Titulación: GRADO EN INGENIERÍA INFORMÁTICA
Año Académico: 2024/25 – 2º cuatrimestre
Asignatura: Ficheros y bases de datos (2º curso)
Memoria de Prácticas 1: Diseño Relacional y Carga de Datos

uc3m | Universidad Carlos III de Madrid

Profesor:	**Daniel Esteban Villamil Sierra**	**Grupo**	**85**
Alumno/a:	**Álvaro González Fúnez**	**NIA:**	**100451281**
Alumno/a:	**Jhonatan Barcos Gambaro**	**NIA:**	**100548615**

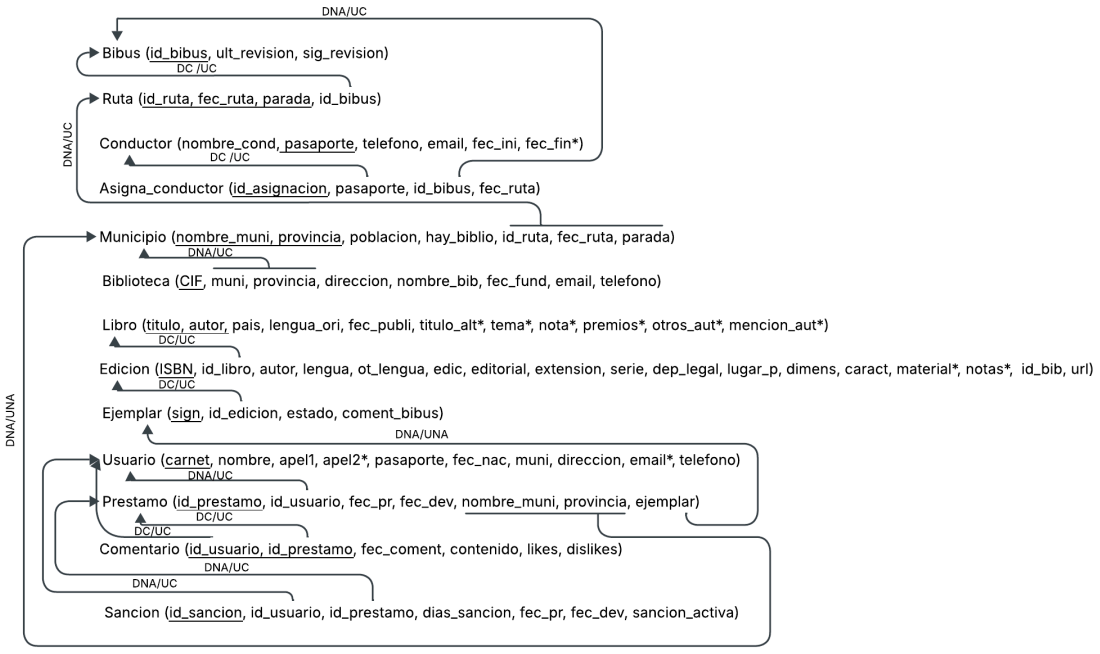
1. Introducción

Este proyecto consiste en el diseño e implementación de una nueva base de datos para Fundicu.org, la fundación para la Difusión de la Cultura.
Hemos trabajado con las unidades móviles, sus rutas y conductores, los libros que se transportan y los usuarios que forman parte de este proceso.
En primer lugar, hemos creado un esquema relacional, incorporando su semántica implícita y la semántica explícita contemplada. Posteriormente, creamos las tablas nuevas con sus restricciones correspondientes en SQL y finalmente insertamos datos de las antiguas tablas en las nuevas.

2. Diseño Relacional

Esta sección se subdivide en tres apartados:

- Esquema relacional: diseño completo, con la notación de grafo/esquema relacional vista en clase (puede entregarse hecho a mano o con las herramientas de dibujo proporcionadas por Microsoft Office). Es importante que el grafo se visualice claramente.



- Semántica implícita: supuestos semánticos que, por referirse a información ausente en la descripción explícita (es decir, no se encuentran en el enunciado), es necesario añadir para completar el diseño.

