**Unidad 1: Introducción a las Bases de Datos**

**Lección 1: Que son las bases de datos**

Antes de definir que es una base de datos repasemos algunos conceptos básicos:

**Dato:** Un dato es la unidad mínima de información. Es atómico y posee un significado por sí solo. Los datos pueden ser números, letras o símbolos que describen objetos, condiciones o situaciones. Ejemplos: Casa, carro, uno, azul, alto, alegre, etc.

**Información:** Es un conjunto de datos organizados con una secuencia lógica y posee un significado para la persona o sistema que recibe dicha información.

Ejemplo: La casa de la colina es roja y está hecha de madera.

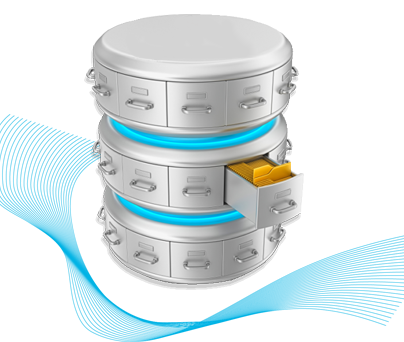
Pregunta: La frase “Arriba ocho de inteligencia en el inclusive amanece.” puede ser considerado como información?

Respuesta: No. Aunque es un conjunto de datos, no posee una secuencia lógica y organizada, tampoco posee un significado coherente para quien recibe dicho mensaje.

**Sistema de Información:** Como todo sistema, este está compuesto por una serie de elementos, cada uno con una función definida y que trabajan coordinadamente para administrar información.

Un buen sistema de información debe brindar los medios o mecanismos para poder ingresar la información, consultarla u obtenerla posteriormente, editarla o cambiarla y procesarla para generar nuevos datos.

**Base de Datos**

Una base de datos es similar a un almacén o bodega donde se almacena grandes cantidades de información la cual deseamos recuperar posteriormente. Por ejemplo, imaginemos que tenemos una empresa que vende algún tipo de producto y deseamos tener almacenados los datos personales y los números de teléfono de todos nuestros clientes, que posiblemente sean millones de personas; para ello necesitaríamos algún medio o sistema donde podamos registrar dicha información. Aunque en su forma más simple podríamos considerar la información guardada en un archivo ya sea de texto, una hoja de cálculo u otros, como una base de datos; veremos que existe una serie de características deseables en toda base de datos tales como:

|  |  |
| --- | --- |
| Título del recurso educativo | Base de datos |
| Descripción: | 1 imagen representación de una base de datos |
| Unidad | 1 |
| Lección | 1 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/bd.png |

* **Seguridad:** Esta característica asegura que sólo personas autorizadas puedan acceder a la información.
* **Integridad:** Esta característica es necesaria para que la información se mantenga sin pérdidas de datos.
* **Independencia**: Esta característica es fundamental ya que una buena base de datos debería ser independiente del sistema operativo o programas que interactúen con ella.
* **Consistencia:** Esta característica indica que la información se guarde sin duplicidades y de manera correcta.

**Lección 2: Sistemas de Bases de datos frente a los sistemas de archivos**

Antes de la llegada de los sistemas de gestión de bases de datos (SGBD), las empresas normalmente almacenaban la información usando sistemas de archivos, pero mantener la información allí no es lo más adecuado ya que tiene una serie de inconvenientes importantes:

**Redundancia de información:** La redundancia tiene que ver con la repetición de información. En un sistema de archivos es difícil evitar estas repeticiones. Los sistemas de bases de datos aseguran que esta redundancia sea mínima, lo cual implica también un menor consumo de espacio en disco duro.

|  |  |
| --- | --- |
| Título del recurso educativo | Redundancia de información |
| Descripción: | 1 imagen representación de Redundancia |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/redundancia.jpg |

**Dificultad en el acceso a los datos:** En el sistema de archivos, la obtención, consulta y modificación de los datos se realiza de manera poco práctica y eficiente y se dificulta más a medida que crece la información. Los sistemas de Bases de datos poseen mecanismos para realizar consultas para extraer la información como esta sea requerida y de manera rápida.

|  |  |
| --- | --- |
| Título del recurso educativo | Dificultad Acceso a datos |
| Descripción: | 1 imagen Dificultad Acceso a datos |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/accesodatos.jpg |

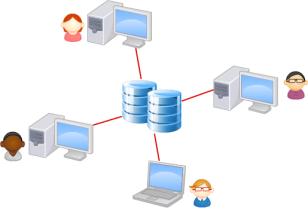
**Problemas de seguridad:** En los sistemas de bases de datos se pueden definir los usuarios que pueden acceder a la información o incluso determinar que porción de información puede ver cada tipo de usuario. Esto es algo más difícil de establecer en los sistemas de archivos, ya que en general son de acceso menos restringido y se pueden copiar más fácilmente.

|  |  |
| --- | --- |
| Título del recurso educativo | Seguridad de la información |
| Descripción: | 1 imagen de seguridad de la información |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/seguridad\_informacion.jpg |



**Problemas de integridad:** Los valores de los datos almacenados en la deben satisfacer ciertas restricciones de consistencia. La información no se puede perder o sufrir cambios que le hagan perder coherencia.

|  |  |
| --- | --- |
| Título del recurso educativo | Integridad de la información |
| Descripción: | 1 imagen de integridad de la información |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/integridad\_informacion.jpg |

**Dificultad en el acceso concurrente:** El acceso concurrente significa que varios usuarios puedan acceder a la información al mismo tiempo y desde diferentes lugares. Esta característica es difícil de lograr con los sistemas de archivos tradicionales y fácilmente realizables con la mayoría de sistemas de bases de datos.

|  |  |
| --- | --- |
| Título del recurso educativo | Acceso Concurrente |
| Descripción: | 1 imagen de acceso concurrente |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/acceso\_concurrente.png |

Estas son algunas de las razones por las cuales se prefiere un sistema de base de datos frente a los sistema de archivos para administrar la información en una empresa, sin embargo entre las desventajas de estas podríamos mencionar los costos mayores de implementación y la necesidad de personal especializado para administrarla.

**Lección 3:** **Sistemas de Gestión de bases de datos**

Un sistema de gestión de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

**Algunos de gestores de Bases de Datos**

**1. MySQL:** Cuya sigla en inglés se traslada a My Structured Query Language o Lenguaje de Consulta Estructurado. MySQL es un sistema de gestión de bases de datos relacional y es un software de código abierto ampliamente utilizado en todo el mundo. Existe también una versión comercial, y en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario.

|  |  |
| --- | --- |
| Título del recurso educativo | Logo MySQL |
| Descripción: | 1 imagen del logo de MySQL |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/logo\_mysql.png |

**2. Oracle:** Una base de datos Oracle es una colección de datos tratada como una unidad. El propósito general es almacenar y recuperar información relacionada. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando el soporte de transacciones, estabilidad, escalabilidad, y soporte multiplataforma.

|  |  |
| --- | --- |
| Título del recurso educativo | Logo Oracle |
| Descripción: | 1 imagen del logo de Oracle |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/logo\_oracle.png |

**3. PostgreSQL:** Es un Sistema de gestión de bases de datos relacional orientado a objetos. PostgreSQL es un proyecto de código abierto el cual no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales.

|  |  |
| --- | --- |
| Título del recurso educativo | Logo PostgreSQL |
| Descripción: | 1 imagen del logo de PostgreSQL |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/logo\_postgresql.jpg |



**4. Microsoft** **Access:** Microsoft Access es un sistema de gestión de bases de datos incluido en el paquete ofimático de Microsoft Office. Una base de datos de Access almacena la informacion en un solo archivo, junto con otros objetos, como formularios, informes, macros y módulos.

|  |  |
| --- | --- |
| Título del recurso educativo | Logo Microsoft Access |
| Descripción: | 1 imagen del logo de Microsoft Access |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/access.png |

**Lección 4: Conceptos de modelos lógicos y conceptuales**

Un modelo es una representación simbólica de un objeto, sistema, idea o cualquier cosa del mundo real. Nos ayuda a abstraer o a entender más fácilmente la realidad. Un requerimiento básico para cualquier modelo, es que debe describir al sistema con suficiente detalle para hacer predicciones válidas sobre el comportamiento de este.

Como cualquier sistema, las bases de datos se pueden diseñar utilizando modelos, en este caso se utilizaran herramientas para crear modelos conceptuales y lógicos.

**Modelo conceptual:** El modelo conceptual explica de manera general un sistema. Ayuda a tener una visión simplificada sin entrar en especificaciones muy técnicas. Este modelo puede ser fácilmente entendible por cualquier persona ya que tan solo muestra una aproximación inicial de lo que se está modelando.

