Escuela Profesional de
Ciencia de la Computación

BDI Fase 3

Base de Datos I

Dr. Edward Hinojosa C.
Dr. Edgar Sarmiento C.

Universidad Nacional de San Agustín de Arequipa

2020/Semestre Par

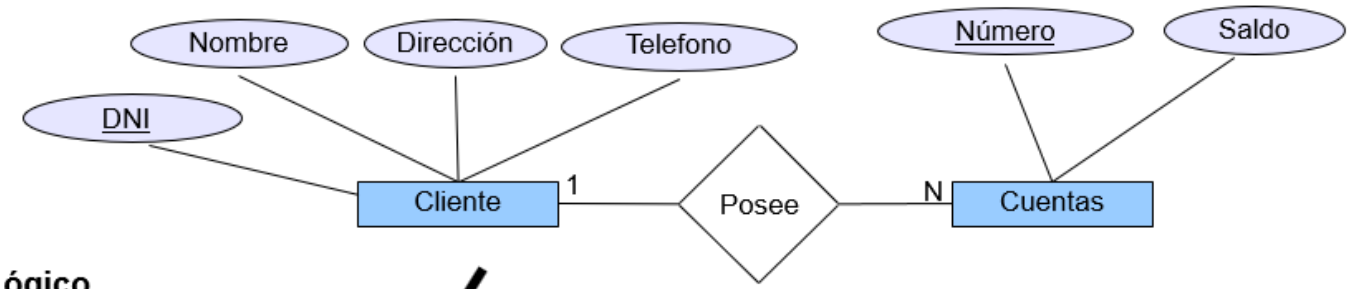
Índice

ehinojosa@unsa.edu.pe

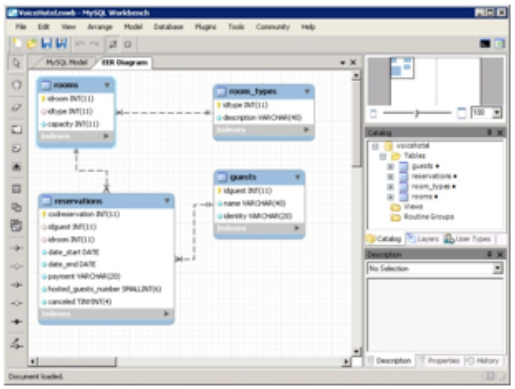
Fases del Proyecto de BD

Levantamiento y Análisis de Requisitos

Modelo Conceitual (Diagrama Entidad-Relación)



Modelo Lógico (Modelo Relacional)



Modelo Físico (Instrucciones SQL)

```
#tabla cliente
CREATE TABLE  Cliente(
  DNI            CHAR(14),
  Nombre         VARCHAR(45),
  Direccion      VARCHAR(45),
  Telefono       CHAR(13),
  PRIMARY KEY  (DNI));
```

