4.Relational Model :

Basic Concepts

Relational Model

- Most widely used database model : Relational Model

- Examples of Relational DBMS
  - MySQL, DB2, Oracle, Sybase, SQL server, . . .

- This model is based on the concept of a **relation**.

- A relation is a mathematical concept based on a **set**.

- Advantages of Relational Model
  - Simple (user friendly) data structure
  - Provide Data Abstraction
  - Provide Data Independence
  - Provide High-level programming

Relation : Definition

- Formally, a given n sets D_1, D_2, ... D_n, a **relation r** is a subset of D_1 x D_2 x . . . x D_n. (x : Cartesian Product)

- Here, $<a_1, a_2, ..., a_n>$ where each $a_i \in D_i$ is called a **tuple**.

- In other words, a relation is a **finite set of tuples**.

- Example : Let $D_1 = \{0, 1\}$, $D_2 = \{a, b, c\}$; Then,

  D_1 x D_2 = {<0, a>,<0, b>, <0, c>, <1, a>, <1, b>, <1, c>}

  - r_1 = {<0, a>, <0, b>, <1, c>} is one possible relation.
  - r_2 = {<1, a>, <1, b>} is another relation.

Relation : Definition

- Another Example : Student

  stud_name = {abe, cal, bob, ann}
  dept_name = {physics, math, computer}
  gender = {male, female}

- **r** = {(abe, computer, male), (cal, physics, male),
  (ann, math, female), (bob, math, male)}

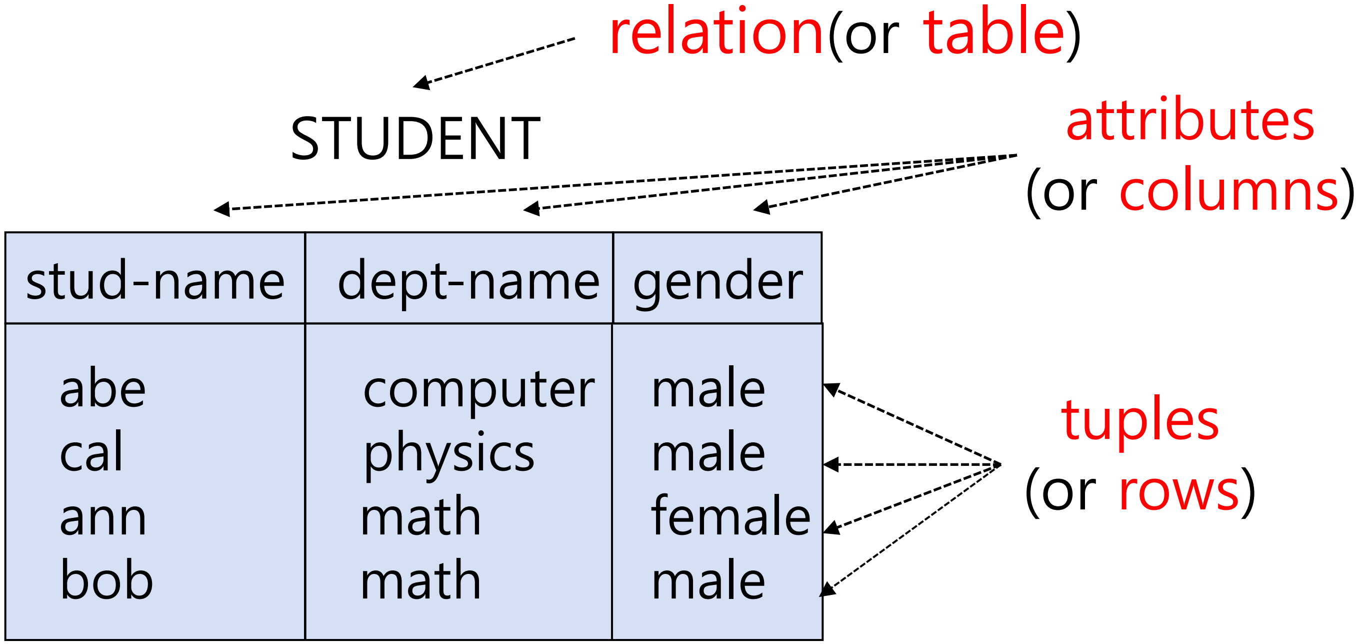
  is a relation over stud_name x dept_name x gender

- (abe, computer, male) is an example of tuple. Thus, this relation **r** consists of 4 tuples;

- Show another examples of relations;

Relation : Table 표현

- We can represent relation as "**table**"; A table consists of **rows** and **columns**.
- Each **row** corresponds to a **tuple**; It represents "entity" or "relationship".
- Each **column** corresponds to an **attribute**; It represents structure of table.



relation(or table)

STUDENT

attributes (or columns)

tuples (or rows)

stud-name	dept-name	gender
abe	computer	male
cal	physics	male
ann	math	female
bob	math	male

Properties of Relation

- The number of tuples in a relation is **finite**.
- The order of tuples in a relation is **not** important.
- Any duplicated tuples in a relation are **not** allowed.
- Each attribute in a relation must have a **distinct** name.
- Values of Attributes:

