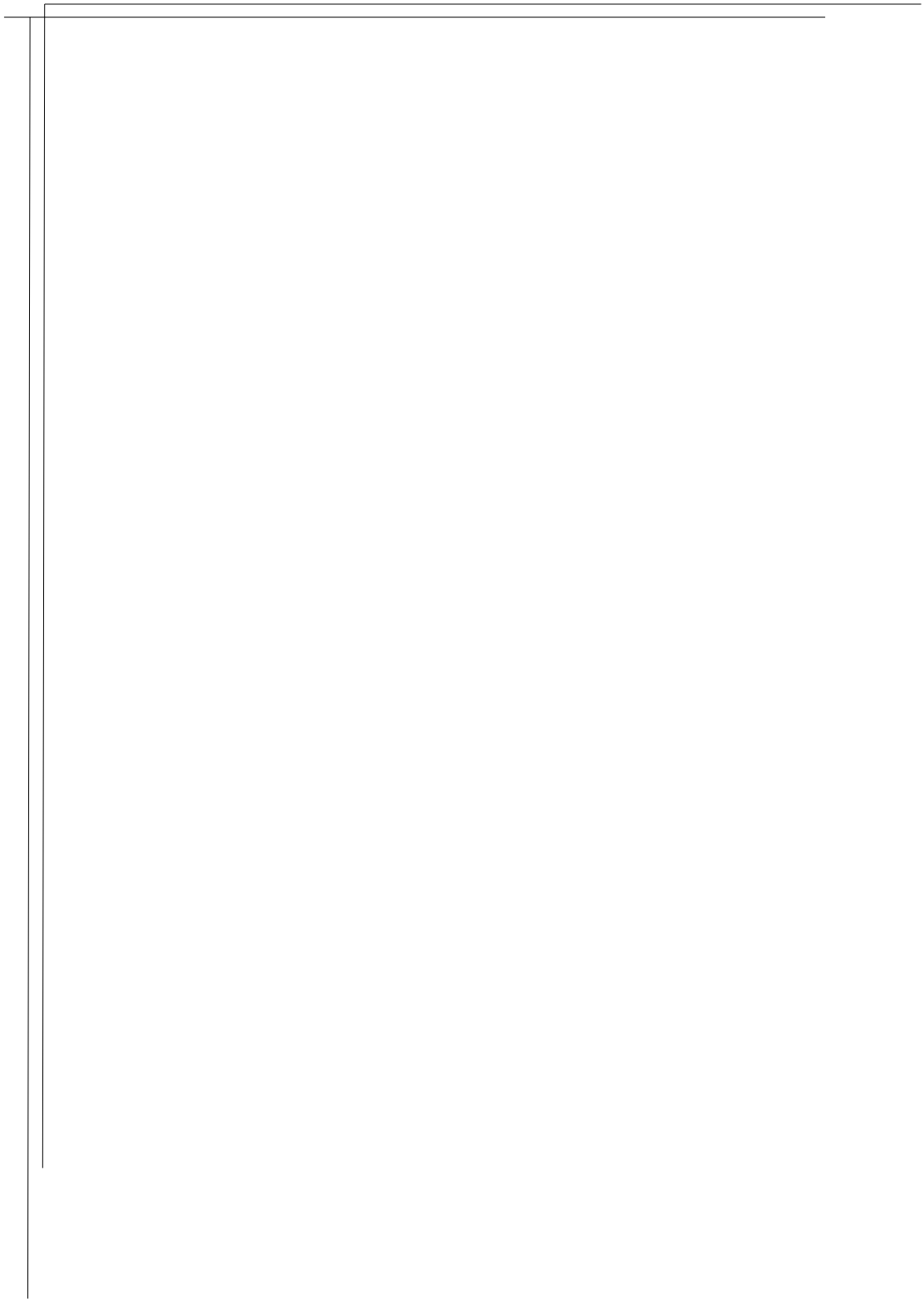
SGBD

Héctor Costa Guzman

Índex

Capítol 1. SGBD



1.1 Introducció

1.1.1 Definició dels SGBD

SGBD significa **S**istema de **G**estió de **B**ase de **D**ades o en anglès i és un grup de programes l'objectiu del qual és definir, construir i manipular una base de dades.

- Definir una base de dades: establir els tipus de dades, estructures i restriccions a les dades que s'emmagatzemaran.
- Construir una base de dades: és el procés d'emmagatzemar les dades sobre algun mitjà d'emmagatzematge.
- Manipular una base de dades: inclou funcions com a consulta, actualització, etc. de bases de dades.
- Si el sistema suporta bases de dades relacionals es diu RDBMS en anglès o SGBDR en espanyol.

1.1.2 Història dels SGBD

Els sistemes de bases de dades tenen les seves arrels en el projecte nord-americà Apolo d'enviar l'home a la lluna, als anys seixanta. En aquella època, no hi havia cap sistema que permetés gestionar la immensa quantitat d'informació que requeria el projecte.

La primera empresa encarregada del projecte, NAA (North American Aviation), va desenvolupar un programari anomenat GUAM (General Update Access Method) que estava basat en el concepte de que diverses peces petites s'uneixen per formar una peça més gran, i així successivament fins que el producte final està ensamblat.

Aquesta estructura, que té la forma d'un arbre, és el que s'anomena una estructura jeràrquica.

A meitat dels seixanta, es va desenvolupar IDS (Integrated Data Store), de General Electric. Aquest treball va ser dirigit per un dels pioners en els sistemes de bases de dades, Charles Bachmann. IDS era un nou tipus de sistema de bases de dades conegut com a sistema de xarxa, que va produir un gran efecte sobre els sistemes d'informació d'aquella generació.

Els sistemes jeràrquic i de xarxa constitueixen la primera generació dels SGBD.

Però aquests sistemes presenten alguns inconvenients:

- Cal escriure complexos programes d'aplicació per a respondre a qualsevol tipus de consulta de dades, per simple que aquesta sigui.

- La independència de dades és mínima.
- No tenen un fonament teòric.

El 1970 Codd, dels laboratoris d'investigació d'IBM, va escriure un article presentant el model relacional. En aquest article, presentava també els inconvenients dels sistemes previs, el jeràrquic i el de xarxa.

Llavors, es van començar a desenvolupar molts sistemes relacionals, apareixent els primers a finals dels setanta i principis dels vuitanta.

IBM va desenvolupar per provar la funcionalitat del model relacional, proporcionant una implementació de les seves estructures de dades i les seves operacions. Això va conduir a dos grans desenvolupaments:

- El desenvolupament d'un llenguatge de consultes estructurat anomenat SQL, que s'ha convertit en el llenguatge estàndard dels sistemes relacionals.
- La producció de diversos SGBD relacionals durant els anys vuitanta, com DB2 i SLQ / DS de IBM, i ORACLE d'ORACLE Corporation.

Avui en dia hi ha centenars de SGBD relacionals, tant per a microordinadors com per a sistemes multiusuari, encara que molts no són completament fidels al model relacional.

Com a resposta a la creixent complexitat de les aplicacions que requereixen bases de dades, han sorgit dos nous models: el model de dades orientat a objectes i el model relacional estès. No obstant això, a diferència dels models que els precedeixen, la composició d'aquests models no està clara. Aquesta evolució representa la tercera generació dels SGBD.

1.1.3 Inconvenients dels sistemes de gestió d'arxius

Aquests són alguns inconvenients de la primera generació de BBDD:

- Redundància i inconsistència de les dades: no hi ha un control del tipus d'arxiu, així que aquests poden canviar i tenir formats diferents, a més les dades poden repetir-se en diferents llocs.
- Dependència de les dades física-lògica: com que l'estructura d'aquestes dades ve codificada per les aplicacions, qualsevol canvi en l'estructura física de les dades implica que el programador tingui que buscar i modificar aquestes aplicacions manualment.
- Dificultat per tenir accés a les dades: quan es fa una consulta no prevista dóna lloc a la re-codificació de la aplicació en questió.
- Separació i aïllament de les dades: com que les dades estan repartides en diferents arxius i possiblement en diferents formats, es molt difícil crear noves aplicacions que manipulin correctament les dades. Quan això passa, s'han de sincronitzar els diferents arxius i fer coincidir les dades.
- Dificultat per a l'accés concurrent: és molt difícil que hi hagi diferentes persones actualitzant dades a la vegada, ja que s'utilitzen aplicacions diferents.
- Problemes en la seguretat de les dades: com que les aplicacions es van creant a mesura que es necessiten, la seguretat es difícil d'implantar.
- Problemes d'integritat de dades: els valors dels arxius han de ser consistents. No es pot inserir una entrada a un camp, si aquest camp no s'ha creat anteriorment.

1.1.4 Ventatges dels SGBD

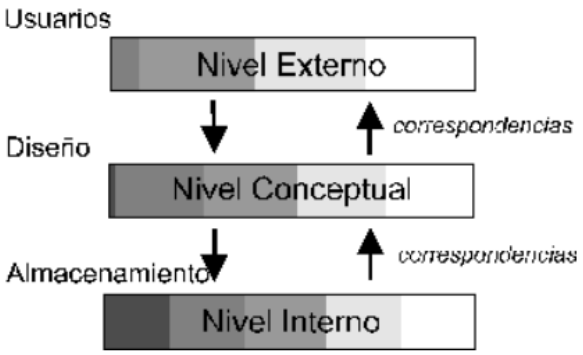
Els numersos inconvenients de la primera generació de BBDD van desembocar en la segona geneació, els SGBD. L'objectiu primordial d'un gestor és proporcionar eficiència i seguretat a l'hora d'extreure o emmagatzemar informació a les BBDD.

Han de complir els serveis següents:

- Millora en la integritat de dades: la integritat s'expressa mitjançant restriccions o regles que no es poden violar. Aquestes restriccions es poden aplicar tant a les dades, com a les seves relacions, i és el SGBD qui s'ha d'encarregar de mantenir-les.
- Millora en la seguretat: la seguretat de la base de dades és la protecció de la base de dades front a usuaris no autoritzats. Sense unes bones mesures de seguretat, la integració de dades en els sistemes de bases de dades fa que aquests siguin més vulnerables que en els sistemes de fitxers.
- Millora en l'accessibilitat a les dades: Molts SGBD proporcionen llenguatges de consultes o generadors d'informes que permeten a l'usuari fer qualsevol tipus de consulta sobre les dades, sense que sigui necessari que un programador escrigui una aplicació que realitzi aquesta tasca.
- Millora en la productivitat: L'SGBD proporciona moltes de les funcions estàndard que el programador necessita escriure en un sistema de fitxers. El fet de disposar d'aquestes funcions permet al programador centrar millor en la funció específica requerida pels usuaris. Gràcies a aquestes eines, el programador pot oferir una major productivitat.
- Millora en el manteniment: els SGBD separen les descripcions de les dades de les aplicacions. Això és el que es coneix com **independència de dades**, gràcies a la qual se simplifica el manteniment de les aplicacions que accedeixen a la base de dades.
- Augment de la concurrència: La majoria dels SGBD gestionen l'accés concurrent a la base de dades i garanteixen que no es produeixin problemes d'aquest tipus.
- Millora en els serveis de còpies de seguretat i de recuperació: Els usuaris han de fer còpies de seguretat cada dia, i si es produeix algun error, utilitzar aquestes còpies per a restaurar-los.

1.2 Arquitectura dels SGBD

El 1975, el comitè ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) va proposar una arquitectura de tres nivells per als sistemes de bases de dades, que resulta molt útil a l'hora d'aconseguir aquestes tres característiques.