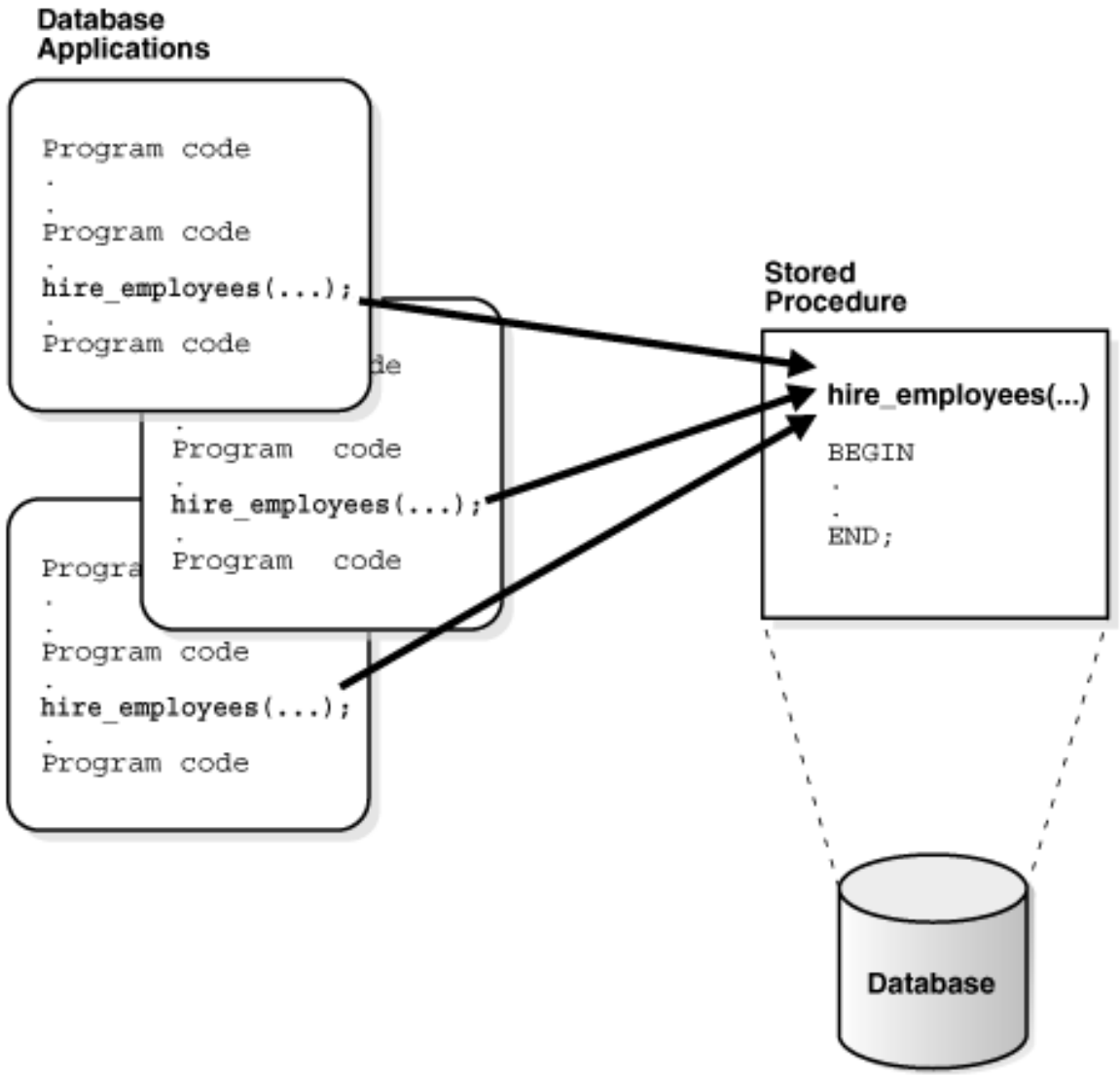
ehinojosa@unsa.edu.pe

Procedimientos Almacenados (Stored Procedures)

- SP es un conjunto de comandos al cual es atribuido un nombre.
- Este conjunto se encuentra almacenado en la BD y puede ser invocado en cualquier momento por el SGBD o por un sistema que utiliza la misma.
- SP son utilizados para mover gran parte de código de manipulación de datos para el servidor, ello elimina la transferencia de datos del servidor al cliente, por la red, para manipulación $\rightarrow$ reducción de tráfico de la red $\rightarrow$ mejor performance general.

ehinojosa@unsa.edu.pe

Procedimientos Almacenados (Stored Procedures)



ehinojosa@unsa.edu.pe

Procedimientos Almacenados (Stored Procedures)

- Siempre que una aplicación cliente envía un comando SQL para el servidor, el comando tiene que tener la sintaxis analizada y, a continuación, es enviado a un programa optimizador del SGBD para la formulación de un plan de ejecución.
- Los SP son analizados y optimizados una única vez (cuando son creados), presentan mejor rendimiento que los comandos SQL enviados por las aplicaciones.

ehinojosa@unsa.edu.pe

Procedimientos Almacenados (Stored Procedures)

- SP pueden reducir el tiempo de desarrollo y manutención de los sistemas pues las SP pueden ser compartidos por todas las aplicaciones existentes. El mantenimiento es sencillo porque es posible alterar un SP sin tener que alterar y/o recompilar cada aplicación cliente.
- Puede ser más veloz dado que generalmente el servidor es una de las máquinas más poderosas en la red.

ehinojosa@unsa.edu.pe

Procedimientos Almacenados - Sintaxis

```
CREATE
    [DEFINER = user]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = user]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic: {
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
}

routine_body:
    Valid SQL routine statement
```

ehinojosa@unsa.edu.pe

Procedimientos Almacenados (Stored Procedures)

- Un SP es invocado con el comando CALL.
- No se puede invocar un SP en una expresión.
- Podemos utilizar las sentencias de control IF, ELSE, LOOP, WHILE, . . . (Tarea)

ehinojosa@unsa.edu.pe

Procedimientos Almacenados (Stored Procedures)

- Se consideran 3 tipos de parámetros (Tarea):
  - Un parámetro IN pasa un valor a un procedimiento. El procedimiento puede modificar el valor, pero la modificación no es visible para la persona que llama cuando el procedimiento vuelve.
  - Un parámetro OUT pasa un valor del procedimiento a la persona que llama. Su valor inicial es NULL dentro del procedimiento, y su valor es visible para la persona que llama cuando el procedimiento vuelve.
  - Un parámetro INOUT es inicializado por la persona que llama, puede ser modificado por el procedimiento, y cualquier cambio realizado por el procedimiento es visible para la persona que llama cuando el procedimiento regresa.

ehinojosa@unsa.edu.pe

Procedimientos Almacenados - Ejemplo

- Cree un SP que inserte un nuevo departamento.

