Tecnologías de Información y Comunicaciones (TIC) para la Gestión

Profesores:
Robert Cercós B.
Victor Rebolledo L.
Sebastían A. Ríos.
Juan D. Velásquez. (coordinador)

Temario

- Introducción.
- Redes, Internet y Web.
- Cliente servidor de múltiples capas.
- La capa de datos.
- La capa de negocios.
- La capa de presentación.
- Administración de prouyectos informáticos.

Capítulo IV
La Capa de Datos

Temario

- Modelo Entidad-Relación
- Lenguaje SQL.
- Estructuras de acceso rápido.

Modelo Entidad Relación

Introducción

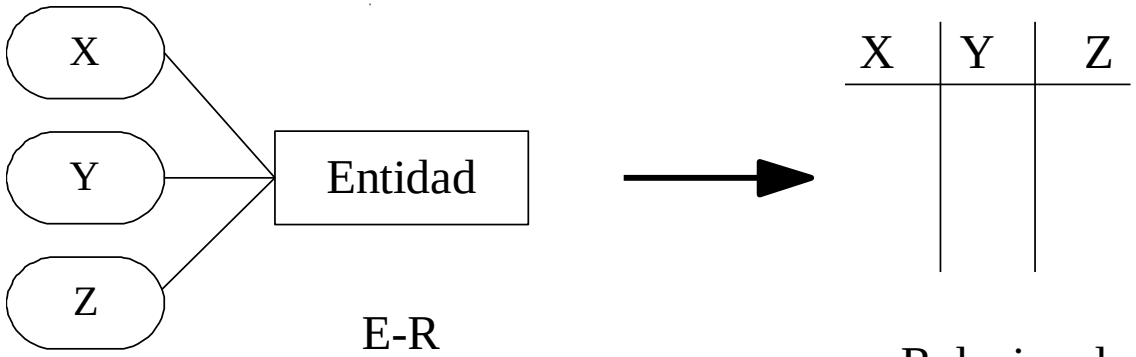
- Modelo Entidad Relación.
- Desarrollado por E. Codd 1970.
- Modelo de datos basado en representación de registros.
- Hoy en día se trata del modelo más usado en aplicaciones comerciales.
- Se compone de:
  - Estructura de datos.
  - Integridad de datos.
  - Manipulación de datos.

Modelo Relacional

- El modelo relacional es una forma de ver los datos es decir, es una receta para representar los datos, mediante tablas, y la receta para manipular esa representación mediante operadores.
- El modelo relacional se preocupa de tres aspectos de los datos : su estructura, su integridad y su manipulación.
- Desde una visión histórica este modelo es relativamente nuevo, los primeros sistemas de bases de datos estaban basados en el modelo de redes o jerárquicos, orientados a una implementación física de la base de datos.
- Con la introducción del modelo relacional se desarrolló una teoría orientada a las bases de datos relacionales. Esta teoría ayuda al diseño de las bases de datos y al proceso de consultas del usuario.

Modelo Relacional (2)

- En este modelo la base de datos es vista por el usuario como una relación de tablas. Cada fila de una tabla es un registro o tupla y los atributos son columnas o campos.



- Cada una de las tablas de la base de datos debe tener un nombre único. Generalmente corresponde al nombre de la entidad.
- Cada columna tiene asociado un dominio que es el conjunto de valores posibles para esa columna.

Modelo Relacional (3)

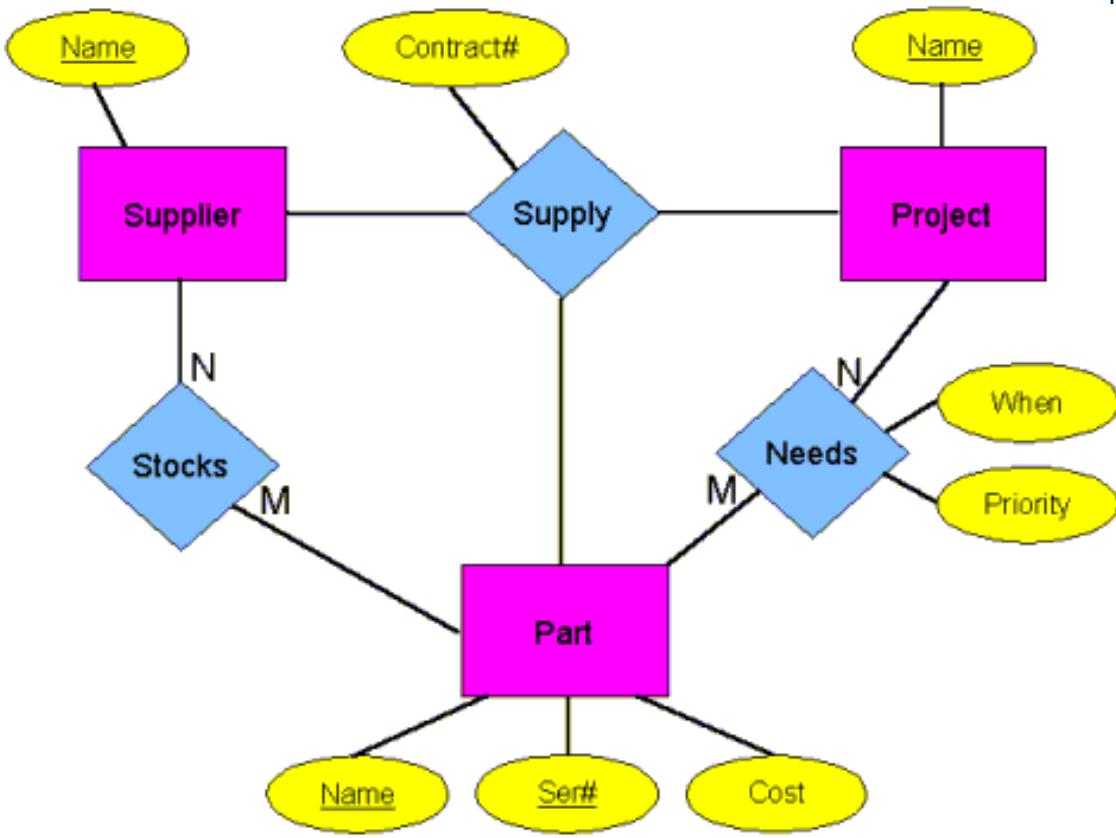
- El orden en que se listan las filas no tiene importancia.
- Si las columnas están rotuladas (tiene un nombre), entonces el orden de las columnas no tiene importancia.
- Representación de Conjuntos de Entidades

Material

Código	Nombre	Valor	Cantidad
001	Cuaderno	500	1000
002	Papel Continuo	20	3000
003	Clip	15	1500

Modelo relacional

- ¿Cómo se diseña la base de datos?
- Usando el modelo entidad relación, se crea un repositorio de datos que permite:
  - Contestar cualquier presunta sobre los datos.
  - Minimizar la redundancia



El modelamiento de datos

- Características
  - Es un desarrollo Top-Down
  - La idea general es hacer una abstracción del negocio y llevarlo una representación esquemática
- Una vez creado el modelo de datos, se usará una herramienta de software para implementarlo.
- Si el tipo de modelo es el Entidad Relación, se aconseja que la herramienta este orientada a ese tipo de estrategia.

Modelo llevado a la base de datos

- Los SABDR (Sistemas Administradores de Bases de Datos Relacionales) proveen un lenguaje estandarizado para llevar el modelo relacional a una representación computacional
- Algunos SADBR poseen objetos que permiten definir reglas complejas del negocio, que han sido impuestas al modelo de datos.
- Una vez tomada la decisión respecto de cuál será el SABDR a usar, hay que considerar :
  - Desempeño
  - Reglas de integridad de datos
  - Integración con otros sistemas en desarrollo o en producción.
  - Documentación

¿Porqué usamos el modelo entidad relación?

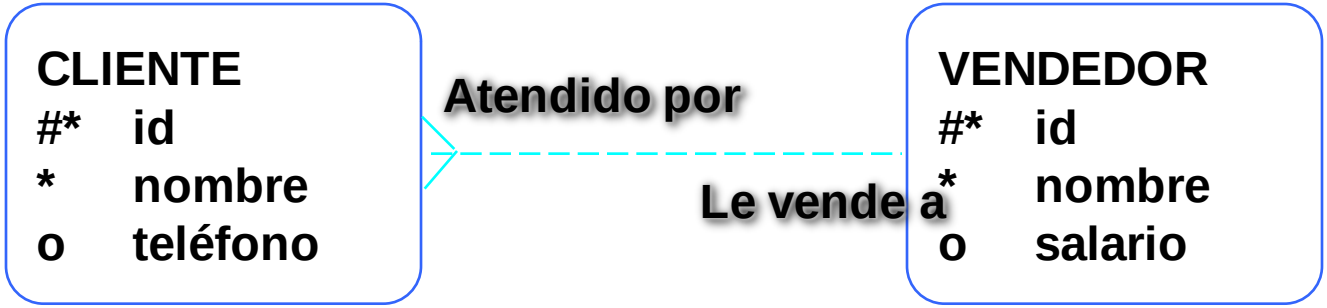
- Es relativamente fácil de entender para la contraparte técnica o para un cliente, en comparación con las antiguas formas de modelamiento de datos.
- Elimina la redundancia de los datos.
- Cualquier consulta que por sobre los datos se realice, es posible de ser contestada.
- Ya está estandarizado y los desarrolladores lo entienden fácilmente.
- Permite dimensionar los requerimientos de hardware para la base de datos.

Conceptos

- Entidad
  - Son los objetos que puedo caracterizar dentro del problema a modelar, por ejemplo clientes, vendedores, etc
- Atributo
  - Son las características que definan la entidad
  - Por ejemplo, en un cliente : Carné, edad, etc.
- Relación
  - Es la asociación directa que ocurre entre dos entidades.
  - Por ejemplo, en una esculea hay alumnos y profesores, que serían las entidades, la posible relación es " ensená a "

Modelo Entidad Relación

- Ejemplo



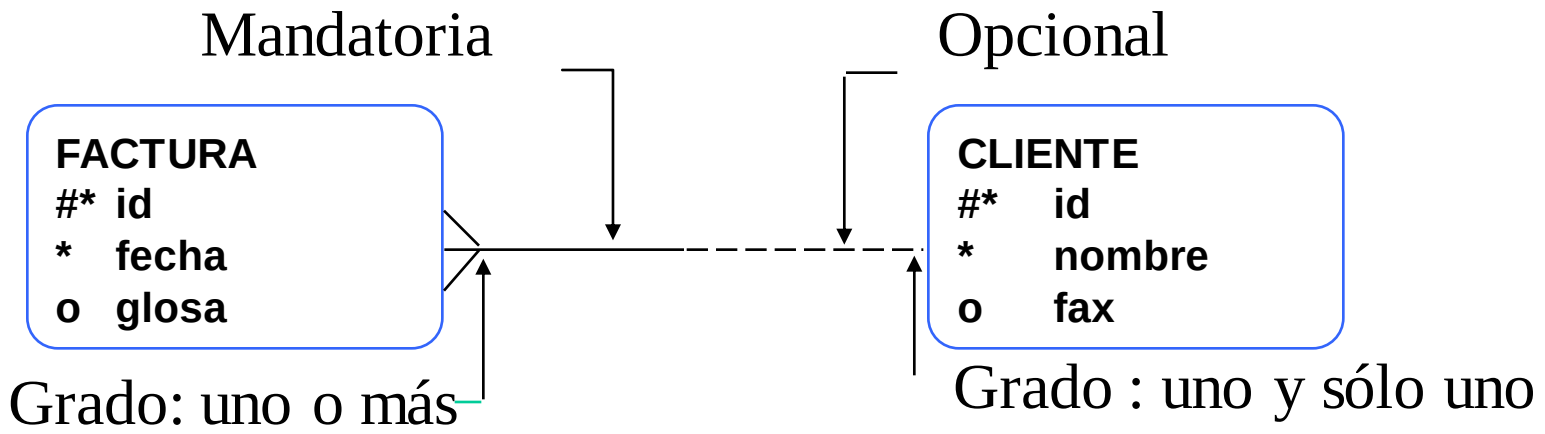
- Situaciones
  - ". . . Un vendedor le puede vender artículos a uno o más clientes . ."
  - ". . . Algunos vendedores aun no han sido asignados a clientes . . ."

Convenciones

- Entidad
  - Se encierra en una caja
  - Posee un solo nombre, generalmente un sustantivo
- Atributo
  - nombre en singular
  - Si es indispensable, se coloca un "*"
  - Si es opcional, un "o"
- Relación
  - Identificador único (UID)
  - Llave primaria marcada con "#"
  - Llave secundaria marcada con "(#)"

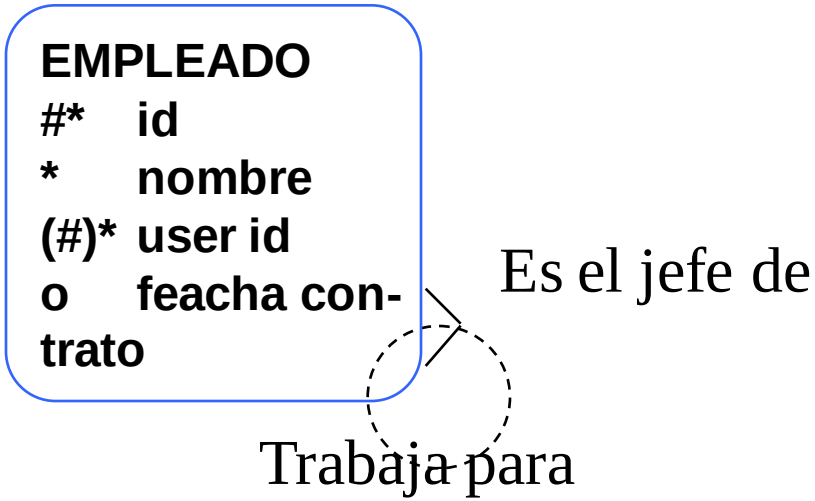
Ejemplo

- Nomenclatura
  - Las relaciones pueden ser del tipo mandatoria o del tipo opcional.
- Example
  - Cada FACTURA debe ser de uno y sólo un CLIENTE.
  - Cada CLIENTE puede tener una o más FACTURAS.



