VERİ TABANI DERS NOTLARI

Ders #3: Relational Model (İlişkisel Model)

References

- ER examples from *"Elmasri, Navathe*, Fund. of Database Systems, 5th ed., Addison Wesley"

Konu Başlıkları

- İlişkisel Veri Modeli (Relational Model)
- İlişkisel Model'de Kısıtlamalar (Constraints)
- İlişkisel Model'de Şema (Schemas) ve Olgu
- Dönüştürme: Varlık-Bağıntı Modeli (E)ER → İlişkisel Model (RM)
  - **Varlık Kümelerini dönüştürme**
  - **Bağıntı Kümelerini dönüştürme**
  - **Genellemelerin dönüştürülmesi**
  - **Kümelemelerin dönüştürülmesi**

Genel Bakış:

- İlişkisel Model, bildirim (declarative) esaslı veri işlemeye imkan sağlayan bir modeldir.
  - Bildirim esaslı model, kullanıcıya «NE İSTEDİĞİNİ» aktarma imkanı sağlar. Verinin nasıl yerleşeceği veya nasıl erişileceği ile ilgilenilmez. Diğer bir ifade ile, donanım ve gerçekleştirim ortamından veri bağımsızlığı *(data independence)* sağlanmış olur.
- İlişkisel model gerçeklemeleri SQL isimli veri işleme (tanımlama / sorgulama) dili kullanır.
- Model, ilk geliştiricisi **Edgar F. Codd** (IBM Research) tarafından şu klasik makalede tanıtılmış:"**A Relational Model of Data for Large Shared Data Banks**," Communications of the ACM, June 1970.Bu makale, veritabanı yönetimi alanında inkilap niteliğinde sonuçlara yol açmış ve yazarına (Dr. Codd) **ACM Turing Award** ödülünü kazandırmıştır(1981)
- Öncü (tarihi) ilişkisel VTYS örnekleri: System R (IBM) ve Ingres (UC-Berkeley)

İlişkisel Modelde Kavramlar (Concepts)

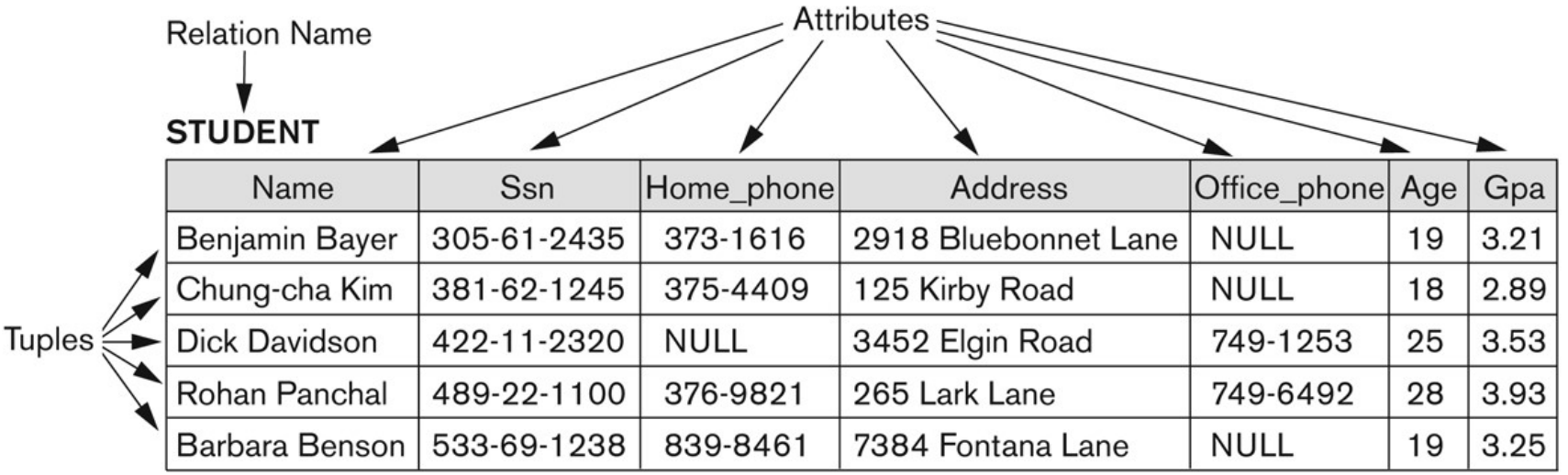
- İlişkisel veri modeli, ilişki (r*elation)* kavramı üzerine bina edilmiş. İlişki (r*elation*) kavramı, özde matematiksel bir kavram ve küme (*set*) kavramı üstüne bina edilmiş.
  - Veri yönetiminde ilişkisel yaklaşımın gücü, ilişkiler teorisi ve biçimsel temellere oturmuş olmasından kaynaklanmaktadır.
  - Pratikte, SQL tabanlı standart bir model var.
  - Fakat, biçimsel model ile pratik model arasında çok önemli farklılıklar var. Biçimsel modelin tamamen gerçeklenmesi oldukça zordur ve çok nadirdir.

- İlişkisel veri modeli, kullanıcıların VT'nındaki bütün «veri ve bağıntıları» tablolar (çizelgeler) biçiminde görmesine izin verir. Böylece, "ilişkisel modele dayalı bir VT'nın temel yapı taşı TABLO (ÇİZELGE)dir" denilebilir.

İlişkisel Model (Relational Model, RM)

- RM verileri, ilişkiler topluluğu olarak modeller.
- **İlişki (relation):** iki boyutlu bir değerler tablosu (**table** of values).
- İlişkide, satır kümesi var (**set of rows**). Satırlar **tuples-çoklu** diye de adlandırılır.
- Satır (çoklu) tekrarına izin verilmez. Satırlar (çoklular) bir set oluşturur, o zaman sıralı olması koşulu yoktur.
- Her satırdaki veri elemanları belirli gerçekleri (**nesne** yada **bağıntı**) gösterir (**entity** or **relationship**).
- Her kolon (**column**) bir niteliği gösterir. Kolon başlığı, o kolondaki verinin anlamını belirtir (nitelik-attribute).
  - Nitelik yalın değer içermeli yani çok-değerli niteliğe izin yok!
  - Her nitelik bir tanım kümesi (domain, type) elemanlarından bir değer alır.
  - Niteliklerin satır içindeki sıraları önemli değildir. Bu sıranın farklı olması satırları birbirinden farklı kılmaz. Örneğin: <0,a> satırı, <a,o> ile aynıdır; yeterki değerin hangi nieliğe ait olduğu belli olsun.

İlişki Örneği



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Key of a Relation

- Her ilişkinin biricik (unique) bir adı vardır.
- Key of a Relation:
  - The key uniquely identifies that row in the table. In the STUDENT table, SSN is the key.
  - Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table. Called *artificial key* or *surrogate key*

Schema-Şema

- Bir ilişki şeması (schema, description):
  - R(A1, A2, ...An) biçiminde gösterilir.
  - R ilişki adı (**name**), A1, A2, ..., An ise ilişkinin nitelikleri(**attributes**)
  - **İlişkinin derecesi:** ilişki şemasındaki nitelik sayısı
  - Example:

    CUSTOMER (Cust-id, Cust-name, Address, Phone#)

    ilişki ismi: CUSTOMER

    ilişki derecesi: 4

    Nitelikler: Cust-id, Cust-name, Address, Phone#.
- A relation is a set of tuples (rows). A tuple is an ordered set of values (enclosed in angled brackets '< … >'). Each value is derived from an appropriate domain.
  - A row in the CUSTOMER relation is a 4-tuple and consists of four values: <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404)894-2000">

Tanımlar -3

- **Domain: geçerli nitelik değer kümesi**
  - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
  - A domain has a data-type or a format defined for it.
    - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
    - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.
  - The attribute name designates the role played by a domain in a relation:
    - Used to interpret the meaning of the data elements corresponding to that attribute
    - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings
- Example: attribute **Cust-name** is defined over the domain of character strings of maximum length 25
  - **dom(Cust-name)** is varchar(25)
  - **The role** these strings play in the CUSTOMER relation is that of the *name of a customer*.
- The **relation state, r(R)** is a subset of the **Cartesian product** of the domains of its attributes
  - each domain contains set of all possible values the attribute can take.