depa_id	nombre	edificio	presupuesto
1	Computacion	1	4785.30
4	Ingenieria	2	1748.85
5	Biomedicas	3	21447.59
NULL	NULL	NULL	NULL

```
CALL Insertar_Departamento(7, 'Sociales', 4, 7895.40);
```

depa_id	nombre	edificio	presupuesto
1	Computacion	1	4785.30
4	Ingenieria	2	1748.85
5	Biomedicas	3	21447.59
7	Sociales	4	7895.40
NULL	NULL	NULL	NULL

```
USE universidad;

DELIMITER //
CREATE PROCEDURE Insertar_Departamento(IN id INT, IN nomb VARCHAR(30), IN edif INT, IN presu DECIMAL(12,2))
BEGIN
    INSERT INTO departamento VALUES (id, nomb, edif, presu);
END;
//
DELIMITER ;
```

ehinojosa@unsa.edu.pe

Procedimientos Almacenados - Ejemplo

- Cree un SP que devuelva el código, nombre completo y teléfonos de o de los alumnos alumnos que comiencen con la palabra de búsqueda ingresada en su primer apellido.

estu_id	nombres	prim_apel	segu_apel	depa_id	prof_id
123456789	Juan	Perez	Rodriguez	1	1
333445555	Frank	Velazquez	Flores	1	1
453453453	Daniela	Acco	Olvarez	1	2
666884444	Pedro	Lima	Maldonado	1	2
888665555	Francisco	Linares	Gomez	1	1
987654321	Luisa	Santos	Ferrel	1	2
987987987	Mateo	Vela	Marruecos	1	2
999887777	Alice	Jimenez	Portugal	1	1

estu_id	telefono
123456789	888888888
123456789	999999999
987654321	333333333
987654321	555555555
987654321	777777777

Código	Nombre Completo	Teléfono
123456789	Perez Rodriguez, Juan	888888888
123456789	Perez Rodriguez, Juan	999999999

```
CALL Buscar_Alumno('pe');
```

```
USE universidad;

DELIMITER //
DROP PROCEDURE IF EXISTS Buscar_Alumno//
CREATE PROCEDURE Buscar_Alumno(IN apel VARCHAR(30))
BEGIN
    SELECT e.estu_id 'Código',
    CONCAT(e.prim_apel, ' ', e.segu_apel, ', ', e.nombres) 'Nombre Completo',
    t.telefono 'Teléfono'
    FROM estudiante e
    INNER JOIN estudiante_telefono t
    ON e.estu_id = t.estu_id
    WHERE UPPER(prim_apel) LIKE CONCAT(UPPER(apel) ,'%');
END;
//
DELIMITER ;
```

ehinojosa@unsa.edu.pe

Procedimientos Almacenados - Ejemplo

- Cree un SP que reduzca el presupuesto en cierta cantidad de un departamento siempre y cuando el presupuesto sea mayor a 10 mil soles, si es menor o igual reducir la mitad de la cantidad.

depa_id	nombre	edificio	presupuesto
1	Computacion	1	4785.30
4	Ingenieria	2	1748.85
5	Biomedicas	3	21447.59
7	Sociales	4	7895.40

```
CALL Reducir_Presupuesto(5, 500.00);
CALL Reducir_Presupuesto(7, 500.00);
```

depa_id	nombre	edificio	presupuesto
1	Computacion	1	4785.30
4	Ingenieria	2	1748.85
5	Biomedicas	3	20947.59
7	Sociales	4	7645.40

```
USE universidad;

DELIMITER //
DROP PROCEDURE IF EXISTS Reducir_Presupuesto//
CREATE PROCEDURE Reducir_Presupuesto(IN id INT, IN presu DECIMAL(12,2))
BEGIN
    DECLARE presu_ante DECIMAL(12,2);
    DECLARE limite DECIMAL(12,2);

    SET limite = 10000.00;
    SET presu_ante = (SELECT d.presupuesto from departamento d WHERE d.depa_id = id);

    IF presu_ante > limite THEN
        UPDATE departamento d SET d.presupuesto = d.presupuesto - presu WHERE d.depa_id = id;
    ELSE
        SET presu = presu / 2;
        UPDATE departamento d SET d.presupuesto = d.presupuesto - presu WHERE d.depa_id = id;
    END IF;
END;
//
DELIMITER ;
```

ehinojosa@unsa.edu.pe

Funciones

- Se diferencian de los SP:
  - Solamente pueden tener parámetros de entrada IN y no parámetros de salida OUT o INOUT.
  - Deben retornar en un valor con algún tipo de dato definido.
  - Pueden usarse en el contexto de una expresión SQL.
  - Solo retornan un valor individual, no un conjunto de registros.

ehinojosa@unsa.edu.pe

Funciones

- Cree un función que calcule la edad de un profesor.

prof_id	nombres	prim_apel	segu_apel	fech_naci	depa_id
1	Edward	Hinojosa	Cardenas	NULL	1
2	Edgar	Sarmiento	Calisaya	1980-12-26	1
3	Juan Carlos	Gutierrez	M	1983-10-17	1
NULL	NULL	NULL	NULL	NULL	NULL

```
SELECT Calcular_Edad(3);
```

