Dpto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Elementos de Bases de Datos

Segundo Cuatrimestre 2015

Clase 7:

Modelo de Datos – Restricciones de Integridad

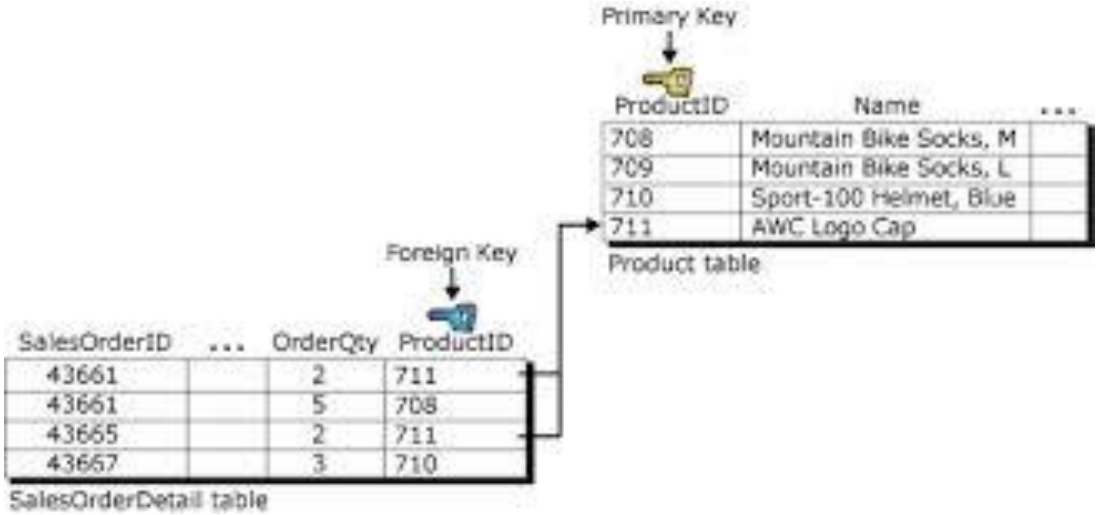
Mg. María Mercedes Vitturini
[mvitturi@uns.edu.ar]

Servicios del SMBD - Repaso

1. Soporte de al menos un MODELO DE DATOS.
2. Disponibilidad de *Lenguajes de alto nivel* para administrar o manipular la base de datos
   - *Lenguaje de Manipulación de Datos* (LMD/DML).
   - *Lenguaje de Definición de Datos* (LDD/DDL).
3. Mantener integridad y consistencia de datos.
4. Brinde *seguridad* y facilidades en la *administración de datos*.
5. Provea manejo de transacciones (commit y rollback).
   - Provea *control de concurrencia* y capacidades para *compartir datos*.
   - Permita *recuperaciones de fallos*.
6. Brinde *seguridad* y facilidades en la *administración de datos*.

Integridad de Datos con SQL

Primary Key,
Foreign Key,
Unique,
Check,
Not Null



Lenguajes - Repaso

- ***DDL*** - Lenguaje de definición de datos.
  - Definición de esquema de relaciones y vistas.
  - Definición de usuarios, roles y privilegios.
  - Definición de reglas de integridad.

- ***DML*** - Lenguaje de manipulación de datos.
  - Consultar datos almacenados.
  - Modificar el contenido de los datos almacenados: agregar nuevo contenido, borrar o modificar un dato.
  - Sólo datos que cumplan con las restricciones del DDL

SQL

SQL - DDL Provee instrucciones para:

- Definir una base de datos.
- Definir esquemas de relación.
  - Asociar con cada atributo un dominio.
  - Crear índicies,
  - Definir restricciones
  - Definir el esquema de autorización
- Especificar la organización y estructura del almacenamiento físico en disco.

SQL – DML

- Controla que las restricciones anteriores se satisfagan cada vez que
  - Se agregan datos
  - Modifican datos
  - Borran datoa

Restricciones de integridad

Las *restricciones de integridad en el modelo de datos* ”protegen” a la base de datos de daños, preservando a los datos en estados “consistentes”.