Relación Recursiva

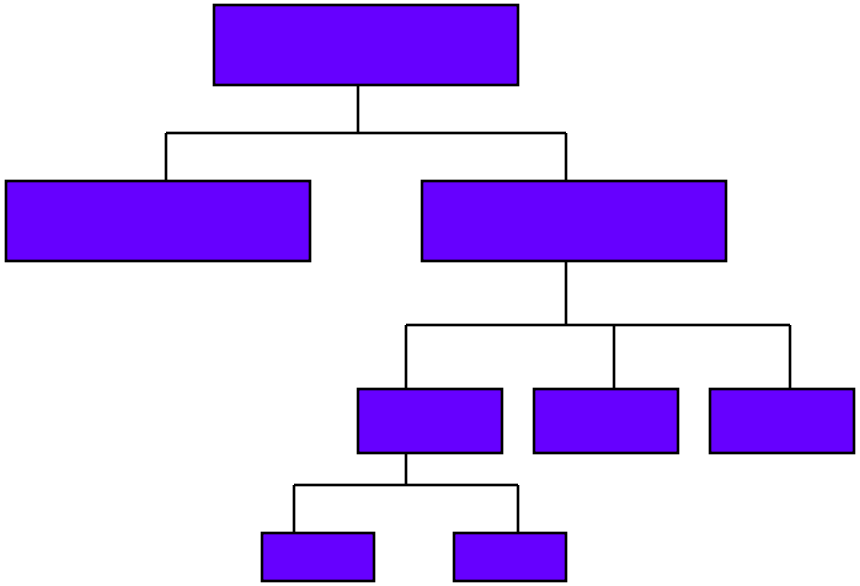
- Se trata de la relación que se genera entre la entidad y y si misma.
- Se representa a través de un bucle

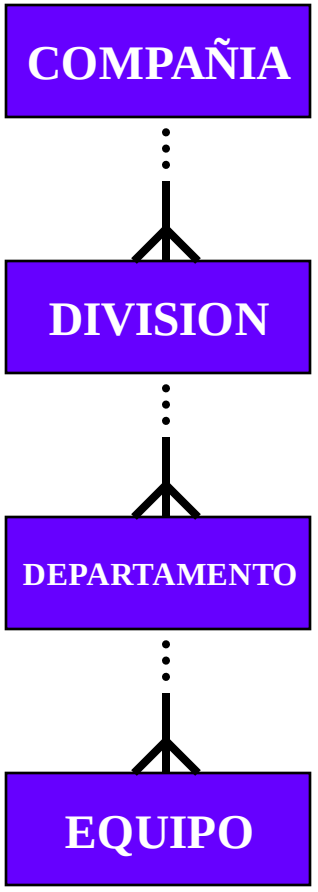


Tipos de Relaciones

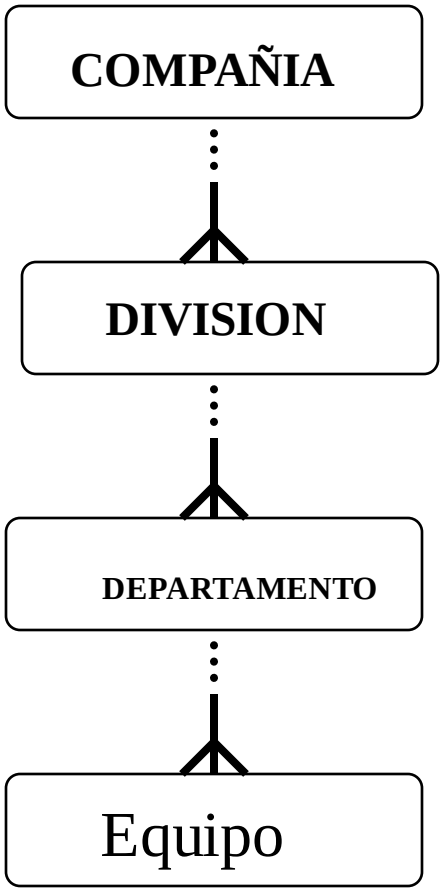
- Uno a uno
  - Define una relación biunivoca, por ej. Un computador tiene sólo una y sólo una Mather Board y una Mather Borad está en un único computador
  - Ejemplo: el matrimonio (se supone)
- Muchos a uno
  - Muchas instancias de una entidad pueden ser atendidas por una sola instancia de otra entidad
  - Ejemplo: Las facturas y su detalle
- Muchos a mucho
  - Muchas instancias de una entidad son atendidas por muchas instancias de otras entidades.
  - EEjemplo: Cursos y alumnos.

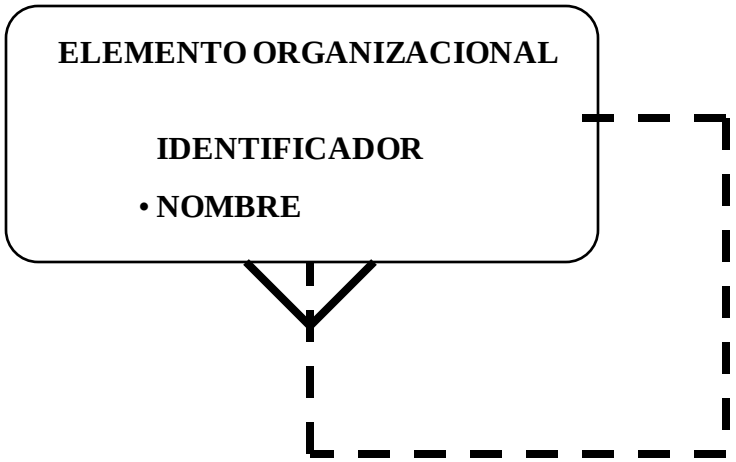
Relación Jerárquica :



