Resumen Bases de Datos 1

Clase 1

Modelo de datos

- Provee una notación para describir los datos
- Se define a partir de
  - Estructura de los datos
  - Restricciones sobre los datos
  - Operaciones con los datos (optativo)
- Conjunto de conceptos que pueden usarse para describir la estructura de una base de datos

Tipos de Modelos de Datos

- Modelos lógicos
  - Basado en objetos
    - Modelo de Entidades y Relaciones
    - Modelo Basado en Objetos
  - Basados en registros
    - Modelo Relacional
- Modelos Físicos

Modelo de Entidades y Relaciones

Se define a partir de:

- Estructura
  - Entidad
    - Es una "cosa o concepto" que puede ser identificada y distinguible de otra "cosa o concepto".
  - Relación
    - Es una asociación de entidades.
    - Ejemplos: Juan con dni 1234567 es_dueño_de un auto modelo 2015 cuya patente es PRI.
  - Atributo
    - Representa información acerca de una entidad o una relación.
    - Ejemplos: nombre, dni, modelo, patente.
- Restricciones
  - Cardinalidad
    - Determina el número de veces en el que puede participar una entidad en una relación.
    - Indica dependencia (importancia de la cardinalidad mínima).

        - total o de existencia: participación obligatoria (al menos uno).
        - parcial: participación no obligatoria (puede ser cero).
        - puede ser:
            - Uno a uno.
            - Uno a muchos
            - Muchos a muchos
- Identificador o clave
    - Restricción de unicidad del valor del atributo
    - Sirven para identificar de manera única a una entidad
    - Toda entidad posee al menos una posible clave o identificador
        - Puede ser:
            - Simple
            - Compuesto
- Grado
    - Representa el número máximo de veces que una entidad puede estar relacionada con otra.
    - Ejemplos:
        - 1,N (grado N)
        - 1,1 (grado 1)
- Acerca de los nombres
    - No se pueden repetir los nombres de los atributos en una misma entidad ni en una misma relación.
    - No se pueden repetir nombres ni para entidades, ni para relacionales, ni para ninguna de ellas

Definición

Conjunto de entidades:
Es un conjunto de entidades del mismo tipo
Ejemplos:
El conjunto de todas las personas que poseen un nombre y tienen un dni puede llamarse PERSONA
El conjunto de todos los autos que poseen información del modelo y de la patente puede llamarse AUTO

Definición

Dominio de un atributo:
Conjunto de valores que puede tomar un atributo en particular
Ejemplo: nombre puede ser una cadena de máximo 50 letras
del abecedario

Definición

Conjunto de relaciones
Es un conjunto de relaciones del mismo tipo
Ejemplo:
ES_DUEÑO_DE es un conjunto de relaciones entre las entidades PERSONA Y AUTO

Important

Los términos entidad y conjunto de entidades serán intercambiables, haciendo abuso del vocabulario
Los términos relación y conjunto de relaciones serán intercambiables, haciendo abuso del vocabulario
Tener en cuenta que toda entidad debe, al menos, tener un atributo

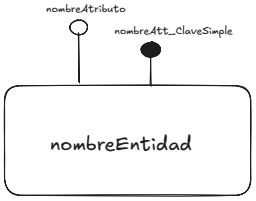
Notación Gráfica

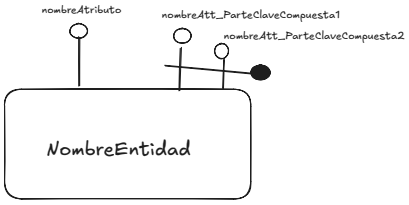
- Diagrama de entidades y relaciones
  - Representación gráfica de la estructura de los datos
  - Entidad

  - Relación

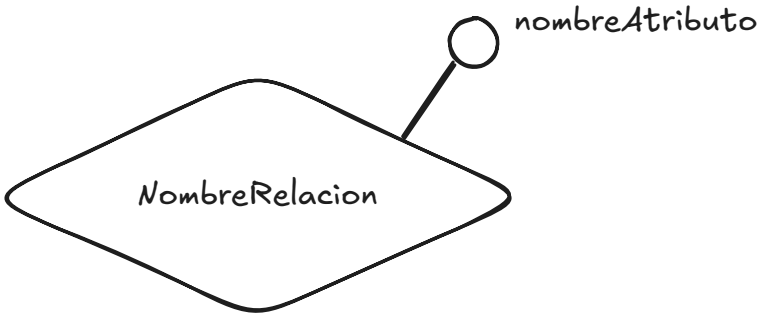
  - Atributo

  - Cardinalidad
  - (cardMin,cardMax)
  - Notación de atributos descriptores e identificadores simples y compuestos en una entidad.