Calcular_Edad(3)
37

```
USE universidad;

DELIMITER //
DROP FUNCTION IF EXISTS Calcular_Edad//
CREATE FUNCTION Calcular_Edad(id INT) RETURNS INT DETERMINISTIC
BEGIN
    DECLARE edad INT;
    DECLARE fn date;

    SET fn = (SELECT p.fech_naci from profesor p WHERE p.prof_id = id);
    SET edad =
        (SELECT date_format(NOW(), '%Y' ) -
            date_format(fn, '%Y' ) -
            (date_format(NOW(), '00-%m-%d') < date_format(fn, '00-%m-%d')));

    RETURN edad;
END;
//
DELIMITER ;
```

ehinojosa@unsa.edu.pe

Funciones

- Cree un SP que muestre la edad de los docentes que se hayan registrado su fecha de nacimiento (use la función anterior).

```
CALL Profesores_Edad();
```

Código	Nombre Completo	Edad
2	Sarmiento Calisaya, Edgar	39
3	Gutierrez M, Juan Carlos	37

```
USE universidad;

DELIMITER //
DROP PROCEDURE IF EXISTS Profesores_Edad//
CREATE PROCEDURE Profesores_Edad()
BEGIN
    SELECT p.prof_id 'Código',
    CONCAT(p.prim_apel, ' ', p.segu_apel, ', ', p.nombres) 'Nombre Completo',
    Calcular_Edad(p.prof_id) Edad
    FROM profesor p
    WHERE p.fech_naci IS NOT NULL;
END;
//
DELIMITER ;
```

ehinojosa@unsa.edu.pe

Triggers

- Un Trigger es un SP creado para ejecutarse automaticamente cuando ocurra un evento en nuestra base de datos.
- Dichos eventos son generados por los comandos INSERT, UPDATE y DELETE.
- Son utilizados para ejecutar un bloque de instrucciones que proteja, restrinja o preparen la información de nuestras tablas, al momento de manipular nuestra información.

ehinojosa@unsa.edu.pe

Trigger

```
CREATE
    [DEFINER = user]
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

ehinojosa@unsa.edu.pe

Triggers - Identificadores NEW y OLD

- Si debemos relacionar el trigger con columnas especificas de una tabla debemos usar los identificadores OLD y NEW.
- OLD indica el valor antiguo de la columna y NEW el valor nuevo que pudiese tomar. Por ejemplo: OLD.idproducto ó NEW.idproducto.
- Si usamos la sentencia UPDATE podremos referirnos a un valor OLD y NEW, ya que modificaremos registros existentes por nuevos valores.
- Si usamos INSERT solo usaremos NEW, ya que su naturaleza es únicamente de insertar nuevos valores a las columnas.
- Y, si usamos DELETE usaremos OLD debido a que borraremos valores que existen con anterioridad.

ehinojosa@unsa.edu.pe

Triggers - BEFORE y AFTER

- Estas clausulas indican si el Trigger se ejecuta antes o después del evento.
- Existen ciertos eventos que no son compatibles con estas sentencias. Por ejemplo, si tuvieras un Trigger AFTER que se ejecuta en una sentencia UPDATE, sería ilógico editar valores nuevos NEW, sabiendo que el evento ocurrió.

ehinojosa@unsa.edu.pe

Triggers - BEFORE en la sentencia UPDATE

- Cree un Trigger que valide la fecha de nacimiento de un docente antes de ser actualizada, si la fecha es mayor a la actual, entonces asignar el valor de null.

prof_id	nombres	prim_apel	segu_apel	fech_naci	depa_id
1	Edward	Hinojosa	Cardenas	NULL	1
2	Edgar	Sarmiento	Calisaya	1980-12-26	1
3	Juan Carlos	Gutierrez	M	1983-10-17	1
NULL	NULL	NULL	NULL	NULL	NULL

```
USE universidad;

DELIMITER //
DROP TRIGGER IF EXISTS validar_profesor_edad//
CREATE TRIGGER validar_profesor_edad
BEFORE UPDATE ON profesor FOR EACH ROW
BEGIN
    IF NEW.fech_naci > NOW() THEN
        SET NEW.fech_naci = NULL;
    END IF;
END;
//
DELIMITER ;
```

```
UPDATE profesor p SET p.fech_naci = '2021-10-14' WHERE prof_id = 1;
```

```
UPDATE profesor p SET p.fech_naci = '2019-10-14' WHERE prof_id = 1;
```

prof_id	nombres	prim_apel	segu_apel	fech_naci	depa_id
1	Edward	Hinojosa	Cardenas	2019-10-14	1
2	Edgar	Sarmiento	Calisaya	1980-12-26	1
3	Juan Carlos	Gutierrez	M	1983-10-17	1

ehinojosa@unsa.edu.pe

Triggers - AFTER en la sentencia INSERT

- Cree un Trigger que aumente el presupuesto del departamento en 1000 soles cuando una profesor es insertado a su departamento.

depa_id	nombre	edificio	presupuesto
1	Computacion	1	4785.30
4	Ingenieria	2	1748.85
5	Biomedicas	3	20947.59
7	Sociales	4	7645.40

