UF1. NF1. INTRODUCCIÓ A LES BASES DE DADES

1. INTRODUCCIÓ

En el món actual existeix una cada vegada major demanda de dades. Aquesta demanda sempre ha estat palesa en empreses i societats, però en aquests últims anys la demanda s'ha disparat més a causa de l'accés multitudinari a les xarxes com Internet.

Abans de l'aparició de les aplicacions informàtiques, les empreses tenien com a úniques eines de gestió de dades: els calaixos, carpetes i arxius en les quals s'emmagatzemaven les dades. En aquest procés manual, el temps requerit per manipular aquestes dades era enorme. No obstant això, el procés d'aprenentatge era senzill, ja que s'usaven elements molt familiars per a l'usuari.

Per aquesta raó, la informàtica ha adaptat les seves eines per a que els elements que l'usuari utilitza a l'ordinador s'assemblin als quals utilitzava manualment. Així, en informàtica, se segueix parlant d'arxius (fitxers), formularis, carpetes (directoris), etc.

Ara començarem per veure els conceptes bàsics de les bases de dades, relacionant-los amb els sistemes de gestió d'arxius, per després començar a parlar dels sistemes gestors de bases de dades i els seus tipus.

2. DADES I INFORMACIÓ

En primer lloc, anem a establir una diferència entre dos termes bàsics que se solen utilitzar amb bastant freqüència en entorns de bases de dades i sistemes d'informació. Es tracta de la diferència entre dades i informació.

Al parlar de dades ens referim a fets aïllats, com per exemple, Joan té un compte corrent a CaixaBank. En canvi, al parlar d'informació, ens estem referint a donar-los processats, organitzats o resumits. Per exemple, la resposta a la pregunta *Quin és el saldo dels comptes corrents de Joan?* seria informació.

Les dades solen ser magnituds numèriques o captades, però també poden ser noms o conjunts de símbols; o valors qualitatius; o frases senceres; o imatges, sons, colors, olors. Les dades no són informació més que en un sentit ampli d'informació de partida o inicial, però les dades per si mateixes no permeten l'adopció de la decisió més convenient perquè no aporten els coneixements necessaris.

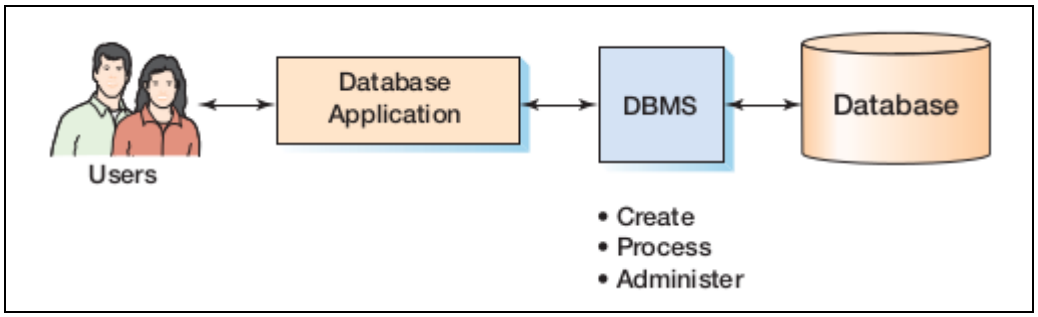
Només una elaboració adequada de les dades (un procés de les dades) ens proporcionarà el coneixement desitjat. La informació és, doncs, el resultat d'aquesta transformació, d'aquest PROCÉS DE LES DADES.

Per tant, en el món empresarial, la informació és un recurs vital, pel que les empreses sempre han fet ús dels mitjans que estaven al seu abast per al processament de dades i la gestió d'informació. Entre aquests mitjans es troben els Sistemes d'Informació, que organitzen les dades per produir informació.

El desenvolupament dels Sistemes de Bases de dades s'ha convertit en crucial per als Sistemes d'Informació, ja que són el seu nucli: són els que li proporcionen les dades que necessiten per a l'elaboració de la informació.

3. SISTEMA DE BASE DE DADES

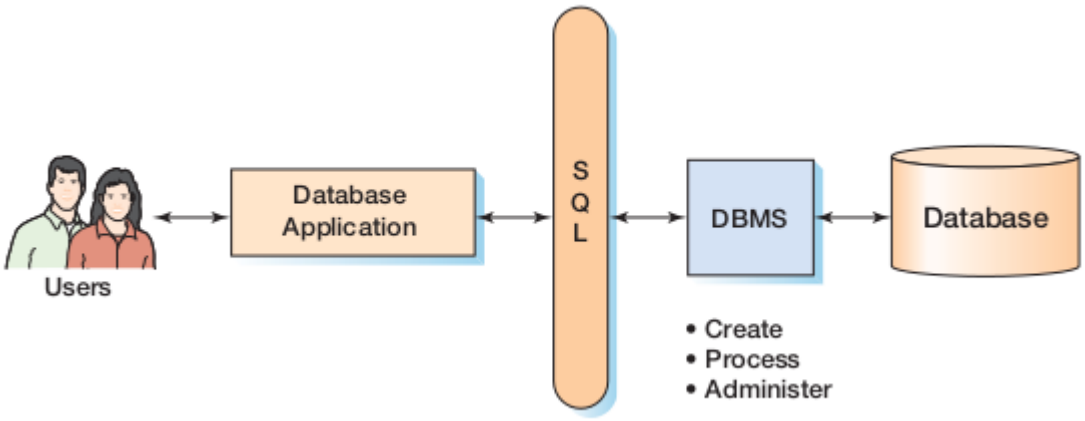
3.1. Components



Típicament consta de quatre components:

- **Usuaris**: utilitzen la base de dades per mantenir un seguiment de les seves coses. Utilitzen formularis per a llegir, entrar i consultar les dades i ells produeixen informes amb la conseqüent informació.
- **Aplicació d'accés a la BD**: conjunt d'un o més programes que serveixen com a intermediari entre l'usuari i el SGBD. Llegeixen o modifiquen la base de dades mitjançant l'enviament de peticions SQL al SGBD. També presenten a l'usuari els formularis i informes. Aquests programes poden ser adquirits a companyies de *software*, però normalment son escrits manualment.
- **Sistema Gestor de la Base de Dades (SGBD, en anglès DBMS, DataBase Management System)**: conjunt de programes i procediments utilitzat per crear, processar i administrar la base de dades. Rep les peticions codificades en SQL i les tradueix en accions sobre la base de dades. A causa de la seva gran complexitat, les empreses mai desenvolupen el seu propi SGBD, sinó que adquireixen les llicències per fer ús d'un existent (Veure Apartat 3.3.).
- **Base de dades**: col·lecció de dades emmagatzemades seguint algun tipus d'estructura. Ex: en SGBD relacionals, les dades s'emmagatzemen com de taules i relacions entre elles (Veure Apartat 3.2).

Afegir, que donada la importància de **SQL** (*Structured Query Language*), reconegut com a estàndard internacional comprensible per molts dels SGBD en el processament de la base de dades i el fet de que les aplicacions d'accés típicament envien sentències SQL al SGBD per al seu processament, es pot considerar com un cinqué component del sistema (Veure Apartat 3.4).



3.2. Base de dades

Una Base de Dades (BD) és una col·lecció o dipòsit de dades integrades, emmagatzemades en un suport secundari (no volàtil) i amb redundància controlada. Les dades, que han de ser compartides per diferents usuaris i aplicacions, han de mantenir-se independents d'ells i la seva definició (estructura de la BD), única i emmagatzemada juntament amb les dades, s'ha de recolzar en un model de dades, el qual ha de permetre captar les interrelacions i restriccions existents en el món real. "Els procediments d'actualització i recuperació, comuns i ben determinats, facilitaran la seguretat del conjunt de les dades", De Miguel et al. (1999).

Vegem en què consisteix cadascun dels aspectes esmentats en aquesta definició de Base de dades.

La BD és un conjunt de dades relatives a una determinada parcel·la del món real (per exemple, una biblioteca, una empresa petroquímica, una universitat, etc.,) que s'emmagatzemen en un suport informàtic no volàtil (és a dir, dispositius de memòria secundària com discos, cintes, etc.) que fan que les dades no desapareguin, "quan no s'estan usant".