Definitions – *example*

- Formally, given R(A1, A2, ..., An)
  - r(R) ⊂ dom (A1) X dom (A2) X ....X dom(An)
  - R(A1, A2, …, An) is the **schema** of the relation
  - r(R): a specific **state** (or "value" or "population") of relation R – this is a *set of tuples* (rows)
    - r(R) = {t1, t2, …, tn} where each ti is an n-tuple
    - ti = <v1, v2, …, vn> where each vj *element-of* dom(Aj)
- Let R(A1, A2) be a relation schema:
  - Let dom(A1) = {0,1}, Let dom(A2) = {a,b,c}
  - Then: dom(A1) X dom(A2) is all possible combinations:

    {<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }

    The relation state r(R) ⊂ dom(A1) X dom(A2)
  - For example: r(R) could be {<0,a> , <0,b> , <1,c> }
    - this is one possible state (population or extension) r of the relation R, defined over A1 and A2.
    - It has three 2-tuples: <0,a> , <0,b> , <1,c>
- All values are considered **atomic (indivisible)**. A special **null** value is used to represent values that are **unknown** or **inapplicable** to certain tuples.
- Notation:
  - We refer to **component values** of a tuple t by: t[Ai] or t.Ai This is the value vi of attribute Ai for tuple t.

Definitions – *example:* Relational Database Schema

- A set S of relation schemas that belong to the same database.
- S is the name of the whole **database schema**
- S = {R1, R2, ..., Rn}
- R1, R2, …, Rn are the names of the individual **relation schemas** within the database S

- Here is the COMPANY database schema with 6 relation schemas:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Definition Summary

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

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of constraints in the relational model:
  - **Key** constraints (*anahtar kısıtı*)
  - **Entity integrity** constraints (*varlık bütünlük kısıtı*)
  - **Referential integrity** constraints (*ima bütünlük kısıtı*)
- Another implicit constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)
- Yet another constraint is **Semantic constraints**..

1-Key Constraints

- **Superkey** of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
    - This condition must hold in *any valid state* r(R)
- **Key** of R:
  - The only way to access only 1 record.
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but *not* a key.
- In general:
  - Any *key* is a *superkey* (but not vice versa)
  - Any set of attributes that *includes a key* is a *superkey*
  - A *minimal* superkey is also a key

1-Key Constraints

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key.**
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

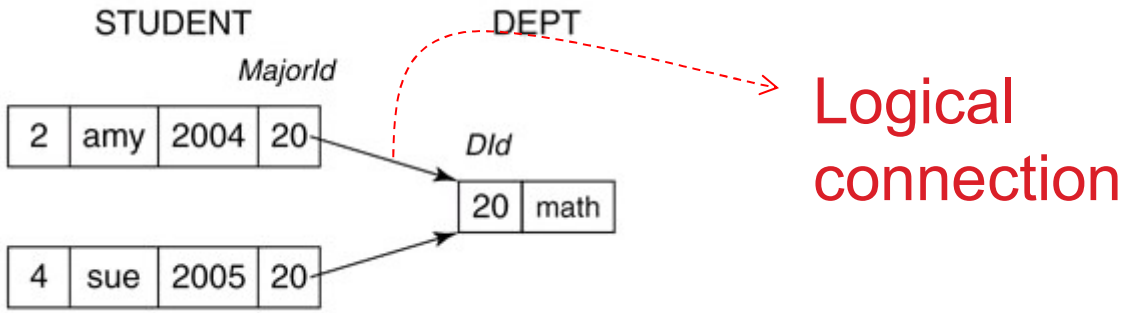
Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

2-Entity Integrity

- **Entity Integrity:**
  - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).
    - This is because primary key values are used to *identify* the individual tuples.
    - t[PK] ≠ null for any tuple t in r(R)
    - If PK has several attributes, null is not allowed in any of these attributes
  - Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key. (*UNIQUE kısıtlaması*)

3-Referential Integrity

- A constraint involving **two** relations (*The previous constraints involve a single relation.)*
- Used to specify a **relationship** among tuples in two relations: The **referencing relation** and the **referenced relation.**
- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.
- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null.**
  - *In case (2), the FK in R1 should **not** be a part of its own primary key.*



Other Types of Constraints

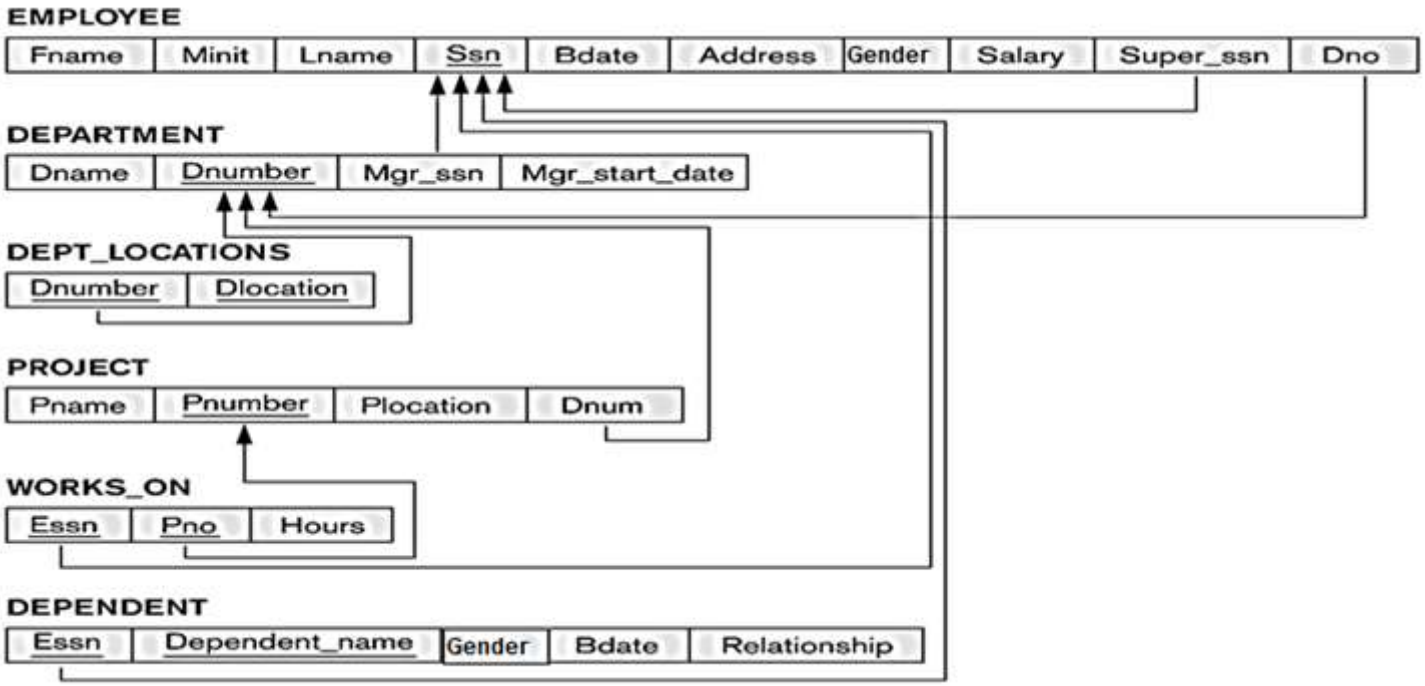
- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these. SQL-99 allows triggers and ASSERTIONS to express for some of these