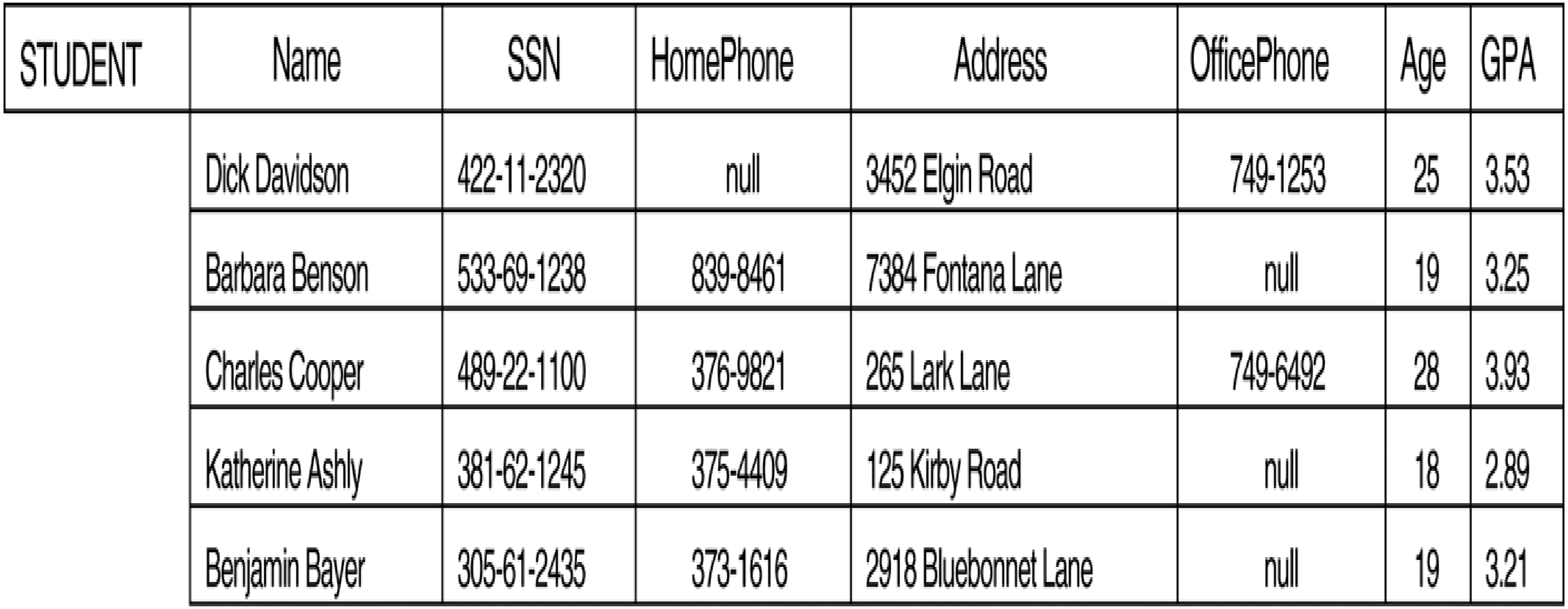
  (1) Values of each attribute must be **atomic**(indivisible).

  - Intersection of row and column has **single** value.
  - Multi-valued(or, composite) attributes are not allowed.

  (2) Special value "**NULL**" is allowed.

  - NULL means "unknown", "unavailable", or "undefined".

Relation : Another Example



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21

- Order of tuples does not matter.
- All tuple are distinct; No duplicates are allowed.
- Each attribute has atomic(= only single) value.
- Some attributes have NULL values.

Properties of Relation

- Values in a Tuple:
  - Each attribute value in a tuple must be within its data type.
  - For example, each attribute value in student tuple must be within;

    Name: CHAR(20); SSN: CHAR(9); . . . Age: INT, . .

- Some Useful Notation:
  - We refer to component values of a tuple t by
    - ✓ t[A_i] or t.A_i
    - ✓ This is the value Vi of attribute Ai for tuple t
    - ✓ For example, t[Name], t[Age], . . .
  - Similarly, t[A_i, A_j, . . , A_n] refers to the subtuple of t containing the values of attributes A_i, A_j, . . , A_n, respectively in tuple t

Key

- **Super key**
  - A set of attributes **K** of a relation R such that no two tuples in R must have the same value for **K**.
  - Values of **K** can identify all tuples in R uniquely.
- **Key**
  - A "minimal" superkey **K** that does not does not contain a subset of attributes that is itself a super key.
  - Removal of any attribute from **K** results in a set of attributes that is no more a super key (thus, can not identify tuples)
- Every key is super key, but reverse is not true.

Key

A	B	C	D
10	10	20	20
15	20	10	18
20	15	18	15
10	18	15	18

(1) 위 relation에서 super key 들을 모두 찾아라.

{B}, {C}, {A, D}, {A, B}, {A, C}, {B, C}, {B, D}, {C, D},

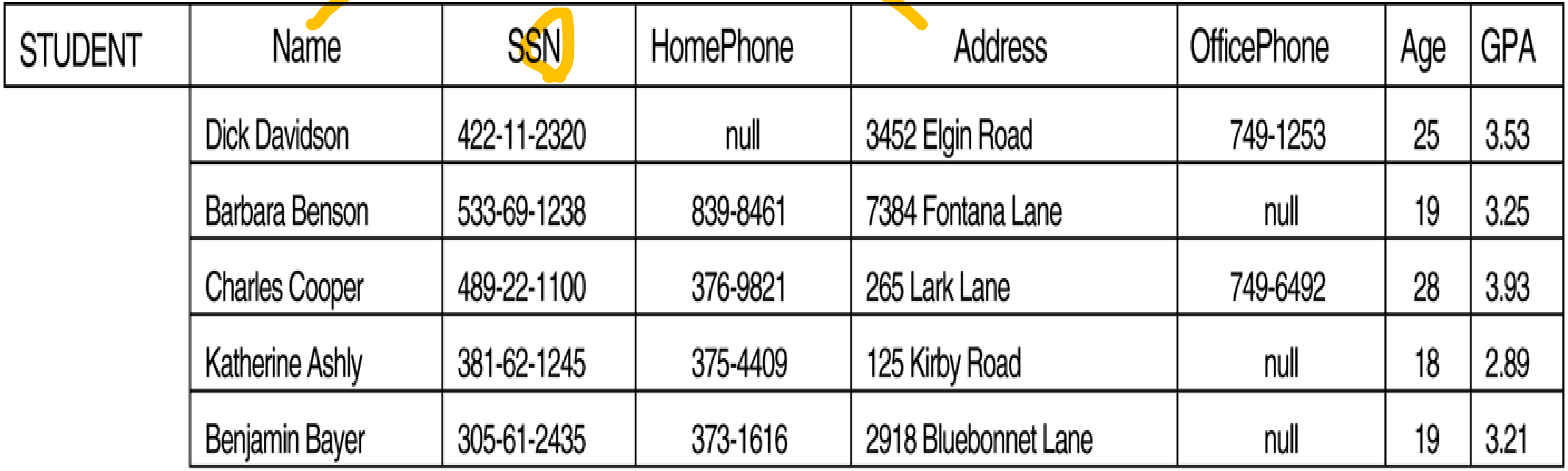
{A, B, C}, {B, C, D}, {A, C, D}, {A, D, B}, {A, D, B, C}