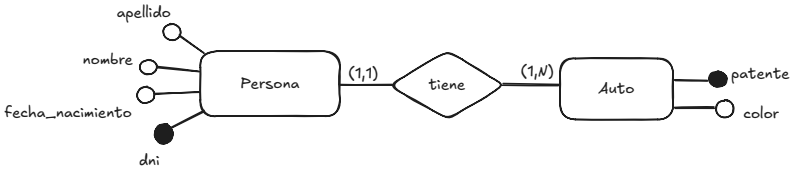




  - Notación de atributo en la relación.



Ejemplo de un diagrama de Entidades y relaciones.



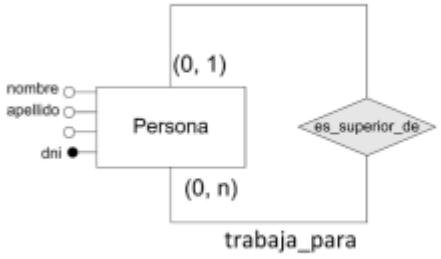
¿Cómo se lee la restricción de cardinalidad?
Una persona tiene al menos un auto y a lo sumo n
Y un auto es poseído por una única persona

Rol de una entidad en una relación

Indica la función que tiene la entidad en la relación.
Ejemplo:

- tutor_de
  Juan con dni 123456 es tutor_de Maria cuyo dni es 234567.
  Esta última, tiene el rol de tutelada_por

Ejemplo de un diagrama de Entidades y Relaciones –Rol-

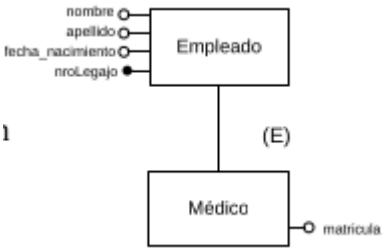


¿Cómo se lee la restricción de cardinalidad?
Una persona es superior de cero o muchas otras personas
Una persona trabaja para a lo sumo una persona

Especialización

Es el resultado de tomar un subconjunto de entidades de un nivel para formar un conjunto de entidades de nivel más bajo.
Ejemplo:

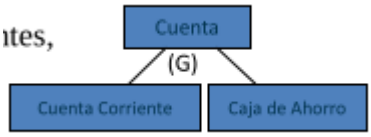
- Tenemos empleados de un hospital. De los médicos nos interesa su matrícula. Puede haber empleados que no son médicos.



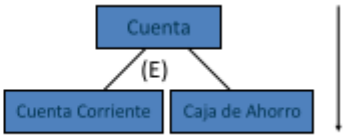
Generalización

Es el resultado de tomar uno o más conjuntos de entidades (de nivel más bajo) y producir un conjunto de entidades de un nivel más alto.
Ejemplo:

- Distintos tipos de cuenta: cajas de ahorro y cuentas corrientes, pero ambas son consideradas cuentas.



Mecanismos de abastracción



Generalización: No hay otro tipo de cuentas
Especialización: Podría haber otra caja de ahorro especial

Agregación

Para entenderlo mejor, supongamos el siguiente ejemplo:

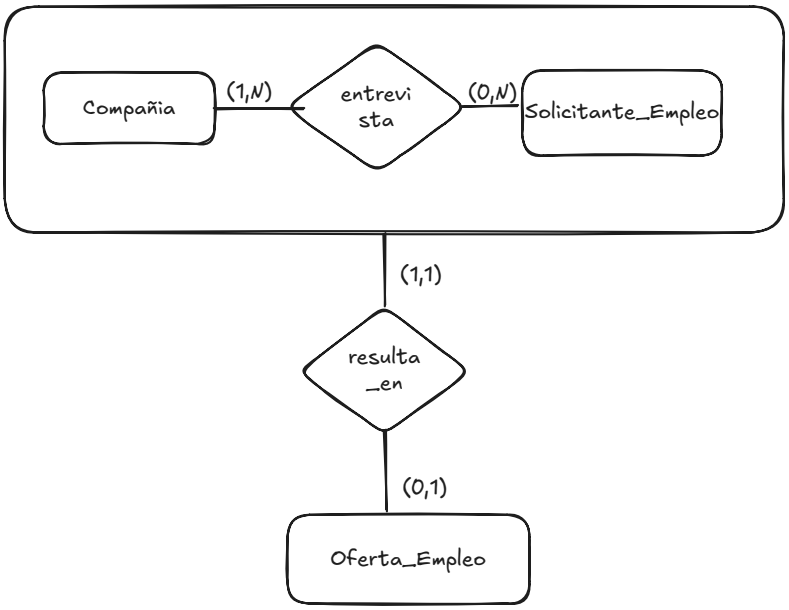
- se guardan entrevistas de solicitantes de empleo a varias compañías

- Supongamos, además que algunas entrevistas resultan en ofertas de empleo, pero otras no.
- Un problema del modelo de entidades y relaciones es que:
  - No es posible expresar relaciones entre relaciones existentes

