Database Management

Veritabanı Nedir?

→ Birden çok uygulama tarafından kullanılan,
→ Gereksiz yinelemelerden arınmış,
→ Düzenli bir şekilde saklanan,
→ Birbiriyle ilişkili (uyumlu olarak),
→ Sürekli, fakat statik olmayan,
→ Belirli bir amaç için bir araya getirilmiş,
→ Veri Topluluğu'dur.

→ VT gerçek zamanlı çalışmalıdır.
→ VT büyüklüğü için bir kısıtlama yoktur.

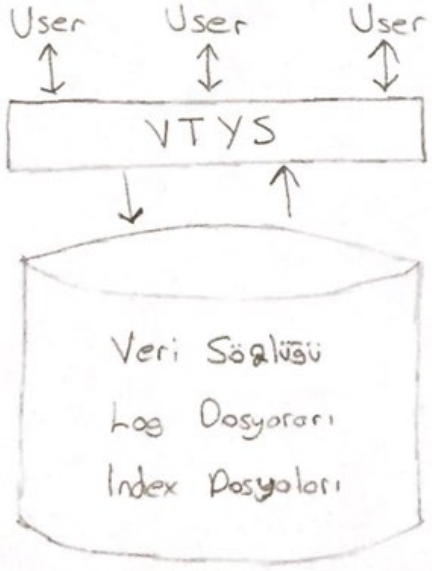
örnek: Facebook (2013)
- 300 petaB (~3.10^5 B)
  - Günlük 4 petaB
- 2.45 milyar aktif kullanıcı
- 30k sunucu (2010)

Veri tabanı yönetim sistemi nedir? (VTYS)

- VTYS genel olarak şu olanakları sağlar:
  - VT tanımlanması, gerçeklenmesi, kullanımı, paylaşımı,
  - Kontrollü veri tekrarı,
  - Sorgu işlemede verimli erişim methodları kullanımı,
  - Çoklu kullanıcılı hizmet, veri kurtarma ve yedekleme imkanı,
  - Farklı kullanıcı arayüzlerine imkanı,
  - Üst seviyeli karmaşık iş kısıtlamalarının tanımlanması, gerçeklenmesi ve sağlanması,
  - Güvenlik tanımlamaları ve sağlanması
  - PostgreSQL, MySQL, Oracle, DB2

- VT sistemine, gerek işletim sistemi gerek diğer kullanıcılar doğrudan erişemez; ancak VTYS üzerinden erişebilir.

Genel VT/VTYS Yapısı



Veri Sözlüğü: Veritabanı tanımlarının (metadata) saklandığı dosyalardır.

Log Dosyaları: Verinin doğruluk ve kurtarılması amaçlı dosyalardır.

Index Dosyaları: Fiziksel erişim dosyalarıdır.

Bazı Terimler

Uygulama Programı: Veri için istek veya sorgu göndererek VT'ye erişim yapan program.

Sorgu: VT'deki verilerin alınmasına (retrieve) neden olan işlem.

Transaction: Bazı verilerin VT'den okunmasına ve bazı verilerin VT'ye yazılmasına neden olan hareket, iş.

Veri Modeli

- Gerçek dünya verilerini kavramsal ve mantiksal seviyede düzenlemek için kullanılan yapı ve kavramlar bütünü olarak tanımlanır.

- E-R modeli, (E)ER, UML
- İlişkisel model (RM), object Model, Object-Relational Model, XML

- Genel VT tasarımı;
  1. Kavramsal Tasarım
  2. Mantiksal Düzenleme, Varlık ve Bağıntıların Belirlenmesi
  3. Veri tipleri, değer aralığı, uzunluk belirlenmesi
  4. Veri bütünlüğü kısıtlamalarının belirlenmesi
  5. Fiziksel tasarım tercihleri (Index)
  6. Kullanıcıların belirlenmesi ve güvenlik ayarları

Veri Modellerinin Kategorileri:

- **Kavramsal Modelleme :** Kullanıcıların veriye erişiminin konseptini tanımlar.
- **Mantiksal Modelleme :** Veriler ve aralarındaki ilişkilerin modellenmesidir.
- **Fiziksel Modelleme :** Verilerin boyutlarıyla beraber değerlendirilerek modellenmesidir.

