**6.1 VIDEO (ERD)**

Gercek hayattaki iliskiyi bir veritananina donusturme olarak dusunulebilir.

- ER Diagrams ☺ ☺ ;)

Modeling

Entity = Varliktir veri anlaminda bir varliktir. Bir dunyayi modellerken o konuyla alakali entity olarak koyabileceginiz hersey olur yani mesela iste sinif, ogrenci, akademisyen mesela bir universite icin.

Entity’ler aslinda bir tabloya denk geliyor. Icindekiler ise yani kolon isimlerimiz ise attribute’dir.

Mapping Cardinality Constraints

- one to one - many to one

- one to many - many to many

- Mesela ogrencilerimiz var ve her bir ogrenci bir tez aliyor olsun. (one to one). One to one iliskileri biz veritabaninda cok sevmeyiz, genelde tek bir tabloya indirgenebilir. Aslinda bir butun tabloyu ikiye boldugumuzde bu iliski ortaya cikar.

- En cok sevdigimiz one to many’dir. Bir danismanin bir suru ogrencisi olabilir ama bir ogrencinin bir tane danismani olur mesela. Bu konuya ogrenci tarafindan bakacak olursak bu durumda ogrenci cok danisman tek oldugu icin many to one olur. Aslinda many to one diye bisey yoktur.

- Many to many = Ders var ogrenci var, bir dersi bir suru ogrenci aliyor, bir ogrenci bir suru ders aliyor ;)

Keys

- Super Key - Candidate Key - Primary Key

Multivalued

Bir satira birden fazla deger eklenebilme ozelligi. Mesela bir musterinin birden cok telefon numarasi olabilir.

Roles

Iliskiler her zaman farkli entity’ler (tablolar) arasinda tanimlanmiyor yani bir entity’nin kendi icinde de iliskisi olabilir. Mesela employer tablosu (entity) oldugunu dusunelim calisan da bir isci, yonetici de bir isci ama ikisi arasindaki iliskiyi yani birinin otekinin yoneticisi oldugunu tanimlamamiz gerekir iste bunlara da ‘role’ denir.

Partification of an Entity Set in a Relationship Set

Mandatory – Mesela bir borcun olabilmesi icin bir musterinin olmasi mutlaka gereklidir. Musteri olmazsa borc olamaz.

Optional – Ama mesela her customer’in borcu olmak zorunda mi? tabiki hayir.

Dolayisiyla customer’in varligi icin loan’a ihtiyac yok ama loan’in varligi icin customer’a ihtiyac var.

\*\*\* ER Diagraminda yukarida anlatilan herseyin iste sekili gosteriliyor ben yazmadim. Atiyorum attribute ucgenle gosterilir ne biliyim iste primary key olan ozelligin alti cizilidir gibi…

One to many iliskileri genelde araya bir tablo koyarak veya baska bir yontemle binary-ikili iliskiye cevirmeye calisiriz.

Weak Entity Sets

Primary key icermeyen entity’lerdir.

Specialization

Palamut baliktir, ogrenci insandir tarzinda iliskilerdir. ( ISA iliskisi )

Isci ve musteri insandir. Patron, calisan, yonetici iscidir tarzi zincirleme de olabilir vs…

Many to many iliskiler nasil one to many ‘ye indirgenir?

Ogrenci ve ders adinda iki entity’miz olsun ve bu ikisine baktigimizda aralarinda many to many iliski vardir. Bunu one to many’ye indirgemek icin araya Ders Kaydi adinda baska bir entity ekleyip ogrenciden ders kaydina one to many iliski olusturuyoruz ve ayni sekil de bir ders kaydinda da bir cok ders olabilir yine one to many. Bu sayede one to many iliskiyi kurmus olduk.

**6.2 VIDEO (UML – Unified Modelling Language)**

Gorsel bir dildir. OOP ile birlikte cikmistir. Class diyagramlarini gostermek icin cikti ilk etapta. Daha sonra her turlu gosterimde kullanildi. (Standart bir sekilde)

Skecth (kaba, ustun koru cizim) ve Blueprint (detayli cizim) adinda 2 tip UML yaklasimi vardir.

Platform ve teknoloji bagimli veya bagimsiz olabilir.