Actions in case of integrity violation

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine
- Specifying constraints **in SQL**
  - NOT NULL may be specified on an attribute
  - Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases
  - referential integrity constraints (foreign keys).
  -

Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table
  - Can also point the the primary key of the referenced relation for clarity
- COMPANY **relational schema diagram:**



Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations (CRUD- Create, read, update and delete operations) for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple
- Next slide shows an example state for the COMPANY database

Populated database state for COMPANY

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Gen	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Gen	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.
- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

Specifying constraints in SQL

```
CREATE TABLE DEPT (
  DNAME   VARCHAR(10)  NOT NULL,
  DNUMBER INTEGER    NOT NULL,
  MGRSSN  CHAR(9) DEFAULT '000',
  MGRSTARTDATE      CHAR(9),
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMP
                ON DELETE SET DEFAULT ON UPDATE CASCADE);
 CREATE TABLE EMP(
   ENAME    VARCHAR(30)NOT NULL,
   ESSN     CHAR(9),
   BDATE    DATE,
   DNO      INTEGER  DEFAULT 1,
   SUPERSSN CHAR(9),
   PRIMARY KEY (ESSN),
   FOREIGN KEY (DNO) REFERENCES   DEPT ON DELETE SET
   DEFAULT ON UPDATE  CASCADE,
   FOREIGN KEY (SUPERSSN) REFERENCES EMP ON DELETE SET
                NULL ON UPDATE CASCADE);
```

ÖNEMLİ: Yukarıdaki SQL DDL tablo oluşturma komutları 2 farklı sıra için de (DEPT, EMP ve EMP,DEPT) çalıştırmak mümkün olmuyor. Çünkü imalar mevcut omayan başka bir tabloya işaret edemez. O yüzden, ima kısıtlarını temel tabloyu oluşturduktan sonra ALTER TABLE ... yardımcı komutu ile ekleyebilriz.

Aynı durum DROP TABLE için de geçerli.

Populate DB with SQL

- Aynı DİKKAT, INSERT TABLE için de geçerli. Bu sefer UPDATE komutu yardımı ile eksikler tamamlanır. O yüzden önceki sayfadaki VT'da eklemeler aşağıdaki gibi olması gerek: örneğin:

```
INSERT INTO EMP VALUES
  ('James', '888665555','10-NOV-27',null,null);
INSERT INTO EMP VALUES
  ('Franklin','T','Wong','333445555','08-DEC-45','638 Voss, Houston,
    TX','M',40000,'888665555',null);
.....
INSERT INTO DEPT VALUES ('Research', 5, '333445555', '22-MAY-78');
INSERT INTO DEPT VALUES ('Headquarters', 1, '888665555', '19-JUN-71');

UPDATE employee SET dno = 5 WHERE ssn = '333445555';
UPDATE employee SET dno = 1 WHERE ssn = '888665555';
```

Exercise-1

(Taken from Exercise 5.15)

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course: **Draw a relational schema diagram**

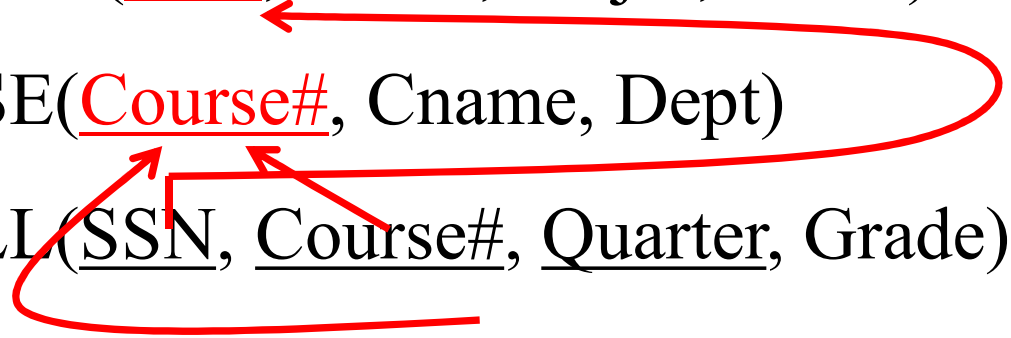
STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)



Exercise-2 : possible violation when inserting

- INSERT may violate any of the constraints:
  - Domain constraint:
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - Key constraint:
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - Referential integrity:
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - Entity integrity:
    - if the primary key value is null in the new tuple

Exercise-3: possible violation when deleting

- DELETE may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL (see Chapter 8 for more details)
      - RESTRICT option: reject the deletion
      - CASCADE option: remove all referencing tuples
      - SET NULL option: set the foreign keys of the referencing tuples to NULL
  - One of the above options must be specified during database design for each foreign key constraint

Exercise-4: possible violation when updating

- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

ER-to-Relational Mapping Outline

- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation Types
  - Step 4: Mapping of Binary 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.

- **Mapping EER Model Constructs to Relations**
  - Step 8: Options for Mapping Specialization or Generalization.
  - Step 9: Mapping of Shared Classes