Sup_id	Mecanismo	Descripción
I_1	Clave primaria	Asigna_conductor se identifica mediante id_asignacion
I_2	Clave primaria	Sanción se identifica con id_sancion
I_3	Clave primaria	Ruta se identifica como id_ruta, fec_ruta y parada
I_4	Clave primaria	Prestamo se identifica con id_prestamo
I_5	Semántica	Los ejemplares se relacionan con las ediciones, de la misma edición hay varios ejemplares.
I_6	Semántica	El nombre completo del conductor se almacena en nombre_cond.
I_7	Semántica	Establecemos la fecha de la última ITV como atributo de bibus.
I_8	Semántica	Establecemos la fecha de la siguiente ITV como atributo de bibus
I_9	Semántica	Añadimos el atributo 'sancion_activa' a la tabla sancion.

Tabla 1: Semántica implícita

- Semántica explícita no contemplada en el diseño: supuestos semánticos indicados en el enunciado que no han podido representarse en el esquema relacional. Para cada uno de los supuestos, crea una fila en la tabla presentada a continuación.

Sup_id	Descripción
S_1	No tenemos en cuenta que los usuarios puedan tener máximo 2 préstamos al mismo tiempo.
S_2	No contamos con que las bibliotecas puedan tener 2 libros cada 10 habs. de su municipio.
S_3	No valoramos que un bibús solo realiza una ruta diaria.
S_4	No contamos con que los libros que estén dados de baja no puedan ser utilizados.
S_5	No consideramos que los municipios sin bibliotecas tengan una parada de 15 min para préstamos.
S_6	No contemplamos que las sanciones se apliquen en función de las semanas de retraso.
S_7	No tenemos presente que un usuario con sanción no pueda reservar ni hacer préstamos y pierde sus reservas activas
S_8	No valoramos que un conductor pueda estar en ruta, descanso o sucursal.
S_9	No contamos con que el estado del ejemplar sea entre {nuevo, bueno, gastado, muy usado, deteriorado}
S_{10}	No garantizamos que los comentarios del bibusero sean hasta 500 caracteres
S_{11}	No registramos cuándo se dan de baja los ejemplares
S_{12}	No podemos implementar que los préstamos sean de dos semanas

Sup_id	Descripción
S_1	No tenemos en cuenta que los usuarios puedan tener máximo 2 préstamos al mismo tiempo.
S_2	No contamos con que las bibliotecas puedan tener 2 libros cada 10 habs. de su municipio.
S_3	No valoramos que un bibús solo realiza una ruta diaria.
S_4	No contamos con que los libros que estén dados de baja no puedan ser utilizados.
S_5	No consideramos que los municipios sin bibliotecas tengan una parada de 15 min para préstamos.
S_{13}	Tomamos las reservas (préstamos futuros) como simples préstamos.
S_{14}	No hacemos que la sanción inhabilite los préstamos o reservas

Tabla 2: Semántica explícita no contemplada

3. Implementación de la Estática Relacional en SQL (LDD)

Esta sección complementa al fichero con el script de creación de la base de datos (NEWcreation.sql). Añadiendo los siguientes apartados:

Semántica explícita re-incorporada: Incluir aquellos supuestos de la Tabla 2 que se han podido contemplar con las sentencias de definición de SQL.

Sup_id	Descripción de la solución
S_6	Las sanciones se aplican en función de las semanas de retraso, TO_NUMBER(TO_DATE(l.return, 'DD/MM/YYYY HH24:MI:SS') - TO_DATE(l.date_time, 'DD/MM/YYYY HH24:MI:SS')) - 14 AS dias_sancion

Tabla 3: Semántica explícita re-incorporada

Semántica implícita: (continúa la numeración donde terminó en la tabla 1)

