Disseny de Bases de Dades

Report

Autors:
Ferran Cantariño i Iglesias - 173705
Marc Rabat Pla - 172808
Xavier Tres Martínez - 160052

Disseny de bases de dades

Professor: Miquel Cornudella Gaya

ÍNDEX

1. Descripció General de la Base de Dades 3
2. Disseny Entity - Relationship Model 4
2.1. Introducció al problema i plantejament 4
2.2. Realització de l'ER Model 5
2.3. ER Model Final 10
3. Implementació 18
3.1. Plantejament 18
3.2. Definició del Model Relacional 18
4. Funcionalitats 26
5. Views 28
6. Triggers 28
7. Execució del codi 29
8. Conclusions 29

1. Descripció General de la Base de Dades

La base de dades UPFinder que hem dissenyat i implementat per aquesta pràctica, té com a objectiu la gestió de la informació necessària per representar el funcionament de la universitat. Pretén ser un model capaç de substituir el sistema que fa servir actualment la UPF.

El disseny de la base de dades permet crear i gestionar els diferents tipus de persones que d'una manera o altra són actors a la universitat, ja siguin professors, estudiants o realitzin qualsevol treball per a la UPF.
Per altra banda, es gestionen els diferents cursos i graus que ofereix la universitat i com aquests es relacionen amb els usuaris (siguin estudiants o professors), cursos en què estan matriculats, classes que s'imparteixen, etc. La base de dades tindrà un comportament diferent segons el que necessitin fer els diferents tipus d'usuari, restringint o permetent l'accés de les diferents seccions.
A més a més, també es té en compte la gestió dels llocs on es realitzen aquests cursos, juntament amb la gestió dels diferents espais físics de la universitat.

Tot i que la funció principal de la base de dades és el que s'acaba de descriure, entrant una mica més en detall veurem que altres finalitats de UPFinder és la gestió del material de la universitat i dels cursos. També es treballa una part de la gestió econòmica pel que fa els budgets dels departaments, els PDI i les subvencions rebudes pels projectes de recerca.

2. Disseny Entity - Relationship Model

2.1. Introducció al problema i plantejament

Després d'haver analitzat detalladament el document de requeriments de la base de dades que se'ns demanava implementar, vam haver de definir, abans de tot, quina seria la metodologia a seguir i quines eines ens serien necessàries per a realitzar el projecte.

Pel que fa a la metodologia, vam creure que era prioritari identificar minuciosament totes les Entity Sets que entraven en joc en la base de dades. Aquest primer pas és molt important ja que ens va ajudar a entendre la dimensió del problema i a interioritzar els requeriments. Un cop fet aquest pas tan bàsic, però no per això menys essencial, vam subdividir el nostre problema. Pensem que per afrontar un problema com aquest, el qual no se'ns havia plantejat en anterioritat, el millor és intentar jerarquitzar-lo i entendre'n bé cadascuna de les respectives parts implicades. Les prioritats que vam definir van ser les següents:

1. **Definir totes les Entity Set del nostre problema.**
   Simplement donar un nom a les entitats identificades i representar-les mitjançant la representació adequada.

2. **Definir els atributs de totes les Entity Set definides.**
   A partir de les entitats prèviament definides, definir-ne els seus atributs. En cas de tenir dubtes en algun d'ells, anotar-lo a l'entitat que es correspongui.

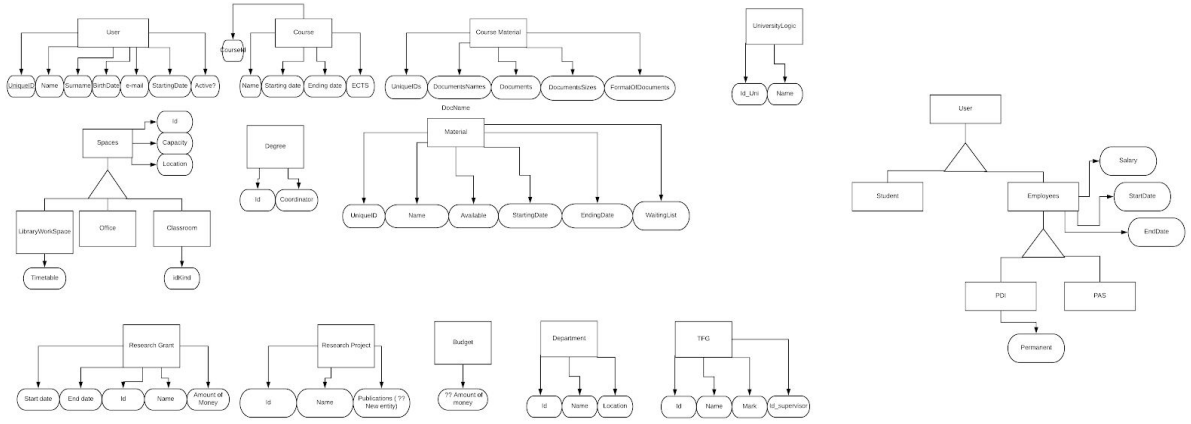
3. **Modelar les relacions entre les nostres entitats.**
   Començar primer de tot per relacions entre entitats molt bàsiques: identificar herències i dependències fortes entre entitats. Primer plantegem les relacions entre tot parell d'entitats, per més endavant compactar-les i eliminar aquelles que ja no són necessàries o redundants.

Per tal de realitzar aquestes tasques, ens vam adonar que necessitàvem una bona eina de disseny que ens permetés crear els models de forma senzilla, còmoda i que a la vegada oferís un bon nivell de visualització. Creiem que és molt important poder visualitzar fàcilment tots els elements d'una base de dades per evitar conflictes a l'hora d'interpretar els dissenys. D'entre les eines disponibles per a realitzar-ho, vam escollir LucidChart, una eina online que ens permet crear els nostres models de

forma senzilla i organitzada gràcies a l'assistent que ofereix per a saber si tenim les figures alineades. A la vegada, permet l'edició en línia amb varis usuaris, la qual cosa ens aniria bé per tal de dividir-nos la feina.

2.2. Realització de l'ER Model

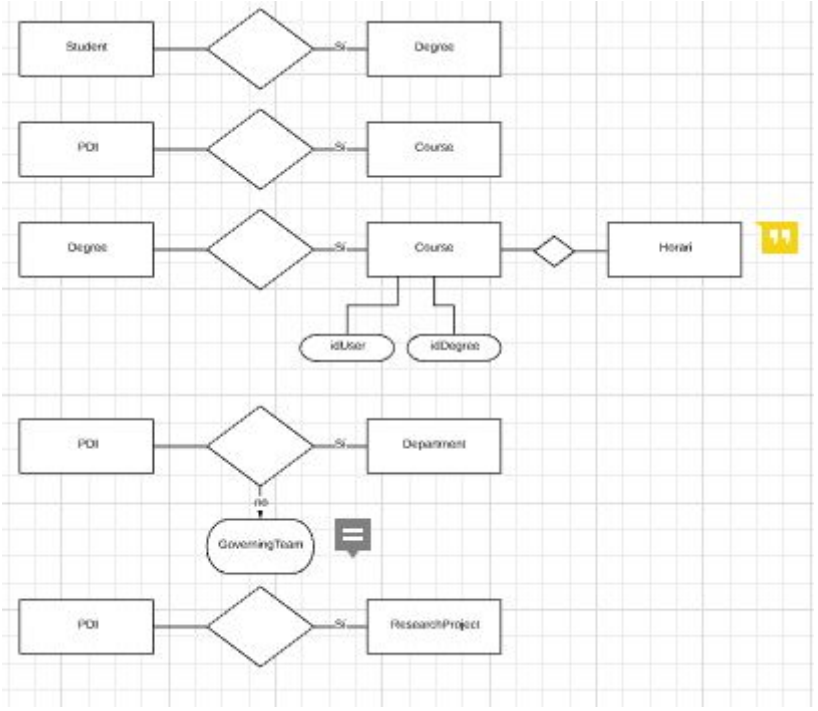
En el següent esquema, es pot observar les tasques que vam realitzar inicialment per a la definició d'entitats, atributs i algunes relacions simples. Com es pot observar, no totes les relacions plantejades es veuen reflectides en aquest esquema ja que això ens va ser necessari per a començar a entendre com realitzar els primers dissenys per a una base de dades. A mesura que se'ns anaven acudint noves formes de relacionar les nostres entitats, les anotàvem a paper o les comentàvem en una pissarra, per tant aquest esquema ens va servir més que res que per orientar la feina i entendre com havíem de realitzar els models, que més endavant aniríem refinant de forma cíclica.



Com es pot observar, vam crear les següents definicions més simples:

- **User**: UniqueID, Name, Surname, BirthDate, e-mailm StartingDate, Active?
- **Course**: CourseId, Name, StartingDate, EndingDate, ECTS
- **CourseMaterial**: UniqueIDs, DocumentsNames, Documents, DocumentsSizes, FormatOfDocuments
- **University**: Id_Uni, Name
- **Degree**: Id, Cordinator
- **Research Grant**: StartDate, EndDate, ID, Name, Amount of Money
- **Research Project**: Id, Name, Publications (?? new entity)
- **Budget**: ?? Amount of money
- **Department**: Id, Name, Location
- **TFG**: Id, Name, Mark, Id_supervisor