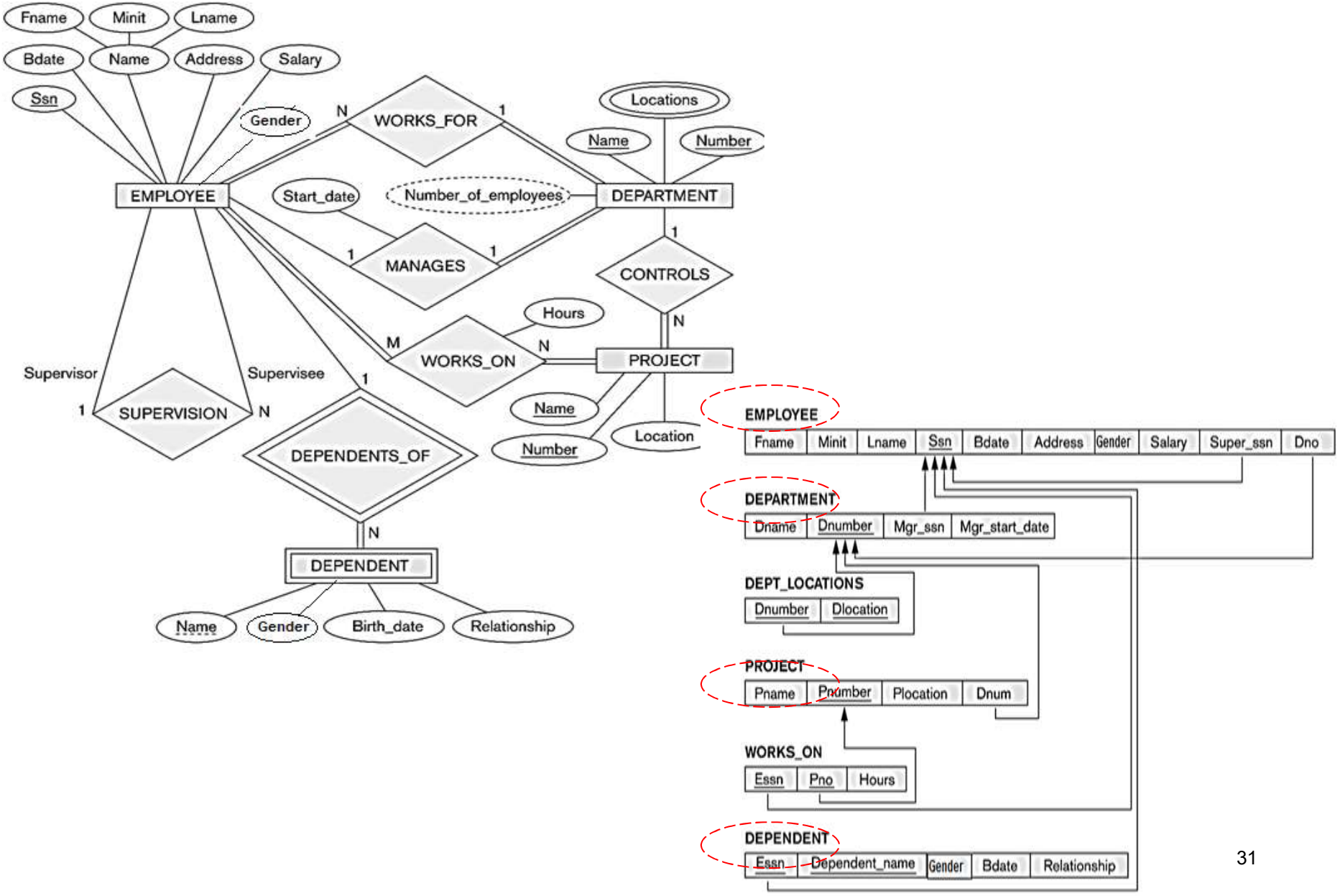
ER-to-Relational Mapping Algorithm

- **Step 1**: **Mapping of Regular Entity Types.**
  - For each regular (strong) entity type E in the ER schema, **set up a relation R** that includes all the simple attributes of E.
  - Choose one of the key attributes of E as the primary key for R.
  - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.
  - Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.

    SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.
- **Step 2: Mapping of Weak Entity Types**
  - For each weak entity type W in the ER schema with owner entity type E, **set up a relation R** & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
  - Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
  - The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.
  - **Example:** Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
  - Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN). The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

FIGURE 7.1 *from elmasri navathe*

The ER conceptual schema diagram for the COMPANY database.



ER-to-Relational Mapping Algorithm

- **Step 3: Mapping of Binary 1:1 Relationship Types**
  - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- There are three possible approaches:
  1. **Foreign Key approach:** Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. (WHY?)
     - Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
  2. **Merged relation option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
  3. **Cross-reference or relationship relation option:** The third alternative is to **set up a third relation R** for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

ER-to-Relational Mapping Algorithm

- **Step 4: Mapping of Binary 1:N Relationship Types.**
  - For each regular binary 1:N relationship type R (*from T to S*), identify the relation S that represent the participating entity type at the N-side of the relationship type.
  - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
  - Include any simple attributes of the 1:N relation type as attributes of S.
- Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
  - For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.
- **Step 5: Mapping of Binary M:N Relationship Types.**
  - For each regular binary M:N relationship type R, ***set up a new relation S*** to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
  - Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.

    The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively. Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

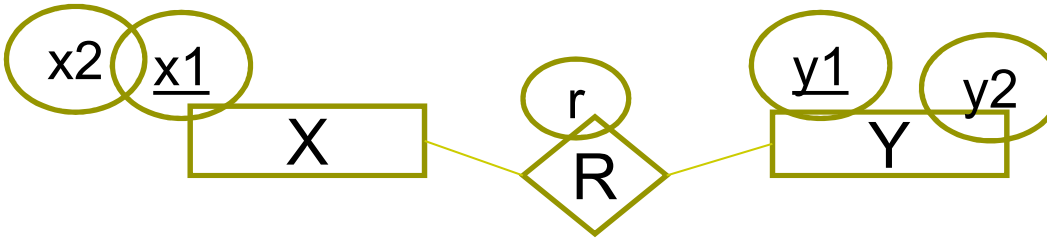
ER-to-Relational Mapping Algorithm

- **Step 6: Mapping of Multivalued attributes.**
  - For each multivalued attribute A, **set up a new relation R**.
  - This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.
  - **Example:** The relation DEPT_LOCATIONS is created.

    The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {DNUMBER, DLOCATION}.
- **Step 7: Mapping of N-ary Relationship Types.**
  - For each n-ary relationship type R, where n>2, **set up a new relation** , S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
  - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.
  - **Example:** The relationship type SUPPY in the ER on the next slide.

    This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME} in case the cardinalities are N:N:N

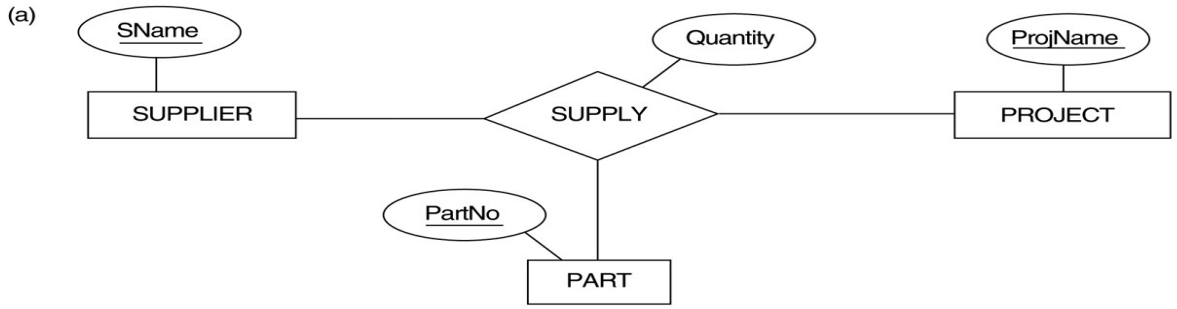
Mapping relationships:



- R: 1-1 ise aşağıdaki dönüşümler olabilir. (Altı çizili BA, üstü çizili yabancı anahtar olsun)
  1. R (x1, y1, r) veya R (x1, y1, r) : Yeni bir tablo eklediğimiz için «performans ve veri tekrarı» yönünden iyi değil.
  2. X (x1, x2, ~~Y~~, r) Y (y1, y2) : Performans ve veri tekrarı yönünden iyi.
  3. X (x1, x2) Y (y1, y2 , ~~X~~, r) : Performans ve veri tekrarı yönünden iyi.

  Sadece X'in var olma bağımlılığı «total» olsaydı, 2.çözümü tercih ederdik. Sadece Y'in var olma bağımlılığı «total» olsaydı, 3.çözümü tercih ederdik. Her ikisinin varolma bağımlılığı «total» ise 2,3 fark etmez Her iksinin varolma bağımlılığı «partial» ise 2,3 gene fark etmez. . Ancak yer kaybı noktasında yabancı anahtar daha küçük olan tabloya eklenir.
- R: 1-N ise aşağıdaki dönüşümler olabilir.
  1. Sadece R (x1, y1, r) : Yeni bir tablo eklediğimiz için «performans ve veri tekrarı» yönünden iyi değil.
  2. X (x1, x2) Y (y1, y2 , ~~X~~, r) : Performans ve veri tekrarı yönünden iyi.
- R: N-N ise aşağıdaki dönüşüm olabilir.
  1. Sadece R (x1, y1, r)

