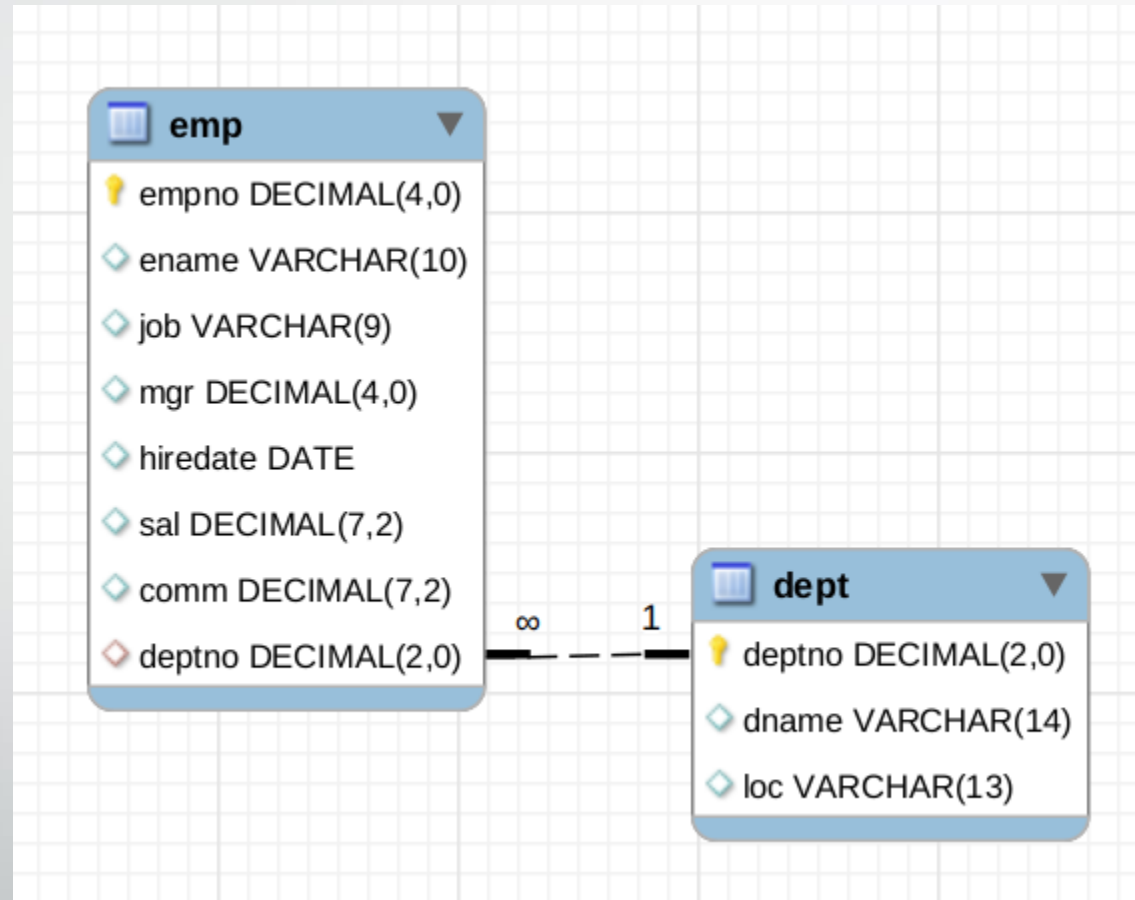




Data Definition Language

Permite definir las estructuras que almacenarán los datos

Tablas Empleado y Departamento



DESC y SHOW

- DESC nos permite ver la descripción de una tabla
- SHOW CREATE TABLE nos permite ver la sentencia de creación

```
SHOW CREATE TABLE emp;
```

```
CREATE TABLE `emp` (  
  `empno`      decimal(4,0) NOT NULL,  
  `ename`      varchar(10)  DEFAULT NULL,  
  `job`        varchar(9)   DEFAULT NULL,  
  `mgr`        decimal(4,0) DEFAULT NULL,  
  `hiredate`   date         DEFAULT NULL,  
  `sal`        decimal(7,2) DEFAULT NULL,  
  `comm`       decimal(7,2) DEFAULT NULL,  
  `deptno`     decimal(2,0) DEFAULT NULL,  
  PRIMARY KEY (`empno`),  
  KEY `deptno` (`deptno`),  
  CONSTRAINT `emp_ibfk_1` FOREIGN KEY (`deptno`) REFERENCES `dept` (`deptno`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

CREATE TABLE

Permite crear tablas

- Nombre de tabla
- Para cada columna: Tipo, Null, Valor por defecto, Incremental
- Restricciones
 - Clave primaria: Primary Key
 - Clave foránea: Foreign Key
 - Unicidad: Unique
 - Comprobaciones: Checks

CREATE TABLE

- Permite crear tablas