A més, no ha d'existir redundància, és a dir, no han d'existir duplicitats perjudicials ni innecessàries (si pot ser un determinat tipus de dada, per exemple, les dades d'un client d'una empresa, només han d'aparèixer en un lloc en la BD). En ocasions, és necessària certa redundància (a nivell d'emmagatzematge físic, ara com ara direm que el nivell físic concerneix a com s'emmagatzemen les dades en els fitxers de la BD) que millora l'eficiència de la BD, per exemple, davant determinats tipus de consultes de dades. No obstant això, aquesta redundància sempre ha de ser controlada pel sistema perquè no es produeixin inconsistències; pensa què succeiria si les dades dels clients d'una empresa es repeteixen en diverses parts de la BD i no es controlés: pot ocórrer que si un client canvia d'adreça postal i només s'actualitza aquesta informació en un dels llocs, llavors la BD quedaria en estat inconsistent (el client apareix amb dades diferents, en diferents parts de la BD).

D'altra banda, les BD han d'atendre a múltiples usuaris de l'organització (informàtics que desenvolupen programes d'accés a la BD, administratius, usuaris de les aplicacions, etc.) així com a diferents aplicacions (per exemple, aplicacions de comptabilitat, de facturació, etc.), totes elles accedint a les dades contingudes en la BD de l'empresa.

3.2.1. Metadades d'una base de dades

Una base de dades és autodescriptiva, en el sentit que conté una descripció d'ella mateixa. Per tant, les bases de dades també emmagatzemen descripcions de les dades que es guarden.

Aquestes dades de descripció s'anomenen **metadades** (dades sobre les dades). La forma i format de les metadades varia segons el SGBD utilitzat.

Les metadades es poden consultar per determinar les taules existents particulars, els índexs o altres estructures existents en la base de dades.

3.2.2. Nivells d'abstracció en una base de dades

En qualsevol sistema d'informació (tant fitxers com bases de dades) es considera que es poden observar les dades des de dos punts vista:

- **Vista externa**. Aquesta és la visió de les dades que posseeixen els usuaris externs.
- **Vista física**. Aquesta és la forma en la qual realment estan emmagatzemats les dades.

En un sistema d'arxius, els usuaris veuen les dades des de les aplicacions creades pels programadors. Aquesta vista poden ser formularis, informes visuals o en paper,... però la realitat física d'aquestes dades, tal qual s'emmagatzemen en els discos, no la veuen. Aquesta visió està reservada als administradors.

En el cas dels sistemes de base de dades i segons el model ANSI, s'afegeix una tercera vista, que és la vista conceptual. Aquesta vista se situa entre la física i l'externa. Es parla doncs en bases de dades de la utilització de tres esquemes o models per representar les dades, entenent per model les normes que permeten crear esquemes (dissenys de la base de dades).

Esquema físic

Es correspon amb la vista del suport físic informàtic, en quant a que es refereix a la forma que s'organitzen les dades en l'emmagatzematge físic (índexs o punters, longitud dels camps, camins d'accés a les dades, particions de memòria, etc.). Aquesta visió la requereix només l'administrador.

Esquema conceptual

Es correspon amb la visió total de les dades a nivell lògic. És a dir, totes les dades i les seves relacions. Aquesta vista global s'interposa entre el nivell extern i el nivell físic sent independent tant de l'equip com de cada usuari en particular.

Es tracta de la proposta teòrica de la base de dades i, és per tant, el primer pas per crear una nova de base de dades. Consisteix en el disseny d'un model per aquest nivell.

Esquema extern

Es correspon amb la visió de la BD que cada usuari extern té en particular. A aquesta vista individual que cada usuari té de la BD se la denomina vista externa. Això significa que no tots els usuaris necessiten conèixer la BD completa sinó que únicament necessiten una vista parcial d'ella (la qual li permeti portar a terme el seu treball); per exemple, un administratiu que treballi elaborant les nòmines dels empleats d'una empresa no necessita conèixer les dades relatives a les vendes de productes d'aquesta empresa.

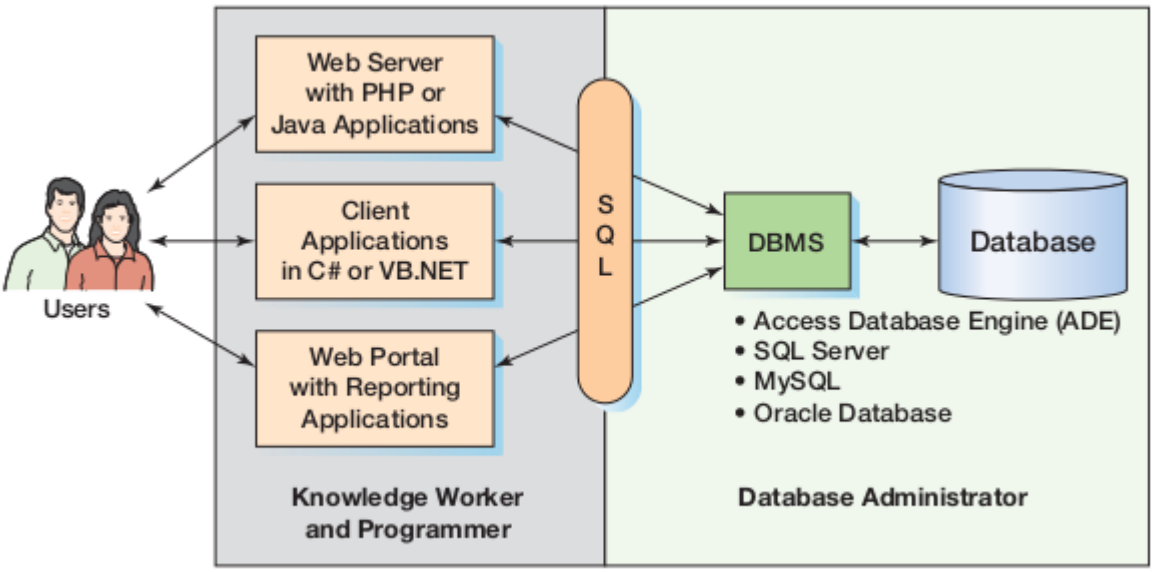
3.3. SGDB (en anglès, DBMS)

Un Sistema Gestor de Bases de Dades (SGBD) és un conjunt coordinat de programes, procediments, llenguatges, eines, etc., que subministra, tant als usuaris no informàtics com als analistes, programadors o administradors d'una base de dades, els mitjans necessaris per crear, processar i administrar les dades emmagatzemades en una base de dades, garantint la seva seguretat.

L'èxit del SGBD resideix a mantenir la seguretat i integritat de les dades.

Exemples de SGBD actuals són: Oracle Database, Oracle MySQL, DB2 d'IBM, SQL Server de Microsoft, PostgreSQL, Microsoft Access, MariaDB, ...

La figura següent mostra l'arquitectura del sistema d'una BD; en ella s'observa com l'únic usuari que treballarà directament amb el SGBD és l'administrador. Els programadors d'aplicacions d'accés sempre faran ús d'SQL com a llenguatge intermedi de peticions de dades al SGBD i així accedir a la base de dades que conté tota la informació.



3.3.1. Objectius dels SGBD

Un SGBD ha de oferir els següents serveis:

- **Creació i definició de la BD**. Especificació de l'estructura, el tipus de dades, les restriccions i relacions entre ells mitjançant llenguatges de definició de dades. Tota aquesta informació es corresponen amb les taules i metadades de la base de dades i el SGBD proporcionarà mecanismes per a la seva gestió.
- **Manipulació de les dades** realitzant consultes, insercions, actualitzacions o esborrats.
- **Accés controlat a les dades de la BD** mitjançant mecanismes de seguretat d'accés als usuaris.
- **Redundància mínima de les dades** per controlar l'espai físic ocupat per la BD.
- **Integritat i consistència de les dades** utilitzant mecanismes per evitar que les dades siguin perjudicades per canvis no autoritzats.
- **Accés compartit a la BD**, controlant la interacció entre usuaris concurrents.
- **Mecanismes de recolzament, recuperació i còpies de seguretat** per restablir la informació en cas de fallades en el sistema, així com **l'exportació i importació de dades**.