VT Kullanıcıları

- VT Sorumlusu (Admin, DBA)
  → VT tasarımından işletimine kadar her şeyinden sorumlu kişiller).
  → VT erişim yetkilerini tanımlama ve kullanımı kontrol
  → Sistem için gerekli yazılım ve donanım desteğini belirler.
  → Güvenlik için yedekleme ve kurtarma işlemleri

- Tasarımcı (Designer)
  → Verinin her aşamada modellenmesi (yapısı, içeriği ve kısıtlamaları ile).
  → Gerçekleme öncesi aşamalardan sorumludur.
  → VT üzerinde fonksiyon ve işlemleri tanımlar.
  → Veriler için uygun yapıları belirleme.

- Sistem Çözümlegici (System Analyst)
  → Son kullanıcıların gereksinimlerini belirler.

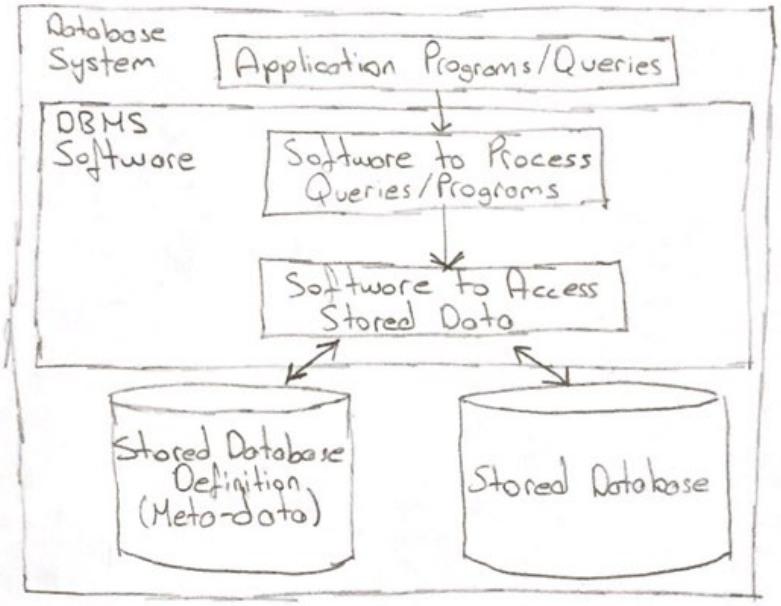
- VT Uygulama Yazılımcısı
  → VT erişimi yapacak uygulamaları tasarlam ve gerçekleme

- Son Kullanıcılar

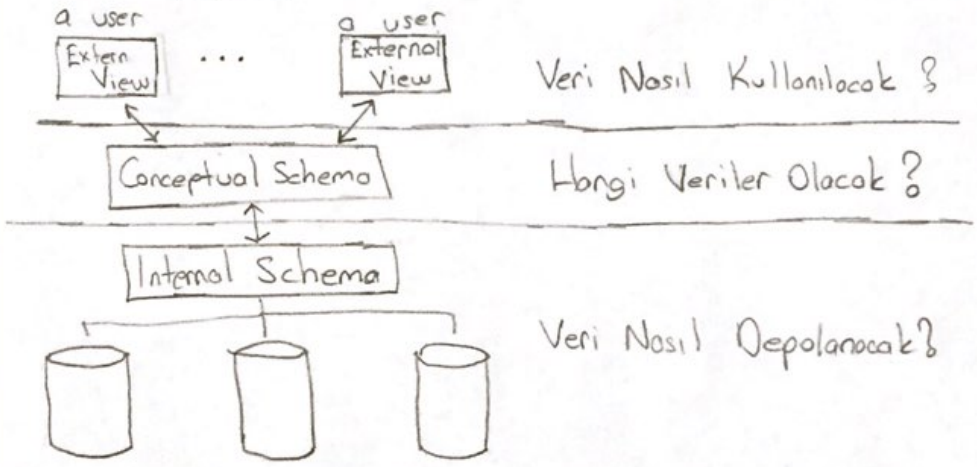
VTYS Dilleri

- Data Definition Language (DDL)
  → Konsept şemanın belirlenmesinde kullanılır
- Data Manipulation Language (DML)
  → Şema içerisinde bulunan verinin değiştirilmesi, silinmesi ve eklenmesinde kullanı