```
CREATE TABLE dept (  
  deptno decimal(2,0) NOT NULL,  
  aname  varchar(14) default NULL,  
  loc    varchar(13) default NULL,  
  PRIMARY KEY (deptno)  
);
```

```
CREATE TABLE emp (  
  empno  decimal(4,0) NOT NULL,  
  ename   varchar(10) default NULL,  
  job     varchar(9)  default NULL,  
  mgr     decimal(4,0) default NULL,  
  hiredate date       default NULL,  
  sal     decimal(7,2) default NULL,  
  comm    decimal(7,2) default NULL,  
  deptno  decimal(2,0) default NULL,  
  PRIMARY KEY (empno),  
  FOREIGN KEY (deptno) REFERENCES dept(deptno)  
);
```

CREATE TABLE

```
CREATE TABLE coche (  
    bastidor      varchar(17)      NOT NULL,  
    modelo        varchar(20)      default NULL,  
    marca          varchar(20)      default NULL,  
    caballos       decimal(5,0)     default NULL,  
    propietario    decimal(4,0)     default NULL,  
    PRIMARY KEY (bastidor),  
    FOREIGN KEY (propietario) REFERENCES emp(empno)  
);
```

PRIMARY KEY (bastidor),

FOREIGN KEY (propietario) REFERENCES emp(empno)

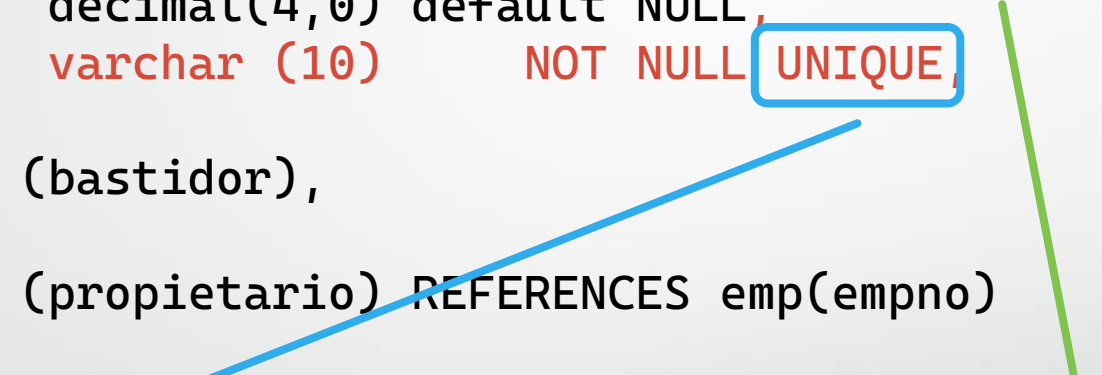
);

La **clave primaria** es un campo o a una combinación de campos que **identifica de forma única a cada fila** de una tabla

La **clave foránea** es un campo, o una combinación de campos en los que sus valores se corresponden con los de otro conjunto de atributos que es **clave candidata en otra tabla**.

Creación de tablas (CREATE TABLE)

```
CREATE TABLE coche (  
    bastidor      varchar(17)      NOT NULL,  
    modelo        varchar(20)      default NULL,  
    marca         varchar(20)      default NULL,  
    caballos       decimal(5,0)     default NULL CHECK(caballos > 0),  
    propietario   decimal(4,0)     default NULL,  
    matricula      varchar (10)      NOT NULL UNIQUE,  
  
