sobre la fecha y duración de las actividades deportivas que se organizan en un colegio.

Los nombres de los monitores no son únicos y los nombres de las actividades tampoco.

REALIZA (Cod\_Actividad, Nombre\_Actividad, DNI\_Monitor, Nombre\_monitor, Sala, Fecha, Hora\_I, Hora\_F)

T1 (**Cod\_Actividad**, Nombre\_Actividad)

T2 (**DNI\_Monitor**, Nombre\_Monitor)

T3 (***Cod\_Actvidad***, ***DNI\_Monitor***, Sala, Fecha, Hora\_I, Hora\_F)

# practica 7

Requisitos:

• SGBDR: Oracle.

• Herramientas: Data-modeler, sql-developer

Para el supuesto propuesto realiza:

• Diseño conceptual.

• Diseño lógico.

• Implementación.

• Carga de datos de pruebas.

• Disparador que penalice a los lectores que devuelvan los libros prestados con retraso. La penalización consiste en bloquear al lector para que no pueda realizar prestamos durante tantas semanas como días de retraso haya acumulado.

Supuesto

Se desea diseñar una solución para el sistema de información de una biblioteca. Las personas socias de la biblioteca disponen de un código de socio y almacenamos su dni, dirección, teléfono, nombre y apellidos.

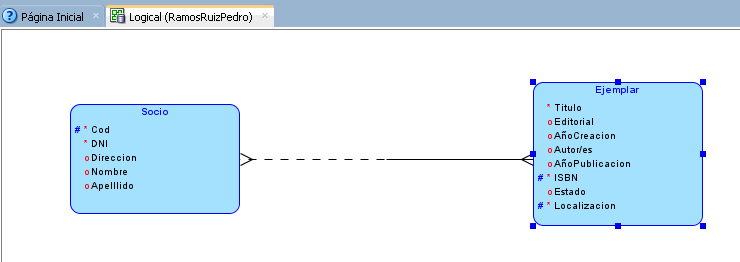
La biblioteca almacena libros que presta a los socios y socias, de ellos se almacena su título, su editorial, el año en el que se escribió el libro, el nombre completo del autor (o autores), el año de edición y el ISBN.

De cada libro en la biblioteca existen varios ejemplares. Necesitamos conocer el estado y la localización de cada ejemplar.

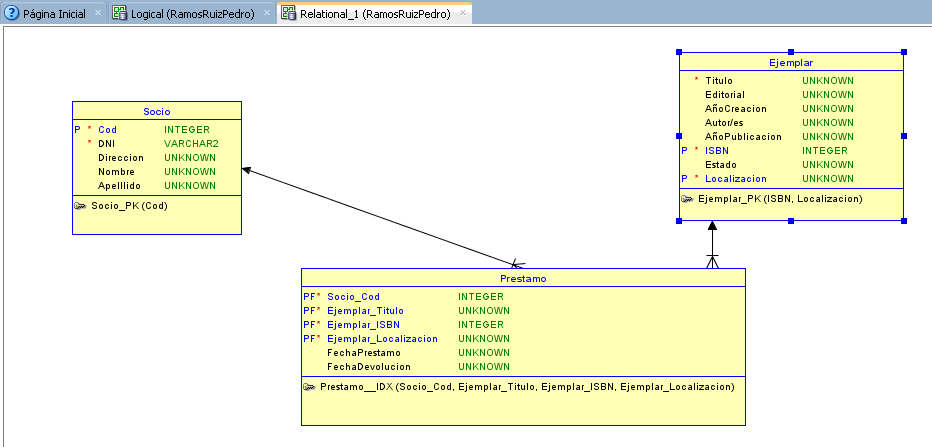
Se desea diseñar un sistema que permita una gestión ágil de los préstamos. Un socio puede retirar varios ejemplares al mismo tiempo. El sistema debe permitir la reclamación a los usuarios de aquellos libros no devueltos.

Incorporar todas las suposiciones semánticas que permitan un desarrollo más completo del modelo.

