Modelo de Datos

Diseño Conceptual

Universidad Católica Ntra. Sra. de la Asunción

Cardinalidad – Integridad Referencial

- Definición Cardinalidad:

La cardinalidad de una relación se define como la cantidad de ocurrencias de una tupla en la tabla hija con relación a la tabla padre. En base a la cardinalidad se pueden definir distintos tipos de relaciones.

- Relación uno a muchos.

  Notación: Tabla Padre ----|------←-----Tabla Hija

  - Dependiente. Cuando la columna en la tabla hija forma parte de la clave primaria.

  - No Dependiente: Cuando la columna en la tabla hija NO forma parte de la clave primaria.

  En caso que en la tabla hija la ocurrencia de tuplas sea exigida se indica en la notación con un | (siempre debe existir).

  En caso que en la tabla hija la ocurrencia de tuplas sea opcional se indica en la notación con una O (puede no existir).

Cardinalidad – Integridad Referencial

- Relación uno a uno.

  Notación: Tabla Padre ----|------|-----Tabla Hija

  -En este caso la relación es dependiente (ambas columnas son clave primaria), y en la myoria de los casos los atributos pueden ser conjugados en una sola tabla.

Cardinalidad – Integridad Referencial

- Relación muchos a muchos.

  Notación: Tabla 1 -----→------←-----Tabla 2

  -La relación muchos a muchos no se encuentra definida en el modelo de datos físico.

  -No existe una condición de tabla padre/hija

  -Para resolver esta relación se crea una tabla asociativa.

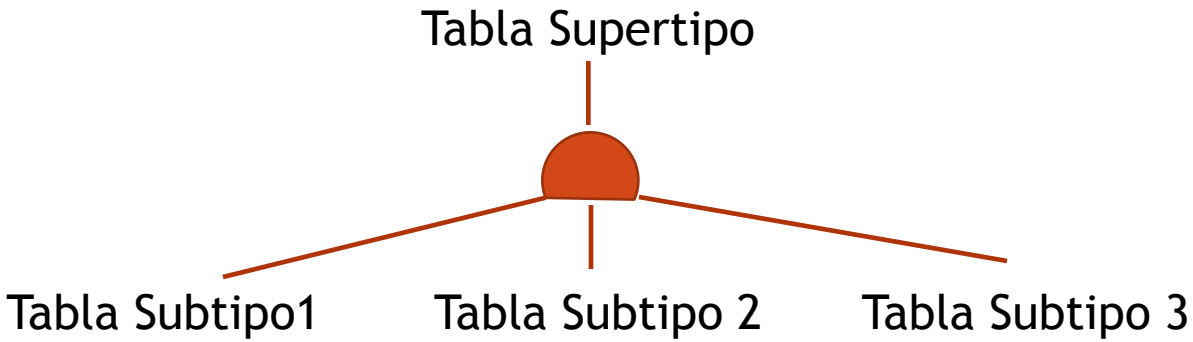
  -La tabla asociativa creada es dependiente de las tablas 1 y 2 (hija), es decir las columnas relacionadas con las tablas son parte de la clave primaria

  -Se pueden crear atributos adicionales en las tablas asociativas

Cardinalidad – Integridad Referencial

- Relación supertipo/subtipo.

  Notación:



-Los tipos de relación supertipo/subtipo se dan cuando existe un conjunto de atributos repetidos en varias tablas.

- En ese caso los atributos comunes son identificados en la tabla supertipo y los atributos distintivos en las tablas subtipo.

- La relacion de una tabla subtipo es dependiente y a su vez con una cardinalidad de uno a uno.

Modelo de Datos - Conceptos

- Instrumento que se aplica al Universo del Discurso para obtener una estructura de datos.

- Está definido por un Conjunto de Conceptos, Reglas y Convenciones que nos permiten describir los datos del Universo del Discurso.

- Se divide en dos componentes, un componente estático y otro dinámico.

- Dentro del componente estático se establecen un conjunto de objetos permitidos y un conjunto de objetos no permitidos (restricciones) que pueden ser inherentes al modelo o establecidas por el usuario.

Modelo de Datos - Conceptos

- Instrumento que se aplica al Universo del Discurso para obtener una estructura de datos.