    PRIMARY KEY (bastidor),  
  
    FOREIGN KEY (propietario) REFERENCES emp(empno)  
);
```



Restricción de unicidad: **UNIQUE**
Un conjunto de atributos no puede tener valores iguales en distintas filas.

Restricción de comprobación: **CHECK**
Permite declarar una condición que deben cumplir uno o más atributos.

Creación de tablas (CREATE TABLE)

- Crear tabla con campo autoincrementado

```
CREATE TABLE test (  
    id INT NOT NULL AUTO_INCREMENT,  
    PRIMARY KEY (id)  
);
```

- Crear tabla a partir de los resultados de una query

```
CREATE TABLE emp2 AS SELECT * FROM emp;  
CREATE TABLE dept2 AS SELECT * FROM dept;
```

No copia claves

Estas tablas las vamos a usar para más ejemplos

Modificación de tablas (ALTER TABLE)

- Añadir columnas

```
ALTER TABLE dept2  
ADD observations varchar(255),  
ADD best datetime;
```

- Modificar columnas

```
ALTER TABLE dept2  
CHANGE COLUMN best best_date date;
```

- Eliminar columnas

```
ALTER TABLE dept2  
DROP COLUMN observations;
```

Modificación de tablas (ALTER TABLE)

- Añadir clave primaria

```
ALTER TABLE dept2  
ADD PRIMARY KEY (deptno);
```

```
ALTER TABLE emp2  
ADD PRIMARY KEY (empno);
```

- Añadir clave foránea

```
ALTER TABLE emp2  
ADD FOREIGN KEY (deptno) REFERENCES dept(deptno);
```

DROP / TRUNCATE

- TRUNCATE Nos permite borrar el contenido de la tabla
 - La tabla seguirá existiendo

```
TRUNCATE TABLE dept2;
```

- DROP
 - Nos permite borrar tablas

```
DROP TABLE coche;
```



Data Modeling Language II

Consultar y manipular datos de la base de datos

Consultas involucrando a varias tablas

INSERT

Nos permite añadir datos a una tabla

```
INSERT INTO dept (deptno, dname, loc)
VALUES (50, 'MERCHANDISING', 'SAN FRANCISCO');
```

Para los campos en los que no queramos establecer un valor:

- Pasarle el valor **NULL** explícitamente
- No especificar esa columna en el **INSERT**

```
INSERT INTO dept (deptno, dname)
VALUES (61, NULL);
```

INSERT

También es opcional especificar el orden de los campos

```
INSERT INTO dept  
VALUES (62, 'MERCHANDISING', 'SAN FRANCISCO');
```

Para ello necesitamos saber en qué orden enviarlos

```
DESC dept;
```

#	Field	Type	Null	Key
1	deptno	decimal(2,0)	NO	PRI
2	dname	varchar(14)	YES	
3	loc	varchar(13)	YES	

UPDATE

Nos permite modificar datos de una tabla

Los cambios se aplican a todas las filas recuperadas.

Si no indicamos WHERE se modificarán todas las filas de la tabla

Actualizar un dato

```
UPDATE dept
SET  dname = 'ADVERTISEMENT',
      loc  = 'CALIFORNIA'
WHERE deptno = 61;
```

UPDATE

1. Modifica el valor de la columna sal en todas las filas de la tabla emp.

```
UPDATE emp  
SET sal = sal + 100;
```

2. Modifica el valor de varias columnas para las filas que cumplen la condición.

```
UPDATE emp  
SET comm = 110, deptno = 10  
WHERE comm IS NULL;
```


DELETE

Nos permite eliminar datos de una tabla

```
DELETE FROM <table>  
WHERE condition
```

Los cambios se aplican a todas las filas recuperadas.

Si no indicamos WHERE se modificarán todas las filas de la tabla

Eliminar un dato

