PRESENTACIÓN BBDD

TEMARIO

TEMA 1. Introducción a las BBDD.
TEMA 2. Modelo y Álgebra Relacional.
TEMA 3. Diseño conceptual y paso a tablas. ← *A esta altura se hace el primer parcial*
TEMA 4. Administración de objetos a la BD. Gestión de usuarios y permisos sobre objetos de la BDs.
TEMA 5. SQL.
TEMA 6. Acceso programático. JDBC.

EVALUACIÓN

EVALUACIÓN CONTINUA:
Dos exámenes parciales, uno el día 01/05
y otro el día 09/06

Modelo Relacional	→	10-15%
Modelo E/R y Paso a tablas	→	20-25%
Seguridad	→	10%
SQL	→	20%
Acceso Programático	→	15%

Práctica 1 - Diseño BBDD, diagrama E/R	→	10%
Práctica 2 - Implementación BDs	→	5%
Práctica 3 - JDBC y SQL	→	15%

EVALUACIÓN POR PRUEBA FINAL:
Único examen el día 09/06

Modelo Relacional	→	15%
Modelo E/R y Paso a tablas	→	25%
Seguridad	→	15%
SQL	→	25%
Acceso Programático	→	20%

EVALUACIÓN EXTRAORDINARIA:
Examen el día 08/07

Se seguirán los mismos criterios de evaluación que haya solicitado cada alumno durante el curso (ya sea por evaluación continua o por evaluación final).

La nota mínima de cada una de los parciales, así como de las prácticas es de 5/10 en la evaluación continua.
La práctica 1 se entrega en torno a la semana 6, la práctica 2 se entrega en torno a la semana 9 del curso y la práctica 3 se entrega en torno a la semana 15 del curso.

Por evaluación final la nota mínima de cada una de las partes (o bloques) es de 5/10.

TEMA 1: INTRODUCCIÓN A LAS BBDD

1. ¿POR QUÉ SON IMPORTANTES LAS BBDD?

Las BBDD son importantes ya que almacenan datos, que son el soporte para la extracción del conocimiento.

2. ¿QUÉ ES UNA BD?

Una **BD** es un conjunto estructurado de datos coherentes y consistentes. Por ello es importante que la información se estructure antes de almacenarse.

Ejemplos de datos estructurados: nombre, apellidos, fecha de nacimiento de una persona; color predominante, tipo o tamaño de una imagen.

Ejemplos de datos NO estructurados: una persona o una imagen.

3. ¿QUÉ ES UN SISTEMA DE GESTIÓN DE BBDD (SGBD)?

Un **SGBD** es un software que permite crear una BD, acceder a su contenido e insertar, borrar, modificar y consultar los datos que contiene.

Sus principales **propiedades** son: (1) la independencia de los datos; (2) el acceso eficiente a ellos; (3) su integridad y seguridad; (4) su administración y acceso concurrente; y (5) la recuperación de los mismos.

Las **ventajas** que presenta son: (1) reutilización de datos y programas; (2) control de redundancia; (3) consistencia; (4) independencia de los datos y aplicaciones (*modelo conceptual*); e (5) integridad y seguridad.

4. NIVELES DE ABSTRACCIÓN

a. *NIVEL EXTERIOR O DE VISTAS* → es el más cercano al usuario final. Está escrita mediante esquemas externos. Para cada uno de ellos tenemos:
   i. Define la porción de BD que interesa a un usuario concreto
   ii. Pueden existir varias vistas de un mismo esquema conceptual
   iii. Varias vistas se pueden superponer entre sí
   iv. Está escrito usando un *modelo de datos conceptual o lógico*

b. *NIVEL CONCEPTUAL* → da soporte a la independencia entre los datos y las aplicaciones. Incluye restricciones semánticas sobre los datos y proporciona una visión completa de todos los requerimientos y elementos de interés para la organización. Estos esquemas definen la estructura lógica de toda la BD (entidades, tipos de datos, relaciones y restricciones - tales como integridad y seguridad, entre otras -), ocultando los detalles físicos.

c. *NIVEL FÍSICO O INTERNO* → es el más cercano a la máquina. Define la estructura física de almacenamiento de toda la BD (tipos de registros almacenados, secuencia física, estructuras de almacenamiento, de acceso, etc.). Es dependiente del gestor de BBDD que se use.

5. ¿QUÉ ES UN MODELO DE DATOS?

Los modelos de datos proporcionan mecanismos de abstracción que permiten la representación de aquella parcela del mundo real cuyos datos nos interesa registrar (Universo del Discurso). Existe una gran colección de herramientas para describir los datos, sus relaciones, las restricciones y la semántica (modelo relacional, modelo de datos entidad/relación, modelo basado en objetos o modelo de datos semiestructurados - XML -). También existen otros modelos más antiguos como los modelos de redes de trabajo o los modelos jerárquicos.

6. ¿EN QUÉ CONSISTE EL DISEÑO DE LAS BBDD?

El diseño de las BBDD es un proceso que consiste en decidir qué datos vamos a almacenar, de qué entidades queremos guardar información, para cada entidad qué propiedades (o atributos) vamos a guardar y qué relaciones existen entre las entidades. **→ PRÁCTICA 1.**

TEMA 2: MODELO RELACIONAL

1. ELEMENTOS DEL MODELO

Los elementos básicos o ítems elementales de la información son los **atributos.** El conjunto de los atributos de un determinado problema forma el **universo** de dicho problema. Mientras que el conjunto de **tablas** (cada una con un nombre identificatorio distinto) son las relaciones entre los atributos. Cada atributo puede tomar sus valores en un conjunto llamado **dominio**. Los atributos NO pueden tener el mismo nombre, pero sí el mismo dominio.