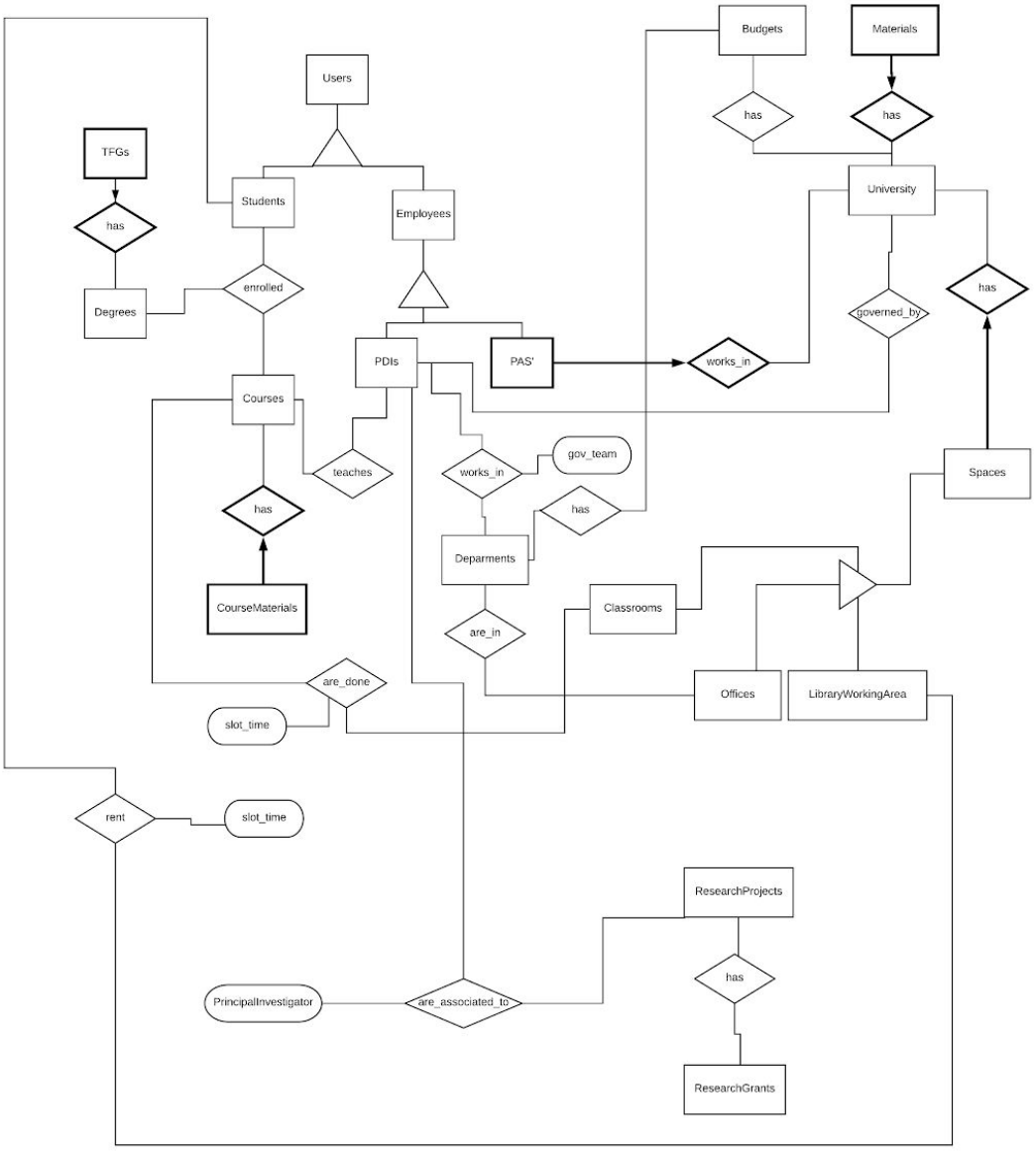
Com déiem anteriorment, inicialment anotàvem els atributs més dubtosos o que pertanyerien possiblement a una futura relació amb els símbols d'interrogació per tal de tenir-ne constància. Inicialment, es pot observar com la nomenclatura tant de les taules com d'alguns dels atributs no eren molt precisos i podien portar a confusió. Per sort, hem anat definint una nomenclatura de forma quasi implícita a mesura que hem anat millorant les versions dels nostres esquemes. Així mateix, també vam definir les herències més bàsiques com les de User amb els respectius fills Student i Employees (i Employees amb fills PDi i PAS).



Fet el pas anterior, i com també comentàvem, vam seguir per intentar relacionar les entitats més bàsiques entre elles, tot i repetir-se. Finalment les unificaríem en els posteriors models. Qualsevol cosa important que haguéssim de destacar, la marcàvem amb un comentari i la discutíem posteriorment entre els membres del grup.

El següent pas a realitzar doncs, va ser realitzar el model de la millor forma possible. En aquest model no vam incloure les cardinalitats entre les entitats ja que vam preferir acabar-les de definir en fer la implementació del model, així que simplement les teníem presents d'una forma intuïtiva.

A continuació presentem l'últim model que vam realitzar abans de començar a realitzar la implementació amb codi.



Com es pot observar en aquest diagrama, vam optar per no presentar els atributs de cadascuna de les entitats o presentar únicament els més rellevants per tal de millorar la llegibilitat del model. Com hem explicat, la definició dels atributs els guardàvem apart en un altre document. En aquest model, vam identificar els següents punts a destacar:

- Modelitzem les següents relacions:
  Relacions d'herència
  - Users
    - Students
    - Employees
      - PDIs
      - PAS'
  - Spaces
    - Classrooms
    - Offices
    - LibraryWorkingArea

  Altres relacions
  - Un Degree té associat un TFG. TFGs és weak entity de Degrees ja que en desaparèixer el grau, lògicament també n'hauria de desaparèixer el respectiu TFG.
  - Students té una relació enrolled amb els cursos. Aquesta taula de relació ens permet identificar si un estudiant està matriculat a un cert curs. En la taula de relació d'enrolled, tenim un atribut que és l'id del degree i que ens permet saber a quin degree pertany una determinada assignatura. La relació és ternària entre les tres entitats.
  - Students també té una relació amb els materials de la universitat a partir de la taula de relació rent que tindrà l'atribut amb time slot de disponibilitat corresponent a aquell material.
  - Courses té una relació amb CourseMaterials, ja que els conté. Això a la vegada ho representem amb una weak entity, per tant si desapareix el curs, també en desapareixeria el material associat.
  - Courses també té una relació amb teaches, la qual és la taula de relació que identifica quins d'aquests són impartits per quin professor.
  - A la vegada, els Courses es realitzen en una aula determinada. Aquest fet el reflecteix la taula are_done, que conté l'slot time corresponent a les hores del curs.
  - PDIs té una relació amb els departaments als quals pot treballar. La taula de relació que ho representa és works_in i es guarda la variable per a saber si aquest és membre del government team.
  - A la vegada els PDIs tenen una relació amb University ja que és qui els contracta.
  - Els departaments d'alguna forma estan formats per les oficines respectives, per tant creem la taula are_in entre departaments i oficines.
  - Pel que fa als PAS', aquests treballen per la Universitat, per tant hi tenen una relació a partir de la taula works_in. Hem creat una weak

entity ja que si desapareix la universitat, també n'haurien de desaparèixer els treballadors respectius.
- La resta de relacions, no les teníem molt clares, així que les vam representar en el nostre model com vam creure convenient de forma que ja les especificaríem millor més endavant.

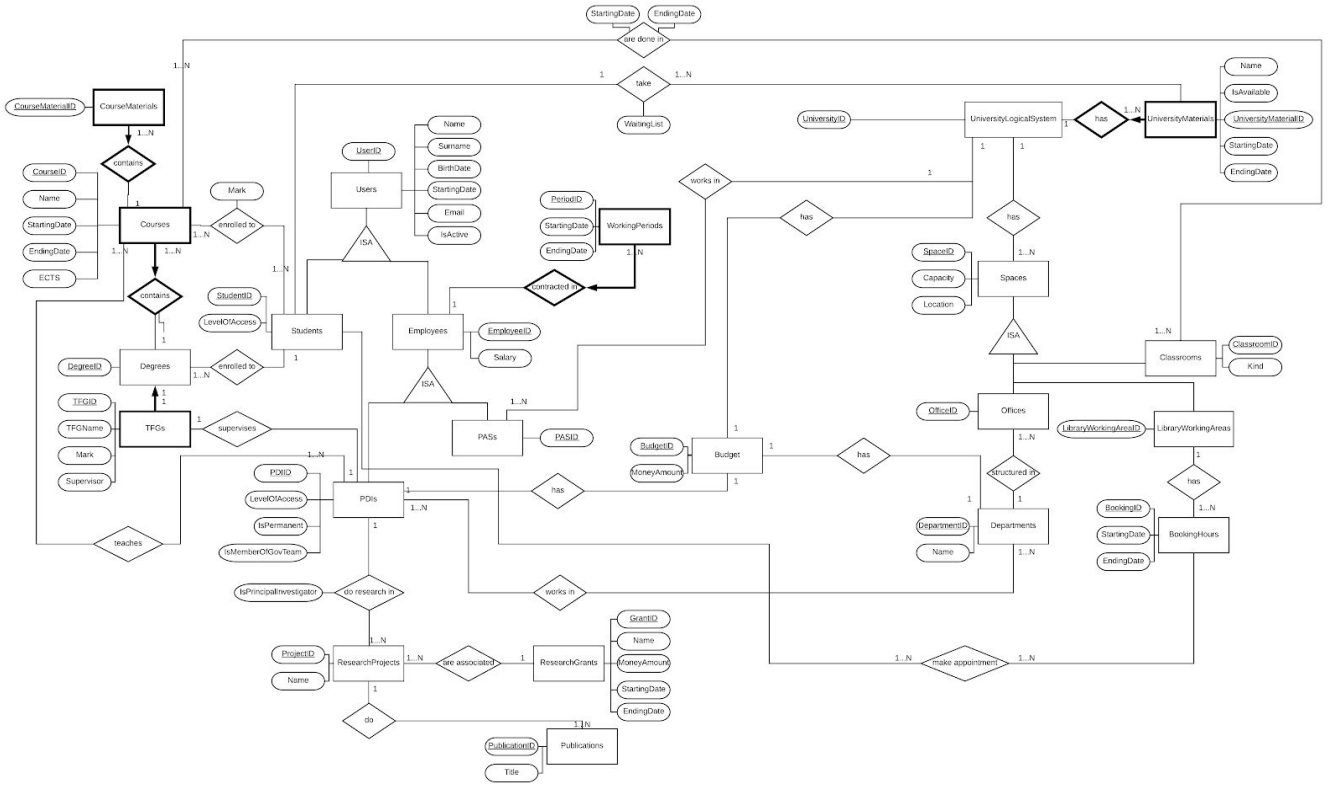
D'aquesta part en destaquem sobretot dos fets que van ser molt rellevants en les posteriors millores:

- Per una banda, estàvem sent incoherents a l'hora d'interpretar l'entitat Universitat. L'entitat d'Universitat no hauria de tenir el comportament que créiem que hauria de tenir sinó simplement ser una entitat de suport per aquelles gestions relacionades amb la Universitat. De no haver sigut així, llavors hauríem d'haver tractat tota la resta d'elements del sistema de la mateixa forma que ho féiem amb els PAS', la qual definíem com a weak entity, amb la qual cosa d'eliminar la Univeristat, n'haurien hagut de desaparèixer totes els elements corresponents, cosa que no era el que volíem representar. Per tant, ho vam arreglar en la versió definitva del model.
- Per altra banda, havíem pensat realitzar una relació ternària entre Students, Courses i Degrees, ja que ens semblava un model força lògic per a representar el problema. No obstant, vam creure que era una representació bastant més rebuscada del que realment necessitàvem que era una relació dos-a-dos entre les tres entitats implicades. D'aquesta forma, obtenir la informació de les relacions Student-Degree, Degree-Course i Course-Student es tornava una forma de veure-ho més trivial i segurament més òptima.