3.3.2. Evolució dels SGBD

Era	Anys	Productes més destacats	Descripció
PreDB	Abans de 1970	Sistemes de fitxers	Totes les dades son emmagatzemades en arxius separats. La integració de dades és molt complexa. L'espai d'emmagatzematge és car i limitat.
Early DB	1970-1980	ADABAS, System2000, Total, IDMS, IMS	Són els primers productes que ofereixen taules relacionades. CODASYL DBTG i models de dades jeràrquics (DL/I) prevalen.
Aparició del model relacional	1978-1985	DB2, Oracle	Els primers SGBD relacionals essencialment van triomfar. Amb el temps, els avantatges han anat perden pes.
SGBD per microcomputadors	1982-1992+	Access, dBase-II, R:base, Paradox	Sorprenent! Un SGBD en un microcomputador. A partir de 1990 el mercat ha sigut exclusiu de Microsoft Access.
SGBD orientats a objectes	1985-2000	Oracle ODBMS i altres	Mai aplicat. Requereix convertir la base de dades relacional. Massa feina per poder percebre el resultat.
Web databases	1995 - present	IIS (Internet Information Server de Microsoft), Apache, PHP, ASP.NET, Java	La característica "stateless" del protocol HTTP va ser un problema inicialment. Les primeres aplicacions van ser simples transaccions d'un estat. Posteriorment, una lògica més complexa s'ha desenvolupat.
Open source DMBS	1995 - present	MySQL, PostgreSQL, MariaDB i altres	Els SGBD *open source* també proporcionen la funcionalitat i característiques que tenen els SGBD comercials amb un cost reduït.
XML i Web services	1998 - present	XML, SOAP, WSDL, UDDI i altres estàndards	XML proporciona grans beneficis a les aplicacions web amb bases de dades. Potser algunes substitueixen a les bases de dades relacionals.
NoSQL Movement	2009 - present	Apache Cassandra, MongoDB, Redis, CouchDB, dbXML, MonetDB/xQuery i altres	El moviment NoSQL és realment un moviment NoRelationalDB que substitueix les bases de dades relacionals per estructures de dades no relacionals. L'aproximació NoSQL, que és utilitzada per Facebook i Twitter, sovint es basa en XML.

3.4. SQL

SQL, *Structured Query Language*, és un llenguatge estandarditzat per la ISO (*International Organization for Standardization*), és a dir, que tots els SGBD que el suporten han de tenir la mateixa sintaxis a l'hora d'aplicar el llenguatge. Es divideix en 4 subllenguatges, el total de tots permet al SGBD complir amb les seves funcionalitats:

- **Llenguatge DDL**: llenguatge de definició de dades (*Data Definition Language*). Permet crear tota l'estructura d'una base de dades (des de taules fins a usuaris o rols), les relacions entre les dades i les regles que han de complir. Les seves clàusules són de tipus DROP (esborrar), ALTER (modificar) i CREATE (crear).
- **Llenguatge DML**: llenguatge de manipulació de dades (*Data Manipulation Language*). Aquest llenguatge permet: SELECT (seleccionar/buscar), INSERT (inserir), UPDATE (actualitzar) i DELETE (esborrar). Si la funció es redueix només a buscar dades, utilitza el terme **DQL** (*Data Query Language*).
- **Llenguatge DCL:** llenguatge de control de dades (*Data Control Language*). Inclou comandes com GRANT i REVOKE que permeten a l'administrador gestionar l'accés a les dades contingudes en la base de dades.
- **Llenguatge TCL**: llenguatge de control de transaccions (*Transaction Control Language*). El propòsit d'aquest llenguatge és permetre executar varies comandes de forma simultània com si fossin una comanda atòmica o indivisible. Si es possible, s'aplica la transacció (COMMIT), i si, en algun pas de l'execució passa quelcom inesperat, es poden desfer tots els passos realitzats (ROLLBACK).

4. DISSENY D'UNA BD

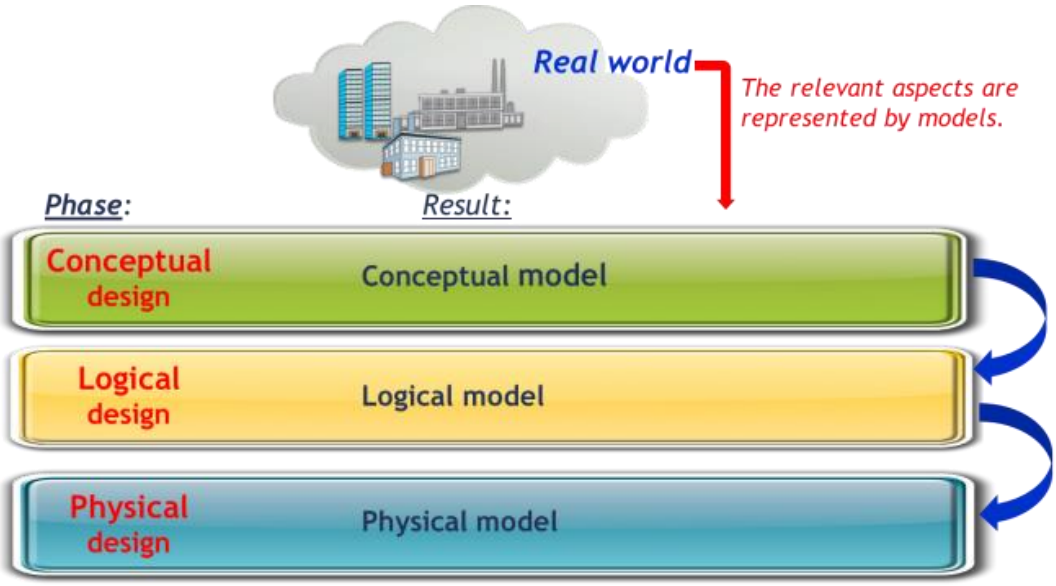
4.1. Etapes en el disseny d'una BD

El primer pas que es realitza és la recopilació d'informació i anàlisi de requeriments, durant el qual, els dissenyadors entrevisten als futurs usuaris del Sistema de Base de dades per a comprendre i redactar els requeriments d'informació. El resultat d'aquest pas serà un conjunt de requeriments redactat de forma concisa.

Una vegada que s'ha realitzat l'anàlisi de requeriments, el pas següent és crear el **disseny (o esquema) conceptual** per a la base de dades mitjançant un model de dades conceptual d'alt nivell. L'esquema conceptual és una descripció concisa dels requeriments d'informació dels usuaris, i conté informació sobre les relacions i restriccions. Aquest esquema conceptual pot servir com punt de referència per a assegurar-se que se satisfan tots els requeriments dels usuaris. Representem el món real en esquemes conceptuals E/R. L'esquema E/R ve a ser per a una BD com els plànols d'una casa per a un arquitecte.

Una vegada que hagi acabat el procés de disseny cal començar a implementar el SBD amb un SGBD comercial. La majoria dels SGBD actuals utilitzen un model de dades d'implementació, pel que cal traduir el model de dades d'alt nivell al model de dades d'implementació, el que es coneix com el procés de **disseny lògic** de la base de dades o transformació de models de dades. El resultat és un esquema de la base de dades especificat en el model de dades d'implementació del SGBD. Nosaltres transformarem els esquemes conceptuals E/R a esquemes relacionals.

L'últim pas és la fase de **disseny físic** de la base de dades, durant la qual s'especifiquen les estructures d'emmagatzematge internes i l'organització dels arxius de la base de dades. A més, s'han de dissenyar i implementar els programes d'aplicació per a les transaccions que es van recopilar en l'anàlisi de requeriments funcionals.



Procés de disseny d'una Bases de Dades

4.2. Què és el model ER?

El model ER (o E/R o E-R) és un **model de dades conceptual d'alt nivell** i que se sol usar bastant en el disseny de bases de dades, amb independència de la màquina en la qual s'implementin. Va ser **proposat per Peter Chen en 1976**. Des de llavors molts autors s'han interessat per ell, estudiant-lo i ampliant-lo, aconseguint així diverses variants del model (diferents formes de representació dels objectes), però totes elles parteixen del mateix concepte: el coneixement del món real que es desitja representar a través d'una anàlisi dels requisits o especificacions del problema.

