



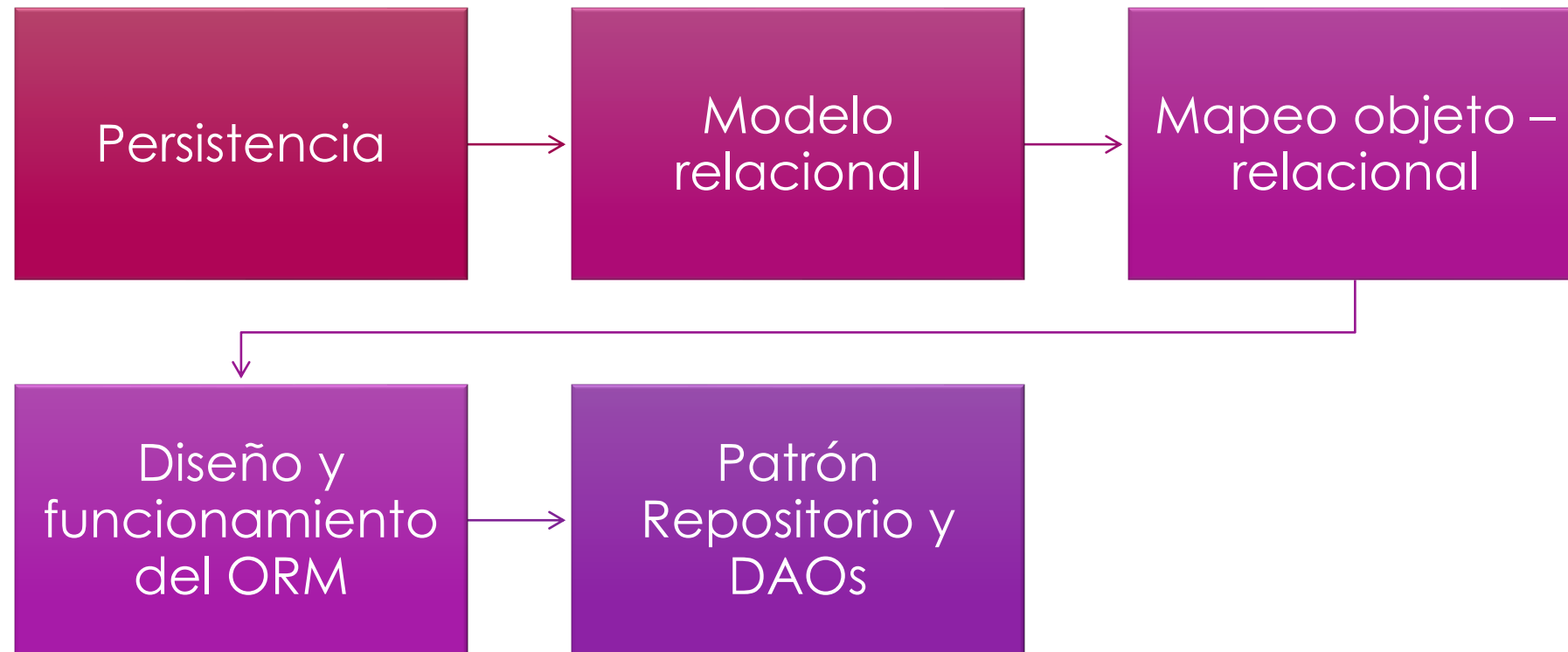
UTN.BA

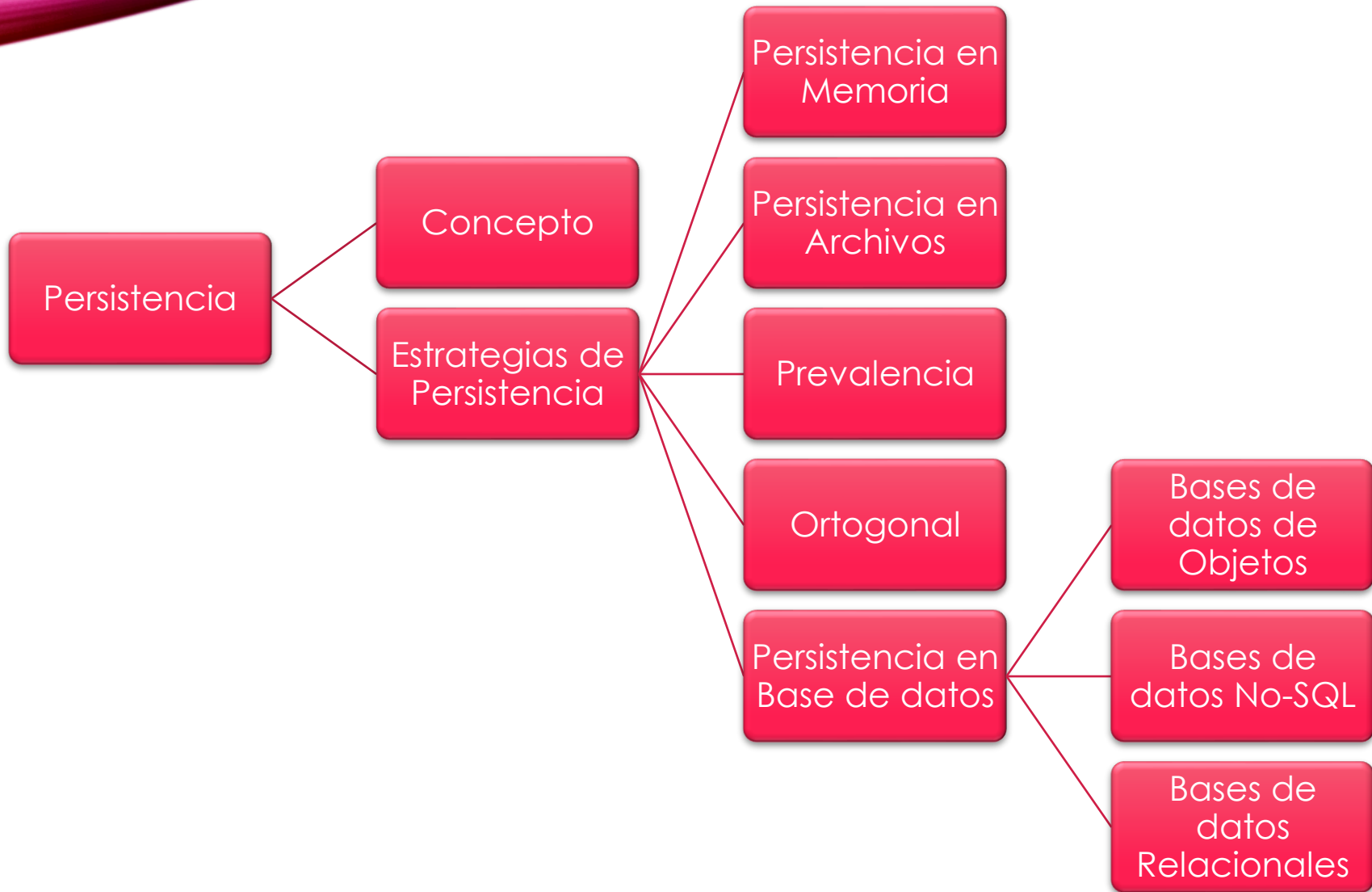
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

Persistencia en medios
relacionales mediante
ORM

PERSISTENCIA DE DATOS

TEMARIO GENERAL







PERSISTENCIA

En términos informáticos, la persistencia se refiere a que el estado de un sistema sobrevive más allá del proceso que lo creó; o dicho en otras palabras, sobrevive más allá de una única ejecución.

Esto se logra, en la práctica, almacenando dicho estado en algún (o algunos) medio persistente, tal como una base de datos o archivos.



ESTRATEGIAS DE PERSISTENCIA

Existen varias estrategias de persistencia de datos un aplicativo:

- *Persistencia en Memoria*
- *Persistencia en Archivos*
- *Prevalencia*
- *Ortogonal*
- *Persistencia en Base de datos*
 - *Bases de datos de Objetos*
 - *Bases de datos No-SQL*
 - *Bases de datos Relacionales*



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Memoria

- *“La persistencia en memoria es la capacidad de un dato u objeto de seguir existiendo luego de ser utilizado en distintas operaciones.”*
- Si bien la memoria RAM es un medio volátil, lo que se contradice con el concepto de persistencia puro; existen, también, memorias persistentes.