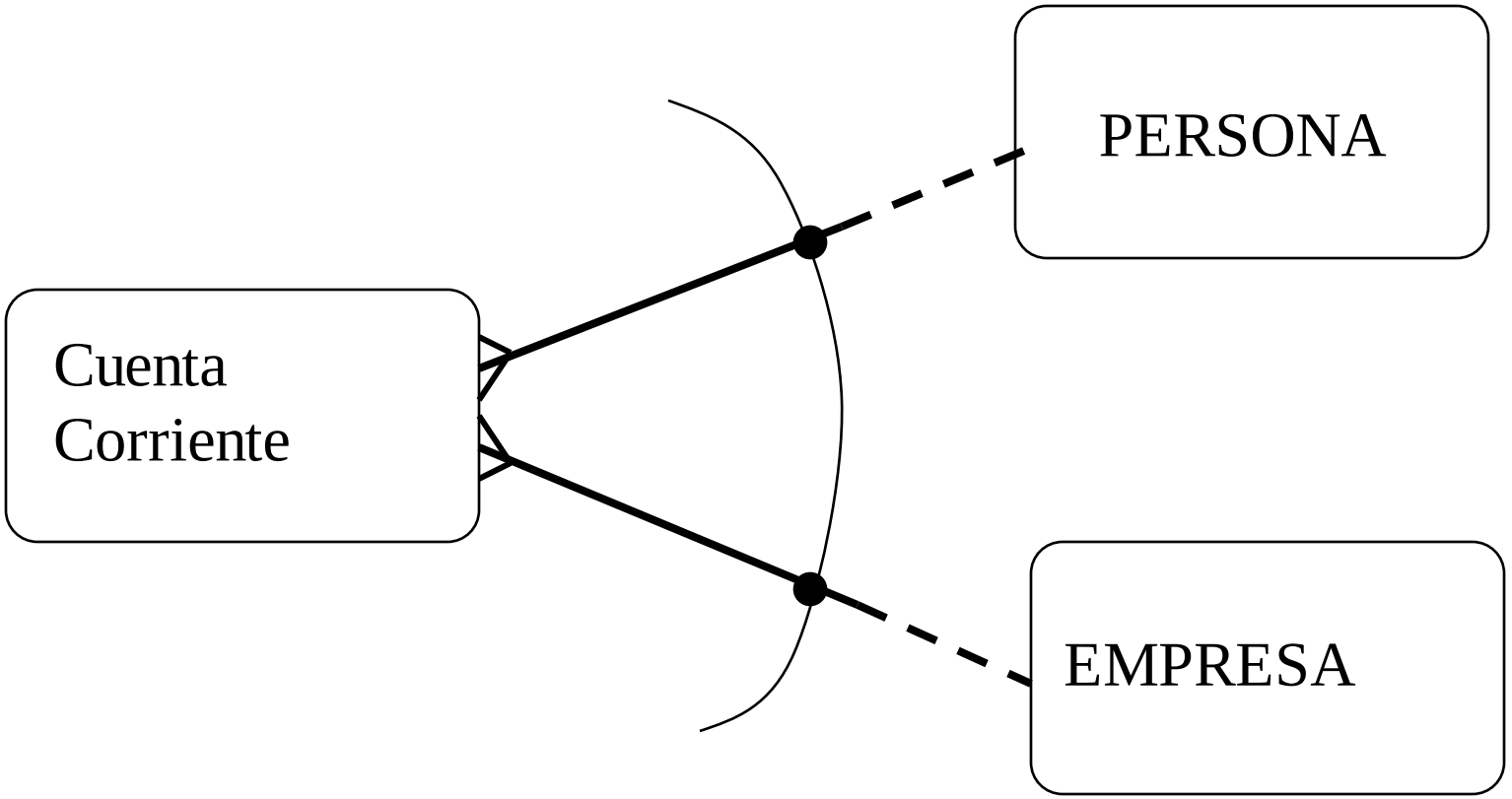


Otra representación de la jerarquia





Relación : ARCOS



ATRIBUTOS

Obtenga todos los Atributos Necesarios y Suficiente

\# : UID

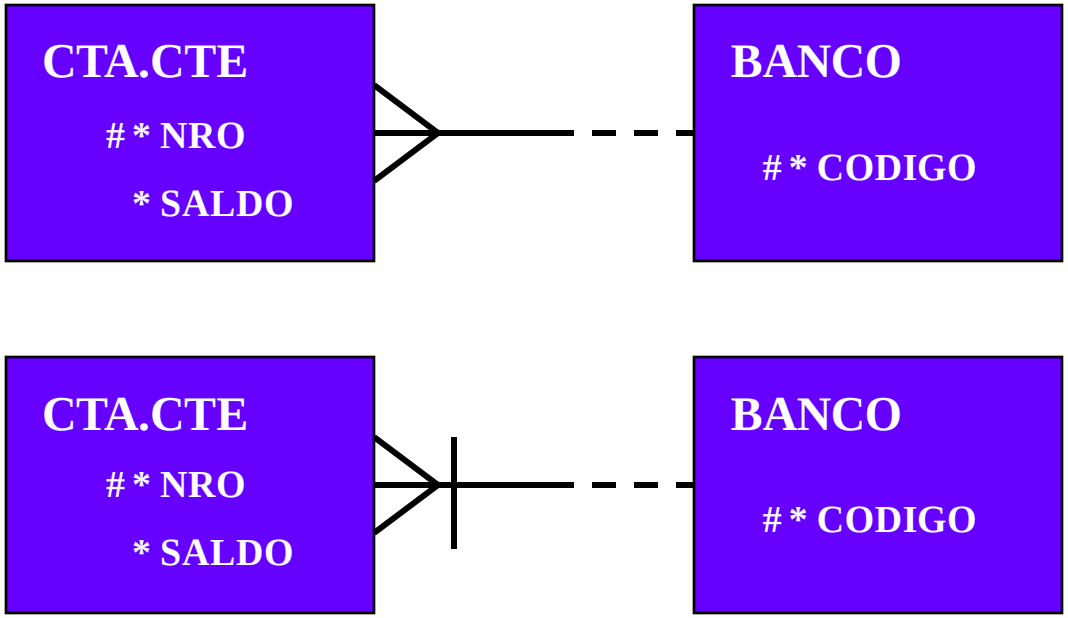
* : OBLIGATORIO

o : OPCIONAL

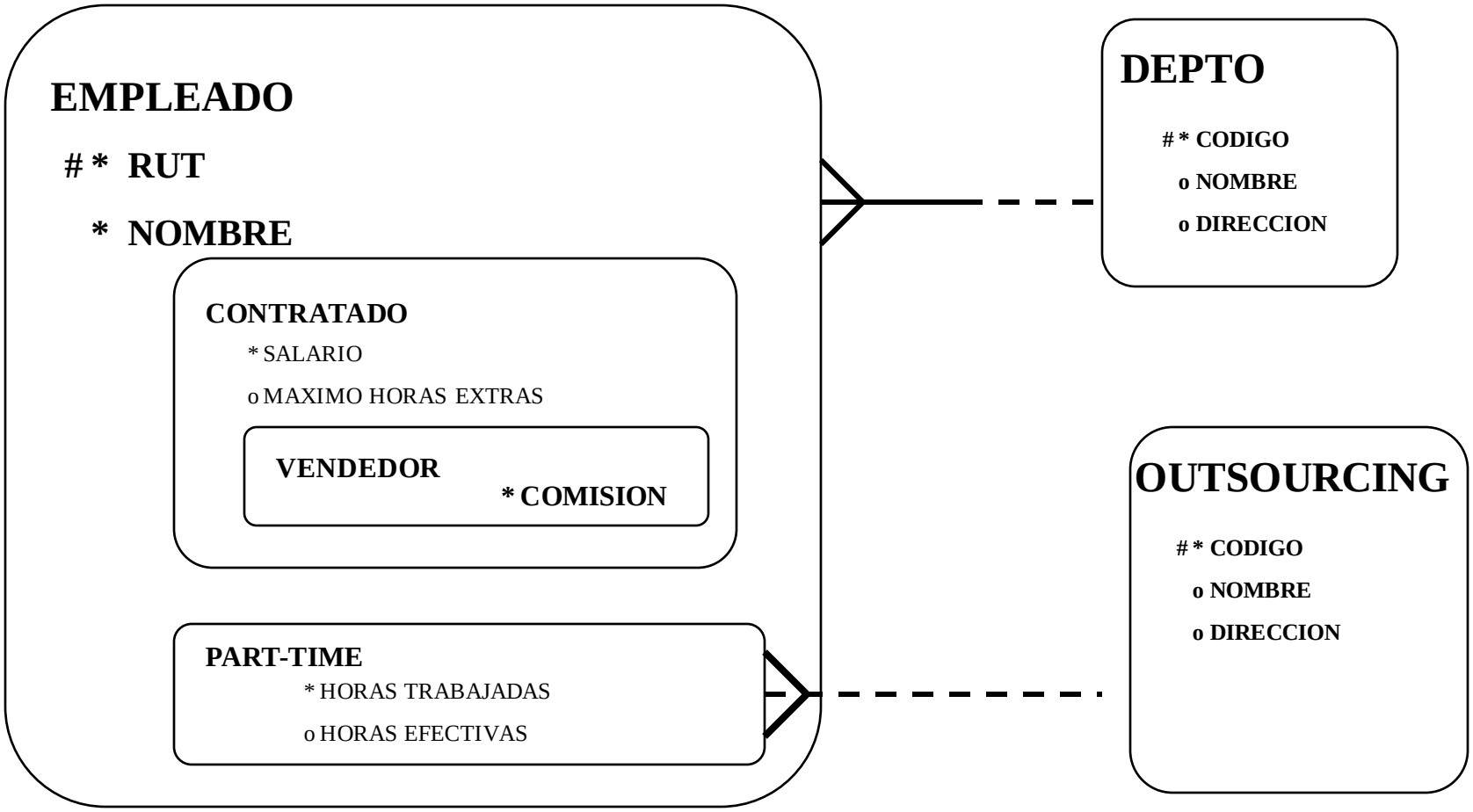
Atributos : UID

COMPUESTA

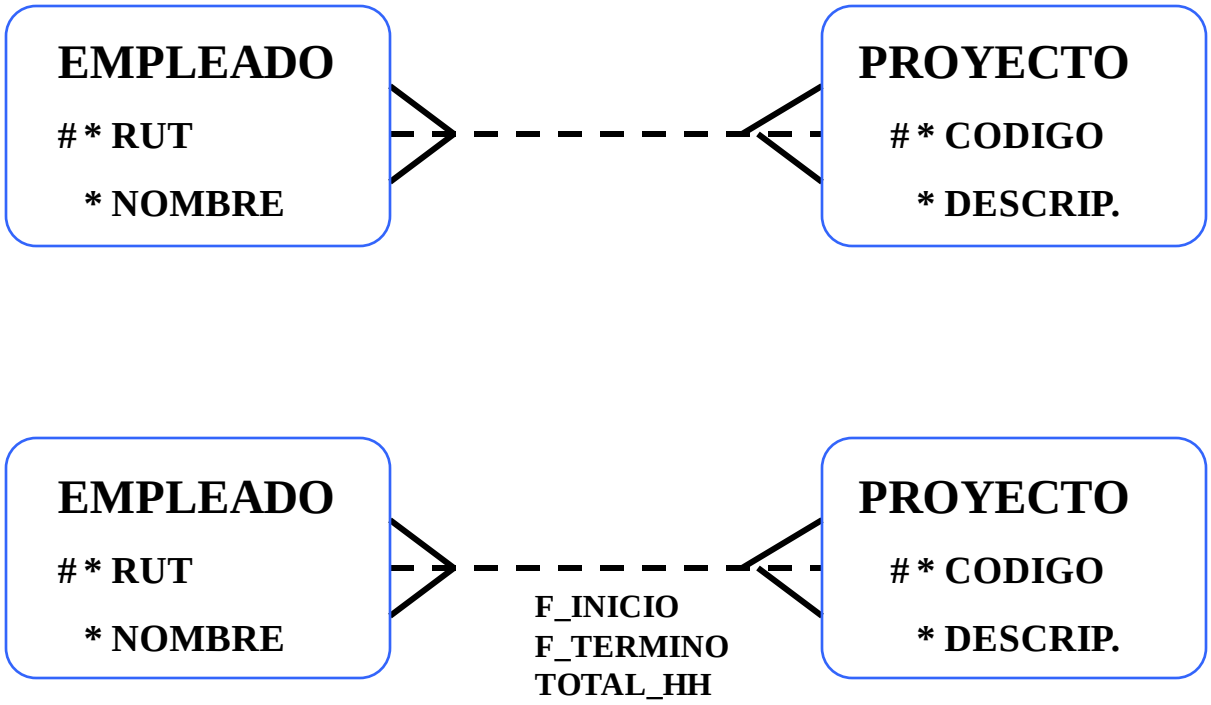
A TRAVES DE LA RELACION



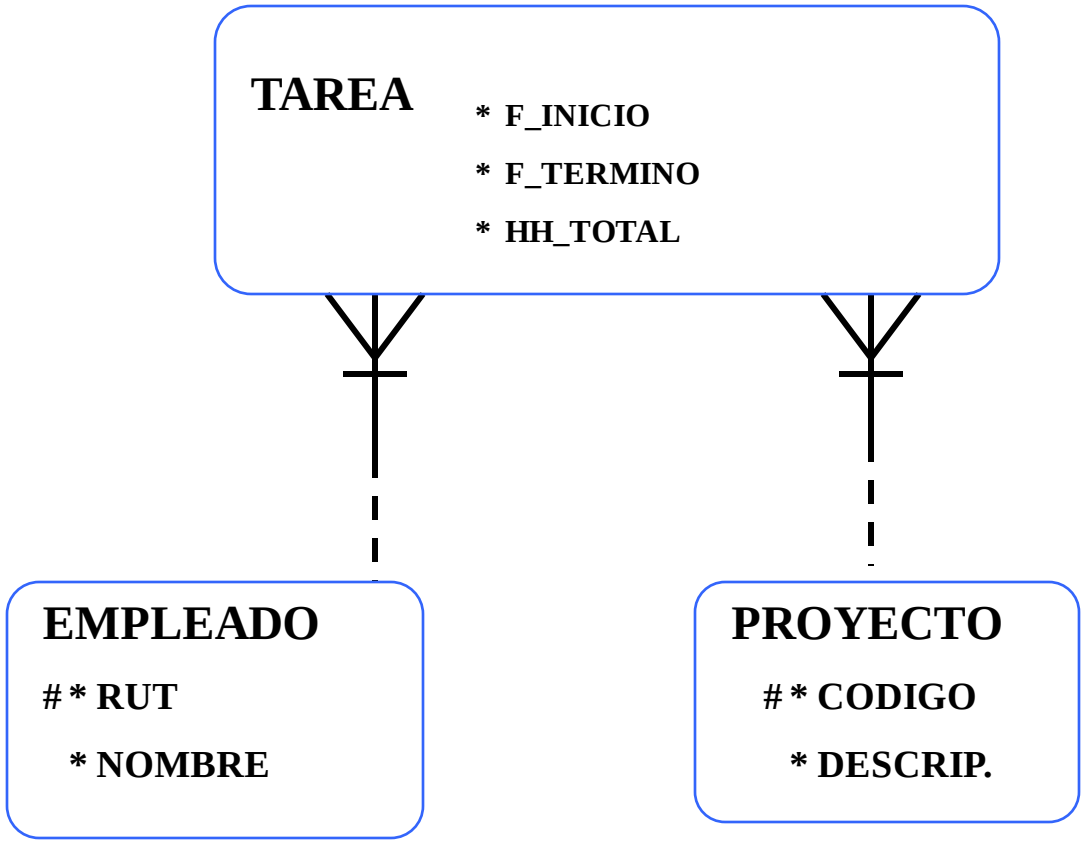
Atributos : Superclases y relaciones



Relación con atributos

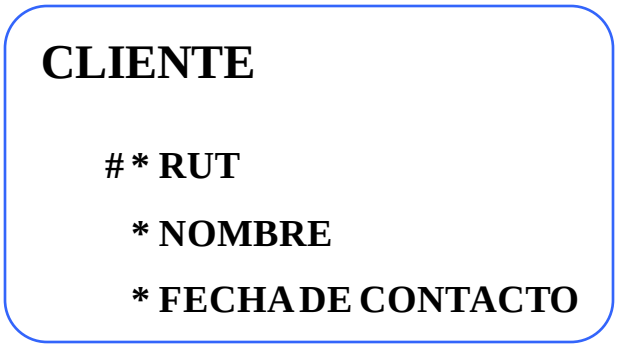


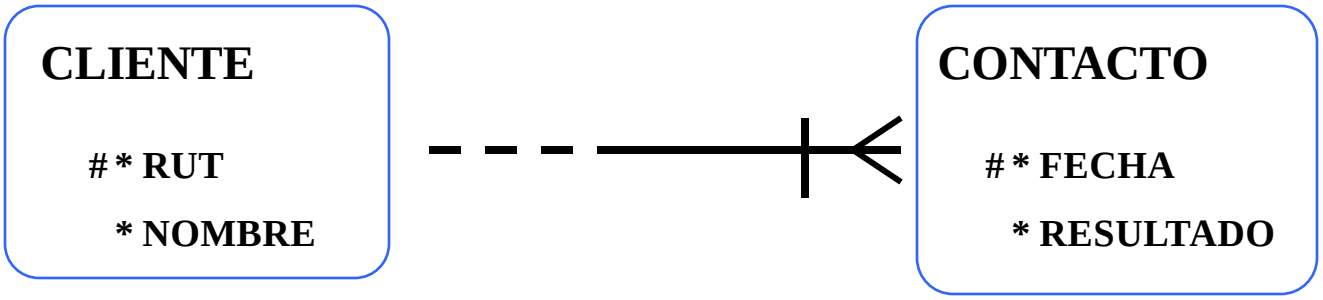
Relación con atributos



NORMALIZACION

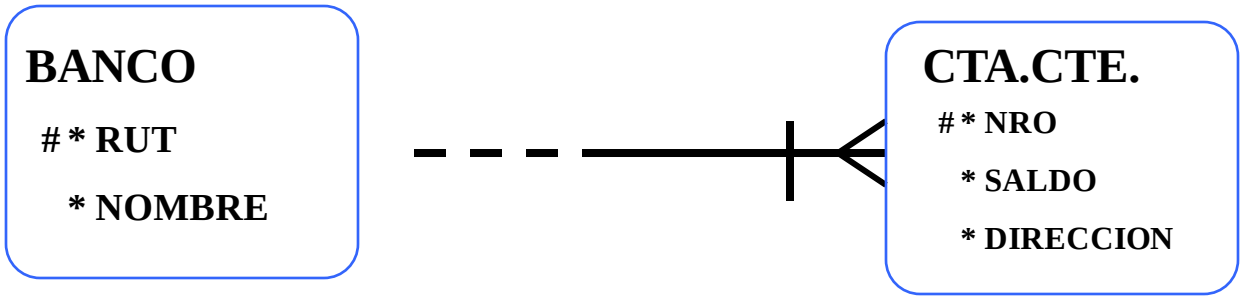
Forma Normal	Regla
1FN	Atributo atómico (Escalar)





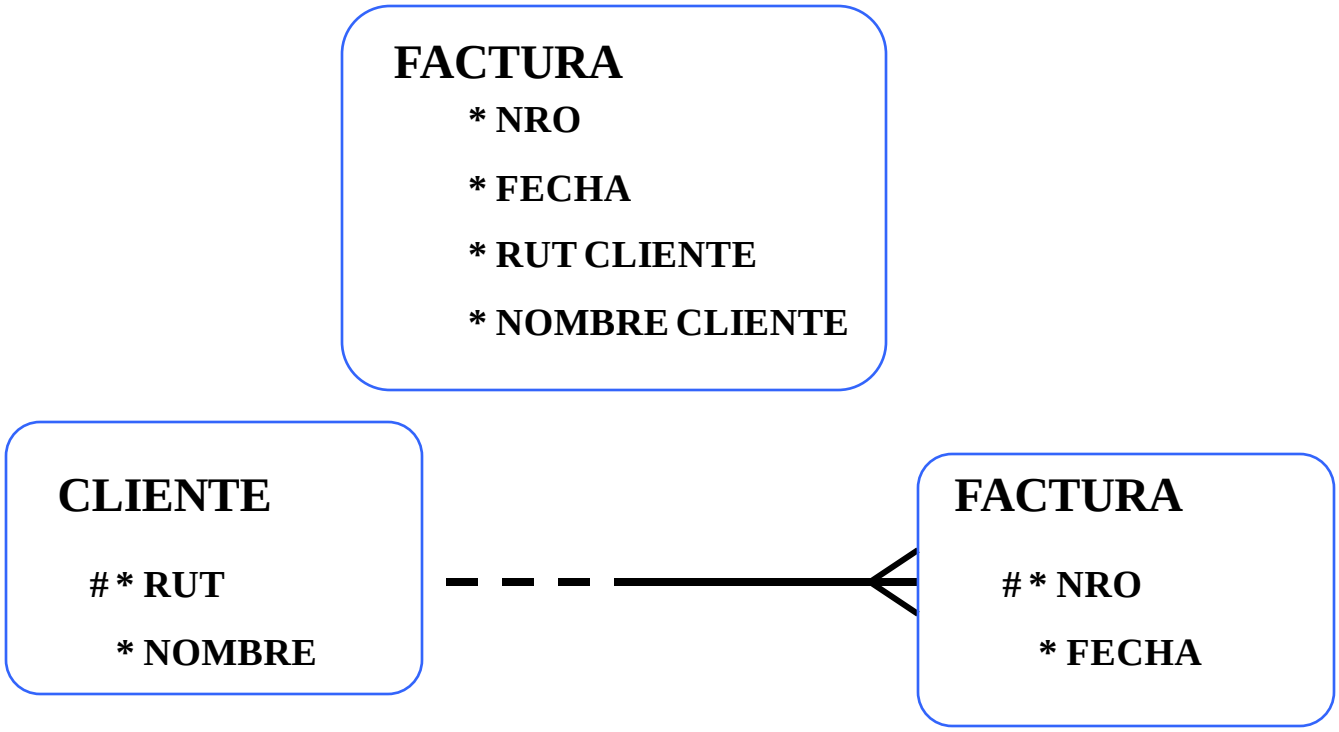
NORMALIZACION

Forma Normal	Regla
2FN	Si atributo es accesado a través de la UID, debe ser la UID completa