ER-to-Relational Mapping Algorithm-*step 7 example*



SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

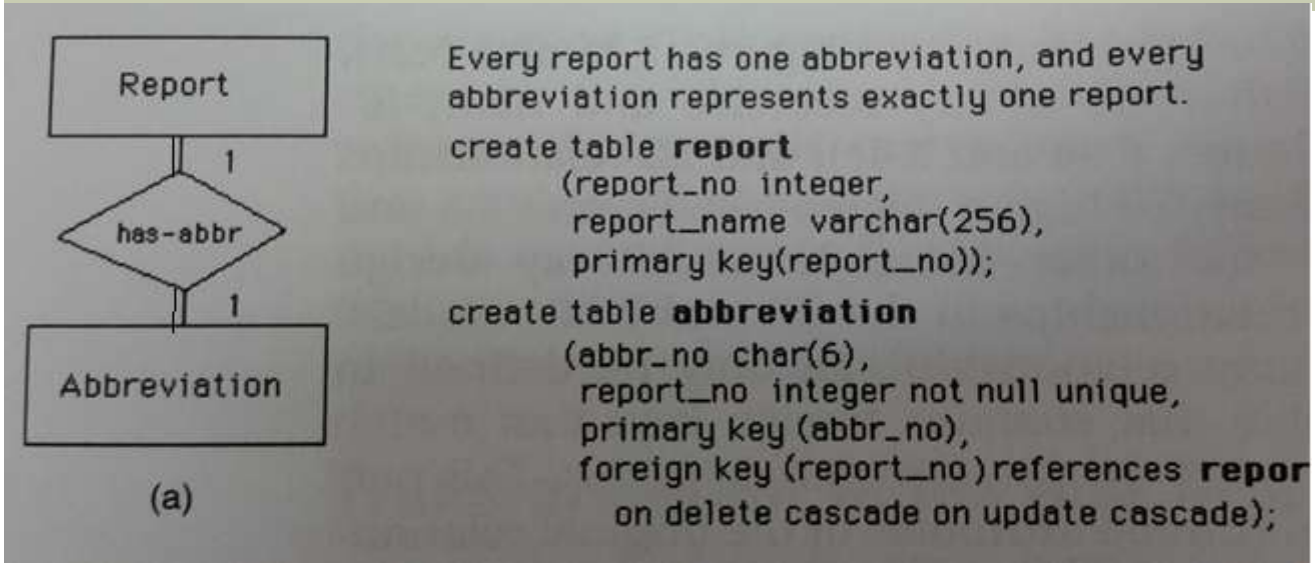
Note that PK depends on the cardinalities of SUPPLY

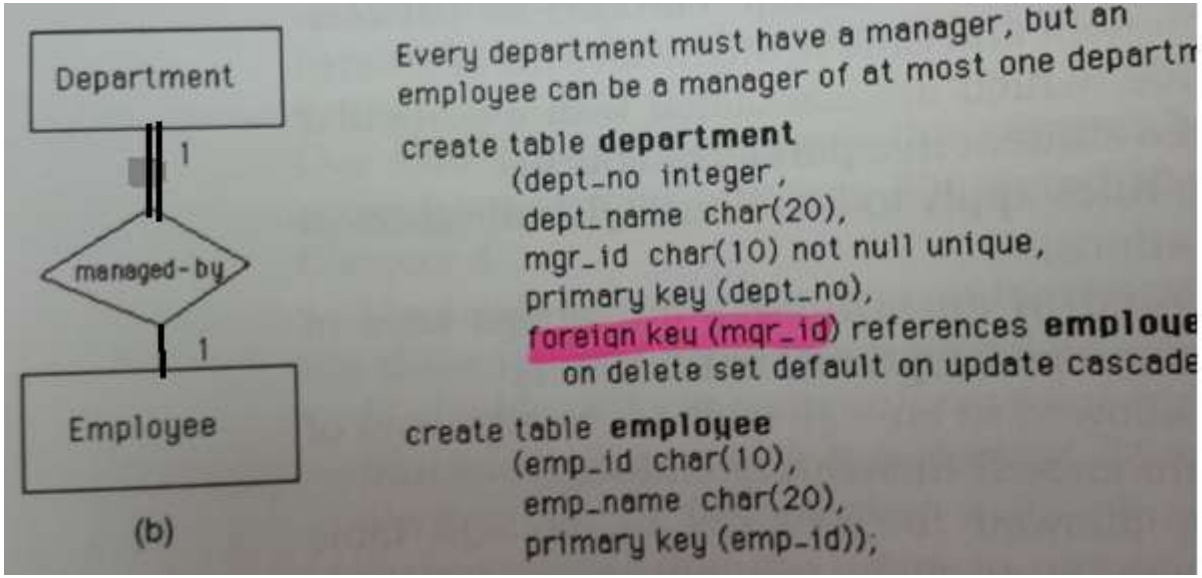
Summary of Mapping constructs and constraints

Table 7.1 Correspondence between ER and Relational Models

ER Model	Relational Model
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foreign key (or "relationship" relation)
M:N relationship type	"Relationship" relation and two foreign keys
n-ary relationship type	"Relationship" relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Örnek

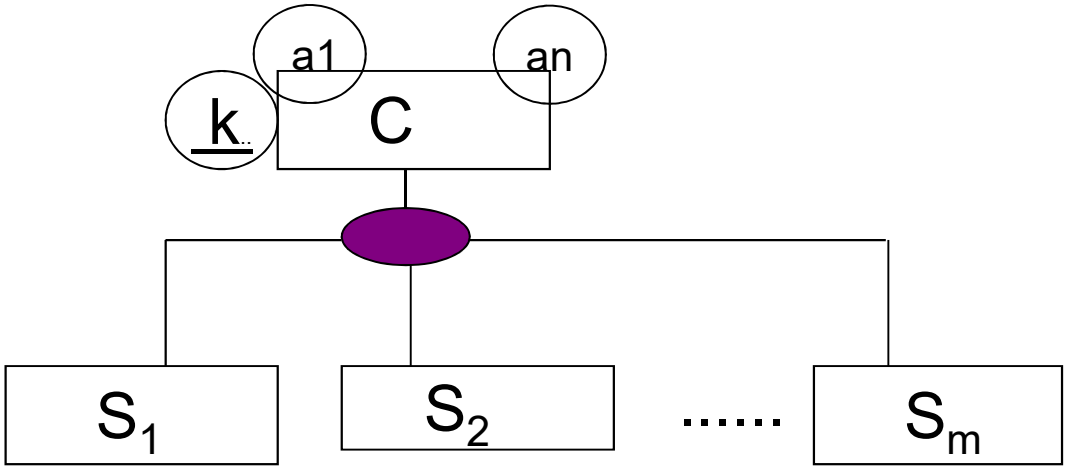




Report ile Abbre. Tablolarının satır sayıları aynı olmak zorunda.Yabancı anhtarın ekleneceği tablo fark etmez. Ancak Department, employee'den daha az sayıda satır içermesi mümkün!.O yüzden yabancı anahtarın Department'a eklenmesi daha avantajlı!

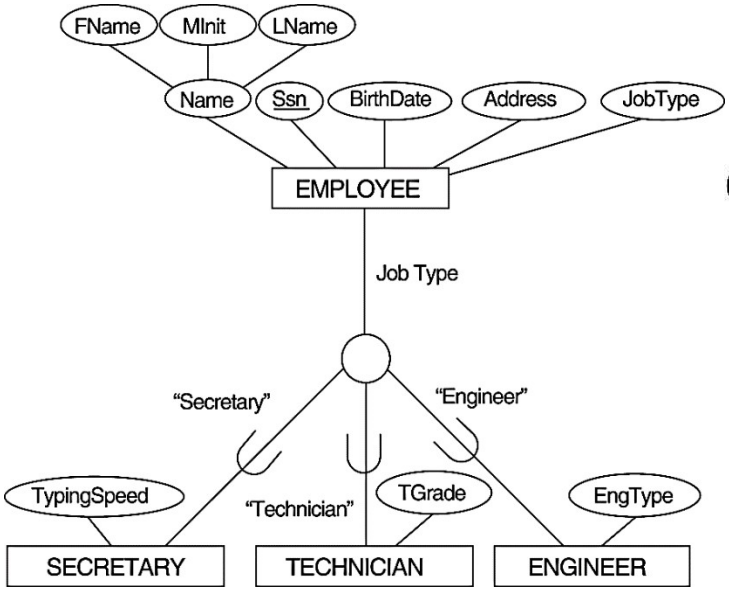
Mapping EER Model Constructs to Relations

- **Step8: Options for Mapping Specialization or Generalization.**
  - Convert each specialization with m subclasses {S1, S2,….,Sm} and generalized superclass C, where the attributes of C are {k,a1,…an} and k is the (primary) key, into relational schemas using one of the four following options:
    - Option 8A: Multiple relations-Superclass and subclasses
    - Option 8B: Multiple relations-Subclass relations only
    - Option 8C: Single relation with one type attribute
    - Option 8D: Single relation with multiple type attributes



Mapping EER Model Constructs to Relations

- **Option 8A: Multiple relations-Superclass and subclasses**
  - **set up a single relation L** for C with attributes Attrs(L) = {k,a1,…an} and PK(L) = k. **set up relation Li** for each subclass Si, 1 < i < m, with the attributesAttrs(Li) = {k} U {attributes of Si} and PK(Li)=k.
  - This option works for any specialization (total or partial, disjoint of over-lapping).