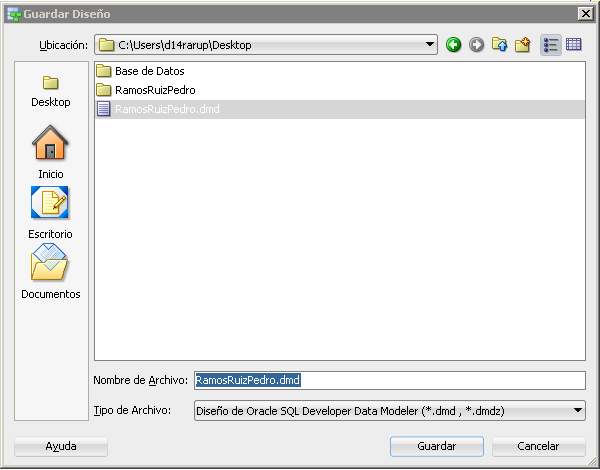
Diseño conceptual.



Diseño lógico.



Implementación



# practica 8

Requisitos:

• SGBDR: Oracle.

• Esquema HR .

Consultas sql:

1. Mostrar detalles de los trabajos donde el salario mínimo es mayor que 10.000

SELECT job\_title

FROM jobs

WHERE min\_salary > 10000;

1. Ver el primer nombre y la fecha de ingreso de los empleados dados de alta entre 2002 y 2005.

SELECT first\_name, hire\_date

FROM employees

WHERE hire\_date > ‘01/01/02’ AND hire\_date < ‘31/12/05’;

SELECT first\_name, hire\_date

FROM employees

WHERE TO CHAR (hire\_date, YYYY) BETWEEN 2002 AND 2005’;

1. Nombre y fecha de ingreso de los empleados que se sean IT Programmer o Sales Man.

SELECT e.first\_name, e.hire\_date, j.job\_title

FROM employees e, Jobs j

WHERE e.job\_id = j.job\_id

AND j.job\_title=’IT Programmer’

OR j.job\_title=’Sales Man’;

1. Empleados que ingresaron en la empresa después del 1 de enero de 2008.

SELECT \*

FROM employees

WHERE hire\_date > ‘01/01/08’;

1. Detalle de los empleados con ID 150 o 160.

SELECT \*

FROM employees

WHERE id=150

OR id=160;

1. Mostrar first name, salary, commission pct, and hire date de los empleados con salario inferior a 10000.

SELECT first\_name, salary, commission\_pct, hire\_date

FROM employees

WHERE salary < 10000;

1. Mostar el trabajo, y la diferencia entre el salario mínimo y máximo de los trabajos con el salario máximo en el rango 10000 a 20000.

SELECT job\_title, max\_salary-min\_salary AS diferencia

FROM Jobs

WHERE max\_salary >= 10000

AND max\_salary <= 20000;

1. Mostrar el nombre, salario y salario redondeado a miles, de los empleados.

SELECT first\_name, salary, ROUND(salary,-3)

FROM employees

1. Mostar los datos de los trabajos en orden descendiente de título.

SELECT \*

FROM Jobs

ORDER BY job\_tittle desc;

1. Mostrar aquellos empleados cuyo nombre o apellidos comiencen con la letra S.

SELECT \*

FROM employees

WHERE first\_name LIKE ‘S%’ OR last\_name LIKE ‘S%’;

1. Mostrar los empleados que Display employees who joined in the month of May.

SELECT \*

FROM employees

WHERE To\_Char(Hire\_date,’MON’)=’MAY’;

1. Mostrar detalle de los empleados sin porcentaje de comisión, el salario en el rango 5000 a 10000 y departamento 30.

SELECT \*

FROM employees

WHERE salary <10000 AND salary >5000 AND department\_id=30 AND comisión\_pct is null;

1. Mostrar nombre y experiencia en años de los empleados.

SELECT first\_name, ((SYSDATE - hire\_date) / 365) AS experiencia

FROM employees;

1. Mostrar nombre de los empleados que entraron en la empresa en 2001.

SELECT first\_name

FROM employees

WHERE hire\_date > ‘01/01/01’;

SELECT first\_name

FROM employees

WHERE TO\_CHAR(hire\_date, ‘yyyy’)=2001;

1. Mostrar nombre y apellidos de los empleados convirtiendo la primera letra en mayúscula y el resto en minúscula.

SELECT INITCAP (first\_name), INITCAP (last\_name)

FROM employees

1. Mostrar detalle de los empleados que han ingresado en la empresa este año.

SELECT \*

FROM employees

WHERE hire\_date > ‘01/’01/15’;