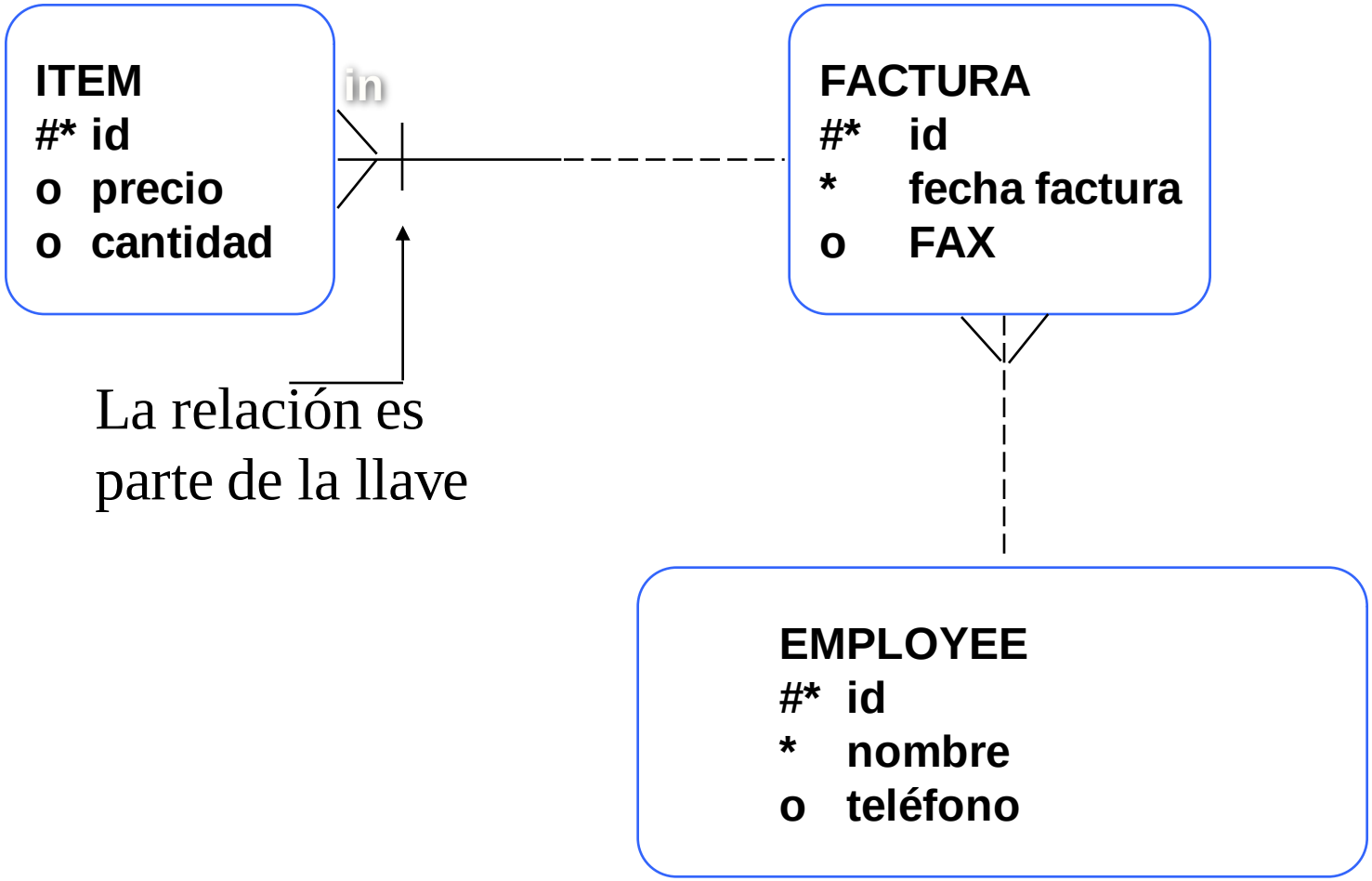


NORMALIZACION

Forma Normal	Regla
3FN ... B-C	Un atributo que no es parte de la UID solo puede ser accesado a través de la UID



Ejemplo del MER



Restrictores de integridad (constraint)

- Aseguran la consistencia de los datos
- Pueden ser hechos a nivel de la Base de Datos o en la aplicación
- Son llaves primarias, foráneas, etc.

Tipo	
Entidad	Aquellos campos que no forman parte de la PK pueden ser NULL
Referencial	No hay detalles sin maestros
Columna	Los datos deben ser del mismo tipo que la columna
otros	Triggers de bases de datos

Llave primaria (Primary Key :PK)

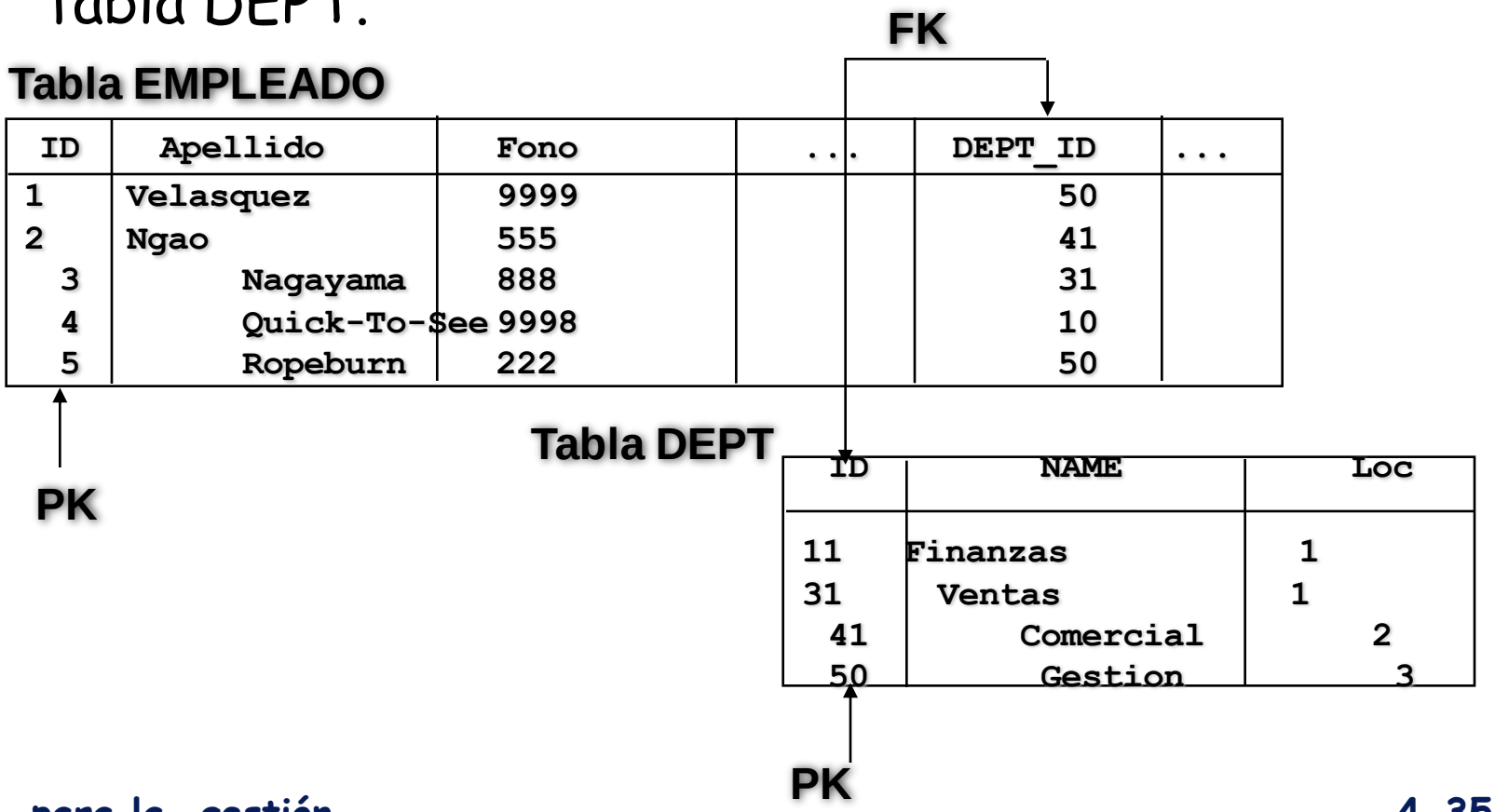
- Aseguran la NO existencia de filas duplicadas-
- Se trata de un atributo o conjunto de ellos, que definen en forma única a una fila.
- Implícitamente, definen una estructura de datos que permita asegurar la NO repetición de datos.
- Ejemplo: En la entidad ALUMNO, el número de matrícula identifica en forma única a un alumno.

Llave foránea (Foreign Key :FK)

- Es una columna o conjunto de estas, que referencian a una PK o UK en la misma tabla o en otra.
- Sirven para definir relaciones tales como las master-detail
- El valor contenido en la FK debe calcar con el valor de la PK que referencia o ser NULL
- Si una FK es parte de una PK, entonces no puede contener NULL.
- Ejemplo : No se puede borrar un maestro, sin haber borrado primero el detalle.

FK Ejemplo

La columna dept_id es la FK en la tabla EMPLEADO y referencia a la columna id en la tabla DEPT.



El MER llevado a la B.D.

- Las entidades definidas, se transforman en tablas, a partir de comandos que interpreta el SABDR.
- El nombre que tienen las tablas, coincide con el de las relaciones.
- Hay que crear objetos adicionales de apoyo, tales como :
  - Indices
  - Triggers
  - Vistas
  - Restrictores

Ejercicios

Una facultad posee alumnos, los cuales toman varios cursos durante un semestre. Los profesores que dictan los ramos, pertenecen a uno y sólo un departamento dentro de la facultad y pueden dictar más de un curso durante el semestre.

Es información importante de los alumnos, su nombre, matrícula, año ingreso, fecha de nacimiento, sexo y un registro de todos los cursos que ha tomado, registrando si los aprobó o no y en que semestre.

Los cursos que toma un alumno, tienen un código, un nombre y son dictado por un departamento en específico. Poseen además, un identificador de sección, por cuanto puede ser dictado más de una vez en el semestre, pero solo pos un profesor a la vez.

Para tomar un curso, el alumno debe haber cursado previamente otros y haberlos aprobado.

Establezca un modelo entidad relación para el problema anteriormente planteado. En caso de ser necesario, haga los supuestos que estime convenientes.

Introducción a SQL

Objetivos

- Propósito e importancia de SQL.
- Como recuperar datos desde una Base de Datos usando SELECT :
  - Uso de la componente WHERE para agregar condiciones a las Consultas.
  - Ordenar los resultados de las Consultas usando ORDER BY.
  - Uso de Funciones Adicionales.
  - Agrupar datos usando GROUP BY y HAVING.
  - Uso de Sub-Consultas.

Objetivos

- Usar Join entre tablas.
- Operaciones de Conjuntos :
  - UNION, INTERSECT, EXCEPT
- Como realizar transacciones sobre la Base de Datos usando :
  - INSERT, UPDATE, and DELETE.
- Tipos de datos soportados por SQL-92.
- Como crear y borrar tablas.

Objetivos de SQL

- Un lenguaje de base de datos debe permitir :
  - Crear las estructuras de la Base de datos y sus relaciones.
  - Realizar inserción, modificación y borrado de datos.
  - Realizar consultas simples y complejas.
- Estas tareas se deben realizar con un mínimo esfuerzo del usuario.
- La estructura y sintaxis de comandos debe ser fácil de aprender.
- Debe ser portable.

Objetivos

- **Se conforma con palabras inglesas estándares.:**

```
CREATE TABLE staff   (sno VARCHAR(5),
                      lname VARCHAR(15),
                      salary DECIMAL(7,2));
INSERT INTO          staff
VALUES ('SG16', 'Brown', 8300);
SELECT sno, lname, salary
FROM staff
WHERE salary > 10000;
```

Historia de SQL

- En 1974, en el Laboratorio de IBM en San Jose D. Chamberlin definió un lenguaje llamado 'Structured English Query Language' o SEQUEL.
- Una versión mejorada de SEQUEL/ fue definida en 1976, pero su nombre fue cambiado por razones legales SQL.
- IBM produjo posteriormente un DBMS prototipo llamado System R, basado en SEQUEL/2
- En 1987, ANSI e ISO publicaron un estándar inicial para SQL.
- En 1989, ISO publicó una adición que definió ' una característica de realce de la integridad '.
- En 1992, la primera revisión importante al estándar de ISO ocurrió, designado SQL2 o SQL/92.

Sentencia SELECT

```
SELECT  [DISTINCT | ALL]
  {* | [column_expression [AS new_name]] [,...] }
FROM            table_name [alias] [, ...]
[WHERE  condition]
[GROUP  BY      column_list]        [HAVING
  condition]
[ORDER BY       column_list]
```

Sentencia SELECT

FROM	Especifica las tablas que se usaran
WHERE	Establece los filtros.
GROUP BY	Permite agrupar los datos.
HAVING	Permite generar filtros sobre los grupos de datos.
SELECT	Especifica las columnas que se consultaran.
ORDER BY	Especifica el orden de los datos.