(2) 위 relation에서 key 들을 모두 찾아라.

{B}, {C}, {A, D}

참조: Super key는 유일성(unique)만 만족하고, Key는 유일성과 최소성(minimal) 모두 만족함. 따라서, tuple들을 식별하기 위해 항상 key를 사용하고, 각 relation에는 key가 존재해야 함.

Key



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21

● What is Key(s)? What is super key(s)?

- Every student's SSN value must be distinct;
- There may have the same student's names, but with distinct addresses.
- There may have the same addresses, but with distinct student's names.
- But there must not have same student's name with same addresses. (or same addresses with the same names)

Key

- **Candidate Key**
  - There may have more than one key in a relation; In this case, each of the keys is called a **candidate** key;
- Types of Keys
  - Simple Key : Consists of single attribute;
  - Composite Key : Consists of 2 or more attributes;
- **Primary Key**
  - If a relation has several (candidate) keys, we must choose one for identification use in practice; The chosen key is called a **Primary Key** (**PK**). (**PK**는 밑줄(underline)로 표시함)
  - Every relation must have its own primary key.

Key : Example

- Consider the following CAR Relation;

  CAR(License-number, Engine-serial-number, Make, Model, Year)

  - Key 1 = {License-number}
  - Key 2 = {Engine-serial-number}

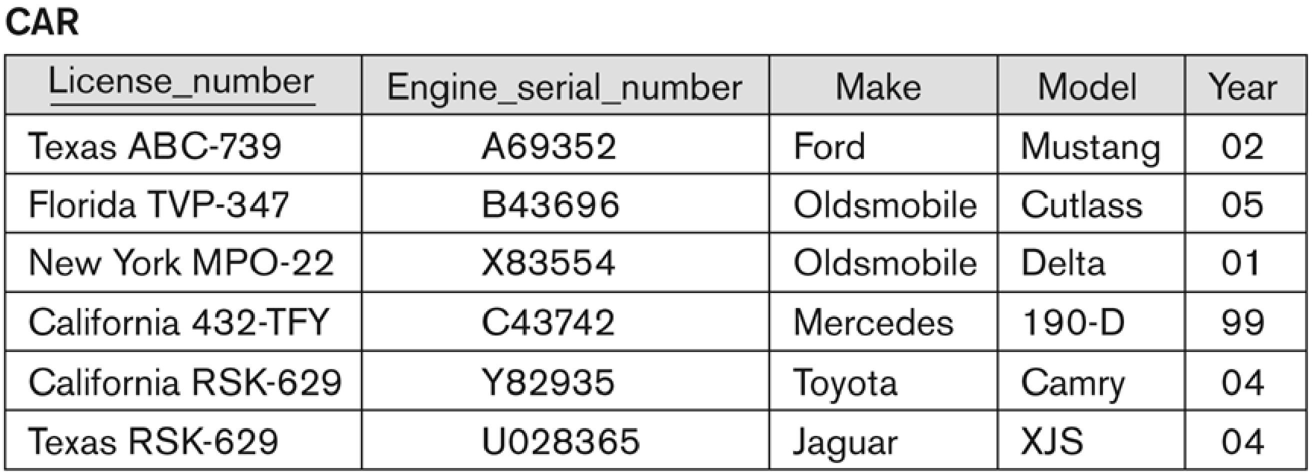
- Both are also super keys of CAR. {Engine-serial-number, Make} is a super key, but not a key. {License-number, Model} is also a super key, but not a key.

- Any set of attributes that includes a key is a super key.

  A minimal super key is a key.

- There are two (candidate) keys. Primary key is used in practice to identify each tuple in a relation. Here, we select {License-number} as primary key in CAR relation, which is underlined.

CAR Relation



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.

- keys = {License-number}, {Engine-serial-number}
- Super keys = Any set of attributes including {License-number}
  or {Engine-serial-number}
- Primary key = {License-number}

Good Primary Keys

- Stable : Do not change over the life of the database
- Definitive : Values always exist
- Numeric : ID ‘12345’ is better than name ‘michael Jordan’
- Minimal : Fewest attributes as possible (3 or fewer)
- Short : Are not too long length (bytes)
- Security : No sensitive information hidden

Key : Exercise

- Consider the following "Company" relations:

  EMPLOYEE (eno, ename, age, addr, super_eno, work_dno)

  DEPARTMENT (dno, dname, phone, mgr_eno)

  PROJECT (pno, pname, control_dno)

  WORK-ON (eno, pno, hours)

- Under your assumptions, answer following questions:

  (1) Find (all) super keys.

  (2) Find (all) keys.

  (3) Specify primary keys.