VTYS Ana İşlevleri



3-Şema Mimarisi



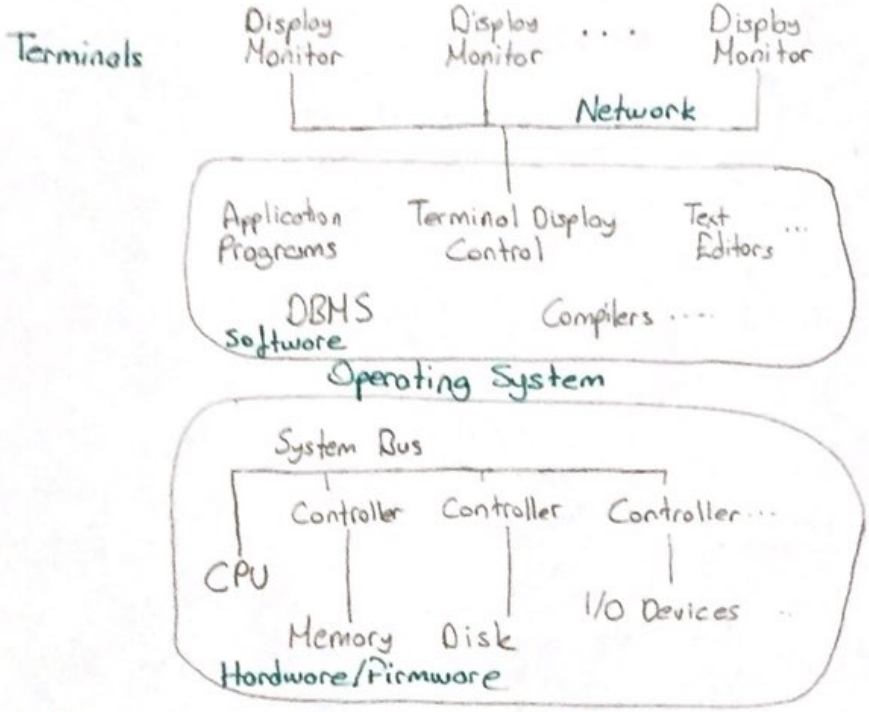
Veri Bağımsızlığı

Fiziksel Veri Bağımsızlığı:
Mantiksal şemayı değiştirmeden fiziksel şema üzerinde değişiklik yapılabilmesi.

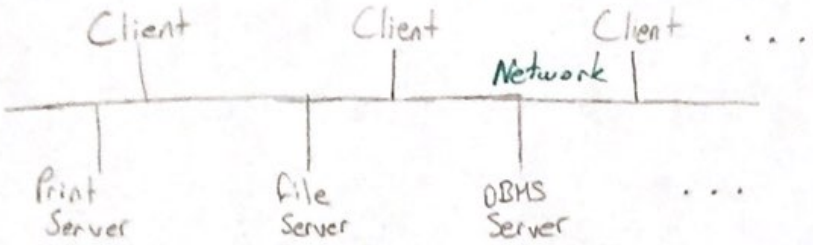
Mantiksal Veri Bağımsızlığı:
Dış şemaları ve herbiri uygulama verilerini değiştirmeden mantiksal şema üzerinde değişiklik yapılabilmesi.

Merkezi VTYS Mimarisi

- VTYS yazılımı, donanımı, üzerinde çalıştığı makine donanımı, işletim sistemi, kullanıcı arayüzleri ile tüm sistemleri tek bir sistem olarak kullanma.
- Kullanıcılar uzak terminallerden bağlantı kurabilirler.



2-Katmanlı Sunucu-İstemci VTYS Mimarisi



VTYS Türleri

- Veri Modeline Göre
  - → Tradational : İlişkisel, Ağ tabanlı, Hiyerarşik
  - → Emerging : Nesne yönelimli, Nesnel ilişkili, XML
- Diğer Türler
  - → Tek-kullanıcılı vs Multi-kullanıcılı
  - → Merkezi vs Dağıtık
- Homojen VTYS'ler
- Heterojen VTYS'ler
- Çoklu VTYS'ler

VT Tasarımı

- Gereksinimlerin toplanması ve analizi (Requirements collection and analysis)
  - → VT Tasarımcısı, veri gereksinimlerini anlamak ve belgelemek için VT kullanıcıları ile görüşme yapar
  - → Çıktı : Veri Gereksinimleri
  - → Uygulama için işlevsel gereksinimler (functional requirements)
- Kavramsal Şema
  - → Kavramsal bir tasarım
  - → Veri gereksinimlerini açıklamalı
  - → Varlık türleri detaylı açıklama (entity types)
  - → Bağıntıları detaylı açıklama (relationships)
  - → Kısıtlamaları ayrıntılı açıklama (constraints)
  - → Üst-düzeyli veri modelinden gerçekleştirilecek veri modeline dönüşüm.
- Mantıksal tasarım veya Veri modeli eşlemesi (Data Model Mapping)