- Está definido por un Conjunto de Conceptos, Reglas y Convenciones que nos permiten describir los datos del Universo del Discurso.

- Se divide en dos componentes, un componente estático y otro dinámico.

- Dentro del componente estático se establecen un conjunto de objetos permitidos y un conjunto de objetos no permitidos (restricciones) que pueden ser inherentes al modelo o establecidas por el usuario.

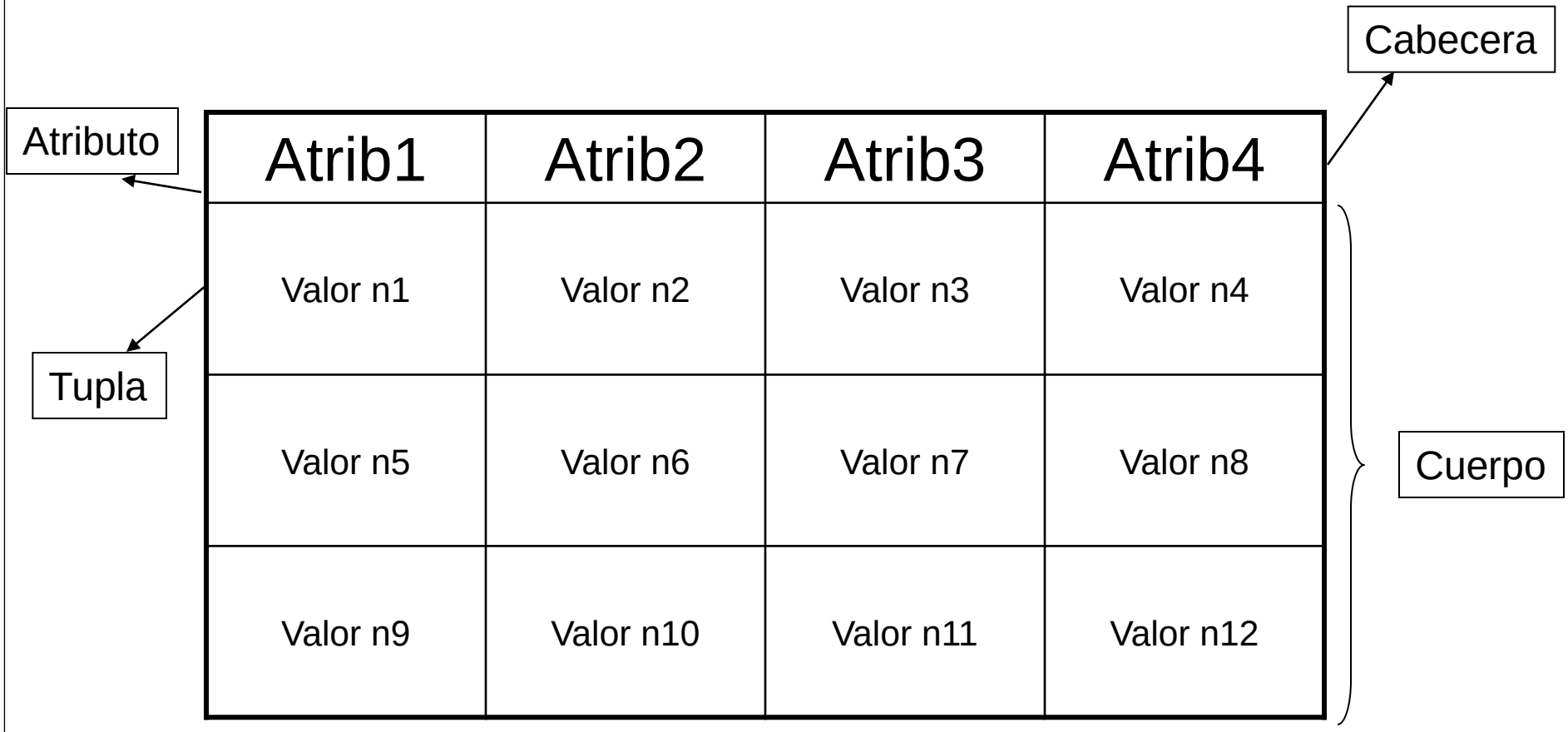
Modelo de Datos Relacional - Introducción

- Propuesto por E.F. Codd en 1970, su estructura fundamental es la “relacion”. Se basa en el concepto matemático de relación, que gráficamente se representa mediante una tabla.