El model ER consisteix en un conjunt de conceptes, regles i notacions que permeten formalitzar la semàntica del món real que es pretén modelar (també denominada Univers del Discurs) en una representació gràfica o diagrama que denominem esquema de la Base de Dades.

4.3. Versions

Almenys 3 versions diferents del model ER són utilitzats actualment.

Un d'ells, el model Information Engineering (IE), va ser desenvolupat per James Martin en 1990. Aquest model utilitza "pota de corb" per indicar la multiplicitat d'una relació, i per tant, es diu **IE Crow's Foot model**.

El 1993, el National Institute of Standards and Technology va proposar altra versió del model ER model com estàndard. Aquesta versió es va dir **Integrated Definition 1, Extended (IDEF1X)**. Aquest estàndard incorpora les idees bàsiques del model clàssic ER, però utilitza símbols gràfics diferents. Encara que aquest model és un estàndard nacional i utilitzat al govern dels EEUU, és difícil d'entendre i utilitzar.

Al mateix temps, per afegir més complexitat, la metodologia orientada a objectes **Unified Modeling Language (UML)** va adoptar el model ER, però va introduir els seus propis símbols per donar una nova perspectiva a la programació orientada a objectes.

L'existència de diferències a causa de diferents versions del model ER ha causat confusió i embolics. El problema és que els productes de software de modelatge de dades utilitzen diferents tècniques. Per exemple, Erwin utilitza unes i Microsoft Visio unes altres. Quan es crea un model de dades, serà necessari conèixer la versió concreta del model ER que es vol utilitzar.

Nosaltres utilitzarem el modelatge IE Crow's Foot (pota de corb).

5. MODELATGE AMB CROW'S FOOT MODEL

5.1. Entitats

Qualsevol objecte (real o abstracte) que existeix a la realitat i que conté informació que volem emmagatzemar a la BD.

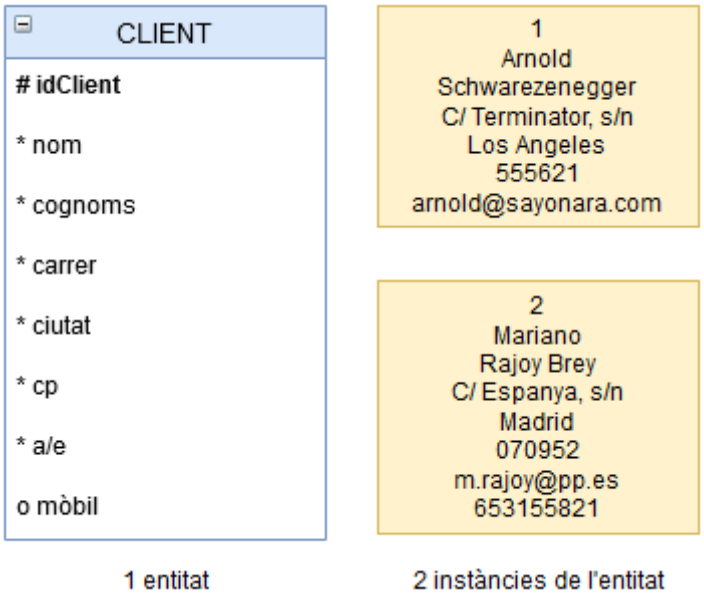
Les **entitats** representen conjunts d'elements amb existència pròpia i que es caracteritzen per les mateixes propietats. Generalment són persones, coses, llocs,... És a dir, conceptes sobre els quals necessitem guardar informació i distingibles dels altres objectes. La seva representació gràfica es fa per mitjà d'un quadrat dintre del qual s'escriu el nom de l'entitat en **majúscules**, en **singular** i amb la seva llista de característiques (també s'anomenen atributs o camps.

L'identificador (també anomenat característica principal o clau primària) és aquella que diferencia cadascuna de les instàncies d'una entitat. S'ha de posar amb un coixinet (#) i en negreta.

La resta de característiques van amb asterisc (*) (obligatori) o amb un *null* (o) (opcional) davant.

Els identificadors formats per dos o més atributs es diuen **identificadors compostos**.

Exemple: entitat CLIENT. Cada client concret que emmagatzemen dins de la BD direm que és una instància de la l'entitat CLIENT.



És important destacar que un mateix concepte no té perquè representar-se sempre de la mateixa forma (per exemple, com una entitat o com un atribut). Així, si estiguéssim modelant una Base de dades per a una botiga de roba, probablement tindríem una entitat denominada "PEÇA" i un dels seus atributs podria ser "color" (vermella, negra, etc.). No obstant això, si estiguéssim parlant d'una Base de dades per a gestionar la informació d'un taller de vehicles dedicat a treballs de xapa i pintura, el concepte de color pot tenir tal importància que passi a ser una entitat "COLOR", doncs té existència pròpia i un conjunt de propietats (codi de color, teixidura, tipus de barreja, etc.)

Per poder distingir un exemplar d'un altre, dintre d'un mateix tipus d'entitat, el model E/R obliga que cada vegada que definim un tipus d'entitat es defineixi un atribut que identifiqui cada

exemplar, és a dir, un **identificador principal (en inglés, UID unique identifier)**. Per tant en tots els tipus d'entitat ha d'aparèixer de forma obligatòria una característica que identifiqui de forma única cadascun dels exemplars. Aquesta és la representació que ens proporciona el model E/R per a distingir aquest tipus d'atribut de la resta d'atributs que componen el tipus d'entitat. En un tipus d'entitat només pot aparèixer un identificador principal, però poden existir diferents atributs que també identifiquin els exemplars d'aquesta; aquest tipus d'atributs es denominen **identificadors alternatius** (IA).

5.2. Relacions

Les entitats poden estar associades amb una o més **relacions**.

En Crow's Foot Model representarem les relacions mitjançant una **línia continua** que uneixi les entitats implicades. Les relacions tenen un nom que les descriu.

Grau de la relació

Una relació pot implicar associar dues o més entitats. El número d'entitats implicades en la relació es el **grau de la relació**. Les relacions de grau 2 és diuen binàries, mentre que les de grau 3 es diuen ternàries.

Quan és transforma el model de dades en un disseny de base de dades relacional, les relacions de qualsevol grau són tractades com a relacions binàries i tots el productes de software de modelatge de dades requereixen l'expressió de les relacions com de tipus binari.

Cardinalitat

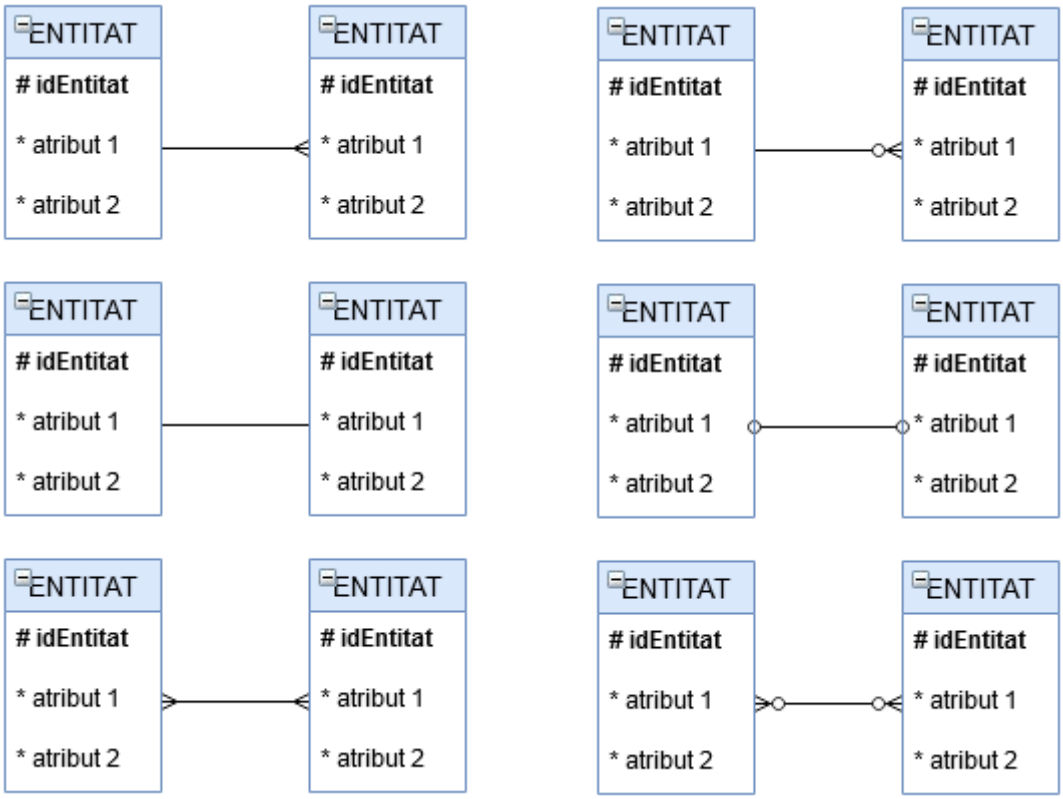
Es defineix com el nombre màxim d'instàncies d'una entitat que poden estar associades amb una altra instància d'altra entitat.