Por ejemplo:
En el universo T={A,B,C,D} cuyos atributos son A, B, C, D y los dominios de estos atributos son:

Dom(A)={a,b,c}
Dom(B)={1,2,3,4}
Dom(C)=conjunto de los números naturales.
Dom(D)={a,b,c}

Otro concepto importante es el de las **anomalías**, que son inconsistencias en las *actualizaciones* (donde algunas no se pueden llegar a ejecutar) y en los *borrados* (que podrían ocasionar pérdidas de información relevante). Para que esto no ocurra, debemos partir del universo de atributos y del conjunto de dependencias funcionales existentes entre ellos.

Por ejemplo:

NOTAS:

Alumno	Asignatura	Nota	Ciudad	Provincia
Pepe	BBDD	7.5	Pozuelo	Madrid
María	Ing. Soft	5	Majadahonda	Madrid

Ahora mismo si desconocemos el valor de cualquiera de los atributos primarios no podríamos insertar una nueva fila (o tuplas)
El conjunto de atributos de este problema es {Alumno, Asignatura, Nota, Ciudad, Provincia}
El conjunto de relaciones es {AlumnoAsignatura → Nota, Alumno → Ciudad, Ciudad → Provincia}
Una posible clave (que no tiene por qué ser única) es {Alumno, Asignatura}.

ANOMALÍA DE INSERCIÓN:

Si queremos introducir a Juan que vive en Leganés (provincia de Madrid). Como no contiene la información de ninguna asignatura, no podemos introducir esta tupla en la tabla. Para solucionar este problema dividimos en dos tablas diferentes nuestro problema.

NOTAS2

Alumno	Asignatura	Nota
Pepe	BBDD	7.5
María	Ing Soft	5

DOMICILIO

Alumno	Ciudad	Provincia
Pepe	Pozuelo	Madrid
María	Majadahonda	Madrid
Juan	Leganés	Madrid

ANOMALÍA DE BORRADO

En el caso de que ahora queramos borrar a Pepe de nuestra BDs, entonces, al ser el único que vive en Pozuelo, perdemos esta información. Para solucionarlo, lo que podemos hacer es dividir la información en dos tablas diferentes y conservar de esta forma la información.

NOTAS2

Alumno	Asignatura	Nota
María	Ing Soft	5

DOMICILIO2

Alumno	Ciudad
Maria	Majadahonda
Juan	Leganés

DOMICILIO3

Ciudad	Provincia
Pozuelo	Madrid
Majadahonda	Madrid
Leganés	Madrid

2. DEPENDENCIAS FUNCIONALES

Son la expresión de implicaciones elementales que se dan en el problema que se está modelando. Por ejemplo, X→ Y es cierta si a cada valor de X corresponde un valor ÚNICO de Y. El conjunto de todas estas relaciones (notado por L) más el conjunto de atributos (notado por T) es lo que se conoce como **esquema de la relación.** Se nota como R(T,L)

Este esquema se puede representar mediante una relación definida en T con las restricciones L. A esto se le conoce como **relación universal**. Esta relación no se implementa, si no sus proyecciones.

Además, las dependencias funcionales verifican unas propiedades conocidas como las **propiedades de Armstrong**:

a) **Reflexividad**: X → X, ∀X
   Ejemplo: Si tenemos A, entonces A → A
b) **Aumentatividad**: Si X → Y y X' ⊇X, entonces X' → Y
   Ejemplo: Si tenemos A → C y AB ⊇A, entonces AB → C
c) **Proyectividad**: Si X → Y y Y' ⊆Y, entonces X → Y'
   Ejemplo: Si tenemos A → BC y C ⊆BC, entonces A → C
d) **Aditividad**: Si X → Y y Z → V, entonces X∪Z → Y∪V
   Ejemplo: Si tenemos A → C y B → D, entonces AB → CD
e) **Transitividad**: Si X → Y y Y→ Z, entonces X→ Z
   Ejemplo: Si tenemos A → B y B → C, entonces A → C

Una clave K de R(T,L) es un subconjunto de T tal que K → T y ningún subconjunto de K → T.

Por ejemplo: (en el ejemplo anterior)
Alumno,Asignatura → Nota
Alumno → Ciudad
Ciudad → Provincia

¿Cuál es la clave en este caso?
Solo con ALUMNO deducimos tanto CIUDAD como PROVINCIA, pero no podemos llegar a deducir ni ASIGNATURA ni NOTA.
Solo con ASIGNATURA no deduzco nada.
Solo con CIUDAD deduzco PROVINCIA pero nada más (ni ALUMNO, ni ASIGNATURA, ni NOTA)
Si cogemos como clave ALUMNO y ASIGNATURA ya puedo deducir todo mi universo T. Por lo que es la clave (y además es ÚNICA)

3. CIERRE DE UN CONJUNTO DE ATRIBUTOS RESPECTO DE UN CONJUNTO DE DEPENDENCIAS

Dado el esquema R(T,L) y un subconjunto de T, X⊂T. El cierre de X^+ es un superconjunto de X, $X^+ \supseteq$X tal que X → X^+ y no existe ningún X'⊇ X^+ tal que X^+→ X'. Dicho de otra forma, el cierre es el conjunto máximo que depende de K. Este cálculo se realiza mediante un proceso iterativo que concluye cuando dos iteraciones coinciden o se alcanza T.

Ejemplo 1:

T= {A,B,C,D,E,F}

L= {A → BC, C→ D, AC → E, AB → C, A → F}

¿Cuál es el cierre de ABE? → donde ABE es un conjunto de elementos (o atributos) pertenecientes a T

ABE^0= ABE	Buscamos dependencias cuyo implicante (lado izq.) sea subconjunto de ABE y añado su implicado (lado derecho)
ABE^1= ABCE	Usando la primera dependencia añado C al cierre de ABE
ABE^2= ABCDE	Usando la segunda dependencia añado D al cierre de ABE
= ABCDE =	Usando la tercera y la cuarta dependencia obtengo el mismo resultado que en el paso anterior, pero no hemos alcanzado T (cuando sí que podemos)*
ABE^+= ABCDEF	Usando la quinta dependencia añado F al cierre ABE, obteniendo T

Luego, el cierre de $ABE^+ = ABCDEF = T$

OJO: BUSCAD SIEMPRE TODOS LOS CAMBIOS POSIBLES!!!!