UML cizimi icin yaklasimlar

- Selale (waterfall) = Adim adim her stepin tasarlandigi ve gosterildigi bir surec.

- Iteratif = Sistemin modellendigi ve tekrar tekrar o model uzerinden degisiklikler yapildigi surec.

Mesela diyelim ki hesap makinesi yazacagiz cok kestiremedigimiz, tahmin edemedigimiz bir yapi yok bu halde predictive planlama kullaniriz aksi takdirde eger bir belirsizlik varsa mesela degisimler, iste strateji degisiklikleri falan her neyse bu durumda Adaptive(Agile) yaklasim kullaniliyor.

UML Tipleri

- Use Case = Bakis acisidir. Sistemdeki aktorler ortaya cikariliyor. Sistemde kim olacak ne yapacak ;)

- Information Flow = Bir sistemi cozen analistin bir sorunu cozebilmek icin bir polis gibi bilgi izi surmesi.

Bilgi nerelerden geciyor bunlar izleniyor.

- Class Diagram = Sistemimizde tasarladigimiz classlarin birbiri ile iliskisini gosterir. (Inheritance vs…) (Design Patterns) (Domain Models) (Implementation) (API – hybernet)

- Package Diagrams = Bir package’I olusturan classlarin birbiri ile iliskisini gosterir.

- Component Diagrams = Mesela bir “WebStore” bilesenimiz var ve bunun da kendi icinde iste SearchEngine, Shopping Cart, Authentication vs… gibi bilesenleri var iste bu tum bilesenlerin iliskilerinin gosterildigi diagramlardir.

- Deployment Diagrams = Bir verinin veya iste bizim olusturdugumuz birseyin fiziksel olarak o an nerde oldugunun gosterildi diagramlardir. Network deployment yapabiliriz mesela iste cihazin agin surasinda.. Veya iste veritabaniniz ve web sunucunuz var iste bunla ayni sunucu da calisiyor veya ayri calisiyor tarzi iliskileri gosterir.

- Activity Diagrams = Bir islemin hangi asamalardan gectigini gosterir. Hareketleri, aktiviteleri gosterir. Iste satin alma islemi mesela once calisana geliyor sonra yoneticiye sonra bolge temsilciligine salliyorum yani bunun gibi seyler.

- State Machine Diagrams = Automato’da kullanilan bir sistemin, bir kullanicinin, bir makinenin vs.. durumlarini gosteren diagramlardir. Iste kahve makinesi mesela parayi attim iste parayi kabul etti kahve verdi sonra beklemeye gecti bilmem ne tarzi makine durum diagramlari iste.

- Sequence Diagrams = Objelerin (Nesnelerin) calisma sirasinda birbirleriyle nasil iletisime gectiklerini nasil birbirlerinin atiyorum fonksiyonlarini cagirdiklari etkilesim kurduklarini falan gosterir. Iste mesela internetten alisveris yaparken atiyorum birsey aratiyoruz sonra iste aciklamalarini bakiyoruz istersek (optional) ondan sonra begenirsek iste sepete atiyoruz en sonda sepettekileri satin aliyoruz mesela… Her bir surec ayri ayri anlatililir. Burda bir musterinin siparis verme sureci anlatilmistir.

- Communication Diagrams = Haberlesme icin, sistemin haber vermesi icin kullanilan diagramlardir.

**7. VIDEO (Normalization – Normal Sekiller)**

Atiyorum iste sabit degerlerin mesela iste bir tabloda computer science icin maas 1000 lira ise ve herkes icin bu ayni ise biz her yeni kayit ekledigimizde bunu tekrar tekrar girip hem vakit hem de veritabanimizdan hafiza kaybederiz bos yere hafizayi doldururur bu bilgiler hem de veri islenmesi sirasinda buyuk vakit kaybi oluyor.

Bundan dolayi tablomuzu bolebiliriz ve bu sekilde tekrarlardan kurtulabiliriz. Mesela bolum kismini ayri bir tablo olarak aliriz ve bolum ve maasi sadece orada belirtiriz mesela.