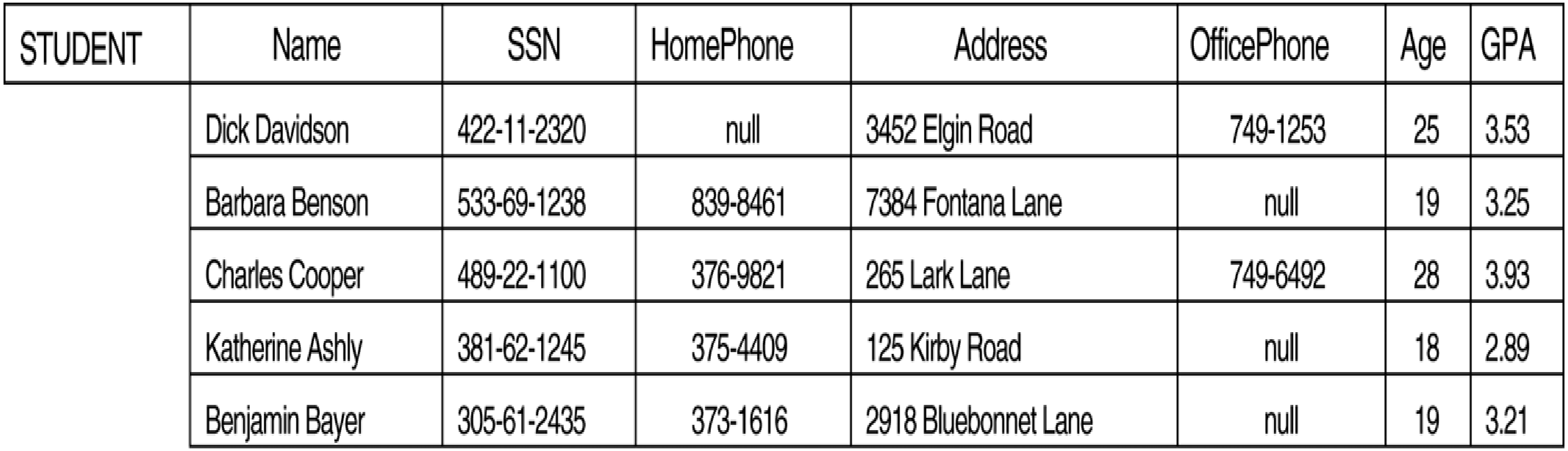
(Relational) Integrity Constraints

- Integrity Constraints are conditions that must be satisfied by all relations; There are three main types of constrains;
  - **Key** Integrity
  - **Entity** Integrity
  - **Referential** Integrity

- **Key Integrity**
  - Given any key **K**, for any two tuples t1 and t2 in a relation R, t1[**K**] ≠ t2[**K**].

- **Entity Integrity**
  - **Primary key** in a relation R must not contain **null** values in any tuple in R; That means; **t[PK] ≠ null** for any tuple **t** in R
  - If primary has several attributes, **null** is not allowed in any of these attributes

Violating Key/Entity Integrity



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21

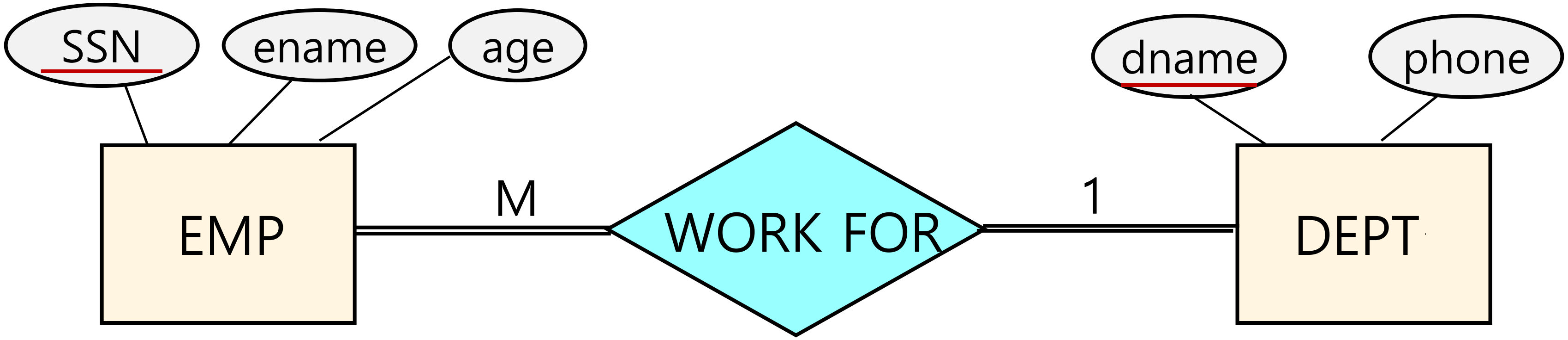
- 다음 각 연산에서 Key Integrity, Entity Integrity의 위반 유무를 판단하라. (단, Primary Key = {SSN})
  - Insert a student with <papa jones, 489-22-1100, 290-7118, . . >
  - Insert a student with only SSN value is unknown.
  - Insert a student with <mama jones, 123-45-6789, null, null, null, . . >
  - Update Charles Cooper’s SSN by 533-69-1238.
  - Delete students with GPA = 3.25.

Referential Integrity

- This specifies a ***relationship*** among tuples in relations.
- When **referencing** relation **R1** wants to relate **referenced** relation **R2**, you must include a **common** attribute(s).
- The common attribute in **referenced** relation **R2** is **Primary Key (PK)**.
- The common attribute in **referencing** relation **R1** is called **Foreign Key (FK)**.
- Tuples in **referencing** relation **R1** have FK that reference the PK of **referenced** relation **R2**.