(a)

EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

SSN	TypingSpeed

TECHNICIAN

SSN	TGrade

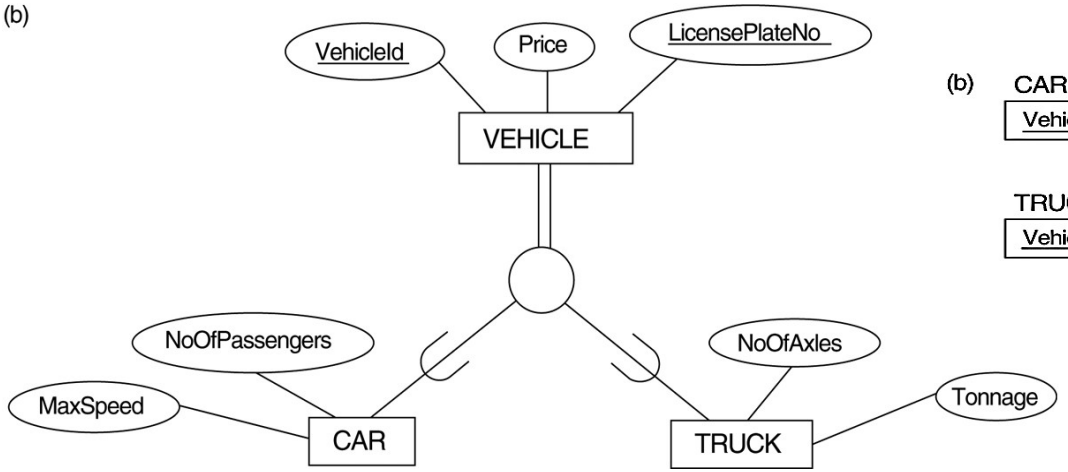
ENGINEER

SSN	EngType

Mapping EER Model Constructs to Relations

- **Option 8B: Multiple relations-Subclass relations only**
  - **set up a single relation Li** for each subclass Si, 1 < i < m, with the attributes Attr(Li) = {attributes of Si} U {k,a1…,an} and PK(Li) = k.
  - This option only works for a specialization whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses).

(b)



(b) CAR

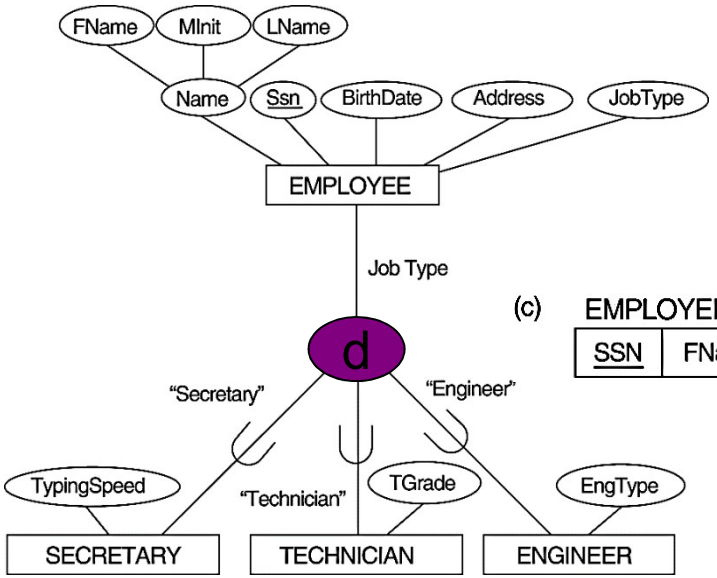
VehicleId	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

TRUCK

VehicleId	LicensePlateNo	Price	NoOfAxles	

Mapping EER Model Constructs to Relations

- **Option 8C: Single relation with one type attribute**
  - **set up a single relation L** with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L) = k. The attribute t is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs.
  - Used for disjoint type of specializations...



Dom(t) ={1,2,....m}

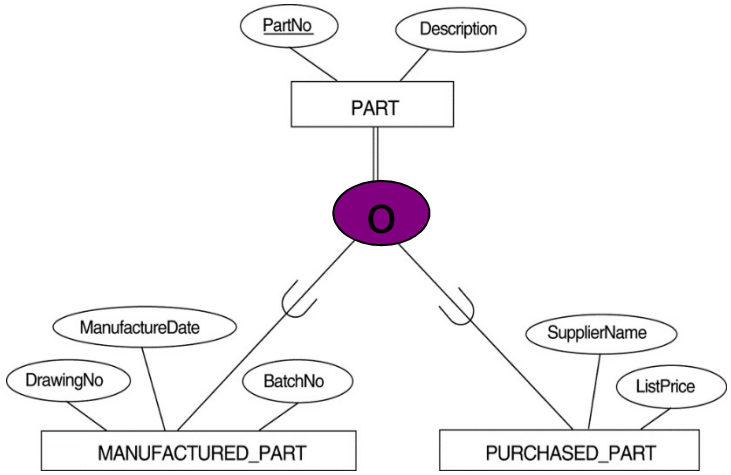
İf specialization is partial t can be null.

(c) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	EngType

Mapping EER Model Constructs to Relations

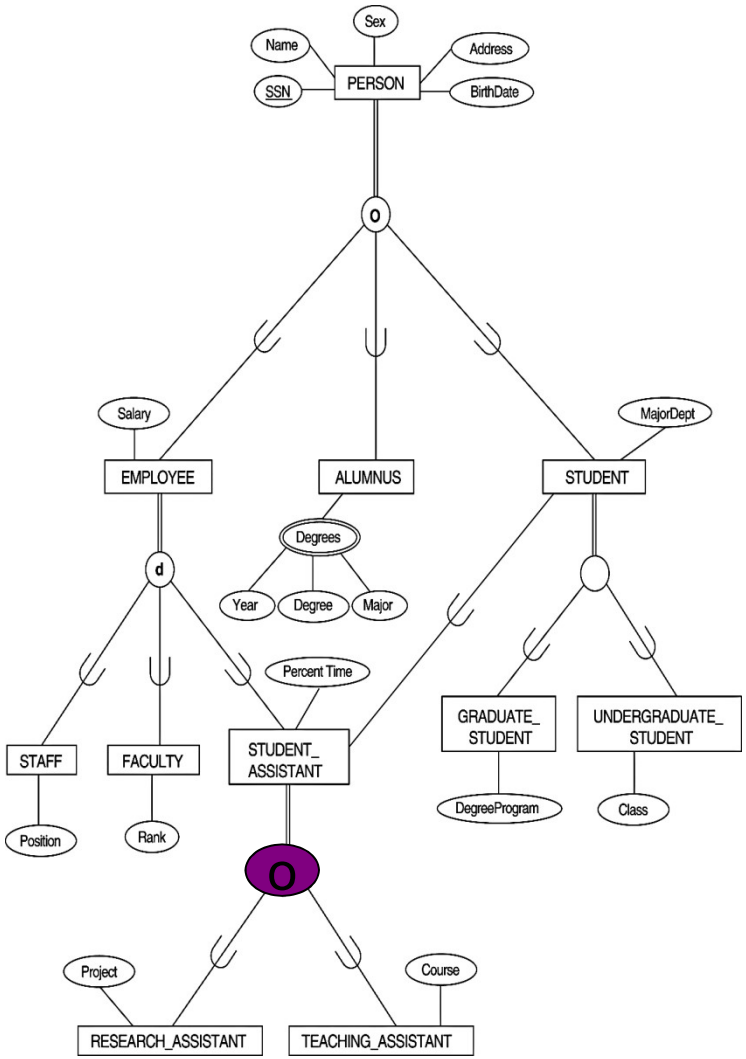
- **Option 8D: Single relation with multiple type attributes**
  - **set up a single relation schema** L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t_1, t_2,…,t_m} and PK(L) = k. **Each t_i, 1 < i < m, is a Boolean type attribute indicating whether a tuple belongs to the subclass S_i.**
  - Used for specializations having overlapping subclasses