En el contexto de las bases de datos este debe ser el primer paso en su modelado. Este se enfoca en las relaciones entre conjuntos de datos y cómo influyen entre sí. Este modelo no depende de un sistema de gestión de base de datos en particular.

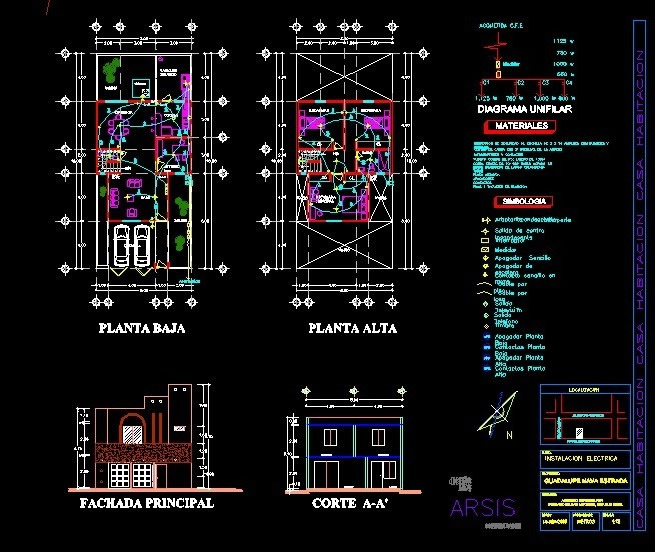
Para dar un ejemplo de un modelo conceptual imaginémonos que queremos describir la casa en la que vivimos, para ello podríamos tener un diagrama como el siguiente.

|  |  |
| --- | --- |
| Título del recurso educativo | Diagrama conceptual de una casa |
| Descripción: | 1 imagen del diagrama conceptual de una casa |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/concepto\_casa.png |

**Modelo Lógico:** El modelo lógico representa un sistema de manera más formal y técnica que el modelo conceptual. Describe el sistema de manera más específica y se acerca mucho más a la realidad. Este modelo puede ser menos entendible para una persona común pero si por el experto en el área.

En bases de datos el modelo lógico explica con detalle la estructura y otras especificaciones de esta. Este modelo puede depender del sistema de gestión de base de datos que se escoja para administrarla.

Siguiendo con el ejemplo anterior, si se quisiera describir la casa en la que vivimos de manera más específica o técnica, podríamos utilizar un diagrama como el siguiente:



|  |  |
| --- | --- |
| Título del recurso educativo | Diagrama lógico de una casa |
| Descripción: | 1 imagen del diagrama lógico de una casa |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/logico\_casa.jpg |

**WebGrafia**

https://unefazuliasistemas.files.wordpress.com/2011/04/fundamentos-de-bases-de-datos-silberschatz-korth-sudarshan.pdf

<http://www.aprenderaprogramar.com/index.php?option=com_attachments&task=download&id=500>

**Módulo 2: Modelamiento de la base de datos**

**Lección 1: Conceptos de entidades y atributos.**

Para empezar a modelar nuestra base de datos primero tenemos que obtener los requerimientos, estos determinan la información que de desea administrar desde el sistema de gestión de bases de datos. Para obtener estos requerimientos existen diferentes técnicas como son las entrevistas, lluvia de ideas, cuestionarios, reuniones grupales, etc.

Cuando ya tenemos claros los requerimientos, se procede a identificar dentro de estas, entidades y relaciones.

**Entidad:** Una entidad es una cosa u objeto del mundo real, también puede ser un concepto abstracto y es distinguible de todos los demás objetos. Una entidad tiene un conjunto de propiedades o atributos que la caracterizan.

Ejemplos: Personas, Animales, Casas, Autos, etc.

Cada uno de los ejemplos anteriores corresponde a una entidad dado que son objetos del mundo real claramente distinguibles, y como veremos adelante, a cada uno de estos se le pueden extraer propiedades o atributos.

Otro ejemplo de una entidad no tan evidente podría ser “Inscripción”

**Atributos:** Los atributos son las características o propiedades de una entidad. Cada uno de los elementos de la entidad poseen los mismos atributos y a cada atributo se le asigna un valor único por cada elemento. Tomando la entidad “Persona” como ejemplo, identificamos algunas propiedades en ella como son: identificación, nombres, apellidos, fecha de nacimiento, sexo, etc.

Un elemento en particular de la entidad persona podría ser: Sandra Toro identificada con el documento 123456, nacida el 15 de agosto de 1984 de sexo femenino.

Así mismo otro elemento de persona podría ser: Juan Pérez identificado con él documento 987654, nacido el 12 de Febrero de 1981 de sexo masculino.

Atributos

**Personas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identificación** | **Nombre** | **Apellido** | **Fecha Nacimiento** | **Sexo** |
| 123456 | Sandra | Toro | 15 Agosto 1984 | Femenino |
| 987654 | Juan | Pérez | 12 Febrero 1981 | Masculino |

Valores

|  |  |
| --- | --- |
| Título del recurso educativo | Atributos y valores |
| Descripción: | 1 imagen representación atributos con sus valores |
| Unidad | 2 |
| Lección | 1 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/tabla\_datos.PNG |

**Atributo Principal:** Cuando hemos identificado todos los atributos que hacen parte de la entidad, debemos identificar cuál de estos corresponde al atributo principal, teniendo en cuenta que este es el atributo o conjunto de atributos que hacen único a cada instancia o elemento de una entidad.

Para el caso de personas es fácil reconocer la “identificación” como el atributo principal, ya que no se encontrarán allí 2 personas diferentes que posean el mismo número de identificación. Esta característica no sucede con el resto de atributos. Si tomamos como ejemplo el nombre, veremos que no cumple la regla ya que existen varias personas que se puedan llamar igual, y así mismo para los apellidos. Del mismo modo diferentes personas podrían haber nacido en la misma fecha y ser del mismo sexo.

**Lección 2: Relaciones y Cardinalidad**

El siguiente paso el modelado luego de haber identificado las entidades y atributos, es identificar las relaciones y multiplicidades.

**Relación:** Una relación es una asociación entre diferentes entidades. Es un vínculo que nos permite definir una dependencia, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable.

Si tomamos como ejemplo la entidad “Estudiante” y la entidad “Libros” dentro del contexto de un colegio, podríamos ver que entre estas existe una relación, la cual llamaremos “Prestar”, dado que el estudiante presta libros y análogamente estos libros son prestados por estudiantes.

**Tipos de relación:**

* **Reflexiva:** Esta se da cuando una entidad se relaciona consigo misma. Tomando como ejemplo la entidad persona, esta podría ser reflexiva a través de la relación “Trabajar”, dado que una persona puede trabajar para otra persona.
* **Binaria:** Esta se presenta cuando existe una relación únicamente entre 2 entidades. Como ejemplo consideremos las entidades “Persona” y “Auto” las cuales se pueden relacionar a través de la relación “Poseer”.
* **N-Aria:** En este caso en una misma relación intervienen más de 2 entidades. Como ejemplo tomemos las entidades “Periodista”, “Articulo” y “Periódico” las cuales se pueden relacionar a través de la relación “Escribir”.

**Cardinalidad**: Llamada también multiplicidad. Indica la cantidad de elementos o instancias de una entidad A que se relacionan con una instancia de una Entidad B y viceversa. Esta puede ser de 3 tipos.

* **Uno a Uno (1:1):** Esta se da cuando un registro o elemento de una entidad A se relaciona únicamente con un solo registro de una entidad B y viceversa.

**Ejemplo:** Tomando la entidad “Profesor” y la entidad “Cubículo”, con la relación “Asignar”, tenemos que, a un profesor se le asigna solo un cubículo y un cubículo se le asigna a un solo profesor.

* **Uno a Muchos (1:M):** Esta se da cuando un registro de una entidad A se relaciona con cero o varios registros de una entidad B,y cada registro de la entidad B se relaciona únicamente con un registro de la entidad A.

**Ejemplo:** Si se tiene las entidades “Ciudad” y “País” con la relación “Pertenecer” veríamos que a un país pertenecen muchas ciudades pero cada una de esas ciudades pertenece a un solo país. Tomando un caso en particular, en Colombia hay muchas ciudades entre ellas Medellín, pero Medellín solo se encuentra en un solo país, el cual es Colombia

* **Muchos a Muchos (N:M):** Esta se da cuando un registro de una entidad A se relaciona con cero o varios registros de una entidad B, y un registro de una entidad B se relaciona con cero o varios registros de la entidad A.