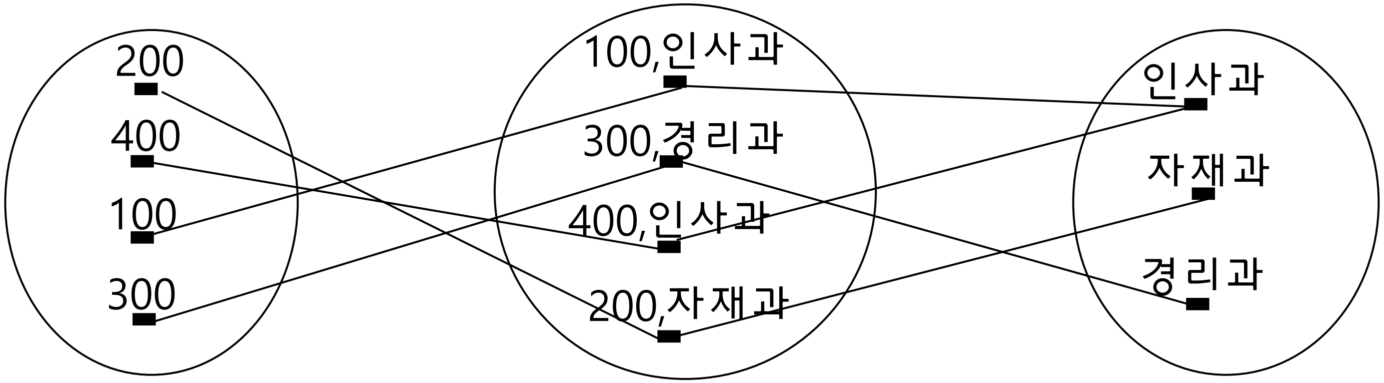
Referential Integrity

- 다음 ER Diagram을 relation 구조로 표현하라.

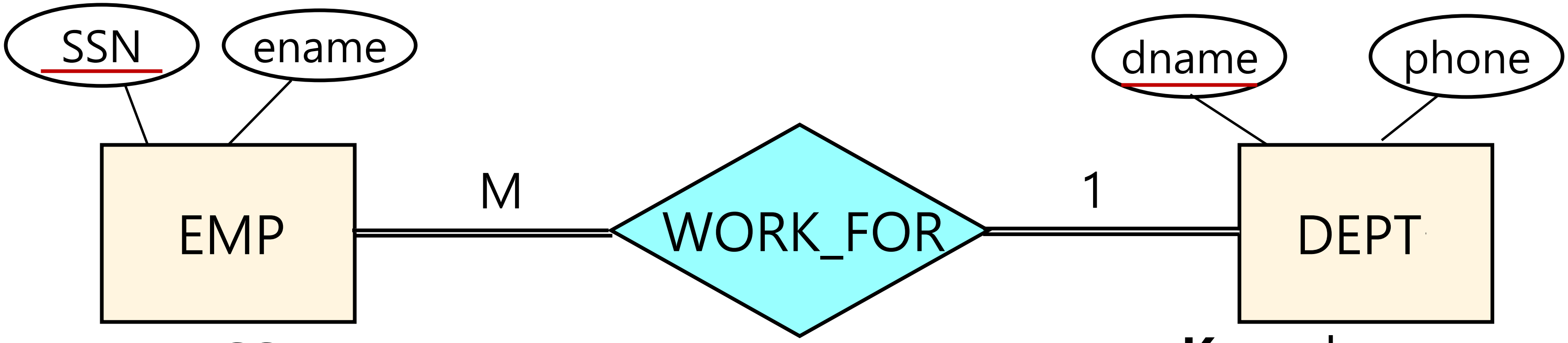


- Entity Type EMP와 DEPT은 (자연스럽게) relation 구조로 mapping 됨.

그렇다면, Relation Type WORK FOR는?



Referential Integrity



Key: SSN

Key: dname

EMP (**Referencing**)

SSN	ename	age	dname
200 .	. .	. .	자재과
400 .	. .	. .	인사과
100 .	. .	. .	인사과
300 .	. .	. .	경리과

WORK FOR

DEPT (**Referenced**)

dname	phone
인사과	. . . .
자재과	. . . .
경리과	. . . .

PK: SSN
FK: dname

PK: dname

Referential Integrity

- The value in **FK** of **referencing** relation **R1** must be either

  (1) an existing value of the corresponding primary key **PK** in the **referenced** relation **R2**,

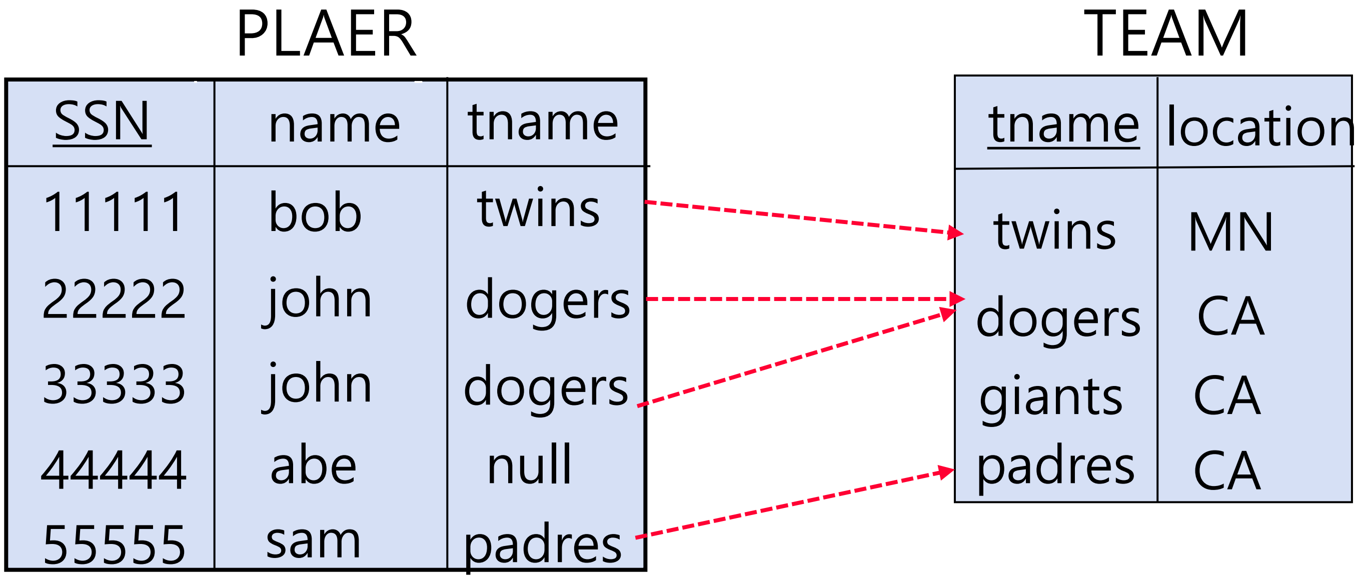
  (In case (1), every **FK** values in referencing relation **R1** must **exist** in **PK** in the referenced relation **R2**)