Definicion

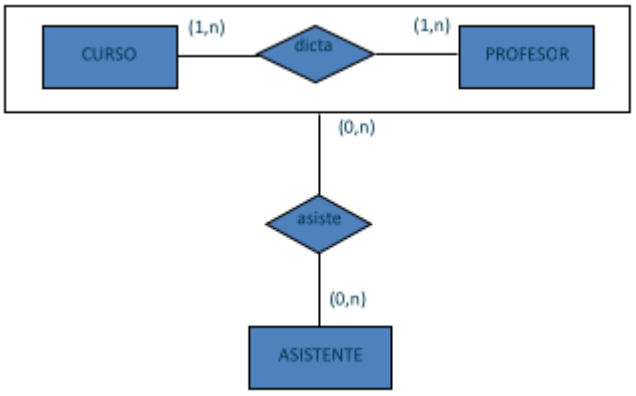
Agregación: Es un mecanismo de abstracción en el cual una relación binaria (junto a las dos entidades relacionadas) se trata como entidad de alto nivel
Nota: la cardinalidad máxima para cada entidad de la relación siempre es mayor a 1.

Volviendo al ejemplo anterior, se soluciona de la siguiente forma:



Otro ejemplo:

- Un profesor puede dictar uno o varios cursos. Una vez asignado un profesor a un curso es posible que se registren asistentes a dicha asignación.



Modelo Basado en Registros

Modelo de Relacional

Se define a partir de:

Estructura

- Relación
  - Representa los datos como tablas bidimensionales llamadas relaciones
  - Ejemplo: Persona

- Atributo
  - El nombre de cada columna de la relación o tabla.
- Esquema
  - Está formado por el nombre de una relación y su conjunto de atributos - Ejemplo: Persona(dni, edad, nombre)
  - Nota: los atributos de un esquema son un conjunto y no una lista, por lo tanto, no hay un orden físico.
- Tupla
  - Son las filas de una relación (excepto sus encabezados).
  - Posee un solo componente para cada atributo de la relación
  - Ejemplo (123456, 54, Juan) es una tupla con tres componentes de la relación Persona

Restricciones

- Clave de una relación
  - Un conjunto de atributos conforma una clave en la relación cuando a dicho conjunto no se le permite tomar dos valores iguales en todos los atributos de la clave
  - Ejemplo: Persona(dni, edad, nombre)
- Dominio de un atributo
  - Cada componente de cada tupla debe ser atómica, es decir, debe ser un tipo elemental (no puede ser una lista, un registro, etc)
- Acerca de nombres
  - Unicidad en nombres de esquemas, relaciones y atributos dentro de un esquema

Transformación 1 a 1 de Modelos

Desde Modelo de Entidades y Relaciones al Modelo Relacional

- Los modelos de datos son independientes entre si
- Puedo crear un Modelo Relacional sin previamente haber creado un Modelo de Entidades y Relaciones
- En este tipo de transformación TODAS las entidades y Relaciones se transforman en un Esquema de Relación

Cómo convertir del modelo de entidades y relaciones (E/R) al modelo relacional.

- Convertir cada conjunto de entidades en una relación (con igual nombre) con el mismo conjunto de atributos.
- Convertir cada relación del modelo de entidades y relaciones en una relación (del modelo relacional), de igual nombre
  - Para cada entidad involucrada en la relación, se toma el o los atributos claves como parte del esquema de la relación (del modelo relacional).
  - Si la relación (del modelo de entidades y relaciones) posee atributos, éstos también forman parte del esquema de la relación.
  - Si una entidad está involucrada más de una vez en una relación, con diferentes roles, se renombrará el atributo para evitar nombres duplicados, adoptando el nombre del rol de la

entidad en la relación

Nota: Las reglas anteriores cubren la mayoría de los casos para convertir de un modelo a otro. Otras reglas particulares serán vistas a continuación

- Entidad

CUENTA(**númeroCuenta**, saldo)

- Relaciones
  - (asumiendo que la entidad CLIENTE posee al atributo numeroCliente como clave, mientras que CUENTA al atributo numeroCuenta)

tiene(**numeroCliente**, numeroCuenta)

o

tiene(numeroCliente, **numeroCuenta**)

tiene(numeroCliente, **numeroCuenta**)

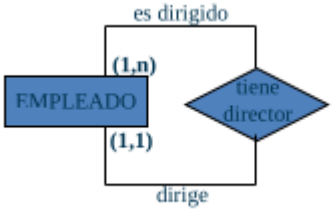
- Entidades y relaciones

PROFESOR(**codigoProfesor**, nombre, título)

CURSO(**codigoCurso**, título, tema)

DICTA(**codigoProfesor, codigoCurso**, fecha)

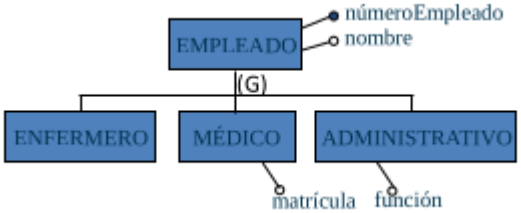
- Rol



TIENE_DIRECTOR(**númeroEmpleado,** númeroDirector)