```
INSERT INTO profesor VALUE (4, 'Juan', 'Perez', 'Perez', '1984-7-14', 4);
```

prof_id	nombres	prim_apel	segu_apel	fech_naci	depa_id
1	Edward	Hinojosa	Cardenas	2019-10-14	1
2	Edgar	Sarmiento	Calisaya	1980-12-26	1
3	Juan Carlos	Gutierrez	M	1983-10-17	1
4	Juan	Perez	Perez	1984-07-14	4
NULL	NULL	NULL	NULL	NULL	NULL

depa_id	nombre	edificio	presupuesto
1	Computacion	1	4785.30
4	Ingenieria	2	2748.85
5	Biomedicas	3	20947.59
7	Sociales	4	7645.40
NULL	NULL	NULL	NULL

```
DELIMITER //
DROP TRIGGER IF EXISTS aumentar_presupuesto//
CREATE TRIGGER aumentar_presupuesto
AFTER INSERT ON profesor FOR EACH ROW
BEGIN
    UPDATE departamento d SET presupuesto = presupuesto + 1000.00 WHERE d.depa_id = NEW.depa_id;
END;
//
DELIMITER ;
```

ehinojosa@unsa.edu.pe

Indexación

- Definiremos que son los índices y sus características.
- Podremos saber teóricamente cómo optimizar nuestras consultas de tipo SELECT mediante la creación y el uso de índices en nuestra base de datos.
- Con el uso de índices, podemos reducir fácilmente, y de forma considerable, el tiempo de ejecución de nuestras consultas de tipo SELECT. Sobretodo, esta mejora en los tiempos de ejecución será mayor cuanto más grandes (mayor cantidad de datos) sean las tablas de la base de datos con la que estemos trabajando.

ehinojosa@unsa.edu.pe

¿Qué es un índice?

- Un índice es un puntero a una fila de una determinada tabla de nuestra base de datos. Recordemos que, un puntero no es más que una referencia que asocia el valor de una determinada columna (o el conjunto de valores de una serie de columnas) con las filas que contienen ese valor (o valores) en las columnas que componen el puntero.