- Componentes
- Estática: Objetos Permitidos
  - Tablas Atributos, Dominios
  - Restricciones
    - Inherentes
      - Clave Primaria
      - Clave Foranea
    - De Usuario
      - Constraints
      - Triggers
- Dinámica: Algebra Relacional

Restricciones - Objetos Permitidos

- De acuerdo a la nomenclatura del Modelo Relacional, la tabla se define como Relación y esta compuesta de:



Restricciones - Objetos No Permitidos

Restricciones Inherentes.

- Clave Primaria (Integridad Identidad):
  Se utiliza para definir el/los atributos que no pueden repetirse y no deben admitir valores nulos en la entidad.

  La definición de una restricción del tipo primary key genera automáticamente un índice.

- Clave Foránea (Integridad Referencial)
  Se utiliza para definir el/los atributos de una entidad que hacen referencia a la clave primaria de la entidad a la cual se hace referencia.

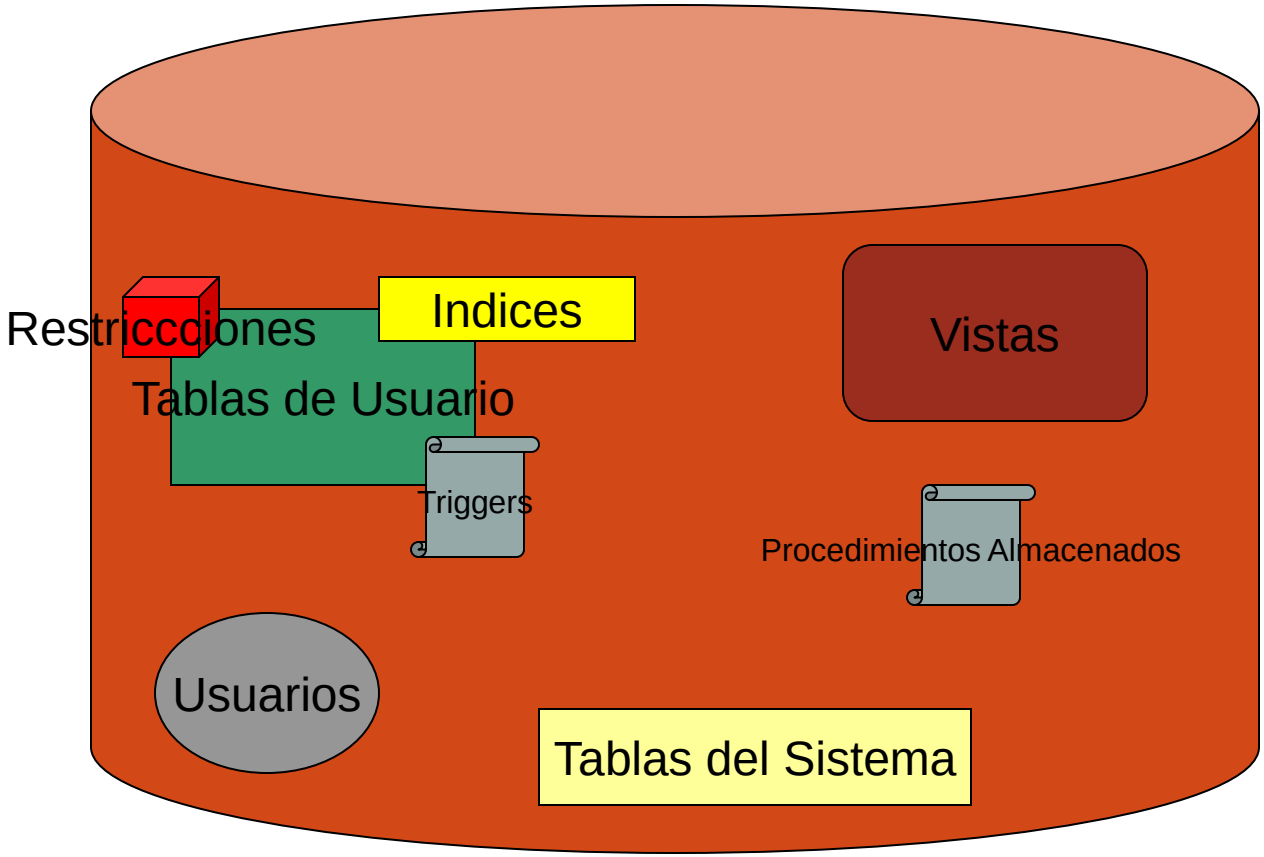
  La definición de una restricclón del tipo foreign key dependiendo del DBMS puede generar automáticamente un índice.