```
DELETE FROM dept,  
WHERE deptno = 61;
```

DELETE

- Borrar las filas que cumplan la condición seleccionada
 - `DELETE FROM emp`
`WHERE ename LIKE '%A'`
- Eliminar todas las filas de la tabla dept
 - `DELETE FROM dept`



Data Modeling Language III

Obtener datos relacionados

Tablas Empleado y Departamento

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	17/12/1980	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	20/02/1981	1600.00	300.00	30
7521	WARD	SALESMAN	7698	22/02/1981	1250.00	500.00	30
7566	JONES	MANAGER	7839	02/04/1981	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	28/09/1981	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	01/05/1981	2850.00	NULL	30
7782	CLARK	MANAGER	7839	09/06/1981	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	09/12/1982	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	17/11/1981	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	08/09/1981	1500.00	0.00	30
7876	ADAMS	CLERK	7788	12/01/1983	1100.00	NULL	20
7900	JAMES	CLERK	7698	03/12/1981	950.00	NULL	30
7902	FORD	ANALYST	7566	03/12/1981	3000.00	NULL	20
7934	MILLER	CLERK	7782	23/01/1982	1300.00	NULL	10

bastidor	propietario	marca
10	ACCOUNTING	NEWYORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

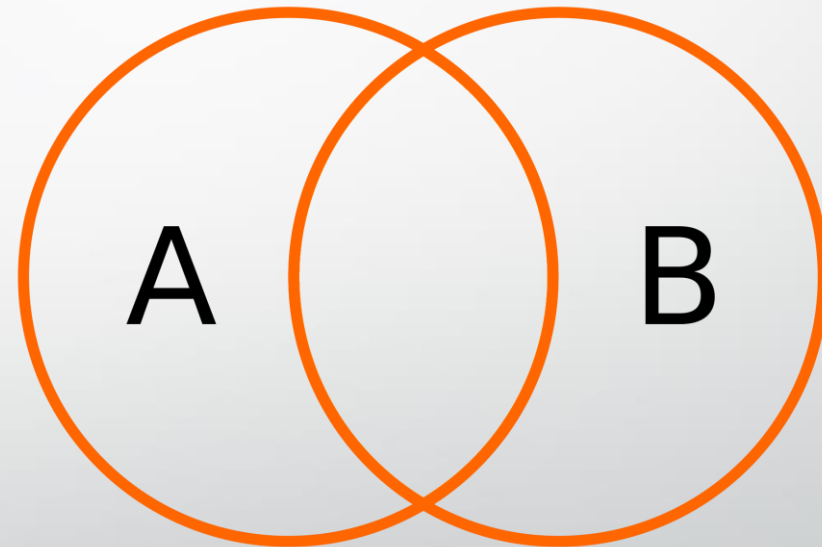
deptno	dname	loc
10	ACCOUNTING	NEWYORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

JOIN

Permite obtener datos a partir de tablas relacionadas entre sí aplicando las condiciones indicadas.

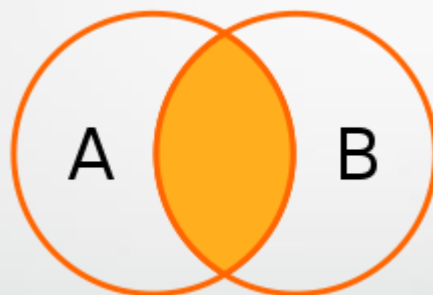
Varios tipos

- **INNER JOIN**
- **LEFT JOIN**
- **RIGHT JOIN**
- **FULL OUTER JOIN**



INNER JOIN

- Obtiene las entidades que tienen valores (no nulos) en las tablas
- Es el tipo de JOIN más utilizado, por lo que es considerado el tipo de combinación predeterminado.



```
SELECT *  
FROM emp e  
INNER JOIN dept d ON d.deptno = e.deptno;
```

INNER JOIN

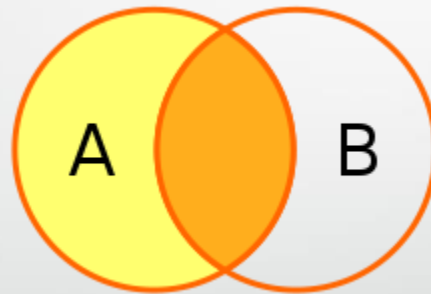
- Seleccionar los datos de los empleados y los departamentos relacionados

