Bases de Datos I

Facultad de Ciencias Exactas
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Restricciones de Integridad

INGENIERÍA DE SISTEMAS

Cursada 2019

Restricciones de Integridad (RI)

"A database is only as good as the information stored in it, and a DBMS must therefore help **prevent the entry of incorrect information**"

(Ramakrishnan, Gehrke)

RI

- condiciones (reglas o chequeos) que restringen los valores posibles en la BD
- descripción de estados correctos en tiempo de diseño
- previenen inconsistencias

Una actividad importante en el diseño de una BD consiste en identificar e incorporar las restricciones de integridad que deben cumplirse

forzar las RI → garantizar instancias legales de la BD

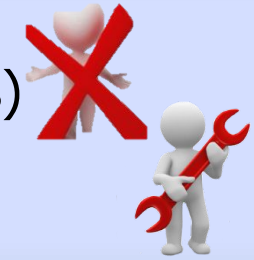
Mantener la Integridad en la BD

DBA /diseñadores de la BD: especifican las RI sobre los datos

- → mediante código en las aplicaciones que acceden a los datos
- → definiendo **restricciones en la BD que el SGBD se ocupa de interpretar**
- → *Tener en cuenta:* en la BD no sólo se almacenan los datos sino la definición completa de su estructura y restricciones sobre ellos (metadatos)

SGBD: evita manipulaciones de datos que no cumplan las RI

- → **rechaza** las operaciones (ante altas, bajas, modificaciones)
- → realiza **acciones reparadoras** adicionales sobre los datos



Ambas respuestas deben dejar la BD en un **estado consistente**

Clasificación de RI

Por su naturaleza:

- **RI inherentes** o **implícitas** del modelo de datos: no requieren ser especificadas *(valores de atributo atómicos, una relación no puede tener tuplas duplicadas)*
- **RI explícitas:** expresadas en el modelo de datos, deben especificarse mediante el DDL *(restricciones de dominio, de clave, de nulidad, integridad de entidades, integridad referencial)*
- **RI semánticas o "reglas del negocio":** establecen restricciones adicionales a incorporar a la BD
  → se pueden plantear en forma Declarativa o Procedural

Clasificación de RI

Por los estados involucrados:

- **RI de estado:** restringen los valores que pueden tomar los datos
  - **Unicidad:** no puede haber claves repetidas
  - **(No) Nulidad:** el valor de un atributo (no) puede ser nulo
  - **Dominio:** los valores de un atributo deben pertenecer a un conjunto (dominio) definido (Ej. *el sueldo tiene 2 decimales y no puede ser negativo)*
  - **Cardinalidad de una relación:** número de veces que una entidad participa de una relación (Ej: *Los empleados pueden asignarse a 5 proyectos como máximo*)
  - **Participación en una relación:** puede ser obligatoria u opcional (Ej: *Un empleado debe/puede estar vinculado a un área*)

- **RI de transición:** restringen los posibles cambios de valores entre estados sucesivos de los datos (Ej: *el sueldo básico de un empleado no puede decrecer) → implica* ***conocer el estado "anterior" y el "nuevo"*** *para compararlos*

RI de unicidad y no-nulidad

- La clave de una relación R es un conjunto no vacío de atributos que identifican unívocamente cada tupla de R
- Si hay más de una clave candidata:

→ **Clave primaria**: es la (única) clave candidata elegida para identificar unívocamente las tuplas de la relación (PRIMARY KEY en SQL)

→ **Clave/s alternativa/s**: otra/s clave/s candidata/s (UNIQUE en SQL)

Nota: SQL no requiere la declaración de tales RI (podría haber tablas sin def. de PK o UNIQUE)

IMPORTANTE: Los atributos que integran la clave primaria no pueden ser ni total ni parcialmente nulos!