L'objectiu de l'arquitectura de tres nivells és el de separar els programes d'aplicació de la base de dades física. En aquesta arquitectura, l'esquema d'una base de dades es defineix en tres nivells d'abstracció diferents:

1. En el **nivell intern** es descriu l'estructura física de la base de dades mitjançant un esquema intern. Aquest esquema s'especifica mitjançant un model físic i descriu tots els detalls per a l'emmagatzematge de la base de dades, així com els mètodes d'accés.
2. En el **nivell conceptual** es descriu l'estructura de tota la base de dades per a una comunitat d'usuaris (tots els d'una empresa o organització), mitjançant un *esquema conceptual*. Aquest esquema oculta els detalls de les estructures d'emmagatzematge i es concentra en descriure entitats , atributs, relacions, operacions dels usuaris i restriccions.
3. En el **nivell extern** es descriuen diversos *esquemes externs* o *vistes d'usuari.* Cada esquema extern descriu la part de la base de dades que interessa a un grup d'usuaris determinat i oculta a aquest grup la resta de la base de dades. En aquest nivell es pot utilitzar un model conceptual o un model lògic per especificar els esquemes.

L'arquitectura de tres nivells és útil per explicar el concepte *d'independència de dades* que podem definir com la capacitat per modificar l'esquema en un nivell del sistema sense haver de modificar l'esquema del nivell immediat superior.

Es poden definir dos tipus d'independència de dades:

- La independència lògica és la capacitat de modificar l'esquema conceptual sense haver d'alterar els

esquemes externs ni els programes d'aplicació. Es pot modificar l'esquema conceptual per ampliar la base de dades o per reduir-la. Si, per exemple, es redueix la base de dades eliminant una entitat, els esquemes externs que no es refereixin a ella no es veuran afectats.

- La independència física és la capacitat de modificar l'esquema intern sense haver d'alterar l'esquema conceptual (o els externs). Per exemple, pot ser necessari reorganitzar certs fitxers físics per tal de millorar el rendiment de les operacions de consulta o d'actualització de dades. Atès que la independència física es refereix només a la separació entre les aplicacions i les estructures físiques d'emmagatzematge, és més fàcil d'aconseguir que la independència lògica.

1.3 Components del SGBD

Els SGBD són paquets de programari molt complexos que han de proporcionar una sèrie de serveis que permetran emmagatzemar i explotar les dades de forma eficient. Els components principals són els següents:

1.3.1 Llenguatges dels SGBD

Els SGBD han d'oferir llenguatges i interfícies apropiades per a cada tipus d'usuari: administradors de la base de dades, dissenyadors, programadors d'aplicacions i usuaris finals.

- Llenguatge de definició de dades (LDD o DDL): s'utilitza per especificar l'esquema de la BD, les vistes dels usuaris i les estructures d'emmagatzematge. És el que defineix l'esquema conceptual i l'esquema intern. L'utilitzen els dissenyadors i els administradors de la BD.

- Llenguatge de manipulació de dades (LMD o DML): Hi ha dos tipus de LMD: els procedurals i els no procedurals. Amb un **LMD procedural** l'usuari (normalment serà un programador) especifica quines dades es necessiten i com cal obtenir-los. Això vol dir que l'usuari ha d'especificar totes les operacions d'accés a dades trucant als procediments necessaris per obtenir la informació requerida.

  Un **LMD no procedural** es pot utilitzar de manera independent per especificar operacions complexes sobre la base de dades de forma concisa. En molts SGBD es poden introduir interactivament instruccions del LMD des d'un terminal o bé embegudes en un llenguatge de programació d'alt nivell. Les bases de dades relacionals utilitzen LMD no procedurals, com SQL (Structured Query Language) o QBE (Query-By-Example).

1.3.2 El Diccionari de Dades

Tot sistema de gestió de bases de dades té programari integral per accedir a les metadades que descriuen una estructura de la base de dades. Aquesta col·lecció de metadades i funcions, **el diccionari de dades de la SGBD**, és necessari per a suportar consultes i llenguatges de manipulació de dades, com ara SQL. El diccionari proporciona la següent informació:

- descripcions detallades de taules i camps
- informació d'indexació
- restriccions d'integritat referencial
- definicions d'esquema de base de dades
- procediments emmagatzemats i disparadors
- informació de control d'accés, com ara noms d'usuari, rols, i privilegis
- paràmetres d'ubicació de l'emmagatzemament
- estadístiques d'ús de la base de dades

1.3.3 Seguretat i Integritat de Dades

Un SGBD ha de garantir la seguretat i integritat de les dades:

- Ha de protegir les dades contra accessos no autoritzats o accidentals. A més ha de proporcionar un absolut control sobre els usuaris que poden accedir a la base de dades.
- Normalment, els SGBD tenen mecanismes per controlar restriccions d'integritat a la base de dades, les restriccions serveixen per protegir la base de dades contra accidents. A més, el propi SGBD es capaç de determinar si es produeix una violació d'aquestes restriccions.
- També proporcionen mecanismes per planificar i realitzar còpies de seguretat i de restauració.
- Finalment, en cas de que hi hagi algun problema, el SGBD ha de ser capaç de recuperar la base de dades a un estat anterior.

1.3.4 L'Administrador de la BD

L'Administrador de Bases de Dades és qui posseeix els coneixements sobre el llenguatge estructurat de consultes (SQL).

A més posseeix les habilitats i destreses necessàries per a la implementació, configuració i posada a punt del motor de la base de dades. És també qui desenvolupa una metodologia d'anàlisi i avaluació de l'estructura de Bases de Dades Relacionals.

Entre les seves responsabilitats es troben:

- Planejar i crear bases de dades
- Administrar l'accés, els recursos i estructures (tant físiques com lògiques) de les mateixes.
- Finalment també té la responsabilitat d'administrar usuaris i els seus privilegis.

1.4 Models de dades

Els models de dades proporcionen a l'usuari una visió abstracte de les dades:

- **Models logics basats en objectes**: els dos més estesos són el model entitat-relació i el orientat a objectes. El *model entitat-relació* (ER) es basa en una percepció del món composta per objectes, anomenats entitats, i relacions entre ells.

  Les entitats es diferencien unes de les altres a través d'atributs. L'orientat a objectes també es basa en objectes, els quals contenen valors i mètodes, entesos com ordres que actuen sobre els valors, en nivells d'anidament.

  Els objectes s'agrupen en classes, relacionant-se mitjançant l'enviament de missatges. Alguns autors defineixen aquests models com a "models semàntics".

- **Models lògics basats en registres**: el més estès és el relacional, mentre que els altres dos existents, jeràrquic i de xarxa, es troben en retrocés. Aquests models es fan servir per a especificar l'estructura lògica global de la base de dades, estructurada en registres de format fix de diversos tipus.

  - *El model relacional:* Des dels anys 80 és el model més utilitzat, ja que permet una major eficàcia, flexibilitat i confiança en el tractament de les dades. La major part de les bases de dades i sistemes d'informació actuals es basen en el model relacional ja que ofereix nombrosos avantatges sobre els 2 models anteriors, com és el ràpid aprenentatge per part d'usuaris que no tenen coneixements profunds sobre sistemes de bases de dades .

    En el model relacional es representa el món real mitjançant taules relacionades entre si per columnes comuns. Les bases de dades que pertanyen a aquesta categoria es basen en el model relacions, l'estructura principal és la relació, és a dir una taula bidimensional composta per línies i columnes.

    Cada línia, que en terminologia relacional es diu tupla, representa una entitat que nosaltres volem memoritzar a la base de dades. les característiques de cada entitat estan definides per les columnes de les relacions, que s'anomenen atributs. Entitats amb característiques comunes, és a dir descrites pel mateix conjunt d'atributs, formaran part de la mateixa relació.

  - *El model de xarxa* està format per col.leccions de registres, relacionats mitjançant punters o lligues a grafs arbitraris.

    Permet la representació de molts a molts, de tal manera que qualsevol registre dins de la base de dades pot tenir diverses ocurrències superiors a ell. El model de xarxa evita redundància en la

informació, a través de la incorporació d'un tipus de registre anomenat el connector. En el model en xarxa es representa el món real mitjançant registres lògics que representen una entitat i que es relacionen entre si per mitjà de fletxes

- *El model jeràrquic* és similar al de xarxa, però els registres s'organitzen com a col.leccions d'arbres. Alguns autors defineixen aquests models com a "models de dades clàssics".

  Pot representar dos tipus de relacions entre les dades: relacions d'un a un i relacions d'un a molts. Aquest model té forma d'arbre invertit en què una branca pot tenir diversos fills, però cada fill només pot tenir un pare.

- **Models físics de dades:** molt poc usats, són el model unificador i el de memòria d'elements. Alguns autors defineixen aquests models com a "models de dades primitius".

1.5 Model entitat-interrelació

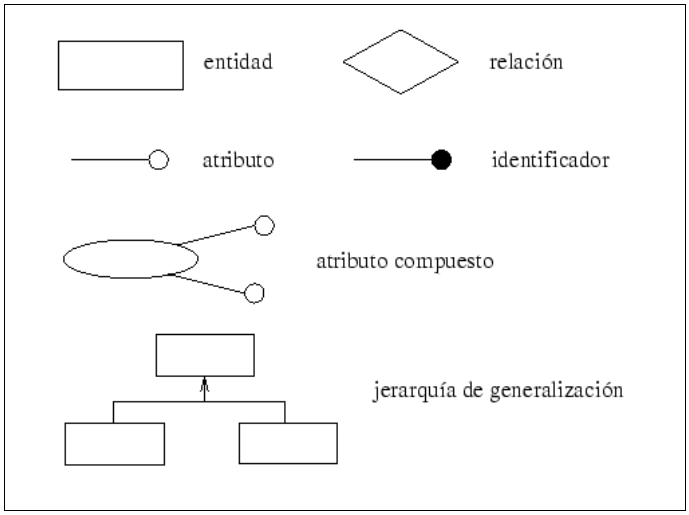
1.5.1 Definició

El model entitat-relació és el model conceptual més utilitzat per al disseny conceptual de bases de dades. Va ser introduït per Peter Chen el 1976. El model entitat-relació està format per un conjunt de conceptes que permeten descriure la realitat mitjançant un conjunt de representacions gràfiques i lingüístiques.

Originalment, el model entitat-relació només incloïa els conceptes d'entitat, relació i atribut. Més tard, es van afegir altres conceptes, com els atributs compostos i les jerarquies de generalització, en el que s'ha anomenat model entitat-relació estès.

El model E-R s'ha guanyat el lloc gràcies a la seva potència i s'ha extés molt rapidament per representar dades. Proposa l'ús de les taules bidimensionals per representar les dades i les seves corresponents relacions.

1.5.2 Conceptes bàsics



- L'entitat: Qualsevol tipus d'objecte o concepte sobre el qual es recull informació: cosa, persona, concepte abstracte o succés. Per exemple: cotxes, cases, empleats, clients, empreses, oficis, dissenys de productes, concerts, excursions, etc. Les entitats es representen gràficament mitjançant rectangles i el seu nom apareix a l'interior. Un nom d'entitat només pot aparèixer un cop en l'esquema conceptual.
- Conjunt d'entitats: és un conjunt de relacions del mateix tipus.
- Entitat forta: Una entitat forta és una entitat l'existència de la qual no depèn d'una altra entitat. És el contrari d'una entitat dèbil.

1.5 Model entitat-interrelació

- Atributs o camps: És una característica d'interès o un fet sobre una entitat o sobre una relació. Els atributs representen les propietats bàsiques de les entitats i de les relacions. Tota la informació extensiva és portada pels atributs. Gràficament, es representen mitjançant boletes que pengen de les entitats o relacions a les que pertanyen.

  Els atributs poden ser simples o compostos. Un atribut simple és un atribut que té un sol component, que no es pot dividir en parts més petites que tinguin un significat propi. Un atribut compost és un atribut amb diversos components, cadascun amb un significat per si mateix.

- Domini: El domini defineix tots els valors possibles que pot prendre un atribut. Pot haver-hi diversos atributs definits sobre un mateix domini.