- La construcción de los índices es el primer paso para realizar optimizaciones en las consultas realizadas contra nuestra base de datos. Por ello, es importante conocer bien su funcionamiento y los efectos colaterales que pueden producir.

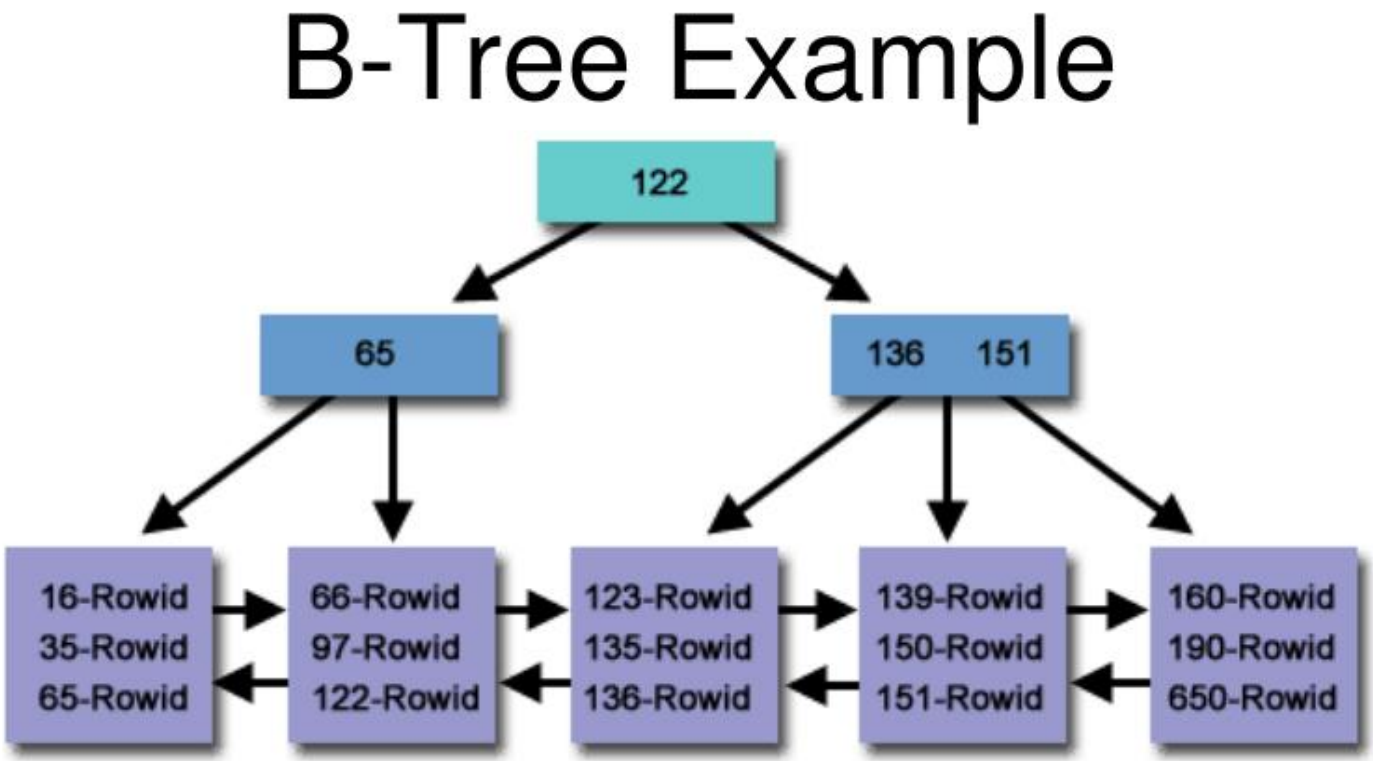
ehinojosa@unsa.edu.pe

¿Qué es un índice?

- Se emplean los índices para encontrar las filas que contienen los valores específicos de las columnas empleadas en la consulta de una forma más rápida. Si no existiesen índices, se podría realizar una búsqueda desde la primera fila de la tabla hasta la última buscando aquellas filas que cumplen los valores empleados en la consulta.
- Esto implica que, cuanto más filas tenga la tabla, más tiempo tardará en realizar la consulta.
- En cambio, si la tabla contiene índices en las columnas empleadas en la consulta, MySQL tendría una referencia directa hacia los datos sin necesidad de recorrer secuencialmente todos ellos.

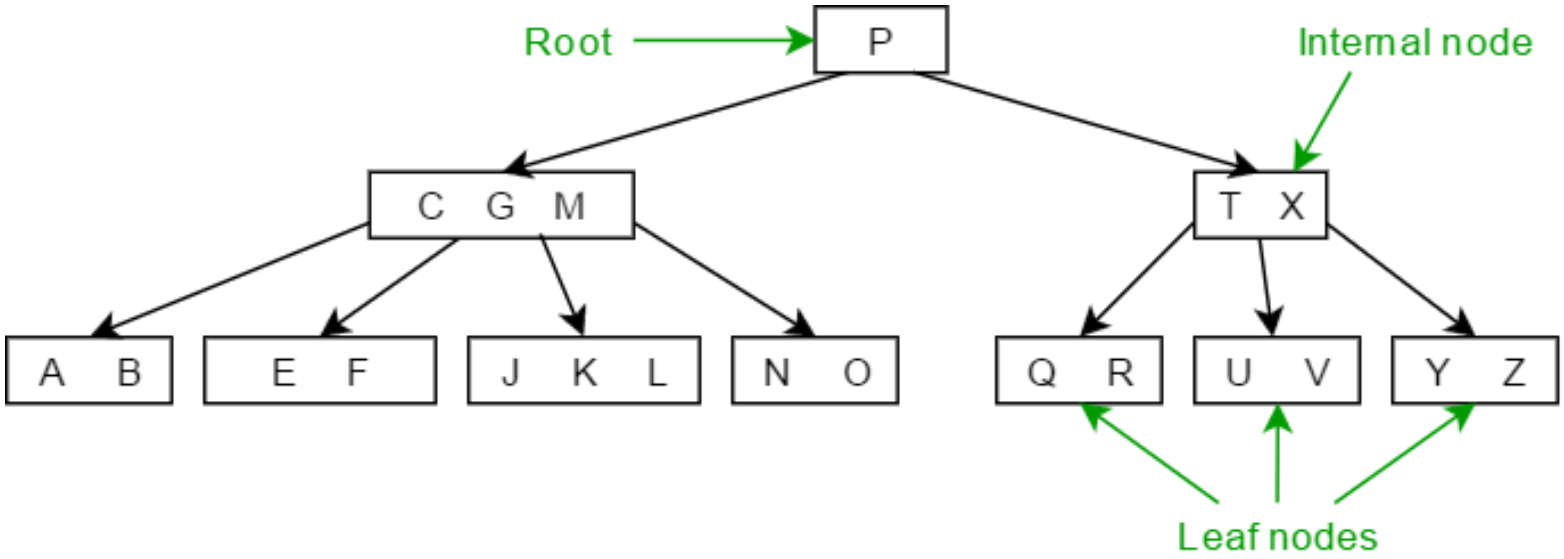
ehinojosa@unsa.edu.pe

¿Qué es un índice?



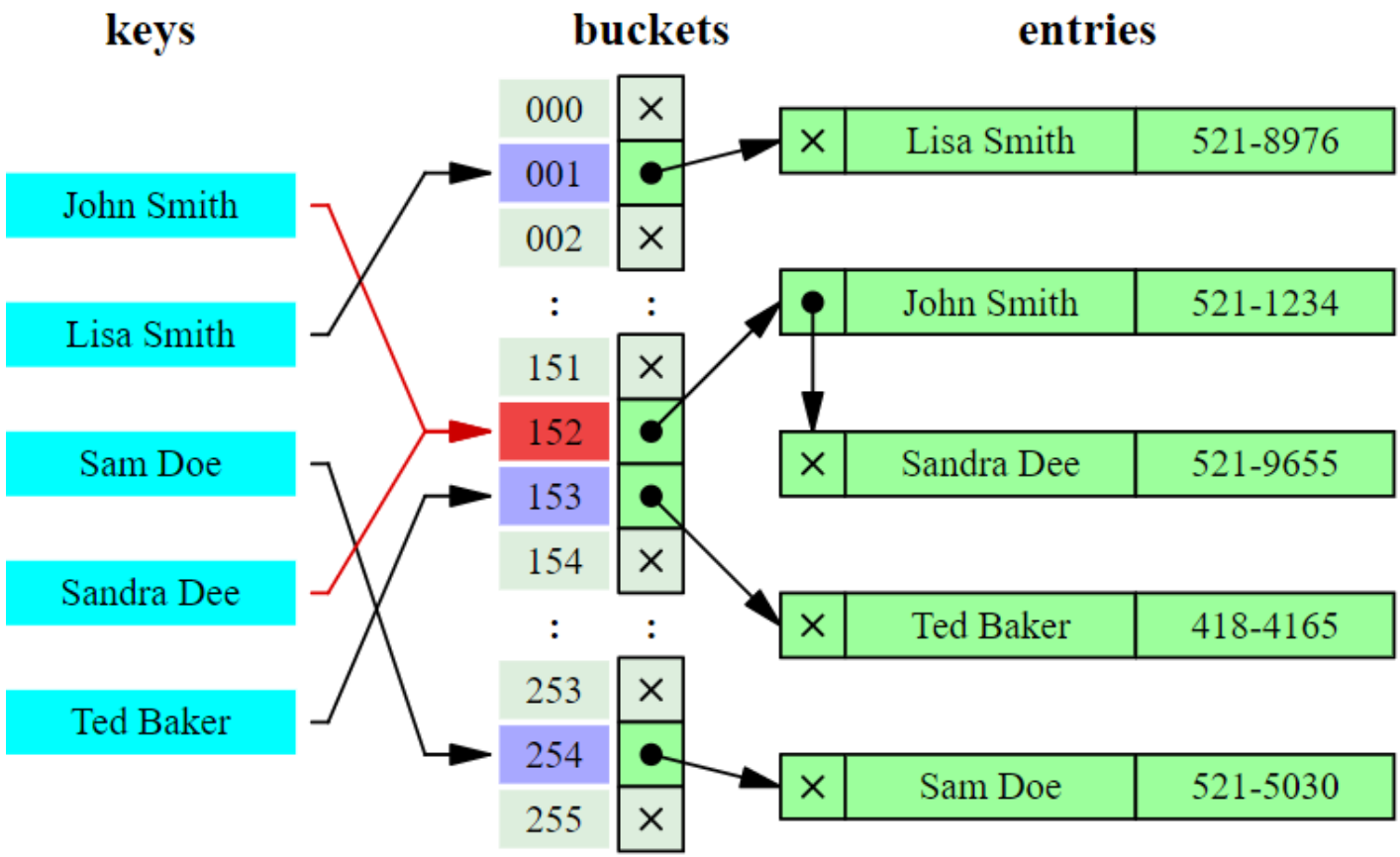
ehinojosa@unsa.edu.pe

¿Qué es un índice?



ehinojosa@unsa.edu.pe

¿Qué es un índice?



ehinojosa@unsa.edu.pe

Tipos de Índices