- Ejemplos de restricciones:
  - El saldo de las cuenta debe ser mayor o igual a 10$.
  - Los clientes deben informar un número de teléfono de contacto.
  - La suma de los préstamos de una sucursal del banco no debe exceder a la suma de los depósitos recibidos.
  - La fecha-hora de cierre de un estacionamiento no debe ser inferior a la fecha-hora de su apertura.

Ejemplo - SQL

```
CREATE TABLE docentes (
    id                  char(5),
    nombre              varchar(20) not null,
    dpto                varchar(20),
    horas_semanales     numeric(5,2),

primary key (ID) constraint pk_dte,
foreign key (depto) references departmentos
constraint fk_a_dpto,
check (horas_semanales>=0.0)
                        );
```

La sintaxis es propia de cada DBMS

Mecanismos de integridad

Formas de definir restricciones:

1. **Restricciones de dominio –** ajustar el dominio de los atributos, restringen cada atributo a un dominio de valores posibles.
2. **Restricciones de integridad referencial –** aseguran que el valor que aparece en una relación para un conjunto de atributos este presente en otra relación para cierto conjunto de atributos.
3. **Disparardores o triggers –** es una orden que ejecuta el sistema como efecto de un cambio.

Restricciones de dominio

Cada *atributo de relación se asocia con un dominio* que limita los valores que las relaciones pueden asumir para dicho atributo. La definición de dominio puede incluir:

- Tipo de datos, tamaño, rango, enumerados, etc.

Características de las restricciones de dominio

- Son simples de implementar.
- Restringen cada atributo a un *dominio de valores posibles*.
- Una restricción especial para un atributo es **no aceptar valores nulos**.

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n*.
- **varchar(n).** Variable length character strings, with user-specified maximum length *n*.
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.
- More are covered in Chapter 4.

Integridad de Entidad

La **integridad de entidad** – si una relación tiene llave primaria entonces:

- Los valores de la llave primaria son válidos en su dominio,
- no están duplicados en tuplas distintas y,
- que los valores de los atributos que pertenecen a la llave primaria no tienen valor nulo.

- Si un atributo que no pertenece a la llave primaria admite valores nulos significa:
  - Que el atributo no es requerido y puede no tener valor nunca,
  - o que el valor del atributo no es conocido inicialmente, y se podrá completar más adelante

SQL – Restricciones en el esquema de relación

- Mecanismos para incluir restricciones al momento de definir el esquema de una relación:
  - **NOT NULL,**
  - **PRIMARY KEY,**
  - **UNIQUE,**
  - **CHECK (P),** *donde P es un predicado*.

- Observación: los ejemplos que utilizan instrucciones SQL de esta clase están escritos con sintaxis de SQL estándar, pueden no funcionar en MySQL.

Restricciones sobre la relación

- Valores no nulos. En la creación del esquema

  … *nombre_sucursal* **char**(15) **not null**, …

- Para llaves primarias:
  - **primary key** (A_1, A_2, …, A_m)
  - Los atributos A_1, A_2, …, A_m no pueden tener valor nulo, aunque no se defina explicitamente la restricción.

- Otras llaves candidatas:
  - **unique** (A1, A2, …, Am)
  - Las llaves candidatas tienen permitidos valores nulos.

SQL – Restricciones de dominio

- La cláusula **check** de SQL provee facilidades adicionales para implementar otras restricciones de dominios.