**Ejemplo:** En el caso de los “Estudiantes” y los “Libros”, existe una relación de muchos a muchos, dado que un estudiante puede prestar varios libros en el transcurso del año, y a su vez cada libro es prestado por muchos estudiantes.

**Lección 3: Modelo entidad Relación**

Un modelo entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de una base de datos así como sus interrelaciones y propiedades. La representación de este se realiza a través de un diagrama con una simbología definida.

**Elementos del modelo entidad relación**

**Entidades:** Estas se representan gráficamente mediante un rectángulo y su nombre va en el interior.

**LIBRO**

**MASCOTA**

**PERSONA**

Ejemplos:

|  |  |
| --- | --- |
| Título del recurso educativo | Entidades |
| Descripción: | 1 imagen representación de las entidades |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/entidades.PNG |

**Atributos:** Los atributos se representan gráficamente mediante un círculo o elipse, su nombre va en el interior y van unidos con una línea a su respectiva entidad. El atributo principal se identifica con una línea que subraya su nombre.

Ejemplo:

**PERSONA**

|  |  |
| --- | --- |
| Título del recurso educativo | Atributos de una entidad |
| Descripción: | 1 imagen representación de entidad con atributos |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/atributos.PNG |

Para este ejemplo, la cédula es el atributo principal.

**Relaciones:** Las relaciones se representan gráficamente mediante un rombo, su nombre va en el interior y une una o mas entidades.

Ejemplos:

**PRESTAR**

**LIBRO**

**ESTUDIANTE**

|  |  |
| --- | --- |
| Título del recurso educativo | Relaciones |
| Descripción: | 1 imagen representación de una relación |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/relación.PNG |

**Cardinalidad:** Finalmente para que nuestro modelo entidad relación esté completo, nos falta representar las cardinalidades en cada relación. Aunque existen diferentes formas de hacerlo, utilizaremos la sintaxis vista en la lección 2.

Ejemplos:

1:1

1

1

**PROFESOR**

**ASIGNAR**

**CUBICULO**

1:M

**PERTENECER**

M

1

**CIUDAD**

**PAIS**

M

N

N:M

**LIBRO**

**ESTUDIANTE**

**PRESTAR**

|  |  |
| --- | --- |
| Título del recurso educativo | Cardinalidad o multiplicidad |
| Descripción: | 1 imagen representación de las cardinalidades |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/cardinalidades.PNG |

Para entender mejor este concepto, tener en cuenta que cada relación se debe leer en ambos sentidos, y así determinar la cardinalidad en cada extremo. También tener en cuenta que siempre se toma un registro de una entidad y se observa con cuantos registros se puede relacionar como máximo en la otra entidad y viceversa.

Si leemos el ejemplo de país y ciudad de izquierda a derecha, decimos que a un país pertenecen **muchas** ciudades. De allí surge la “M” al lado de la entidad Ciudad. Si lo leemos de derecha a izquierda, decimos que una ciudad específica pertenece a **un** único país. De allí surge el “1” al lado de la entidad País

Para ver un ejemplo completo de este modelo, consideremos diseñar un diagrama entidad relación para el siguiente ejercicio.

**Enunciado del problema:**

* En una universidad las facultades son dirigidas por un decano y a su vez, un decano dirige una facultad.
* Cada facultad cuanta con una serie de docentes, pero cada docente solo puede pertenecer a una facultad.
* Cada docente dicta varias asignaturas, así mismo una misma asignatura la pueden dictar docentes diferentes.
* Los estudiantes inscriben las asignaturas que le corresponde cada semestre.
* De las facultades interesa saber cuál es su nombre, ubicación y número de bloque.
* De los decanos se requiere la cedula, nombres, apellidos y celular.
* De los docentes se debe registrar la cedula, nombres, apellidos y título.
* Cada asignatura tiene un código, nombre y número de créditos.
* De los estudiantes se debe registrar identificación, nombres, apellidos y dirección de residencia

Teniendo esta información lo primero que debemos hacer es sacar un listado de las entidades y relaciones que allí identifiquemos.

**Entidades:** Facultad, Decano, Docente, Asignatura y Estudiante.

**Relaciones:**

* Dirigir (Entre Facultad y Decano).
* Pertenecer (Entre Facultad y Docente).
* Dictar (Entre Docente y Asignatura).
* Inscribir (Entre Asignatura y Estudiante).

**Nota:** El objeto de estudio que se va a modelar, en este caso la universidad, no se modela, por esa razón “Universidad” no se tuvo en cuenta como una entidad a pesar que cumple con las características para serlo.

Vamos a plasmar las entidades y relaciones encontradas en el diagrama utilizando la simbología antes vista y al final agregaremos los atributos y las cardinalidades.

1:M

M

1

**Pertenecer**

**DOCENTE**

**FACULTAD**

N

1

1:1

N:M

**Dicta**r

**Dirigi**r

M

N

M

1

N:M

**Inscribir**

**ESTUDIANTE**

**DECANO**

**ASIGNATURA**

|  |  |
| --- | --- |
| Título del recurso educativo | Diagrama Entidad relación |
| Descripción: | 1 imagen diagrama entidad relación |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/modelo\_er.PNG |

**Lección 4: Modelo Relacional**

El siguiente paso en el modelado de la base de datos, es la creación del modelo relacional. Este representa la base de datos por medio de tablas en las cuales se almacenará la información. El diagrama del modelo relacional se crea a partir del diagrama entidad relación siguiendo una serie de reglas.

**Reglas para pasar del modelo entidad relación al relacional**

**1.** Todas las entidades del modelo entidad relación se convierten en tablas en el modelo relacional, y sus atributos en campos de la tabla. Como ejemplo tomaremos la entidad persona con sus atributos y los representaremos así:

|  |
| --- |
| Personas |
| -Cédula  **Tabla**  -Nombres  **Campos**  -Apellido  -Genero |

|  |  |
| --- | --- |
| Título del recurso educativo | Modelo relacional - Tablas |
| Descripción: | 1 imagen de una tabla en el modelo relacional |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/tabla.PNG |

**2.** El atributo principal de cada entidad en el modelo entidad relación lo llamaremos llave primaria en el modelo relacional y por facilidad lo representaremos con las letras PK (Primary Key), las cuales irán a continuación del nombre del campo. Para el caso de personas el atributo principal es la cédula, por lo tanto su representación dentro de la tabla será así:

|  |
| --- |
| Personas  **Llave primaria** |
| -Cédula PK  -Nombres  -Apellido  -Género |

|  |  |
| --- | --- |
| Título del recurso educativo | Modelo relacional – llave primaria |
| Descripción: | 1 imagen representación de llave primaria |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/PK.PNG |

**3.** Donde exista una relación Uno a Uno (1:1) entre 2 entidades, se debe crear una llave que llamaremos foránea, en una de las 2 tablas asociadas.

Una llave foránea es una referencia a una llave en otra tabla y determina una relación existente entre 2 tablas. Una llave foránea en una tabla se asocia con una llave primaria en la otra tabla. Para agregar una llave foránea simplemente se agrega un campo más con las mismas características que el campo que es llave primaria.

Como ejemplo tomemos las entidades profesor y cubículo.

|  |
| --- |
| Profesor  1  **Campos de la tabla cubículo primaria** |
| -Cédula PK  -Nombres  -Apellido  M  -Género  **Campo foráneo** |

|  |
| --- |
| Cubículo |
| -Nro\_cubículo PK  -Descripción  -Ubicación  -Cédula FK |

|  |  |
| --- | --- |
| Título del recurso educativo | Modelo relacional – Uno a uno |
| Descripción: | 1 imagen regla relación uno a uno en relacional |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/uno\_a\_uno.PNG |

Como se ve en el diagrama, la llave primaria de profesor (*Cédula*), se agregó como llave foránea en cubículo y se unieron estos 2 campos mediante una línea. Aunque la multiplicidad era originalmente de uno a uno (1:1) en el diagrama entidad relación, veremos que siempre donde coloquemos una llave foránea la relación puede llegar a ser a muchos (M) registros.

Como la relación es de uno a uno (1:1) también hubiéramos podido hacer el siguiente diagrama:

|  |
| --- |
| Profesor |
| -Cédula PK  1  -Nombres  -Apellido  -Género  M  -Nro\_cubículo FK |

|  |
| --- |
| Cubículo |
| -Nro\_cubículo PK  -Descripción  -Ubicación |

|  |  |
| --- | --- |
| Título del recurso educativo | Modelo relacional – Uno a Muchos caso 1 |
| Descripción: | 1 imagen regla relación uno a Muchos caso 1 |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/uno\_a\_muchos\_1.PNG |

En este caso la llave primaria de la tabla Cubículo (*Nro\_cubículo*) la pasamos como llave foránea a la tabla Profesor. La multiplicidad queda uno (1) al lado de la llave primaria y muchos (M) al lado de la llave foránea.