Addicionalment a això, millorem la nomenclatura de les nostres entitats; intentem que totes elles apareixin en plural i fem que els verbs de les relacions tinguin un sentit més profund més enllà dels *"has"*. Donem importància a aquest fet ja que interioritzar la nostra pròpia nomenclatura a l'hora de treballar ens ha ajudat posteriorment a agilitzar varis processos.

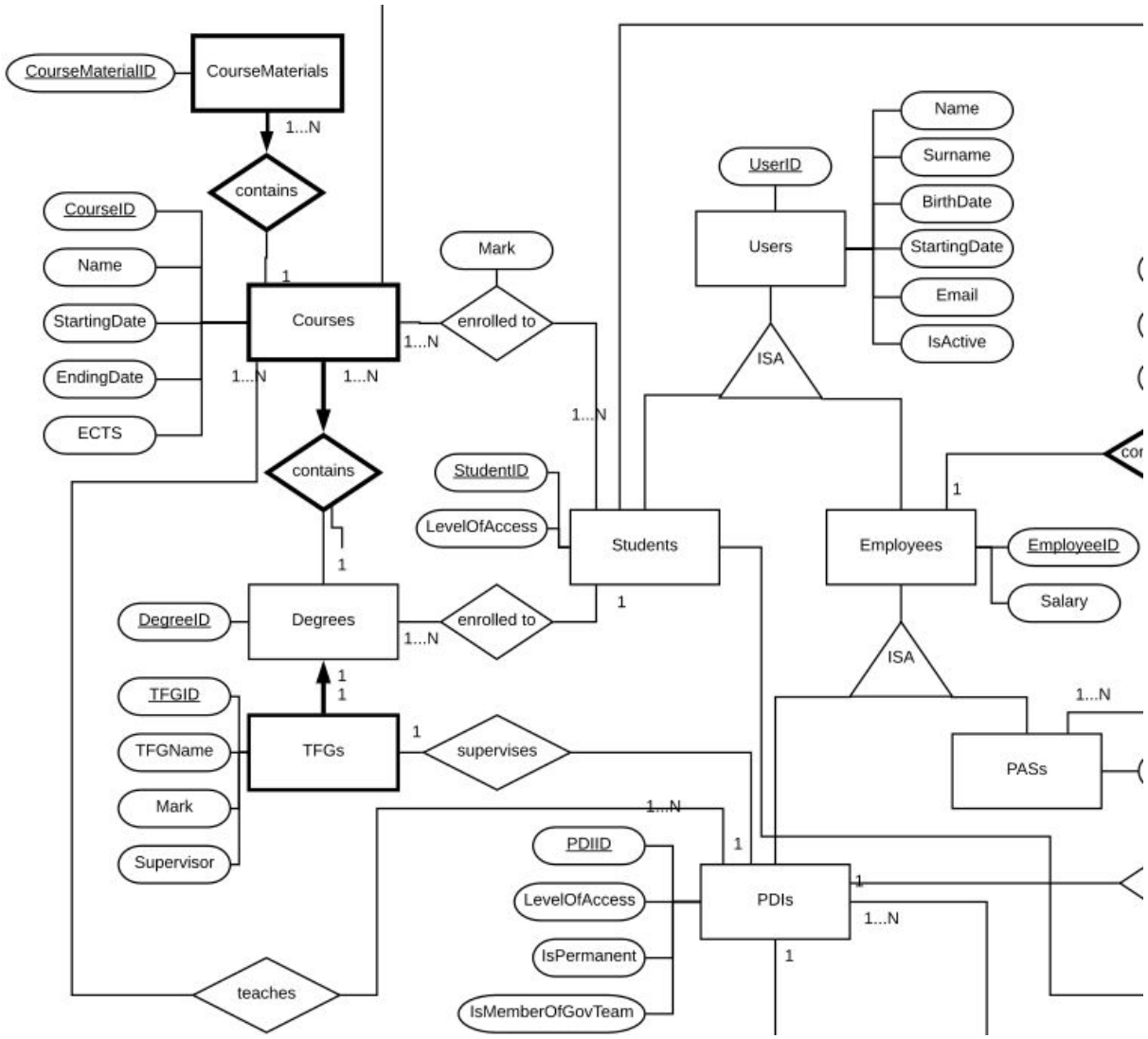
2.3. ER Model Final

Arribats a aquest punt, hem passat a la implementació de l'ER Model a codi SQL, punt que es detallarà en la següent secció. Per a realitzar el diagrama, hem afegit tots els atributs corresponents als Entity Sets així com la cardinalitat de les relacions. Després de fer algunes modificacions en la implementació, el model final ens queda de la següent forma[1]:



Atès que la visualització pot ser complicada en aquest punt, intentarem analitzar part a part el nostre ER Model dividint-lo en totes aquelles parts que tenen relacions molt directes:

[1] Afegim un adjunt de la versió del model en l'annex, per a una millor visualització.



En aquesta part podem observar totes les relacions principals vinculades als usuaris de la base de dades. Es pot veure com hem implementat les herències mencionades en punts anteriors pels usuaris i n'hem definit els respectius atributs. Així doncs, fem que l'entitat pare Users contingui tota la informació comuna dels seus fills Students i Employees. Students especialitza l'entitat de Users afegint l'atribut de nivell d'accés. Employees ho fa a través del salari que reben aquests treballadors. A la vegada PDIs i PASs especialitzen l'entitat pare Employees. Així doncs, PDIs tenen un nivell d'accés, la seva condició de si són permanents o no i una altra per saber si és part del Government Team. Els PASs tenen el seu propi identificador d'usuari i estarà vinculat amb UniversityLogic, que s'explicarà més endavant. A continuació detallem les relacions per cada una de les entitats en aquesta porció del model:

- **Users: UserID**, Name, Surname, Birthdate, StartingDate, Email, IsActive
    - Users representa l'entitat pare dels diferents tipus d'usuari del sistema.

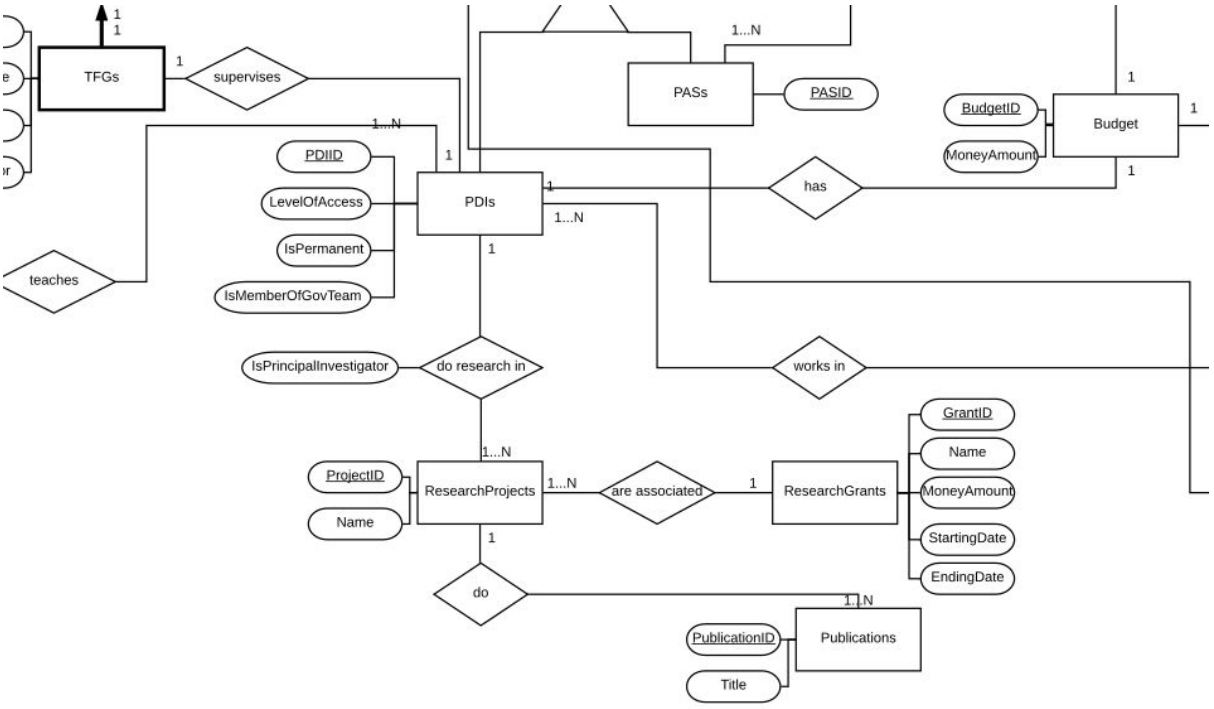
- **Students**: **StudentID**, LevelOfAcces
    - Students es relaciona a través d'una relació enrolledTo N a N amb els cursos que aquest realitzi. Aprofitarem la relació que ens relaciona un estudiant i els seus cursos per afegir la nota de l'estudiant.
    - A la vegada, Students també es relaciona amb Degrees per tal de saber a quin grau està matriculat. La relació és 1 a N
    - Students també es relaciona a les BookingHours dels espais de la universitat (N a N) a través de la taula de relació make appointment, que ens servirà per a gestionar les reserves horàries de l'estudiant. (veure taula ampliada)
    - La última relació de l'estudiant és a partir de la relació take, que és 1 a N i que contindrà un atribut que reflectirà la llista d'espera d'usuaris per tal d'agafar els UniversityMaterials (veure taula ampliada).