Varlık Bağıntı (Entity-Relationship, ER) Modeli

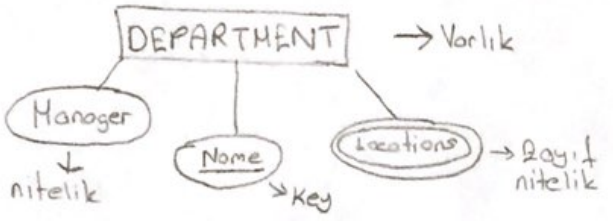
- ER modelde verileri tanımlama :
  - Varlıklar (Entities)
  - Bağıntılar (Relationships)
  - Nitelikler (Attributes)

- UML diyagramları (UML notasyonu):
  - Yazılım geliştirme metodolojisinde kullanılır.
  - ER'deki varlık UML nesnesine denk düşer.
  - UML nesnesinde 3 kısım var :
    isim, nitelikler, operasyonlar

Varlık / Bağıntı Kümeleri

- Var olan ve diğerlerinden ayırt edilebilen her nesneye varlık denir.
- Benzer varlıkların oluşturduğu kümeye varlık kümesi denir.
- Varlıkların nitelikleri vardır.

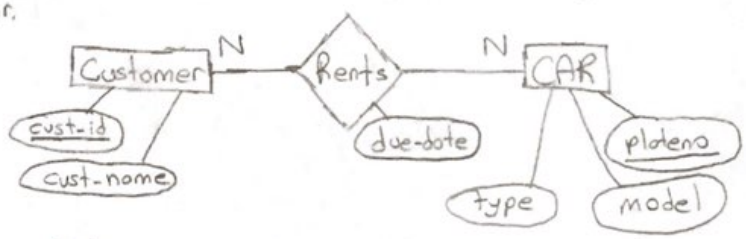
Temel Yapı Taşları : Varlık, Bağıntı



Bir dersi N öğrenci olabilir. $N \geq 0$
Bir öğrenci N tane ders olabilir. $N \geq 0$

Bağıntı Küme Sınırlamaları

- Bağıntı ismi : Manaya uygun bir isim.
- Bağıntıdaki varlıklar: 2 veya daha çok, aynı veya farklı varlıklar arasında
- Bağıntının derecesi : 2, 3, ...
- Eleman sayıları: 1-1, 1-N, N-1, N-N
- ROL tanımlama : Bağıntıdaki varlıkların bağıntıdaki işlevlerini belirleyen bir rolleri vardır.



(N müşteri, N tane araba kiralayabilir.)
(N tane araba N tane kullanıcı tarafından kiralanabilir.)
Bu kiralamlarda due_date değişebilir. $N \geq 0$

Examples

Entity — Employee

Week Entity — Employee-job-history

Relationship — works-in

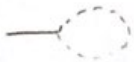
identifier (key) — emp_id

descriptor (non-key) — emp-name

multivalued descriptor — degrees

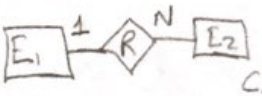
Complex attribute — address (street, city, state, zip)