- Identificador o superclau: Un identificador d'una entitat és un atribut o conjunt d'atributs que determina de manera única cada ocurrència d'aquesta entitat. Un identificador d'una entitat ha de complir dues condicions:

  1. No poden existir dos ocurrències de l'entitat amb el mateix valor de l'identificador.
  2. Si s'omet qualsevol atribut de l'identificador, la condició anterior deixa de complir-se.

- Clau candidata: Una superclau formada pel mínim nombre de camps possibles.

- Clau primària: és una clau cadidata escollida pel dissenyador de la BD. La clau candidata no pot contenir valors nuls, ha de ser senzilla de crear i no pot cambiar amb el temps. Els atributs que formen aquesta clau es representen subratllats.

- Clau aliena o forània: és un atribut d'una taula o addició d'atributs ja existents a una taula; aquesta fins i tot pot ser una clau primària d'una altra taula. No obstant això els atributs d'una clau forana no necessàriament han de formar part de la clau primària de la taula a la qual pertanyen. Tampoc no és obligatòria l'existència d'aquestes claus, o sigui pot existir una taula sense una clau forana.

1.5.3 Generalització i Agregació

La ***generalització*** és un mecanisme d'abstracció que proporciona l'especialització d'una entitat. Aquesta entitat s'anomenarà **supertipus** i estarà formada per **subtipus**.

Si ens trovem amb un seguit d'atributs comuns a una serie d'entitats i uns altres atributs específics d'aquestes entitats, tindrem una generalització.

El supertipus està definit per els *atributs comuns*; i els subtipus per els *atributs particulars*.

Com a característica principal, la Generalització té herència, això significa que els atributs d'un supertipus son heretats pel subtipus.

Per estudiar els tipus de Generalització imaginem un supertipus: empleat. I tres subtipus: administratius, ingeniers i tècnics.

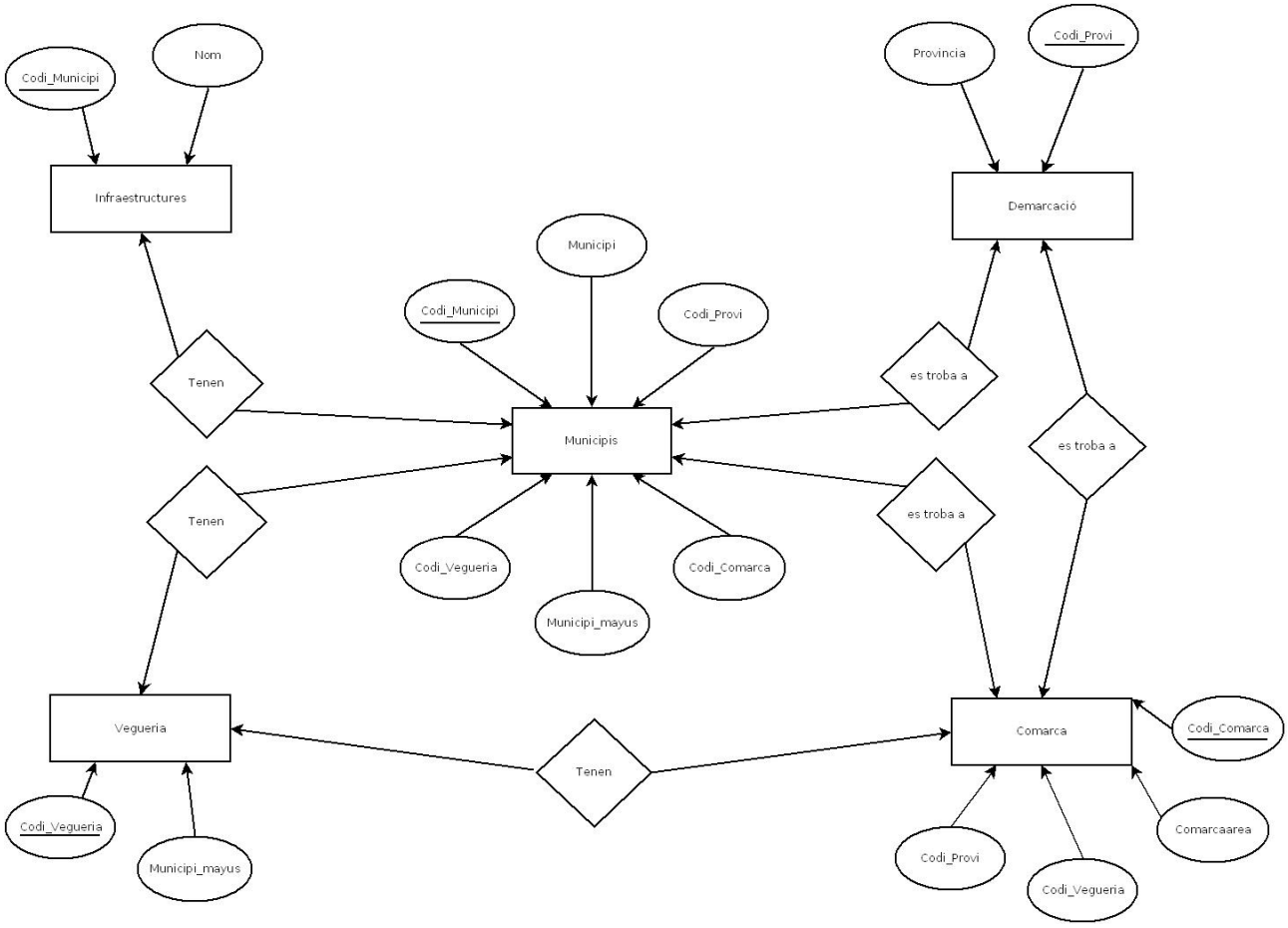
Aleshores tindrem:

- **Generalització Exclusiva**: Si volem referir-nos a que un empleat només pugui ser un dels tres subtipus será una generalització exclusiva.
- **Generalització Solapada o superposada**: Si per contra, els empleats poden tenir més d'un subtipus, parlarem de generalització solapada.

A més també podem parlar de *Generalització Total*: quan no hi hagi ocurrències al supertipus que no pertanyen a cap subtipus, i *Parcial* en cas contrari, per exemple si hi hagués un empleat que no formés part de cap subtipus.

Per altra banda, l'***agregació*** és l'agrupació de dos o més conjunts d'entitats relacionats per a conformar un sol conjunt lògic d'entitats. L'objectiu primordial en l'agregació serà establir relacions entre conjunts d'entitats agrupades.

1.5.4 Exemple del model entitat-relació



1.6 Model de xarxa

Una base de dades de xarxa és una base de dades conformada per una col.lecció de registres, els quals estan connectats entre si mitjançant enllaços en una xarxa. El registre és similar al d'una entitat com les emprades en el model relacional.

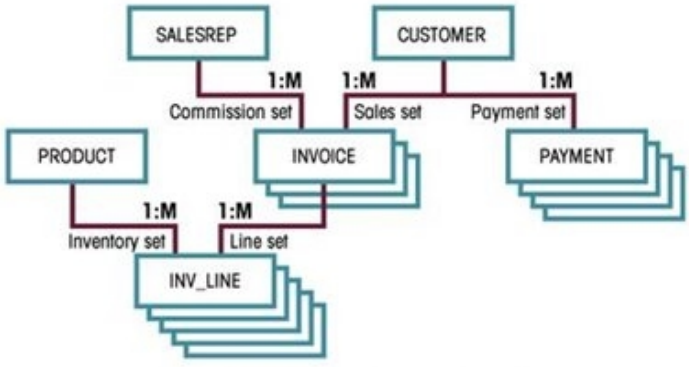
Un registre és una col lecció o conjunt de camps (atributs), on cada un dels conté només un únic valor emmagatzemat, exclusivament l'enllaç és l'associació entre dos registres, així que podem veure-la com una relació estrictament binària.

Una estructura de base de dades de xarxa, anomenada de vegades *estructura de plex*, abasta més que l'estructura d'arbre, perquè un node fill en l'estructura xarxa pot tenir més d'un node pare. En altres paraules, la restricció que en un arbre jeràrquic cada fill pot tenir només un pare, es fa menys severa.

Així, l'estructura d'arbre es pot considerar com un cas especial de l'estructura de xarxa.

El model de xarxa té les seves pròpies denominacions comparades amb el model entitat-relació:

- **Element:** Camp
- **Enllaços:** Relacions
- **Tipus de registre:** Estructura d'una taula (registres formats per 1 o més tipus de propietari)
- **Conjunt:** 1 o més tipus de registre.
- **Cicle:** Es forma quan un registre membre té com a descendent algun dels seus avantpassats
- **Bucle, lazo o loop:** És un cicle on els registres propietaris i membres son els mateixos



1.6.1 Model de dades CODASYL

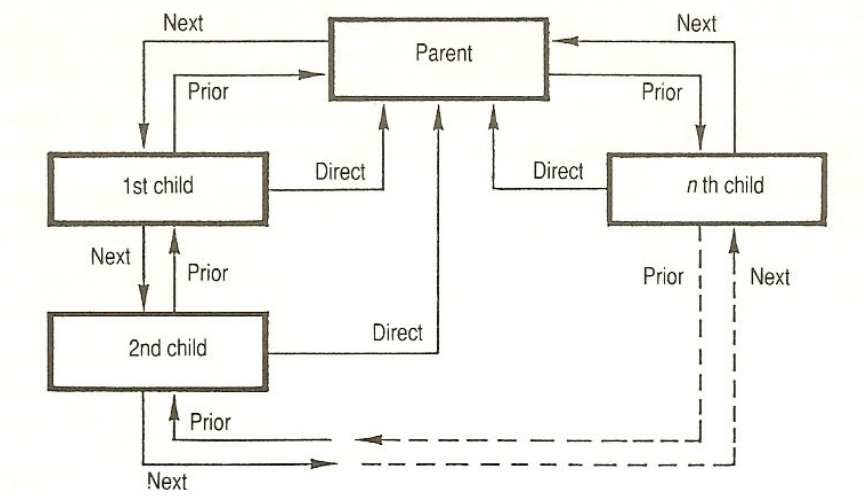
El model CODASYL és un model de dades de tipus xarxa que introdueix restriccions inherents. aquest model constitueix una simplificació del model en xarxa general, en què s'admeten només determinats tipus d'interrelacions i s'inclouen algunes limitacions, que, no obstant, no limiten excessivament la flexibilitat que proporciona el model en xarxa, però si que facilita una instrumentació eficient.

Té els següents objectius:

- **Flexibilitat per als usuaris:** Permetre l'estructuració de les dades de la forma més adaptada a cada aplicació, independentment del fet que tots o part d'aquestes dades poguessin utilitzar-se en altres aplicacions
- **Ús concurrent:** Facilitar a diverses aplicacions recuperar o actualitzar a la vegada les dades de la base.
- **Estratègies de recerca diverses**: Subministrar i permetre l'ús de diverses estratègies de cerca, tant sobre el conjunt de la base com sobre una part d'ella.
- **Seguretat:** Protegir la base de dades d'accessos no autoritzats i d'interaccions indesitjable dels programes.
- **Gestió centralitzada de l'emmagatzematge físic:** Fer els programes independents de l'emmagatzematge físic dels fitxers.
- **Independència de l'emmagatzematge físic:** Fer els programes independents de l'emmagatzematge físic dels fitxers.
- **Flexibilitat en el model de dades:** Permetre l'especificació de diverses estructures de dades, des d'entitats aïllades fins xarxes. El model CODASYL no és excessivament restrictiu pel que fa a les característiques de les entitats i interrelacions que poden ser representades.
- **Facilitat per a l'usuari:** Permetre la interacció de l'usuari amb les dades, però descarregant-ho de les operacions de manteniment de les estructures que han estat declarades.
- **Independència dels programes respecte a les dades:** Aconseguir programes tan independents de les dades com sigui possible amb les tècniques existents.
- **Descripció de dades independent:** Separar la descripció de la base de dades en el conjunt de la de cada programa.
- **Independència respecte als llenguatges:** Dotar de mitjans de descripció de la base de dades que no estiguin restringits a un determinat llenguatge.