- INDEX (NON-UNIQUE): este tipo de índice se refiere a un índice normal, no único. Esto implica que admite valores duplicados para la columna (o columnas) que componen el índice. No aplica ninguna restricción especial a los datos de la columna (o columnas) que componen el índice sino que se emplea simplemente para mejorar el tiempo de ejecución de las consultas.
- UNIQUE: este tipo de índice se refiere a un índice en el que todas las columnas deben tener un valor único. Esto implica que no admite valores duplicados para la columna (o columnas) que componen el índice. Aplica la restricción de que los datos de la columna (o columnas) deben tener un valor único.

ehinojosa@unsa.edu.pe

Tipos de Índices

- PRIMARY: este tipo de índice se refiere a un índice en el que todas las columnas deben tener un valor único (al igual que en el caso del índice UNIQUE) pero con la limitación de que sólo puede existir un índice PRIMARY en cada una de las tablas. Aplica la restricción de que los datos de la columna (o columnas) deben tener un valor único.

ehinojosa@unsa.edu.pe

Tipos de Índices

- FULLTEXT: estos índices se emplean para realizar búsquedas sobre texto (CHAR, VARCHAR y TEXT). Estos índices se componen por todas las palabras que están contenidas en la columna (o columnas) que contienen el índice. No aplica ninguna restricción especial a los datos de la columna (o columnas) que componen el índice sino que se emplea simplemente para mejorar el tiempo de ejecución de las consultas.
- SPATIAL: estos índices se emplean para realizar búsquedas sobre datos que componen formas geométricas representadas en el espacio.

ehinojosa@unsa.edu.pe

Creación de Índices