- **Degrees**: **DegreeID**
    - Degrees es relaciona a través de la relació enrolledTo esmentada.
    - Degrees té una relació 1 a 1 amb TFGs, que contindrà el TFG relatiu per l'estudiant en qüestió.
    - Degrees es relaciona amb els cursos a través de la taula contains (1 a N) la qual és una weak entity, ja que si eliminem el grau, han de desaparèixer els cursos que hi estiguin associats.

- **Courses: CourseID**, Name, StartingDate, EndingDate, ECTS
    - Courses es relaciona amb Degrees com s'ha mencionat.
    - Courses es relaciona amb Students com s'ha mencionat.
    - Els Courses són impartits per un PDI, per tant, hem creat una taula de relació teaches entre el curs i el PDI (N a N).
    - Courses té una relació 1 a N amb tots els materials del curs que conté. Això és reflecteix amb la taula contains per a cursos.
    - Courses es relacionen amb els espais on es realitzen, les classrooms, a través de la relació are done in (N a N) que tindrà especificades una StartingDate i una EndingDate. (veure taula ampliada)

- **TFG: TFGID**, TFGName, Mark, Supervisor
    - TFGs és weak entity de curs. No té sentit que aquesta existeixi si no n'hi ha el seu grau corresponent.
    - TFGs es relaciona 1 a 1 a partir de la relació supervises amb PDI, que és el professor que supervisarà el treball del TFG.

- **CourseMaterials: CourseMaterialID**
    - CourseMaterial es relaciona amb Courses com s'ha mencionat.



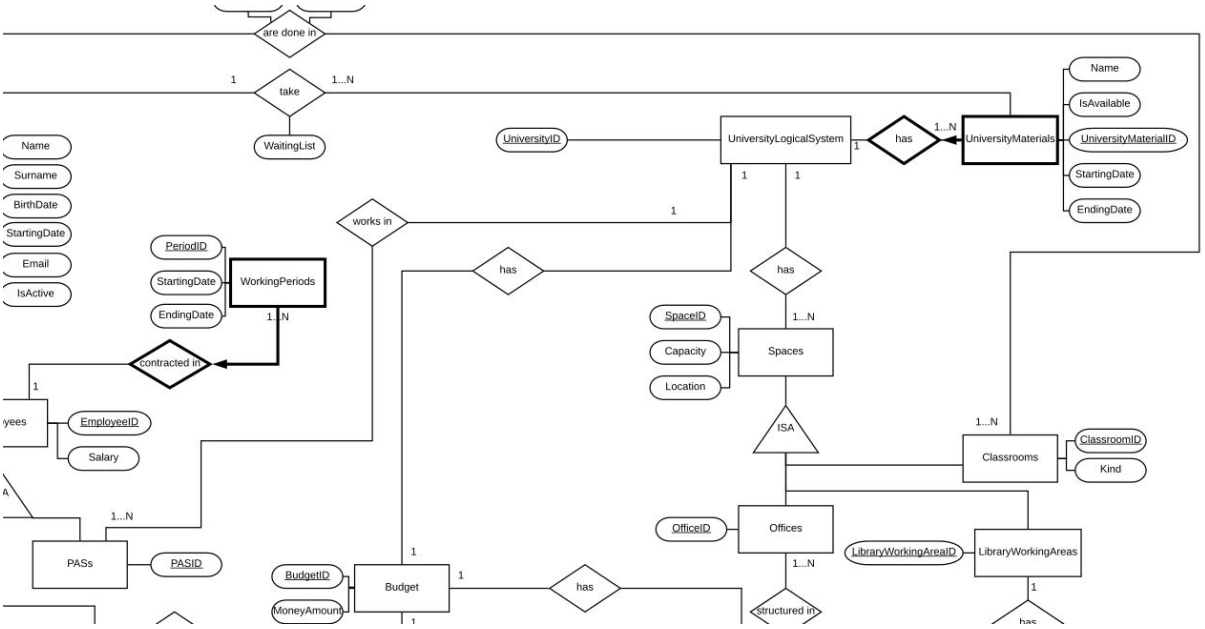
Els PDIs tenen una part molt important en la base de dades i això es demostra amb la quantitat de relacions que té amb altres entitats del diagrama.

- **PDIs: PDIID**, LevelOfAccess, IsPermanent, IsMemberOfGovTeam
    - PDI es relaciona amb Courses com s'ha mencionat.
    - PDI es relaciona amb TFGs com s'ha mencionat.
    - PDI es relaciona amb Budgets que és l'entitat que hem creat per a representar els salaris. La relació és 1 a 1.
    - Com es menciona, un PDI realitza recerca en un ResearchProject. Això ho hem representat amb la relació do research in que conté un atribut que especifica si el professor relacionat és, a més a més, un principal investigador. La relació és 1 a N.
    - A la vegada, un PDI pot treballar en un Department, i ho hem especificat a través de la relació works in N a N.

- **ResearchProjects: ProjectID**, Name
    - ResearchProjects es relaciona amb PDIs com s'ha mencionat.
    - Cada ResearchProject té 1 gran associada (N a 1) que ho hem representat a través de la taula are associated.
    - Al mateix temps, un ResearchProject pot crear publicacions, que ho hem especificat a partir de la relació do entre ResearchProjects i Publications (1 a N).

- **ResearchGrants: GrantID,** Name, MoneyAmount, StartingDate, EndingDate
    - Es relaciona amb ResearchProjects com s'ha mencionat.

- **Publications: PublicationID,** Title
    - Publications es relaciona amb ResearchProjects com s'ha mencionat.



- **Employees: EmployeeID**, Salary
    - Employees es relaciona amb l'entitat WokingPeriods, que conté els diferents períodes de contractació del personal i que es relaciona a través de la relació contracted in. La relació serà 1 a N i WorkingPeriods serà una weak entity de Employees.

- **WorkingPeriods: PeriodID**, StartingDate, EndingDate
    - WorkingPeriods es relaciona amb Employees com s'ha mencionat.

- **PASs: PASID**
    - El PAS està relacionat a través de la taula works in N a 1 amb UniversityLogicalSystem.

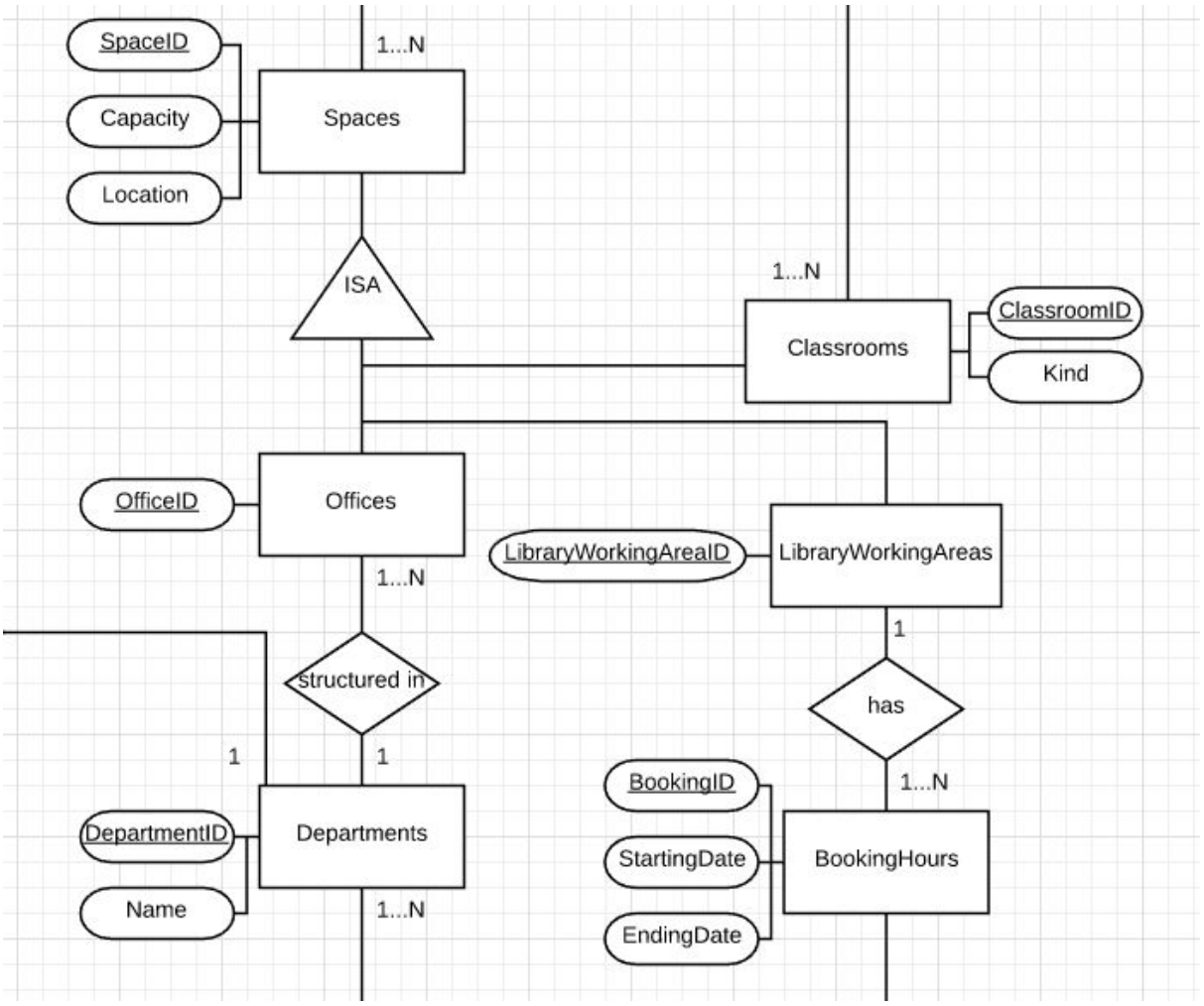
- **Budgets: BudgetID,** MoneyAmount
    - Budgets es relaciona amb PDIs com s'ha mencionat.
    - Budgets es relaciona amb Departments a través de la relació has 1 a 1 que indica que un departament té un salari assignat.
    - Budget té una relació 1 a 1 amb UniversityLogicSystem per tal de representar el salari del que la Universitat disposa.