Sup_id	Mecanismo	Descripción
I_{10}	Check (Tabla Municipio)	Comprobamos que 'hay:biblio' sea 'Y' o 'N'
I_{11}	Clave primaria (Tabla biblioteca)	Generamos un CIF único para biblioteca: 'LIB' || TO_CHAR(DBMS_RANDOM.VALUE(1000000000, 9999999999)) AS CIF
I_{12}	Clave primaria (Tabla Asigna_conductor)	Creamos un id de asigna_conductor único: 'ASG' || TO_CHAR(ROWNUM, 'FM0000')

I_{13}	Clave primaria (Tabla Prestamo)	La clave primaria de prestamo: 'P' || TO_CHAR(DBMS_RANDOM.VALUE(1000000000, 9999999999))
I_{14}	Clave primaria (Tabla Sancion)	La clave primaria de sanción se identifica así: NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY

Tabla 1(cont.): Semántica implícita

Semántica excluida: Al crear la base de datos en SQL específico del SGBD Oracle puede que no se hayan podido contemplar algunas restricciones semánticas explícitas (tabla 2 – tabla 3), o implícitas que no han podido incorporarse (tabla 1).

Sup_id	Descripción semántica	Motivo	Explícita/ Implícita
E_1	No tenemos en cuenta que los usuarios puedan tener máximo 2 préstamos al mismo tiempo.	Sería necesario crear un trigger en prestamo restringir los prestamos por persona	Explícita
E_2	No contamos con que las bibliotecas puedan tener 2 libros cada 10 habs. de su municipio.	No podemos actualizar la cantidad de libros que hay en cada biblioteca en cualquier momento.	Explícita
E_3	No contamos con que los libros que estén dados de baja no puedan ser utilizados.	No tenemos información suficiente para saber cuándo un libro está en tan mal estado como para no poder prestarlo.	Explícita
E_4	No hacemos que la sanción inhabilite los préstamos o reservas	La tabla sancion es dependiente y posterior a prestamo, no queremos generar un bucle de dependencia.	Explícita
E_5	No consideramos que los municipios sin bibliotecas tengan una parada de 15 min para préstamos.	No podemos garantizar que la parada dure ese tiempo, no hay una hora de inicio de la parada y otra de fin de parada.	Explícita
E_6	No incorporamos el atributo estado en la tabla ejemplar.	No tenemos ninguna referencia en las tablas obsoletas sobre el estado de los ejemplares que se prestan.	Implícita
E_7	No incorporamos el atributo coment_bibus en la tabla ejemplar.	No tenemos ninguna referencia en las tablas obsoletas sobre los comentarios de los bibuseros en los ejemplares.	Implícita

Tabla 4: Semántica excluida en la creación de tablas

4. Carga de datos (LMD)

Esta sección describirá la carga de datos realizada desde las tablas desnormalizadas entregadas junto con la entrega del fichero de carga (NEWload.sql). A tal efecto, se analizará el problema de la carga y se describirá la solución, haciendo hincapié en:

- El orden de tablas que se adopta para volcar en ellas los datos (justificado).
- Los problemas que surgen (campos obligatorios sin valor, defectos en los datos originales, conversiones de datos, etc) y las soluciones que se adoptan para superarlos.

El orden de creación de tablas se ha decidido en base a las restricciones de clave ajena. Estas restricciones impiden que algunas tablas se creen o se borren antes que otras. Las tablas sin estas restricciones deben ser las primeras en crearse y rellenarse, y las últimas en borrarse.
Por dicha razón, nuestra creación de tablas comienza con *Bibus* y *Ruta*, la cual referencia a la anterior. Análogamente, creamos las tablas *Conductor* y *Asigna_Conductor*, la cual referencia a la anterior y servirá como tabla auxiliar referenciando a su vez a *Bibus*. Posteriormente, seguimos con la creación de la tabla *Municipio*, que referencia a *Ruta*, y la tabla *Biblioteca*, la cual referencia a *Municipio*. De forma independiente, creamos las tablas *Libro, Edición* y *Ejemplar,* las cuales referencian a la anterior consecutivamente. Seguidamente, creamos la tabla *Usuario* y *Préstamo,* tabla la cual referencia a *Usuario* al mismo tiempo que a *Municipio* y *Ejemplar* conectando con el conjunto de las tablas independientes definidas anteriormente. Finalmente, creamos las tablas *Comentario* que referencia a *Usuario y Préstamo*, y por último, la tabla *Sanción* que referencia a *Usuario* y *Préstamo*.