Ejemplo 2:

T= {A,B,C,D,E,F}

L= {A → BC, C→ D, AC → E, AB → C, F → A}

¿Cuál es el cierre de ABE?

ABE^0 =ABE

ABE^1 = ABCE (utilizando la primera dependencia)

ABE^2 = ABCDE (utilizando la segunda dependencia)

$ABE^3 = ABE^+$ =ABCDE (utilizando la tercera dependencia)

Como hemos generado dos iteraciones consecutivas iguales, entonces el cierre de $ABE^+ = ABCDE \subset T$

4. SIMPLIFICACIÓN DE UN CONJUNTO DE DEPENDENCIAS

Las dependencias funcionales verifican ciertas propiedades que se usan para transformar un conjunto L en otro equivalente L' sin redundancia. Esto se puede deber a que algún atributo del implicante *sobre* (es decir, se puede eliminar y la dependencia sigue siendo cierta) o a que una dependencia pueda ser deducida de otra (por lo que se puede eliminar).

Para el proceso de simplificación hay un **paso previo**: toda dependencia debe tener un implicado (parte derecha) simple (con un solo atributo).

Ejemplo:

L0:

1 A → BC
2 C → D
3 AC → E
4 AB → C
5 F → A

L1:

1. A → B
2. A → C
3. C → D
4. AC → E
5. AB → C
6. F → A

Comprobamos si sobra algún implicante (lado izq.). Para ello nos centramos en las dependencias que tienen más de un elemento en esta parte, es decir, las dependencias 4 y 5.

I) Comprobamos si quitando A de la dependencia 4, es decir, con C → E, entonces el cierre de C contiene E

C^0= C, C^+= CD. Como el cierre de C NO contiene a E, entonces no puedo suprimir A

Comprobamos si quitando C de la dependencia 4, es decir con A → E, entonces el cierre de A contiene E

A^0=A, A^1= AB, A^2=ABC, A^3=ABCD, A^+=ABCD. Como el cierre de A NO contiene a E, entonces no puedo suprimir C.

Luego la dependencia NO es redundante, por lo que no la podemos eliminar

II) Comprobamos si quitando B de la dependencia 5, es decir, con A → C, entonces el cierre de A contiene C

A^0=A, A^1= AB, A^2=ABC, A^3=ABCD, A^4=ABCDE, A^+=ABCDE. Como el cierre de A contiene C, entonces puedo eliminar esta dependencia ya que es redundante.*

* Otra forma de justificarlo es diciendo que como A → C es nuestra segunda dependencia, entonces al estar repetida podemos prescindir de ella ya que es redundante.

Luego nos queda:

L2:

1. A $\rightarrow$ B
2. A $\rightarrow$ C
3. C $\rightarrow$ D
4. AC $\rightarrow$ E
5. F $\rightarrow$ A

Finalmente nos queda estudiar si tenemos dependencias redundantes:

I) Si eliminamos 1 A $\rightarrow$ B, entonces el cierre de A debe contener B
A^0=A, A^1= AC, A^2=ACD, A^+= ACDE. Como el cierre de A no contiene B, entonces 1 NO es redundante.

II) Si eliminamos 2 A $\rightarrow$ C, entonces el cierre de A debe contener C
A^0=A, A^+= AB. Como el cierre de A no contiene C, entonces 2 NO es redundante.

III) Si eliminamos 3 C $\rightarrow$ D, entonces el cierre de C debe contener D
C^0=C, C^+= C. Como el cierre de C no contiene D, entonces 3 NO es redundante.

IV) Si eliminamos 4 AC $\rightarrow$ E, entonces el cierre de AC debe contener E
AC^0=AC, AC^1= ABC, AC^2=ABCD, AC^+=ABCD. Como el cierre de AC no contiene E, entonces 4 NO es redundante.

V) Si eliminamos 5 F $\rightarrow$ A, entonces el cierre de F debe contener A
F^0=F, F^+=F. Como el cierre de F no contiene A, entonces 5 NO es redundante.

Por lo que el conjunto de dependencias sin redundancia nos queda finalmente:

L+:

1. A $\rightarrow$ B
2. A $\rightarrow$ C
3. C $\rightarrow$ D
4. AC $\rightarrow$ E
5. F $\rightarrow$ A

5. CÁLCULO DE UNA CLAVE

Dado un esquema de relación R(T,L), denominamos **clave** a un subconjunto K de T que verifica las siguientes propiedades:

a) $K \subseteq T$
b) $K \rightarrow T$
c) No existe ningún subconjunto de K tal que $K \rightarrow T$ ⇐ Si este fuera el caso, entonces habríamos calculado una **SUPER CLAVE**

Introducimos todos los atributos que aparecen solo a la izquierda, siendo **I**.
Introducimos todos los atributos que aparecen solo a la derecha, siendo **D**.
Introducimos todos los atributos que aparecen tanto a la izquierda como a la derecha, siendo **ID**.
Introducimos todos los atributos que NO aparecen en ninguna dependencia, siendo **N**.
El **núcleo Z** de la clave es **I∪N.**

Calculamos el cierre del núcleo **Z**. Si es T, el núcleo es clave única. En caso contrario vamos añadiendo atributos de ID hasta que el cierre sea T. Si hemos añadido más de un atributo, debemos comprobar que el conjunto obtenido es clave y no superclave.