Implementación del Modelo de Datos Relacional en un SGBD

- Los sistemas gerenciadores que aplican el modelo relacional se los denominan R-SGBD.

- Una base de datos relacional es un conjunto de relaciones normalizadas. Para representar el esquema de una base de datos relacional se debe dar el nombre de sus relaciones, los atributos de éstas, los dominios sobre los que se definen estos atributos, las claves primarias y las claves ajenas

Base de Datos Relacional - Esquema



Tablas de Usuario

- La cabecera de una relación esta compuesta por un conjunto estático de atributos.
- El atributo de una relación esta definido sobre un dominio y todos los valores contenidos en el mismo son homogéneos.
- El nombre del atributo es único dentro de la relación
- Se conoce como grado a la cantidad de atributos de una relación.

Tablas de Usuario

- El cuerpo de una relación esta compuesto por un conjunto dinámico de registros conocidos como tuplas.

- Una tupla debe poder referenciarse en forma univoca (clave primaria).

- El valor contenido en el atributo/tupla es atómico, es decir se refiere a un dato en específico

- Se conoce como cardinalidad de la relación a la cantidad de tuplas que esta contenga.

Restricciones y Defaults

- Las restricciones o contraints se aplican a los atributos de una entidad o bien a la entidad misma.

- Existen distintos tipos de constraints.
  - Primary Key:
    Se utiliza para definir el/los atributos que no pueden repetirse y no deben admitir valores nulos en la entidad, el constraint es también conocido como de tipo unique.

    La definición de un constraint del tipo primary key genera automáticamente un índice unique clustered (opcional) en la base de datos
  - Foreign Key
    Se utiliza para definir el/los atributos de una entidad que hacen referencia a la clave primaria de la entidad a la cual se hace referencia.

    La definición de un constraint del tipo foreign key dependiendo del DBMS puede generar automáticamente un índice en la base de datos.
  - Check
    Son consideran constraints que se aplican a una columna de la entidad, pudiendo definir valores mínimos y máximos, lista de valores permitidos, máscaras de entrada de datos, entre otros.

Restricciones y Defaults

- Ejemplo 1 de un constraint de tipo Primary Key

Alumnos		
Matricula	char(6)	<pk>
Nombres	varchar(60)	
Apellidos	varchar(60)	
Fecha Nacimiento	datetime	
Edad	numeric(3)	

Script:

```
/*==============================================================*/
/* Table: ALUMNOS                                               */
/*==============================================================*/
create table ALUMNOS (
   MATRICULA            char(6)          not null,
   NOMBRES              varchar(60)      null,
   APELLIDOS            varchar(60)      null,
   FECHA_NACIMIENTO     datetime         null,
   EDAD                 numeric(3)       null
)
go

alter table ALUMNOS
   add constraint PK_ALUMNOS primary key  (MATRICULA)
go
```