Ejemplo1 Todas las Columnas y Filas

Listar todos los datos de la tabla Staff.

```
SELECT sno, fname, lname, address, tel_no,
                position, sex, dob, salary, nin, bno
FROM    staff;
```

- Se puede usar * como abreviación para todas las columnas :

```
SELECT * FROM staff;
```

Ejemplo1 Todas las Columnas y Filas

Table 13.1 Result table for Example 13.1.

sno	*fname*	*lname*	*address*	*tel_no*	*position*	*sex*	*dob*	*salary*	*nin*	*bno*
SL21	John	White	19 Taylor St, Cranford, London	0171–884–5112	Manager	M	1-Oct-45	30000.00	WK442011B	B5
SG37	Ann	Beech	81 George St, Glasgow PA1 2JR	0141–848–3345	Snr Asst	F	10-Nov-60	12000.00	WL432514C	B3
SG14	David	Ford	63 Ashby St, Partick, Glasgow Gll	0141–339–2177	Deputy	M	24-Mar-58	18000.00	WL220658D	B3
SA9	Mary	Howe	2 Elm Pl, Aberdeen AB2 3SU		Assistant	F	19-Feb-70	9000.00	WM532187D	B7
SG5	Susan	Brand	5 Gt Western Rd, Glasgow G12	0141–334–2001	Manager	F	3-Jun-40	24000.00	WK588932E	B3
SL41	Julie	Lee	28 Malvern St, London NW2	0181–554–3541	Assistant	F	13-Jun-65	9000.00	WA290573K	B5

(6 rows)

Ejemplo 1: con limites de despliegue.

- SELECT * FROM staff limit 2.

SL21	John	White	19 Taylor St, Cranford, London	0171-884-5112	Manager	M	1-Oct-45	30000.00	WK442011B	B5
SG37	Ann	Beech	81 George St, Glasgow PA1 2JR	0141-848-3345	Snr Asst	F	10-Nov-60	12000.00	WL432514C	B3

Ejemplo2 Uso de DISTINCT

Lista los números de propiedad de todas las propiedades vistas.

SELECT pno

FROM viewing;

Table 13.3(a) Result table for Example 13.3 with duplicates.

pno
PA14
PG4
PG4
PA14
PG36

(5 rows)

Ejemplo2 Uso de DISTINCT

- **Use DISTINCT para eliminar los codigos duplicados:**

```
SELECT DISTINCT pno
FROM viewing;
```

Table 13.3(b) Result table for Example 13.3 with duplicates eliminated.

pno
PA14
PG4
PG36

(3 rows)

Ejemplo3 Campos Calculados

Produce una lista con los sueldos mensuales de todos los empleados.

```
SELECT sno, fname, lname, salary/12
FROM staff;
```

Ejemplo3 Campos Calculados

Table 13.4 Result table for Example 13.4.

sno	*fname*	*lname*	*col4*
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

(6 rows)

- **Para renombrar las Columnas usar la opcion AS:**

```
SELECT sno, fname, lname,
       salary/12 AS monthly_salary
FROM staff;
```

Ejemplo4
Condición de Búsqueda

Listar todos los empleados con un sueldo mayor a 10.000.

```
SELECT sno, fname, lname, position, salary
FROM staff
WHERE salary > 10000;
```

Ejemplo4 Condición de Busqueda

Table 13.5 Result table for Example 13.5.

sno	*fname*	*lname*	*position*	*salary*
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Snr Asst	12000.00
SG14	David	Ford	Deputy	18000.00
SG5	Susan	Brand	Manager	24000.00

(4 rows)

Ejemplo5 Miembro de un Conjunto

Listar a todos los encargados y directores adjuntos.

```
SELECT sno, fname, lname, position
FROM staff
WHERE position IN ('Manager', 'Deputy');
```

Ejemplo5 Miembro de un Conjunto

Table 13.8 Result table for Example 13.8.

sno	*fname*	*lname*	*position*
SL21	John	White	Manager
SG14	David	Ford	Deputy
SG5	Susan	Brand	Manager

(3 rows)

Ejemplo5 Miembro de un Conjunto

- Existe une versión con negación (NOT IN).
- IN no agrega funcionalidades nuevas ya que se podría conseguir lo mismo con :

```
SELECT sno, fname, lname, position
FROM staff
WHERE position='Manager' OR position='Deputy';
```

- IN es más eficiente con mas elementos.

Ejemplo6 Comparando Patrones

Encontrar todos los empleados con la palabra 'Glasgow' en su dirección.

```
SELECT sno, fname, lname, address, salary
FROM staff
WHERE address LIKE '%Glasgow%';
```

Ejemplo6 Comparando Patrones

Table 13.9 Result table for Example 13.9.

sno	*fname*	*lname*	*address*	*salary*
SG37	Ann	Beech	81 George St, Glasgow PA1 2JR	12000.00
SG14	David	Ford	63 Ashby St, Partick, Glasgow G11	18000.00
SG5	Susan	Brand	5 Gt Western Rd, Glasgow G12	24000.00

(3 rows)

Ejemplo7 Comparando Patrones

- SQL tiene dos patrones especiales para la comparación de símbolos:
  - %: sequencia de 0 o mas caracteres;
  - _ (underscore): cualquier caracter simple.
- LIKE '%Glasgow%' significa una secuencia de caracteres de algún largo que contiene '*Glasgow*'.

Ejemplo7 Ordenamiento por múltiples columnas

Table 13.12(a) Result table for Example 13.12 with one sort key.

pno	*type*	*rooms*	*rent*
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

(6 rows)

Ejemplo8 Ordenamiento por múltiples columnas

- Cuatro columnas en la lista - sin un orden descendiente especificado -el sistema ordenará las filas en el orden en que él escoja.
- Ordenar por orden de renta, específicamente en orden descendiente:

```
SELECT pno, type, rooms, rent
FROM property_for_rent
ORDER BY type, rent DESC;
```

Ejemplo8 Multiple Column Ordering

Table 13.12(b) Result table for Example 13.12 with two sort keys.

pno	*type*	*rooms*	*rent*
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

(6 rows)

SELECT Sentencias agregadas

- Los estándares ISO definen 5 sentencias agregadas:

COUNT retorna el número de valores en una columna específica.

SUM Retorna la suma de valores de una columna

AVG Retorna el promedio de valores de una columna

MIN Retorna el valor más pequeño de una columna

MAX Retorna el valor más grande de una columna

SELECT Sentencias agregadas

- Cada una opera sobre una columna de una tabla y retorna un sólo valor.
- COUNT, MIN, y MAX aplican sobre campos numéricos y no-numéricos, pero SUM y AVG deben usarse sólo en campos numéricos.
- Además con COUNT(*), cada función elimina primero los nulos y opera sólo con los valores que quedan no-nulos.

SELECT Statement - Aggregates

- COUNT(*) cuenta todas las filas de una tabla, sin importar si anula o no los valores duplicados que existan.
- Se puede usar DISTINCT antes del nombre de una columna para eliminar los registros duplicados.
- DISTINCT no tiene efecto con MIN/MAX, pero puede tenerlo con SUM/AVG.
- Funciones agregadas solo pueden ser usadas en SELECT and HAVING.

Ejemplo9 Use of MIN, MAX, AVG

Encontrar el máximo, mínimo y promedio de los salarios del staff.

```
SELECT MIN(salary) AS min,
  MAX(salary) AS max,
  AVG(salary) AS avg
FROM staff;
```

Table 13.15 Result table for Example 13.15.

count	*sum*
2	54000.00

(1 row)

INSERT

```
INSERT INTO table_name [ (column_list) ]
VALUES (data_value_list)
```

- *column_list* es opcional.
- Si se omite, SQL assume una lista de todas las columnas en su orden original.

Ejemplo10 INSERT ... VALUES

```
INSERT INTO staff
VALUES ('SG16', 'Alan', 'Brown',
    '67 Endrick Rd, Glasgow G32 8QX',
    '0141-211-3001', 'Assistant', 'M', '25-May-57',
    8300, 'WN848391H', 'B3');
```

Ejemplo11 INSERT using Defaults

```
INSERT INTO staff (sno, fname, lname, position,
                   salary, bno)
VALUES ('SG44', 'Anne', 'Jones', 'Assistant',
                   8100, 'B3');
```

Ejemplo11 INSERT using Defaults

```
INSERT INTO staff
VALUES ('SG44', 'Anne', 'Jones', NULL, NULL,
  'Assistant', NULL, NULL, 8100, NULL, 'B3');
```

UPDATE

```
UPDATE table_name
SET column_name1 = data_value1
    [, column_name2 = data_value2...]
[WHERE search_condition]
```

- *table_name*
- SET clause

UPDATE

- WHERE clause is optional:
  - If omitted, named columns are updated for all rows in table.
  - If specified, only those rows that satisfy *search_condition* are updated.
- New *data_value(s)* must be compatible with data type for corresponding column.

Ejemplo12 UPDATE All Rows

Give all staff a 3% pay increase.

```
UPDATE staff
SET salary = salary*1.03;
```

Ejemplo13 UPDATE Specific Rows

Give all Managers a 5% pay increase.

```
UPDATE staff
SET salary = salary*1.05
WHERE position = 'Manager';
```

- WHERE clause finds rows that contain data for Managers. Update is applied only to these particular rows.

Ejemplo14 UPDATE Multiple Columns

Promote David Ford (sno = 'SG14') to Manager and change his salary to 18,000.

```
UPDATE staff
SET position = 'Manager', salary = 18000
WHERE sno = 'SG14';
```

DELETE

```
DELETE FROM table_name
[WHERE search_condition]
```

- *table_name* can be name of a base table or an updatable view.
- *search_condition* is optional; if omitted, all rows are deleted from table. This does not delete table. If *search_condition* is specified, only those rows that satisfy condition are deleted.

Ejemplo15 DELETE Specific Rows

Delete all viewings that relate to property PG4.

```
DELETE FROM viewing
WHERE pno = 'PG4';
```

Ejemplo16 DELETE All Rows

Delete all records from the Viewing table.

```
DELETE FROM viewing;
```

ISO SQL Data Types

Table 13.36 ISO SQL data types.

Data type	*Declarations*			
character	CHAR,	VARCHAR		
bit	BIT,	BIT VARYING		
exact numeric	NUMERIC,	DECIMAL,	INTEGER,	SMALLINT
approximate numeric	FLOAT,	REAL,	DOUBLE PRECISION	
datetime	DATE,	TIME,	TIMESTAMP	
interval	INTERVAL			

CREATE TABLE (Basic)

```
CREATE TABLE table_name
(col_name data_type [NULL | NOT NULL] [,...])
```

- Creates a table with one or more columns of the specified *data_type*.
- NULL (default) indicates whether column can contain *nulls*.
- With NOT NULL, system rejects any attempt to insert a null in the column.

CREATE TABLE (Basic)

- Primary keys should always be specified as NOT NULL.
- Foreign keys are often (but not always) candidates for NOT NULL.

Ejemplo17 CREATE TABLE

```
CREATE TABLE staff(
        sno             VARCHAR(5)          NOT NULL,
        fname           VARCHAR(15)         NOT NULL,
        lname           VARCHAR(15)  NOT NULL,
        address         VARCHAR(50),
        tel_no          VARCHAR(13),
        position        VARCHAR(10)  NOT NULL,
        sex             CHAR,
        dob             DATETIME,
        salary          DECIMAL(7,2) NOT NULL,
        nin             CHAR(9),
        bno             VARCHAR(3)          NOT NULL);
```

Ejemplo18 CREATE TABLE

```
CREATE TABLE property_for_rent(
        pno             VARCHAR(5)          NOT NULL,
        street          VARCHAR(25)         NOT NULL,
        area            VARCHAR(15),
        city            VARCHAR(15)  NOT NULL,
        pcode           VARCHAR(8),
        type            CHAR(1)             NOT NULL,
        rooms           SMALLINT            NOT NULL,
        rent            DECIMAL(6,2)        NOT NULL,
        ono             VARCHAR(5)          NOT NULL,
        sno             VARCHAR(5),
        bno             VARCHAR(3)          NOT NULL);
```

DROP TABLE

DROP TABLE tbl_name [RESTRICT | CASCADE]

e.g. DROP TABLE property_for_rent;

- Removes named table and all rows within it.
- With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.
- With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).

SUB-SELECT

- Es para operar sobre sub-conjuntos. Ver sección de subselect del manual.

```
SELECT *
FROM EMPLEADO
WHERE
  RUT IN (
        SELECT RUT
        FROM CLIENTE
        WHERE  EVALUACION='MALA')
```

SUB-SELECT

- Tres tipos de uso en las condiciones (WHERE)
  - X >= (SELECT …) en el caso que el select entregue **1** solo valor a comparar.
  - (X1,X2,X3)>= (SELECT …) en el caso que el select entregue VARIOS valores a comparar.
  - X IN (SELECT …) se buscan los X que se encuentren en el conjunto del select.
  - X >= SOME (SELECT …) se busca los X que cumplen la comparacion (>=) para ALGUN elemento del conjunto del select.
  - X >= ANY (SELECT …) se busca los X que cumplen la comparacion (>=) para TODOS elemento del conjunto del select.

VARIABLES definidas por el usuario.

- Ver sección 9.3 del manual.

```
SET @x=0;
SELECT @x:=@x+1, nombre
FROM CLIENTE
Order by pago desc;
```

```
SET @x = (SELECT MAX(SUELDO) FROM EMPLEADO);
SELECT * FROM EMPLEADO WHERE SUELDO>@X-500;
```

Observación: no usar 2 veces una variable en un select con asignación puede entregar valores random.

Comparación de fechas en MySQL

- UNIX_TIMESTAMP()
- SEC_TO_TIME()
- Fechas se pueden comparar si estan en el mismo formato.

Estructuras de acceso rápido

Índices

- La tecnología que almacena la data debe ser capaz de soportar fácil indexación.
- Algunas de las técnicas que a menudo tienen sentido son el soporte de:
  - Índices secundarios
  - Índices sparse
  - Índices dinámicos, temporales, etc.
- Además el costo de crear y usar el índice no debe ser significativo.

Índices (2)

- A la vez el overhead de monitorear la data no debe ser tan alto ni el monitoreo tan complejo que impida que se realice cuando sea necesario.
- El monitoreo de la actividad del data warehouse determina que data ha y no ha sido usada.
- El monitoreo del data warehouse determina factores como los siguientes:
- Si es necesario realizar una reorganización
- Si un índice está pobremente estructurado
- Si hay mucha o no suficiente data en overflow
- La composición estadística del acceso a los datos
- Espacio restante disponible

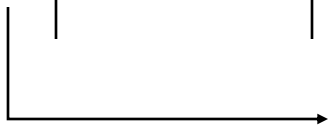
Indices (3)

- Están basados en la teoría de árboles de búsqueda.
- Cuando se indexa una o mas columnas de una tabla, se crea un índice que replica el contenido de las columnas en su estructura y genera enlaces hacia los registros de la tabla.
- Los árboles más usados en este ámbito, son los B*Trees.