Ejemplo 1: (continuación)
Dado el ejercicio anterior donde T={A,B,C,D,E,F} y L={A→B, A→C, C→D, AC→E, F→A}
I = {F}
D = {B, D,E}
ID = {A,C}
N = {}
Z = I∪N = {F}
Calculamos el cierre del núcleo:
$Z^0 = F$, $Z^1 = AF$, $Z^2 = ABF$, $Z^3 = ABCF$, $Z^4 = ABCDF$, $Z^5 = ABCDEF$, $Z^+ = ABCDEF = T$
Entonces Z, es decir, F es clave y es única.

Ejemplo 2: (otro problema diferente, nada que ver con el anterior)
¿Y si tuviéramos ahora Z=AB, qué pasaría?
Si el cierre de AB fuera ABD que está contenido en T (pero no es T)
Entonces tendríamos que añadir símbolos de ID hasta obtener T.
Imaginamos que añadimos C a nuestra clave y obtenemos el cierre de ABC = ABCD que contiene a T pero sigue sin ser T. En este caso debemos añadir otro elemento de ID a la clave (por ejemplo E) y volver a calcular el cierre. Si con esto ya obtenemos T, entonces hay dos posibilidades:

a) Si calculamos el cierre de ABE y NO obtenemos T, entonces la clave es ABCE
b) Si calculamos el cierre de ABE y SI obtenemos T, entonces la clave es ABE (siendo ABCE una SUPER CLAVE de nuestro problema)

6. ALGORITMO DE ULLMAN

i) Eliminar la redundancia en L obteniendo en conjunto equivalente L^{+}
L={A→B, A→C, C→D, AC→E, F→A} del **Ejemplo 1** anterior.

ii) Agrupar en subesquemas los atributos de dependencias con igual implicante (lado izq.)
R1(A,B,C), R2(C,D), R3(A,C,E), R4(F,A)

iii) Si la clave calculada NO aparece en ningún subesquema, se añade R_0**(atributos de la clave)**
Como mi clave es F, y F está contenido en R4, entonces NO debo añadir R0

Por ejemplo, si mi clave hubiera sido BF, entonces como NO EXISTE ningún subesquema que contenga ambos atributos deberíamos añadir R0(B,F), ya que son los elementos que forman mi clave.

El conjunto de subesquemas deducidos es equivalente a R con mejores propiedades de cara al almacenamiento y manejo de los datos.

EXÁMENES MODELO RELACIONAL

JUNIO 2019

Dada la siguiente relación R(T,L), donde T = {P, S, D, A, Z, X, V, E} y
L = {P → SDA, D → Z, AZX → V, SV → E, PX → E}

Calcular el cierre de L sin redundancia, la clave y los subesquemas

TEMA 2: ÁLGEBRA RELACIONAL

1. LENGUAJES DE INTERROGACIÓN

Hay dos familias de lenguajes de interrogación:

- **Cálculo Relacional:** subconjunto del cálculo de predicados de Primer Orden sin símbolos de negación ni de función para evitar que se puedan generar respuestas infinitas.
- **Álgebra Relacional:** conjunto de operaciones que admiten como operandos relaciones y devuelven como resultado relaciones.

Ambos lenguajes son equivalentes y relacionalmente completos, es decir, cualquier información contenida en la BDs, o deducible a partir de ella, puede obtenerse usando solo Álgebra o Cálculo.

Mientras que el Cálculo es no procedimental y está en la base de las BBDD deductivas, el Álgebra es procedimental. Un posible ejemplo de lenguaje basado en el Álgebra podría ser SQL.

A continuación estudiaremos los operadores algebraicos que nos permiten ejecutar queries:

a) Básicos o primitivos:

i) *Unión*: deben estar definidas sobre los mismos atributos. Se constituye por todas las tuplas comunes y no comunes sin repetición.

R

A	B	C
3	5	8
7	5	2
1	6	9

S

A	B	C
4	0	2
1	6	9

R∪S

A	B	C

ii) *Diferencia*: deben estar definidas sobre los mismos atributos. Se constituye por todas las tuplas de R que no están en S. (Sería equivalente a la unión - intersección de los conjuntos)

R-S

A	B	C

iii) *Producto cartesiano*: es otra relación definida sobre la unión de las cabeceras de R y S cuyas tuplas se forman concatenando cada tupla de R con cada tupla de S en todas las formas posibles.

R

A	B	C
1	2	3
4	5	6
7	8	9

S

D	E
1	2
3	4

RxS

A	B	C	D	E

iv) *Selección*: conjunto de filas de R que verifican el criterio o condición expresado mediante la fórmula lógica *f*. Se expresa como $\sigma_f(R)$

$\sigma_{A>2\ \wedge B<6}(R)$

A	B	C

v) *Proyección*: es la relación definida sobre los atributos indicados, con eliminación de posibles filas duplicadas.

$\prod_{A}(R)$

A

b) Derivados:

i) *Intersección:* deben estar definidos sobre el mismo conjunto de atributos. Es el conjunto de filas comunes a ambas relaciones, con supresión de filas repetidas.

R

A	B	C
3	5	8
7	5	2
1	6	9

S

A	B	C
4	0	2
1	6	9

R∩S

A	B	C

ii) *Asociación:* selección según *f* del producto cartesiano de R y S donde la condición *f* debe referirse tanto a atributos de R como de S.

R

A	B	C
1	2	3
4	5	6
7	8	9

S

D	E
1	2
3	4

R∞ $_{A<D}$**S**

A	B	C	D	E

iii) *Unión natural o "join":* deben tener un descriptor común. Se seleccionan las filas con igual valor del descriptor común y se eliminan las columnas repetidas.