**4.** Donde exista una relación de uno a muchos (1:M) en el modelo entidad relación, crearemos las tablas correspondientes en el modelo relacional, y donde se encontraba la multiplicidad muchos (M) se creará obligatoriamente la llave foránea de la misma forma como se hizo en el caso anterior.

Tomemos como ejemplo las entidades Persona y Mascota. En este caso asumiremos que una persona puede tener varias mascotas, pero cada mascota pertenece a una sola persona.

|  |  |
| --- | --- |
| Título del recurso educativo | Modelo relacional – Uno a Muchos caso 2 |
| Descripción: | 1 imagen regla relación uno a Muchos caso 2 |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/uno\_a\_muchos\_2.PNG |

|  |
| --- |
| Personas  1 |
| -Cédula PK  -Nombres  -Apellido  -Género  M |

|  |
| --- |
| Mascotas |
| -Id\_Mascota PK  -Nombre  -Raza  -Genero  -Cédula FK |

**5.** Donde exista una relación de muchos a muchos en el modelo entidad relación, la relación se convierte en una tabla nueva que une las 2 entidades que se estaban relacionando originalmente. La nueva tabla tendrá su propia llave primaria y tendrá 2 llaves foráneas, las cuales corresponden a las llaves primarias de las otras dos tablas. Adicionalmente tendrá los campos adicionales que sean requeridos. El nombre para la nueva tabla puede ser el mismo de la relación u otro que sea más adecuado.

Para la cardinalidad se mantiene la misma regla aplicada en los casos anteriores. Se colocará un 1 donde se encuentran las llaves primarias, y una M donde se encuentra las llaves foráneas, transformándose así en dos relaciones de uno a muchos (1:M).

Tomando como ejemplo las entidades estudiante y libro, y aplicando esta regla, se transformarían en el modelo relacional de la siguiente forma:

|  |
| --- |
| Préstamos |
| -Id\_Préstamo PK  -Nro\_Id FK  M  -Isbn FK  -Fecha\_préstamo |

|  |
| --- |
| Estudiantes  1 |
| -Nro\_Id PK  -Nombres  -Apellidos  -Dirección |

|  |
| --- |
| Libros |
| -Isbn PK  -Título  -Editorial  -Autor |

1

M

|  |  |
| --- | --- |
| Título del recurso educativo | Modelo relacional – Muchos a Muchos |
| Descripción: | 1 imagen regla relación Muchos a Muchos |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/muchos\_a\_muchos.PNG |

La nueva tabla que fue creada a partir de la relación “prestar”, se le puso el nombre préstamo, se le creó una llave primaria llamada “Id\_Préstamo”, se le agregó las dos llaves foráneas a partir de las llaves primarias de estudiantes (Nro\_Id) y libros (Isbn) y se le agregó un nuevo campo llamado “Fecha\_préstamo”.

Poniendo en práctica lo aprendido, procederemos a transformar el modelo entidad relación de la lección 3 a un modelo relacional siguiendo cada una de las reglas anteriores.

|  |
| --- |
| Docentes  1 |
| -Cédula PK  -Nombres  -Apellidos  -Título  -Número FK |

|  |
| --- |
| Facultad  1 |
| -Número PK  -Nombre  -Ubicación  -Cédula FK |

|  |
| --- |
| Decanos  1 |
| -Cédula PK  -Nombres  -Apellidos  -Celular |

M1

M

|  |
| --- |
| Curso |
| -Nro\_Id PK  M1  -Cédula FK  -Código FK  M1 |

|  |
| --- |
| Estudiantes  1 |
| -Nro\_Id PK  -Nombres  -Apellidos  -Dirección |

|  |
| --- |
| Inscripción |
| -Nro\_Inscripción PK  M1  -Código FK  -Nro\_Id FK  -Periodo |

|  |
| --- |
| Asignaturas  1 |
| -Código PK  -Nombre  -Nro\_creditos |

11

M1

|  |  |
| --- | --- |
| Título del recurso educativo | Ejercicio Modelo relacional |
| Descripción: | 1 imagen modelo relacional completo |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: imgs/Modelo\_Relacional.PNG |

**Webgrafia**

https://es.scribd.com/doc/88402619/12/Conjuntos-de-relaciones-binarias-y-n-arias

http://es.slideshare.net/Neoinquisidor/conversion-25836134

**Unidad 3: Bases de datos con MySQL**

**Lección 1: Creación de la Base de datos**

Luego de crear nuestro diagrama relacional el cual surge del modelo entidad relación, estamos listos para empezar a crear en definitiva la base de datos. En este caso utilizaremos el sistema de gestión de base de datos MySQL.

Para crear la base de datos vamos a utilizar una serie de instrucciones en un lenguaje de programación llamado SQL.

SQL, por sus siglas en inglés (Structured Query Language) o “Lenguaje de consulta estructurado”, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

**Creación de las tablas**

Lo primero que vamos a aprender es a crear cada una de las tablas identificadas en el modelo relacional, para ello utilizaremos la siguiente instrucción.

CREATE TABLE <*nombre\_tabla>* (

<nombre\_*columna\_1> < tipo\_de\_dato(tamaño)>,*

<*nombre\_columna\_2> <tipo\_de\_dato(tamaño)*>,

…

);

Todas las expresiones encerradas entre < > son las que debemos entrar a reemplazar.

En <*nombre\_tabla*> colocaremos el nombre de la tabla que estemos creando como por ejemplo: Personas, Libros, Estudiantes, etc.

En cada *<nombre\_columna\_x*> colocaremos los nombres de los campos de la tabla que se está creando.

Dependiendo la naturaleza del campo, debemos definirle un tipo de dato y un tamaño en *<tipo\_de\_dato(tamaño)>.*

**Tipos de Datos en MySQL :**Entre los muchos tipos de datos que maneja MySQL, vamos a ver los que se utilizan con más frecuencia.

* **Int**: Cuando un campo va a contener valores numéricos enteros, podemos definirlo con este tipo de dato. El rango de valores que puede contener se encuentra entre -2147483648 y 2147483647.
* **Varchar:** Si el campo va a contener texto o que es lo mismo, caracteres alfanuméricos, se puede establecer con este tipo de dato.
* **Enum**: Si los valores que el campo va a contener están claramente definidos y no son demasiados, se puede establecer con este tipo de dato. Para entender mejor esto, como ejemplo tomemos el campo “Genero”. Sabemos que los únicos valores que va a almacenar serán “Masculino” y “Femenino”, y se definiría así: enum('Femenino', 'Masculino').
* **Date**: Para los campos que requieran almacenar una fecha se le puede establecer este tipo de dato de la siguiente manera: date('1995-01-25').

**Llaves primarias:** Para indicar que un campo es llave primaria, la manera más simple de hacerlo, es colocar luego de la definición de este, la sentencia “Primary Key”.

**Llaves foráneas:** Para establecer los campos que son llaves foráneas dentro de la tabla, se coloca la siguiente instrucción después de declarar todos los campos.

FOREIGN KEY (<campo\_foraneo>) REFERENCES <tabla\_relacionada>(<campo\_relacionado>)

Donde <campo\_foraneo> corresponde al nombre de uno de los campos que es llave foránea dentro de nuestra tabla; <tabla\_relacionada> corresponde al nombre de la tabla con la cual se está asociando o relacionando nuestra tabla y <campo\_relacionado> corresponde al campo con el cual se está uniendo la llave foránea de nuestra tabla y el campo que es llave primaria en la tabla relacionada.

Esta instrucción se debe repetir por cada campo foráneo que tenga la tabla y los separamos por comas (,).

Con lo explicado podemos empezar a crear nuestra base de datos, para ello utilizaremos el ejemplo que hemos venido trabajando. Vamos a escribir el código SQL para generar las tablas decanos, facultad, docentes, curso y asignaturas.