1. Mostrar cuantos empleados han ingresado en la empresa cada mes en este año.

SELECT TO\_CHAR(hire\_date, ‘MM’), count (\*)

FROM employees

WHERE TO\_CHAR(hire\_date, ‘YYYY’)= TO\_CHAR(SYSDATE,’YYYY’)

GROUP BY TO\_CHAR(hire\_date,’MM’);

1. Mostrar el país y número de ciudades que tiene.

SELECT country\_id, count(\*)

FROM locations

GROUP BY country\_id;

1. Mostrar el salario medio de los empleados de cada departamento que tienen comisiones.

SELECT avg(salary), d.department\_name, e.department\_id

FROM employees e, department d

WHERE e.comission\_pct IS NOT NULL

AND e.deparpartment\_id = d.department\_id

GROUP BY e.department\_id, d.department\_name;

1. Mostrar el id del trabajo, número de empleados en ese trabajo, suma del salario y diferencia entre el salario mayor y menor de cada trabajo.

SELECT job\_id, count(\*), sum(salary) max\_salary – min\_salary AS diferencia

FROM Jobs j, employees e

WHERE j.jobs\_id = e.jobs\_id

GROUP BY job\_id;

1. Muestra el id de aquellos trabajos cuyo salario medio es superior a 10000.

SELECT job\_id, AVG (salary)

FROM employees

GROUP BY job\_id;

HAVING AVG(salary)>10000

1. Mostrar los años en los que se incorporaron a la empresa más de 10 empleados.

SELECT TO\_CHAR(hire\_date, ‘yyyy’)

FROM employees

GROUP BY TO\_CHAR(hire\_date, ‘yyyy’)

HAVING count(employee\_id) > 10;

1. Mostrar los departamentos con más de cinco empleados que tienen porcentaje de comisión.

SELECT d.department\_id, e.department\_name

FROM employees e, department d

WHERE d.department\_id = e.department\_id

AND commission\_pct IS NOT NULL

GROUP BY department\_id, department\_name

HAVING COUNT(commission\_pct) > 5;

1. Mostrar el id de los empleados que han tenido más de un trabajo en el pasado.

SELECT employee\_id

FROM job\_history

GROUP BY employee\_id

HAVING COUNT(\*) > 1;

1. Mostrar el id de aquellos trabajos que han tenido más de 3 empleados más de 100 días.

SELECT job\_id

FROM job\_history

WHERE end\_date-start\_date > 100

GROUP BY job\_id

HAVING COUNT(\*) > 3;

1. Mostrar id de departamento, año y número de empleados dados de alta en ese año.

SELECT department\_id, TO\_CHAR(hire\_date,’yyyy’), COUNT(EMPLOYEE\_ID)

FROM employees

GROUP BY department\_id, TO\_CHAR(hire\_date,’yyyy’);

1. Mostrar los departamentos donde el jefe tiene más de 5 empleados a su cargo.

SELECT department\_id

FROM employees

GROUP BY department\_id, manager\_id

HAVINGE COUNT(employee\_id) > 5;

1. Cambiar el salario del empleado 115 a 8000 si cobra menos de 6000.

UPDATE employees

SET salary = 8000

WHERE employee\_id = 115 AND salary < 6000;

1. Inserta un nuevo empleado: 207, 'ANGELA', 'SNYDER','ANGELA','215 253 4737', SYSDATE, 'SA\_MAN', 12000, 80

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, deparment\_id)

VALUES (207, ‘Angela’, ‘Snyder’, ‘Angela’ 2152534737, SYSDATE, ‘SA\_MAN’, 12000, 80);

1. Borra el departamento 20

DELETE FROM departments

WHERE department\_id=20;

1. Cambia el trabajo del empleado 110 a IT\_PROG, si el empleado pertenece al departamento 19 y su trabajo actual no empieza con IT.