  or

  (2) a **null** value

  (In case (2), the **FK** in **R1** should not be a part of its own primary key.)

Referential Integrity



PLAER

SSN	name	tname
11111	bob	twins
22222	john	dogers
33333	john	dogers
44444	abe	null
55555	sam	padres

TEAM

tname	location
twins	MN
dogers	CA
giants	CA
padres	CA

- Referencing relation

  - **PK** : SSN

  - **FK** : tname

- Referenced relation

  - **PK** : tname

Referential Integrity

EMPLOYEE

SSN	name	Super-SSN
11111	bob	22222
22222	john	33333
33333	john	44444
44444	abe	null
55555	sam	44444

- Referencing and referenced relation is the same.
- 한 relation의 FK가 자기 자신의 relation의 PK를 참조함.
  - What is **PK**?
  - What is **FK**?

Referential Integrity

COURSE

CID	name
CS200	OS
CS250	DB
CS300	PL

ENROLL

CID	SID	credit
CS200	12345	3
CS200	23456	3
CS300	23456	4
CS250	23456	3
CS300	45678	3

STUDENT

SID	name	age
12345	Bob	22
23456	Ann	18
34567	Jim	30
45678	Eve	27

- Referenced:

  - **PK**: CID

- Referencing:

  - two **FK**s: CID and SID

  - **PK**: {CID, SID}

- Referenced:

  - **PK** : SID

- Note: Any NULL value is not allowed in either {CID} or {SID} in ENROLL relation. Why??

Operations Causing Integrity Violation

- Key Integrity, Entity Integrity, and Referential Integrity can be violated by the following operations.
  - INSERT
  - DELETE
  - UPDATE

- If integrity is violated, several optional actions can be taken:
  - Perform the operation but ask to user to correct it.
  - Cancel the operation that causes the violation.

    (RESTRICT option)
  - Trigger additional updates so the violation is corrected.

    (CASCADE, SET NULL option)
  - Execute a user-specified error-correction routine.

Possible Violations for INSERT operation

- **INSERT** may violate any of the constraints:
  - **Key Integrity**:
    - ✓ If the value of a **key** attribute in the new tuple already exists in another tuple in the relation
  - **Entity Integrity**:
    - ✓ If the primary key value is **null** in the new tuple
  - **Referential Integrity**:
    - ✓ If a **foreign key** value in the new tuple references a **primary key** value that does not exist in the referenced relation

Possible violations for DELETE operation

- **DELETE** may violate only referential integrity:
  - If the **primary** key value of the tuple being deleted is referenced from other tuples in the relations.
  - Can be corrected by several actions:
    - RESTRICT option: Reject the delete operation
    - CASCADE option: Propagate the new primary key value into the foreign keys of the 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