```
CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name
    [index_type]
    ON tbl_name (key_part,...)
    [index_option]
    [algorithm_option | lock_option] ...

key_part: {col_name [(length)] | (expr)} [ASC | DESC]

index_option: {
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | {VISIBLE | INVISIBLE}
  | ENGINE_ATTRIBUTE [=] 'string'
  | SECONDARY_ENGINE_ATTRIBUTE [=] 'string'
}

index_type:
    USING {BTREE | HASH}

algorithm_option:
    ALGORITHM [=] {DEFAULT | INPLACE | COPY}

lock_option:
    LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
```

ehinojosa@unsa.edu.pe

¿Por qué no crear índices en todas las columnas?

- No todas las soluciones son perfectas y tampoco lo iba a ser la creación de índices. La creación de índices también tiene efectos negativos. Estos efectos negativos es bueno conocerlos ya que pueden ocasionar efectos colaterales no deseados.
- Uno de ellos es que las operaciones de inserción, actualización y eliminación que se realicen sobre tablas que tengan algún tipo de índice (o índices), verán aumentado su tiempo de ejecución. Esto es debido a que, después de cada una de estas operaciones, es necesario actualizar el índice (o los índices) presentes en la tabla sobre la que se ha realizado alguna de las operaciones anteriores.
- Otro efecto negativo es que los índices deben ser almacenados en algún lugar. Para ello, se empleará espacio de disco. Por ello, el uso de índices aumenta el tamaño de la base de datos en mayor o menor medida.

ehinojosa@unsa.edu.pe

Normalización

- El proceso de normalización de una base de datos relacional consiste en aplicar una serie de reglas para evitar a futuro realizar consultas innecesariamente complejas.
- En otras palabras están enfocadas en eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

ehinojosa@unsa.edu.pe

Normalización

- Las bases de datos se normalizan para:
  - Evitar la redundancia de datos.
  - Proteger la integridad de los datos.
  - Evitar problemas de actualización de los datos en las tablas.
- Para poder decir que nuestra base de datos está normalizada deben respetarse 3 niveles de normalización.

ehinojosa@unsa.edu.pe

Primera Forma Normal: 1FN

- La primera forma normal significa que los datos están en un formato de entidad, lo que significa que se han cumplido las siguientes condiciones:
  - Eliminar grupos repetidos en tablas individuales.
  - Crear una tabla independiente para cada conjunto de datos relacionados.
  - Identificar cada conjunto de relacionados con la clave principal.

ehinojosa@unsa.edu.pe

Primera Forma Normal: 1FN

Tabla no normalizada

Estudiante	Tutor	Habitacion	Clase 1	Clase 2	Clase 3
1606	Sarmiento	438	111-01	111-02	111-03
2602	Valle	222	201-01	201-02	201-03

Primera Forma Normal: Eliminar Grupos Repetidos

Estudiante	Tutor	Habitacion	Clases
1606	Sarmiento	438	111-01
1606	Sarmiento	438	111-02
1606	Sarmiento	438	111-03
2602	Valle	222	201-01
2602	Valle	222	201-02
2602	Valle	222	201-03

ehinojosa@unsa.edu.pe

Segunda Forma Normal: 2FN

- La segunda forma normal asegura que cada atributo describe la entidad, para ello se debe:
  - Crear tablas separadas para el conjunto de valores y los registros múltiples, estas tablas se deben relacionar con una clave externa.
  - Los registros no deben depender de otra cosa que la clave principal de la tabla, incluida la clave compuesta si es necesario

ehinojosa@unsa.edu.pe

Segunda Forma Normal: 2FN

Segunda Forma Normal: Eliminar Datos Redundantes

Estudiantes:

Estudiante	Tutor	Habitacion
1606	Sarmiento	438
2602	Valle	222

Registro:

Estudiante	Clases
1606	111-01
1606	111-02
1606	111-03
2602	201-01
2602	201-02
2602	201-03

ehinojosa@unsa.edu.pe

Tecera Forma Normal: 3FN

- La tercera forma normal comprueba las dependencias transitivas, eliminando campos que no dependen de la clave principal., para ello se debe:
  - Eliminar aquellos campos que no dependan de la clave.
  - Ninguna columna puede depender de una columna que no tenga una clave.
  - No puede haber datos derivados.

ehinojosa@unsa.edu.pe

Tecera Forma Normal: 3FN

Tercera Forma Normal: Eliminar Columnas No Depende De Clave

Estudiantes:

Estudiante	Tutor
1606	Sarmiento
2602	Valle

Facultad:

Nombre	Habitacion
Sarmiento	438
Vale	222

Registro:

Estudiante	Clases
1606	111-01
1606	111-02
1606	111-03
2602	201-01
2602	201-02
2602	201-03

ehinojosa@unsa.edu.pe

¡GRACIAS!

ehinojosa@unsa.edu.pe