R

A	B	C
1	2	3
4	5	6
7	8	1

S

C	D
1	2
3	4

R∞S

A	B	C	D

iv) *División:*

R

A	B	C	D
1	3	1	5
3	9	5	2
1	2	5	2
3	9	1	5

S

C	D
5	2
1	5

R/S

A	B

2. TEOREMA DE DELOBEL Y CASEY

Si proyectamos R(T) en R1(T1) y R2(T2) de modo que $T1 \cup T2$ exista y $T1 \cap T2 \neq \emptyset$,
se tiene $R1 \infty R2 \supseteq R$ donde puede haber una posible inclusión de tuplas “extrañas”.
Pero si tenemos

(i) $T1 \cap T2 \rightarrow T1\text{-}T2 \in L^{+}$; ó

(ii) $T1 \cap T2 \rightarrow T2\text{-}T1 \in L^{+}$

entonces no se producen este tipo de tuplas “extrañas”.

Ejemplo:

Supongamos que {R1(A,B,C), R2(C,D), R3(D,E)} es el resultado de aplicar a R({A,B,C,D,E}, L) el algoritmo de Ullman, y sea la interrogación “**valores de C para los cuales E=0**” ¿contiene tuplas extrañas?

Debemos hacer un join de R2 con la selección de los elementos de R3 que cumplan E=0 y proyectar C de todo ello → $\Pi_{C}(R2 \infty\ \sigma_{E=0}(R3))$

Esto será correcto si se cumple el Tma de Delobel y Casey. Para ello debe suceder:

$R2 \cap R3 = \{D\}$
$R2 - R3 = \{C\}$
$R3 - R2 = \{E\}$

Luego, si en el cierre de L tenemos cualquiera de las siguientes dependencias, $D \rightarrow C$ ó $D \rightarrow E$, entonces podemos asegurar por el Tma de Delobel y Casey que NO existen tuplas “extrañas”.
En caso contrario no podemos afirmar (ni negar) nada.

EXÁMENES MODELO Y ÁLGEBRA RELACIONAL

JULIO 2016

Calcular un conjunto de L no redundante dada la relación universal R(T,L), donde
T = {B,C,E,F,T,M}
L = {E → FTM, E → BC, B → FM, BC → T}

Formular luego, usando los operadores del Álgebra Relacional la interrogación:
"*Valores de E para los que B = 5 y M = p*"

JUNIO 2020 - MATES + INFOR -

1) Preguntas breves de teoría del modelo relacional:

a) Qué es una clave foránea **→ No lo hemos visto**

b) Contesta verdadero o falso:
Los axiomas o propiedades de Armstrong para las relaciones funcionales son 5:
Reflexividad, Aumentatividad, Proyectividad, Aditividad y Transitividad.

c) Dado el siguiente modelo relacional:

RELACIÓN EMPLEADO

cod_empleado	nombre_completo	cod_proyecto
E093720	John Doe	V9834
E084902	Jane Smith	V9834
E092382	Juan Nadie	V9836
E828737	Luis Doe	V9836
E828737	Luis Doe	V9834

RELACIÓN PROYECTO

cod_proyecto	nombre_proyecto
V9834	Desarrollo Web
V9836	Data Mining

Escribe usando las operaciones de álgebra relacional la consulta que obtenga el nombre y código de los empleados que **no** trabajan en el proyecto denominado 'Data Mining'

2) Ejercicio de simplificación de dependencias funcionales:
Considerando el conjunto de atributos T y las siguientes dependencias funcionales L:

T = {A, B, C, D, E, F, G, H, I}

L:
1. $A \rightarrow CE$,
2. $BE \rightarrow D$,
3. $A \rightarrow BE$,
4. $BDH \rightarrow E$,
5. $AB \rightarrow CE$,
6. $E \rightarrow B$,
7. $D \rightarrow H$,
8. $BG \rightarrow F$

Descomponer la relación R(T,L) en un esquema en 3NF usando en el algoritmo de Ullman. Aplicar la simplificación de dependencias, con eliminación de redundancias, y cálculo del cierre y de la clave, sobre el conjunto de dependencias funcionales R(T,L), según lo visto en los ejemplos de clase.

JUNIO 2020

1. Si el *join* de dos relaciones no se planifica adecuadamente, el resultado puede incluir tuplas falsas:
   a. Verdadero
   b. Falso
   c. Una secuencia de operaciones de Álgebra puede no ser la adecuada para una interrogación en concreto, pero nunca va a devolver como respuesta valores que no estuviesen previamente en la Base.
   d. Para saber *a priori* si pueden darse tuplas falsas, basta con conocer la semántica del modelo concreto (dominios de los atributos).

2. En una consulta "SELECT FROM WHERE" de SQL:
   a. SELECT realiza lo mismo que el operador algebraico de selección
   b. SELECT realiza lo mismo que el operador algebraico de proyección
   c. El bloque completo "SELECT FROM WHERE", equivale a un operador algebraico de unión natural (JOIN)
   d. No puede establecerse ninguna equivalencia entre el SQL y el Álgebra

3. El Álgebra es un lenguaje relacionalmente completo porque:
   a. Las respuestas obtenidas son siempre correctas.
   b. Cualquier información presente en la Base puede obtenerse mediante una secuencia de operaciones sólo de Álgebra, pero pueden aparecer en la respuesta obtenida otros valores espúreos (no correctos); siendo responsabilidad del usuario formular la consulta de modo que no pueda darse esa posibilidad.
   c. Porque basta con formular el objetivo y la condición de búsqueda.
   d. Si una interrogación algebraica es sintácticamente correcta, el resultado será necesariamente correcto y completo