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Memoria

- *Este tipo de persistencia es utilizado mayormente para realizar tests en los sistemas.*
- *También existen implementaciones de Bases de Datos que persisten sus datos en memoria, tal como [SQLite](#).*



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Archivos

La persistencia en Archivos involucra guardar el estado de un Sistema en uno o varios Archivos para que luego éste pueda ser recuperado, y el Sistema pueda continuar funcionando/ejecutando desde el mismo punto, es decir, como “si nada hubiera pasado”.



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Archivos

Existen varios tipos de Archivos/Formatos de intercambio que son utilizados para persistir o transmitir datos, tales como:

- XML
- CSV
- Ancho Fijo
- JSON
- Otros...



ESTRATEGIAS DE PERSISTENCIA

Prevalencia

Es una técnica que almacena el Estado de un Sistema en la Memoria principal, pero que regularmente genera “snapshots” a disco para evitar la pérdida completa de los datos.

Las implementaciones de esta técnica tienen la capacidad de manejar transacciones.



ESTRATEGIAS DE PERSISTENCIA

Prevalencia

Una de las principales ventajas es que no existen transformaciones de datos, ya que éstos son persistidos en el formato que el aplicativo utiliza o “entiende”. Como consecuencia se obtiene un alto grado de transparencia y una mejora notable en la performance.



ESTRATEGIAS DE PERSISTENCIA

Prevalencia

Como desventaja se destaca la falta de interoperabilidad de los datos, además de que debe considerarse una alta capacidad de Memoria ya que la misma debe poder alojar al aplicativo completo.



ESTRATEGIAS DE PERSISTENCIA

Prevalencia

Prevayler es una posible implementación para Java.



ESTRATEGIAS DE PERSISTENCIA

Ortogonal

La persistencia Ortogonal refiere a que la Persistencia del Estado de un Sistema se implementa como una propiedad intrínseca de su entorno de ejecución.

Por este motivo, no se requieren acciones específicas para poder guardar o recuperar datos, ya que esto ocurre de forma “transparente”.



ESTRATEGIAS DE PERSISTENCIA

Ortogonal

Este tipo de persistencia es adoptado por los Sistemas Operativos, permitiendo funcionalidades como la Hibernación; y en Sistemas de Virtualización de Plataformas como VMware o VirtualBox.



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Base de Datos

- *En este caso, el Estado de un Sistema es persistido en una (o varias) Base de Datos.*
- *Los datos persistentes del aplicativo, dependiendo el tipo de Base de Datos, necesitan una transformación antes de ser persistidos.*



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Base de Datos

- *La interoperabilidad es un atributo de calidad que se maximiza en este caso, ya que los datos se persisten de forma independiente a los Sistemas que los manipulan.*
- *Las bases de datos otorgan mecanismos para recuperarse en caso de fallos, además de brindar reglas para resguardar la integridad de los datos.*



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Base de Datos Orientadas a Objetos

- Cada una de las Bases de Datos Orientadas a Objetos está atada a un lenguaje de programación en particular, ya que en cada lenguaje los objetos tiene una representación interna diferente.
- Como ventaja se destaca la no transformación de los datos, cuya característica proporciona una mayor performance.



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Base de Datos Orientadas a Objetos

- *La principal desventaja radica en la interoperabilidad de los datos.*



ESTRATEGIAS DE PERSISTENCIA

Persistencia en Base de Datos Relacionales y No-SQL

Antes de entrar en esta sección, es necesario conocer Teoría sobre el Modelo Relacional.



MODELO RELACIONAL



MODELO RELACIONAL

- Fue postulado por Edgar Frank Codd (IBM) en 1970.
- Surge como una nueva forma de organizar los datos.
- Todos los datos son almacenados en **relaciones**. Cada relación es un conjunto de datos organizados, llamados **tuplas**.
- Facilita la organización de grandes volúmenes de datos y garantiza la **integridad** de los mismos.