A Lossy Decomposition

Tabloyu bolerken arada mesea bilgi kaybi olur nasil ki atiyorum ayni isimde 2 kisi var ve sen iste tabloyu ayirirken isimlerini aldin ama ID’lerini almadin iste bu durumda hangisinin hangisi oldugu anlasilmayacagi icin isler karisacak tekrar ayirdigimiz tabloyu birlestirdigimizde de fazla veri olacak cunku her iki ihtimal icin de hersey ikisine de denenecek ve yazilacak.

- Parcalandiktan sonra birlestiginde ne cikacagini dusunerek tablolar parcalanmalidir. En onemli ozellik.

Ilk Normal Form (1NF)

- Tablolarin baglanti kurulabilir sekilde tasarlanmasi. Yani tablolari bolduk ama bunlar hicbir sekilde birbiriyle baglantisi olmayan tablolar haline geldi. Simdi ilk etapta iste dusunursek daha guzel oldu verilere daha hizli erisecez falan diye ama bunlar tekrar birlesmeye kalksa birlesemeyecek ve bu bir problem.

- Veritabaninizda tekrar eden veriler varsa bunlari kaldirmak

Mesela iste “mucahit aktepe 05416219448” daha sonra yeni bir numara eklenmek istediginde bir daha ayni sekilde “mucahit aktepe 05419421675” bu kaydi ekleyecegiz ama gereksiz yere bir daha mucahit ve aktepe bilgilerini eklemek zorunda kaliyoruz. Iste burda tekrar etme problemi var. Bunu engellemek icin

Mesela her telefon numarasi icin ayri bir kolon tutulabilir bu sayede atiyorum fazladan telefon numarasi olmayan kisilerin o alanlari da null olarak kalir. Bosu bosuna hafizayi kaplamayiz. Sevmedigimiz bir cozumdur cunku daha gelismisler var gorecegiz ilerde. Bu sadece 1NF icin gecerli.

Baska bir cozum olarak kisiler ve telefonlar olarak 2 farkli tablo yapabiliriz. Foreign key olarak da ID kullaniriz. Ama mesela simdi de telefonlar tablosunda o kisiye 2. Telefon numarasi eklenirken yine ID tekrar edecek yani bir repeatition olacak ama biz bunu tekrar olarak kabul etmiyoruz.

Tek bir kolonda tekrar varsa onu tekrar olarak kabul etmeyecegiz. (Ise alimda falan cart curt cok cikar bu tip sorular) Birden fazla kolonda olursa yukaridaki gibi (isim, soyisim) bu tekrar olur.

Ikinci normal sekil (2NF)

Ilk normal forma uyacaktir. Ek olarak candidate key bulunma zorunlulugu vardir.

Birden fazla kolon tekrar ediyorsa sikinti var demektir. Mesela bolumler var ve onlari dahili telefon numaralari var simdi isim soyisim iste bolum falan ekledigimizde hem bolum hem de dahili telefon tekrar edecek yani mesela muhasebe bolumunun telefonu sabit herkes icin ayni hem ora tekrar edecek hem muhasabe kismi bolum kismi yani bu durum problem demektir.

Bunlari 2 tabloya ayirirsak mesela calisanlar (isim, soyisim) ve bolumler (bolum, dahili tel) diye aradaki iliski kaybedilir.

Ama mesela 2 tabloyu su sekilde yaparsak calisanlar (isim, soyisim, kisim) ve bolumler (kisim, bolum, dahili tel) bu sekilde hem iliski korunmus olur hem de 1 kolon tekrar ettigi icin problem kalkmis olur. 2NF icin iyi bir cozum degildir.

Veya baska bir cozum olarak primary key olacak sekilde bir ID tanimlariz ve bu sekilde olayi cozeriz.

Ucuncu normal sekil(3NF)

Functional Dependency = Bir attribute baska bir attribute ile “define” edilebiliyorsa yani tanimlanabiliyorsa buna fonksiyonel bagimlilik denir. Mesela yukaridaki ornekten her bolumun bir dahili telefonu varsa ben atiyorum muhasebe bolumunu onun dahili telefonu ile de belirleyebilirim cunku o dahili telefon sadece muhasebe bolumune ait ve baska bolumde aynisi yok. (unique)

Mesela Primary Key varsa bir tabloda tum attribute’ler primary key ile fonksiyonel bagimlidir.