En cuánto a los problemas surgidos, los analizamos paso a paso, teniendo en cuenta las diferentes tablas que creamos:

- Bibus: En esta tabla almacenamos todos los autobuses utilizados en el sistema. Para ello, hemos escogido como atributos un identificador del propio bibus, la última y la próxima revisión. Debido a que en la tabla obsoleta fsdb.busstops contiene todos los registros de todas las paradas de cada bibus se ha tenido que reducir masivamente para solo contener la información identificadora de cada bibus, sin repetición, teniendo así una tabla de bibuses identificados unívocamente a través de su matrícula. Para ello, utilizamos las cláusulas WHERE y GROUP BY, asegurando que no se repetía ningún bibus.

- Ruta: En esta tabla completamos información acerca de las paradas realizadas por cada bibus. Así pues, contiene los atributos id_ruta, fec_ruta, parada y id_bibus.

- Conductor: En esta tabla almacenamos la información de cada conductor, para su posterior asignación a un bibus. Los atributos necesarios para ello son los denombre_cond, pasaporte, telefono, email, fec_ini, fec_fin. Aquí encontramos un problema similar al definido en Bibus pues un conductor está definido en más de una ocasión en la tabla obsoleta por lo que para almacenarlo sin repetición hemos tenido que reducir masivamente la inserción, solucionándolo del mismo modo que con el problema de Bibus.

- Asigna_Conductor: Esta tabla la cual contiene la información id_asignación, pasaporte, id_bibus, fec_ruta. Dicha tabla surge como necesidad de resolver el problema de relacionar la relación Conductor con Bibus, problema solucionado a través de la creación de esta tabla auxiliar, la cual referencia a Bibus y a Conductor al mismo tiempo. Por otra parte, nos encontramos con un problema respecto al identificador de asignación, pues este no está definido en la tabla de inserción fsdb.busstops, por lo tanto se ha creado un identificador único para cada asignación a través de una concatenación de un string "ASG" y un valor numérico para cada fila.

- Municipio: En esta tabla almacenamos todos los Municipios registrados en la tabla fsdb.busstop donde tienen parada los bibuses. Para ello, almacenamos la siguiente información como atributos: nombre_muni, provincia, poblacion, hay_biblio, id_ruta, fec_ruta y parada. Como problema encontrado es que algunos municipios pueden no tener una referencia válida a una ruta, para solucionarlo, se permite la creación de municipios sin id_ruta en caso de ser necesario.

- Biblioteca: A través de esta tabla almacenamos todas las Bibliotecas registradas a través de los siguientes atributos CIF, muni, provincia, direccion, nombre_bib, fec_fund, email y teléfono. Los problemas que nos hemos encontrado con esta relación son los siguientes: Por una parte, en la tabla de inserción fsdb.busstop tenemos una fila por cada parada del bibus, es decir, no todas las paradas hay biblioteca física, por lo tanto, para solucionar dicho problema hemos almacenado las Bibliotecas de forma única (sin repetición) tal que el atributo HAS_LIBRARY = 'Y'. Por otra parte, para registrar dichas bibliotecas de forma unívoca se ha tenido que crear un identificador CIF (no existente previamente) a través de la concatenación de un string "LIB" y un valor numérico para cada fila.

- Libro: En esta tabla almacenamos todos los libros y su información relevante como titulo, autor, pais, lengua_ori, fec_publi, titulo_alt*, tema*, nota*, premios*, otros_aut* y mencion_aut*. Destacar que en esta tabla, se almacena una edición por libro, en caso de existir más de una, así como no diferenciamos entre ejemplares, siendo así la representación más general de los libros en nuestra BBDD. Uno de los problemas que nos hemos encontrado es que los libros pueden estar repetidos en la tabla fsdb.acervus, para solucionarlo hemos utilizado GROUP BY title, main_author y hemos filtrado valores nulos. Otro de los problemas que nos hemos encontrado, el cual nos ha dificultado en gran medida la programación sql ha sido que al insertar los datos desde fsdb.acervus a nuestra tabla Libro, nos daba un error de almacenamiento pues no había memoria suficiente en dicho momento.