Composite Attribute

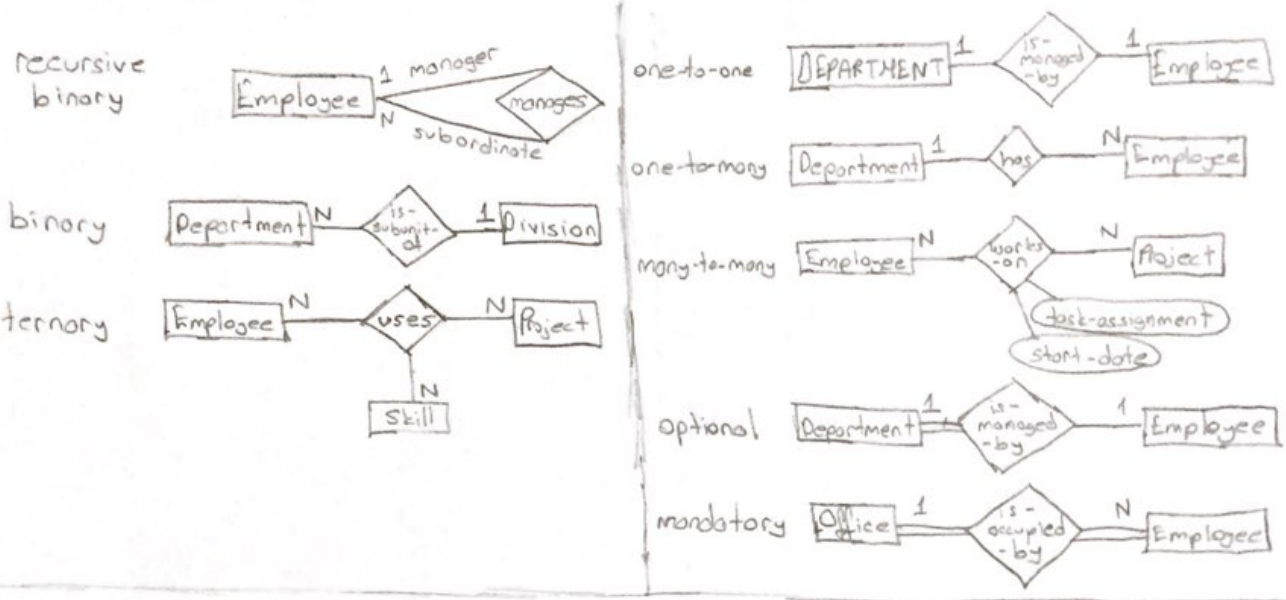


Derived Attribute

total participation of E_2 in R



Cardinality Ratio 1:N for E_1:E_2 in R



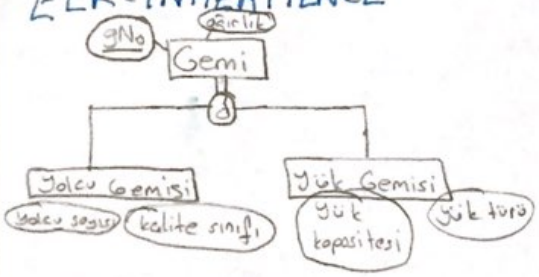
Anahtar ve Güçlü / Zayıf Varlık Kümeleri

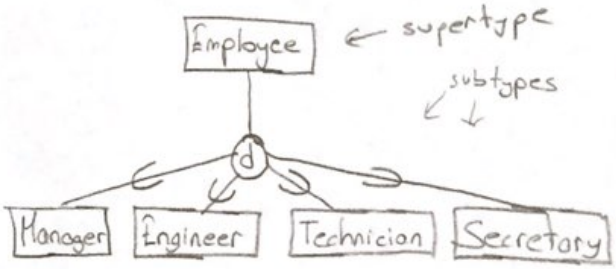
- Varlık/Bağıntı kümesindeki varlık/bağıntıları birbirinden ayırt etmek için kullanılan nitelik veya nitelik grubuna anahtar denir.
  - → Super Anahtar (Superkey): (ÖĞRENCİ_NO, AD, SOYAD)
  - → Aday Anahtar (Candidate Key): ÖĞRENCİ_NO

- Hiçbir anahtar yoksa zayıf varlık olarak adlandırılır.

- Zayıf varlık kümesi, güçlü bir varlık kümesi ile 1-1 veya 1-N'lik var olma bağımlı bir bağıntı kurmalıdır.

EER-INHERITENCE





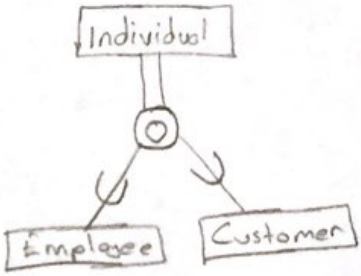
~~ER (disjoint, total)~~

ER (disjoint, total)