4. "SELECT A,B from R,S where C=7" equivale en Álgebra a:
   a. Seleccionar en R, S o ambas; las filas en las que C=7, efectuar la unión natural (*join*) de los conjuntos de tuplas así obtenidos, proyectar sobre A y B
   b. Efectuar la unión natural (*join*) de R y S; seleccionar las filas en las que C=7 y proyectar sobre A y B
   c. Las respuestas que comienzan con "Seleccionar en R, S..." y "Efectuar la unión natural..." son ambas correctas, pero la que comienza con "Seleccionar en R, S..." es más eficiente en términos de tiempo de ejecución, ya que se trabajará (en general) con relaciones de cardinalidad más baja
   d. No puede hablarse de equivalencia, porque el SQL es un lenguaje de alto nivel y el Álgebra es procedimental

5. El *join* es un operador derivado:
   a. Porque puede suprimirse y no pasa nada
   b. Porque equivale a una secuencia de producto cartesiano, selección y proyección
   c. Porque solo puede efectuarse si una de las relaciones contiene la clave
   d. El *join* es un operador básico o fundamental

6. Dado el esquema R(T,L) y un descriptor X (subconjunto de T), ¿cómo puedo saber si la dependencia $X \rightarrow Y$ (no perteneciente a L) es cierta?
   a. Viendo si existe en L alguna dependencia cuyo implicante sea X y cuyo implicado sea superconjunto de Y
   b. Viendo si en L hay alguna dependencia cuyo implicante sea X y cuyo implicado sea subconjunto de Y
   c. Viendo si existe en L alguna dependencia tal que la unión de implicante e implicado es superconjunto de Y
   d. Calculando el cierre de X respecto de L y comprobando si el segundo miembro de la dependencia bajo estudio está contenido en dicho cierre.

7. Con relación al conjunto de dependencias {AB → C, CD → A, AC → D}, ¿es cierta la dependencia BC → A?
   a. Sí
   b. No
   c. No se puede saber si no se conoce la semántica del esquema (contexto)
   d. Puede ser cierta o falsa, según contexto

8. Con relación al conjunto de dependencias {AB → C, CD → A, AC → D}, ¿es cierta la dependencia ABD → CD?
   a. Sí
   b. No
   c. Las dependencias funcionales son intrínsecas a los datos, así que sería necesario conocer su significado
   d. Habría que calcular el cierre del conjunto y ver si contiene esa dependencia

9. El núcleo de un esquema es:
   a. La intersección de todas sus claves
   b. La unión de todas sus claves
   c. La clave de menor longitud de entre todas las claves de ese esquema
   d. La intersección de todos los subesquemas que lo componen

10. El núcleo de un esquema puede ser el conjunto vacío:
    a. A veces
    b. Nunca
    c. Sólo si se da el caso de que haya tuplas inconsistentes con las dependencias especificadas en L
    d. Sólo en el caso de que no haya dependencias

11. Si el cierre del núcleo es el universo T, entonces el núcleo es la clave única del esquema:
    a. A veces
    b. Siempre
    c. En general, será una superclave
    d. Sólo se dará ese caso si el esquema no tiene dependencias

12. En el esquema {AB → C, CD → A, AC → D} el núcleo es B:
    a. Verdadero
    b. Falso, el núcleo es D
    c. Falso, el núcleo puede ser A o C, porque son los atributos que aparecen más veces en el lado izquierdo de alguna dependencia
    d. Falso, el núcleo es ADC

13. El algoritmo de Ullman aplicado al conjunto de dependencias no redundante {AB → C, CD → A, AC → D}, produce como resultado:
    a. R1(A,B,C); R2(C,D,A); R3(A,C,D)
    b. R1(A,B,C); R2(C,D,A), ya que R3 es una permutación de R2 y puede suprimirse y la clave está incluida en R1
    c. R1(A,B,C), R2(C,D,A), R0(A,B); para garantizar *lossless join* en todos los casos
    d. R1(A,B,C), R2(A,B,D)

14. El Algoritmo de Ullmann es *lossless join* porque:
    a. El número de subesquemas que constituyen el resultado es el menor posible
    b. Porque el *join* de todos los subesquemas que constituyen el resultado coincide siempre con R, sin introducir tuplas erróneas
    c. Sólo será *lossless join* si en todos los subesquemas se contiene alguna clave
    d. No puede afirmarse que lo sea siempre, porque a veces el *join* de todos los subesquemas resultantes es superconjunto de R

15. Si la clave no forma parte de alguno de los subesquemas que constituyen el resultado de un diseño realizado mediante el algoritmo de Ullmann, se añade un subesquema R0 definido sobre una clave para:
    a. Garantizar el *lossless join* del conjunto
    b. No se incluye nada: El resultado siempre es *lossless join*
    c. No basta: Debe incluirse un subesquema adicional por cada clave
    d. Es necesario conocer todas las claves.

16. El *join* permite reconstruir la relación universal a partir de sus proyecciones:
    a. Siempre
    b. Sólo si el conjunto de proyecciones es *lossless-join*
    c. Sólo si los join sucesivos se van planteando entre proyecciones con algún atributo común
    d. En cada caso, debe comprobarse antes de validar el diseño

17. La intersección es un operador derivado porque:
    a. Puede expresarse usando sólo el operador de diferencia
    b. Para poder realizar la intersección de dos relaciones, éstas tienen que ser compatibles para la unión
    c. El operador de intersección es básico porque procede de la Teoría algebraica de conjuntos
    d. Permite obtener lo que tienen en común dos relaciones.

18. El *join* de dos relaciones sólo es posible si:
   a. Ambas tienen cabeceras con un descriptor común.
   b. Ambas tienen el mismo valor en el descriptor común.
   c. El resultado de un join no puede ser el conjunto vacío (relación sin tuplas), así que para plantearlo es necesario conocer antes el contenido de las relaciones entre las que se va a realizar.
   d. Ambas tienen la misma cardinalidad.