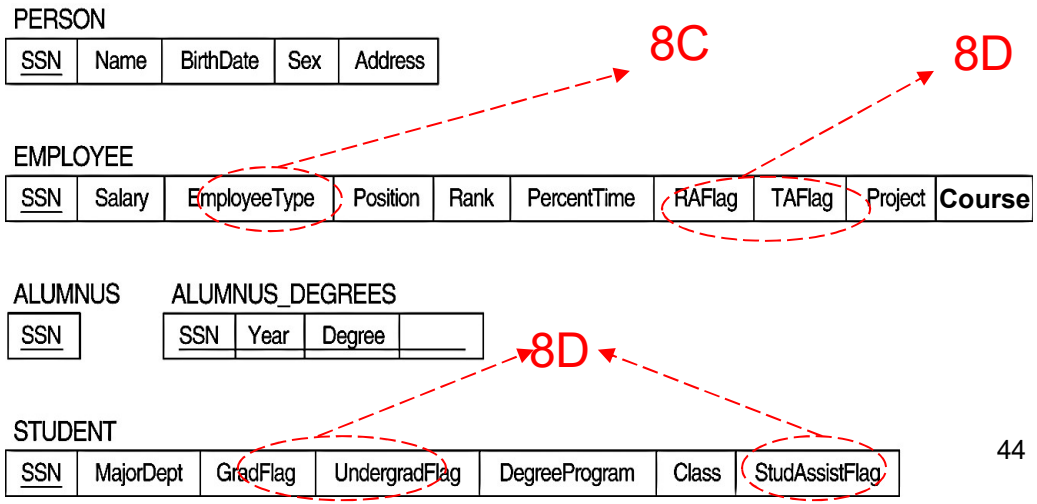
(d) PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice

Mapping EER Model Constructs to Relations



- **Step9** : Mapping of Shared Subclasses (Multiple Inheritance)
  - A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance.
  - We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm. Below both 8C and 8D are used for the shared class STUDENT_ASSISTANT.



Example-1

- Dünya üzerindeki gemi limanları ve gemi hareketlerini tutan bir veri tabanı tasarımı.
- Limanların hangi ülkede hangi okyanusta olduğu saklanıyor. Limanların sadece isimleri saklanıyor. Farklı ülkelerde aynı isimde limanlar olabilir.
- **Bir geminin bir limanı ziyaret ile ilgili sadece giriş ve çıkış tarihleri saklanmalı. Gemi aynı limana çok defalar uğrayabilir**. Her geminin bağlı olduğu bir liman vardır.
- Gemilerin tipleri *(denizaltı, yolcu, tekne, savaş gemisi gibi.)* var, hangi tiplerin varolduğu listelenebilmeli..
- Gemi hareketleri, geminin tarihçesi ile tutulmalı. Bu tarihçe bilgisi geminin hangi gün va zamanda hangi enlem boylamda bulunduğu bilgilerinden örnekler saklamaktadır. Gemi tabiki aynı noktada farklı zamanlarda bulunabilir.

Example-1: ER

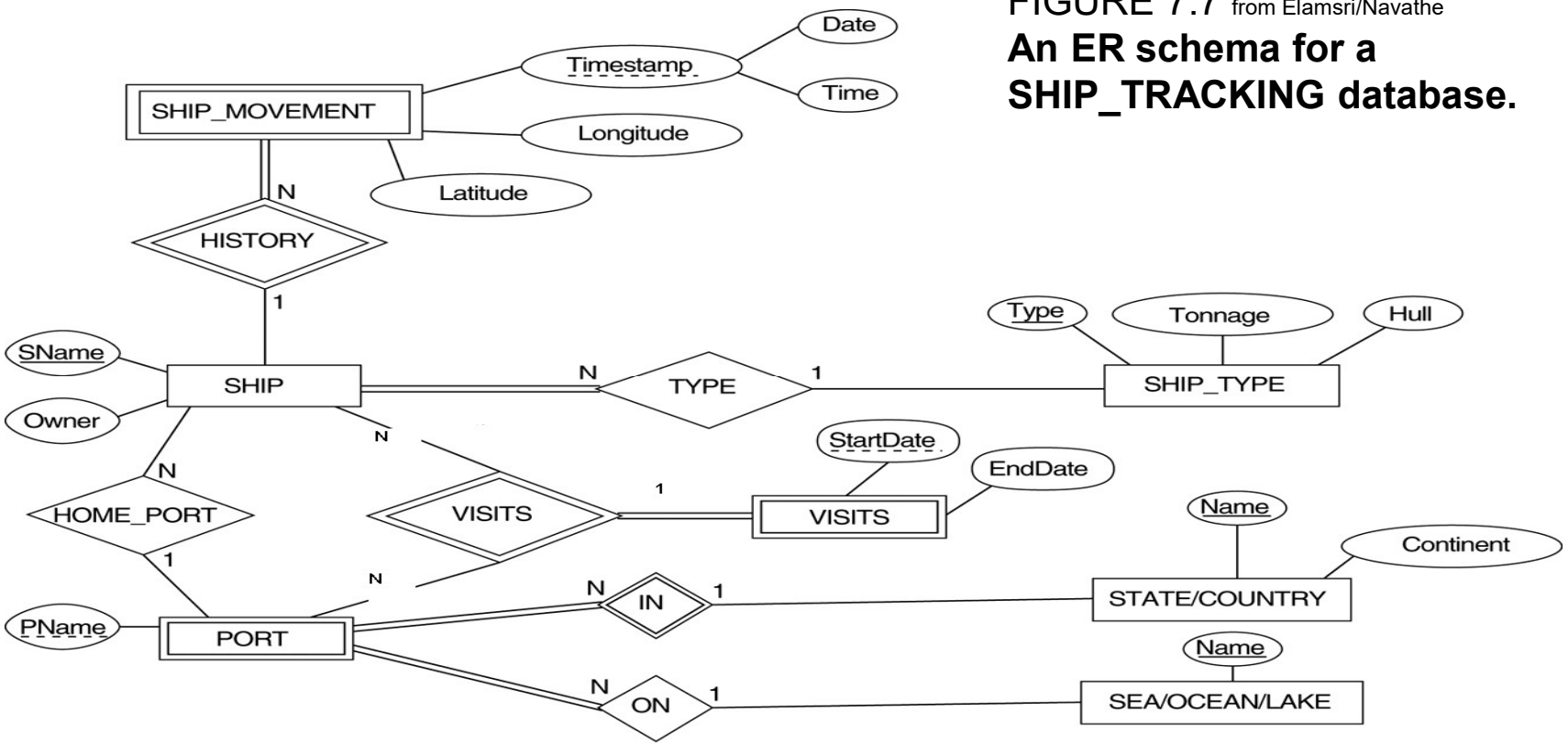


FIGURE 7.7 from Elamsri/Navathe
An ER schema for a SHIP_TRACKING database.