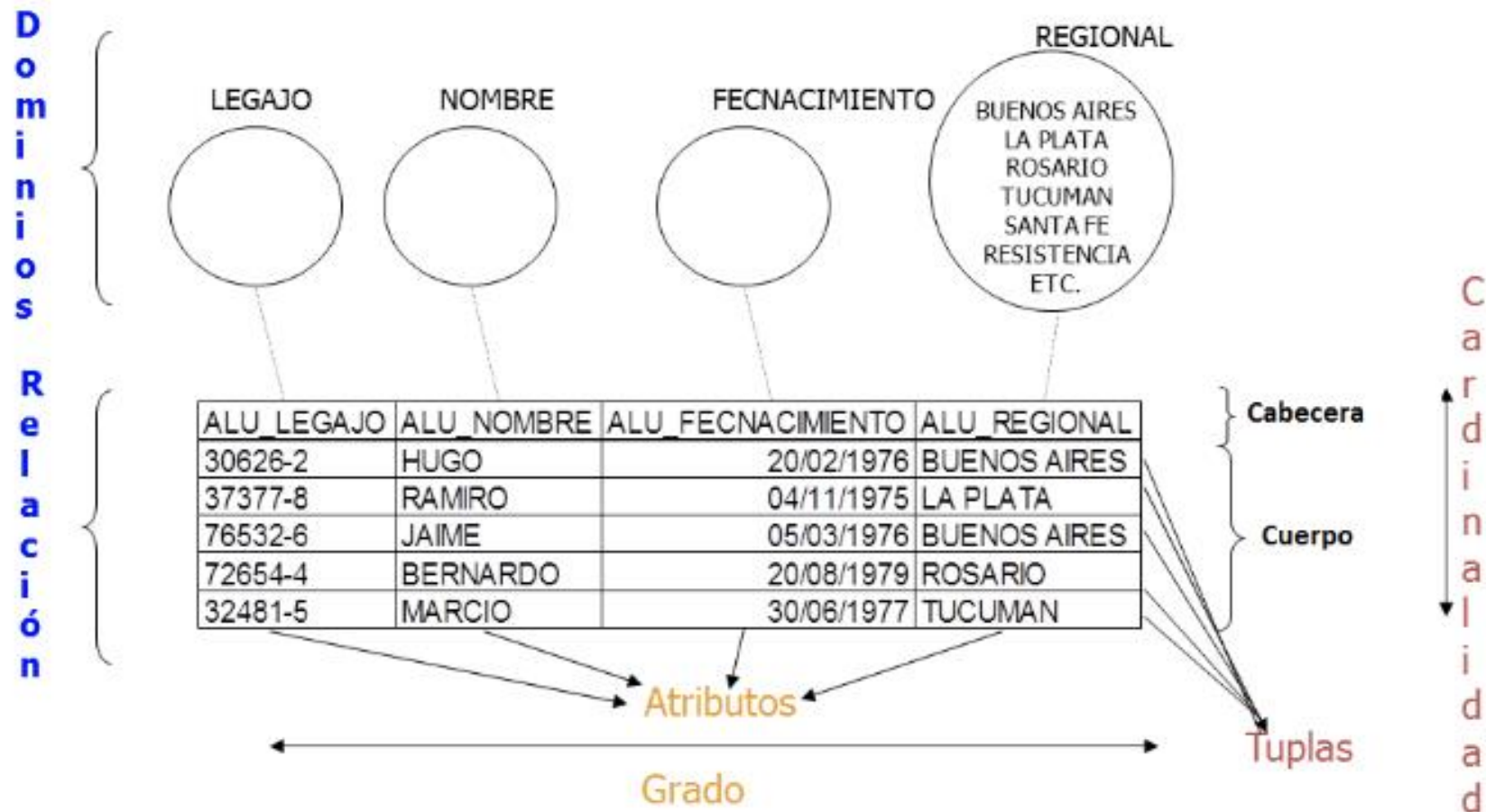
MODELO RELACIONAL

- **Relación:** es la “entidad” donde se almacenan los datos.
- **Tupla:** es el registro.
- **Atributo:** es un Campo dentro de la Tupla.

MODELO RELACIONAL

- ***Dominio***: es el conjunto de valores posibles que puede tomar un Atributo. Es la menor unidad semántica de información.
- ***Cardinalidad***: número de tuplas.
- ***Grado***: número de Atributos.
- ***Clave Primaria***: es el identificador único de cada tupla.

MODELO RELACIONAL





BASE DE DATOS RELACIONAL

- *Es un tipo de Base de Datos que cumple con el Modelo Relacional.*
- *En una BDR, todos los datos se almacenan y se accede a ellos por medio de relaciones previamente establecidas.*
- *Las relaciones que almacenan datos son llamadas “**relaciones base**” y su implementación es llamada “**tabla**”.*



BASE DE DATOS RELACIONAL

En una BDR los datos están organizados en relaciones llamadas Tablas, donde cada tupla es representada por una Fila, que a su vez contiene múltiples Columnas (atributos), y donde cada celda puede tomar alguno de los valores definidos en su dominio.