Addicionalment en el Crow's Foot Model també hem d'especificar els valors mínims existents entre les relacions, que prendran els valors 0 o 1.

Els possibles valors màxims que podrà valdre la cardinalitat de la relació entre dues entitats A i B determinaran que la relació tingui:

- **Cardinalitat 1:1** especifica que una entitat A pugui estar vinculada mitjançant una relació a una i només una ocurrència d'altra entitat B. A la seva vegada una ocurrència de l'entitat B només pot estar vinculada a una ocurrència de l'entitat A. Per exemple, es pot limitar el número de directors de departament mitjançant una relació 1:1. Així, un empleat només pot ser cap d'un departament i un departament només pot tenir un cap.
- **Cardinalitat 1:N** especifica que una entitat A pot estar vinculada mitjançant una relació a vàries ocurrències d'altra entitat B. Malgrat això, una de les ocurrències de l'entitat B només podrà estar vinculada a una ocurrència de l'entitat A. Per exemple, un manager gestiona les carreres de varis actors i un actor només podrà tenir un representant.
- **Cardinalitat N:N** especifica que una entitat A pot estar vinculada mitjançant una relació a varies ocurrències de l'entitat B, i a la seva vegada, una ocurrència de l'entitat B pot estar vinculada a varies de l'entitat A. Per exemple, un empleat pot treballar per a varis projectes i al mateix temps, en un mateix projecte poden treballar varis empleats.

Utilitzarem la notació següent per indicar la cardinalitat de les relacions, utilitzant la web www.draw.io:

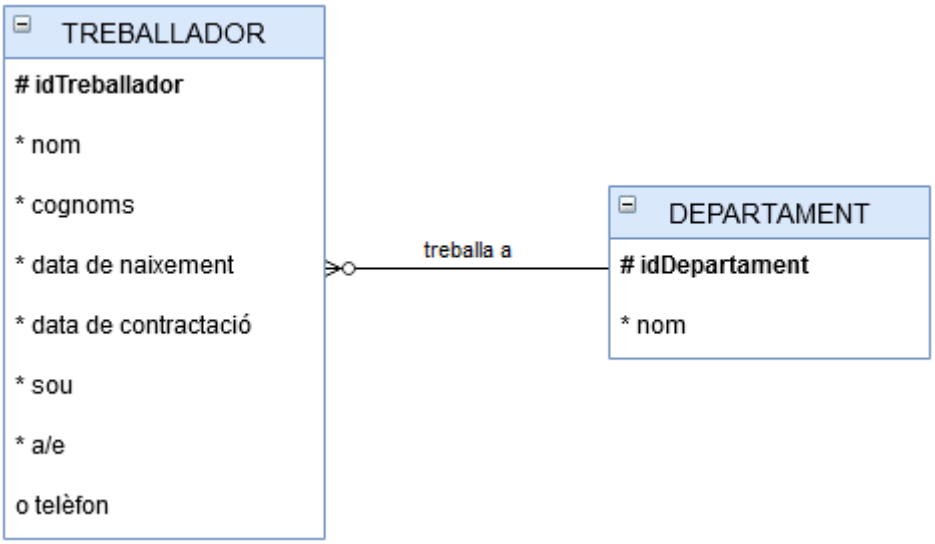


5.3. Exemples introductoris

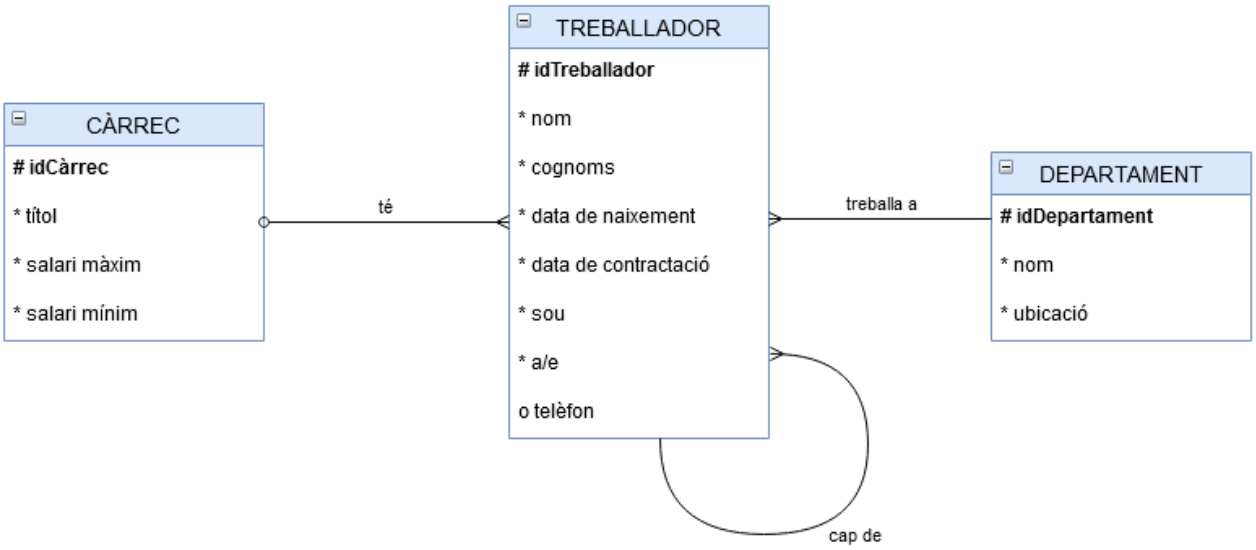
Exemple 1: Departament RRHH

El departament de recursos humans d'una gran companyia necessita emmagatzemar dades sobre cada un dels seus TREBALLADORS. Es necessita fer un seguiment del seu nom, adreça electrònica (a/e) i opcionalment el seu telèfon. A cada empleat se li assigna un número d'empleat únic. La companyia està dividida en DEPARTAMENTS. Cada empleat està assignat a un departament, per exemple, comptabilitat, vendes o desenvolupament. Cada departament té un nombre únic.

El seu model seria el següent:

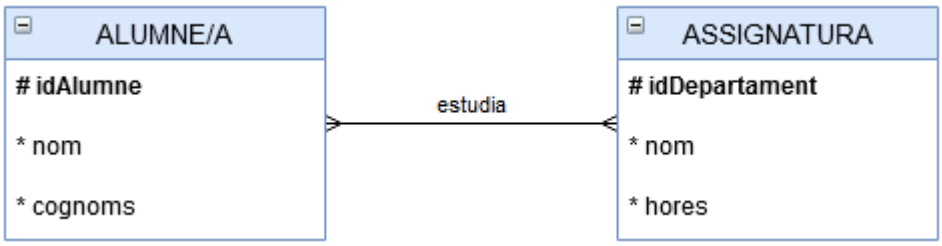


Si avaluéssim un model més complex, on es tinguessin en compte els càrrecs, caps i ubicacions dels departaments, el model de dades s'ampliaria sent la següent figura una possible solució:



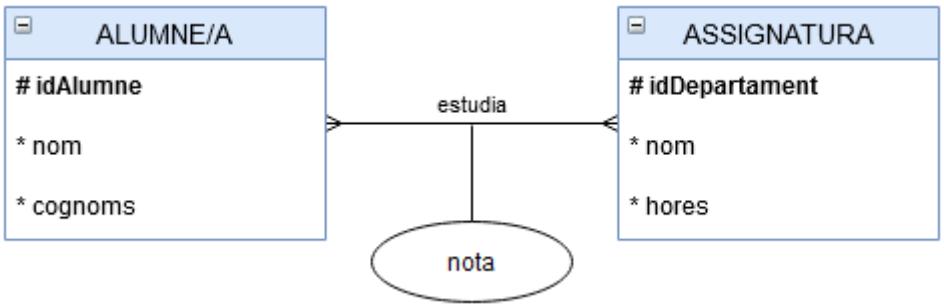
Exemple 2: Centre educatiu

En un centre educatiu, un ALUMNE/A pot estudiar una o diverses assignatures. Cada ASSIGNATURA pot ser estudiada per un o més alumnes/as.