1.6 Model de xarxa

- **Interfícies amb múltiples llenguatges:** Proporcionar una arquitectura que permeti que la descripció de la base de dades, així com la manipulació de la mateixa, pugui tenir interfícies amb múltiples llenguatges.



1.7 Model Jeràrquic

En el model Jeràrquic, l'estructura de dades bàsica utilitzada per representar les dades d'aquest model de base de dades és l'arbre.

Els sistemes gestors de bases de dades (SGDB) jeràrquics tenen les següents característiques:

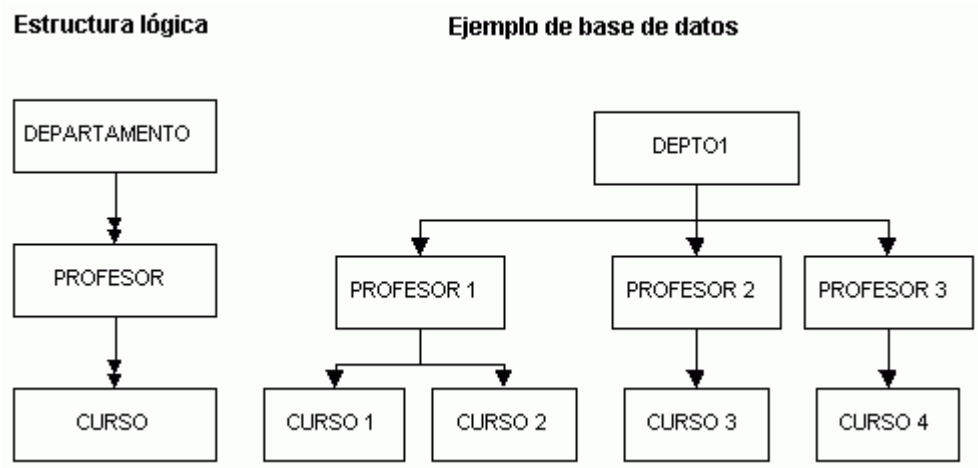
- Els registres estan disposats en forma d'arbre
- Els registres estan relacionats amb relacions d'un a un o un a molts.
- Quan s'elimina un registre pare s'ha d'eliminar tots els registres fills

IMS i el seu llenguatge de consulta DL/I són l'exemple més conegut de base de dades jeràrquica. Aquests sistema s'utilitzava molt en l'inici de la història de les bases de dades.

Aquest tipus de bases de dades van tenir el seu boom als inicis de la història de les bases de dades. Actualment han perdut força envers les bases de dades relacionals tot i que encara s'utilitzen força.

Algunes bases de dades actual també utilitzen estructures de dades jeràrquiques: repositoris Ldap o X.500 (i les eines que els utilitzen com Active Directory o Samba) i bases de dades jeràrquiques i distribuïdes com DNS.

El model jeràrquic i les estructures en arbre actualment s'utilitzen en altres sistemes d'emmagatzemament com dades geogràfiques, sistemes de fitxers, el registre de Windows o els documents XML.



1.8 Model Orientat a Objectes

En una base de dades orientada a objectes, la informació es representa mitjançant objectes com els presents en la programació orientada a objectes. Quan es integra les característiques d'una base de dades amb les d'un llenguatge de programació orientat a objectes, el resultat és un sistema gestor de base de dades orientada a objectes (ODBMS, object database management system).

Un ODBMS fa que els objectes de la base de dades apareixen com a objectes d'un llenguatge de programació en un o més llenguatges de programació als que doni suport. Un ODBMS estén els llenguatges amb dades persistents de forma transparent, control de concurrència, recuperació de dades, consultes associatives i altres capacitats.

Les bases de dades orientades a objectes es dissenyen per a treballar bé en conjunció amb llenguatges de programació orientats a objectes com Java, C #, Visual Basic.NET i C + +. Els ODBMS usen exactament el mateix model que aquests llenguatges de programació.

Els ODBMS són una bona elecció per a aquells sistemes que necessiten un bon rendiment en la manipulació de tipus de dades complexos.

Els ODBMS proporcionen els costos de desenvolupament més baixos i el millor rendiment quan es fan servir objectes gràcies al fet que emmagatzemen objectes en disc i tenen una integració transparent amb el programa escrit en un llenguatge de programació orientat a objectes, en emmagatzemar exactament el model d'objecte utilitzat a nivell aplicatiu, el que redueix els costos de desenvolupament i manteniment.

1.8.1 Estructura

Un objecte està format per:

- **Un conjunt de propietats o atributs** que contenen les dades de l'objecte, serien els atributs del model E-R.
- **Un conjunt de missatges** als que l'objecte respon.
- **Un conjunt de mètodes**, que és codi per implementar els missatges. Un mètode retorna un valor com a resposta al missatge.

1.8.2 Jerarquia de classes

En programació orientada a objectes, una classe és una construcció d'un llenguatge de programació usada per agrupar camps relacionats i mètodes. Està conceptualment relacionat amb classes de la teoria de conjunts.

Una classe és un paquet cohesionat que consisteix en un tipus concret de metadada. Descriu les regles que marquen el comportament de l'objecte; les referències a aquests objectes són fetes com a instànciesd'una classe. Una classe té tant una interfície com una estructura. La interfície descriu com es pot interactuar amb la classe i la instància a través de mètodes, mentre l'estructura descriu com es particiona la dada en camps dintre de la instància. Una classe és el tipus més específic d'un objecte en relació a una capa en concret. Una classe també pot tenir una representació (metaobjecte) en temps d'execució, que proveeix suport per manipular la metadada relativa a la classe.

Els llenguatges de programació que donen suport a les classes tenen diferències subtils en com donen suport a les diferents funcionalitats pròpies de les classes. La majoria donen suport a diferents formes d'herència. Molts llenguatges també donen suport a les funcionalitats que aporta l'encapsulació, per exemple, els especificadors d'accés.

1.8.3 Herència

En l'orientació a objectes, l'herència és una forma de crear classes noves (les instàncies de les quals s'anomenen objectes) fent servir classes que ja s'han definit abans. Les primeres, que es coneixen com classes derivades, prenen o hereten els atributs i comportament de les darreres, que es coneixen com a classes base. Gràcies a això, es permet reutilitzar un codi ja existent minimitzant-ne les modificacions necessàries.

Hom es refereix a una herència també com una generalització, perquè s'hi captura una relació jeràrquica entre les classes d'objectes. Per exemple, una «fruita» és una generalització d'una «poma« o una «taronja». Per això, es diu que una fruita és una abstracció d'una poma o una taronja; i per tant, que les darreres hereten les Propietats comunes d'una fruita.

Un exemple interessant es «vehicle». Podem deduïr que tenim un camió, un cotxe o un tractor.

Un dels avantatges més interessants de l'herència és que diferents mòduls amb interfícies similars poden controlar-se amb un mateix codi compartit, reduint així la complexitat del programa.

1.8.4 Polimorfisme

En programació orientada a objectes s'anomena polimorfisme a la capacitat que tenen els objectes d'una classe de respondre al mateix missatge o esdeveniment en funció dels paràmetres utilitzats durant la seva invocació. Un objecte polimòrfic és una entitat que pot contenir valors de diferents tipus durant l'execució del programa.

Dit d'una altra manera, el polimorfisme consisteix en aconseguir que un objecte d'una classe es comporti com un objecte de qualsevol de les seves subclasses, depenent de la forma de cridar als mètodes d'aquesta classe o subclasses. Una manera d'aconseguir objectes polimòrfics és mitjançant l'ús de punters a la superclasse. D'aquesta manera podem tenir dins d'una mateixa estructura (arrays, llistes, piles, cues, ...) objectes de diferents subclasses, fent que el tipus base d'aquestes estructures sigui un punter a la superclasse.

Es pot classificar el polimorfisme en dues grans classes:

Polimorfisme dinàmic (o polimorfisme paramètric) és aquell en què el codi no inclou cap tipus d'especificació sobre el tipus de dades sobre el qual es treballa. Així, pot ser utilitzat a tot tipus de dades compatible.

Polimorfisme estàtic (o polimorfisme ad hoc) és aquell en el que els tipus a què s'aplica el polimorfisme han de ser explicitats i declarats un per un abans de poder ser utilitzats.

El polimorfisme dinàmic unit a l'herència és el que de vegades es coneix com a programació genèrica.

També es classifica en herència per redefinició de mètodes abstractes i per mètode sobrecarregat. El segon fa referència al mateix mètode amb diferents paràmetres.

Una altra classificació agrupa els polimorfisme en dos tipus: Ad-Hoc que inclou al seu torn sobrecàrrega d'operadors i coerció, Universal (inclusió o controlat per l'herència, paramètric o genericitat).

1.8.5 Llenguatge Unificat de Modelat (UML)

L'UML (Unified Modeling Language, Llenguatge de Modelat Unificat) és un llenguatge per especificar, dissenyar, construir i documentar sistemes, inicialment de programari orientat a objectes.

L'UML intenta definir un llenguatge homogeni (un llenguatge gràfic) per modelar totes les fases del desenvolupament d'una aplicació, des de les especificacions del programa per part del client fins al disseny detallat per al programador. L'UML és un "llenguatge" per especificar, i no un mètode o un procés. Ofereix un estàndard per descriure un "pla" del sistema (model), incloent aspectes conceptuals com ara processos de negocis i funcions del sistema, i aspectes concrets com expressions de llenguatges de programació, esquemes de bases de dades i components de programari reutilitzables. L'UML es pot usar en una gran varietat de formes per suportar una metodologia de desenvolupament de programari (tal com el Procés Unificat de Rational), però no especifica en si mateix quina metodologia o procés usar.

Va ser ideat, segons la definició de Rational, per a especificar, construir, visualitzar i documentar els diversos aspectes d'un sistema basat en programari. Tot i aquests inicis, després s'ha aplicat a altres àmbits que no són l'informàtica com és l'economia per definir processos dins d'un negoci. Té l'avantatge de ser un llenguatge gràfic, i fàcil d'entendre. És el llenguatge d'aquest tipus més conegut i utilitzat en l'actualitat, tot i que encara no és un estàndard oficial, està recolzat en gran manera pel OMG (Object Management Group).

Característiques:

- **Generalització:** La generalització s'utilitza per modelar l'herència en els llenguatges orientats a objectes. Una de les característiques de l'herència és que permet simplificar la construcció de classes relacionades
- **Associació:** representen les relacions entre instàncies de classe. Es representa gràfica-ficament com un línia que connecta les classes relacionades.
    - **Nom:** un nom descriptiu que indica la naturalesa de l'associació.
    - **Rol:** per indicar el rol que desenvolupa cada una de les classes en l'associació. Es s'identifica per un nom als finals de la línia, descriu la semàntica de la relació ció en el sentit indicat.
    - **Multiplicitat:** descriu la cardinalitat de la relació, quants objectes d'una classe participen en la relació.
- **Realització:** és una relació pactada i establerta entre classes, en la qual una classe especifica un contracte (per exemple, un interfície) que l'altra garanteix que complirà. Per indicar que una classe realitza una interfície, es mostra una línia discontínua acabada en fletxa buida que va de la classe que implementa la interfície a la interfície.