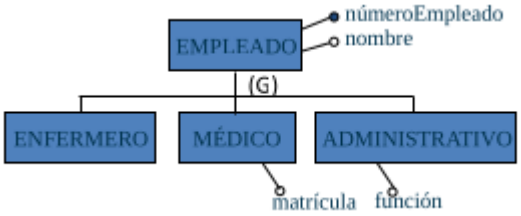
EMPLEADO(**númeroEmpleado**, nombre)

- Generalización (Tres estrategias)
  - Una tabla para el conjunto de entidades de nivel más alto



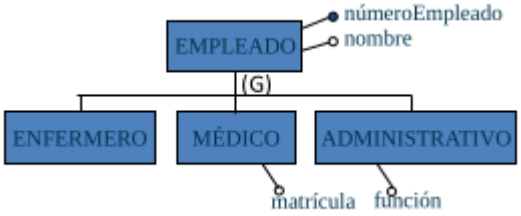
EMPLEADO (**númeroEmpleado**, nombre, tipoEmpleado, matrícula, función)

- Una tabla para cada conjunto de entidades de nivel más bajo



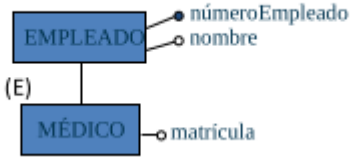
  ENFERMERO(**númeroEmpleado**, nombre)
  MÉDICO(**númeroEmpleado**, nombre, matrícula)
  ADMINISTRATIVO(**númeroEmpleado**, nombre, función)

- Una tabla para el conjunto de entidades de nivel más alto, y una tabla para cada conjunto de entidades del nivel más bajo



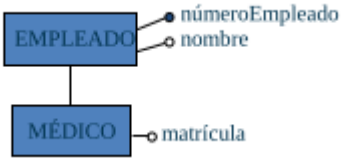
  EMPLEADO(**númeroEmpleado**, nombre)
  ENFERMERO(**númeroEmpleado**)
  MÉDICO(**númeroEmpleado**, matrícula)
  ADMINISTRATIVO(**númeroEmpleado**, función)

- Especialización (dos estrategias)
  - Una tabla para el conjunto de entidades de nivel más alto



    EMPLEADO(**númeroEmpleado**, nombre, tipoEmpleado, matricula)
    Esta opción tiene la principal desventaja de tener que manejar un tipo para distinguir que tipo de empleado es, en este ejemplo.

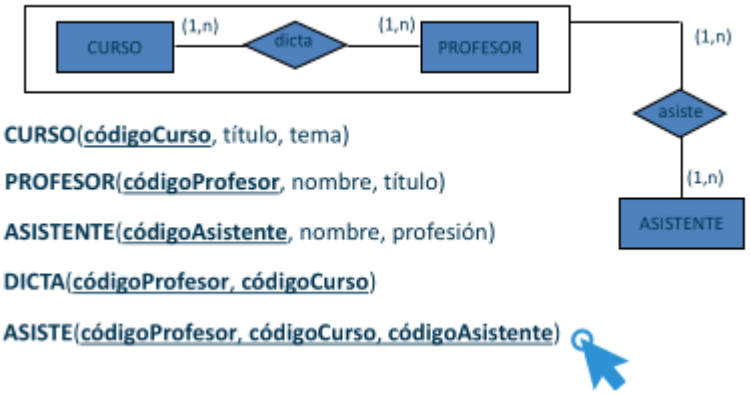
  - Una tabla para el conjunto de entidades de nivel más alto, y una tabla para cada conjunto de entidades del nivel más bajo.



    EMPLEADO(**númeroEmpleado**, nombre)
    MÉDICO(**númeroEmpleado**, matrícula)

- Agregación
  - Todas las Entidades y Relaciones involucradas, marcando claves según cardinalidad

- Foco en los atributos de la relación con la agregación



Clase 2

Motivación

Uno de los principales principios del diseño de bases de datos es la Normalización.

- Organiza los datos siguiendo reglas
  - Minimiza redundancia
  - Reduce anomalías
- Puede mejorar la mantenibilidad y según el caso, la performance
  Un mal diseño puede implicar refactorizar la base de datos
- Requiere mucho tiempo y expertise
- Deuda técnica
  El diseño de un esquema de bases de datos es uno de los factores más importantes de los que depende el éxito de la base de datos relacional.

La normalización es un paso clave en el diseño de una base de datos

- Toma una **relación grande** como entrada y la **descompone** en relaciones más pequeñas las cuales están libres de redundancia de datos y otras anomalías como la inserción/eliminación
  - La descomposición se realiza siguiendo reglas/pasos
- Puede ser manual o automática
  Normalización se aplica a bases de datos relacionales
- El modelo relacional, sigue siendo el modelo dominante en la industria.