Però, que passaria si desitgem poder enregistrar la nota que aconsegueix un/a alumne/a per a cada assignatura? A quina entitat pertanyeria l'atribut "nota"? Si posem "nota" a l'entitat ALUMNE/A, com sabrem per a què ASSIGNATURA és? Si posem "nota" a l'entitat ASSIGNATURA, com sabrem quin ALUMNE/A va obtenir aquesta nota?

Solució: Agregar l'atribut "nota" dintre d'un cercle i unir-lo a la relació entre les dues entitats, tal i com mostra la següent imatge:



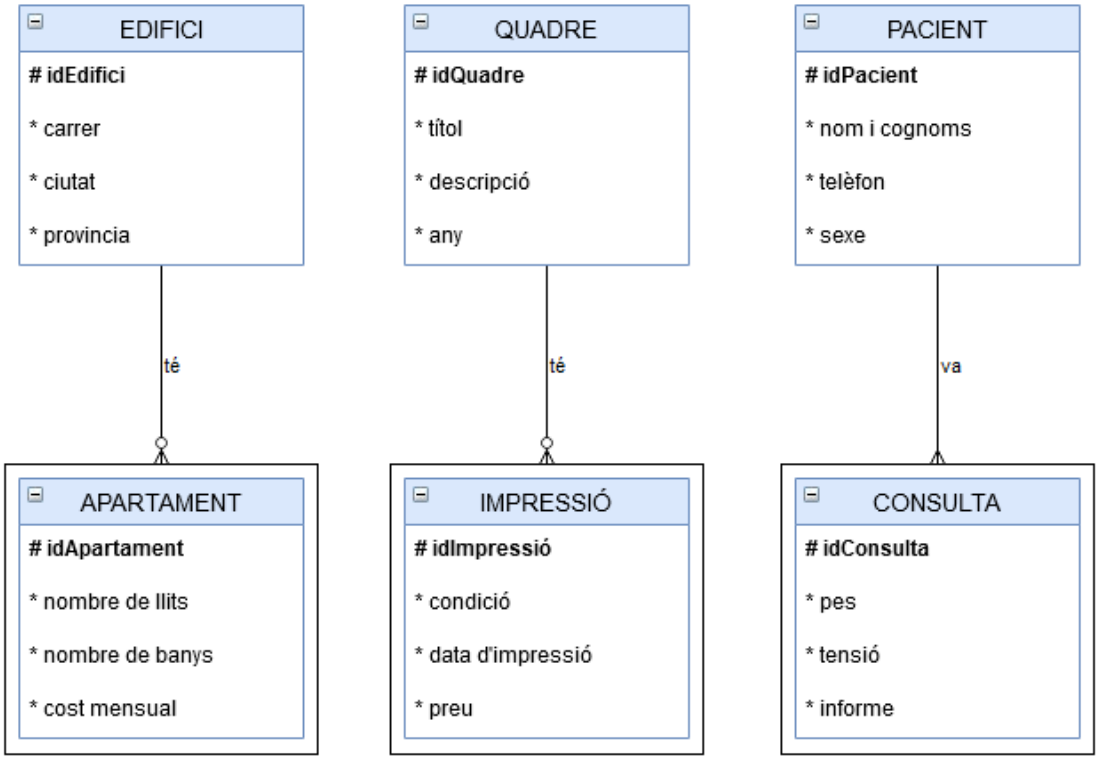
6. MODELATGE ESTÈS AMB CROW'S FOOT MODEL

6.1. RELACIONS DE DEPENDÈNCIA

6.1.1. Entitat forta vs Entitat dèbil

Una entitat forta és aquella que representa quelcom que pot existir de forma independent.

El model ER inclou un tipus especial d'entitat dèbil anomenada **ID-dependent entity** que depèn de la existència d'una entitat forta. Una ID-dependent entity és una entitat els identificadors de la qual inclouen l'identificador de l'entitat de la qual depèn.



En cadascun dels exemples, l'entitat dèbil no pot existir sense que el seu pare (l'entitat forta de la qual depèn) existeixi. Per tant, la cardinalitat mínima de l'entitat dèbil cap al seu pare sempre serà 1.

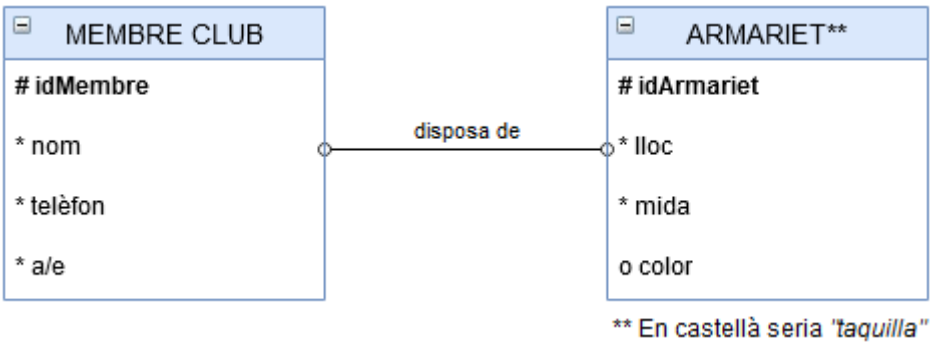
No obstant això, el pare podria requerir també tenir una entitat dèbil sempre associada segons els requeriments de l'aplicació. Per exemple, APARTAMENT i IMPRESSIÓ són opcionals, però CONSULTA és obligatori. Les restriccions vindran donades per la natura de la aplicació i no cap requeriment lògic.

En el model ER, utilitzarem una entitat amb **una rectangle sòlid per fora per representar la relació entre l'entitat dèbil i la forta**.

Les entitats dèbils estableixen restriccions en el procés de creació de la base de dades que es construirà en base al model. Les fileres que representin a l'entitat pare hauran de ser creades abans de que qualsevol filera filla pugui ser creada. A més, quan una filera pare sigui esborrada, totes les fileres filles també ho hauran de ser.

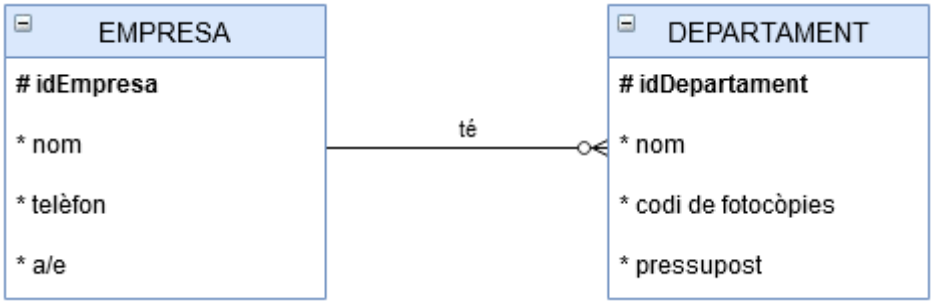
6.1.2. Exemples d'entitats fortes relacionades

Cardinalitat 1:1

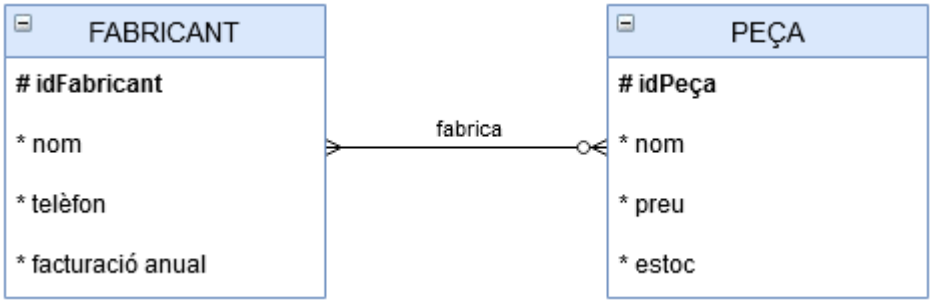


És poc probable que tots els armariets estiguin ocupats, i haurà alguns que estiguin sense utilitzar.