Los valores nulos representan información desconocida o inaplicable
Si algún atributo clave fuera nulo → las tuplas no podrían ser identificadas

RI de unicidad y no-nulidad


```
CREATE TABLE NombreTabla
( { nombre_columna  TipoDato  [NOT NULL]
        [UNIQUE]
        [DEFAULT valorDefecto]
        , }
  [ [CONSTRAINT nom_restr] PRIMARY KEY (lista_ColumnasPK),]
  { [ [CONSTRAINT nom_restr] UNIQUE (listaColumnas),] }
  + Otras cláusulas... ) ;
```

NOTA:
[...]: opcional
{...}: uno o más
PALABRA RESERVADA

sólo uno de los atributos podría ser definido como PRIMARY KEY

o: **ALTER TABLE** NombreTabla

ADD [CONSTRAINT nom_restr**] PRIMARY KEY (**lista_ColumnasPK**);**

ídem UNIQUE

(Chequear la especificación de la sentencia del DBMS utilizado)

RI de unicidad y no-nulidad

- La restricción NOT NULL sobre un atributo (clave o no) hace que el SGBD rechace cualquier intento de colocar un nulo en esa columna
- Las claves primarias deben ser no nulas *(algunos SGBD lo declaran implícitamente, otros requieren especificar sus atributos como NO NULOS)*
- Las claves primarias (PK) y alternativas (UNIQUE) se pueden especificar en la definición de la tabla (o en la definición del atributo si es único), o mediante sentencias de alteración de definición de la tabla
- Se pueden asociar varias restricciones UNIQUE a una tabla, pero sólo una restricción PRIMARY KEY
- Se puede especificar un valor por defecto en una columna mediante la cláusula DEFAULT

RI Referencial (RIRs)

- Impone condiciones a los valores de las **claves extranjeras** (FK)

El conjunto de valores de la FK de una relación *R* debe coincidir con el conjunto de valores de los atributos en una relación *R´*-que son clave (prim. o altern.)- al que la FK hace referencia, o bien ser nulo

si es posible (por la definición de los datos)

R **[** *lista_col_referencia* **] <<** ***R´*** **[** *lista_col_referenciada* **] : (b,m)**

b= acción referencial ante baja

m= acción ref. ante modificación (a derecha)

- ***R*** y ***R´*** podrían coincidir
- *lista_col_referencia* y *lista_col_referenciada* deben tener igual número de columnas y tipos de datos compatibles (no requiere igualdad de nombres)

RI Referencial (RIRs)


```
CREATE TABLE NombreTabla
    ( { nombre_columna  TipoDato  [NOT NULL] ... ,}
      [ [CONSTRAINT PK_nom] PRIMARY KEY (lista_columnasPK),]
      { [ [CONSTRAINT U_nom] UNIQUE (lista_columnas),] }
      { [ [CONSTRAINT FK_nom] FOREIGN KEY (lista_col_referencia)
            REFERENCES nombreTablaRef [(lista_col_referenciada)]
            [ MATCH {FULL | PARTIAL | SIMPLE} ]
            [ON UPDATE AccionRef]
            [ON DELETE AccionRef] ] }  .....
    );
```

comportamiento ante borrado o modificación de datos referenciados

o: **ALTER TABLE** NombreTabla

ADD [CONSTRAINT FK_nom] **FOREIGN KEY (**lsta_col_referencia**) ..... ;**

AccionRef = NO ACTION | CASCADE | SET NULL | SET DEFAULT | RESTRICT

RIRs – Acciones Referenciales

R [*lista_col_referencia*] << *R´* [*lista_col_referenciada*]: (**b**, m)

¿Qué sucede al intentar borrar una tupla referenciada en *R´* por la FK en *R*?

→ ***Rechazar* la operación**

→ **NO ACTION:** impide borrar la tupla referenciada en *R´* si hay referencias en *R (es la opción por defecto en SQL estándar)*

→ **RESTRICT:** misma semántica pero se chequea antes de otras RI

→ ***Aceptar la operación y realizar acciones reparadoras adicionales***

→ **CASCADE:** borra la tupla referenciada en R´ y propaga el borrado a las tuplas en R que la referencian mediante la FK

→ **SET NULL:** borra la tupla referenciada en R´ y las tuplas que la referenciaban en R ponen a nulo la FK (si se admiten nulos)

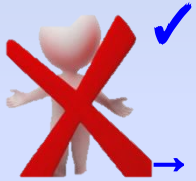
→ **SET DEFAULT:** borra la tupla referenciada en R´ y las tuplas que la referenciaban ponen en la FK el valor por defecto definido para la misma

RIRs – Acciones Referenciales

R [*lista_col_referencia*] << *R´* [*lista_col_referenciada*]: (b, **m**)