Síntesis

- El éxito de una base de datos relacional depende del diseño de su esquema
- Existen procesos manuales y automáticos
- Con impacto en la industria
- Tema relevante y actual
- El modelo relacional domina el mercado

Teoría de Diseño de Bases de Datos Relacionales

Conceptos generales

- Anomalía
- Dependencia Funcional
- Dependencia Funcional Trivial

Anomalía

- Problema que surge a raíz del diseño de una relación

Anomalía de redundancia

- Información que se repite innecesariamente en diferentes tuplas

Anomalía de actualización

- Se puede actualizar el valor de una tupla, sin actualizar los de otras tuplas

Anomalías de inserción

Insertar valores en ciertos atributos de una relación y no otros me produce valores nulos.

Anomalías de borrado

Borrar ciertos valores de una tupla, puede llevarme a perder la información de la tupla completa.

Dependencia Funcional

- Es una restricción entre subconjuntos de atributos de una relación.

Si dos tuplas (t1 y t2) de una relación R , coinciden en todos los atributos A1, A2,...,An; entonces DEBEN también coincidir en los atributos B1, B2,..,Bm. Para toda tupla de R.

- Esto se escribe: A1, A2,...,An -> B1, B2,..,Bm
- Y se lee: A1, A2,...,An "determina funcionalmente a" B1,B2,...,Bm
  Cuando en R se cumple una d, estamos indicando una restricción sobre toda la relación R y no sólo sobre algunas tuplas de R.

Dicho de otra manera:

- Una dependencia funcional de la forma X -> Y se cumple en R si:
  - Para todos los pares de tuplas t1 y t2 de la relación, cuando se cumple que t1[x]=t2[x]
  - Entonces se cumple t1[y]=t2[y].

Ejemplo 1:

- Dada la relación
  - PERSONA(dni, nombre, edad, fechaNacimiento)
- Y valga en persona la
  - df: dni -> nombre, edad, fechaNac

- La df enunciada, indica que, si dos tuplas t1 y t2 de la relación PERSONA tienen el mismo valor en el atributo dni, deben necesariamente tener los mismos valores en los atributos nombre, edad y fechaNac.

Ejemplo 2:

Dada la relación: PERSONA(dni, nombre, edad, fechaNac, nroLegajo)
Donde

- Una persona posee un único número de legajo asignado
- Un número de legajo pertenece a una sola persona
- Se pueden enunciar las siguientes dfs
  - df1) dni -> nombre, edad, fechaNac, nroLegajo
  - df2) nroLegajo -> nombre, edad, fechaNac, dni

Dependencia Funcional Trivial

- Caso especial para una dependencia funcional

Es una df de la forma:
A1, A2,...,An -> B1, B2,..,Bm
Tal que:
{B1, B2,..,Bm} ⊆ {A1, A2,...,An }

Ejemplo:

Dada la relación:

- CONTRATADO(nroContradado, dni, nombrePersona, inicioActividad)
  Donde valen las siguientes dependencias funcionales:
- df1) dni -> nombrePersona
- df2) nroContratado, dni -> inicioActividad
  Algunas dependencias funcionales triviales válidas en CONTRATADO son:
- dft1) dni-> dni
- dft2) nroContratado, dni -> nroContratado

Clase 3

Clave de una relación

Los atributos {A1,A2,...,An} son la clave de una relación R si cumplen:

- {A1,A2,...,An} determinan funcionalmente a todos los restantes atributos de la relación R
- No existe un subconjunto de {A1,A2,...,An} que determine funcionalmente a todos los restantes atributos de la relación R
  - Esto implica que la clave es un conjunto minimal

Ejemplo:
PERSONA(dni, nombre, edad, fechaNac)

- df1: dni->nombre,edad,fechaNac
- Clave: {dni}

Clave candidata de una relación

En caso de existir dos o más subconjuntos de atributos {A1, A2, ...,AN}, {B1,B2,...,Bk}, ..., {N1,N2,...,Nm} en una relación R tales que:

- {A1, A2, ...,AN} determinan funcionalmente a todos los atributos restantes de la relación R
- {B1,B2,...,Bk}, ... Y {N1,N2,...,Nm} también por si mismos determinan al resto de los atributos de R
- No existe subconjunto de {A1, A2, ...,AN} o {B1,B2,...,Bk}, ... o,{N1,N2,...,Nm} que determine funcionalmente a todos los atributos de R
- Entonces {A1, A2, ...,AN}, {B1,B2,...,Bk}, ...,{N1,N2,...,Nm} son CLAVES CANDIDATAS para la relación R

Clave candidata / de una relación:

Ejemplo 1:

- Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo)
  Donde
  - Una persona posee un único número de legajo asignado
  - Un número de legajo pertenece a una sola persona
- Se pueden enunciar las siguientes dfs
  - df1) dni -> nombre, edad, fechaNac, nroLegajo
  - df2) nroLegajo -> nombre, edad, fechaNac, dni
  ¿Clave o claves candidatas?
  Clave candidata 1 (cc1): {dni}
  Clave candidata 2 (cc2): {nroLegajo}