UPDATE employees

SET job\_id = ‘IT\_PROG’

WHERE employee\_id= 110

AND department\_id=19

AND NOT job\_id LIKE ‘IT%’;

1. Inserta una fila en la tabla departamentos con jefe 1w9 y alguna localización de la ciudad de Tokyo.

INSERT INTO departaments (deparment\_id, department\_name, manager\_id, location\_id)

VALUES (200, ‘DAW’,’1w9’,1200);

1. Muestra el nombre del departamento y su número de empleados.

SELECT departament\_name, count(\*)

FROM departaments d, employees e

WHERE d.department\_id = e.department\_id

GROUP BY department\_name;

1. Muestra el nombre del departamento y el nombre de su jefe.

SELECT departament\_name, first\_name

FROM departments d, employees

WHERE d.manager\_id = e.employee\_id;

1. Muestra nombre del departamento, nombre del jefe y ciudad.

SELECT department\_name, first\_name, city

FROM department d, employees e, locations l

WHERE e.employee\_id = d.manager\_id

AND d.location\_id = l.location\_id

1. Muestra nombre del país, ciudad y nombre de departamento.

SELECT country\_name, city, department\_name

FROM country c, departments d, locations l

WHERE d.location\_id = l.location\_id

AND l.country\_id = c.country\_id;

1. Muestra título del trabajo, nombre del departamento, apellido del empleado y fecha de inicio para todos los trabajos entre 2000 y 2005.

SELECT job\_title, department\_name, last\_name, start-date

FROM jobs j, department d, employees e, job\_history h

WHERE j.job\_id = e.job\_id

AND d.department\_id = e.department\_id

AND e.employee\_id = h.employee\_id

AND TO\_CHAR(start-date,’yyyy’) BETWEEN 2000 AND 2005;

1. Mostrar detalle de los departamentos con salario máximo mayor a 10000.

SELECT d.\*

FROM departaments d, employees e, Jobs j

WHERE d.department\_id = e.department\_id

AND e.job\_id = j.job\_id

AND max\_salary > 10000;

1. Mostrar detalle de los departamentos cuyo jefe es ‘Smith’.

SELECT \*

FROM departments d,

(SELECT department\_id

FROM deparment d, employees e

WHERE d.manager\_id = e.employee\_id

AND last\_name\_name = ‘Smith’) s

WHERE d.department\_id = s.deparment\_id

1. Mostrar trabajos donde se han incorporado empleados este año.

SELECT \*

FROM Jobs

WHERE job\_id IN

SELECT job\_id

FROM employees

WHERE TO\_CHAR (hire\_date,’yyyy’) = TO\_CHAR (SYSDATE, ‘yyyy’

1. Mostrar empleados que no tuvieron ningún puesto en el pasado.

SELECT \*

FROM employees

WHERE employee\_id NOT IN

SELECT employee\_id

FROM job\_history;

1. Mostrar título del trabajo y salario medio de aquellos empleados que no tuvieron ningún puesto en el pasado.

SELECT job\_title, AVG(salary)

FROM employees

GROUP BY job\_title

HAVING employees\_id IN

SELECT employee\_id

FROM job\_history

1. Mostrar detalle de los jefes con más de 5 empleados.

SELECT \*

FROM employees

WHERE employees\_id IN

SELECT manager\_id

FROM employees

GROUP BY manager\_id

HAVING COUNT(\*) > 5

1. Mostrar los departamentos que no tuvieron alta de empleados en los últimos dos años.

SELECT department\_name

FROM departments

WHERE department\_id NOT IN

SELECT department\_id

FROM employee

WHERE (SYSDATE – hire\_date( / 365 < 2

1. Mostrar detalle de los departamentos en los que el salario máximo es mayor a 10000 para empleados que no tuvieron ningún puesto en el pasado.

SELECT \*

FROM departments

WHERE deparment\_id IN

SELECT department\_id

FROM employees

WHERE employee\_id IN

SELECT employee\_id

FROM job\_history

GROUP BY department\_id

HAVING MAX(salary) > 10000

1. Mostrar detalle del puesto actual de los empleados que en el pasado trabajaron como IT Programmers.

SELECT \*

FROM Jobs

WHERE job\_id IN

SELECT job\_id

FROM employees

WHERE employee\_id IN

SELECT employee\_id

FROM job\_history

WHERE job\_id=’IT\_PROG’

1. Mostrar la ciudad del empleado cuyo id es 105.

SELECT city

FROM locations l, departments d, employees e

WHERE l.location\_id = d.location\_id

AND d.manager\_id = e.employee\_id

AND employee\_id = 105

1. Muestra el tercer salario más alto de todos los empleados.
2. Mostrar jefes y empleados a su cargo.

SELECT e1.first\_name, e2.first\_name AS jefe

FROM employees e1, employees e2

WHERE e1.employee\_id = e2.manager\_id

1. Mostrar el jefe con el sueldo más alto.

SELECT MAX(salary), first\_name)