BASE DE DATOS RELACIONAL

Clave Primaria

Atributo que identifica de manera unívoca a cada fila de la tabla. Existen de 3 tipos: Naturales, Subrogadas o Compuestas.



BASE DE DATOS RELACIONAL

Índices

Es una estructura de datos que mejora la velocidad de las operaciones, por medio de un identificador único de cada fila de una tabla. Permite un rápido acceso a los registros de una tabla en una base de datos.



BASE DE DATOS RELACIONAL

Clave Foránea

Es un grupo de una o más columnas en una tabla que referencian a la clave primaria de otra tabla. Una Foreign Key puede ser parte de una Primary Key.

Tanto las PK como las FK son índices



BASE DE DATOS RELACIONAL

Integridad Referencial

La integridad referencial es una propiedad que establece que la clave externa de una tabla de referencia siempre debe corresponder a una fila válida de la tabla a la que se haga referencia. La integridad referencial garantiza que la relación entre dos tablas permanezca sincronizada durante las operaciones de actualización y eliminación.



BASE DE DATOS RELACIONAL

Tipos de Datos más utilizados

- *VARCHAR: Cadenas de caracteres compuestas por letras, números y caracteres especiales*
- *INTEGER: Números enteros naturales*
- *DOUBLE: Valores numéricos con punto flotante*
- *DATETIME: Formato para manejo de fechas*



BASE DE DATOS RELACIONAL

Constraints

Son restricciones al modelo que se utilizan para limitar el tipo de dato que puede ingresarse en una tabla. Se especifican cuando la tabla se crea por primera vez, o posteriormente realizando una actualización.



BASE DE DATOS RELACIONAL

Constraints más comunes

- **Not Null:** No acepta valores nulos
- **Unique:** No acepta valores repetidos
- **Check** (o enum): Verifica que los valores cumplan determinada condición
- **Primary Key:** Clave Primaria
- **Foreign Key:** Clave Foránea



BASE DE DATOS RELACIONAL

Relaciones

- Son asociaciones entre tablas.
- Se describen en la estructura de la Base de Datos.
- Las relaciones describen cómo se vinculan una o más tablas entre sí.



BASE DE DATOS RELACIONAL

Cardinalidad y Modalidad

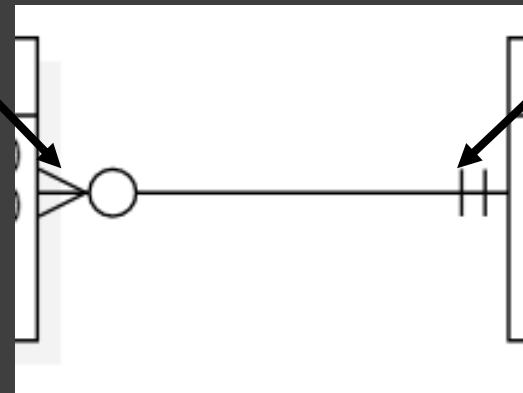
- La **cardinalidad** es la cantidad máxima de relaciones que tiene un registro de una entidad (tabla) con respecto a la otra tabla.
- La **modalidad** es la cantidad mínima de relaciones que tiene un registro de una entidad (tabla) con respecto a la otra tabla.

BASE DE DATOS RELACIONAL

Cardinalidad y Modalidad

Cardinalidad

Modalidad





BDR – ONE TO ONE

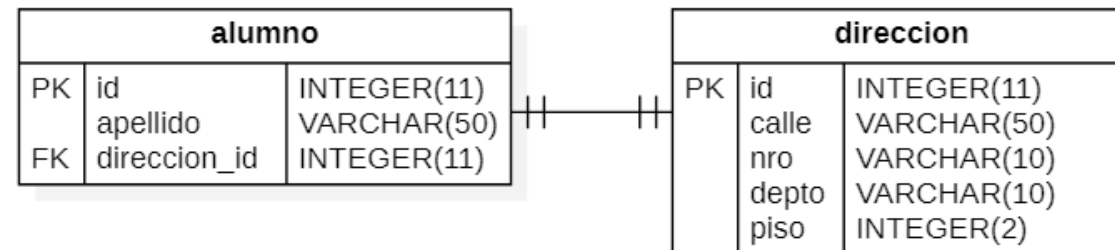
Relación One to One

- Es una relación entre dos entidades (tablas) A y B, en la cual un registro de la entidad A se corresponde con un único registro de la entidad B.
- La relación puede ser unidireccional, en cualquier sentido, o bidireccional.