Ejemplo de indexación

emp

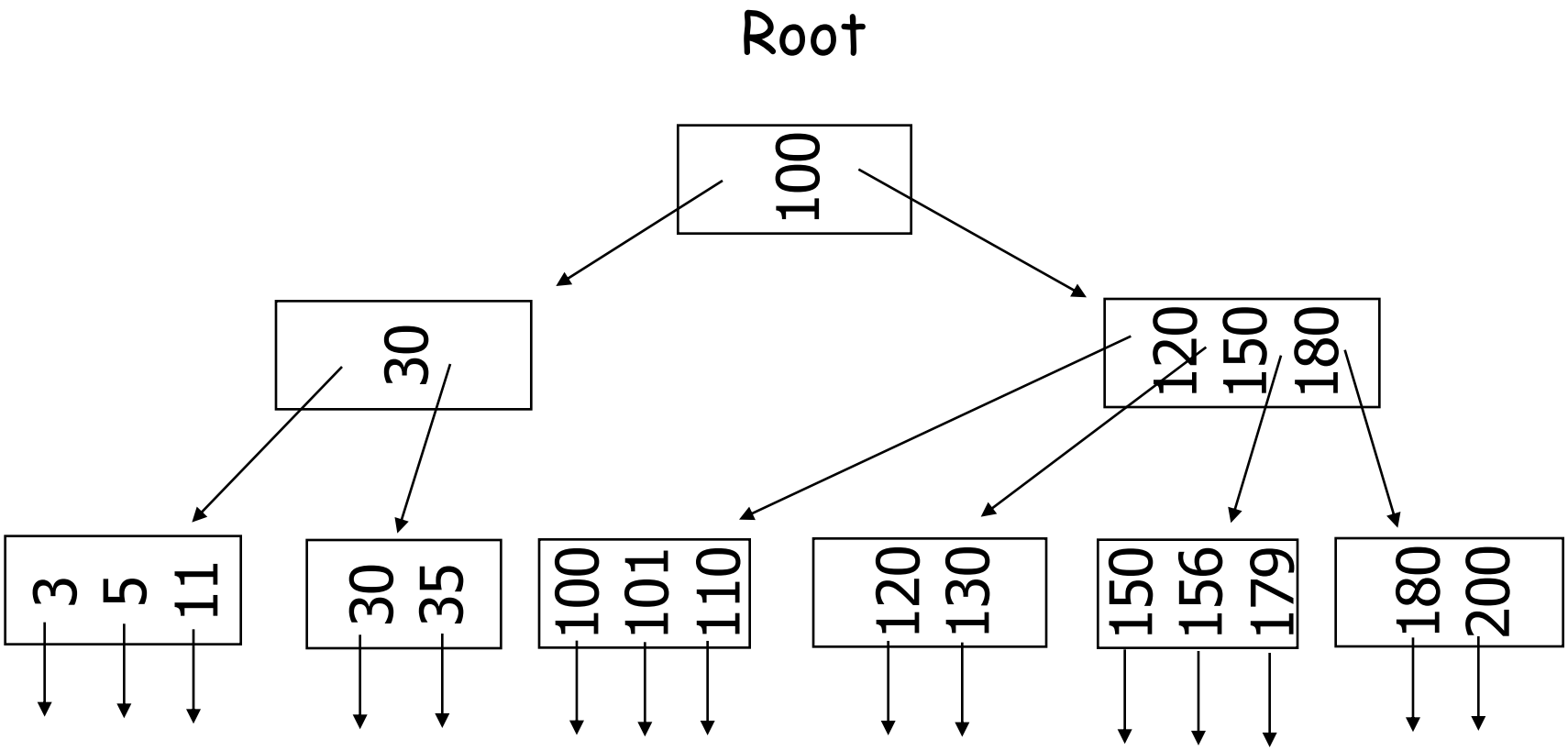
Id	Nombre	Dirección	Sueldo
3	Eugenia		
5	Anita		
11	Rodrigo		
30	Joaquín		
35			
100			
101			
110			
120			



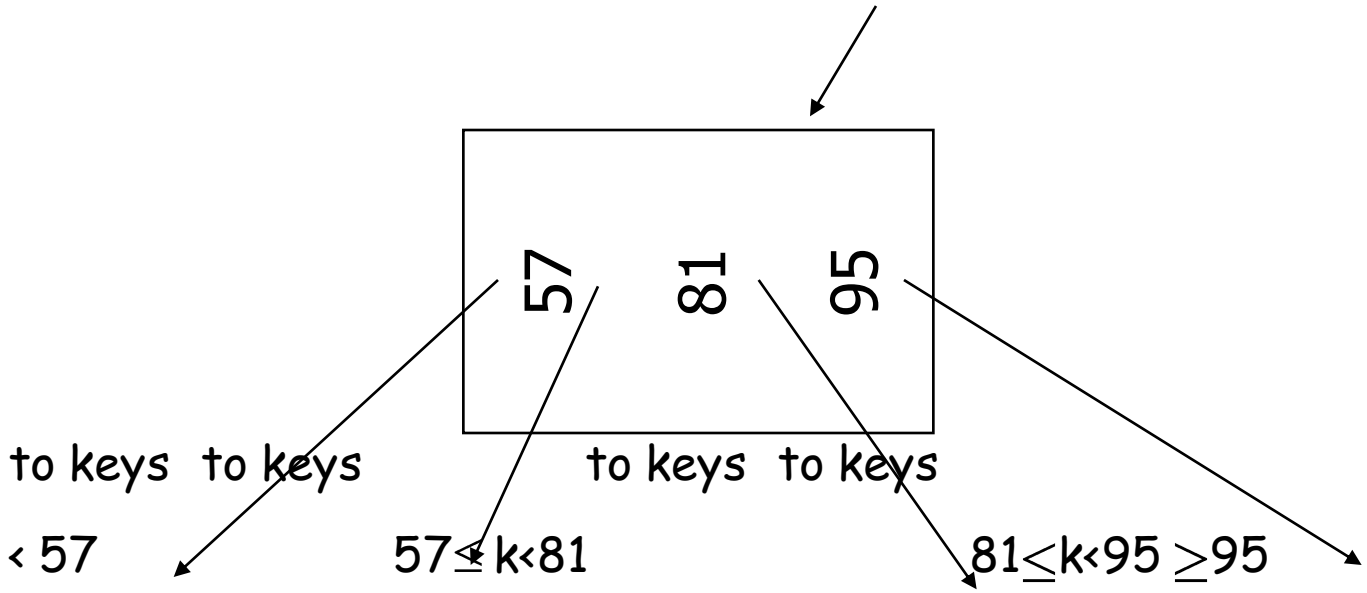
PK, se implementa un índice tácito

B+Tree Example

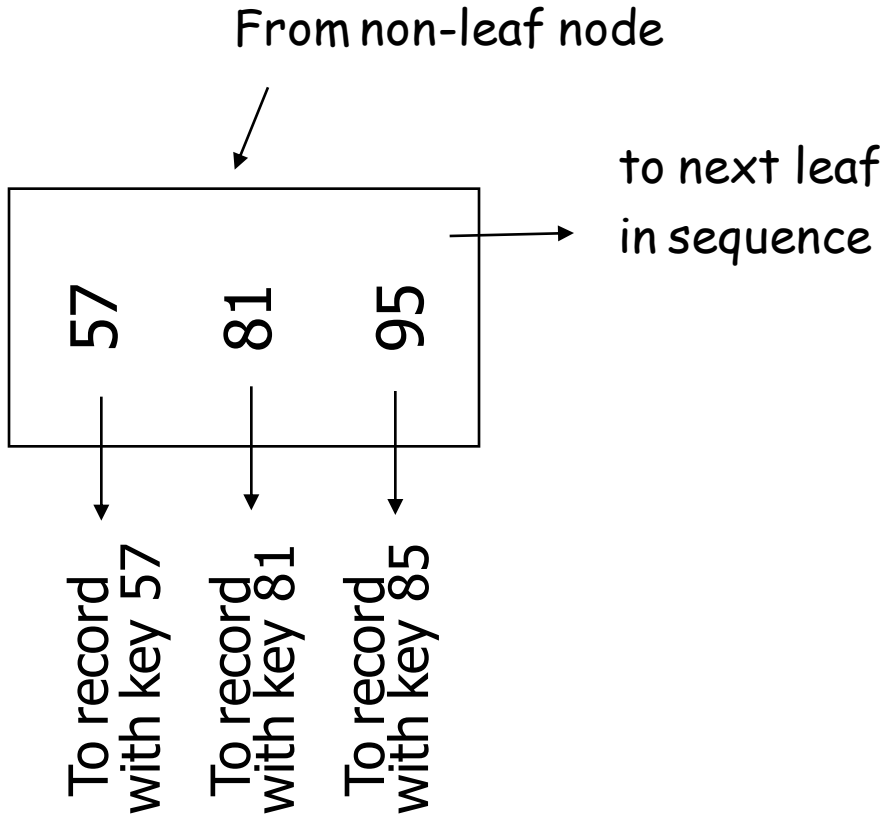
n=3



Ejemplo de un nodo



Hoja


From non-leaf node
to next leaf
in sequence
57
81
95
To record with key 57
To record with key 81
To record with key 85

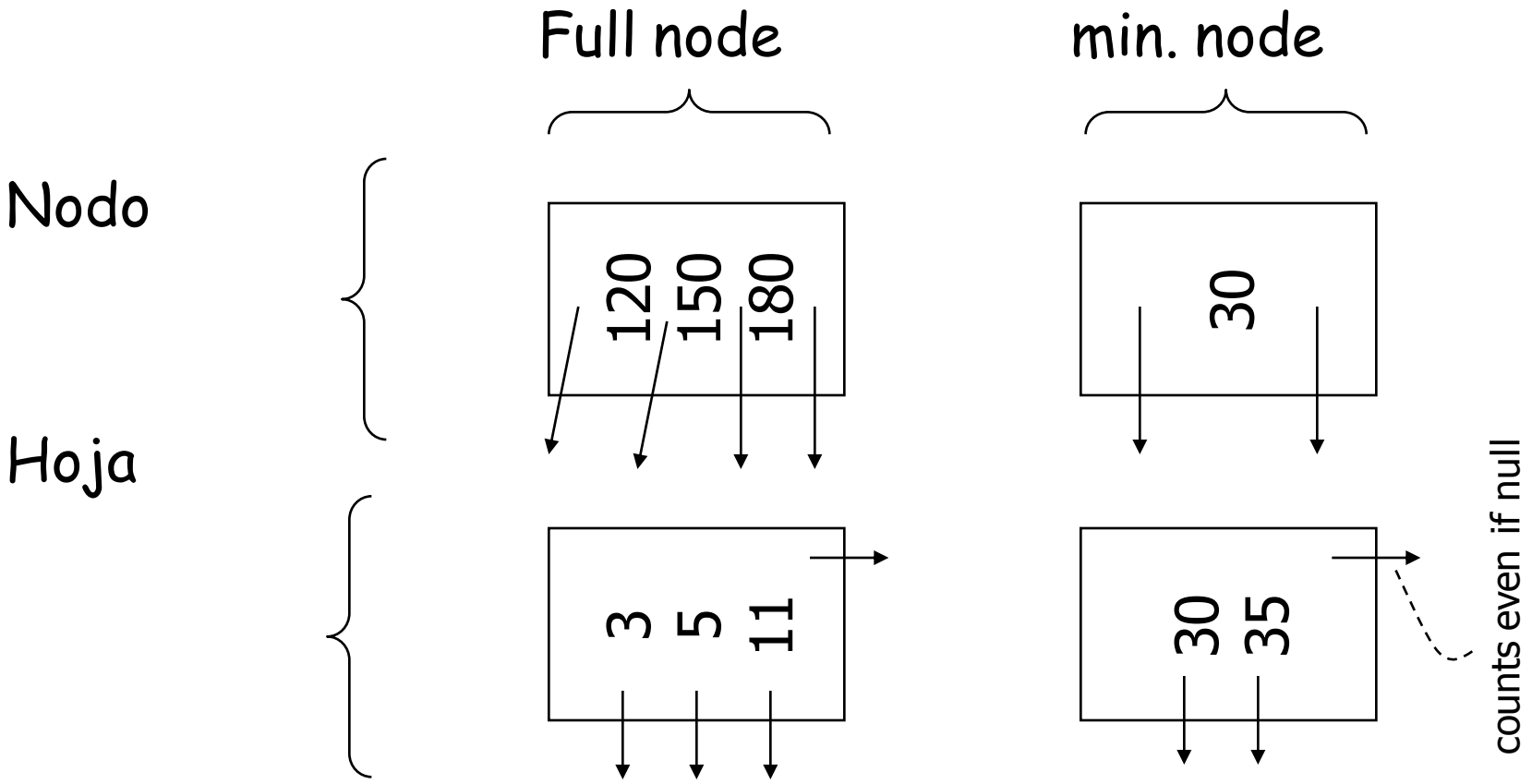
Balancenado el B*Tree

- Al menos debemos tener

  Nodos: $\lceil (n+1)/2 \rceil$ punteros

  Holas: $\lfloor (n+1)/2 \rfloor$ punteros a datos

n=3


Full node
min. node
Nodo
Hoja
120
150
180
30
3
5
11
30
35
counts even if null

B+tree rules

Árbol de orden 3

(1) Todas las hojas al mismo nivel mínimo (árbol balanceado)

(2) Las hojas apuntan a los datos que están siendo almacenados.