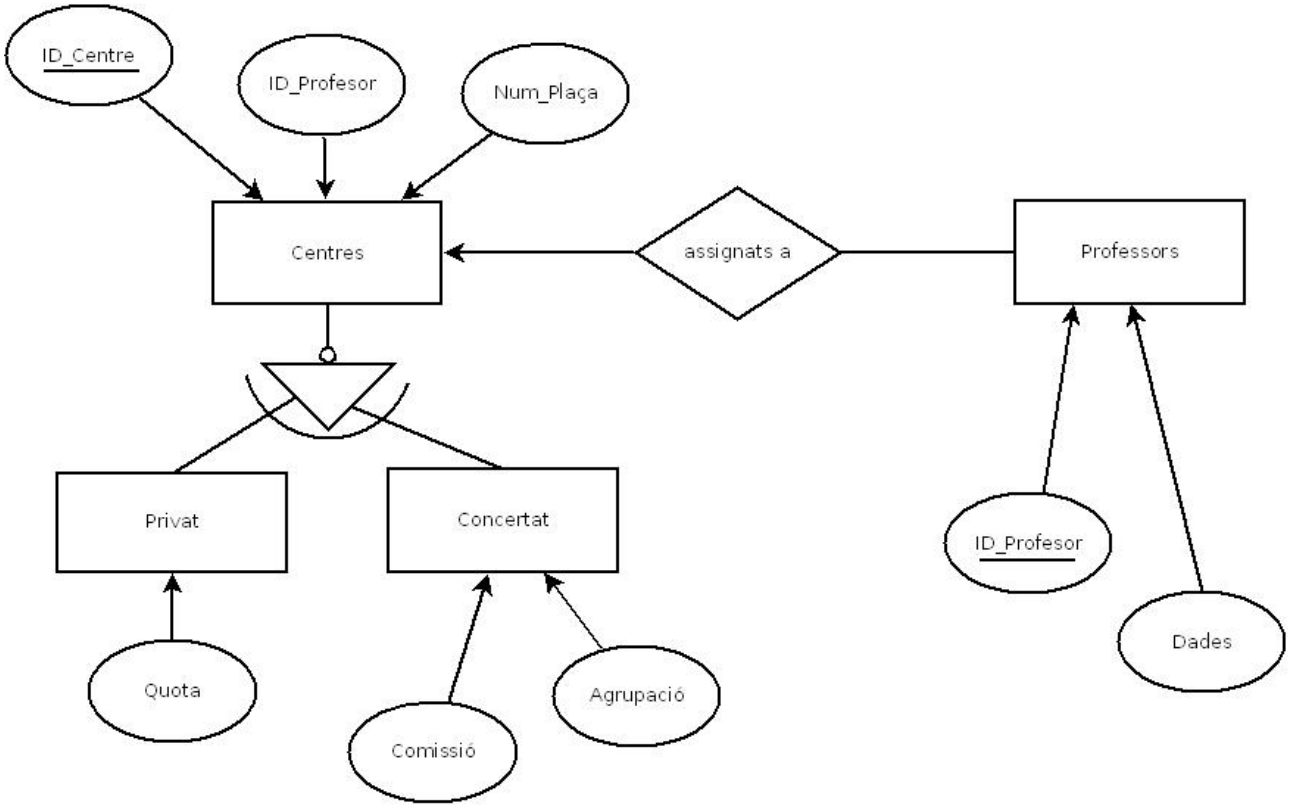
- **Dependència:** és una relació entre entitats independents i dependents. L'element dependent requereix conèixer a l'element independent i que estigui present. Les dependències es fan servir per modelar relacions en les quals un canvi en l'element independent pot requerir canvis en l'element dependent.

1.9 Activitats Complementàries

1. Realitza el diagrama de dades en el model ER, que representi el següent enunciat:

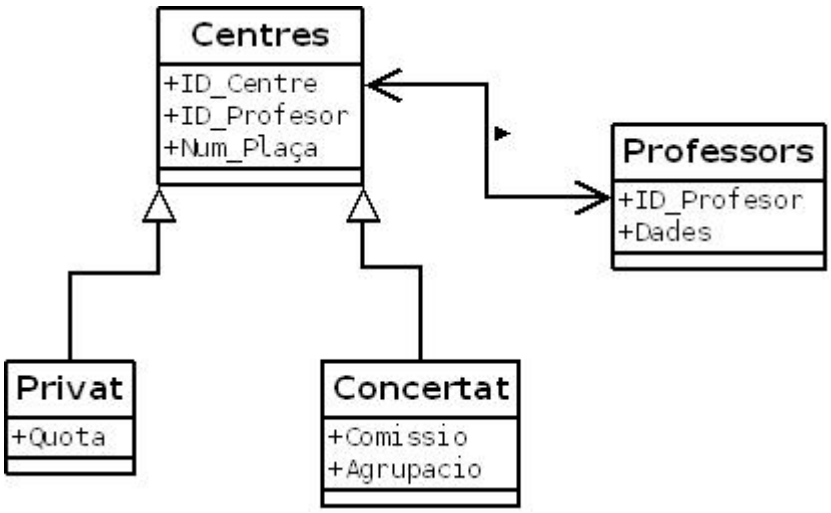
La Conselleria d'Educació gestiona diversos tipus de centres: públics, privats i concertats. Els privats tenen un atribut específic que és la quota i els concertats l'agrupació i la comissió. També assigna places als professors de la comunitat per impartir classe en aquests centres. Un professor pot impartir classe en diversos centres.

Defineix les entitats, els possibles atributs; identifica els atributs clau, i les dades importants per a la relació entre les entitats. Indica també la cardinalitat.

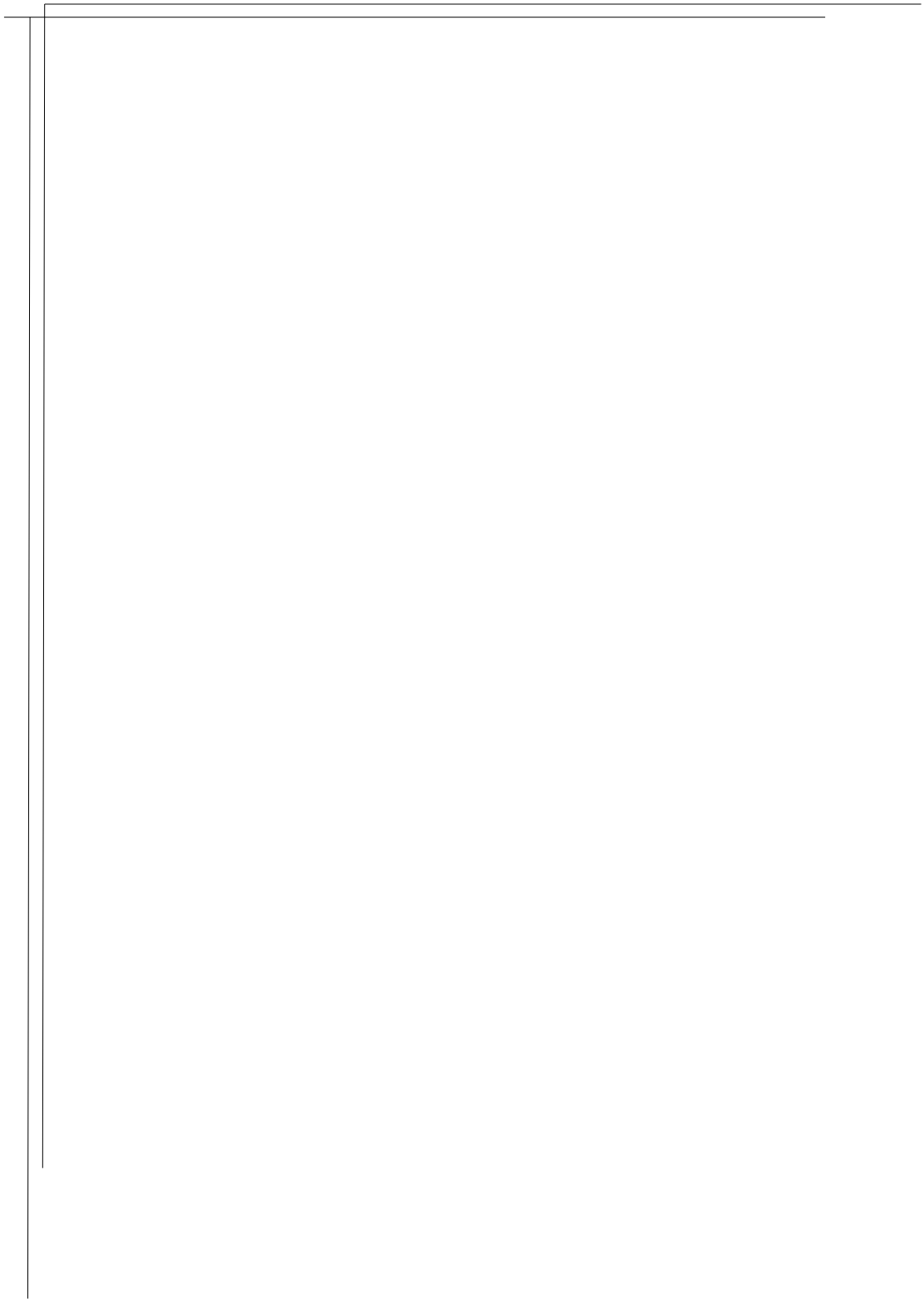


2. Representa l'exercici anterior en el model OO:

Indica les possibles operacions. Considera les associacions navegables en ambdós sentits. Indica el nom de l'associació, els rols, la multiplicitat i les generalitzacions.



Capítol 2. El Model Relacional



2.1 Introducció

El model relacional per a la gestió d'una base de dades és un model de dades basat en la lògica de predicat i en la teoria de conjunts.

És el model **més utilitzat en l'actualitat** per a modelar problemes reals i administrar dades dinàmicament. Després de ser postulades les seves bases el 1970 per **Edgar F. Codd**, dels laboratoris IBM a San José (Califòrnia), no va trigar a consolidar-se com un nou paradigma en els models de base de dades.

La seva idea fonamental és l'**ús de «relacions**». Aquestes relacions podrien considerar-se en forma lògica com a conjunts de dades anomenats «tuples». Tot i que aquesta és la teoria de les bases de dades relacionals creades per Edgar F. Codd, la majoria de les vegades es conceptualitza d'una manera més fàcil d'imaginar, és a dir, pensant en cada relació com si fos una taula que està compostes per registres (cada fila de la taula seria un registre o tupla), i columnes (també anomenades camps).

El model relacional es basa en dues branques de les matemàtiques: **la teoria de conjunts i la lògica de predicats** de primer ordre. El fet que el model relacional estigui basat en la teoria de les matemàtiques és el que el fa tan segur i robust. Al mateix temps, aquestes branques de les matemàtiques proporcionen els elements bàsics necessaris per crear una base de dades relacional amb una bona estructura, i proporcionen les línies que s'utilitzen per formular bones metodologies de disseny.

Hi ha qui ofereix una certa resistència a estudiar complicats conceptes matemàtics per a tan sols dur a terme una tasca molt concreta. És habitual escoltar queixes sobre que les teories matemàtiques en què es basa el model relacional i les seves metodologies de disseny, no tenen rellevància en el món real o que no són pràctiques. No és cert: les matemàtiques són bàsiques en el model relacional.

La teoria matemàtica proporciona la base per al model relacional i, per tant, fa que el model sigui **predictible, fiable i segur**. La teoria descriu els elements bàsics que s'utilitzen per a crear una base de dades relacional i proporciona les línies a seguir per construir-la.

2.2 El Model Relacional

Les 12 regles de Codd són un sistema de regles propostes per Edgar F. Codd, del model relacional per a les bases de dades, dissenyat per definir què requereix un sistema d'administració de base de dades.

Codd es va adonar que existien bases de dades en el mercat les quals deien ser relacionals, però l'únic que feien era guardar la informació a les taules, sense estar aquestes taules literalment normalitzades, aleshores aquest va publicar 12 regles que un veritable sistema relacional hauria de tenir encara que en la pràctica algunes d'elles són difícils de realitzar. Un sistema pot considerar-se "més relacional" com més siga la regla.

2.2.1 Regles

Regla 1: la regla de la informació, tota la informació a la base de dades és representada unidireccionalment, per valors en posicions de les columnes dins de files de taules.