CREATE TABLE decanos(

cedula varchar(20) primary key,

nombres varchar(200),

apellidos varchar(200),

celular varchar(20)

);

CREATE TABLE facultad(

numero int(5) primary key,

nombre\_facultad varchar(200),

ubicacion varchar(200),

cedula\_dec varchar(20),

FOREIGN KEY (cedula\_dec) REFERENCES decanos(cedula)

);

CREATE TABLE docentes(

cedula varchar(20) primary key,

nombres varchar(200),

apellidos varchar(200),

título varchar(200),

numero\_fac int(5),

FOREIGN KEY (numero\_fac) REFERENCES facultad(numero)

);

CREATE TABLE asignaturas(

codigo int(5) primary key,

nombre varchar(200),

nro\_creditos int(3)

);

CREATE TABLE curso(

nro\_curso int(5) primary key,

cedula\_doc varchar(20),

codigo\_asg int(5),

FOREIGN KEY (cedula\_doc) REFERENCES docentes(cedula),

FOREIGN KEY (codigo\_asg) REFERENCES asignaturas(codigo)

);

Tener en cuenta que el orden en que se crean las tablas es importante. No se puede crear una tabla que tenga una llave foránea apuntando a una tabla que aún no existe. Por esta razón “Curso” se creó después de “Asignaturas”, ya que “Curso” contiene la llave foránea “codigo\_asg” que apunta a “Asignaturas”. Si “Curso” se hubiera creado antes, “Asignaturas” no existiría y se generaría un error en el sistema donde se esté realizando esta creación.

**Lección 2: Agregar y modificar datos**

Ya tenemos lista nuestra estructura de Base de Datos, sin embargo esta se encuentra sin información almacenada.

**Inserción de datos:** Para ingresar información en la Base de Datos utilizamos la siguiente instrucción en MySQL:

INSERT INTO <nombre\_tabla> (<campo\_1>, <campo\_2>, … , <campo\_n>)

VALUES('<valor\_1>', '< valor\_2>', … ,'< valor\_n>');

Donde, <nombre\_tabla> corresponde al nombre de la tabla donde se van a ingresar los datos.

<campo\_1>, <campo\_2>, … , <campo\_n>, Corresponden a cada uno de los campos de la tabla en los cuales se va a ingresar valores.

'<valor\_1>', '< valor\_2>', … ,'< valor\_n>', Corresponden a los valores que se ingresarán en cada uno de los campos anteriores.

Para ver un ejemplo, la inserción de un registro en la tabla decanos, seria de la siguiente forma.

INSERT INTO decanos(cedula, nombres, apellidos, celular)

VALUES('189787650', 'Juan Andrés', 'Pérez Sánchez' , '3109876543');

Después de ejecutar por primera vez esta instrucción, la tabla decanos se verá de la siguiente forma:

**Decanos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cédula** | **Nombres** | **Apellidos** | **Celular** |
| 189787650 | Juan Andrés | Pérez Sánchez | 3109876543 |

Si quisiéramos hacer la siguiente inserción sobre la tabla facultad:

INSERT INTO facultad(numero, nombre\_facultad, ubicacion, cedula\_dec)

VALUES('001', 'Medicina', 'Bloque 3, piso 2' , '1088976543');

Esta inserción nos generaría un error, dado que en el campo “cedula\_dec”, la cual es una llave foránea asociada con el campo “cedula” de la tabla decanos, estamos tratando de enviar la cédula 1088976543, y hasta el momento en la tabla decanos no existe ningún registro con esa cedula. Esta acción estaría violando la integridad referencial que mantiene el sistema de gestión de base de datos.

Vemos entonces en este caso que la tabla facultad depende de la tabla decanos por tener una llave foránea que se asocia con esta. De este modo una inserción válida en “Facultad”, será aquella donde en el campo “cedula\_dec” se ingrese una cédula que ya exista en “Decanos”. Esto aplica para todos los casos de tablas que dependan de otras.

**Actualización de datos:** En algún momento necesitaremos modificar o actualizar la información o algún dato que previamente hemos almacenado en la base de datos. Para poder realizar esto, utilizaremos la siguiente instrucción en MySQL.

UPDATE <nombre\_tabla>

SET <campo\_1>=<valor\_1>, <campo\_2>=<valor\_2>,...,<campo\_n> =<valor\_n>

WHERE <campo\_x> = <valor\_x>;

Donde, <nombre\_tabla> corresponde al nombre de la tabla en la cual se va a actualizar los datos.

<campo\_1>, <campo\_2>, … , <campo\_n>, Corresponde a cada uno de los campos en los cuales se hará la modificación de su valor.

<valor\_1>, < valor \_2>, … , < valor\_n>, Corresponde a cada uno de los valores que se actualizarán en cada uno de los campos.

En este momento la tabla pude tener una gran cantidad de registros o filas. Para determinar a cuales registros se les debe hacer la modificación, se evalúa la condición que sigue a continuación de la palabra “WHERE”.

<campo\_x> es un campo cualquiera de la tabla y <valor\_x> es un valor cualquiera a evaluar en el campo definido. Esto quiere decir que la actualización se realizará solo en aquellos registros donde el campo <campo\_x> posea el valor <valor\_x>.

Para entender mejor esta instrucción realicemos el siguiente ejemplo.

Supongamos que tenemos la tabla decanos con la siguiente información almacenada.

**Decanos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cédula** | **Nombres** | **Apellidos** | **Celular** |
| 1234234 | Juan | Pérez | 3103216547 |
| 5678546 | Carlos | Vélez | 3126545678 |
| 3242344 | Mario | Toro | 3008464647 |
| 9585835 | Sandra | Ruiz | 3114046784 |
| 1064933 | Pedro | Flórez | 3157864667 |
| 5373548 | Diana | Sánchez | 3206766674 |
| 1324459 | Paula | Vélez | 3103858583 |

Si quisiéramos cambiarle el nombre al decano Juan Pérez por Julián Parra, utilizaríamos la siguiente instrucción:

UPDATE decanos

SET nombres='Julián', apellidos='Parra'

WHERE cedula = '1234234';

Si quisiéramos cambiar el nombre a todas las personas de apellido Vélez por Andrés, utilizaríamos la siguiente instrucción:

UPDATE decanos

SET nombres='Andrés'

WHERE apellidos = 'Vélez';

Después de ejecutar las dos instrucciones anteriores, la tabla quedaría de la siguiente manera:

**Decanos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cédula** | **Nombres** | **Apellidos** | **Celular** |
| 1234234 | Julián | Parra | 3103216547 |
| 5678546 | Andrés | Vélez | 3126545678 |
| 3242344 | Mario | Toro | 3008464647 |
| 9585835 | Sandra | Ruiz | 3114046784 |
| 1064933 | Pedro | Flórez | 3157864667 |
| 5373548 | Diana | Sánchez | 3206766674 |
| 1324459 | Andrés | Vélez | 3103858583 |

**Eliminación o borrado de registros:** En el momento de requerir borrar un registro o conjunto de registros de una tabla específica utilizaremos la siguiente instrucción:

DELETE FROM <nombre\_tabla>  
WHERE <campo\_x> = <valor\_x>;

Donde, <nombre\_tabla> corresponde al nombre de la tabla en la cual se van a eliminar los registros.

<campo\_x> es un campo cualquiera de la tabla, y <valor\_x> es un valor cualquiera a evaluar en el campo definido. Esto quiere decir que se borraran aquellos registros donde el campo <campo\_x> posea el valor <valor\_x>.

Para entender mejor esta instrucción realicemos el siguiente ejemplo.

Supongamos que tenemos la tabla decanos con la siguiente información almacenada.

Si queremos eliminar la fila o registro donde se encuentra Sandra Ruiz utilizaremos la siguiente instrucción:

DELETE FROM decanos  
WHERE cedula = '9585835';

Después de ejecutar esta instrucción la tabla anterior quedará de la siguiente manera:

**Decanos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cédula** | **Nombres** | **Apellidos** | **Celular** |
| 1234234 | Julián | Parra | 3103216547 |
| 5678546 | Andrés | Vélez | 3126545678 |
| 3242344 | Mario | Toro | 3008464647 |
| 1064933 | Pedro | Flórez | 3157864667 |
| 5373548 | Diana | Sánchez | 3206766674 |
| 1324459 | Andrés | Vélez | 3103858583 |

Como vemos, se ha eliminado el registro donde se encontraba Sandra Ruiz.

**Nota:** Tener cuidado cuando se realiza actualizaciones de datos o borrado de registros, ya que estas acciones pueden generar problemas de integridad referencial, ya que se puede realizar una modificación o eliminación incoherente en una llave de la cual depende un campo en otra tabla.

**Lección 3: Consultas simples a la base de datos**

Hasta este momento ya hemos construido la estructura de la Base de datos y la hemos llenado con información. A continuación aprenderemos como extraer o realizar consultas de información sobre esta.