```
SELECT e.*, d.*  
FROM emp e INNER JOIN dept d ON e.deptno = d.deptno;
```

#	empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	dname	loc
1	7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10	10	ACCOUNTING	NEW YORK
2	7839	KING	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10	10	ACCOUNTING	NEW YORK
3	7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10	10	ACCOUNTING	NEW YORK
4	7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20	20	RESEARCH	DALLAS
5	7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20	20	RESEARCH	DALLAS
6	7788	SCOTT	ANALYST	7566	1982-12-09	3000.00	NULL	20	20	RESEARCH	DALLAS
7	7876	ADAMS	CLERK	7788	1983-01-12	1100.00	NULL	20	20	RESEARCH	DALLAS
8	7902	FORD	ANALYST	7566	1981-12-03	3000.00	NULL	20	20	RESEARCH	DALLAS
9	7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	30	SALES	CHICAGO
10	7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	30	SALES	CHICAGO
11	7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
12	7698	BLAKE	MANAGER	7839	1981-05-01	2850.00	NULL	30	30	SALES	CHICAGO
13	7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	30	SALES	CHICAGO
14	7900	JAMES	CLERK	7698	1981-12-03	950.00	NULL	30	30	SALES	CHICAGO

LEFT JOIN

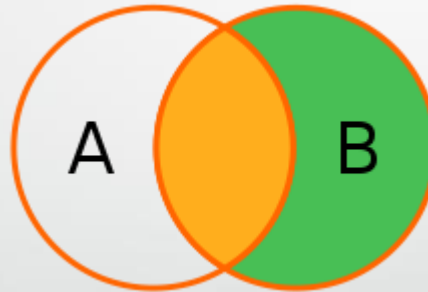
- **LEFT JOIN** devuelve todos los valores de la tabla izquierda relacionados con los valores de la tabla de la derecha correspondientes, si los hay. En caso que no haya relación devuelve un valor nulo en los campos de la tabla de la derecha.



```
SELECT *  
FROM emp e  
LEFT JOIN dept d ON d.deptno = e.deptno;
```


RIGHT JOIN

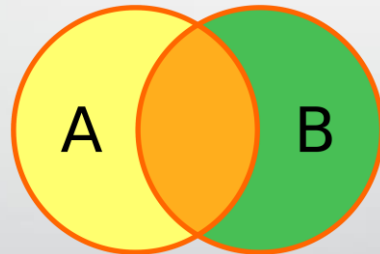
- **RIGHT JOIN** devuelve todos los valores de la tabla derecha relacionados con los valores de la tabla de la izquierda correspondientes, si los hay. En caso de que no haya relación devuelve un valor nulo en los campos de la tabla de la izquierda.



```
SELECT *  
FROM emp e  
LEFT JOIN dept d ON d.deptno = e.deptno;
```

FULL OUTER JOIN

- Esta operación presenta los resultados de tabla izquierda y tabla derecha aunque alguna no tengan correspondencia en la otra tabla. La tabla combinada contendrá, entonces, todos los registros de ambas tablas y presentará valores nulos NULLs para registros sin pareja.



FULL OUTER JOIN

- MySQL no soporta esta operación de forma nativa, por lo que hay que unir los resultados que producen las consultas LEFT JOIN y RIGHT JOIN.

```
SELECT *  
FROM emp e  
LEFT JOIN dept d ON d.deptno = e.deptno;  
UNION  
SELECT *  
FROM emp e  
RIGHT JOIN dept d ON d.deptno = e.deptno;
```



Data Modeling Language IV

Agrupando resultados

SELECT

El orden de ejecución de las cláusulas es

- FROM (obligatoria)
- WHERE (opcional)
- GROUP BY (opcional)
- HAVING (opcional)
- SELECT (obligatoria)
- ORDER BY (opcional)
 - ASC.: orden ascendente (por defecto)
 - DESC: orden descendente

GROUP BY

- Nos permite agrupar los resultados filtrados por una función de agrupamiento.

```
SELECT COUNT(*)  
FROM emp  
WHERE ename LIKE '%A%'  
GROUP BY deptno;
```

- Todas las columnas deben devolver datos únicos por grupo

GROUP BY

- Todas las columnas deben devolver datos únicos por grupo

```
SELECT MAX(sal), deptno  
FROM emp  
GROUP BY deptno;
```

- ¿Qué pasaría en este caso?

```
SELECT MAX(sal), deptno, ename  
FROM emp  
GROUP BY deptno;
```

HAVING

- Nos permite aplicar condiciones sobre los grupos

```
SELECT MAX(sal), MIN(sal), COUNT(empno), deptno  
FROM emp  
GROUP BY deptno  
HAVING MIN(sal) > 1000;
```

- Todas las condiciones se deben poder aplicar sobre cada grupo