Possible violations for UPDATE Operation

- 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

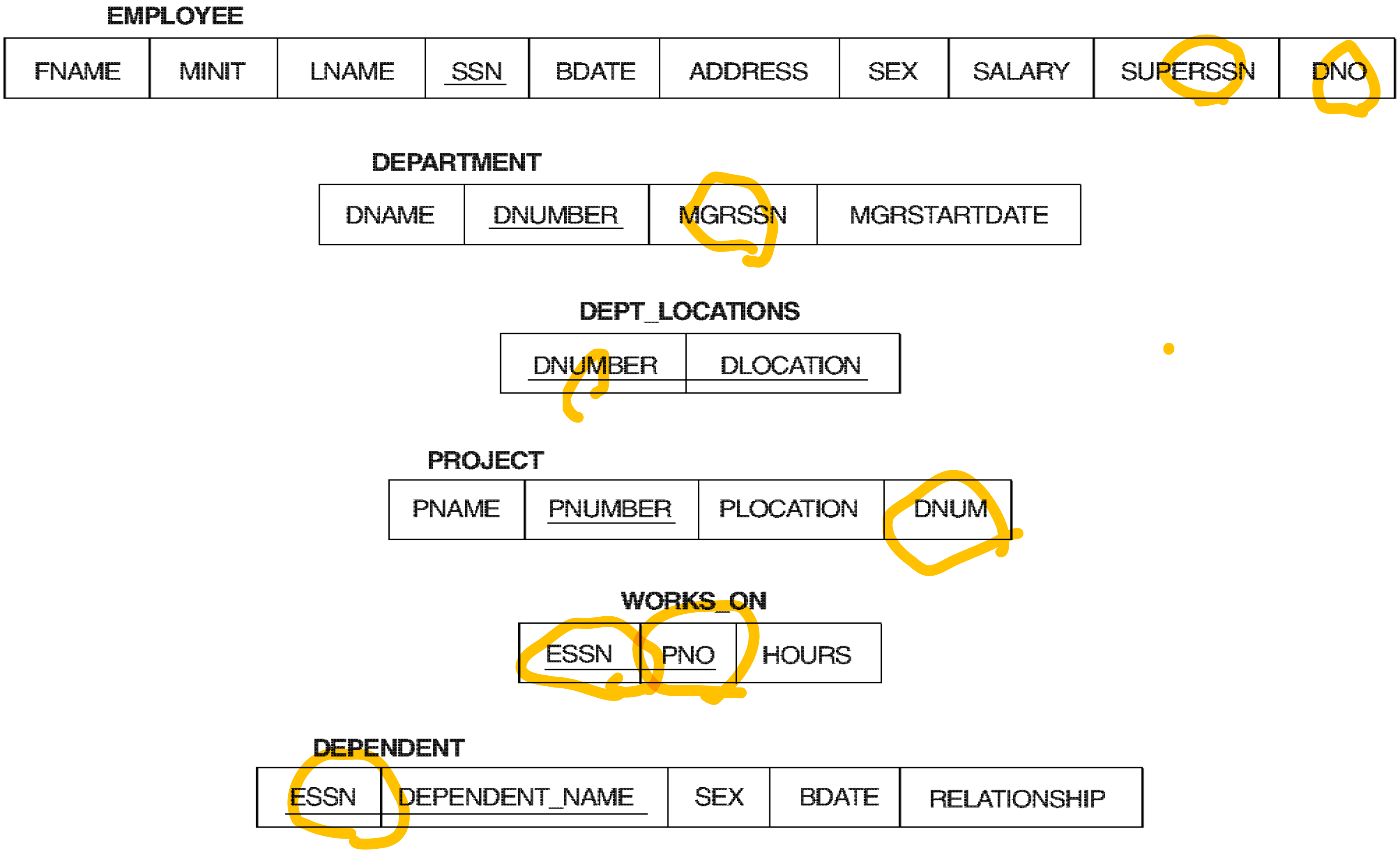
Violating Referential Integrity

- 다음 각 연산에서 referential integrity를 위반 유무를 판단하라.
  - Insert <77777, sam, eagles> into PLAYER; 위반했음; 사용자 실수
  - Delete <55555, sam, padres> from PLAYER; 위반 안 했음
  - Delete <twins, MN> from TEAM; 위반했음
  - Update tname “dogers” by “winners” from TEAM; 위반했음
  - Delete employee(s) with name ‘abe’ from EMPLOYEE; 위반했음
  - Delete <23456, Ann, 18> from STUDENT; 위반했음
  - Update CID CS200 by CS400 from COURSE; 위반했음
  - Delete <34567, Jim, 30> from STUDENT; 위반 안 했음

- 다음 relation들에서 foreign key를 명시하라.

Figure 7.5 Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

Note : FK와 이와 상응하는 PK는 반드시 이름이 같을 필요 없음;



EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

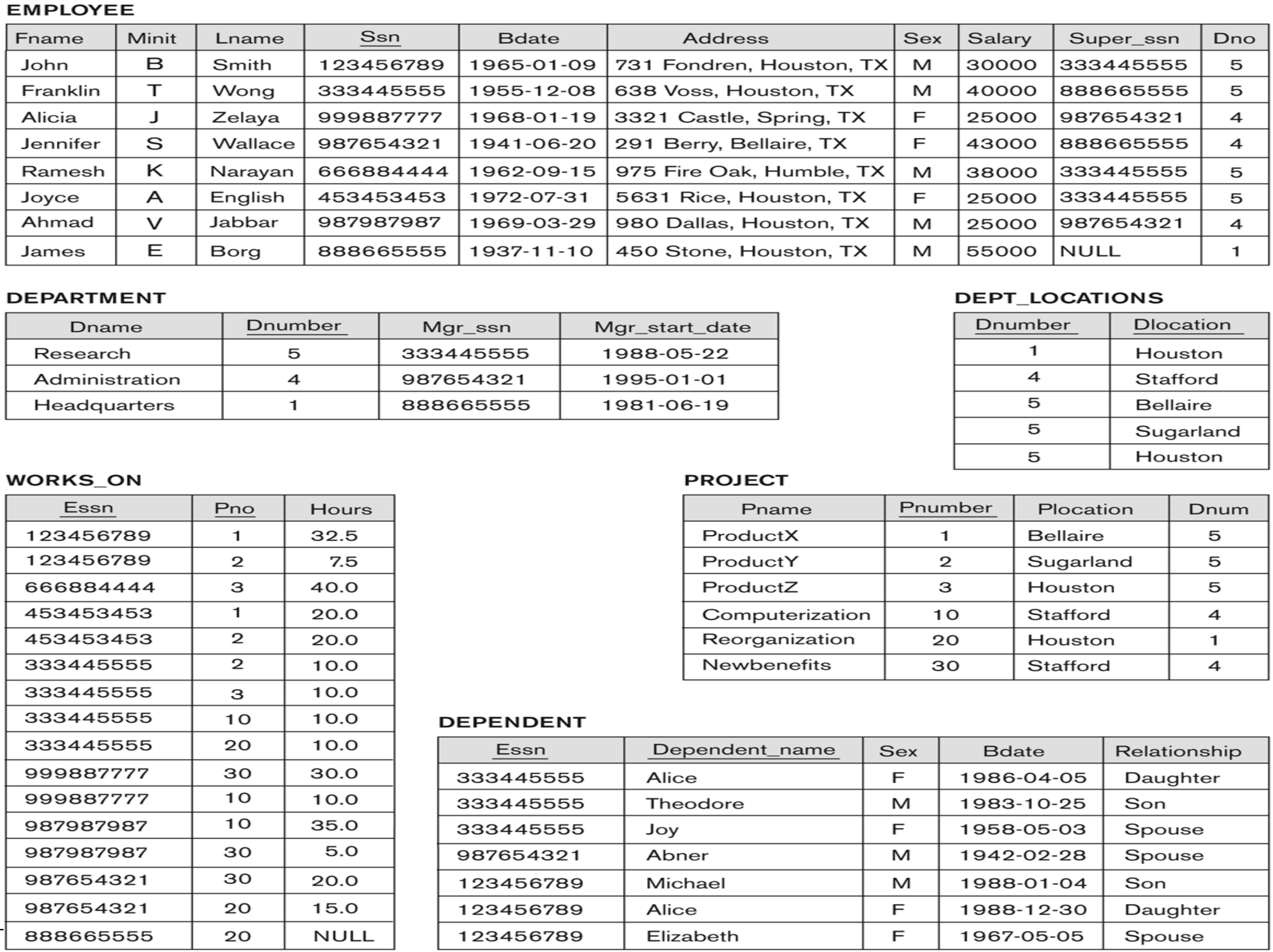
WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

One possible database state for the COMPANY relational database schema.