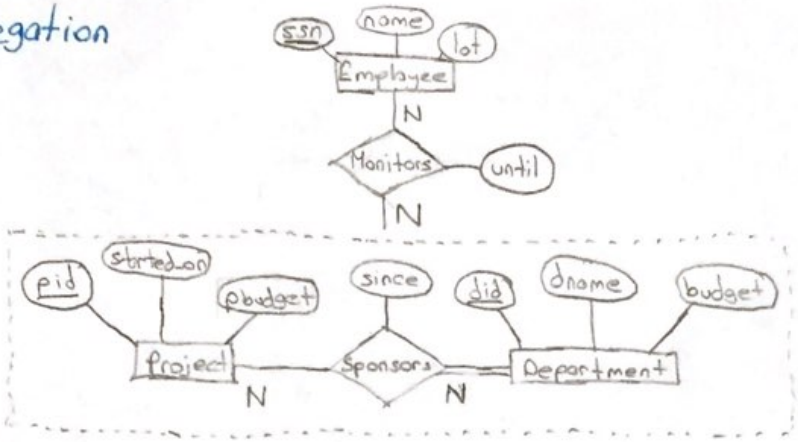
ER : (disjoint, partial)

d → AND

o → OR



ER-Aggregation



(UML'i biliyor diye çizmedim ss aldım.)

Örnek1: Company DB

- Şirket DEPARTMENT'lardan oluşur.
  - Her departmanın bir ismi, numarası ve yöneticisi vardır.
  - Yöneticisinin işe başlama tarihini tutmak isteriz.
  - Bir departmanın birden fazla konumda lokasyonu olabilir
- Her departman belirli sayıdaki PROJECT'i kontrol eder.
  - Her projenin eşsiz(unique) bir ismi, unique bir numarası ve tekil bir lokasyonu vardır.
- Her bir EMPLOYEE'nin aşağıdaki verilerini depolarız.
  - SSN, Adres, Maaş, Doğum Tarihi
- Her employee bir departmanda çalışmak zorundadır ama hiç veya birden fazla projede çalışabilir.
  - Her employee'nin hangi projede kaç saat çalıştığı bilgisini tutarız.
- Her employee'nin yine bir employee olan supevisor'u vardır.
- Her employee hiç veya birçok DEPENDENT'a sahip olabilir.
  - Her bir dependent için aşağıdaki bilgiler tutulur.
    - → İsim, Cinsiyet, Doğum tarihi, İlişki türü (relationship)

Başlangıç Tasarımı

DEPARTMENT
number BA
name
manager
managerStartDate
locations

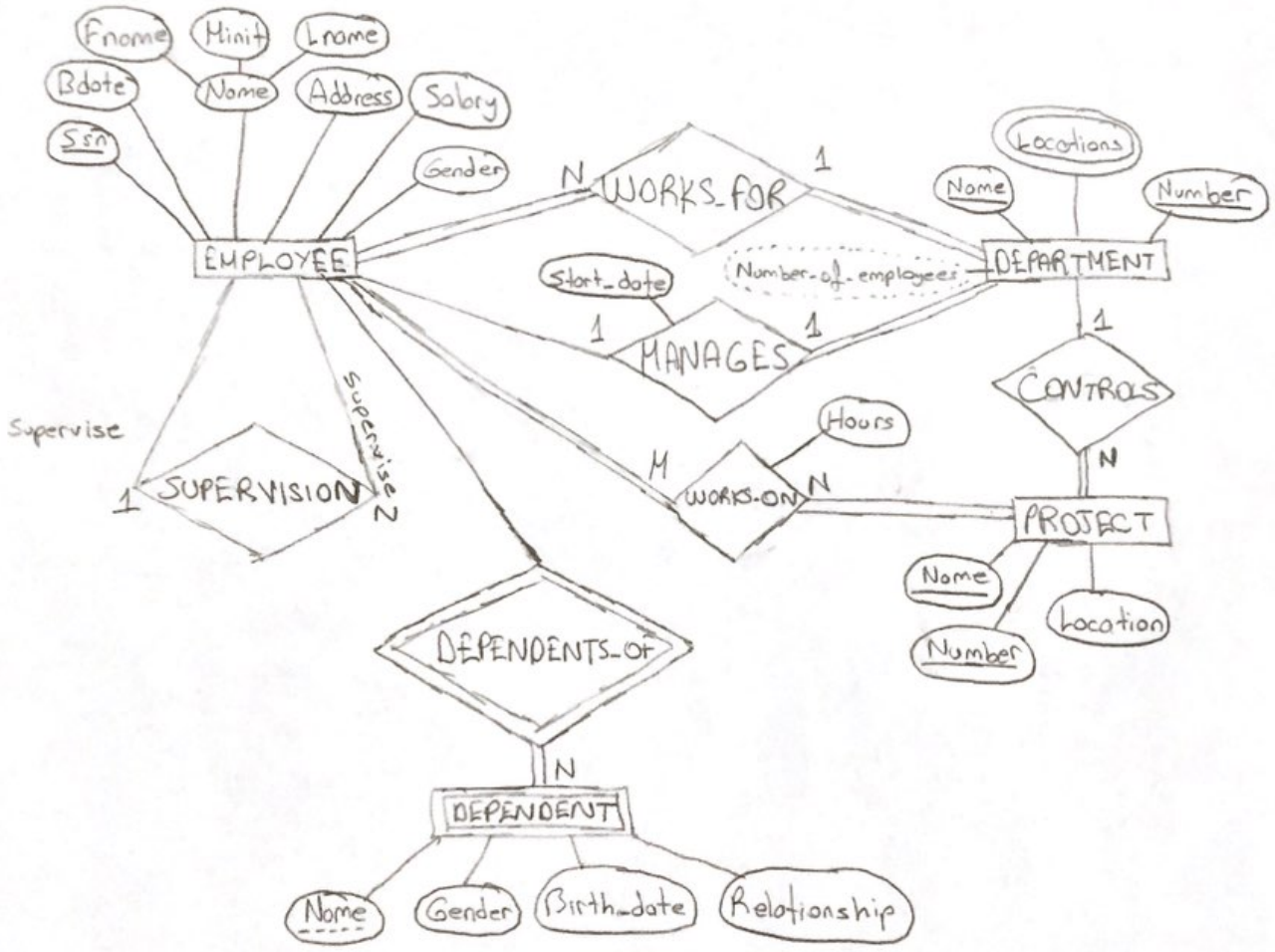
PROJECT
number BA
name
locations
controllingDept.