FROM

SELECT salary, first\_name

FROM employees e

WHERE e.employees\_id IN

SELECT manager\_i

FROM departments

GROUP BY first\_name

# practica 9

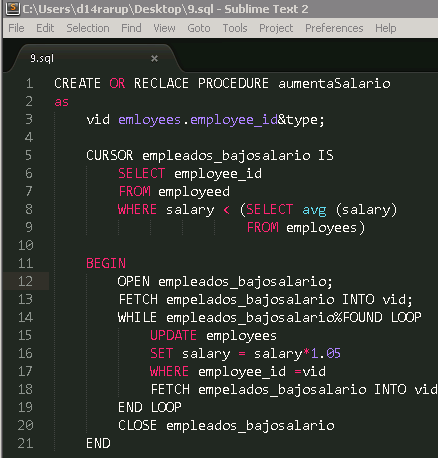
Requisitos:

• SGBDR: Oracle.

• Esquema HR.

• PL/SQL

1. Escribe un disparador que incremente en un 5 % el salario de todos los empleados cuyo sueldo esté por debajo de la media.

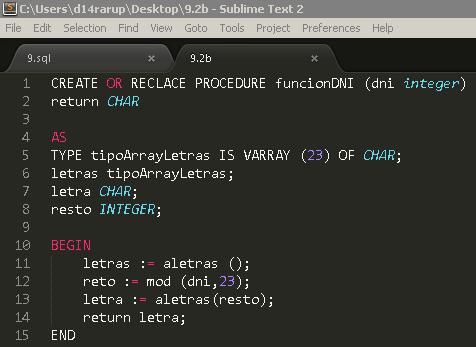


1. Realiza:

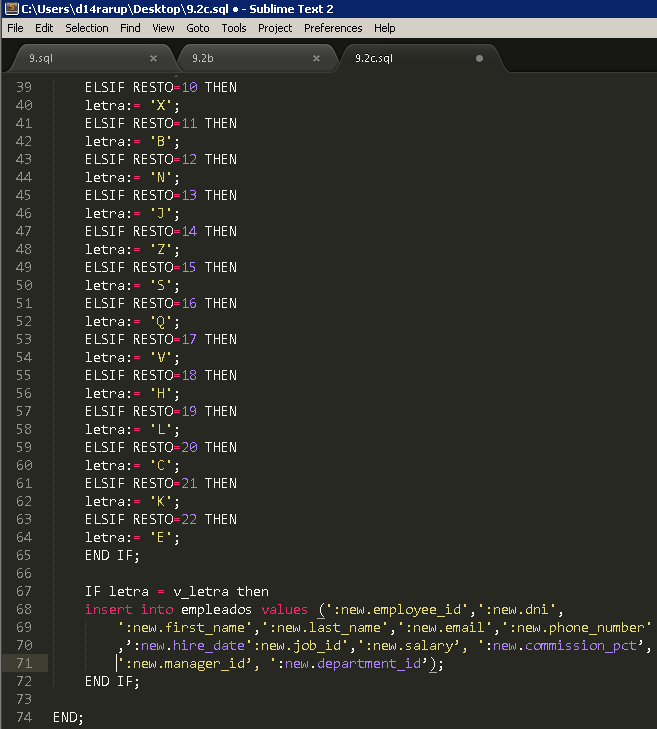
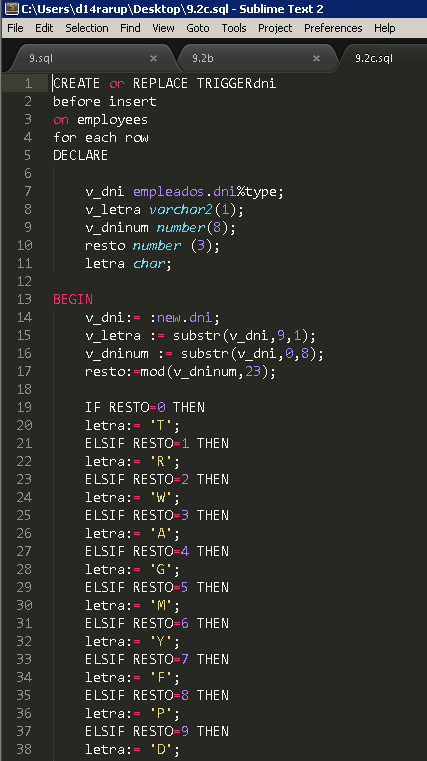
* Añade a la tabla empleados el campo nif.