X 🡪 Y ( X, Y’ye fonksiyonel bagimlidir demektir)

X 🡪 Y, Y 🡪 Z (X’in Y’ye Y’nin de Z’ye bagli olmasi durumunda X’in Z’ye bagli olmasi durumu gecisli bagimliliktir)

- 2. Normal sekli tanimlamak zorundadir. (Her yeni gelen bir oncekinin uzerine katarak gidiyor ozetle)

- Anahtarlar birbirine fonksiyonel olarak bagliysa ve gecisli bagimlilik varsa bunlari istemiyoruz bunlari ortadan kaldiracaz. Tek basina fonksiyonel bagimlilik olabilir ama birlikte olmaz.

Simdi dusunelim bir tablomuz var (Ogrenci adi, dogum tarihi, dersin adi, ve kredisi). Burada dersin adi ve kredisi tekrar eden kolonlar dolayisiyla bir problem var.

Oncelikle fonksiyonel bagimliliklari cikartacagiz. (Adim adim once 3NF’ye sonra 2NF ve 1NF ye bunu uyduracagiz.)

Ornek fonksiyonel bagimlilik olarak mesela dersin adiyla dersin kredisi arasinda fonksiyonel bir bagimlilik vardir. Dersin adindan kredisine ulasabiliriz ama dersin kredisinden dersin adina erisemeyiz cunku ayni kredili farkli dersler olabilir. Dolayisiyla burada 2 yonlu fonksiyonel bagimlilik yok sadece tek yonlu var. (Dersin adi dersin kredisini define ediyor)

Ogrenci Adi ve Dogum Tarihiyle Dersin adina erisebiliyorsak ve dersin adiyla da dersin kredisine erisebiliyorsak bu durumda burda gecisli fonksiyonel bagimlilik vardir. (A🡪B , B🡪C)

Bu tabloyu ikiye bolersek ogrenci adi, dogum tarihi ve dersin adi 1. Tablo, Dersin adi ve kredisi ise 2. Tablo olacak sekilde ayri iki tablo yaptigimizda bu problemden kurtulmus oluruz ve bu sayede (1NF, 2NF ve 3NF saglanmis olur)

Boyce Codd Normal Form

- 3NF’ yi saglamak zorundadir.

- A🡪B seklinde bir fonksiyonel bagimlilik varsa B primary key olmak zorundadir.

- 3NF tasariminda A anahtari bir aday anahtar(candidate key) olmak zorunda degildir ancak BCNF seklinde bunun tersine A🡪B seklindeki bir fonksiyonel bagimlilik durumunda A bir aday anahtar olmalidir.

\*\*\* Bu konular hocanin sitesinde var ordan bakabilirsin direkt. (bilgisayarkavramlari)

Musteri No, gorusme tarihi, gorusme saati, personel no ve oda no. bu kolonlara sahip bir tablomuz var elimizde.

Oda no, personel no, musteri no tekrar ediyor problem var.

1. Musteri No, Gorusme Tarihi 🡪 Gorusme saati, personel no ve oda no

2. Personel no, gorusme tarihi, gorusme saati 🡪 musteri no

3. Oda no, gorusme tarihi, gorusme saati 🡪 musteri no, personel no

4. Personel no, gorusme tarihi 🡪 oda no (bu bir cozum degildir)

Tablomuz yine 2 parcaya ayriliyor.

- Musteri no, gorusme tarihi, gorusme saati, personel no

- personel no, gorusme tarihi, oda no

Multi Valued Dependency - MVD

Sayet herhangi bir kolon, ayni kalmak sartiyla, ikinci bir kolondaki veriler degisiyorsa, bu durumda ilk kolonun ikinci kolonun uzerinde coklu degerle ifadesi soz konusudur.

Denormalization

Diyelim ki en mukkemmel normalization’I yaptin ama yine de mukemmel birsey yapmamis olabilirsin. Atiyorum hafizadan cok guzel kazanmissindir ama hizdan kaybetmissindir falan.

Bazen normalize etmek istemeyiz tamamen, denormalize etmeyi tercih ederiz cunku veriye hizli erismemiz gereklidir. Mesela google big table veriler normalzie edilmemistir arama sonuclarina ordaki anahtar kelimelere veritabaninda hizlica erisebilmek icin.