EMPLOYEE
Ssn BA
fname
lname
department
birthDate
salary
worksOnProjects
supervisor
address

DEPENDENT
name
dependsToEmp
birthDate
relationships

İlk Tasarım Sonrası İyileştirmeler

- Eğer mevcut nitelik başka bir varlığa işaret ediyorsa:
  - Nitelik → Bağıntı haline dönüşmeli
- Eğer mevcut nitelik çok değer alabiliyorsa:
  - Nitelik → Varlık kümesi haline dönüşmeli
- Eğer mevcut varlık kümesi sadece 1 niteliğe sahip ve bir bağıntısı varsa:
  - Varlık kümesi → Nitelik haline dönüşmeli
- Varlık kümeleri arasında "genelleme" (kalıtım) mümkünse yapılmalı.
- Bağıntıların dereceleri tekrar değerlendirilmeli, gerekirse "kümeleme" ile daha açık yapılamalar kullanılmalı.
- Eğer bir bağıntı diyagramdan çıkarıldığında bilgi kaybı olmuyorsa buna fazladan gereksiz (redundant) denir.



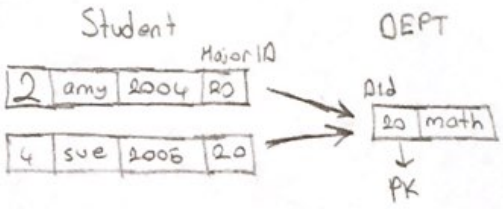
İlişkisel Modeli

- İlişkisel modelde bildirim (declorative) esaslı veri işlenir.
- Her ilişkinin biricik (unique) bir adı vardır.
- Satırı diğer satırlardan ayıran yegane özellik anahtardır(key).
  - Vatandaş, TC Kimlik No - ID
- Verilen $R(A_1, A_2, \ldots, A_n)$ ilişkisinde:
  - $r(R) \subset dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$; $r(R)$: Relation state
  - $r(A_1, A_2, \ldots, A_n)$ ise ilişkinin şemasını oluşturur.
  - $r(R)$: R ilişkisinin spesifik bir state'idir ("value", "population") ; satırların kümesidir. (tuples)
  - $r(R) = \{t_1, t_2, \ldots, t_n\}$ n satırın kümesi
    - $t_i = \langle v_1, v_2, \ldots, v_n \rangle$ her v_j $dom(A_j)$'nın bir elementidir.
- $R(A_1, A_2)$ bir ilişkisel şema olsun:
  - $dom(A_1) = \{0,1\}$, $dom(A_2) = \{a,b,c\}$
  - $dom(A_1) \times dom(A_2)$ ifadesinin olası kombinasyonları:
    - $\{\langle 0,a\rangle, \langle 0,b\rangle, \langle 0,c\rangle, \langle 1,a\rangle, \langle 1,b\rangle, \langle 1,c\rangle\}$ → bütün alt kümeleri bir $r(R)$'dir.
    - Relation state $r(R) \subset dom(A_1) \times dom(A_2)$
- Tüm değerler atomic (indivisible) olarak değerlendirilir. (Daha fazla alt parçaya bölünemez)
- Bilinmeyen değerler için NULL kullanılır.