ALTER TABLE employees SET dni varchar(9)

* Crea una función que calcule la letra que corresponde a un dni.



* Crea un disparador que utilice la función anterior y garantice que los datos introducidos en el campo nif son correctos.
* Prueba la función y el disparador introduciendo un nuevo empleado con tus datos personales.



# practica 10

Requisitos:

• Diseño lógico.

• SGBDR: Oracle.

• Herramientas: Sql-developer.

Realiza:

* Importar esquema anexo.
* Consultas sql:

1. Obtener el nombre y el equipo de los ciclistas menores de 30 años que hayan ganado alguna etapa

SELECT nombre, nomeEquipo,

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM etapa

GROUP BY dorsal;

1. Obtener el nombre y el equipo de los ciclistas mayores de 32 años que hayan ganado algún puerto.

SELECT nombre, nomeEquipo

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM puerto

AND edad > 32

GROUP BY dorsal;

1. Obtener los datos de las etapas que pasan por algún puerto de montaña y que tienen salida y llegada en la misma población.

SELECT \*

FROM etapas

WHERE numetapa IN

SELECT numetapa

FROM puerto

AND llegada = salida

GROUP BY numetapa;

1. Obtener las poblaciones que tienen la meta de alguna etapa, pero desde las que no se realiza ninguna salida.

SELECT llegada

FROM etapa

WHERE NOT llegada = salida;

1. Obtener las poblaciones las de la llegada pero que no están en la salida.

SELECT llegada

FROM etapa

WHERE NOT llegada = salida;

1. Obtener el nombre y el equipo de los ciclistas que han ganado alguna etapa llevando el maillot amarillo, mostrando también el número de etapa

SELECT c.nombre, c.nomeEquipo, e.numetapa

FROM ciclista c, etapa e

WHERE dorsal IN

(SELECT dorsal

FROM etapa

WHERE dorsal IN

SELECT dorsal

FROM lleva l, maillot m

WHERE l.codigo = m.codigo

AND color = ‘amarillo’

AND c.dorsal = e.dorsal

1. Obtener las poblaciones de salida y de llegada de las etapas donde se encuentran puertos con altura superior a 1300 metros.

SELECT salida, llegada

FROM etapas

WHERE numetapa IN

SELECT numetapa

FROM puerto

WHERE altura > 1300

GROUP BY numetapa;

1. Obtener el número de las etapas que tienen algún puerto de montaña, indicando cuantos tiene cada una de ellas.

SELECT numetapa, count(nompuerto)

FROM puerto

GROUP BY numetapa;

1. Obtener el nombre y la edad de los ciclistas que han llevado dos o más maillots en una misma etapa.

SELECT nombre, edad

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM lleva

WHERE count(dorsal) > 2

GROUP BY dorsal;

1. Obtener el nombre y el equipo de los ciclistas que han llevado algún maillot o que han ganado algún puerto.

SELECT nombre, namequipo

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM lleva

OR dorsal IN

SELECT dorsal

FROM puerto

GROUP BY dorsal

1. Obtener los números de las etapas que no tienen puertos de montaña.

SELECT numetapa

FROM etapa

WHERE numetapa NOT IN

SELECT numetapa

FROM puerto

1. Obtener la edad media de los ciclistas que han ganado alguna etapa.

SELECT AVG(edad)

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM etapa

GROUP BY dorsal

1. Obtener el nombre de los puertos de montaña que tienen una altura superior a la altura media de todos los puertos.

SELECT nompuerto

FROM puerto

WHERE altura > avg(altura)

1. Obtener las poblaciones de salida y de llegada de las etapas donde se encuentran los puertos con mayor pendiente.

SELECT llegada, salida

FROM etapas

WHERE numetapa IN

SELECT numetapa

FROM puerto

WHERE pendiente = MAX(pendiente)

1. Obtener el dorsal y el nombre de los ciclistas que han ganado los puertos de mayor altura.

SELECT dorsal, nombre

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM puerto

WHERE altura = max(altura)

GROUP BY dorsal;

1. Obtener los datos de las etapas cuyos puertos (todos) superan los 1300 metros de altura.
2. Obtener la edad del ciclista más joven, la del más veterano y la edad media de los ciclistas que han participado en la vuelta.