¿Qué sucede si se intenta modificar el valor de la clave de la tupla referenciada en *R´* por la FK en *R*? (modificación a derecha)

→ ***Rechazar*** **la operación**



- ✓ **NO ACTION:** no permite modificar el valor de la clave en la tupla referenciada en *R´* si hay referencias en *R* *(es la opción por defecto)*
- → **RESTRICT :** misma semántica pero se chequea antes de otras RI

→ **Aceptar la operación y realizar acciones reparadoras adicionales**

- ✓ **CASCADE:** modifica el valor de la clave de la tupla referenciada en *R´* y propaga en R la modificación a las tuplas que la referencian
- ✓ **SET NULL:** modifica la tupla referenciada en *R´* y las tuplas que la referenciaban en *R* ponen a nulo la FK (si admite nulos)
- ✓ **SET DEFAULT:** modifica la tupla referenciada en *R´* y las tuplas que la referenciaban en *R* ponen en la FK el valor por defecto definido

Ejercicios

Suponga la existencia de las siguientes tuplas en las respectivas tablas

¿Cómo proceden las operaciones según las diferentes acciones referenciales?

Nota: Sólo se incluyen los atributos que se consideran relevantes para el ejercicio.
Considere la instancia dada y resultados individuales, no acumulativos)

FK → PK

Empleado			
idE	Nombre	...	**proy**
1	E1	...	101
2	E2	...	101

Proyecto	
IdProy	NombreP
101	PP
102	QQ

1. **DELETE FROM Proyecto WHERE IdProy = 101;**
2. **DELETE FROM Proyecto WHERE IdProy = 102;**
3. **DELETE FROM Proyecto WHERE IdProy > 100;**
4. **UPDATE Proyecto set IdProy = 201 where IdProy = 101;**
5. **UPDATE Proyecto set IdProy = 202 where NombreP = 'QQ';**
6. **UPDATE Proyecto set NombreP='RR' where IdProy = 101;**
7. **UPDATE Empleado set proy=102 WHERE Nombre = 'E2';**

Desarrollo en clase

RIRs – Tipos de Matching

R [*lista_col_referencia*] << ***R'*** [*lista_col_referenciada*]

- Indican los requisitos que deben cumplir los conjuntos de valores de atributos de la FK en *R*, respecto de los correspondientes valores en la clave referenciada en *R´*
- Tienen sentido cuando la FK se define sobre 2 o más atributos y alguno/s de ellos puede/n contener nulos

Una RIR se satisface si para cada tupla en la tabla *R* se verifica que:

Ninguna de las columnas de la FK en *R* es NULL y existe una tupla en la tabla referenciada *R´* cuyos valores de clave coinciden con los de tales columnas, o ...

- al menos una de las columnas en la FK en *R* es NULL (**MATCH SIMPLE**)
- los valores de los atributos no nulos de la FK en *R* se corresponden con los respectivos valores de la clave en al menos una tupla de *R´* (**MATCH PARTIAL**)
- todas las columnas de la FK en *R* son NULL (**MATCH FULL**)

Ejercicios

PROYECTO (IdProy, CodA, TipoA) AREA (IdArea, TipoA, ...)

AREA

IdArea	TipoA	...
I1	T1	...
I1	T2	...
I2	T1	...

Indique si las siguientes **altas en Proyecto** serían aceptadas/ rechazadas por el SGBD, según la instancia dada y considerando los distintos tipos de matching *(suponiendo que la FK admite nulos):*

insert into PROYECTO (IdProy, CodA, TipoA)....

	MATCHING		
	Simple	**Parcial**	**Full**
values (1 , I1 , T1);	√	√	√
values (2 , I2 , T2);	X	X	X
values (3 , null , null);	√	√	√
values (4 , null , T1);	√	√	X
values (5 , I3 , null);	√	X	X

Desarrollo en clase

BIBLIOGRAFÍA

Date, C., "An Introduction to Database Systems". 7º ed., Addison Wesley, 2000

Elmasri, R., Navathe, S., "Fundamentals of Database Systems", Addison Wesley, 2011 (y 2016)

Silberschatz, A., Korth, H, Sudarshan, S., "Database System Concepts", McGraw Hill, 2002

Sumathi S., Esakkirajan S., Fundamentals of Relational Database Management Systems, 2007