Regla 2: la regla de l'accés garantit, totes les dades han de ser accessibles sense ambigüitat. Aquesta regla és essencialment una nova exposició del requisit fonamental per a les claus primàries. Diu que cada valor escalar individual a la base de dades ha de ser lògicament direccionable especificant el nom de la taula, la columna que el conté i la clau primària.

Regla 3: tractament sistemàtic de valors nuls, el sistema de gestió de base de dades ha de permetre que hi hagi camps nuls. Ha de tenir una representació de la "informació que falta i de la informació inaplicable" que és sistemàtica, diferent de tots els valors regulars.

Regla 4: catàleg dinàmic en línia basat en el model relacional, el sistema ha de suportar un catàleg en línia, el catàleg relacional ha de ser accessible als usuaris autoritzats. És a dir, els usuaris han de poder tenir accés a l'estructura de la base de dades (catàleg).

Regla 5: la regla comprensiva del subllenguatge de les dades, el sistema ha de suportar com a mínim un llenguatge relacional que:

1. Tingui una sintaxi lineal.
2. Pugui ser utilitzat recíprocament i dins de programes d'ús.
3. Suporti operacions de definició de dades, operacions de manipulació de dades (actualització així com la recuperació), seguretat i integritat i operacions d'administració de transaccions.

Regla 6: regla d'actualització, totes les vistes que són teòricament actualitzables han de ser actualitzables pel sistema.

Regla 7: alt nivell d'inserció, actualització i cancel.lació, el sistema ha de suportar subministrar dades al mateix temps que s'insereixi, actualitza o aquest esborrant. Això significa que les dades es poden recuperar

d'una base de dades relacional en els sistemes construïts de dades de files múltiples i / o de taules múltiples.

Regla 8: independència de dades físic, els canvis en el nivell físic (com s'emmagatzemen les dades, si en arranjaments o en les llistes encadenades els etc.) No ha de requerir un canvi a una sol licitud basada en l'estructura.

Regla 9: independència de dades lògica, els canvis al nivell lògic (taules, columnes, files, etc) no han de requerir un canvi a una sol licitud basada en l'estructura. La independència de dades lògica és més difícil d'aconseguir que la independència física de dades.

Regla 10: independència de la integritat, les limitacions de la integritat s'han d'especificar per separat dels programes de l'aplicació i s'emmagatzemen a la base de dades. Ha de ser possible canviar aquestes limitacions sense afectar innecessàriament les aplicacions existents.

Regla 11: independència de la distribució, la distribució de les porcions de la base de dades a les diverses localitzacions ha de ser invisible als usuaris de la base de dades. Els usos existents han de continuar funcionant amb èxit:

1. quan una versió distribuïda de l'SGBD es va introduir per primera vegada
2. quan es distribueixen les dades existents es redistribueixen en tot el sistema.

Regla 12: la regla de la no subversió, si el sistema proporciona una interfície de baix nivell (de registre a la vegada) i després que aquesta interfície no es pugui utilitzar per subvertir el sistema, per exemple: sense passar per seguretat relacional o limitació d'integritat.

2.3 Estructura del Model Relacional

La relació és l'element bàsic en el model relacional i es pot representar com una taula:

Nom

Atribut 1	Atribut 2	.....	Atribut n	
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	**Tupla 1**
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	**Tupla 2**
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	**.**
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	**.**
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	**Tupla n**

Hi podem distingir un conjunt de columnes, anomenades **atributs**, que representen propietats de la mateixa i que es caracteritzen per un nom, i un conjunt de files anomenades **tuples** que són les ocurrències de la relació. Hi ha també uns **dominis** on els atributs prenen els seus valors.

El nombre de files d'una relació s'anomena **cardinalitat** de la relació i el nombre de columnes és el grau de la **relació**.

Exemple: AUTOR

Nom	Nacionalitat	Institució
Pepe	España	O.N.U.
John	EE.UU.	O.M.S.
Pierre	Francia	N.A.S.A.

Una relació es pot representar en forma de taula, però tindrà una sèrie d'elements característics:

- No hi pot haver files duplicades, és a dir, totes les tuples han de ser diferents.
- L'ordre de les files és irrellevant.
- La taula és plana, és a dir, a la cruïlla d'una fila i una columna només hi pot haver un valor (no s'admeten atributs multivaluats).

2.3.1 Dominis i Atributs

Un D (**domini**) és un conjunt finit de valors homogenis i atòmics caracteritzats per un nom; diem homogenis perquè són tots del mateix tipus i atòmics perquè són indivisibles.

Tot domini ha de tenir un nom pel qual ens podem referir a ell i un tipus de dades, així el tipus de dades del domini "nacionalitats" és una tira de caràcters de longitud 10.

El domini "nacionalitats" té valors: Espanya, França

Exemples de dominis serien:

- Colors: És el conjunt dels colors D = (vermell, verd, blau,)
- Números de DNI: És conjunt de nombres del DNI vàlids, formats per vuit dígits.
- Edat: Edats possibles dels treballadors entre 18 i 80 anys.

Un **atribut** és el paper que té un determinat domini en una relació.

És molt usual donar el mateix nom a l'atribut i al domini. En el cas que siguin diversos els atributs d'una mateixa taula definits sobre el mateix domini, caldrà donar-los noms diferents, ja que una taula no pot tenir dos atributs amb el mateix nom.

Per exemple els atributs edat_física i edat_mental poden estar definits sobre el mateix domini edat, o els atributs preu_compra i preu_venta poden estar definits sobre el mateix domini d'enters de longitud 5.

Un **domini compost** es pot definir com una combinació de dominis simples que té un nom ia la qual es poden aplicar certes restriccions d'integritat. Per exemple, un usuari pot necessitar utilitzar, a més dels tres dominis Dia, Mes i Any, un domini compost anomenat Data que seria la combinació dels tres primers, i al que podríem aplicar les adequades restriccions d'integritat per tal que no apareguessin valors no vàlids per a la data, quelcom semblant passa amb el nom i els cognoms, que, segons les aplicacions, pot ser convenient tractar-los en conjunt o per separat.

2.3.2 Relació

Matemàticament, una **relació** es pot definir com un subconjunt del producte cartesià d'una llista de dominis, on cada element de la relació, tupla, és una sèrie de n valors ordenats.

En aquesta definició matemàtica de relació, que és la que apareix en els primers treballs de Codd, no es fa referència als atributs, és a dir, al paper que tenen els dominis en la relació i, a més, en ella l'ordre dels valors dins o d'una tupla és significatiu. Per tal d'evitar aquests inconvenients, es pot donar una altra definició de relació més adequada al punt de vista de les bases de dades, per a això cal distingir, dos conceptes en la noció de relació:

- **Intensió** o Esquema de relació, denotat R (Al: D1, A2: D2, ..., An: Dn) és un conjunt de n parells atribut-domini subjacent (Ai: Di). La intensió és la part definitòria i estàtica de la relació, que es correspon amb la capçalera quan la relació es percep com una taula.
- **Extensió** o ocurrència (instància) de relació (anomenada de vegades simplement relació), denotada per r (R) és un conjunt de m tuples (t1, t2, ... tm) on cada tupla és un conjunt de n parells atribut-valor.

Per exemple:

1. *Intensió d'una relació:*

   AUTOR (NOM:Noms, NACIONALITAT:Nacionalitats, INSTITUCIO: Institucions)

2. Extensió d'una relació

   AUTOR

Nom	Nacionalitat	Institució
Pepe	España	O.N.U.
John	EE.UU.	O.M.S.
Pierre	Francia	N.A.S.A.

2.3.3 Claus

Una **clau candidata** d'una relació és un conjunt no buit d'atributs que identifiquen unívoca i mínimament cada tupla. Per la pròpia definició de relació, sempre hi ha almenys una clau candidata, ja que en ser la relació un conjunt no existeixen tuples repetides i per tant, el conjunt de tots els atributs identificarà unívocament a les tuples.

Una relació pot tenir més d'una clau candidata, entre les quals cal distingir:

- **Clau primària**: és la clau candidata que l'usuari escollirà, per consideracions alienes al model relacional, per identificar les tuples d'una relació.
- **Clau alternativa**: són aquelles claus candidates que no han estat seleccionades.

Per altra banda, s'anomena **clau aliena** d'una relació R2 a un conjunt no buit d'atributs els valors han de coincidir amb els valors de la clau primària d'una altra relació R1. La clau aliena i la corresponent clau primària han d'estar definides sobre els mateixos dominis.

2.3.4 Tipus d'Operacions

A més de definir les claus alienes, cal determinar les conseqüències que poden tenir certes operacions (esborrat i modificació) realitzades sobre tuples de la relació esmentada; es poden distingir, en principi, les següents opcions:

- **Operació restringida:** això és, l'esborrat o la modificació de tuples de la relació que conté la clau primària referenciada: només es permet si no hi ha tuples amb aquesta clau en la relació que conté la clau aliena. Això ens portaria, per exemple, que per poder esborrar una editorial de la nostra base de dades no hauria d'haver cap llibre que estigués publicat per aquesta editorial, en cas contrari el sistema impediria l'esborrat.
- **Operació amb transmissió en cascada:** és a dir, l'esborrat o la modificació de tuples de la relació que conté la clau primària referenciada comporta l'esborrat o modificació en cascada de les tuples de la relació que contenen la clau aliena. En el nostre exemple, equivaldria a dir que en modificar el nom d'una editorial en la relació EDITORIAL, s'hauria de modificar aquest nom en tots els llibres de la nostra base de dades publicades per aquesta editorial.
- **Operació amb posada a nuls:** això és, l'esborrat o la modificació de tuples de la relació que conté la clau primària referenciada comporta posar a nuls els valors de les claus alienes de la relació que referència. Això ens portaria a que quan s'esborra una editorial, als llibres que ha publicat aquesta editorial i que es troben en la relació LLIBRES se'ls col.loqui l'atribut *nom_e* a nuls. Aquesta opció, òbviament, només és possible quan l'atribut que és clau aliena admet el valor nul.
- **Operació amb posada a valor per defecte:** és a dir, l'esborrat o la modificació de tuples de la relació que conté la clau primària referenciada comporta posar el valor per defecte a la clau aliena de la relació que referència.
- **Operació que desencadena un procediment d'usuari:** en aquest cas, l'esborrat o la modificació de tuples de la taula referenciada posa en marxa un procediment definit per l'usuari.

2.4 Restriccions del Model Relacional

En el model relacional, hi ha restriccions, és a dir, estructures o ocurrències no permeses, i cal distingir entre restriccions inherents i restriccions d'usuari.

2.4.1 Restriccions Inherents

A més de les derivades de la definició matemàtica de "relació" com eren que:

- No hi ha dos tuples iguals.
- L'ordre de les tuples no és significatiu.
- L'ordre dels atributs (columnes) no és significatiu.
- Cada atribut només pot prendre un únic valor del domini, no admetent per tant els grups repetitius.

2.4.2 Restriccions Semàntiques o d'usuari

Podem considerar la restricció d'usuari, dins del context relacional, com un predicat definit sobre un conjunt d'atributs, de tuples o de dominis, que ha de ser verificat pels corresponents objectes per tal que aquests constitueixin una ocurrència vàlida de l'esquema.