Ejemplo 2:

- Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo, carrera)
  - Donde
  - Una persona puede cursar diversas carreras
  - Nombre indica como se llama la persona
  - Una persona posee un único número de legajo asignado para cada carrera que cursa
  - Un número de legajo pertenece a una sola persona de una carrera
  - df1) dni -> nombre, edad, fechaNac
  - df2) nroLegajo, carrera -> dni
  - df3) dni, carrera -> nroLegajo
  ¿Clave o claves candidatas?
  Clave candidata 1 (cc1): {nroLegajo, carrera }
  Clave candidata 2 (cc2): {dni, carrera }

Super clave de una relación

Los atributos {A1,A2,...,An} son superclave de una relación R si cumplen:

- {A1,A2,...,An} determinan funcionalmente los restantes atributos de la relación R
- Notar que:
  - Una clave está contenida en una superclave
  - Una superclave no necesariamente es minimal (como lo es la clave por la segunda condición de su definición)

Ejemplo:

- PERSONA(dni, nombre, edad, fechaNacimiento)
  df1: dni->nombre,edad,fechaNac
- superclave: {dni, nombre}

Axiomas de Armstrong

- Permiten inferir nuevas dependencias funcionales dado un conjunto base que resultó evidente
- Aplicándolos hallo un conjunto completo y seguro donde todas las dependencias funcionales halladas son correctas
- Al generar todas las dependencias funcionales algunas son triviales

Axiomas Básicos

- Reflexión
- Aumento
- Transitividad
  Axiomas que se deducen a partir de los básicos
- Unión
- Descomposición
- Pseudotransitividad

Reflexión

X es un conjunto de atributos e Y ⊆X entonces X -> Y
Demostración:
Si Y⊆X y existen dos tuplas diferentes de R tales que t1[x]=t2[x] por definición de dependencia funcional t1[y]=t2[y]

Aumento

Si X ->Y ;
Z es un conjunto de atributos,
entonces
Z,X -> Z,Y
Demostración:
Asumamos que X->Y vale pero X,Z->Y,Z no vale
Si X->Y entonces cada vez que

1. t1[x]=t2[x] implica
2. t1[y]=t2[y]
   Por otro lado, cada vez que
3. t1[x,z]=t2[x,z] implica
4. t1[y,z] <> t2[y,z]
   De 1) y 3)se deduce t1[z]=t2[z]
   De 2) y 4) se deduce que t1[y,z]=t2[y,z]

Transitividad

Si X ->Y ;
Y ->Z,
entonces X -> Z
Demostración:

1. X ->Y
2. Y ->Z
   t1[x]=t2[x] implica por 1)
   t1[y]=t2[y] implica por 2)
   t1[z]=t2[z] entonces
   X -> Z

Unión

Si X ->Y ;
X ->Z,
entonces X -> Y,Z
Demostración:

1. X ->Y
2. X ->Z
   Si X ->Y , por aumentación vale que X ->XY
   Si X ->Z , por aumentación vale que X,Y ->Y,Z
   Luego por transitividad, X ->Y,Z

Descomposición

Si X ->Y,Z
entonces X -> Y , X -> Z
Demostración:
X ->Y,Z
por reflexividad vale que Y,Z ->Y
Luego, por transitividad X->Y
Por reflexividad también vale que Y,Z ->Z
Luego por transitividad, también vale que X->Z

Pseudotransitividad

Si X ->Y; Y,Z->W entonces X, Z-> W
Demostración:
X ->Y
por aumento vale que X,Z ->Y,Z
Por otro lado, se sabe qué Y,Z->W
Luego por transitividad, vale que X,Z->W

Clausura de un conjunto de atributos

X^+

- Sea F un conjunto de dependencias funcionales sobre un esquema R y,
- Sea X un subconjunto de R

La clausura del conjunto de atributos X respecto de F, se denota X^+ y es el conjunto de atributos de A tal que la dependencia X->A puede deducirse a partir de F, por los axiomas de Armstrong. Es decir, X^+, son todos los atributos determinados por X en R.

Algoritmo para hallar X^+

```
Result:= X
While (hay cambios en result)
do
For (cada dependencia
funcional Y->Z en F ) do
if (Y ⊆ result) then
result := result ∪ Z
```

¿Como funciona?