- **UniversityLogicalSystem: UniversityID**
    - UniversityLogicalSystem és una entitat de suport que hem definit per a tal de gestionar aspectes més genèrics referents a la universitat. Així doncs, definim les següents relacions.
        - UniversityLogicalSystem té una relació amb UniversityMaterial ja que són els materials dels quals ha de disposar la universitat. UniversityMaterials l'hem definit com una weak entity ja que de desaparèixer aquest sistema lògic de la universitat, aquests també haurien de fer-ho. No obstant, remarquem (i com s'ha discutit amb anterioritat), que aquesta entitat no fa referència a la universitat en sí.
        - A la vegada també es relaciona amb l'entitat d'espais de la universitat i que fa referència als diferents espais que podria contenir aquesta universitat. No l'hem tractat de weak entity ja que aquest podrien existir sense la necessitat del sistema lògic de la universitat.
        - UniversityLogicalSystem també té una relació amb els treballadors que no formen part del professorat, els PASs, i per tant s'hi relaciona a través de la relació works in.
        - També es relaciona amb Budget com s'ha mencionat.

- **UniversityMaterials: UniversityMaterialID**, Name, IsAvailable, StartingDate, EndingDate
    - University Materials té una relació a partir de la relació take amb els estudiant. Això ho hem modelat per tal de representar el fet que un estudiant pugui llogar el material que s'ofereix des de la universitat.

- **Departments**: **DepartmentID,** Name, BudgetID, OfficeID
    - Departments té una relació amb Offices. Segons la nostra entesa, hem considerat que 1 departament pot estar estructurat en 1 o més oficines (per tant 1 a N)

Pel que fa els diferents espais de la universitat, hem fet servir la jerarquia ISA per fer-ne la representació. Els possibles espais que puguin existir formen part de l'entitat Spaces i tenen en comú un identificador del Space (SpaceID), el nombre de persones que caben a la sala (Capacity) i la descripció del lloc físic on es troba l'espai (Location) com per exemple el nom del edifici, la planta, etc.
D'aquesta manera, fem servir l'entitat pare Spaces i aconseguim no repetir atributs que són comuns a tots els tipus d'espai.



A partir d'aquí, representem els espais concrets amb entitats que hereten de l'entitat Spaces. En són tres:

- **Offices:** Tenen un identificador que les relaciona amb els diferents departaments mitjançant la relació structured, de manera que una oficina és d'un departament tot i que el departament pugui tenir varies oficines.

- **LibraryWorkingAreas:** Disposen d'un identificador (LibraryWorkingAreaID) que ens permet relacionar aquest espai amb la taula de reserves (BookingID, StartingDate, EndingDate), la qual té un identificador de reserva amb les dates d'inici i final de les reserves que poden fer els estudiants. L'enllaç de la LibraryWorkingArea amb la taula de reserves la fem a traves de la relació Has (1..N).

- **Classrooms:** Aquesta entitat té informat el tipus de classroom (Kind), a més de l'identificador (ClassroomID) com en els dos casos anteriors. La funció

d’aquesta entitat és la de guardar la informació del lloc on es realitzen les diferents classes dels cursos. Per aquest motiu fem servir la relació areDoneIn que ens relaciona l’entitat Courses amb les seves Classrooms. Així es pot registrar on s’imparteix cada curs. Aprofitant l’herència de l’entitat Spaces, seria possible saber si la sala on s’ha d’impartir un curs, té prou capacitat per acollir tots els estudiants que estiguin matriculats en el curs i hagin d’assistir a la classe.

Com hem mencionat, aquest model és el que nosaltres hem considerat com a final, després d’haver-ne fet una representació amb taules. Hem de tenir present però que és possible que no puguem “traduir” explícitament aquest model a l’hora d’implementar-lo i que es poden donar lleugeres variacions.

3. Implementació

3.1. Plantejament

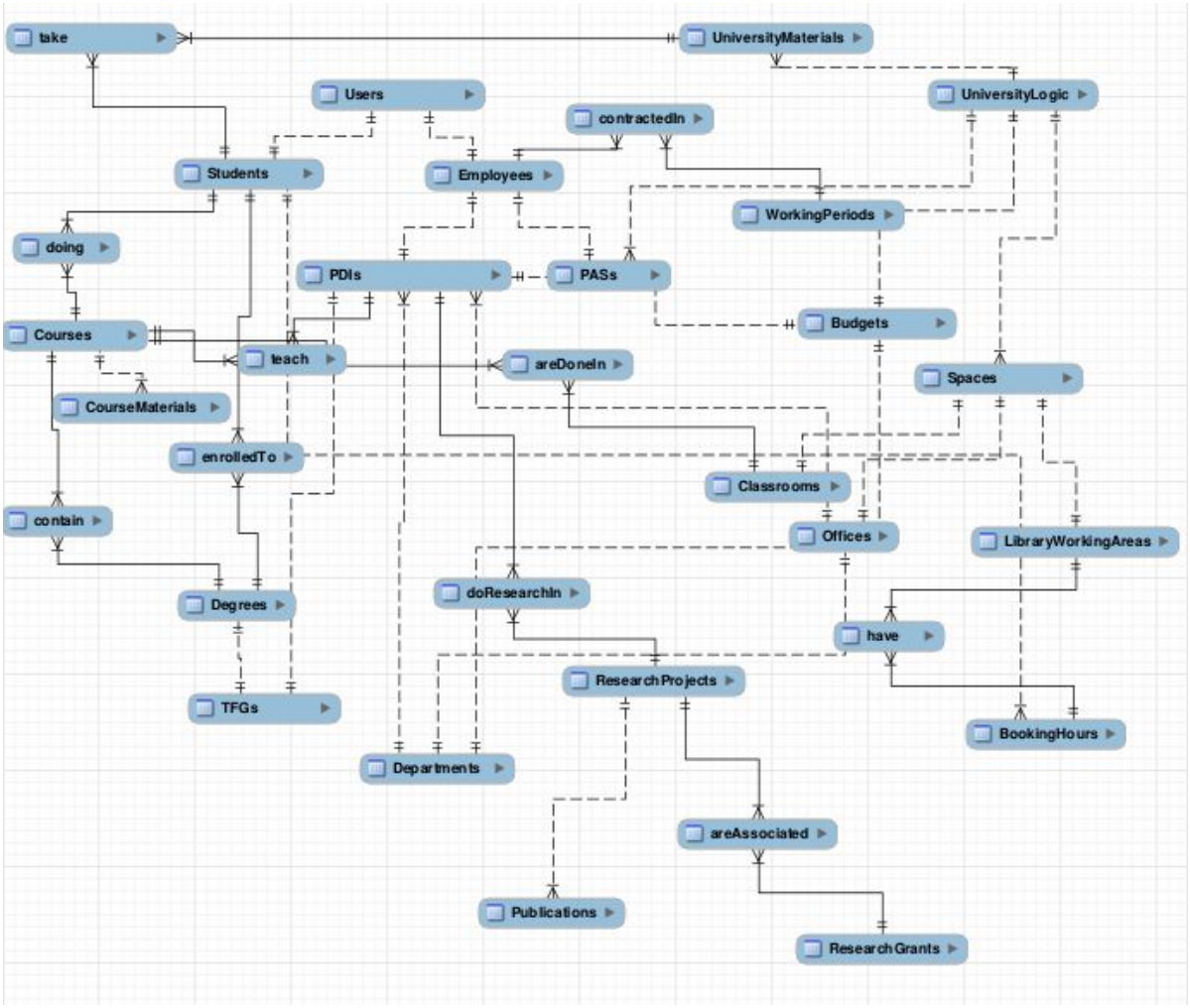
Respecte al plantejament pensat per a fer la implementació primer de tot hem definit quin seria el sistema de gestió de bases de dades a utilitzar. En el nostre cas hem escollit MySQL ja que tot i no tenir-hi molta experiència, és un dels sistemes més estesos i utilitzats a l'hora de realitzar bases de dades.

Pel que fa la implementació en codi SQL del model, el que vam fer primerament va ser començar amb la creació de les taules amb l'eina Sublime Text. Després d'haver implementat pràcticament la meitat de les taules a representar, vam veure que el procés de creació de les taules es podia tornar molt farragós i que donada la poca experiència que teníem en escriure codi en SQL ens podia portar a diversos errors de nomenclatures que més endavant ens podrien haver dut molts contratemps, amb la qual cosa vam prendre la decisió de migrar el codi ja realitzat a MySQL Workbench, que és un assistent que ens podia ajudar a realitzar aquesta tasca de forma més ràpida i segura. Cal remarcar però, que després d'haver-lo utilitzat ens hem adonat també de la poca llibertat que ens oferia aquesta eina, però hem prioritzat poder guanyar temps per davant d'aquests possibles contratemps.