- **La clau primària (PRIMARY KEY):** Una taula sol tenir una columna o una combinació de columnes els valors identifiquen de forma única cada fila de la taula, s'anomenen claus principals.
- **Unicitat (UNIQUE):** Han de ser úniques.
- **Obligatorietat (Not NULL):** Un camp que no pot ser buit.
- **Integritat referencial (FOREIGN KEY):** és una restricció de comportament ja que ve imposada pel món real i és l'usuari qui la defineix en descriure l'esquema relacional, és també de tipus implícit, ja que es defineix en l'esquema i el model la reconeix (o així alguns productes) sense necessitat que es programi ni que s'hagi de escriure cap procediment per a obligar a que es compleixi.
- **Verificació (CHECK):** assegura que els valors que es van a emmagatzemar correctament.
- **Asercions (ASERTION):** Una asserció és una restricció general que fa referència a una o més columnes de més d'una taula
- **Disparadors (TRIGGER):** s'activen automàticament al passar l'esdeveniment (INSERT, UPDATE o DELETE) sobre la taula i executa accions definides en cas de complir les condicions especificades

2.5 Transformació E-R → Relacional

2.5.1 Regles

1. Totes les **entitats** del model entitat-relació es transformen en **taules**
2. Els **atributs** d'una entitat es transformen en **camps** dins de la taula
3. Els **identificadors únics** es transformen en **claus primàries**
4. Les **Relacions N:M** es transformen en una taula que tindrà com a clau primària, la concatenació dels atributs clau de les entitats que relaciona.
5. Les **Relacions 1:N :**
   - **Propagar la clau + Atributs:** Si l'entitat que participa amb cardinalitat màxima *un* ho fa també amb cardinalitat mínima *un*, llavors és propaga l'atribut de l'entitat que te cardinalitat màxima 1 a la qual te cardinalitat màxima N, desapareixent el nom de la relació. Si hi ha atributs a la relació aquests també és propaguessin.
   - **Transforma la Relacio en una taula + Atributs:** Si l'entitat que participa amb cardinalidad màxima un ho fa també cardinalidad mínima zero, llavors es crea una nova taula formada per les claus de cada entitat i els atributs de la relació. La clau primària de la nova taula serà l'identificador de l'entitat que participa amb cardinalidad màxima N.
6. Les **Relacions 1:1 :**
   - **Propagar la clau + Atributs:** Si una de les entitats posseïx cardinalidad (0,1) i l'altra (1,1), convé propagar la clau de l'entitat amb cardinalidad (1,1) a la taula resultant de l'entitat amb cardinalidad (0,1). Si ambdues entitats posseïxen cardinalidades (1,1) es pot propagar la clau de qualsevol d'elles a la taula resultant de l'altra.

     Transforma la Relacio en una taula + Atributs: Si les entitats posseïxen cardinalidades (0,1), la relació es converteix en una taula.
7. Les **Relacions Reflexives** suposarem que es tracta d'una relació binària amb la particularitat que les dues entitats són iguals i aplicarem les regles vistes en els punts anteriors.
8. Les **Generalitzacions:**
   - **Integració:** De les taules filles a taula pare, l'inconvenient es la redundància de dades, ja que queden molts valors nuls.
   - **El.liminació del supertipus:** De la taula pare a les taules filles, tindrem l'inconvenient de

redundància de la informació i la multiplicació de de les relacions .

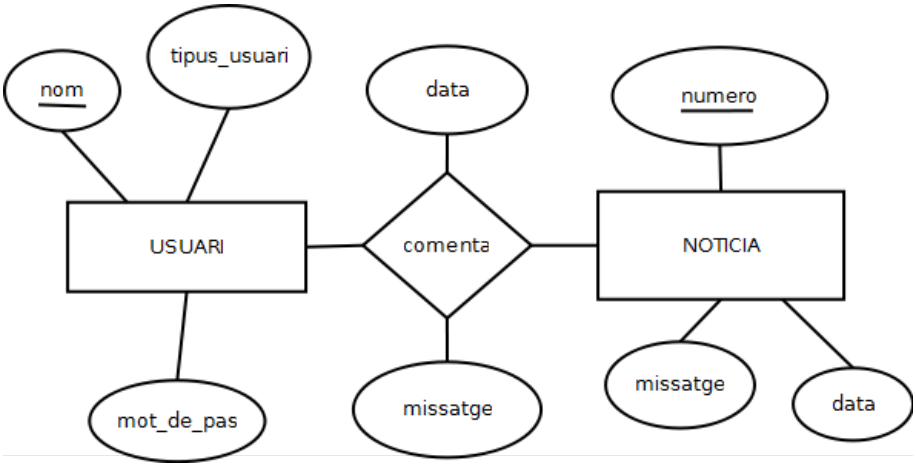
- **Inserció de Relació:** Relacionar les diferents taules amb la relació explícita 1:1.

9. En les **Relacions N-àries** s'aplica la mateixa regla que per a les relacions N: M

2.5.2 Exemples de transformació

Exemple 1: Relació N:M

Un usuari comenta una notícia

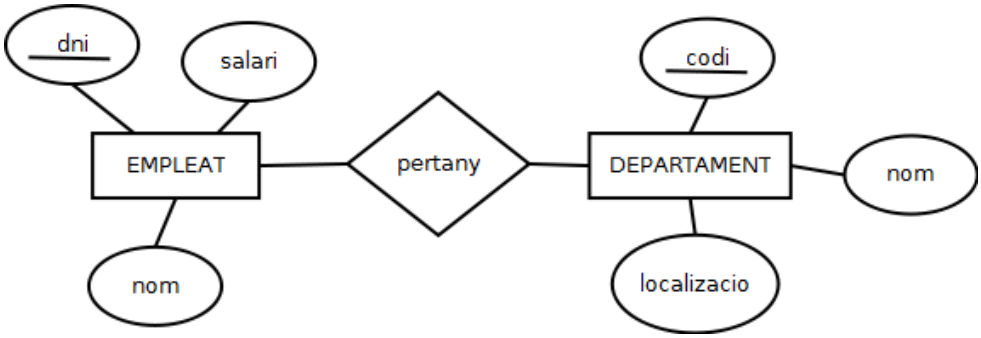


El model relacional quedaria de la següent forma (en negreta les claus primàries) :

- USUARI (**nom**, mot_de_pas, tipus_d'usuari)
- NOTICIA (**numero,** data, missatge)
- COMENTARI (**nom_usuari, numero_noticia**, data, missatge)

Exemple 2: Relació 1:N → Propagació

Un empleat pertany a un departament obligatòriament

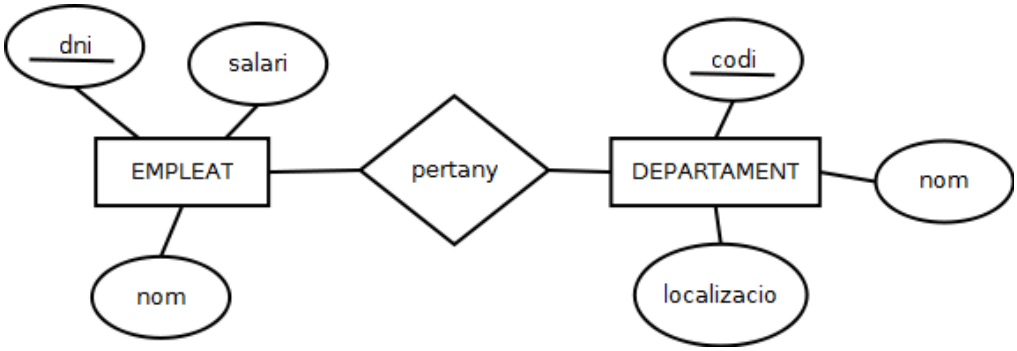


En el model relacional *es propagaria l'atribut codi de departament* i quedaria de la següent forma (en negreta les claus primàries) :

- EMPLEAT (**dni**, nom, salari, codi_departament)
- DEPARTAMENT (**codi,** nom, localització)

Exemple 3: Relació 1:N → Taula

El mateix cas anterior, però aquí un empleat pot no pertànyer a un departament

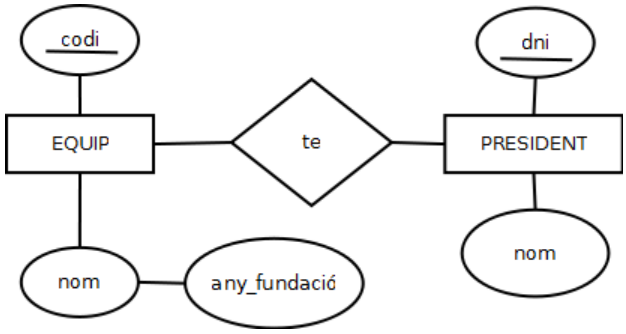


En el model relacional *es crearia una nova taula formada pel dni_empleat i codi_departament*, quedaria de la següent forma (en negreta les claus primàries) :

- EMPLEAT (**dni**, nom, salari)
- DEPARTAMENT (**codi,** nom, localitzacio)
- COMENTARI (**dni_empleat,** codi_departament)

Exemple 4: Relació 1:1 → Propagació

Cada Equip te un President, o movem la clau de PRESIDENT a EQUIP o movem la clau d'EQUIP a PRESIDENT



Model relacional, opció 1: *movem la clau de PRESIDENT a EQUIP*:

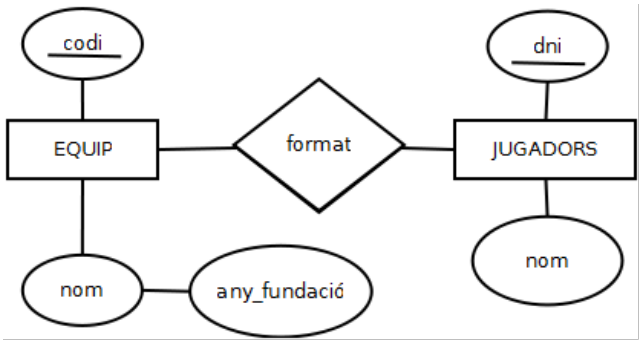
- EQUIP (**codi**, nom, any_fundacio)
- PRESIDENT (**dni,** nom,codi_equip)

Model relacional, opció 2: *movem la clau d'EQUIP a PRESIDENT*:

- EQUIP (**codi**, nom, any_fundacio,dni_president)
- PRESIDENT (**dni,** nom)

Exemple 5: Relació 1:1 → Taula

Cada Equip està format per diversos Jugadors, la relació es converteix en taula



En el model relacional *es crearia una nova taula per la relacio "format"*, i quedaria de la següent forma (en negreta les claus primàries) :

- EQUIP (**codi**, nom, any_fundació)
- JUGADORS (**dni**, nom)
- FORMANTS (**codi_equip, dni_jugador**)

2.6 Pèrdua semàntica transf. ER → R

La transformació del model entitat relació a l'esquema relacional implica una pèrdua de semàntica degut a l'ús de disparadors, procediments emmagatzemats i aplicacions externes.

Una vegada obtingut l'esquema relacional resultant del model entitat relació que representava la base de dades, normalment tindrem una bona base de dades. Però altres vegades, a causa de fallades en el disseny o a problemes indetectables en aquesta fase del disseny, tindrem un esquema que pot produir una base de dades que incorpori aquests problemes:

- **Redundància:** Es diu així a les dades que es repeteixen contínua i innecessàriament per les taules de les bases de dades.
- **Ambigüitats:** Dades que no clarifiquen suficientment el registre al que representen.
- Pèrdua de restriccions d'integritat.
- Anomalies en operacions de modificació de dades.

El fet que a l'inserir un sol element calgui repetir tuplas en una taula per a variar unes poques dades. O que eliminar un element suposi eliminar diverses tuplas. El principi fonamental resideix que les taules han de referir-se a objectes o situacions molt concretes. El que ocorre és que conceptualment és difícil obtenir aquest problema. La solució sol ser dividir la taula amb problemes en altres taules més adequada ,

2.7 Normalització d'esquemes R

2.7.1 Dependencia funcional

Una dependència funcional és una connexió entre un o més atributs. Per exemple, si coneixem el valor de DataNaixament podem conèixer el valor d'Edat. Les dependències funcionals del sistema s'escriuen utilitzant una fletxa, de la següent manera: DataNaixament → Edat Aquí a DataNaixament se'l coneix com un determinant. És pot llegir de dues maneres DataNaixamen determinat edat o Edat és funcionalment dependent de DataNaixamen.