Ejemplo:
Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo, carrera)
Donde

- Una persona puede cursar diversas carreras
- Nombre indica como se llama la persona
- Una persona posee un único número de legajo asignado para cada carrera que cursa
- Un número de legajo pertenece a una sola persona de una carrera
  - df1) dni -> nombre, edad, fechaNac
  - df2) nroLegajo, carrera -> dni
  - df3) dni, carrera -> nroLegajo
  Clave candidata 1 (cc1): {nroLegajo, carrera }
  Clave candidata 2 (cc2): {dni, carrera}
  Por ejemplo, podríamos preguntarnos: ¿Es cierto que a partir de los atributos de cc1, puedo recuperar los atributos restantes de PERSONA?
- Para ello podemos ejecutar el algoritmo de X+ instanciándolo con la información de PERSONA
  Hallar (nroLegajo, carrera)^+

Result= (nroLegajo, carrera)
Paso 1) Tomamos la dep. fun: dni -> nombre, edad, fechaNac, {dni} no está incluido en result, no
agrego nada a result => (nroLegajo, carrera)
Paso 2) Tomamos la dep. fun. nroLegajo, carrera -> dni , {nroLegajo, carrera} está incluido en result,
agrego {dni} a result => (nroLegajo, carrera, dni)
Paso 3) Tomamos la dep. fun. dni, carrera -> nroLegajo, {dni, carrera} está incluido en result, agrego
{nroLegajo} a result => (nroLegajo, carrera, dni)
Como ya recorrí todas las dependencias funcionales y result cambió vuelvo a iterar
Paso 1) Tomamos la dep. fun. dni -> nombre, edad, fechaNac, {dni} está incluido en result, agrego
{nombre, edad, fechaNac} a result => (nroLegajo, carrera, dni , nombre, edad, fechaNac)

Important

- De la misma manera podríamos haber probado con la otra clave candidata
- Recordar que este algoritmo NO asegura que el conjunto de atributos de partida sea mínimo

---

Hasta ahora vimos, para una relación R

- Cómo hallar la o las claves candidatas
  - Y como usar el algoritmo de la clausura de atributos para corroborar que a partir de un subconjunto de atributos de R se puede recuperar al resto de los atributos de la relación (aunque éste no asegura que dicho subconjunto sea mínimo)
- Cómo hallar dependencias funcionales
  - Y su conjunto completo mediante los Axiomas de Armstrong
  A continuación:
- Considerando las dependencias funcionales y las claves candidatas, veremos conceptos necesarios para realizar un proceso que permite generar relaciones que cumplan ciertas condiciones de un buen diseño (quitando anomalías) con el fin de normalizar esquemas

Normalización de esquemas de relación.

Algunos conceptos

¿Cómo generar relaciones que cumplan ciertas condiciones de un buen diseño?

Descomposición o particionamiento de un esquema

- Es una forma aceptada de eliminar las anomalías de una relación

- Consiste en separar los atributos de una relación en dos nuevas relaciones (bajo ciertos criterios)
- Al particionar, no se debe perder
  - Información
  - Dependencias funcionales

Descomposición de un esquema

Dado un esquema R donde vale una dependencia funcional X->Y R se descompone/particiona en

- R1(X, Y)
- R2(R-Y)
  Al descomponer, no se debe perder:
- Información
- Dependencias funcionales
  - Validación simple
  - Validación formal mediante un algoritmo

Pérdida de información

Si a un esquema R, se lo particiona en dos subesquemas R1 y R2
entonces, se debe cumplir alguna de las siguientes condiciones:

- R1 ∩ R2 clave en el esquema R1
  o bien,
- R1 ∩ R2 clave en el esquema R2

Pérdida de dependencias funcionales

Al particionar, se debe verificar que cada una de las dependencias funcionales que valían en el esquema R, sigan valiendo en alguna de las particiones Ri.
Cuando se chequean las dependencias funcionales pueden ocurrir dos cosas:

- los atributos de la dependencia funcional original quedaron todos incluidos en alguna de las particiones generadas.
  => Validación simple
- Los atributos de la dependencia funcional original quedaron distribuidos en más de una partición
  => Validación formal mediante un algoritmo

Formas normales

- Criterio para determinar grado de vulnerabilidad a inconsistencias y anomalías
- Existen diferentes formas normales
- Al lograr aplicar una mayor forma normal se logrará menor vulnerabilidad

Algunas formas normales

- [1]Primera Forma Normal (1FN)
- [2]Segunda Forma Normal (2FN)
- [2-1]Tercera Forma Normal (3FN)
- [2-2]Forma Normal de Boyce y Codd

Forma Normal de Boyce y Codd (BCNF)

- Conocida por su acrónimo en inglés de BCNF
- Particionar para llevar un esquema a esta FN, asegura que:
  - las anomalías dejan de estar (sólo puede quedar redundancia),
  - que no se pierda información y,
  - en algunos casos, asegura que no se pierdan dependencias funcionales

Un esquema de relación está en BCNF sí, siempre que una dependencia funcional de la forma X->A es válida en R, entonces se cumple que:

- X es superclave de R (mínima)
  o bien
- X->A es una dependencia funcional trivial

- Hasta ahora vimos que
  - Las formas normales se usan para sacar anomalías
  - A mayor forma normal, menos vulnerabilidad
- A continuación
  - Intentaremos llevar, en principio, un esquema a la Forma Normal de Boyce y Codd para normalizarlo

Para normalizar un esquema (en principio) y propiciar un buen diseño, vamos a valernos de:

