Del Diseño Intuitivo al Diseño Conceptual de BDR

- Continuación de la clase anterior.
- Dependencias funcionales, normalización y DER.

Antes de comenzar.. hablemos de los datos que no existen

Supongamos la tabla ALQUILER que representa los alquileres de libros que hacen los socios de una biblioteca:

ALQUILER(Socio,Fecha,Libro,FechaDevolucion)

Asumiendo que Fecha es de tipo fecha-hora y que representa la fecha-hora en que el socio se llevó el libro
Asumiendo que Libro es algo unívoco que representa a un libro en particular de la biblioteca.
Entonces, cada vez que un socio se lleva un libro, el bibliotecario debera insertar un tupla en ALQUILER para registrar ese hecho, el problema es que **no hay dato posible para FechaDevolucion**, en esos casos, se dice que **FechaDevolucion es un campo/atributo no obligatorio** y cuando no tenemos un valor posible usamos el valor **NULL**

Socio	Fecha	Libro	FechaDevolucion
Juan	10/08/26 15:11	981-1234	15/08/26 08:21
Ana	15/08/26 10:12	973-4785	NULL
Pedro	15/08/26 11:23	178-85741	NULL
Juan	16/08/26 10:31	851-74158	NULL

Ana, Pedro, Juan tienen libros en su poder que aún no han devuelto.
FechaDevolucion siempre comienza con valor NULL y cuando alguien devuelve un libro, se cambia NULL por la fecha-hora de devolución que corresponda.

NULL or NOT NULL … esa es la cuestión..

- Las claves primarias **PK** y claves candidatas **CC no pueden contener valores Nulos**
- Las claves extranjeras **FK pueden contener valores Nulos**, esto tiene implicancias muy fuertes en el Modelo Relacional:

”Toda factura **debe** ser de un cliente determinado” → FK no nula de FACTURA hacia CLIENTE

FACTURA (0,N) – CLIENTE (**1**,1)

en terminos relacionales, la tabla quedaría asi:

FACTURA(Numero,Fecha,Monto,IdCliente)

PK(Numero)

FK(IdCliente,CLIENTE)

”Toda factura **puede** ser de un cliente o no (consumidor final)” → FK nula de FACTURA hacia CLIENTE

FACTURA (0,N) – CLIENTE (**0**,1)

en terminos relacionales, la tabla quedaría así:

FACTURA(Numero,Fecha,Monto,**IdCliente NULL**)

PK(Numero)

FK(IdCliente,CLIENTE)

- Un diseño con tablas con muchos campos no obligatorios, puede ser indicio de un mal diseño (ej: software enlatado, no a medida)
- Cuando en el diseño nos equivocamos y **ponemos a un campo optativo como obligatorio**, es tremendo para **el usuario** porque va a estar **inventando datos toda su vida**!
- Cuando en el diseño nos equivocamos y **ponemos a un campo obligatorio como optativo**, podemos violar la integridad de la BD porque **el usuario podría obviar o eliminar un dato que es importante** para el UoD
- El NULL es un valor, existe, pero su existencia indica ausencia de valor.

¿Por qué normalizar?

- Algunos datos se repiten innecesariamente → REDUNDANCIA.
- La REDUNDANCIA genera INCONSISTENCIA y anomalías (de inserción, de actualización, de borrado).

La INCONSISTENCIA genera dudas, confusión y falta de confianza en nuestro trabajo

Tener INCONSISTENCIAS al comienzo de la implementación de un nuevo sistema puede ser muy peligroso

Dependencia funcional

- Si conozco CodAlumno (PK), conozco Nombre y Carrera.
- CodAlumno → Nombre, Carrera.
- La clave determina otros datos.

Ejemplo PRODUCTO

- PRODUCTO(CodigoProducto, Descripcion, Precio)
  PK(CodigoProducto)
- CodigoProducto → Descripcion, Precio.

1FN

- Cada celda contiene un único valor.
- Evitar listas dentro de una columna.
- CLIENTE(IdCliente, Nombre, **Telefonos**)
  PK(idCliente)

1FN corregida

- CLIENTE(IdCliente, Nombre)
  PK(IdCliente)
- TELEFONO(IdCliente, Telefono)
  PK(IdCliente,Telefono)
  FK(IdCliente,CLIENTE)

2FN

- La PK compuesta determina completamente los atributos.
- INSCRIPCION(Alumno,Materia,NomAlumno,NomMateria,Nota)
  PK(Alumno,Materia)

Alumno,Materia → NomAlumno, NomMateria,Nota

Pero en realiadad lo anterior es incorrecto, las DF's son:
Alumno → NomAlumno
Materia → NomMateria
Alumno,Materia → Nota

2FN corregida

- ALUMNO(Alumno, NomAlumno)
  PK(Alumno)
- MATERIA(Materia, NomMateria)
  PK(Materia)
- INSCRIPCION(Alumno, Materia, Nota)
  PK(Alumno,Materia)
  FK(Alumno,ALUMNO)
  FK(Materia,MATERIA)

3FN

- Evitar dependencias transitivas.
- ALUMNO(CodAlumno,Nombre,CodCarrera,NomCarrera)
  PK(CodAlumno)

  esto es redundante → puede generar incosistencias y ademas tiene anomalías de actualización

Análisis de las DF's de esta relación:
CodAlumno → Nombre, CodCarrera
CodCarrera → NomCarrera

3FN corregida

- CARRERA(CodCarrera,NomCarrera)
  PK(CodCarrera)
- ALUMNO(CodAlumno,Nombre,CodCarrera)
  PK(CodAlumno)
  FK(CodCarrera,CARRERA)

Resuelve los problemas de redundancia, por lo tanto, los de inconsistencia y las anomalias.