De la normalització (lògica) a la implementació (física o real) pot ser suggerir tenir aquestes dependències funcionals per aconseguir l'eficiència en els taules.

Dependències completes i parcials

Un conjunt d'atributs (I) té una dependència funcional completa sobre altre conjunt d'atributs (X) si I té dependència funcional de X i a més no es pot obtenir de X un conjunt d'atributs més petit que aconsegueixi una dependència funcional d'I.

Per exemple en una taula de clients, el conjunt d'atributs format pel nom i el dni produïxen una dependència funcional sobre l'atribut cognoms. Però no és plena ja que el dni només també produïx una dependència funcional sobre cognoms. El dni sí produïx una dependència funcional completa sobre el camp cognoms. Una dependència funcional completa es denota com X=>I

Dependències transitives

És més complexa d'explicar, però té també utilitat. Es produïx quan tenim tres conjunts d'atributs X, I i Z. I depèn funcionalment de X (X->I), Z depèn funcionalment d'I (I->Z). A més X no depèn funcionalment d'I. Llavors ocorre que X produïx una dependència funcional transitiva sobre Z.

Això es denota com: (X-->Z) Per exemple si X és l'atribut Nombre de Classe d'un institut, i I és l'atribut Codi Tutor. Llavors X->I (el tutor depèn funcionalment del nombre de classe). Si Z representa el Codi del departament, llavors I->Z (el codi del departament depèn funcionalment del codi tutor, cada tutor només pot estar en un departament).

Com no ocorre que I->X (el codi de la classe no depèn funcionalment del codi tutor, un codi tutor es pot correspondre amb diversos codis de classe). Llavors X-->Z (el codi del departament depèn transitivament del codi de la classe).

Dependències multivaluades o de valors múltiples

Una dependència multivaluada és una sentència que s'escriu : X → I i el significat intuïtiu de la qual és que a

cada valor de X se li associa un conjunt de valors d'I independent del context (si X i I són subconjunts de T , el context és Z=T- (X u I)).

Això vol dir que la dependència multivaluada assigna a cada valor de Dom (X) un valor de P (Dom (I)) , i aquesta assignació no varia amb el context .

És molt important no confondre l'existència d'una dependència multivaluada (des d'ara , mvd) amb el fet que entre els dominis de X i I pugui establir-se una correspondència 1 : n , el que és un error bastant freqüent.

Dependències de combinació

Una dependència de combinació és un tipus de dependència en la qual, per exemple, per a una relación R composta per una sèrie de subconjunts dels atributs de R denominats A, B, ..., Z, la relació R exhibirà una dependència de combinació si i només si tot valor legal de R és igual a la combinació de les seves projeccions sobre A, B, ..., Z.

2.7.2 Regles de normalització

Primera forma normal 1FN

Una taula es troba en primera forma normal si impedeix que un atribut d'una tupla pugui prendre més d'un valor. La taula:

TRABAJADOR		
DNI	**Nombre**	**Departamento**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección Gestión

Visualment és un taula, però no una taula relacional (el que en terminologia de bases de dades relacionales es diu relació). No complix la primera forma normal. Ho compliria si:

TRABAJADOR		
DNI	**Nombre**	**Departamento**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección
12345345G	Andrea	Gestión

Segona forma normal 2FN

Ocorre si una taula està en primera forma normal i a més cada atribut que no sigui clau, depèn de forma

funcional completa respecte de qualsevol de les claus. Tota la clau principal ha de fer dependents a la resta d'atributs, si hi ha atributs que depèn només de part de la clau, llavors aquesta part de la clau i aquests atributs formaran altra taula. Exemple:

ALUMNOS				
DNI	**Cod Curso**	**Nombre**	**Apellido1**	**Nota**
12121219A	34	Pedro	Valiente	9
12121219A	25	Pedro	Valiente	8
3457775G	34	Ana	Fernández	6
5674378J	25	Sara	Crespo	7
5674378J	34	Sara	Crespo	6

Suposant que el DNI i el nombre de curs formin una clau principal per a aquesta taula, només la nota té dependència funcional completa. El nom i els cognoms depenen de forma completa del DNI. La taula no és 2FN, per a arreglar-lo:

ALUMNOS		
DNI	**Nombre**	**Apellido1**
12121219A	Pedro	Valiente
3457775G	Ana	Fernández
5674378J	Sara	Crespo

ASISTENCIA		
DNI	**Cod Curso**	**Nota**
12121219A	34	9
12121219A	25	8
3457775G	34	6
5674378J	25	7
5674378J	34	6

Tercera forma normal 3FN

Ocorre quan una taula està en 2FN i a més cap atribut que no sigui clau depèn transitivament de les claus de la taula. És a dir no ocorre quan algun atribut depèn funcionalment d'atributs que no són clau. Exemple:

ALUMNOS				
DNI	**Nombre**	**Apellido1**	**Cod Provincia**	**Provincia**
12121349A	Salvador	Velasco	34	Palencia
12121219A	Pedro	Valiente	34	Palencia
3457775G	Ana	Fernández	47	Valladolid
5674378J	Sara	Crespo	47	Valladolid
3456858S	Marina	Serrat	08	Barcelona

La Província depèn funcionalment del codi de província, el que fa que no estigui en 3FN. L'arranjament seria:

ALUMNOS			
DNI	**Nombre**	**Apellido1**	**Cod Provincia**
12121349A	Salvador	Velasco	34
12121219A	Pedro	Valiente	34
3457775G	Ana	Fernández	47
5674378J	Sara	Crespo	47
3456858S	Marina	Serrat	08

PROVINCIA	
Cod Provincia	**Provincia**
34	Palencia
47	Valladolid
08	Barcelona

Forma normal de Boyle - Codd (FNBC)

Ocorre si una taula està en tercera forma normal i a més tot determinant és una clau candidata. Exemple:

TUTORÍAS		
DNI	**Asignatura**	**Tutor**
12121219A	Lenguaje	Eva
12121219A	Matemáticas	Andrés
3457775G	Lenguaje	Eva
5674378J	Matemáticas	Guillermo
5674378J	Lenguaje	Julia
5634823H	Matemáticas	Guillermo

Aquesta taula està en tercera forma normal (no hi ha dependències transitives), però no en forma de Boyce - Codd, ja que (DNI, Assignatura) ->Tutor i Tutor->Assignatura. En aquest cas la redundància ocorre per dolenta selecció de clau. La redundància de l'assignatura és completament evitable. La solució seria:

TUTORÍAS	
DNI	**Tutor**
12121219A	Eva
12121219A	Andrés
3457775G	Eva
5674378J	Guillermo
5674378J	Julia
5634823H	Guillermo

ASIGNATURASTUTOR	
Asignatura	**Tutor**
Lenguaje	Eva
Matemáticas	Andrés
Matemáticas	Guillermo
Lenguaje	Julia

Cuarta forma normal 4FN

Ocorre aquesta forma normal quan una taula està en forma normal de Boyce Codd i tota dependència multivaluada és una dependència funcional. Tenint la següent taula:

Nº Curso	Profesor	Material
17	Eva	1
17	Eva	2
17	Julia	1
17	Julia	2
25	Eva	1
25	Eva	2
25	Eva	3

La solució serien dues taules:

Nº Curso	Material
17	1
17	2
25	1
25	2
25	3

Nº CursoProfesor	
17	Eva
17	Julia
25	Eva

Cinquena forma normal 5FN

És la més complexa i polèmica de totes. Polèmica doncs no està clar en moltes ocasions que sigui una solució millor que el no arribar a aquest nivell de normalització. Va Ser definida també per Fagin. És rar trobar-se aquest tipus de problemes quan la normalització arriba a 4FN. Es deuen a restriccions molt concretes. Exemple:

Proveedor	Material	Proyecto
1	1	2
1	2	1
2	1	1
1	1	1

Indiquen codis de material subministrat per un proveïdor i utilitzat en un determinat projecte.

Si ocorre una restricció especial com per exemple: Quan un proveïdor ens ha subministrat alguna vegada un determinat material, si aquest material apareix en altre projecte, farem que el proveïdor ens subministri també aquest material per a aquest projecte.

Això ocorre en les dades com el proveïdor nombre 1 ens va subministrar el material nombre 1 per al projecte 2 i en el projecte 1 utilitzem el material 1, apareixerà la tupla proveïdor 1, material 1 i projecte 1. La dependència que produïx aquesta restricció és llunyana i la hi crida de reunió. Per a aquesta restricció aquesta divisió en taules seria vàlida:

Proveedor	Material
1	1
1	2
2	1

MaterialProyecto	
1	2
2	1
1	1

Aquesta descomposició no perd valors en aquest cas, sabent que si el proveïdor ens subministra un material podrem relacionar-li amb tots els projectes que utilitzen aquest material.

Resumint, una taula no està en cinquena forma normal si hi ha una descomposició d'aquesta taula que mostri la mateixa informació que l'original.

Bibliografia

http://www.alegsa.com.ar/Dic/sgbd.php

http://www3.uji.es/~mmarques/f47/apun/node6.html

http://www.mailxmail.com/curso-sistemas-bases-datos/sgbd-inconvenientes-sistema-gestion-archivos-primera-parte

http://www3.uji.es/~mmarques/f47/apun/node7.html

http://www.mailxmail.com/curso-sistemas-bases-datos/sgbd-inconvenientes-sistema-gestion-archivos-segunda-parte

http://www3.uji.es/~mmarques/f47/apun/node33.html

http://www.mailxmail.com/curso-sistemas-bases-datos/sgbd-componentes-lenguajes

http://www3.uji.es/~mmarques/f47/apun/node36.html

http://ca.wikipedia.org/wiki/Diccionari_de_dades#El_component_diccionari_de_dades_d.27un_SGBD

http://analisis692.blogspot.com/2009/05/sistema-sgbd-es-el-software-que-permite.html

http://tramullas.com/documatica/2-3.html

http://www3.uji.es/~mmarques/f47/apun/node32.html

http://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

http://html.rincondelvago.com/bases-de-datos_1.html

http://www.victorgarcia.org/pfc/modeloER/extendedER.php

http://www.monografias.com/trabajos19/administracion-base-datos/administracion-base-datos.shtml

http://es.wikipedia.org/wiki/Base_de_datos_de_red

http://dbpedia.org/page/CODASYL

http://html.rincondelvago.com/modelo-de-datos.html

http://es.wikipedia.org/wiki/Base_de_datos_jerárquica

http://es.wikipedia.org/wiki/Base_de_datos_orientada_a_objetos

http://ca.wikipedia.org/wiki/Classe_(informàtica)

http://ca.wikipedia.org/wiki/Herència_(programació)

http://es.wikipedia.org/wiki/Polimorfismo_(informática)

http://es.wikipedia.org/wiki/ODMG

http://ca.wikipedia.org/wiki/Uml

http://www3.uji.es/~mmarques/f47/apun/node43.html

http://es.wikipedia.org/wiki/Modelo_relacional

http://usuarios.lycos.es/cursosgbd/UD3.htm

http://www.htmlpoint.com/sql/sql_03.htm

http://es.wikipedia.org/wiki/12_reglas_de_Codd

http://petra.euitio.uniovi.es/docencia/cursos/tercero/sis.ges.bas.dat/apuntes/12codd98.html

http://en.wikipedia.org/wiki/Foreign_key

http://www.dataprix.com/127-aserciones

http://www.slideshare.net/bdatos/triggers

Bibliografia

http://alejandria.nidaval.com/scripts/Editorial.dll?SE=2_1_0_T4_A587_82

http://www.alejandrox.com/2007/10/transformacion-del-modelo-entidad-relacion-al-modelo-relacional/

http://es.wikipedia.org/wiki/Normalización_de_bases_de_datos

http://alarcos.inf-cr.uclm.es/DOC/BDA/doc/teo/ant/BDa-T7i.doc