Ejemplo:

- Asegurar que todas cuenta tengan saldo mínimo de 100$:

```
CREATE TABLE Cuentas (
    ...
    saldo          DECIMAL,
    ...
    CHECK (saldo>=100.00),
    ...);
```

Ejemplo

```
CREATE TABLE socios (
  nro_socio       INTEGER NOT NULL,
  tipo_dni        VARCHAR (5),
  nro_dni         NUMERIC (8,0),
  nombre          VARCHAR(45) NOT NULL,
  apellido        VARCHAR(45) NOT NULL,
  tipo_socio      VARCHAR(8),

  PRIMARY KEY (nro_socio),
  UNIQUE (tipo_dni, nro_dni),
  CHECK (tipo_socio IN ('Plateado', 'Dorado')),
  CHECK (tipo_dni IN ('DNI', 'LC', 'LE'))
);
```

Integridad Referencial

La restricción de **integridad referencial** mantiene *consistencia entre filas de dos relaciones*. Asegura que los valores de uno o más atributos que aparecen instanciados en una relación, además están presentes como valores atributos de otra relación.

- Dados *r*(R) y *s*(S) y la operación *r*|><|*s,* puede darse que una tupla $t \in r$ que no haga join con ninguna tupla de *s*.
- Bajo estas circunstancias *t* se denomina *colgante*. No siempre es correcto que se permitan tuplas colgantes.

Integridad Referencial

- Dados:

  CURSAN (aluLU, matCódigo)

  MATERIAS (matCódigo, matNombre).

- De *cursan* |><| *materias*, se espera:
  - Que toda tupla de *cursan* tenga join con alguna de las materias de la relación *materias*.
  - Sin embargo, no habría problemas que exista alguna tupla de *materias* no haga join con ninguna tupla de *cursan.*
  - El atributo *matCódigo* de cursan debe tener definida su correspondiente **"clave foránea"** hacia materias, indicando que toda materia que pertenezca *cursan* referencia a alguna materia de la relación *materias*.

Clave Foránea – Definición Formal

- Formalmente, sean $r_1(R_1)$ y $r_2(R_2)$ dos relaciones con claves primarias K_1 y K_2 respectivamente. Se dice que un subconjunto α de R_2 es una clave foránea que hace referencia a K_1 de la relación r_1 si se exige que para cada t_2 de r_2 haya una tupla t_1 de r_1 tal que $t_1[K_1]=t_2[\alpha]$.

- Observar que la definición de clave foránea en r_2 referencia a la clave primaria de r_1.

Integridad Referencial y E-R

- Si el esquema de base de datos relacional se obtiene a partir del modelo E/R entonces, cada esquema de relación que proviene de un conjunto de relaciones deberá tener referencias foráneas a las entidades que referencia en el DER.

- Lo mismo ocurre con:
  - Los conjuntos de entidades débiles con relación al conjunto de entidades fuertes del que depende.
  - Las conjuntos de entidades que se relacionan por una relación de herencia “is-a” con su padre.

Ejemplo: Definición en SQL

```
CREATE TABLE  atenciones (
        at_numero           integer NOT NULL,
        pa_numero           integer NOT NULL,
        me_matricula        integer NOT NULL,
        at_fecha            date NOT NULL,
   PRIMARY KEY (at_número) CONSTRAINT pk_atenciones,
   UNIQUE KEY (pa_numero, me_matricula, at_fecha)
            CONSTRAINT uk_atenciones,
   FOREIGN KEY (me_matricula) REFERENCES medicos
            CONSTRAINT fk_at_medicos,
   FOREIGN KEY (pa_numero)  REFERENCES pacientes
            CONSTRAINT fk_at_pacientes
);
```

Seguridad

Algunas herramientas para preservar la seguridad /accesos indebidos de la base de datos

Definir Vistas

- En ciertos casos interesa *restringir o por el contrario completar el acceso a la información* a ciertos usuarios.
- Ejemplo:
  - Sea un esquema, no todos deberían tener acceso a los datos de salario o domicilio de los empleados

EMPLEADOS (id, nombre, domicilio, salario).

- La **vista** es un medio para ocultar parte de la información, permitiendo crear nuevas *"relaciones virtuales".*

Vistas

- Definición:

CREATE VIEW *nom_vista* **AS** *< query expression >*

donde, *nom_vista* es el nombre asignado a la "nueva relación virtual" y *< query expression >* es una expresión SQL válida.