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

Referential Integrity Constraints

- İki ilişki arasındaki kısıtlardır.
  - Referencing relation
  - Referenced relation
- Triggers (Semantic Constaints'leri kontrol eden yapı)
- Semantic Integrity Constraints
  - Örnek; "Bir projede çalışanların haftalık çalışma saati max 56 olabilir."
- Foreign Key
  - Referans verilen ilişkideki FK ya null olabilir ya da referans verilen Tablonun karşılık gelen PK'i.



ER-to-Relational Mapping Outline

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types
- Foreign key approach
- Merged Relation Option
- Cross-Reference or Relationship Relation Option

Step 4: Mapping of binary $\underset{(N:1)}{1:N}$ Relationship Types.

Step 5: Mapping of binary M:N Relationship Types.

Step 6: Mapping of multivalued attributes.

Step 7: Mapping of N-ary Relationship Types.

İlişkisel Cebir (Relational Algebra - RA)

- SQL endüstriyel standartken, RA'nın herhangi bir standartı yoktur!!!
- SQL: Sonuç odaklı tanımlayıcı bir sorgulama dili
- RA: Görev odaklı prosedürel bir sorgulama dili.
- VTYS'ler SQL sorgusunu çalıştırmak için RA'ya dönüştürür.
- RA bir sorgu ağacı olarak ifade edilebilir.
  - Sorgu içerisindeki tablolar ve operatörler için çalıştırma sırası belirleme

Operatörler

- Select: σ
- Project: Π
- Sort: S
- Rename: ρ
- Extend: E
- Aggregate: $\mathcal{F}$
- Group by: $\mathcal{F}$
- Union: $\cup$
- Intersection: $\cap$
- Set Difference: $-$
- Division: $\div$
- Product: $\times$
- Theta-join: $\bowtie$
- Natural-join: $*$ or $\bowtie$
- Semijoin: $\ltimes$
- Antijoin: $\triangleright$
- Full Outer join: $\overset{\bowtie}{}$
- Left Outer Join: $\overset{\bowtie}{}$
- Right Outer Join: $\overset{\bowtie}{}$

SELECT işlemi

- Notasyon: $\sigma_p(r)$
  - p: seçim kriteri
  - $\wedge$(and), $\vee$(or), $\neg$(not) işlemleriyle birleşmiş
- Tanım: $\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$
- $\{\langle attribute \rangle\}$ op $\{\langle attribute \rangle$ or $\langle constant \rangle\}$
  - op: $>, \geq, <, \leq, =, \neq$

Örnek

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \text{ ve } D>5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

PROJECT işlemi

- Notasyon: $\Pi_{A_1, A_2, \ldots, A_n}(r)$
  - $A_1, A_2, \ldots$ özellik adları
  - r: ilişki adı
- Sonuç listelenmemiş kolonların silinmesinden sonra ortaya çıkan
  - ilişkilerde çoklu satırlı sonuçlar silinir

Örnek

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
β	1
β	2

A ve C'yi al, birbirine benzer sütun varsa çıkar

RENAMING İşlemi

- Notasyon: $\rho_{oldName \to newName}(r)$

Örnek

$\rho_{Father \to Parent}(Paternity) \cup \rho_{Mother \to Parent}(Maternity)$

→Father ve Mother sütunlarının adını Parent yapıp bunları birleştiriyor.

Theta-JOIN İşlemi

- Türetilmiş bir işlemdir.
- Notasyon: $r_1 \bowtie r_2 = \sigma(r_1 \times r_2)$

Örn

Employee	Project
Smith	A
Black	A
Black	B

Employees

Code	Name
A	Venus
B	Mars

Projects

Employees $\bowtie_{project=code}$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	B	Mars

Natural-JOIN işlemi

- Notasyon : $r_1 * r_2$ veya $r_1 \bowtie r_2$
- Theta-JOIN'e benziyo, aynı isimli kolonlar üzerinden birleştirip unique değerleri alıyoruz.

(Diğer joinler ss alındı.)