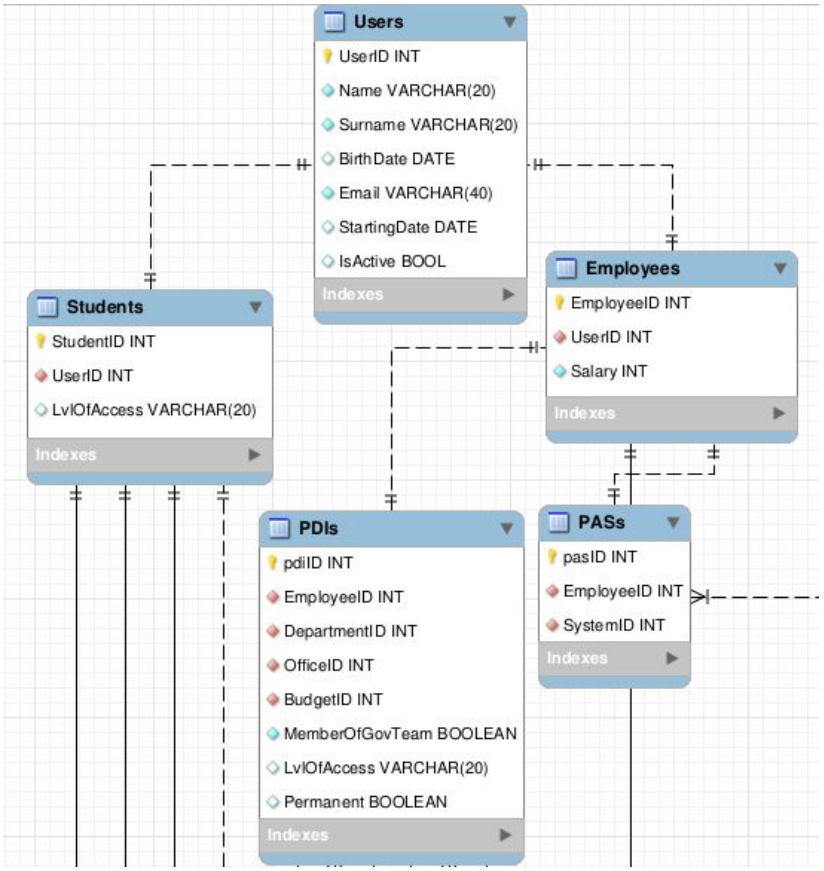
La idea doncs que vam plantejar va ser la de traduir en tot el que fos possible totes les entitats a taules per a després relacionar-les en base al model que havíem plantejat dins de l'eina MySQL Workbench. Un cop fet això doncs, vam procedir a modelitzar les relacions que es poden fer de forma simple a partir de les opcions que ens ofereix el workbench per a afegir les relacions. El workbench però, ens assigna automàticament els noms de les taules de relació que es genera així com les foreign keys que hem hagut de modificar manualment per tal que s'ajustéssin a la nomenclatura utilitzada en el nostre model. Tots aquests detalls els veurem en el següent apartat.

3.2. Definició del Model Relacional

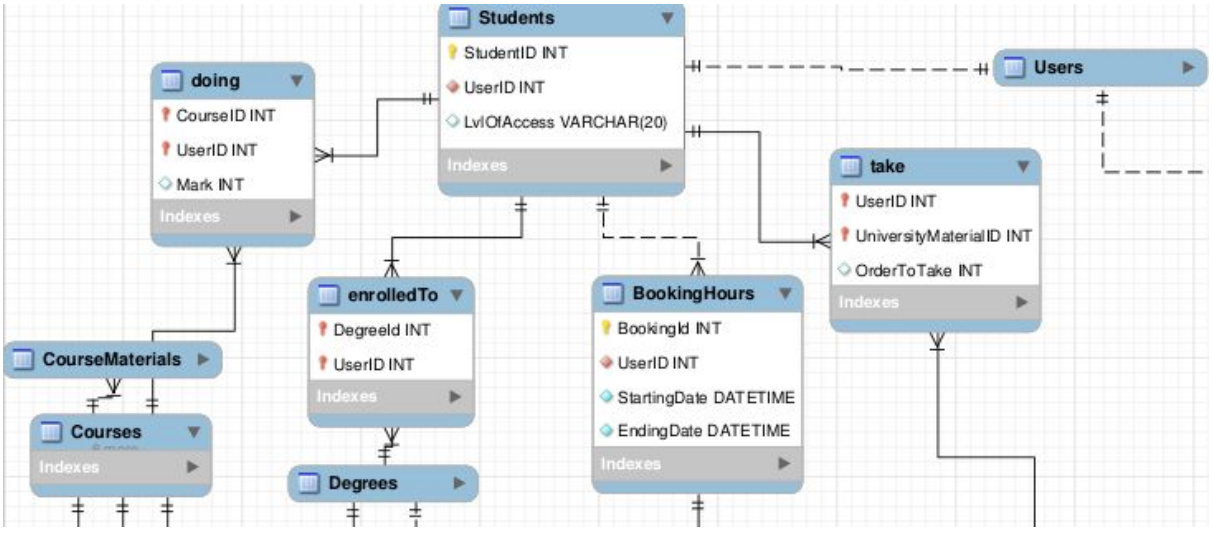
Entenem que per a definir el nostre model relacional el més important és definir el nom de la nostra base de dades, que s'anomena UPFinder. A partir d'aquest tret tant bàsic, però no menys important hem prosseguit amb les definicions de les taules. El model relacional resultant ha estat el següent:



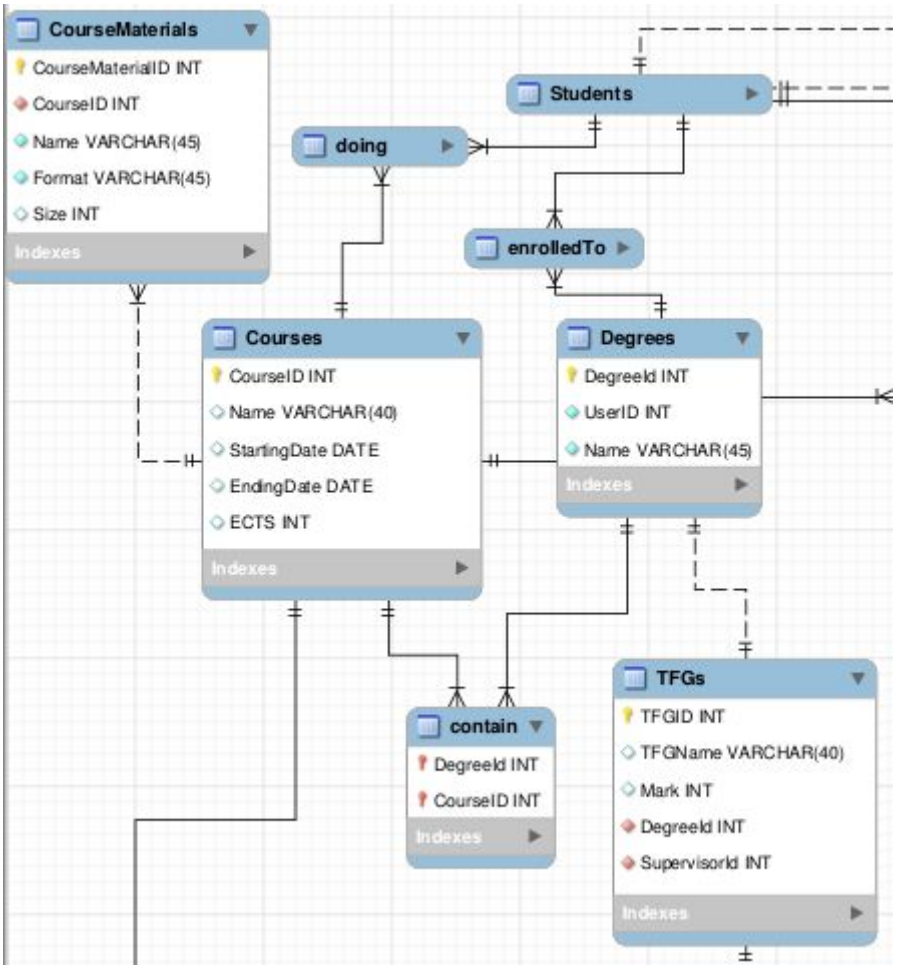
A continuació oferim una explicació detallada per a les taules implementades, que per tal d'agilitzar-ne l'explicació intentarem agrupar-la en diferents parts. En cadascuna de les taules es pot veure com definim els tipus de dades per a representar els diferents atributs.



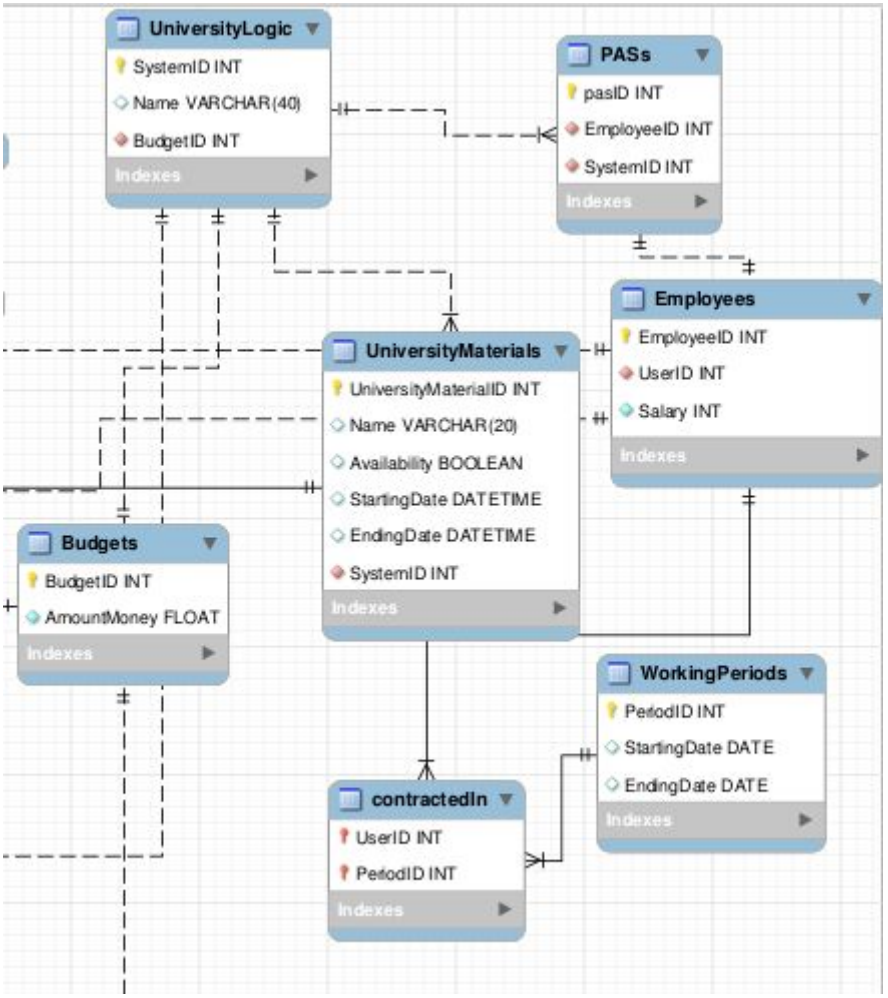
Pel que fa als usuaris, que són uns dels principals actors del sistema hem cregut que la millor forma de reflectir l'herència és mitjançant foreign keys per a representar la relació entre pares-fills, així doncs la relació en les herències és 1 a 1, ja que representen el mateix usuari però s'especialitza a partir dels atributs no comuns. Així doncs, la taula principal és Users, que té com a clau primària UserID. Aquest usuari tindrà els atributs Name, Surname, BirthDate, Email, StartingDate (per a representar en quina època va començar) i IsActive (si aquest està en actiu o no). Els seus fills són Students i Employees. Students disposarà d'un StudentID, que és la clau primària de l'entitat així com els atributs UserID i quin és el seu nivell d'accés, que hem definit com un string (Read / Upload). Els Employees disposen de l'EmployeeID que l'identifica així com el seu salari. Amdbues entitats tenen un atribut UserID que és la clau forània cap a Users. Hem fet que per aquestes claus forànies es realitzi un DELETE ON CASCADE ja que en desaparèixer un usuari de la taula pare, també ho faci per a les taules filles. A la vegada, aquest mateix raonament l'hem aplicat entre Employees i PDIs i PASs. Els PDIs tenen el pdiID que n'és la clau primària, DepartmentID per a saber amb quin departament està relacionat aquest professor, OfficesID per a saber en quina oficina treballa, BudgetID per a conèixer-ne el seu pressupost, si és membre d'un GovernmentTeam a partir d'un booleà i de la mateixa forma si aquest és permanent. Els PASs, que representen la resta d'empleats de la universitat. Aquest consta del seu PASID, l'EmployeeID i el SystemID que es correspon al sistema lògic de la universitat.