(3) Número de punteros/llaves:

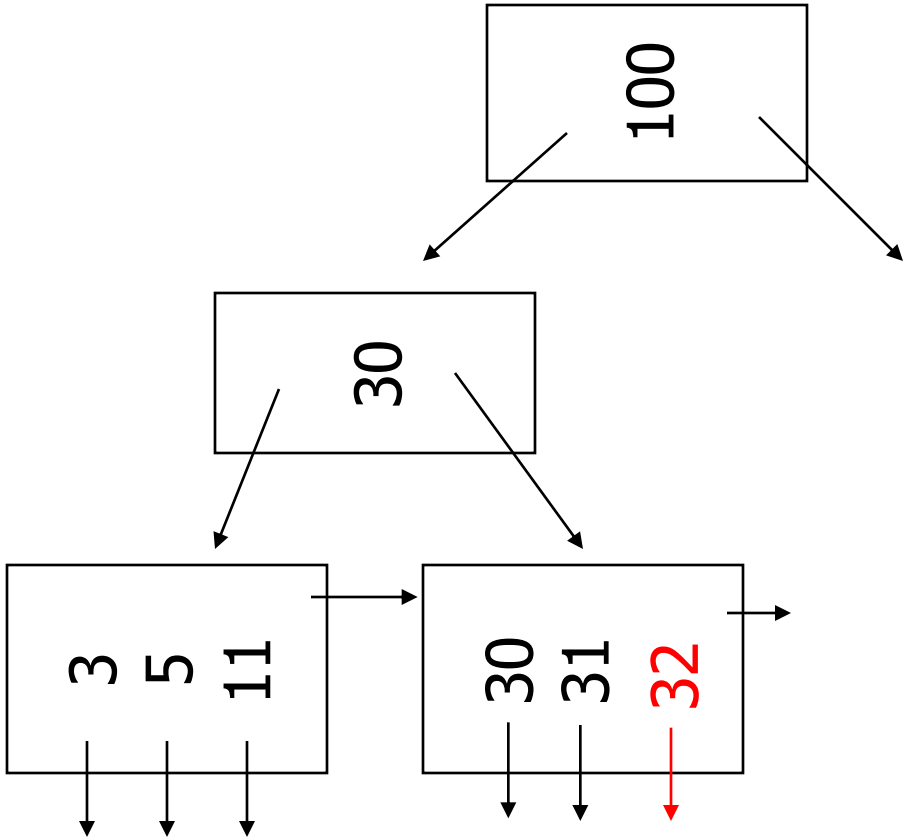
	Max ptrs	Max keys	Min ptrs→data	Min keys
Nodos	n+1	n	$\lceil (n+1)/2 \rceil$	$\lceil (n+1)/2 \rceil - 1$
Hojas	n+1	n	$\lfloor (n+1)/2 \rfloor$	$\lfloor (n+1)/2 \rfloor$
Root	n+1	n	1	1

Insertando datos en el B*Tree

(a) Caso simple (hay espacio disponible en la hoja)
(b) Hoja llena
(c) Nodo lleno
(d) Nueva raíz