- las dependencias funcionales del esquema
- las claves candidatas
- la definición de BCNF
- la descomposición o particionamiento del esquema

Proceso de normalización

Cómo llevar un esquema R a BCNF (cuando se puede)
Hallar dependencias funcionales y claves candidatas

1. Analizar si en el esquema R existe alguna dependencia funcional que lleva al esquema a no cumplir con la definición de BCNF
   1. SI existe tal dependencia funcional, particionar el esquema en dos nuevos esquemas Ri, Ri+1, contemplando la dependencia funcional en cuestión.
      Analizar las 2 particiones generadas preguntándose:

1) Se pierde información?
1) NO, entonces sigo a 1.1.2
2) SI. La partición es errónea. Re analizar
2) Se pierden Dependencias funcionales?
1) NO, entonces sigo a 1.1.3
2) Si. Veremos este caso en breve
3) Determinar en qué forma normal esta Ri, Ri+1, si no están en BCNF, reiniciar desde 1, sino pasar a 1.2

2. Si NO existe, el esquema está en BCNF

Clase 4

Forma Normal de Boyce y Codd (BCNF)

Un esquema de relación está en BCNF sí, siempre que una dependencia funcional de la forma X->A es válida en R, entonces se cumple que:

- X es superclave de R
  o bien
- X->A es una dependencia funcional trivial

En general, al llevar un esquema a BCNF

Particionar llevando a un esquema en BCNF

- Las anomalías dejan de estar (sólo puede quedar redundancia)
- Que no se pierda información
- En algunos casos, asegura que no se pierdan dependencias funcionales

---

Para normalizar un esquema (en principio) y propiciar un buen diseño, vamos a valernos de:

- las dependencias funcionales del esquema
- las claves candidatas • la definición de BCNF
- la descomposición o particionamiento del esquema

Iniciando el proceso de normalización de un esquema a partir de un ejemplo

Considerando que el esquema se encuentra en 1FN.

Proceso de normalización

FIESTAS (#salon, dirección, capacidad, fecha_fiesta, nombre_invitado, mesa_invitado, servicio_contratado, dni_invitado)
Donde:

- Hay diversos salones, pero de cada salón se conoce la dirección y la capacidad máxima
- Una fiesta se brinda en un salón y en una fecha determinada. Notar que en un mismo salón se realizan diferentes fiestas en diferentes fechas
- Para cada fiesta se conocen cada uno de los servicios contratados, los que pueden ser varios
- Para cada fiesta se saben quiénes son los invitados y donde se debe sentar cada uno de ellos
  Hallar la o las claves candidatas y las dependencias funcionales

Dependencias Funcionales

1. #salon → dirección, capacidad
2. #salon, fecha_fiesta, dni_invitado → mesa_invitado
3. dni_invitado-> nombre_invitado

Siguiente paso

- Analizar si en el esquema R existe alguna dependencia funcional que lleva al esquema a no cumplir con la definición de BCNF

Fiestas cumple con la def de BCNF?Forma Normal de Boyce y Codd (BCNF)

Por ejemplo, analizo la df1:

1. #salon -> dirección, capacidad
   {#salon} no es superclave del esquema FIESTAS
   Fiestas no cumple con la definición de BCNF

En caso de no estar en BCNF

- SI existe tal dependencia funcional, particionar el esquema en dos nuevos esquemas Ri , Ri+1 , contemplando la dependencia funcional en cuestión.

FIESTAS (#salon, dirección, capacidad, fecha_fiesta, nombre_invitado, mesa_invitado, servicio_contratado, dni_invitado)
cc: (#salon, fecha_fiesta, dni_invitado, servicio_contratado)

1. ?#salon → dirección, capacidad
   1. [3]Dado que FIESTAS NO CUMPLE CON LA DEFINICION DE BCNF, descompongo/particiono FIESTAS considerando la dependencia funcional 1.
   - F1(**#salon** , dirección, capacidad)
   - F2 = Fiestas – {dirección, capacidad} Es decir,
     - F2 tiene los siguientes atributos:
       - F2 (**#salon, fecha_fiesta**, nombre_invitado, mesa_invitado, **servicio_contratado, dni_invitado**)

Analizar las 2 particiones generadas preguntándose:

- [4]Se pierde información?
  - NO, entonces, verifico si se pierden DF

- SI. La partición es errónea. Re analizar

Con el particionamiento propuesto:

F1 ∩ F2 es clave en el esquema {#salon}

Entonces, no se perdió información

> **Important**
>
> Quedé en diapositiva 22

---

1. El esquema no debe tener atributos polivalentes o compuestos ↩
2. Se determinan a partir de las dependencias funcionales ↩ ↩ ↩
3. Un esquema R donde vale una dependencia funcional X->Y se descompone como R1(X, Y) R2(R-Y) ↩
4. R1 ∩ R2 clave en el esquema R1 o R1 ∩ R2 clave en el esquema R2 ↩