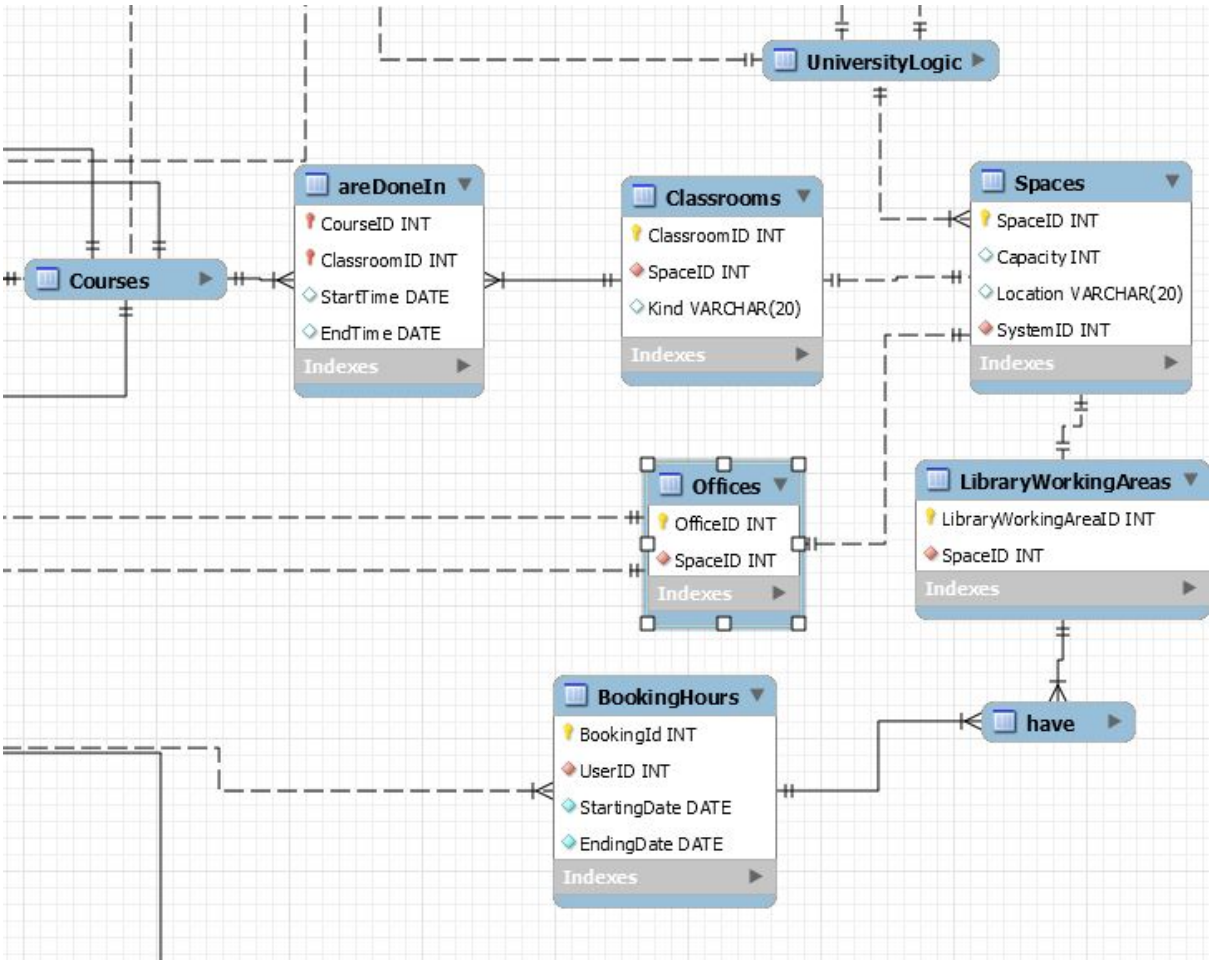
Tot seguit especificarem les relacions que hem modelat per Students. Com es pot observar, es modela la relació entre els estudiants i els seus cursos a través d'una taula que contindrà el CourseID i el UserID que ens permetrà enllaçar aquestes dues entitats. Addicionalment a això, hem pensat que la millor forma de definir la nota d'un estudiant era aprofitar aquesta mateixa taula de relació per a tenir la nota d'un estudiant dins d'un curs. En aquesta taula tenim doncs les dues foreign keys per a cada una de les entitats implicades. Per la clau forània dels estudiants, hem fet un ON DELETE CASCADE ja que de desaparèixer-ne un d'aquests, ja no té més sentit que segueixi existint en aquesta taula de relació. La clau primària de la taula està formada per la tupla CourseID-UserID (com veurem en la resta de relacions d'aquest mateix tipus). A la vegada, i de la mateixa forma, aquest estudiant es relaciona amb els Degrees als quals està matriculat a partir de la taula enrolledTo i conté primary key formada per DegreeId i UserID. Aquestes dues relacions ens permetran conèixer de forma molt accessible les dades de l'estudiant. Altrament, un usuari podrà realitzar N reserves als espais reservats dins la universitat, per tant modelem aquesta relació 1 a N on BookingHours contindrà la BookingID (PK), una clau forània UserID i la informació sobre la data en la que comença i acaba la reserva. Hem escollit el format DATETIME per a poder representar les hores. Finalment, l'estudiant es relaciona a través de la taula take amb els UniversityMaterials. Per a modelar la cua d'espera hem pensat que el millor era afegir l'atribut OrderToTake que serà un valor auto-incremental que ja ens servirà per a representar l'ordre de peticions.



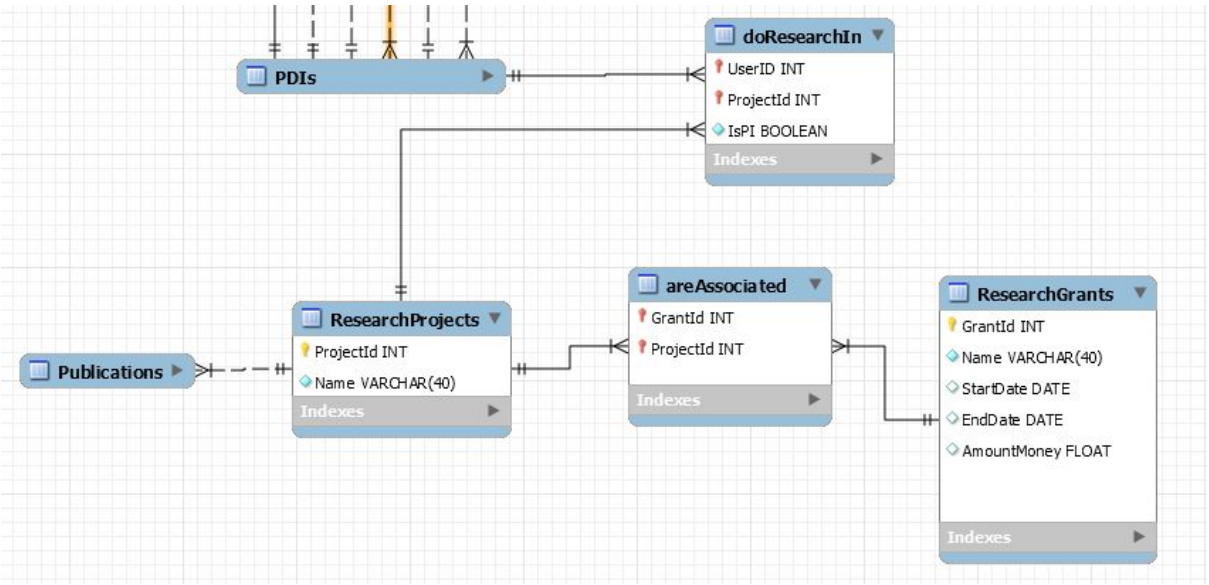
Un curs disposarà d'un CourseID (PK), el nom, la data d'inici de les classes de l'assignatura i la data final i el número de crèdits ECTS de l'assignatura. Els cursos es relacionen a través de la taula contain (que en el sentit de l'explicació seria llegida com a "estan continguts en") Degrees. La idea d'aquesta taula de relació segueix la mateixa idea que la mencionada anteriorment pels estudiants. Els graus tenen el seu propi identificador, així com l'id de l'usuari (en aquest cas l'estudiant) que hi està matriculat i el nom d'aquest grau. A un grau li correspon un TFG (relació 1a1), i aquest TFG ha de contenir l'identificador del TFG, el nom, la nota del TFG i a la vegada a quin Degree es correspon i qui n'és el seu supervisor (claus foràanies a les entitats respectives). Per a representar el material del curs ho hem fet mitjançant una relació 1 a N ja que un curs pot contenir-ne varis. Per aquesta taula hem modelat la taula de CourseMaterials en base als requeriments de l'enunciat.