BDR – ONE TO ONE

Relación One to One

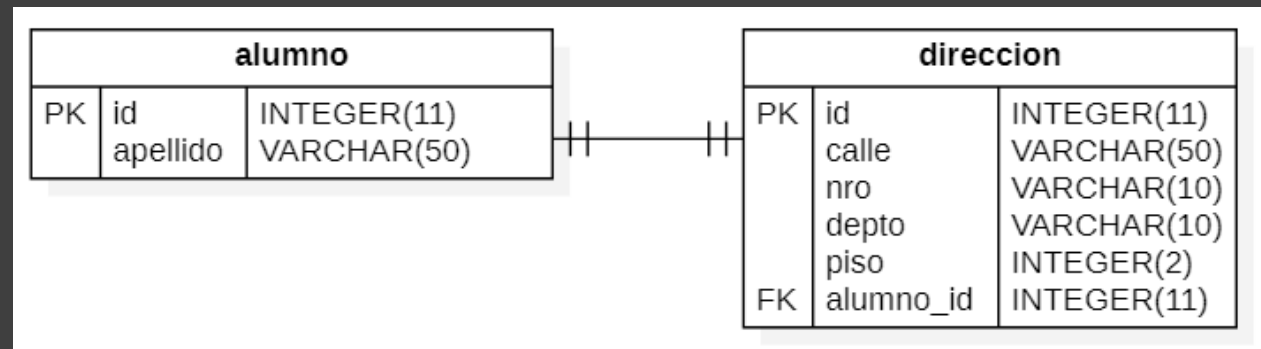
- **Unidireccional desde el lado A:** existe un registro en la tabla B que está siendo referenciado por un único registro de la tabla A.



BDR – ONE TO ONE

Relación One to One

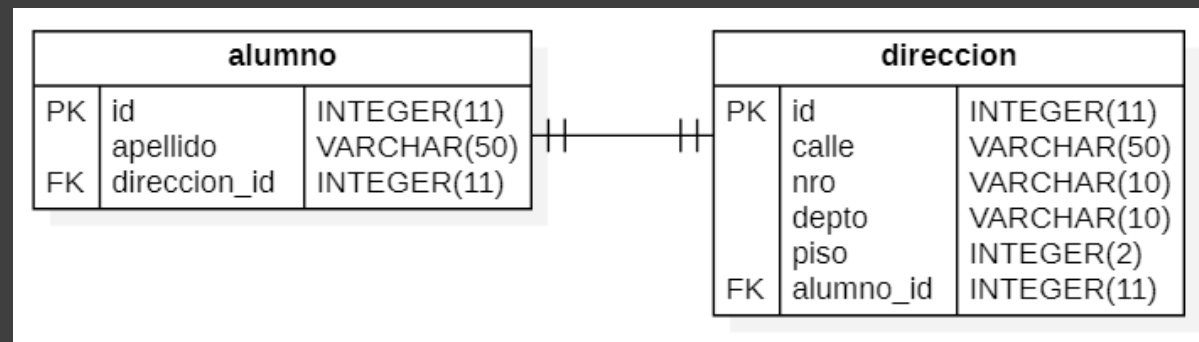
- **Unidireccional desde el lado B:** existe un registro en la tabla A que está siendo referenciado por un único registro de la tabla B.



BDR – ONE TO ONE

Relación One to One

- **Bidireccional:** existe un registro en la tabla A que está siendo referenciado por un único registro en la tabla B; y a su vez, este registro de la tabla B está siendo referenciado por el registro de la tabla A.