Cardinalitat 1:N



Cardinalitat N:N

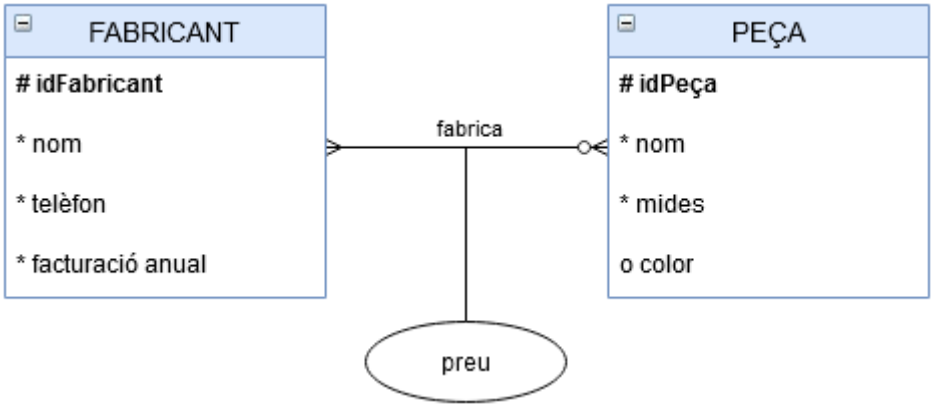


Un FABRICANT fabrica moltes peces (PEÇA), i una peça és subministrada/fabricada per molts fabricants. Per tant, la relació és N:N. A causa de que no tots els fabricants fabriquen el mateix, la relació del FABRICANT amb la PEÇA ha de ser opcional. No obstant això, cada peça ha de ser subministrada des d'un lloc, de manera que la relació amb FABRICANT és obligatòria.

6.1.3. Exemple de relació de dependència d'identificació

Patró d'Associació

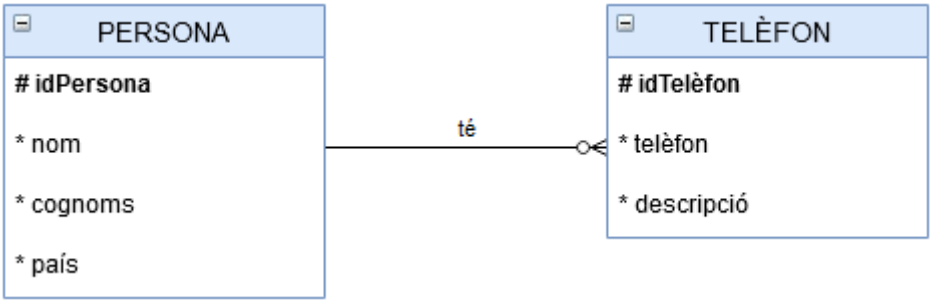
Tal i com hem vist anteriorment, si partim d'una relació N:N entre un FABRICANT i una PEÇA que fabrica, i volguéssim incloure el preu d'aquesta peça segons el fabricant, veiem que aquest atribut preu no és ni atribut del FABRICANT ni de la PEÇA, sinó de la seva combinació:



Patró d'atributs multivaluats

En el model ER, els atributs sempre hauran de tenir un únic valor.

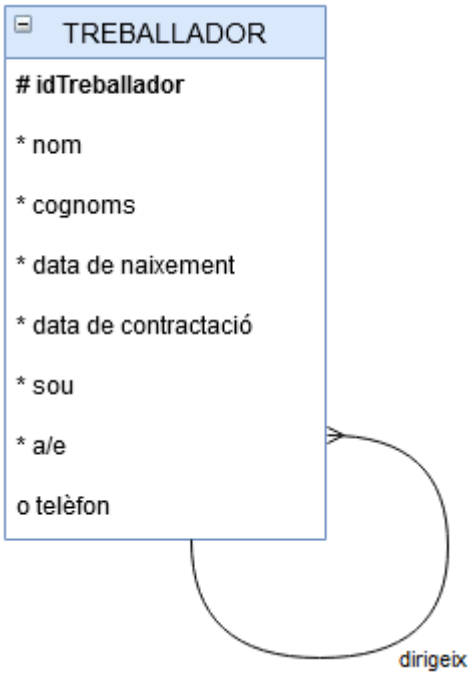
Per tant, mostrarem els atributs multivaluats com entitats a banda amb una relació 1:N amb l'entitat de la qual són atributs i amb la mateixa clau primària que aquesta. A continuació es mostra com representar els atributs multivaluats TELÈFON i PERSONA :



6.2. RELACIONS REFLEXIVES

Una relació reflexiva és aquella que associa dues entitats del mateix tipus.

Per exemple, quan un treballador pot gestionar a un altre grup de treballadors (zero o molts) i un treballador té un únic cap, llavors tindrem una relació reflexiva 1:N com la següent:



7. MODEL RELACIONAL

El model de dades conceptual es transformarà en un disseny de base de dades relacional.

Això vol dir que les nostres entitats, atributs, relacions i identificadors únics es convertiran en camps d'una base de dades relacional.

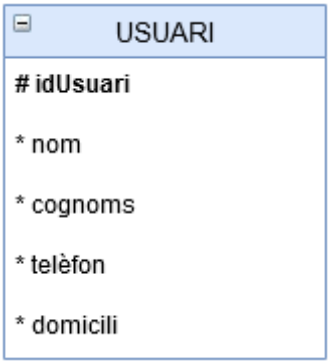
Una base de dades relacional és una base de dades que l'usuari veu com un recull de taules bidimensionals, on cadascuna conté files i columnes.

La següent taula conté dades de treballadors:

	idTreballador	nom	cognoms	idDepartament
	100	Steven	Spielberg	90
fila →	101	Lex	Luthor	90
	102	Peter	Parker	90
	200	Quentin	Tarantino	10
	205	Jimmy	Hendrix	110
		↑ columna		

7.1. Entitats

Quan passem una entitat al model relacional, ho farem posant el nom de l'entitat en minúscules i en plural seguit de tots els seus atributs entre parèntesis i separats per coma, tal i com mostra el següent exemple:



L'entitat anterior quedarà representada en el model relacional com:

```
usuaris(idUsuari, nom, cognoms, telefon, domicili)
```

Atenció: S'ha de tenir en compte que els atributs quan els passem al model relacional no poden tenir caràcters especials com accents, ce trencada,...

7.1. Clau primària (PK, Primary Key) i Clau aliena (Foreign Key, FK)

La **clau primària (PK)** és una columna o joc de columnes que identifica de manera única cada fila d'una taula. Tota taula ha de tenir una clau primària i una clau principal ha de ser única. Una clau primària permet accedir a un registre concret d'una base de dades (sense generar confusions ni conflictes ja que és únic i no nul).

Identificarem la clau primària d'una taula subratllant-la amb una línia contínua i **posant-la en negreta**.

Una **clau aliena (FK)** és una columna o combinació de columnes d'una taula que conté valors que coincideixen amb el valor de clau primària d'una altra taula.

Identificarem la clau aliena d'una taula subratlla-la amb una línia discontínua i *posant-la en cursiva*.