La sintaxis para realizar una consulta simple sobre la base de datos es la siguiente:

SELECT <campo\_1>, <campo\_2>, … , <campo\_n>

FROM <tabla\_1>, <tabla\_1>, … ,<tabla\_n>

WHERE <condición\_1> <AND | OR > <condición\_2> <AND | OR > … <condición\_n>;

Donde <campo\_1>, <campo\_2>, … , <campo\_n> corresponde a cada uno de los campos de las diferentes tablas que se quieren mostrar en el resultado de la consulta.

<tabla\_1>, <tabla\_1>, … ,<tabla\_n> corresponde a cada una de las tablas de donde se desea extraer la información.

Cada una de las condiciones: <condición\_1>, <condición\_2>, <condición\_n>, tiene la siguiente estructura:

<campo\_x> <operador> <valor\_x> ó <campo\_x> <operador> <campo\_y>

Donde <campo\_x> y <campo\_y> son campos de cualquiera de las tablas, y <valor\_x> es un valor cualquiera a evaluar en el campo definido.

<operador> puede ser cualquiera de los operadores de comparación que se muestran a continuación.

|  |  |  |
| --- | --- | --- |
| **Op.** | **Nombre** | **Descripción** |
| < | Menor que | Teniendo A<B, indica que A sea menor que B |
| > | Mayor que | Teniendo A>B, indica que A sea mayor que B |
| <= | Menor o igual que | Teniendo A<=B, indica que A sea menor o igual que B |
| >= | Mayor o igual que | Teniendo A>=B, indica que A sea mayor o igual que B |
| <> | Diferente de | Teniendo A<>B, indica que A sea diferente de B |
| = | Igual a | Teniendo A=B, indica que A sea igual que B |

Entre dos condiciones va uno de los operadores lógicos AND ó OR. El operador AND indica que el par de condiciones que se están evaluando se deben cumplir para el resultado de la consulta, mientras que el operador OR solo requiere que uno de las 2 condiciones se cumpla.

Tomando como ejemplo la tabla de decanos original, realizaremos una consulta que nos muestre todos los decanos de apellido “Vélez”. Para esto realizaremos la siguiente consulta en MySQL:

SELECT cedula, nombres, apellidos, celular

FROM decanos

WHERE apellidos = 'Veléz';

El resultado de esta consulta sería el siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| **cedula** | **nombres** | **apellidos** | **celular** |
| 5678546 | Carlos | Vélez | 3126545678 |
| 1324459 | Paula | Vélez | 3103858583 |

Si quisiéramos consultar todos los decanos de nombre “Diana” y los decanos de nombre “Mario” crearíamos una consulta con dos condiciones separadas del operador OR de la siguiente manera.

SELECT cedula, nombres, apellidos, celular

FROM decanos

WHERE nombres = 'Diana' OR nombres = 'Mario';

El resultado de esta consulta sería el siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| **cedula** | **nombres** | **apellidos** | **celular** |
| 3242344 | Mario | Toro | 3008464647 |
| 5373548 | Diana | Sánchez | 3206766674 |

Algunos hubieran pensado usar el operador lógico **AND**, sin embargo su uso en este caso estaría mal. Recordemos que este operador exige que ambas condiciones se cumplan, y si analizamos bien, no hay posibilidad que se cumpla que al mismo tiempo una persona se llame Mario y Diana por lo cual no hubiéramos obtenido ningún resultado.

El operador **OR** funciona perfectamente en este caso, dado que si miramos el primer resultado de la consulta vemos que cumple una de las dos condiciones. Este decano se llama Mario o se llama Diana, y efectivamente su nombre es “Mario”. Para el segundo resultado sucede lo mismo.

Ahora realizaremos una consulta que traiga información de 2 tablas que se estén relacionando, para esto supongamos que la tabla facultad contiene la siguiente información.

**Facultad**

|  |  |  |  |
| --- | --- | --- | --- |
| **Número** | **Nombre\_facultad** | **Ubicación** | **Cedula\_dec** |
| 001 | Medicina | Edificio 2 piso 1 | 9585835 |
| 002 | Deportes | Edificio 3 piso 1 | 5373548 |
| 003 | Física | Edificio 2 piso 2 | 1234234 |
| 004 | Bellas Artes | Edificio 1 piso 1 | 1324459 |
| 005 | Química | Edificio 3 piso 2 | 3242344 |
| 006 | Ciencias Humanas | Edificio 1 piso 2 | 1064933 |
| 007 | Ingenierías | Edificio 1 piso 3 | 5678546 |

La consulta que realizaremos nos mostrará todos los registros de la tabla “Decanos” con los campos **cedula, nombres y apellidos**. Y de la tabla “facultad” los campos **nombre\_facultad y ubicación**, de esta manera sabremos que facultad dirige cada uno de los decanos.

Dado que estamos extrayendo información de 2 tablas que se están relacionando a través de las llaves “cedula” y “cedula\_dec”, es obligatorio adicionar a la consulta una condición donde se iguale estos dos campos como lo haremos a continuación:

SELECT cedula, nombres, apellidos, nombre\_facultad, ubicacion

FROM decanos, facultad

WHERE cedula=cedula\_dec;

El resultado de la consulta seria el siguiente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **cedula** | **nombres** | **apellidos** | **nombre\_facultad** | **ubicación** |
| 1234234 | Juan | Pérez | Física | Edificio 2 piso 2 |
| 5678546 | Carlos | Vélez | Ingenierías | Edificio 1 piso 3 |
| 3242344 | Mario | Toro | Química | Edificio 3 piso 2 |
| 9585835 | Sandra | Ruiz | Medicina | Edificio 2 piso 1 |
| 1064933 | Pedro | Flórez | Ciencias Humanas | Edificio 1 piso 2 |
| 5373548 | Diana | Sánchez | Deportes | Edificio 3 piso 1 |
| 1324459 | Paula | Vélez | Bellas Artes | Edificio 1 piso 1 |

Ahora vamos a consultar la misma información anterior pero solo para los decanos de apellido “Veléz”. Para esto construiremos la siguiente consulta:

SELECT cedula, nombres, apellidos, nombre\_facultad, ubicacion

FROM decanos, facultad

WHERE cedula=cedula\_dec AND apellidos='Vélez';

El resultado sería el siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **cedula** | **nombres** | **apellidos** | **nombre\_facultad** | **ubicación** |
| 5678546 | Carlos | Vélez | Ingenierías | Edificio 1 piso 3 |
| 1324459 | Paula | Vélez | Bellas Artes | Edificio 1 piso 1 |

En ambos casos fue necesario colocar la condición **cedula=cedula\_dec**, y después de esta, las demás condiciones que fueran necesarias.

**Lección 4: Funciones en MySQL**

En MySQL tenemos una serie de funciones que podemos agregar a una consulta. Entre ellas, tenemos funciones de agrupación, funciones matemáticas, funciones para manipulación de texto, etc. En esta lección veremos algunas de ellas.

Para explicar mejor cada una de las funciones, supongamos que tenemos una tabla “Empleados” con la siguiente información.

**Empleados**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cedula** | **Nombres** | **Apellidos** | **Genero** | **salario** |
| 98765434 | Sergio | Gálvez | M | 500000 |
| 76543275 | Raúl | Franco | M | 250000 |
| 43532658 | Viviana | Rodríguez | F | 300000 |
| 63297867 | Fernando | Muñoz | M | 400000 |
| 54637829 | Martha | Castro | F | 450000 |
| 34764456 | Rubén | Osorio | M | 350000 |

**Funciones de Agrupación:** Entre las funciones de agrupación se encuentran:

* **SUM(Campo):** Esta función calcula el total de la suma de un campo numérico específico dado. Como ejemplo vamos a obtener el total de la suma de los salarios de todos los empleados. La consulta a realizar es la siguiente:

SELECT SUM(salario) FROM empleados;

El resultado de esta consulta es el siguiente:

|  |
| --- |
| **SUM(salario)** |
| 2250000 |

Si quisiéramos obtener la suma total del salario de las mujeres, la consulta a realizar sería de la siguiente forma:

SELECT SUM(salario) FROM empleados WHERE genero='F';

Obteniendo como resultado:

|  |
| --- |
| **SUM(salario)** |
| 750000 |

* **AVG(Campo):** Esta función calcula el valor promedio de un campo numérico dado. Como ejemplo vamos a obtener el valor promedio de los salarios de todos los empleados. La consulta a realizar es la siguiente:

SELECT AVG(salario) FROM empleados;