Per a representar el Sistema logic de la universitat hem creat la taula UniversityLogic. Aquesta conté un SystemID que no és rellevant però sí que l'hem d'afegir ja que tota taula ha de tenir com a mínim una primary key. Li hem afegit un nom per identificar-lo i l'hem relacionat 1 a 1 amb l'entitat de Budgets, és a dir, que representarà el pressupost disponible per a la universitat. Aquesta universitat a la vegada pot contenir de 1 a N UniversityMaterials. Hem definit aquests materials a través d'un ID, un Name, la disponibilitat (Availability) i les StartingDate i l'EndingDate per tal de representar el temps de lloguer d'aquests materials (com mencionàvem abans a partir de la taula take). Els PASs són fills de la taula Employees i a la vegada N d'aquests treballen en 1 únic "sistema que representa la universitat". Tot i que no es pugui apreciar bé en aquesta imatge, per a capturar el fet que un empleat pugui tenir diversos períodes de contractació, hem creat una taula de relació entre Employees i WorkingPeriods (contractedIn), que ens vincula la informació dels empleats seguint la mateixa lògica en altres escenaris similars. Així doncs, els WorkingPeriods ens serviran per a representar aquests períodes de contractació a partir dels atributs Starting i Ending Dates.



Quant a l'implementació dels espais, com es pot veure a la imatge, les entitats Classrooms, Offices i LibraryWorkingAreas, són filles de l'entitat Spaces. Aquestes tres entitats van relacionades amb l'entitat pare per mitjà d'una Foreign Key que enllaça el SpaceID. D'aquesta manera fem servir els atributs comuns Capacity i Location a la taula pare i per altra banda els específics a les taules filles.
Per relacionar les classes amb els cursos fem servir una taula relacional que vincula l'identificador del curs amb l'identificador de la classe on es fa aquell curs.
Pel que fa les oficines, la mateixa taula ja fa la relació tant amb el PDI com amb el departament mitjançat l'OfficeID.
Respecte a les reserves de LibraryWorkingAreas, la taula Have té la funció de relacionar aquesta entitat amb la taula d'hores de reserva, la qual disposa d'un trigger amb la finalitat de que, si es vol inserir una tupla en aquesta taula, l'hora de reserva ha de ser vàlida i d'una durada de 4 hores com a màxim, contràriament no es podrà fer l'acció.



L'última imatge mostra com està resolta la implementació per tractar els projectes i les subvencions.
Partint dels PDI, aquests poden treballar en diferents projectes, cosa que representa la taula doResearchIn mitjançant el UserID i el ProjectId del projecte al qual esta relacionat. A partir d'aquí, tenim la taula de projectes on s'informa el nom i el número identificador (ProjectId). Aquests projectes poden tenir diverses subvencions, fet que implementem amb la taula ResearchGrants i la taula areAssociated que relaciona la subvenció amb el projecte (amb el ProjectId i el GrantId com a claus). Per les dates de la taula de ResearchGrants fem servir el tipus DATE ja que no necessitem precisar l'hora.
Tenint en compte que tota aquest apartat de la implementació parteix dels PDI i que tant les subvencions com els projectes acaben vinculats al PDI, es pot aplicar un delete en cascada, ja que una subvenció ha d'anar associada a un projecte i el projecte a un PDI.

De la part d'implementació del model relacional en destaquem el fet que havíem fet un mal plantejament a l'hora d'utilitzar les foreign keys en un principi i no érem capaços d'implementar correctament els comportaments desitjats per aquelles entitats que tenien dependències amb altres (com per exemple, casos de weak entities, entre altres). Tot i haver buscat informació i documentar-nos, hem vist que és molt difícil trobar respostes convincents als problemes que ens han aparescut ja que en fòrums com ara StackOverflow poden aparèixer problemes molt variats i que no tots són aplicables en el nostre cas, amb la qual cosa hem hagut de dedicar molt temps en informar-nos d'aquests petits detalls que són tant rellevants.

4. Funcionalitats

Per a dur a terme les funcionalitats, i segons la documentació que hem realitzat, hem vist que podríem dur-les a terme a partir d'Stored Procedures, que són uns mètodes que podem emmagatzemar estàticament. Així doncs, es poden cridar a través de la comanda SQL "CALL nom_de_la_procedure(paràmetres)". Hem cregut que era necessari fer-ho així per tal de tenir una mínima interfície per a utilitzar la base de dades. Aquests procediments es veuen millor reflectits en el codi que s'entrega. Així doncs, hem definit procediments per a:

- **Creació / Eliminació d'usuaris**: En aquest apartat hem afegit els mètodes més bàsics per a poder crear tots els tipus d'usuaris i poder-los eliminar. Eliminant simplement el pare amb deleteUser, podrem eliminar-ne els respectius fills.
    - createUser
    - createStudent
    - createEmployee
    - createTeacher
    - createPas
    - deleteUser

- **Creació / Gestió de cursos i graus:** En aquest apartat inclourem els mètodes referents a la gestió de cursos i de graus.
    - createDegree
    - createCourse
    - deleteDegree
    - deleteCourse

    ○ **Afegir / Remoure usaris (per a cursos/graus):**
    - deleteUserFromCourse
    - addUserToCourse
    - addUserToDegree
    - deleteUserFromDegree

    ○ **Actualitzar materials:**
    - uploadMaterial

    ○ **Assignar notes:**
    - addGrade

    - **Canviar permisos:**
    - changePermissionForUser

- **Gestió d'espais físics:** Aquí definim tots els mètodes referents a la gestió d'espais de la universitat.
    - createSpace
    - createClassroom
    - assignCourseToClassroom

    - Oficines i altres espais:
    - createOffice
    - assignOfficeToPdi
    - assignOfficeToDepartment
    - createLibraryWorkingArea
    - bookLibraryWorkingArea

- **Lloguer de materials:** Mètodes referents a l'addició i reserva de materials.
    - createMaterial
    - bookMaterial

- **Solució a constraints:** Aquests mètodes ens solucionen algunes constraints bàsiques proposades implícitament en l'enunciat.
    - **checkIfStudentPassDegree**: per a verificar si un estudiant ha aprovat o no la carrera, comprovant que ha aprovat totes les assignatures i el tfg. En aquest punt hem hagut d'afegir una millora a la taula de Students i afegir-li l'atribut de TFGMark.
    - **accessDepartmentBudget**: per a verificar si un pdi té accés o no al pressupost del departament al qual pertany.

5. Views

Hem considerat oportú crear algunes views de la nostra base de dades per tal de visualitzar informació útil d'una forma més accessible. Així doncs, hem definit una view **StudentCampusInfo** que conté tota la informació d'un usuari així com els nom, cognom, correu electrònic, cursos que realitza, grau i la nota.

Una altra view l'hem dedicat als **TimeTables** per tal de visualitzar quins professors imparteixen certes classes, en quin horari i a quines aules.

També hem realitzat una view **BudgetsOfDepartment** que ens permet veure el pressupost per a un determinat departament.
A la vegada, hem afegit altres views que no són tant rellevants i que es veuen en el codi.

6. Triggers

Pel que fa als triggers, hem realitzat els següents:

- **checkTwoCoursesSameUpdate:** Verifica que dues assignatures no s'estiguin impartint en la mateixa aula i en el mateix període de temps.
- **checkTwoCoursesSameInsert:** Verifica que no es pugui afegir una assignatura a la mateixa franja horària i aula que una de ja existent.
- **checkTwoCoursesSameTimeForPdi:** Comprova que un PDI no pugui tenir dues classes a la mateixa franja horària.
- **bookNoLongerFourHours:** Assegura que un estudiant no pugui reservar un espai de la biblioteca per un temps superior a 4 hores.

A més a més, en disposició de temps, creiem que s'haurien d'haver realitzat:

- Trigger per a comprovar que el budget d'un departament sigui superior al conjunt de salaris dels pdi que treballen en aquest departament.
- Trigger per a comprovar que el budget de la universitat és suficient per al pagament del salari dels PAS.
- Trigger per a actualitzar el budget personal del pdi després d'haver rebut una grant, així com també el budget del seu departament peritnent.

7. Execució del codi

Per a executar el nostre codi el que s'ha de fer és el següent:

1. Executar el codi UPFinderCode.sql que és el fitxer generat a partir del model del Workbench. (Creació de la base de dades i de les respectives taules i relacions, així com les Views)
2. Executar el codi Inserts.sql que emplenerà amb valors les taules, en l'ordre que es mostra en l'script ja que existeixen dependències.
3. Executar el codi de l'arxiu de Funcionalitats.sql, que conté els triggers i procedures que cobreixen els requeriments mínims i els triggers i procedures respectius a les constraints del disseny.

8. Conclusions

En conclusió, aquesta pràctica ens ha fet adonar de la complexitat que comporta desenvolupar una bona base de dades ja que cal fer moltes consideracions per a satisfer els requeriments del problema que es proposi. Podem entendre doncs quant de difícil es pot tornar aquesta tasca ja que pot tornar-se molt subjectiva i pot haver-hi moltes implementacions vàlides en funció de com es plantegi. Així doncs, entenem perquè els professionals d'aquesta professió són tant valorats.

Pel que fa a la implementació pròpiament de la base de dades, n'extreiem que hem après moltes coses sobre el llenguatge SQL. Degut a alguns contratemps a l'hora de dur a terme la implementació, com la que hem mencionat, no hem pogut assolir el nivell de robustesa per a la nostra base de dades com teníem en ment inicialment. Això ens ha fet perdre un temps molt valuós per a poder aprofundir amb més detall amb procediments de verificació de la integritat d'algunes dades. Malgrat tot això, pensem que ha estat una pràctica profitosa en la qual hem adquirit molts coneixements i sobretot concienciació pel que fa a acostumar-nos a aquest tipus de projectes de certa dimensió que ens podem trobar perfectament en l'àmbit laboral.