19. Sólo puede plantearse el *join* entre relaciones que contengan (al menos una de ellas) alguna clave.
   a. Verdadero
   b. Depende de la semántica del problema
   c. Falso. En todo caso, si al menos una de ellas contiene una clave, el *join*, además de posible, es *lossless join*
   d. Es falso porque entonces habría que determinar todas las claves antes de formular cualquier interrogación a la Base

20. El *join* es un operador derivado porque puede expresarse en términos de:
   a. Selección, proyección, unión de conjuntos.
   b. Unión de conjuntos, proyección, intersección.
   c. Producto cartesiano, selección, proyección.
   d. El *join* es un operador básico.

21. Con relación al conjunto L de dependencias:
   $L = \{AB \rightarrow D, BD \rightarrow A, AD \rightarrow C\}$, ¿cuál es el cierre de AC?
   a. AC
   b. ACD
   c. No puede calcularse el cierre de un descriptor si dicho descriptor no es alguno de los implicantes del conjunto
   d. No puede calcularse dicho cierre si dicho descriptor no es subconjunto de alguno de los implicantes.

22. Con relación al conjunto de dependencias $\{AB \rightarrow D, BD \rightarrow A, AD \rightarrow C\}$ ¿es cierta $AC \rightarrow D$?
   a. Sí
   b. No
   c. No puede afirmarse si es cierta o falsa, al no ser AC ninguno de los implicantes del conjunto
   d. No puede realizarse el estudio, al no haber ningún implicante subconjunto de AC

23. El núcleo de un esquema es:
   a. Lo que queda en L después de eliminar redundancias
   b. La intersección de todas las claves
   c. La unión de todas las claves
   d. La intersección de todos los implicantes

24. En el esquema T={A,B, C, D}, L={AB->D, BD->A, AD->C}, una clave es AB
    a. Verdadero
    b. Falso
    c. AB no es la clave, pero es superconjunto del núcleo
    d. No puede ser clave al no existir ningún implicante subconjunto

25. En el esquema T={A, B, C, D}, L={AB → D, BD → A, AD → C}, BD es una clave
    a. Verdadero
    b. Falso
    c. BD no es clave, pero está contenido en la clave
    d. Es una superclave

26. El Algoritmo de Ullmann aplicado al esquema definido por {A,B,C,D} y el conjunto no redundante de dependencias {AB→D,BD→A,AD→C}, produce como resultado: R1(A,B,D), R2(A,D,C), R0(A,B)
    a. Verdadero
    b. Sobra R0 porque R1 contiene las dos claves.
    c. Debe añadirse además otro subesquema, R01(BD), definido sobre la segunda clave
    d. Debe añadirse un esquema sobre la segunda dependencia, R3(B,D,A)

27. Si el núcleo de un esquema es el conjunto vacío, esto quiere decir que:
    a. No podrán calcularse las claves.
    b. Que el conjunto de dependencias es el conjunto vacío.
    c. Que las claves son disjuntas.
    d. Está mal calculado: el núcleo nunca puede ser vacío

28. El núcleo de {AC → D, AD → B, BC → A}, es:
    a. ABD
    b. C
    c. El conjunto vacío
    d. ABC

29. Las claves del conjunto de dependencias {AC → D, AD → B, BC → A}, son:
    a. ABCD → Es superclave
    b. AC, BC
    c. AC, BC, CD → NO es clave
    d. ABD, BCD → ABD NO contiene el núcleo, por lo que NO puede ser clave

30. Si el cierre del descriptor X respecto del conjunto L de dependencias contiene todos los atributos, entonces:
    a. X es una clave
    b. X contiene todas las claves
    c. X contiene al menos una clave
    d. Ninguna del resto es cierta

JULIO 2020

1. Dos expresiones algebraicas son equivalentes si:
    a. Aplicadas a los mismos operandos, el tiempo de ejecución es el mismo
    b. Aplicadas a los mismos operandos, se obtiene el mismo resultado en ciertos casos
    c. Aplicadas a los mismos operandos, se obtienen siempre los mismos resultados
    d. Ninguna de las dos puede simplificarse más.

2. El Álgebra es un lenguaje relacionalmente completo porque:
    a. Cada operador equivale a otro del Álgebra de Conjuntos
    b. Cualquier acceso a la Base se puede formular sólo en términos de operadores algebraicos
    c. Si se añade un nuevo operador al conjunto, necesariamente será redundante
    d. Si se suprime algún operador del conjunto, éste queda incompleto

3. El bloque "select A from R, S where B=5" equivale a:
    a. Seleccionar filas de R y S en las que B=5, realizar el **join** del resultado y proyectar sobre A
    b. No hay equivalencia entre SQL y Álgebra
    c. Realizar la **unión** (de conjuntos) de R y S, seleccionar filas con B=5 y proyectar sobre A
    d. Seleccionar en R y S las filas con B=5, realizar la **unión** de conjuntos del resultado y proyectar sobre A

4. Si el atributo A sólo aparece en el universo (cabecera) de R; el resultado de Select A from R,S es:
    a. Un conjunto de valores de A que coincide con la proyección de R sobre A
    b. Un conjunto de valores de A que es en general un subconjunto de la proyección de R sobre A
    c. Un conjunto de valores de A que es generalmente un superconjunto de la proyección de R sobre A
    d. Como A pertenece sólo al universo de R, la operación planteada no puede realizarse

5. El *join* (unión natural) de R y S es una operación conmutativa, pero:
    a. Si R tiene más filas que S, debe colocarse como primer factor para disminuir el tiempo de ejecución
    b. Debe ser S el primer factor para disminuir el tiempo de ejecución
    c. Da igual el orden de los factores porque la conmutatividad implica que el tiempo de ejecución es idéntico
    d. No hay una regla general

6. El núcleo del conjunto $\{A \to BC, CD \to E, AE \to D\}$ es:
    a. ABC
    b. ABD
    c. Conjunto vacío
    d. A

7. El núcleo de un conjunto de dependencias es:
    a. La intersección de todas las claves
    b. La unión de todos los implicantes
    c. La unión de todos los implicados
    d. La intersección de los dos anteriores

8. Con relación al conjunto $\{A \to BC, CD \to E, AE \to D\}$, puede afirmarse que:
    a. AC es una clave
    b. ABCDE es clave única
    c. AD es una clave
    d. No hay ninguna clave.