De vegades, una clau primària actua també com a clau aliena d'una altra taula (sobretot en entitats febles) **(PK+FK).** En aquests casos haurem de subratllar-la amb dues línies i ***posant-la en negreta i cursiva a la vegada***.

idTreballador	nom	cognoms	idDepartament
100	Steven	Spielberg	90
101	Lex	Luthor	90
102	Peter	Parker	90
200	Quentin	Tarantino	10
205	Jimmy	Hendrix	100

↑ clau forana

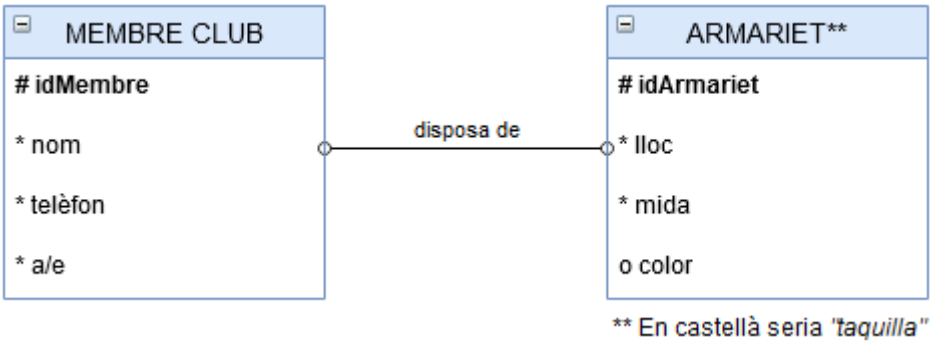
Fa referència a

idDepartament	nom del departament
10	Comptabilitat
20	Facturació
90	Informàtica
100	Màrqueting

↑ clau principal

7.2. Transformació de relacions 1:1

En aquest cas podem referenciar la clau primària de qualsevol de les dues entitats en l'altra entitat. Solament haurem de tenir en compte que si un dels costats d'aquesta relació és opcional, sempre serà millor que sigui l'entitat del costat de la relació opcional la que referenciï a la clau primària de l'altra.



En aquest exemple, i com els dos costats són opcionals, tenim llibertat d'elegir quina entitat referenciarà a l'altra. Així podríem obtindré tant les següents taules:

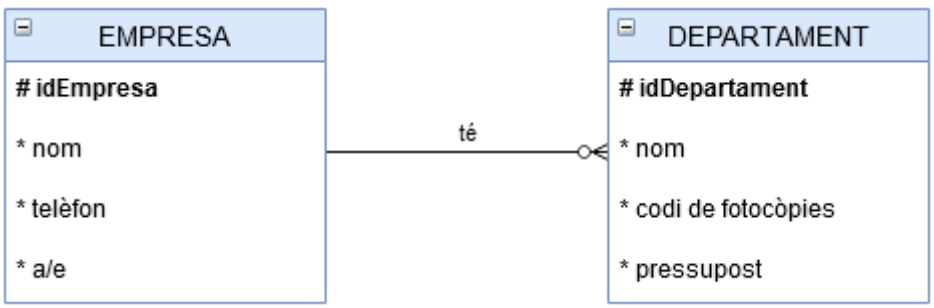
```
membresclub(idMembre, nom, telefon, email)
armariets(idArmariet, , lloc, mida, color, idMembre)
ON {idmembre} REFERENCIA membresclub
```

O les següents:

```
membresclub (idMembre, nom, telefon, email, idArmariet)
ON {idArmariet} REFERENCIA armariets
armariets(idArmariet, lloc, mida, color)
```

7.3. Transformació de relacions 1:N

A partir d'ara, en tots els nostres models haurem d'afegir a la taula que es correspon amb el costat N, una FK que faci referència a la PK de la taula del costat 1 amb el que està associada.



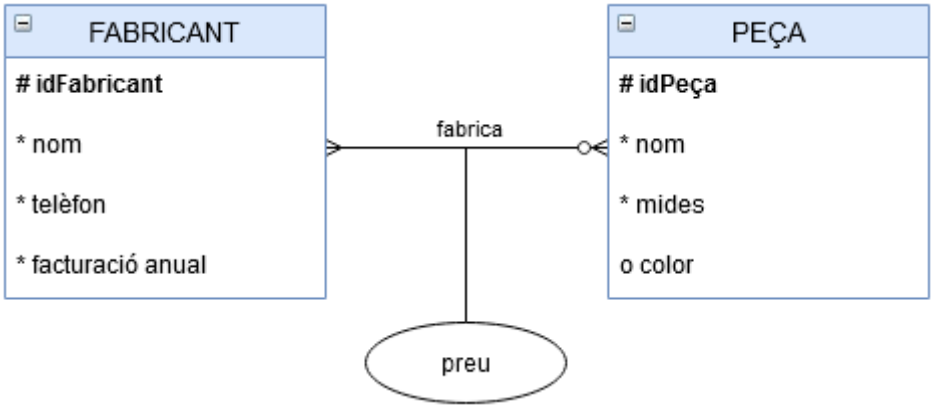
En aquest exemple serà l'entitat DEPARTAMENT que està al costat N la que referenciarà a l'entitat EMPRESA.

```
empreses(idEmpresa, nom, telefon, email)
departaments(idDepartament, , nom, codiDeFotocopies, pressupost,
idEmpresa) ON {idEmpresa} REFERENCIA empreses
```

7.3. Transformació de relacions N:N

Es crea una nova taula que tindrà com clau primària la concatenació dels dos identificadors principals de les entitats que associa. L'ordre dels identificadors es triarà segons es consideri més adequat.

A més cadascun dels atributs que formen la clau primària d'aquesta taula també són claus alienes que referencien a les taules que s'han convertit les entitats relacionades.

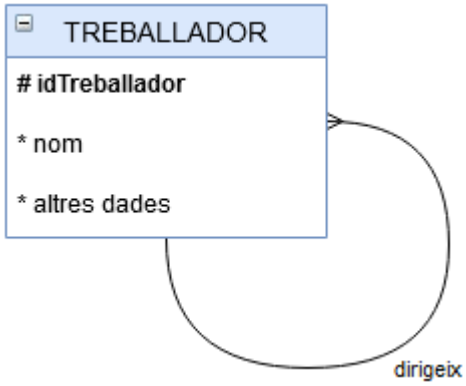


En aquest exemple es crearà una entitat nova (que es pot anomenar com la unió del nom de les dues entitats) "fabricantspeces" (en minúscules) agafant com a clau primària cadascuna de les claus primàries de les dues entitats inicials. Aquestes claus primàries seran a la vegada alienes, ja que referenciaran també a l'entitat corresponent:

```
fabricants(idFabricant, nom, telefon, facturacioAnual)

peces(idPeca, nom, mides, color)

fabricantspeces(idFabricant, idPeca, preu)
ON {idFabricant} REFERENCIA fabricants I

I {idPeca} REPRESENTA peca
```

7.4. Transformació de relacions reflexives

Les relacions reflexives es tracten de la mateixa forma que les altres, només que un mateix atribut pot figurar dues vegades en una taula com resultat de la transformació. Això sí, com no poden haver dos atributs amb el mateix nom, haurem de modificar-lo afegint-li un 2 al final o de la manera que considerem que l'identifica millor.

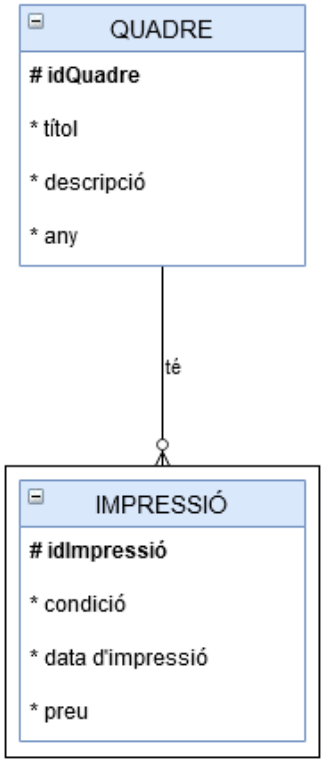


En aquest exemple, com la relació reflexiva és 1:N, serà la mateixa entitat la que es referenciarà a ella mateixa, quedant:

```
treballadors(idTreballador, , altresDades, ..., idTreballadorCap)
```

7.5. Transformació d'entitats febles

Tota entitat feble incorpora una relació implícita amb la seva entitat forta, per tant, l'identificador de l'entitat feble estarà format per la seva pròpia clau primària junt a la clau primària de l'entitat forta, a la qual també referenciarà, per tant, aquesta última serà primària i aliena a la vegada. Aquí s'obliga a una modificació i un esborrat en cascada.



Aquesta relació quedaria reflectida com:

```
quadres(idQuadre, titol, descripcio, any)
impressions(idQuadre,idImpressio, , condicio, dataImpressio, preu)
ON {idQuadre} REPRESENTA quadre
```