Example-1: Relations

SHIP [Sname, Owner, scName, Pname, Type]

PORT[scName, Pname, solName]

STATE/COUNTRY [Name, Continent]

SEA/OCEAN/LAKE [Name]

SHIP_MOVEMENT [Sname, Date, Time, Longitude, Latitude]

SHIP_TYPE [Type, Tonnage, Hull]

VISITS [scName, Pname, Sname, startDate, EndDate]

Example-2

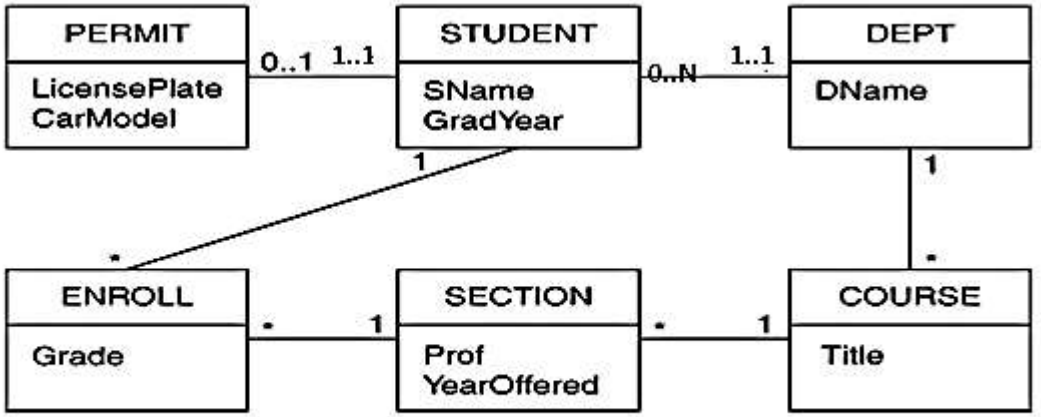
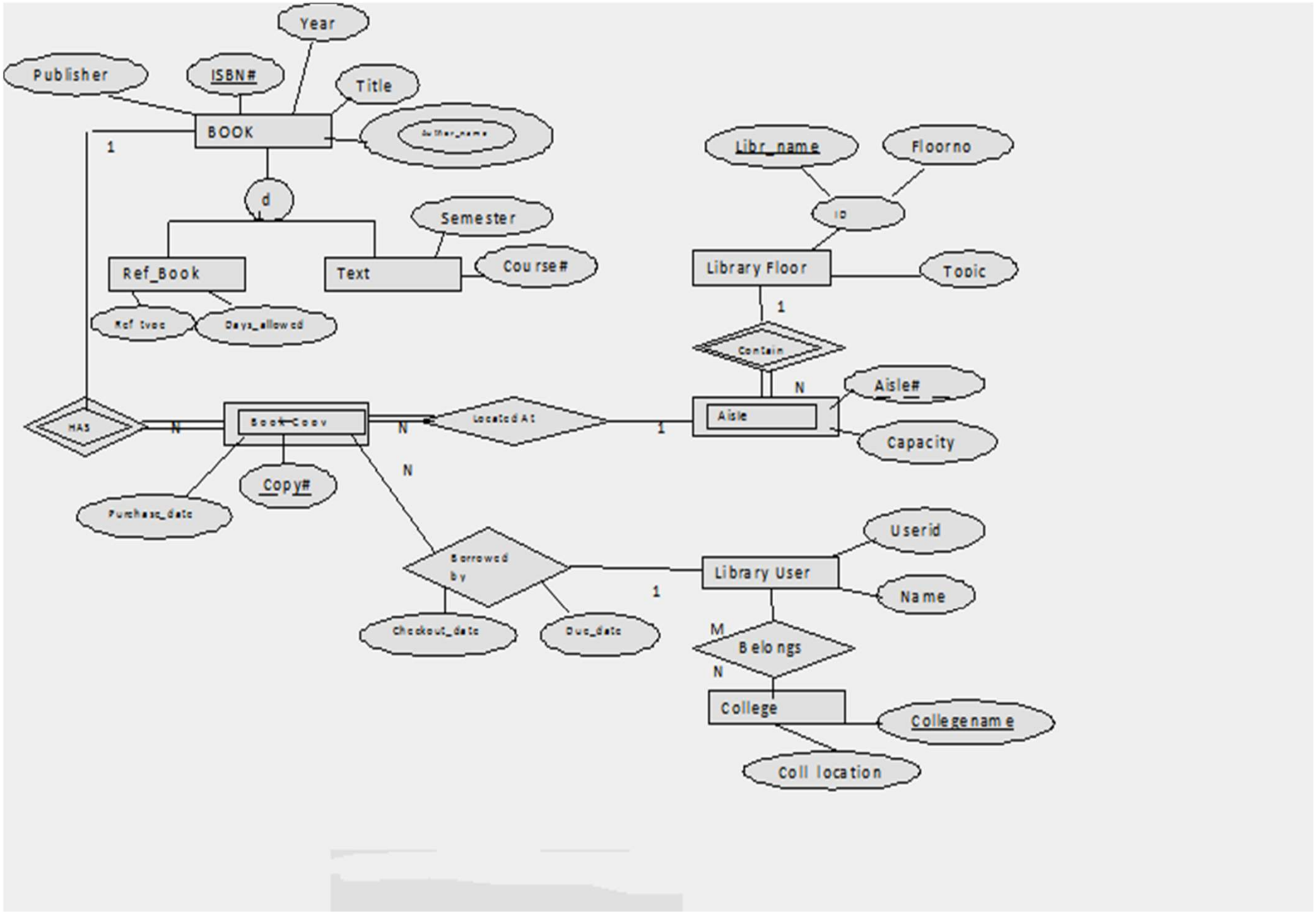


Figure 3-1
A class diagram for the university database

PERMIT(PermitId, LicensePlate, CarModel, *StudentId*)
STUDENT(SId, SName, GradYear, *MajorId*)
DEPT(DId, DName)
ENROLL(EId, Grade, *StudentId*, *SectionId*)
SECTION(SectId, YearOffered, Prof, *CourseId*)
COURSE(CId, Title, *DeptId*)

PERMIT(*StudentId*, LicensePlate, CarModel)
STUDENT(SId, SName, GradYear, *MajorName*)
DEPT(DName)
ENROLL(*StudentId*, *SectionId*, Grade)
SECTION(SectId, YearOffered, Prof, *Title*)
COURSE(Title, *DeptName*)

Example-3


Year
Publisher
ISBN#
Title
BOOK
1
d
Semester
Ref_Book
Text
Course#
Ref_type
Days_allowed
HAS
N
Book_Copy
Located At
N
1
Copy#
Purchase_date
N
Borrowed by
1
Checkout_date
Due_date
Libr_name
Floorno
ID
Library Floor
Topic
1
Contain
N
Aisle#
Aisle
Capacity
Userid
Library User
Name
M
Belongs
N
College
Collegename
Coll location

Example 3

- Book (ISBN#, Publisher, Title, Year)
- Book_Author (ISBN#, Authorname)
- Ref_Book (ISBN#, Ref_type, Days_allowed)
- Text (ISBN#, Semester, Course) ----> PK dikkat!!
- Book_Copy(ISBN#, Copy#, Purch_date, User_id, Check-out_date, Due_date, Libr_name, Floor#, Aisle#)
- Library_floor(Libr_name, Floor#, Topic)
- Library_aisle (Libr_name, Floor#, Aisle#, capacity)
- Library_user (User_id, Name)
- College (Coll_name, Coll_location)
- User_belongs (Userid, College_name)