9. Un recubrimiento equivalente y no redundante de $\{A \to BC, CD \to E, AE \to D\}$ es:
    a. $\{A \to B, A \to C, CD \to E, AE \to D\}$
    b. $\{A \to B, A \to C, CD \to E\}$
    c. $\{A \to B, A \to C, C \to D, D \to E\}$
    d. $\{A \to B, A \to C, C \to E, D \to E, A \to D, E \to D\}$

10. El cierre de AE respecto de $\{A \to BC, CD \to E, AE \to D\}$ es:
    a. AED
    b. AE
    c. ABCDE
    d. BCD

11. Dado un conjunto L de dependencias, un recubrimiento (conjunto equivalente) no redundante del mismo, es:
    a. El mismo conjunto, con las dependencias en orden léxico-gráfico
    b. Un conjunto equivalente con el mínimo número de dependencias
    c. Un conjunto equivalente cuyas dependencias tienen implicados simples, no hay atributos superfluos en los implicantes y no hay dependencias redundantes
    d. El conjunto obtenido al aplicar en todas las formas posibles la propiedad transitiva a las dependencias de L

12. Antes de proceder al cálculo de una clave del conjunto L de dependencias funcionales, es necesario:
    a. Comprobar que L no contiene dependencias repetidas
    b. Comprobar que no hay inconsistencias
    c. Extraer un conjunto equivalente a L (recubrimiento) no redundante
    d. No es necesario hacer nada previamente

13. El Algoritmo de Ullmann aplicado a $\{A \rightarrow BC,\ CD \rightarrow E,\ AE \rightarrow D\}$ produce como resultado:
    a. R1(A, B, C), R2(C, D, E), R3 (A, D, E)
    b. R1(A, B, C), R2(C, D, E), R3(A, D,E), R0(A, D)
    c. R1(A, B, C), R2(C, D, E), R3(A, D, E), R0(A,E)
    d. R(A, B, C, D, E)

14. Las relaciones (subesquemas) obtenidas al aplicar a la relación universal R el algoritmo de Ullmann, tienen la propiedad *lossless-join*. Esto significa que:
    a. El *join* de dichos esquemas coincide con R
    b. El *join* de dichos esquemas incluye a R
    c. El *join* de dichos esquemas está incluido en R
    d. Depende de cada caso

15. Al aplicar el algoritmo de Ullmann a una relación universal R se obtiene un conjunto de subesquemas que:
    a. Hay que normalizar
    b. No hay que normalizar, porque ya están todos en Tercera Forma Normal
    c. El tiempo de análisis del nivel de normalización es alto y no compensa por tanto realizar ese estudio
    d. El nivel de normalización es una propiedad irrelevante.

JUNIO 2018

Calcular un conjunto de L no redundante, una clave y los subesquemas a partir de la relación universal R(T,L):

T = {B,C,E,F,V,M}
L = {E → FVM, E → BC, B → FM, BC → V

JUNIO 2017/2016

Calcular un conjunto de L no redundante, una clave y los subesquemas a partir de la relación universal R(T,L):

T = {A,B,C,D,E,F,G}
L = {B → CDE, DE → F, BG → D, DG → B, CF → D}

Formular en Álgebra la interrogación: *Valores de B para los que F=5 y G=3;* aplicando, si es posible, criterios de optimización y asegurando la no aparición de valores falsos en el resultado.

MARZO 2021

1. Dado el siguiente modelo relacional:

Relación DEPORTISTA

cod_deportista	nombre	apellidos
10012	Lidia	Jordan
10013	Sergio	Garcia
10043	Teresa	Nadie
29349	Luis	Perez
20972	Ángel	Castaño

Relación JUEGA

cod_deportista	cod_deporte
10012	1
10013	4
10012	2
10012	3
10013	3
20972	4

Relación DEPORTE

cod_deporte	nombre_deporte
1	golf
2	baloncesto
3	rugby
4	balonmano

Escribe usando sólo operadores simples de álgebra relacional la operación que obtenga la relación derivada:

a. la relación con los atributos nombre de deporte y nombre y apellidos del deportista, que contenga las tuplas de los deportistas que juegan al balonmano.

b. la relación con los atributos nombre del deporte, que contenga las tuplas de todos los deportes a los que juegan los deportistas con código 10012 o 10013.

2. Dado el siguiente modelo relacional:

Relación R

A	B	C
1	1	1
1	1	2
2	3	3
1	3	3
1	1	3
2	3	1

Relación S

A	B
1	1
1	3

Relación T

B	H
1	a
1	c
1	b

a. Escribe el contenido en forma tabular de la relación derivada de realizar un left outer join sobre los atributos comunes de relaciones anteriores: S ⟕ T

b. Escribe el contenido en forma tabular de la relación derivada de realizar una división sobre los atributos comunes de relaciones anteriores: R ÷ S

3. Ejercicio de simplificación de dependencias funcionales:

Considerando el conjunto de atributos T y las siguientes dependencias funcionales L:
$T = \{A,B,C,D,E,F\}$
$L = \{E \rightarrow C,\ B \rightarrow BE,\ AB \rightarrow CE,\ A \rightarrow CFE\}$

Descomponer la relación R(T,L) en un esquema 3NF usando el algoritmo de Ullman. Aplicar la simplificación de dependencias, con eliminación de redundancias, y cálculo del cierre y de la clave, sobre el conjunto de dependencias funcionales R(T,L), según lo visto en los ejemplos de clase.