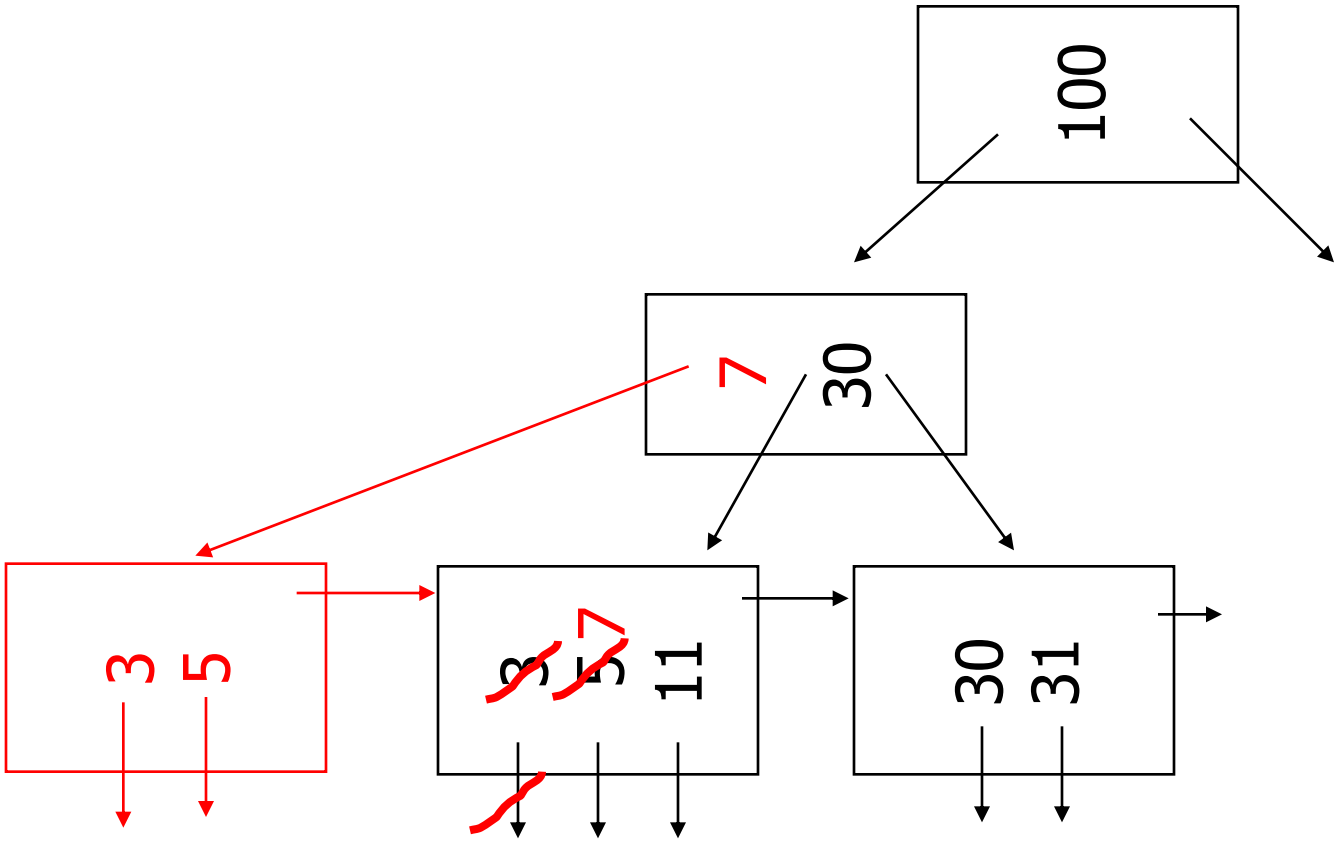
(a) Insert key = 32

n=3



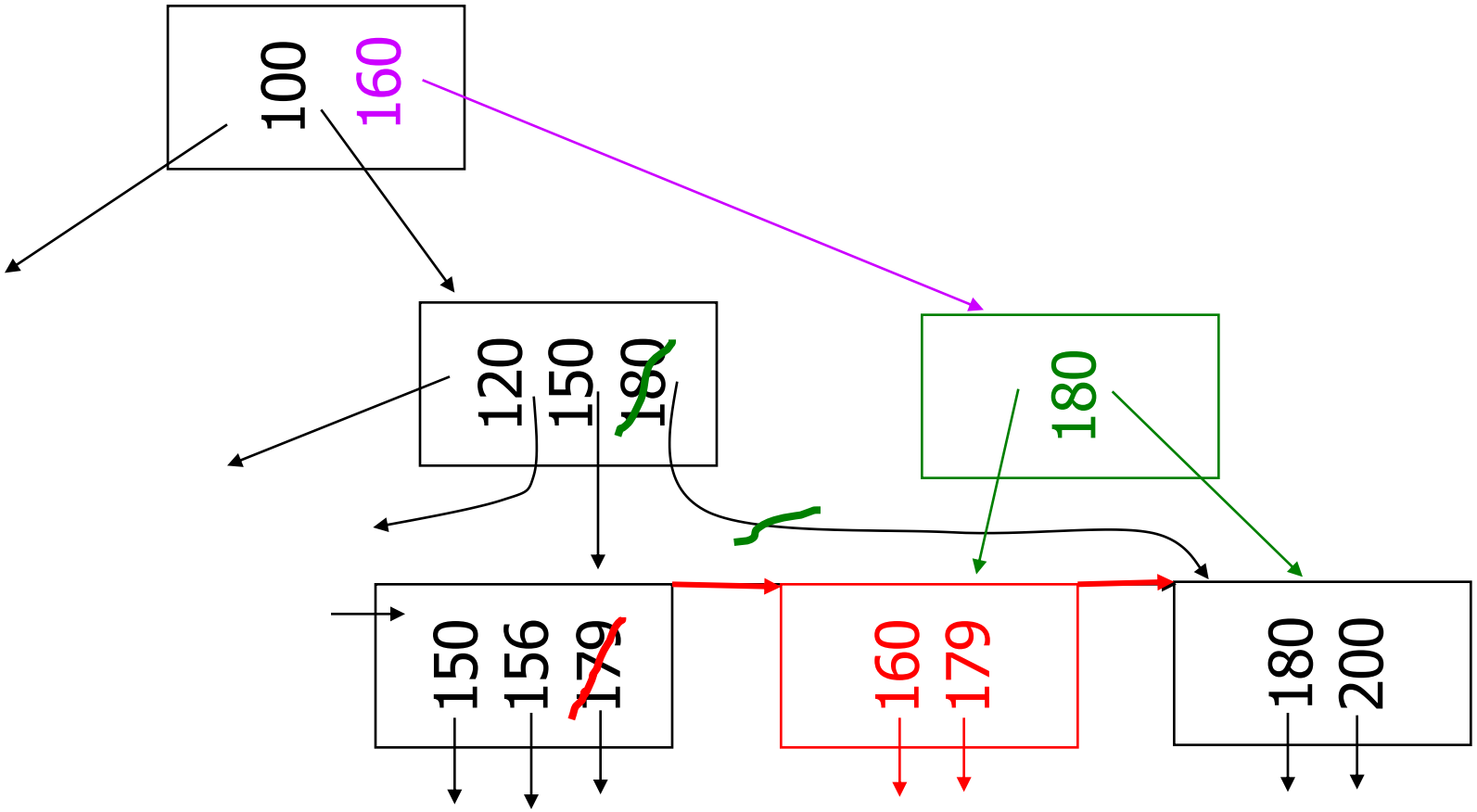
(a) Insert key = 7

n=3



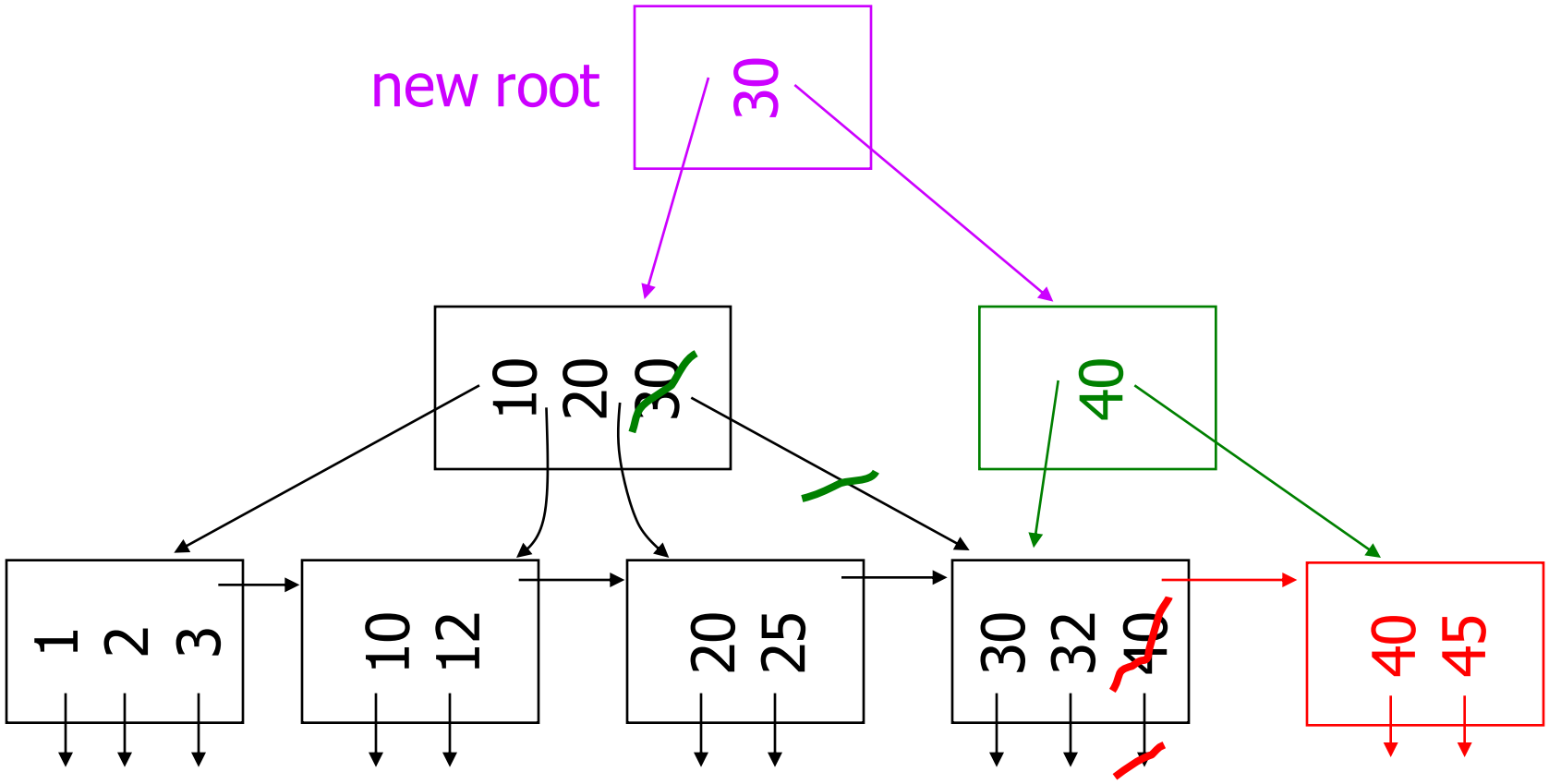
(c) Insert key = 160

n=3



(d) New root, insert 45

n=3



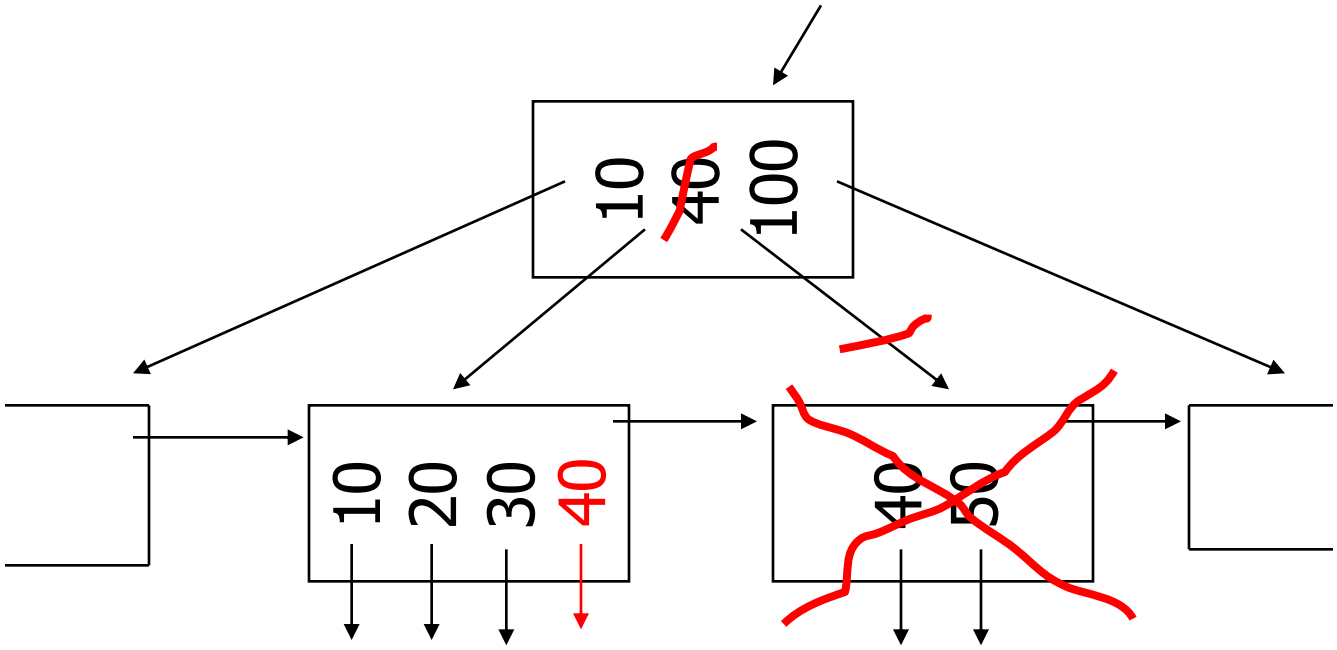
Borrando datos desde un B*Tree

(a) Caso simple (borrado simple)
(b) Unirse con un vecino
(c) Re-distribuir las llaves.
(d) Casos (b) o (c) en un nodo

(b) Unión

- Delete 50

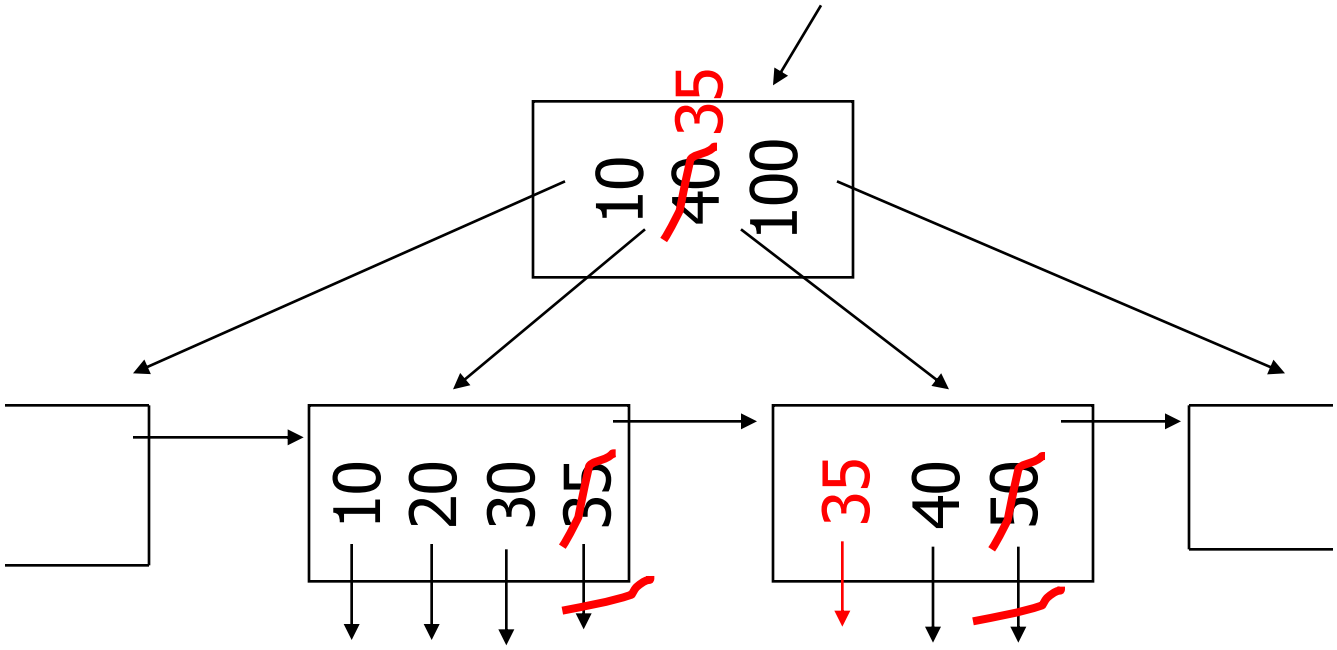
n=4



(c) Redistribución de llaves

n=4

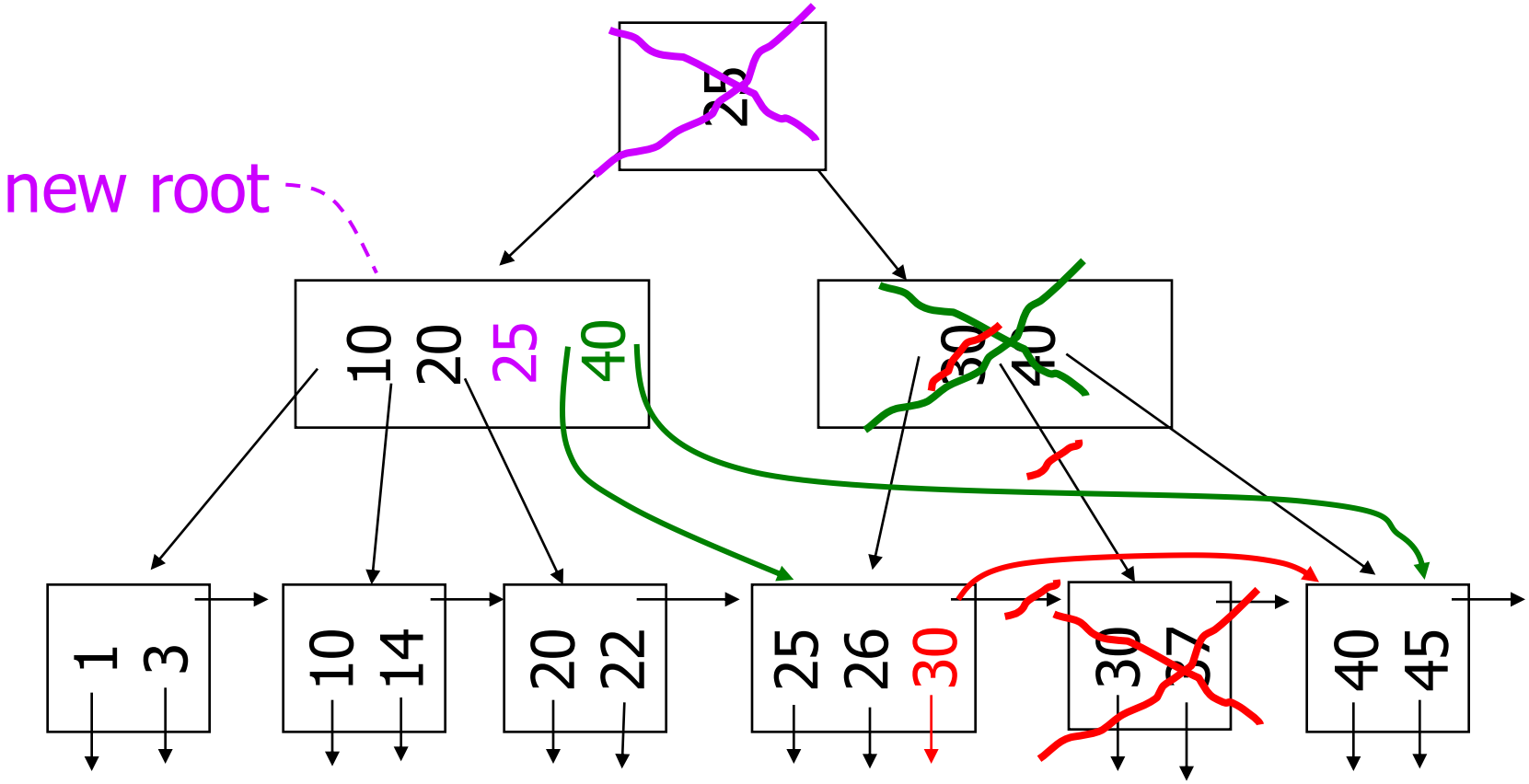
- Delete 50



(d) Union de nodos

- Delete 37

n=4



BitMap Index

- El modelo estrella presenta ciertas particularidades que podrían hacer poco eficiente el uso de B*Tree.
- Por ejemplo, cuando se repite mucho un valor en una columna que se desea indexar.
- Los BitMap Index son una forma de indexar este tipo de columnas.

Ejemplo donde usar un Bitmap Index

CUST #	MARITAL	REGION	GENDER	INCOME_ LEVEL
101	single	east	male	bracket_1
102	married	central	female	bracket_4
103	married	west	female	bracket_2
104	divorced	west	male	bracket_4
105	single	central	female	bracket_2
106	married	central	female	bracket_3

Most attributes have low cardinality

A B+-Tree would be very large on these attributes and not very efficient.

Ref: Oracle Corporation

Bitmap Index on Region

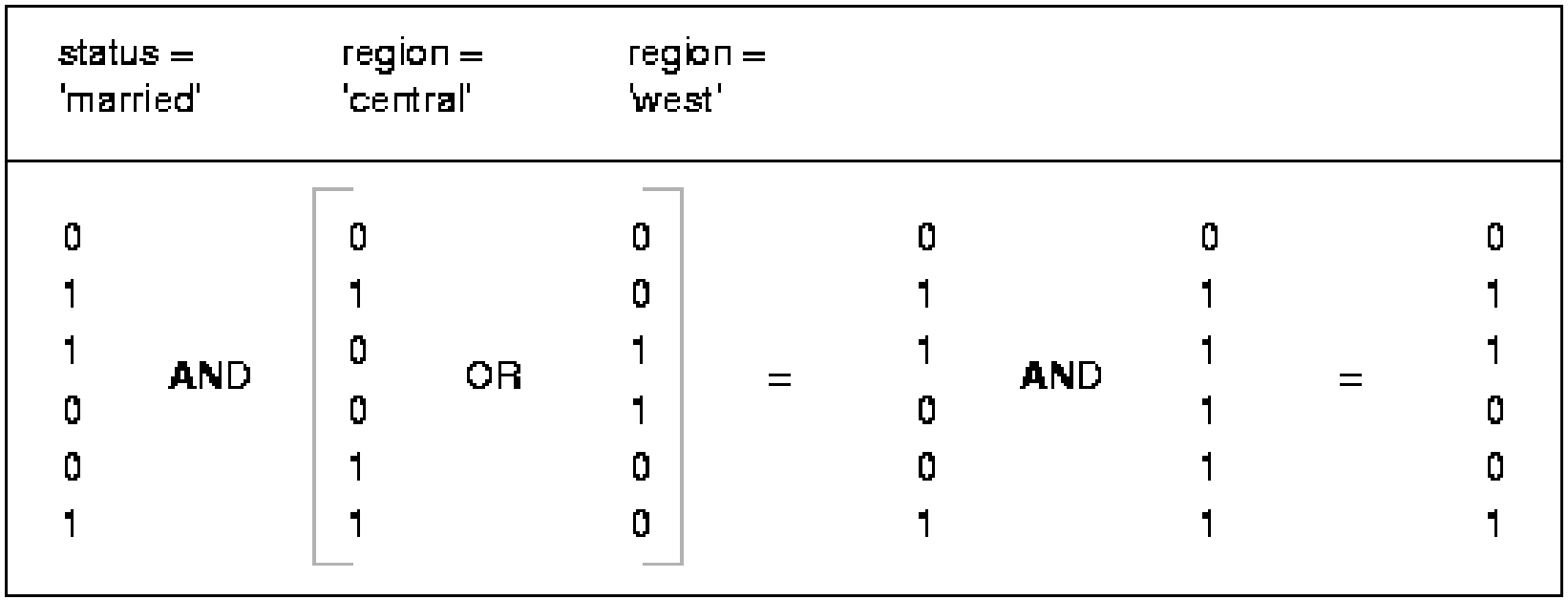
Cust#	REGION='east'	REGION='central'	REGION='west'
101	1	0	0
102	0	1	0
103	0	0	1
104	0	0	1
105	0	1	0
106	0	1	0

A bit is 1 if the tuple has the value represented by the column.

Very little space is required to store these bits.

Executing a Query

```
SELECT COUNT(*)
FROM CUSTOMER
WHERE MARITAL_STATUS = 'married'
AND REGION IN ('central','west');
```



Very efficient bit matching