SELECT MIN(edad), MAX(edad), avg(edad)

FROM ciclista

WHERE ¿participado en la vuelta?

1. Obtener el nombre del equipo y el director del ciclista que gano la etapa más larga.

SELECT nomeequipo

FROM ciclista

WHERE dorsal IN

SELECT dorsal

FROM etapa

WHERE kms = max(kmx)

1. Obtener el dorsal y el nombre de los ciclistas que hayan ganado alguna etapa, mostrando el número de etapas que han ganado.

SELECT c. dorsal, c.nombre, count(dorsal)

FROM ciclista c, etapa e

WHERE c.dorsal = e.dorsal

GROUP BY c.dorsal

1. Obtener el nombre de los ciclistas que pertenecen a un equipo de más de cinco ciclistas y que han ganado alguna etapa, indicando también cuantas etapas han ganado.
2. Nombre y equipo de los ciclistas que han llevado alguna vez el maillot amarillo, indicando durante cuantas etapas lo han llevado.
3. Por cada equipo, color de los maillots que han llevado sus ciclistas
4. Nombre y equipo del ganador de la vuelta (es decir, el que ha lucido el maillot amarillo en la última etapa).
5. Nombre de los equipos que no han llevado el maillot amarillo.

SELECT nomequipo

FROM equipo

WHERE nomeequipo NOT IN

SELECT nomequipo

FROM equipo

WHERE dorsal IN

SELECT dorsal

WHERE lleva

GROUP BY dorsal

GROUP BY nomequipo

1. Nombre y dorsal de los ciclistas mayores de 30 años que han ganado algún puerto, junto con el número de puertos que han ganado.

SELECT nombre, dorsal, count(dorsal)

FROM ciclista c, puerto p

WHERE c.dorsal = p.dorsal

AND edad > 30

GROUP BY dorsal

1. Nombre y director de los equipos que, en alguna etapa, sus ciclistas han llevado tres o más maillots.
2. Nombre de los equipos que solo tienen ciclistas menores de 28 años.
3. Dorsal y nombre del ciclista que ha llevado el maillot amarillo durante más etapas.
4. Obtener los datos de las etapas que pasan por algún puerto de montaña y que tienen salida y llegada en la misma población.
5. Obtener las poblaciones que tienen la meta de alguna etapa, pero desde las que no se realiza ninguna salida.
6. Obtener el nombre y el equipo de los ciclistas que han ganado alguna etapa llevando el maillot amarillo, mostrando también el número de etapa.
7. Obtener los datos de las etapas que no comienzan en la misma ciudad en que acaba la etapa anterior.
8. Obtener el número de las etapas que tienen algún puerto de montaña, indicando cuántos tiene cada una de ellas.
9. Obtener el nombre y la edad de los ciclistas que han llevado dos o más maillots en una misma etapa.
10. Obtener el nombre y el equipo de los ciclistas que han llevado algún maillot o que han ganado algún puerto.
11. Obtener los datos de los ciclistas que han vestido todos los maillots (no necesariamente en la misma etapa).
12. Obtener el código y el color de aquellos maillots que sólo han sido llevados por ciclistas de un mismo equipo.
13. Obtener los números de las etapas que no tienen puertos de montaña.
14. Obtener la edad media de los ciclistas que han ganado alguna etapa.
15. Obtener el nombre de los puertos de montaña que tienen una altura superior a la altura media de todos los puertos.
16. Obtener las poblaciones de salida y de llegada de las etapas donde se encuentran los puertos con mayor pendiente.
17. Obtener el dorsal y el nombre de los ciclistas que han ganado los puertos de mayor altura.
18. Obtener los datos de las etapas cuyos puertos (todos) superan los 1300 metros de altura.
19. Obtener el nombre de los ciclistas que pertenecen a un equipo de más de cinco ciclistas