El resultado de esta consulta es el siguiente:

|  |
| --- |
| **AVG(salario)** |
| 375000 |

* **COUNT(\*):** Esta función calcula la cantidad de registros devueltos en la consulta con las condiciones dadas. En este caso no es muy importante el campo que se le especifique a la función. Lo más común es colocar un asterisco entre los paréntesis. Como ejemplo vamos a obtener la cantidad de empleados que son hombres. La consulta a realizar es la siguiente:

SELECT COUNT(\*) FROM empleados WHERE genero = 'M';

El resultado de esta consulta es el siguiente:

|  |
| --- |
| **COUNT(\*)** |
| 4 |

**Funciones de manipulación de texto:** Entre las funciones de manipulación de texto se encuentran:

* **CONCAT(texto1,texto2,…):** Esta función une todos los textos que se envían dentro de los paréntesis. Como ejemplo vamos a unir los campos nombres y apellidos del empleado **Sergio Gálvez**, separados con un espacio. La consulta a realizar es la siguiente:

SELECT cedula, CONCAT(nombres,' ',apellidos) FROM empleados

WHERE cedula='98765434';

El resultado de esta consulta es el siguiente:

|  |  |
| --- | --- |
| **cedula** | **CONCAT(nombres,' ',apellidos)** |
| 98765434 | Sergio Gálvez |

* **UPPER(texto):** Esta función convierte a mayúscula el texto que se le envía dentro del paréntesis. Como ejemplo vamos a convertir a mayúscula los nombres y apellidos de las mujeres en la tabla “Empleados”. La consulta a realizar es la siguiente:

SELECT UPPER(nombres), UPPER(apellidos) FROM empleados

WHERE genero='F';

El resultado de esta consulta es el siguiente:

|  |  |
| --- | --- |
| **UPPER(nombres)** | **UPPER(apellidos)** |
| VIVIANA | RODRÍGUEZ |
| MARTHA | CASTRO |

* **QUOTE(texto):** Esta función coloca entre comillas el texto que se le envía dentro del paréntesis. Como ejemplo vamos a mostrar entre comillas los nombres y apellidos de las mujeres en la tabla “Empleados”. La consulta a realizar es la siguiente:

SELECT QUOTE(nombres), QUOTE(apellidos) FROM empleados

WHERE genero='F';

El resultado de esta consulta es el siguiente:

|  |  |
| --- | --- |
| **QUOTE(nombres)** | **QUOTE(apellidos)** |
| 'Viviana' | 'Rodríguez' |
| 'Martha' | 'Castro' |

WEBGRAFIA

http://www.w3schools.com/sql/default.asp

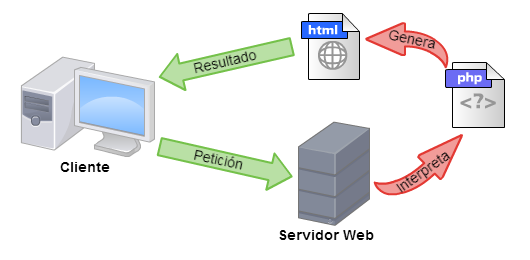
http://manuales.guebs.com/mysql-5.0/functions.html#control-flow-functions

**Unidad 4: Lenguaje PHP y conexión a Base de datos MySQL**

**Lección 1. Introducción a PHP.**

PHP (acrónimo recursivo de ***PHP: Hypertext Preprocessor***) es un lenguaje de programación de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado dentro de código HTML.

**Como Funciona PHP:** El usuario escribe una URL en su navegador y éste envía un petición al servidor web. El servidor lee la página PHP (la interpreta). Envía al navegador del usuario un documento compilado en formato HTML.



|  |  |
| --- | --- |
| Título del recurso educativo | Diagrama funcionamiento general de PHP |
| Descripción: | 1 imagen del funcionamiento de PHP |
| Unidad | 4 |
| Lección | 1 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/esquema\_php.png |

**Ventajas de utilizar PHP:**

1. Lenguaje de programación totalmente libre y de código abierto, es decir, es un lenguaje que se puede usar de forma gratuita.
2. Su aprendizaje es muy rápido, comparado con otros lenguajes de programación.
3. Con PHP se puede tener un fácil acceso a diferentes sistemas de gestión de bases de datos.
4. La programación en este lenguaje se puede hacer desde cualquier editor de texto.
5. Existe una comunidad muy grande alrededor del mundo desarrollando en PHP y mejorándolo día a día.

**Sintaxis básica de PHP:** Para empezar a programar en este lenguaje, se deben escribir las etiquetas **<?php** y **?>**, y que indican a PHP dónde empezar y finalizar la interpretación del código. Este mecanismo permite que PHP pueda ser incrustado en diferentes tipos de documentos.

Así se vería una porción de código PHP:

<?php

echo "¡Hola, esto es código PHP!";

?>

La instrucción **echo** permite escribir en pantalla un texto o el contenido de una variable. Si se trata de un texto, se debe encerrar entre comillas lo que queremos que se muestre, pero si es el contenido de una variable no es necesario hacerlo, simplemente escribimos **echo** seguido del nombre de la variable.

**Separación de instrucciones:** PHP requiere que las instrucciones terminen en punto y coma al final de cada sentencia. La etiqueta de cierre de un bloque de código de PHP automáticamente implica un punto y coma. No es necesario usar un punto y coma para cerrar la última línea de un bloque de PHP.

<?php

echo 'Esto es una prueba';

echo 'Con 2 Instrucciones';

?>

<?php echo 'Esto es otra prueba' ?>

**Variables en PHP:** En PHP no hace falta declarar una variable antes de su uso, ni establecer su tipo.

Su nombre puede ser una combinación de letras (a-Z), números y guiones bajos, precedidos con el signo $.

El primer carácter no puede ser un número, y php diferencia entre mayúsculas y minúsculas. Por ejemplo $var no es igual a $Var.

<?php

$un\_bool = TRUE;   // un valor booleano

$un\_str  = "foo";  // una cadena

$un\_str2 = 'foo';  // una cadena

$un\_int  = 12;     // un entero

$un\_flt = 1.234;   // un número de punto flotante

$fecha = date(Y);  // Podemos asignar el resultado de una función.

?>

El tipo de cada variable se determina teniendo en cuenta la información que esta contiene.

Si se va a asignar un texto, este se debe encerrar entre comillas dobles. Si se va a asignar un valor numérico no es necesario hacer esto.

También es posible asignarle a una variable el valor de otra variable.

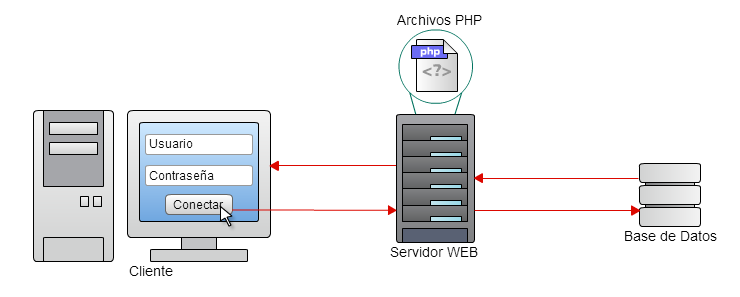
<?php

$variable1 = $variable2;

?>

**Lección 2. Conexión a la base de datos.**

En esta lección aprenderemos como conectarnos desde PHP a una base de datos MySQL que previamente hayamos creado. En el siguiente diagrama podemos ver como se realiza la conexión a la base de datos.



|  |  |
| --- | --- |
| Título del recurso educativo | Diagrama Conexión a la Base de Datos conPHP |
| Descripción: | 1 imagen conexión a base de datos con PHP |
| Unidad | 4 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img/conexión\_bd\_php.png |

En este ejemplo cuando el cliente hace click en el botón **Conectar**, este hace una petición al servidor web para que le envíe una página web, el servidor intérpreta el codigo PHP. El servidor se conecta a la base de datos para obtener la información. El resultado es enviado de vuelta al servidor web, quien a su vez se lo envía al cliente.

Existen diferentes formas de realizar una conexión a la base de datos, dependiendo la versión de PHP.

La versión de un software o tecnología, es un número que se le asigna para indicar su nivel de desarrollo. Generalmente en cada nueva versión se han realizado mejoras a la tecnología.

A partir de la versión 5.5.0 de PHP, se utiliza una extensión llamada **mysqli** (mysql improved), o como a veces se le conoce, la extensión de MySQL mejorada.

Las extensiones son funcionalidades adicionales sobre las funcionalidades propias de PHP. En este caso la extensión **mysqli** nos provee mecanismos o funciones para conectarnos de manera sencilla a través de unas cuantas instrucciones a la Base de datos.