BDR – ONE TO MANY

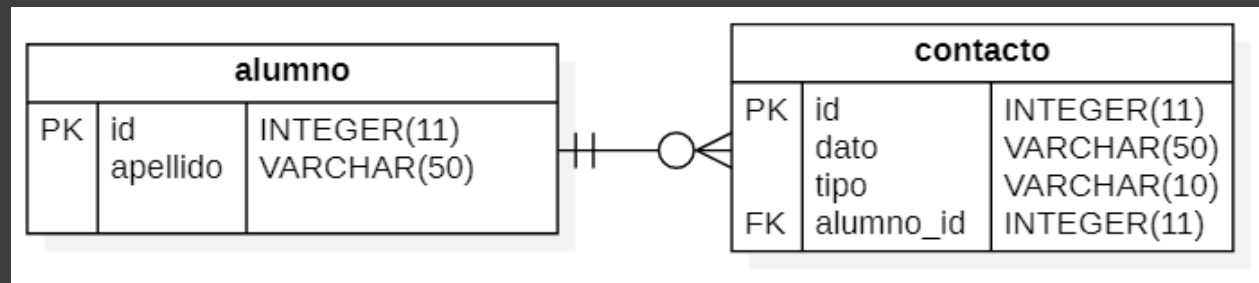
Relación One to Many

- Es una relación entre dos entidades (tablas) A y B en la cual un registro de la entidad A está siendo referenciado por muchos registros de la entidad B.
- Significa que “**A tiene muchos B**”, porque cada registro de la tabla B pertenece a un registro de la tabla A.

BDR – ONE TO MANY

Relación One to Many

“Un alumno puede tener muchos contactos”



BDR – MANY TO ONE

Relación Many to One

- Es una relación entre dos entidades (tablas) A y B en la cual muchos registros de la entidad A están referenciado al mismo registro de la entidad B.
- Significa que “**A pertenece a un B, y B tiene muchos A**”, porque cada registro de la tabla A referencia a un registro de la tabla B, pero B puede ser referenciado muchas veces.
- También se puede leer como “**Muchos A pertenecen al mismo B**”



BDR – MANY TO ONE VS ONE TO MANY

Many to One vs One to Many

Si se tienen en cuenta dos entidades A y B podríamos afirmar que:

- Si A tiene una relación One to Many con la entidad B, entonces B tiene una relación Many to One contra la entidad A.
- Si A tiene una relación Many to One contra la entidad B, entonces B tiene una relación One to Many contra la entidad A.



BDR – MANY TO ONE VS ONE TO MANY

Many to One vs One to Many

La relación entre las entidades A y B, en este caso, pueden llamarse de una u otra forma dependiendo el “lado” que miremos. En otras palabras, es una cuestión de perspectiva.

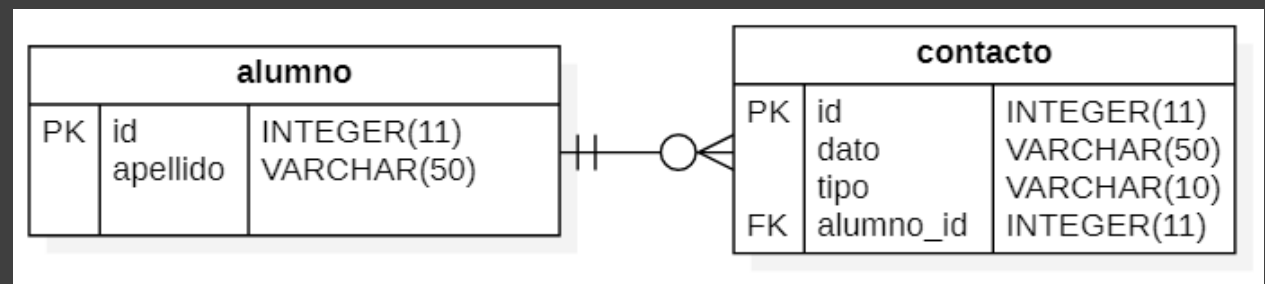
Regla práctica: si existe una relación One to Many, visto desde A hacia B, entonces “del otro lado” existe una relación Many to One (visto desde B hacia A). La inversa también es válida.

BDR – MANY TO ONE VS ONE TO MANY

Many to One vs One to Many

“Un contacto pertenece a un alumno, y un alumno puede tener muchos contactos”.

“Muchos contactos pueden pertenecer al mismo alumno”





BDR – MANY TO MANY

Relación Many to Many

- Es una relación entre dos entidades (tablas) A y B en la cual un registro de la entidad A puede estar siendo apuntado por muchos registros de la entidad B, y un registro de la entidad B puede estar siendo apuntado por muchos registros de la entidad A.
- Significa que “**A tiene muchos B, y B tiene muchos A**”.



BDR – MANY TO MANY

Relación Many to Many

- Esta relación no es posible realizarla en el modelo relacional, ya que sería necesario tener múltiples FKs (sin saber el número exacto) en ambas tablas involucradas en la relación.