Restricciones y Defaults

- Ejemplo 2 de un constraint de tipo Primary Key

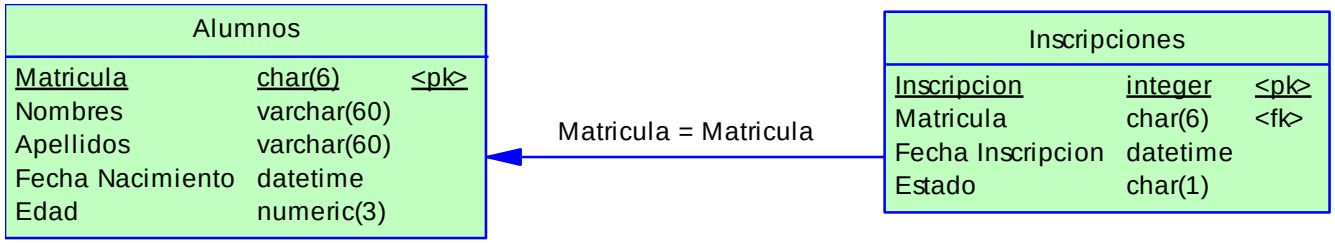
Alumnos		
Matricula	char(6)	<pk>
Nombres	varchar(60)	
Apellidos	varchar(60)	
Fecha Nacimiento	datetime	
Edad	numeric(3)	

Script:

```
/*==============================================================*/
/* Table: ALUMNOS                                               */
/*==============================================================*/
create table ALUMNOS (
   MATRICULA            char(6)          not null primary key,
   NOMBRES              varchar(60)      null,
   APELLIDOS            varchar(60)      null,
   FECHA_NACIMIENTO     datetime         null,
   EDAD                 numeric(3)       null
)
go
```

Restricciones y Defaults

- Ejemplo de un constraint de tipo Foreign Key



Script:

```
/*==============================================================*/
/* Table: INSCRIPCIONES                                         */
/*==============================================================*/
create table INSCRIPCIONES (
   INSCRIPCION          integer          not null,
   MATRICULA            char(6)          not null,
   FECHA_INSCRIPCION    datetime         not null
)
go


alter table INSCRIPCIONES
   add constraint PK_INSCRIPCIONES primary key  (INSCRIPCION)
go

alter table INSCRIPCIONES add constraint FK_INSCRIPCIONES_ALUMNOS foreign key(MATRICULA)
     references ALUMNOS(MATRICULA)
go
```

Restricciones y Defaults

- Ejemplo de un constraint de tipo Check para validar valores mínimos y máximos.

Alumnos		
Matricula	char(6)	<pk>
Nombres	varchar(60)	
Apellidos	varchar(60)	
Fecha Nacimiento	datetime	
Edad	numeric(3)	

Script:

```
/*==============================================================*/
/* Table: ALUMNOS                                               */
/*==============================================================*/
create table ALUMNOS (
   MATRICULA            char(6)          not null,
   NOMBRES              varchar(60)       null,
   APELLIDOS            varchar(60)       null,
   FECHA_NACIMIENTO     datetime          null,
   EDAD                 numeric(3)        null
      constraint CKC_EDAD_ALUMNOS check ((EDAD between 0 and 99 )
)
go
```

Restricciones y Defaults

- Ejemplo de un constraint de tipo Check para validar una lista de valores posibles.

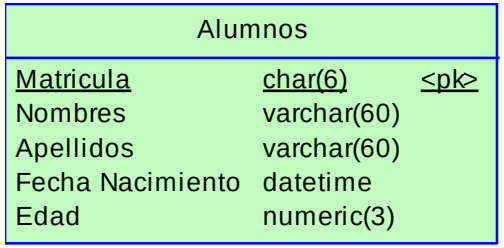
Inscripciones		
Inscripcion	integer	<pk>
Matricula	char(6)	<fk>
Fecha Inscripcion	datetime	
Estado	char(1)	

Script:

```
/*==============================================================*/
/* Table: INSCRIPCIONES                                         */
/*==============================================================*/
create table INSCRIPCIONES (
   INSCRIPCION          integer              not null,
   MATRICULA            char(6)              not null,
   FECHA_INSCRIPCION    datetime             not null,
   ESTADO               char(1)              null
      constraint CKC_ESTADO_INSCRIPC check (ESTADO in ('A','I') )
)
go
```

Restricciones y Defaults

- Ejemplo de un constraint de tipo Check para validar una máscara de entrada de datos.