- Edición: En esta tabla almacenamos todos las ediciones y su información relevante como ISBN, id_libro, autor, lengua, ot_lengua, edic, editorial, extension, serie, dep_legal, lugar_p, dimens, caract, material*, notas*, id_bib y url. Destacar que en esta tabla, se almacenan los libros en función de su edición, es decir, es una representación más específica que la tabla Libro. Uno de los problemas encontrados es el de duplicidad en ISBNs, para solucionarlo utilizamos ROW_NUMBER() para seleccionar la edición más reciente.

- Ejemplar: En esta tabla almacenamos todos los ejemplares y su información relevante como sign, id_edicion, estado y coment_bibus. Destacar que en esta tabla, se almacenan todos los libros, con sus distintas ediciones y ejemplares, es así la representación más específica de los libros. Nos hemos encontrado con un problema debido a referencias inexistentes a Edición, para solucionarlo hemos filtrado valores nulos en signature y isbn.

- Usuario: A través de la implementación de esta tabla, identificamos todos los diferentes usuarios que han realizado algún préstamo almacenado en la tabla fsdb.loans, guardando dicha información en los siguientes atributos: carnet, nombre, apel1, apel2*, pasaporte, fec_nac, muni, direccion, email* y teléfono. El principal problema que hemos encontrado en la implementación de esta relación es la de datos inconsistentes o incompletos. Para solucionarlo, hemos utilizado MAX() para garantizar la inserción de la información más completa.

- Préstamo: En dicha tabla guardamos los registros de los préstamos a través de la siguiente información: id_prestamo, id_usuario, fec_pr, fec_dev, nombre_muni, provincia y ejemplar. Uno de los grandes problemas que nos hemos encontrado con la implementación de esta relación es debido a una mala interpretación inicial del atributo SIGNATURE en la tabla fdsb.loan, el cual lo habíamos tratado como identificador del préstamo. Una vez entendido que este atributo identifica un ejemplar y no un préstamo, ha aparecido un nuevo problema, el de identificar los préstamos de forma unívoca. Para ello hemos creado un identificador a través de la concatenación de un string “PRE” y una secuencia numérica.

- Comentario: En esta tabla almacenamos los comentarios que realizan los usuarios respecto a un libro (proveniente de un préstamo). Para ello, hemos incluído la siguiente información para registrar todos los comentarios: id_usuario, id_prestamo, fec_coment, contenido, likes y dislikes. El principal problema que nos hemos encontrado ha sido el de encontrar comentarios sin contenido o sin un usuario adecuado. Para solucionarlo, hemos filtrado valores nulos en post y user_id.

- Sanción: En esta tabla almacenamos un registro de sanciones de aquellos usuarios que hayan tardado más de 14 días en devolver un préstamo. Para la identificación de cada sanción se han registrado los atributos id_sancion, id_usuario, id_prestamo, dias_sancion, fec_pr, fec_dev y sancion_activa. El principal problema que nos hemos encontrado ha sido el de la aparición de errores al calcular los días correspondientes de sanción a través de la manipulación de fecha de devolución y fecha de préstamo pues estas variables estaban en un tipo de datos CHAR y no DATE. Para solucionarlo hemos debido de castear a través de las funciones TO_DATE o TO_NUM, según fuera conveniente.

Por último, como ya hemos comentado, la práctica ha sido mucho más complicada debido a un error constante de almacenamiento, ese error nos impedía crear las tablas y hacer los inserts correspondientes, no hemos encontrado una solución clara a este problema, y nos ha impedido trabajar fluidamente en la terminal de sqlplus.