BDR – MANY TO MANY

Relación Many to Many

- Para poder implementar esta relación se debe partir la misma en dos relaciones One To Many.
- La entidad A debe tener una relación One to Many contra la entidad C, considerada “*entidad intermedia*”; al igual que la entidad B.



BDR – MANY TO MANY

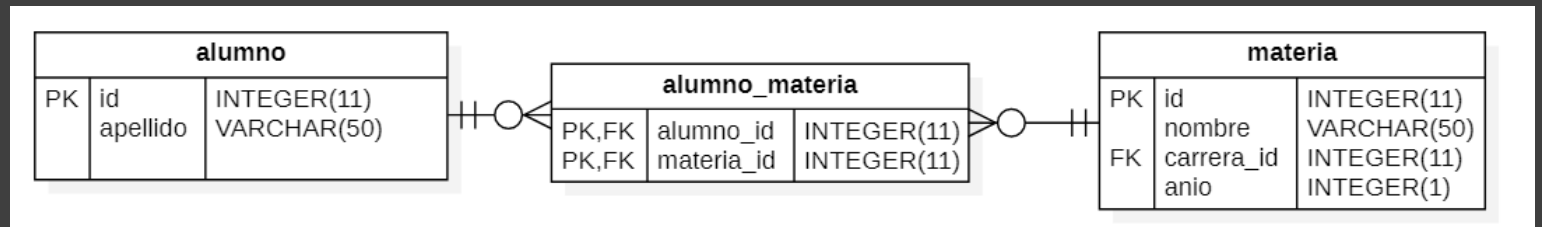
Relación Many to Many

- La entidad C, famosa por ser “*la tabla intermedia*”, debe poseer una FK a la entidad A y una FK a la entidad B.
- Además, la entidad C podría guardar algún dato extra necesario de la relación.

BDR – MANY TO MANY

Relación Many to Many

“Un alumno puede estar cursando varias materias, y una materia puede ser cursada por muchos alumnos”





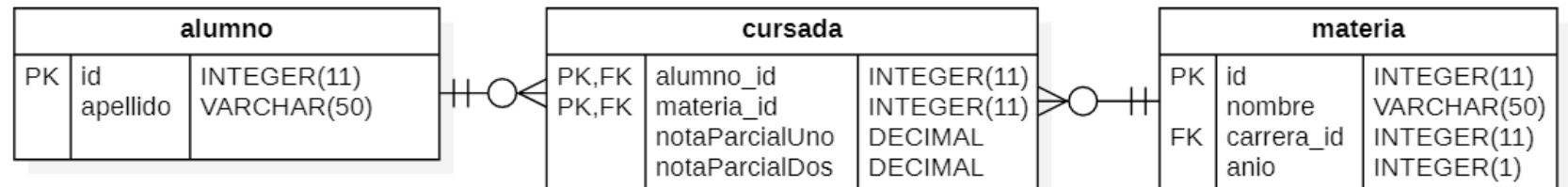
BDR – MANY TO MANY

Relación Many to Many

- En el caso anterior, suponemos que no interesa guardar ningún dato extra sobre las materias que está cursando un alumno en particular.
- Si el dominio lo amerita y necesitamos guardar algún dato extra, como las notas del primer y segundo parcial, el diseño se podría mejorar y “*alumno_materia*” dejaría de ser una simple “*tabla intermedia*”.

BDR – MANY TO MANY

Relación Many to Many



BIBLIOGRAFÍA

- Design the infrastructure persistence layer – Microsoft – En línea [[Artículo](#)]
- Diseño de Datos: Modelo Relacional – Zaffaroni Juan – En línea [[Documento](#)]
- Diseño de Datos: Mapeo Objetos Relacional – Dodino Fernando, Bulgarelli Franco – En línea [[Documento](#)]
- Guía de persistencia de JPA – Bulgarelli Franco, Prieto Gastón – En línea [[Documento](#)]
- Hibernate Inheritance Mapping – Baeldung – En línea [[Sitio Web](#)]
- Hibernate Second Level Cache – Baeldung – En línea – [[Sitio Web](#)]
- Introducción a las bases NoSQL – Dodino Fernando, Tesone Pablo, Bulgarelli Franco – En línea [[Documento](#)]
- Introducción a los Sistemas de Base de Datos, C.J Date, Edit: Pearson, 2001.
- Overview of JPA/Hibernate Cascade Types – Baeldung - En línea [[Sitio Web](#)]