Script:

```
/*==============================================================*/
/* Table: ALUMNOS                                               */
/*==============================================================*/
create table ALUMNOS (
   MATRICULA                 char(6)             not null primary key,
                  check (MATRICULA like ('[C][0][0-9][0-9][0-9][0-9]')),
   NOMBRES                   varchar(60)         null,
   APELLIDOS                 varchar(60)         null,
   FECHA_NACIMIENTO          datetime            null,
   EDAD                      numeric(3)          null)
go
```

Restricciones y Defaults

- La opción NOT NULL es también considerado como un constraint, el cual no permite la ausencia de valor para la columna especificada.

Inscripciones		
Inscripcion	integer	<pk>
Matricula	char(6)	<fk>
Fecha Inscripcion	datetime	
Estado	char(1)	

- Los constraints pueden mantenerse con la sentencia ALTER TABLE

ALTER TABLE add constraint
ALTER TABLE drop constraint
ALTER TABLE modify column NULL | NOT NULL

Restricciones y Defaults

- Los Defaults se consideran también como valores por omisión, es decir el valor que tomara la columna en caso que no sea especificada en la lista de columnas de una sentencia Insert.

Script:

```
/*==============================================================*/
/* Table: INSCRIPCIONES                                         */
/*==============================================================*/
create table INSCRIPCIONES (
   INSCRIPCION          integer          not null primary key,
   MATRICULA            char(6)          not null,
   FECHA_INSCRIPCION    datetime           not null,
   ESTADO               char(1)          null default 'A'
      constraint CKC_ESTADO_INSCRIPC check (ESTADO is null or ( ESTADO in ('A','I') ))
)
go
```

Restricciones y Defaults

- Ejemplo de usos de Defaults.

Inscripciones		
Inscripcion	integer	<pk>
Matricula	char(6)	<fk>
Fecha Inscripcion	datetime	
Estado	char(1)	

Script:

Ejemplo 1:

```
Insert Into Inscripciones
Values(1,'C03934',getdate(),null)
```

Ejemplo 2:

```
Insert Into Inscripciones(Inscripcion,Matricula,Fecha_inscripcion)
Values(1,'C03934',getdate())
```

Restricciones y Defaults - Conclusión

- Los constraints son mecanismos incoporados (built-in) en los Motores de Bases de Datos con el objetivo de forzar la integridad de los datos.

- Se recomienda utilizar constraints sobre los triggers, atendiendo que son realizaciones incorporadas a la tabla, tienen mayor performance y son fáciles de crear y mantener.

- Cuando se escribe código para realizar las mismas acciones que los constraints, puede haber errores, en este sentido los constraints son más consistentes y confiables.

- En este sentido se recomienda el uso de triggers solo en los casos que los constraints no provean la funcionalidad necesaria.

Triggers

- Los triggers como su nombre lo indica son “disparadores” o “desencadenadores” de un conjunto de sentencias de manipulación de datos (DML).

- La funcionalidad principal de los triggers radica en asegurar la consistencia de los datos de las tablas, trabajando en conjunto con los contraints.

- Cada tabla es “sensible” en el sentido que puede detectar los eventos de agregar, modificar y/o eliminar datos, a su vez se pueden detectar los momentos, es decir, “antes de” (before) o “despues de” (after).

Triggers

- Como implementa el Motor de Base de Datos los triggers?
  - Una vez creado un trigger para un evento (Insert, Update, Delete) y para un momento dado (Before, After) sobre una tabla, el Motor de Base de Datos realiza las siguientes acciones.
  - Crea una tabla interna y “temporal” la cual tiene la misma estructura de la tabla sobre la cual se creo el trigger y con los datos que se están manipulando.
  - Para el caso de un Insert, se crea la tabla *inserted*, la cual tambien puede ser referenciada por una variable *new*.
  - Para el caso de un Delete, se crea la tabla *deleted*, la cual tambien puede ser referenciada por una variable *old*.
  - En el caso de un Update se crean ambas tablas.

Triggers. Insert.

- *Insert*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345

SQL Syntax

Insert into personas
Values(4,'Pedro','Gomez','1799205')

Triggers. Insert.

- *Insert*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345

Tabla Inserted

Persona	Primer Nombre	Primer Apellido	Documento
4	Pedro	Gomez	1799205

Triggers. Insert.