La instrucción que nos permitirá crear un enlace, es decir, crear la conexión inicial con la base de datos utilizando la extensión **mysqli** es la siguiente:

$conexion = new mysqli("<Servidor>", "<Usuario>", "<Contraseña>", "<Basedatos>");

Donde**,**

**<Servidor>** corresponde a la URL o dirección del servidor web.

**<Usuario>** es el nombre de usuario configurado previamente para acceder a la base de datos.

**<Contraseña>** Es la contraseña del usuario ingresado.

**<Basedatos>** Nombre de la base de datos a la cual nos queremos conectar.

Ahora a través de la variable conexión se podrá acceder a toda la funcionalidad proporcionada por la extensión mysqli.

En algunas ocasiones se puede presentar un error de conexión a la base de datos. Un mensaje de error de conexión se puede obtener accediendo a la variable **connect\_error** de la siguiente forma:

if ($conexion->connect\_error) {

die("Error al conectar a MySQL:".$conexion->connect\_error);

}

Si existe un error, **connect\_error** muestra un texto que describe el error.

**Lección 3. Consultas a la base de datos.**

Ahora que hemos establecido la conexión con la base de datos desde PHP, el siguiente paso es realizar las consultas a ellas para obtener la información que posteriormente se le mostrará al usuario desde la aplicación, o ejecutar alguna de las instrucciones de SQL aprendidas en el módulo anterior.

La instrucción que nos permitirá realizar una consulta a la base de datos o ejecutar una instrucción SQL utilizando la extensión mysqli, es la siguiente:

$result = $conexion->query("<consulta>") or die("<mensaje\_error>");

En la variable **$result**, quedará el resultado generado por la consulta especifica qué se ejecutó.

La función query() nos permite realizar la consulta y accedemos a esta funcionalidad a través de la variable **$conexion** seguida de los caracteres menos y mayor (->), con la cual habíamos establecido la conexión a la base de datos en la lección anterior. Entre los paréntesis escribimos la instrucción en lenguaje SQL que se quiere ejecutar.

Realmente **<consulta>** puede ser una instrucción de inserción a la base de datos, una instrucción de actualización de información, una instrucción de eliminación de registros de una tabla, una consulta de información almacenada en la base de datos, etc.

Algunos ejemplos de la utilización de esta instrucción son los siguientes:

1. $result = $conexion->query("INSERT INTO personas (cedula, nombres, apellidos) VALUES ('3456765', 'Andrés Fernando', 'Vélez Torres')") or die("No se pudo realizar la inserción");

Una manera más organizada de escribir esta instrucción en PHP, sería utilizando dos variables adicionales. En una de ellas se almacenará la instrucción SQL que se quiere ejecutar y en la otra el mensaje de error que se quiere mostrar, si existe algún fallo en su ejecución. Finalmente el código quedaría como se muestra a continuación.

$sql = "INSERT INTO personas (cedula, nombres, apellidos)

VALUES ('3456765', 'Andrés Fernando', 'Vélez Torres')";

$mensajeError = "No se pudo realizar la inserción";

$result = $conexion->query( $sql ) or die( $mensajeError );

En este ejemplo vimos cómo se realiza una inserción de datos.

1. $sql = "UPDATE personas SET nombres='Carlos Mario' WHERE cedula='3456765' ";

$mensajeError = "No se pudo realizar la actualización";

$result = $conexion->query( $sql ) or die( $mensajeError );

En este ejemplo estamos realizando una actualización de datos.

1. $sql = "SELECT cedula, nombres, apellidos FROM personas ";

$mensajeError = "No se pudo realizar la consulta";

$result = $conexion->query( $sql ) or die( $mensajeError );

En este ejemplo estamos realizando una consulta de información.

El resultado de la ejecución de la instrucción **$conexion->query**, se asignará a la variable **$result**. Si la instrucción SQL es de tipo SELECT y no se presenta fallo en la ejecución; allí quedara definido un objeto desde el cual podremos extraer el resultado de esta consulta. Para otras consultas, esta instrucción asignara el valor TRUE (Verdadero) si tiene éxito. Si se presenta un fallo con cualquier tipo de instrucción SQL, se asignara un valor FALSE (falso).

**Lección 4. Mostrando los resultados.**

Ya hemos ejecutado la instrucción SQL con PHP, ahora debemos mostrar estos resultados en pantalla al usuario. Lo ideal es mostrar estos resultados en una página Web con elementos HTML como por ejemplo una tabla. Recordemos que PHP es un lenguaje que se puede introducir o mezclar con código HTML.

Para obtener el resultado de una consulta que arroja un conjunto de datos tal como SELECT, se utiliza la instrucción **fetch\_array()**; esta devuelve los valores de un registro o fila. Devuelve NULL (Vacío) si no hay filas que mostrar.

Se puede acceder a esta funcionalidad a través de la variable **$result**,con la cual ejecutamos la instrucción SQL en la lección anterior**,** seguida de los caracteres menos y mayor (->), de la siguiente manera:

$fila=$result->fetch\_array();

Dado que al ejecutar la instrucción **$result->fetch\_array()** se obtendrá el resultado de una sola fila, tendremos que hacer un recorrido de todo el conjunto de resultados utilizando un ciclo de la siguiente manera:

while($fila=$result->fetch\_array())

{

echo $fila[0]." -- ".$fila[1].'<br>';

}

En la variable **$fila** se asigna una fila completa a la vez, por eso cada vez que se ejecuta el ciclo debemos obtener el valor de cada campo de la fila.

En **$fila[0]** se encuentra el primer valor de la fila, en **$fila[1]** el segundo, y así sucesivamente hasta **$fila[n]**. Donde n es el número de campos de la fila menos 1. Es decir si la fila tiene 3 campos n será igual a 2. Esto dado que se empieza a obtener valores empezando en 0.

Para entender mejor lo visto hasta el momento supongamos que tenemos la tabla personas con la siguiente información.

**Personas**

|  |  |  |  |
| --- | --- | --- | --- |
| **cedula** | **nombres** | **apellidos** | **genero** |
| 98765434 | Sergio | Gálvez | M |
| 76543275 | Raúl | Franco | M |
| 43532658 | Viviana | Rodríguez | F |
| 63297867 | Fernando | Muñoz | M |

Vamos a asumir que ya estamos conectados a la base de datos a través de la variable **$conexion**.

Ahora vamos a crear las instrucciones en PHP para realizar la consulta de información de todas las personas.

$sql = "SELECT cedula, nombres, apellidos, genero FROM personas ";

$mensajeError = "No se pudo realizar la consulta";

$result = $conexion->query( $sql ) or die( $mensajeError );

Luego vamos a escribir en pantalla el resultado.

while($fila=$result->fetch\_array())

{

echo $fila[0]." -- ".$fila[1]." -- ".$fila[2]." -- ".$fila[3].'<br>';

}

La primera vez que se realice el ciclo, en la variable **$fila** se almacenará la siguiente información:

|  |  |  |  |
| --- | --- | --- | --- |
| 98765434 | Sergio | Gálvez | M |

Con la instrucción **echo $fila[0]." -- ".$fila[1]." -- ".$fila[2]." -- ".$fila[3].'<br>'.**

* En **fila[0]** se encuentra la cédula **98765434**
* En **fila[1]** se encuentra el nombre **Sergio**
* En **fila[2]** se encuentra el apellido **Gálvez**
* En **fila[3]** se encuentra el género **M**

Por lo tanto en pantalla se escribirá: **98765434 -- Sergio -- Gálvez -- M**

En este caso el punto (**.**) en PHP se usa para concatenar o unir textos y **<br>** es una instrucción en HTML que indica un salto de línea.

El ciclo se sigue realizando mientras haya filas para mostrar.

La segunda vez que se realice el ciclo, en la variable **$fila** se almacenará la siguiente información:

|  |  |  |  |
| --- | --- | --- | --- |
| 76543275 | Raúl | Franco | M |

Con la instrucción **echo $fila[0]." -- ".$fila[1]." -- ".$fila[2]." -- ".$fila[3].'<br>'** se escribirá en pantalla: **76543275-- Raúl -- Franco -- M**

Y asi sucesivamente con las demás filas faltantes. Al final el resultado en pantalla sería algo como esto:

98765434 -- Sergio -- Gálvez – M

76543275-- Raúl -- Franco -- M

43532658 -- Viviana-- Rodríguez -- F

63297867 – Fernando – Muñoz -- M

**WEBGRAFIA**

<https://otroblogsobretics.wordpress.com/2011/05/04/conceptos-php/>

<http://www.w3schools.com/php/default.asp>

<http://administraciondesistemas.pbworks.com/f/Manual_PHP5_Basico.pdf>