BCNF

• Todo determinante relevante debe ser clave.

Universo de discurso (UoD) = “Cada profesor dicta una única materia y cada materia se dicta en una única aula”

DICTA(Profesor,Materia,Aula)

..pero, **¿cual es la clave de DICTA?**

Si fuese Profesor → Materia, Aula podria haber distintos profesores que dicten la misma materia e incluso en el mismo aula

Si fuese Materia → una materia no podría tener otro profesor que la dicte

Si fuese Aula → en un aula solo podría dictarse una materia, un profesor podria dictar n materias

Si fuese Profesor,Materia → un profesor podria dictar n materias

Si fuese Profesor,Aula → un profesor podria dictar n materias en distintas aulas

Si fuese Profesor,Materia,Aula → un profesor podria dictar n materias y una materia podria darse en n aulas

..

Analicemos las DF’s:

- Profesor → Materia
- Materia → Aula.

BCNF corregida

- MATERIA(Materia,Aula)
  PK(Materia)
- DICTA(Profesor,Materia)
  PK(Profesor)
  FK(Materia,MATERIA)

MATERIA y DICTA ambas estan BCNF

Dependencias multivaluadas

- Profesor habla distintos idiomas y le gustan distintos hobbies **y ambas son relaciones independientes**.
- No existe relación entre ambos conjuntos (idiomas y hobbies).

Ejemplo 4FN

- PROFESOR | Idioma | Hobby.
- Se generan combinaciones artificiales y redundantes.

Ejemplo: supongamos que un profesor habla Inglés y Portugués y le gusta la Fotografía y el Fútbol

Analizamos las DFM's: (dependencias funcionales multivaluadas)
Profesor -» Hobby
Profesor -» Idioma

Profesor	Hobby	Idioma
Perez	Fútbol	Inglés
Perez	Fútbol	Portugués
Perez	Fotografía	Inglés
Perez	Fotografía	Portugués

4FN corregida

- PROFESOR_IDIOMA(Profesor,Idioma)
  PK(Profesor,Idioma)
- PROFESOR_HOBBY(Profesor,Hobby)
  PK(Profesor,Hobby)

También podrían existir las tablas de profesor, hobby e idioma, en tal caso:

- PROFESOR(Profesor,Nombre, …)
  PK(Profesor)
- HOBBY(Hobby,...)
  PK(Hobby)
- IDIOMA(Idioma,...)
  PK(Idioma)
- PROFESOR_IDIOMA(Profesor,Idioma)
  PK(Profesor,Idioma)
  FK(Profesor,PROFESOR)
  FK(Idioma,IDIOMA)
- PROFESOR_HOBBY(Profesor,Hobby)
  PK(Profesor,Hobby)
  FK(Profesor,PROFESOR)
  FK(Hobby,HOBBY)

Resumen de formas normales

- 1FN: valores atómicos.
- 2FN: depende de toda la clave.
- 3FN: sin dependencias transitivas.
- BCNF: todo determinante debe ser clave
- 4FN: resolver las DFM’s independientes

Modelo Relacional vs DER

Ver simbología DER en introducción del TP4 https://www.grch.com.ar/docs/bd/materia/11077/tp4.pdf

- Tabla ↔ Tipo Entidad (rectángulo con linea simple o doble).
- Columna ↔ Atributo (Ovalo).
- PK ↔ Identificador (Ovalo marcado como clave)
- CC ↔ Identificador (Ovalo marcado como clave débil, no se pueden representar bien)
- FK ↔ Relación (Rombo, también puede un tipo entidad asociativa = tabla interrelación)

Nomenclatura TP4

- Entidades en MAYUSCULAS y singular (con caracteres simples, sin guiones intermedios).
- Atributos capitalizados.
- Cardinalidades mínimas y máximas (MIN,MAX).
- Entidad asociativa para resolver N:N.

Dos caminos posibles para obtener un Modelo Lógico Relacional

- UoD → Diseño intuitivo → Análisis de DF’s → Normalización.
- UoD → DER → Modelo Relacional “sin validar” → Análisis de DF’s → Normalización.

Del Diseño a la Implementación

- Modelo Relacional validado y depurado.
- Seleccionar SGBD relacional.
- Diseño físico acorde con los requerimientos, hardware y software disponible
- Definir Estrategia de implementación de atributos derivados
- Definir tipo de implementación A o B
- Definir tipos de datos y dominios.
- Crear tablas en el SGBD.
- Implementar PK, CC, FK, CK.
- Crear vistas
- Implementar reglas de integridad semánticas (Disparadores)
- Implementar transacciones, procedimientos
- Probar la BD con datos “buenos” y “malos”
- Si todo ok → BD Ok → solo resta desarrollar y probar la aplicación contra esta BD

(lo sombreado son temas de 11078 BD II, 11278 BDD)

Idea final

- Normalizar no es un fin, pero es una herramienta muy importante para facilitar el diseño de BDR
- Permite reducir redundancia y mejorar consistencia.
- Se puede limitar la normalización para limitar la “granularidad” de la BD y aumentar su eficiencia → redundancia controlada