- *En el momento Before el dato todavía no forma parte de la tabla personas, inclusive se tiene la posibilidad de modificar el dato inserted antes de que ingrese a la tabla definitivamente*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345

Tabla Inserted

4	Pedro	Gomez	1799205

Triggers. Insert.

- *En el momento After el registro ya forma parte de la tabla personas, de igual manera también esta registrado en la tabla inserted*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345
4	Pedro	Gomez	1799205

Tabla Inserted

4	Pedro	Gomez	1799205

Triggers. Insert.

- *Como se hace referencia a un dato que se esta insertando.*

  - El motor de base de datos permite hacer el select sobre la tabla inserted o bien a traves de una referencia de la variable new.

Tabla Inserted

Persona	Primer Nombre	Primer Apellido	Documento
4	Pedro	Gomez	1799205

Sybase Anywhere

```
referencing new as nvo

if nvo.documento is not null then ….
```

SQL Server

```
select @documento = documento
from   inserted
```

Oracle

```
:new.documento
```

Triggers. Delete.

- *Delete*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345
4	Pedro	Gomez	1799205

SQL Syntax

```
Delete personas
Where persona = 4
```

Triggers. Delete.

- *Delete*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345
4	Pedro	Gomez	1799205

Tabla Deleted

Persona	Primer Nombre	Primer Apellido	Documento
4	Pedro	Gomez	1799205

Triggers. Delete.

- *Delete*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345
4	Pedro	Gomez	1799205

Tabla Deleted

Persona	Primer Nombre	Primer Apellido	Documento
4	Pedro	Gomez	1799205

Triggers. Delete.

- *En el momento Before el registro ya forma parte de la tabla deleted, de igual manera también sigue registrado en la tabla personas*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345
4	Pedro	Gomez	1799205

Tabla Deleted

4	Pedro	Gomez	1799205

Triggers. Delete.

- *En el momento After el registro ya no forma parte de la tabla personas.*

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343
2	Maria	Gonzalez	234354
3	Luis	Arza	1987345

Tabla Deleted

4	Pedro	Gomez	1799205

Triggers. Delete.

- *Como se hace referencia a un dato que se esta eliminando.*
  - El motor de base de datos permite hacer el select sobre la tabla deleted o bien a traves de una referencia de la variable old.

Tabla Deleted

Persona	Primer Nombre	Primer Apellido	Documento
4	Pedro	Gomez	1799205

Sybase Anywhere

```
referencing old as vjo

if vjo.documento is not null then ….
```

SQL Server

```
select @documento = documento
from   deleted
```

Oracle

```
:old.documento
```

Triggers. Update.

- Que ocurre en el trigger de Update?
  - Se crean las 2 tablas *inserted* y *deleted*, la primera contiene el registro que se esta ingresando de acuerdo a la clausula ***set*** y la segunda contiene el registro que esta siendo reemplazando

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343

SQL Syntax

```
Update personas
Set      primer_nombre = 'Laura'
Where persona = 1
```

Triggers. Update.

- Que ocurre en el trigger de Update?

Tabla Personas

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343

Tabla Inserted

Persona	Primer Nombre	Primer Apellido	Documento
1	Laura	Vazquez	1099343

Tabla Deleted

Persona	Primer Nombre	Primer Apellido	Documento
1	Andres	Vazquez	1099343

Triggers.

- Como definir el momento (before/after) para el trigger ?
  - Las validaciones que tienen que ver con restricciones y que pueden resultar en la no aceptación de un registro, se considera óptimo realizarlas en el momento “antes de” o before.
  - Las actualizaciones de campos desnormalizados se recomienda realizar cuando el registro pudo pasar todas las validaciones previas y es momento de impactar las otras tablas de la Base de Datos, en este caso se considera óptimo relizarlas en el momento “despues de” o after.

Triggers.

- Cuando se necesitan crear triggers?
  - En el caso de validaciones complejas que no pueden ser resueltas por constraints.

    Ejemplo:

    Validar que un alumno que tiene deuda no puede ser inscripto.

Triggers.

- En que momento se necesitan crear triggers?
  - En el caso actualizaciones a otros campos que fueron desnormalizados para mejorar la performance.

    Ejemplo:

    Manejar el saldo de un Alumno.