- La vista NO crea una nueva relación, sino que guarda la expresión de consulta.
- Cada vez que se referencia a la vista, ésta se calcula dinámicamente.

Ejemplos

- Creando la vista:

```
CREATE VIEW vw_exam_aprobados (legajo, materia, calificacion)
AS
    SELECT nro_legajo, materia, nota
    FROM examenes
    WHERE  resultado='Aprobado';
```

- Usando la vista:

```
SELECT legajo, AVG(nota)  prom_parcial
    FROM vw_exam_aprobados
    WHERE  materia='Elementos de Bases de Datos';
```

Vistas

- Características:
  - Las vistas se resuelven remplazando el nombre de la vista *por su expresión*.
  - Dependiendo de la vista y las relaciones que participan, pueden resultar lentas.
  - La vista puede incluir referencias a otras vistas. Esto podría generar un problema recursivo.
  - En ciertos casos, las vistas además admiten que se realicen actualizaciones, esto no siempre funciona.

Seguridad a través de Control de Acceso a los Datos

Autorizaciones sobre los datos:

- **Read (select)** – permite lectura, no modificación de datos.
- **Insert** – permite agregar nuevos datos, pero no modificar los ya existentes.
- **Update** – permite modificar datos, pero no borrarlos.
- **Delete** – permite borrar.

Autorizaciones sobre los esquemas:

- **Index** – crear/borrar índices.
- **Resources** – crear nuevas relaciones.
- **Alteration** – agregar/borrar atributos de una relación.
- **Drop** – borrar relaciones.

Instrucción GRANT

grant <lista_de_privilegios> **on** <relación o vista>
to <lista_usuarios>

- Donde **<lista_usuarios>** puede ser:
  - un id-usuario,
  - **public** (≡ a permiso a todos los usuarios),
  - un rol.
- Y **<lista_de_privilegios>** puede ser:
  - select, insert, update, delete
  - all privileges

grant select **on** socios **to** *mercedes*;

grant all privileges **on** socios **to** *carlos*, *ana*;

revoke all privileges **on** sueldos **to** *rol_ventas*;

revoke <lista_de_privilegios> **on** <relación o vista>
to <lista_usuarios>

Funciones y Procedimientos

- Las **funciones** y **procedimientos** son un medio para "expresar" la lógica del negocio en la base de datos.
- Ventajas:
  + Son reutilizables desde múltiples aplicaciones,
  + Definen un único punto de modificación si las reglas del negocio cambian,
  + En general simplifican las tareas de mantenimiento,
- Desventajas:
  - Afecta a la performance general del sistema,
  - Determinan a las aplicaciones dependientes del software del DBMS.

Funciones

```
create function cnt_libros_prestados (pnro_socio integer)
  returns integer
     begin
          declare vcantidad integer;

          select count (*) into vcantidad
             from prestamos pr
             where pr.nro_socio = pnro_socio
        return vcantidad;
     end;
```

SELECT nro_socio, apellido, nombre

FROM socios

WHERE *cnt_libros_prestados(nro_socio)* >=2;

Procedimientos

```
create procedure sp_libros_prestados ( in pnro_socio
integer, out ocantidad  integer )
      begin
          declare vcantidad integer;

          select count (*) into vcantidad
             from prestamos pr
             where pr.nro_socio = pnro_socio
                    and fecha_devolucion is Null;
          LET ocantidad = vcantidad;
      end;
```

```
Declare cantidadLibros      INTEGER;
call  sp_libros_prestados('1214', cantidadLibros);
```

Triggers o Disparadores

ON *evento* **IF** *precondición* **THEN** *acción*

- Un **disparador** o **trigger** es una orden que el sistema ejecuta automáticamente como efecto secundario de una modificación a la base de datos.
- En un disparador se deben especificar:
  - Condiciones bajo las que se ejecuta:
    - Evento que causa el disparo.
    - Condiciones que se deben satisfacer.
  - Acciones que se van a realizar cuando se ejecute el disparador.

Triggers en SQL

ON *evento* **IF** *precondición* **THEN** *acción*

- Constituyen *eventos* la ejecución de una instrucción SQL de **SELECT, INSERT, UPDATE** o **DELETE**.
- Una *precondición* cualquier condición permitida en un **WHERE**.
- Una *acción* una consulta SQL, **DELETE, INSERT, UPDATE, ROLLBACK, SIGNAL**, o la ejecución de un programa SQL (Store Procedure).

Ejemplo

- Supongamos que en la operatoria de cuentas corrientes de un banco no se deben permitir saldos en las cuentas negativos. Cuando el saldo es negativo se deberá representar por un préstamo.

Algoritmo

- *En caso de un descubierto, proceder de la siguiente forma:*
  - *Dejar el saldo de la cuenta en 0.*
  - *Crear un nuevo registro en la tabla "Préstamos" por la diferencia y agregar el cliente a la tabla "Prestatario".*

En SQL

```
DEFINE TRIGGER descubierto ON UPDATE OF cuenta T
        IF NEW T.saldo < 0
            THEN (
                    INSERT INTO prestamos VALUES
                        (T.nombre-suc, T.nro-cta, -NEW T.saldo)

                    INSERT INTO prestatario
                        (SELECT nombre-cli, nro-cta
                            FROM depositario
                            WHERE T.nro-cta=depositario.nro-cta)

                    UPDATE cuenta S SET s.saldo= 0 WHERE
s.nro-cta = T.nro-cta)
                                )
```

Observaciones

- Las restricciones impuestas dentro del esquema de base de datos siempre se van a respetar.
- Otro forma de incorporar restricciones es incluirlas dentro de los programas de aplicación a través de agregar código apropiado en distintos puntos de control.
  - Si cambian las reglas se deben cambiar *todos* las ocurrencias de código donde la regla está definida.
  - Debe asegurarse que los controles se incluyan *en todas las aplicaciones* que acceden a los datos.
  - Sin embargo, se aprovechan las capacidades de procesamiento del cliente.

Variantes de Join - Outer Join

- El **outer join** es una extensión de la operación *join* que evita la pérdida de información cuando no hay coincidencia.
- Calcula el join y *luego se agregan las tuplas que no hacen match* con las tuplas de la otra relación del join.
- Usa valores *nulos* o indeterminado en los atributos que no se puede determinar el valor.

- *null* significa que el valor es desconocido o no existe.

Outer Join – Ejemplo

- Relación *préstamos*

número-prest	*sucursal-prest*	*importe*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relación *prestado_a*

cliente	*número-prest*
Jones	L-170
Smith	L-230
Hayes	L-155

Outer Join – Ejemplo

- **Join** (*inner-join*)**:** *prestamos |><| prestado_a*

número-prest	*sucursal*	*importe*	*cliente*
L-170 L-230	Downtown Redwood	3000 4000	Jones Smith

- **Left-outer-Join : *prestamos |><| prestado_a***

número-prest	*sucursal*	*importe*	*cliente*
L-170 L-230 L-260	Downtown Redwood Perryridge	3000 4000 1700	Jones Smith ***null***

Outer Join – Example

- **Right-outer-Join:** *prestamos |><| prestado_a*

número-prest	*sucursal*	*importe*	*cliente*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	***null***	***null***	Hayes

- **Full-outer-Join:** *prestamos |><| prestado_a*

Número-prest	*sucursal*	*importe*	*cliente*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

Temas de la Clase de Hoy

- Mecanismos de Integridad
  - Claves, dependencias funcionales, multiplicidad.
  - Integridad en SQL
    - Integridad de dominio.
    - Claves
    - Referencias foráneas
    - Triggers

Bibliografía

- *Database System Concepts*. A. Silberschatz. Capítulo 4.