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	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

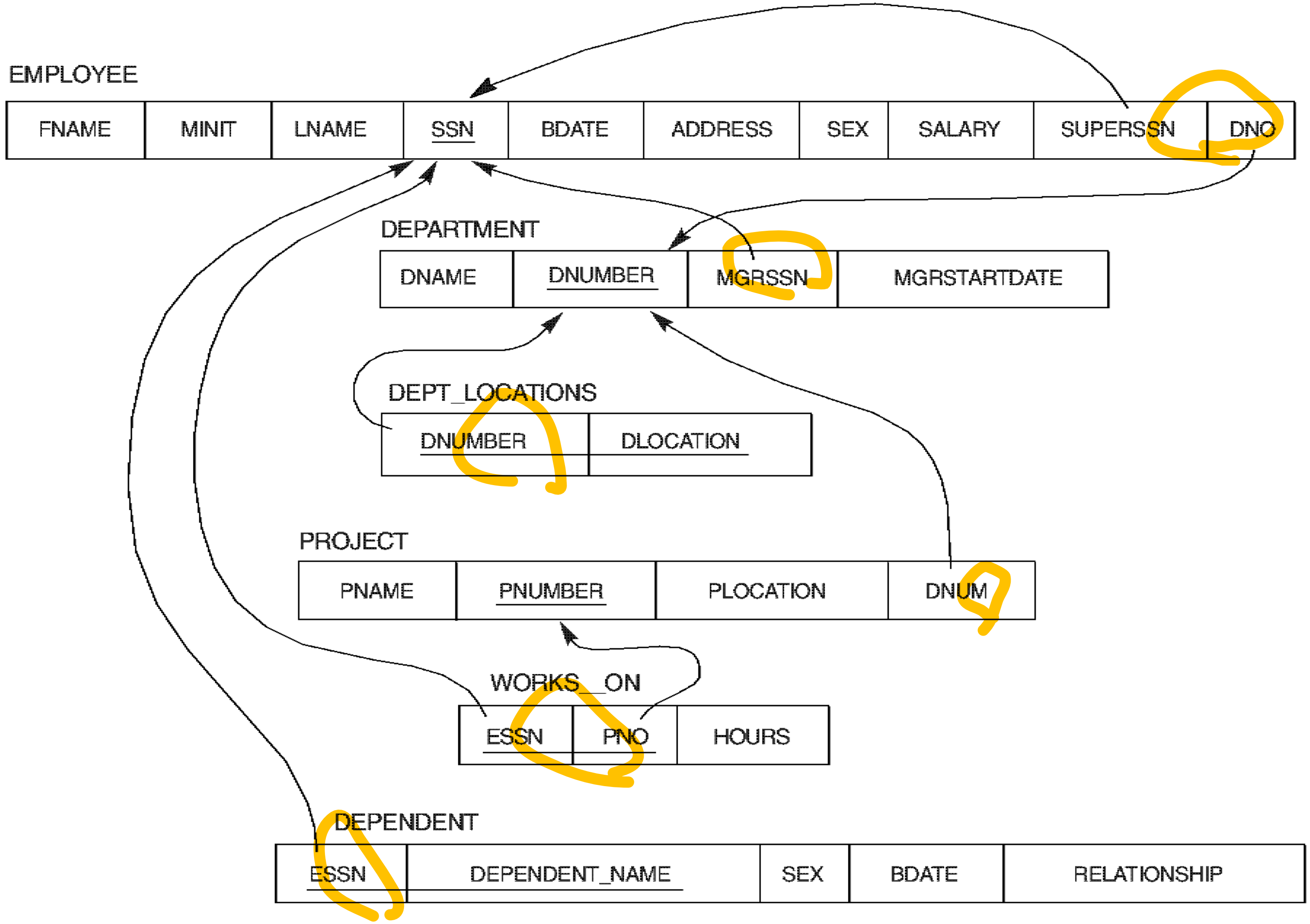
PROJECT

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	Sex	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

Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Referential Integrity : Exercise

- 다음 각 연산에서 referential integrity의 위반 유무를 판단하라. 만약 위반시에 어느 relation들이 영향을 받는가?

  - Delete <123456789, Michael, ... > from DEPENDENT
    : 위반 안 했음. 영향 받는 relation들 없음.
  - Insert new employee <. . . . ., 3> into EMPLOYEE
    : 위반 했음.
  - Delete <Franklin, ... , 33344555, . . > from EMPLOYEE
    : 위반 했음. EMP, DEPT, WORK-ON, DEPENDENT 모두 영향 받음.
  - Update Dnumber 5 by 7 from DEPARTMENT

    : 위반 했음. EMP, DEPT-LOCATION, PROJECT 모두 영향 받음.
  - Delete tuple(s) with Pno = 10 from WORKS-ON

    : 위반 안 했음. 영향 받는 relation들 없음.

Referential Integrity: Exercise

- Draw a relational schema diagram by specifying the FKs.

  Note: Underlined attributes are primary keys.

BRANCH (branch-name, branch-city, assets)

LOAN (loan-number, branch-name, amount)

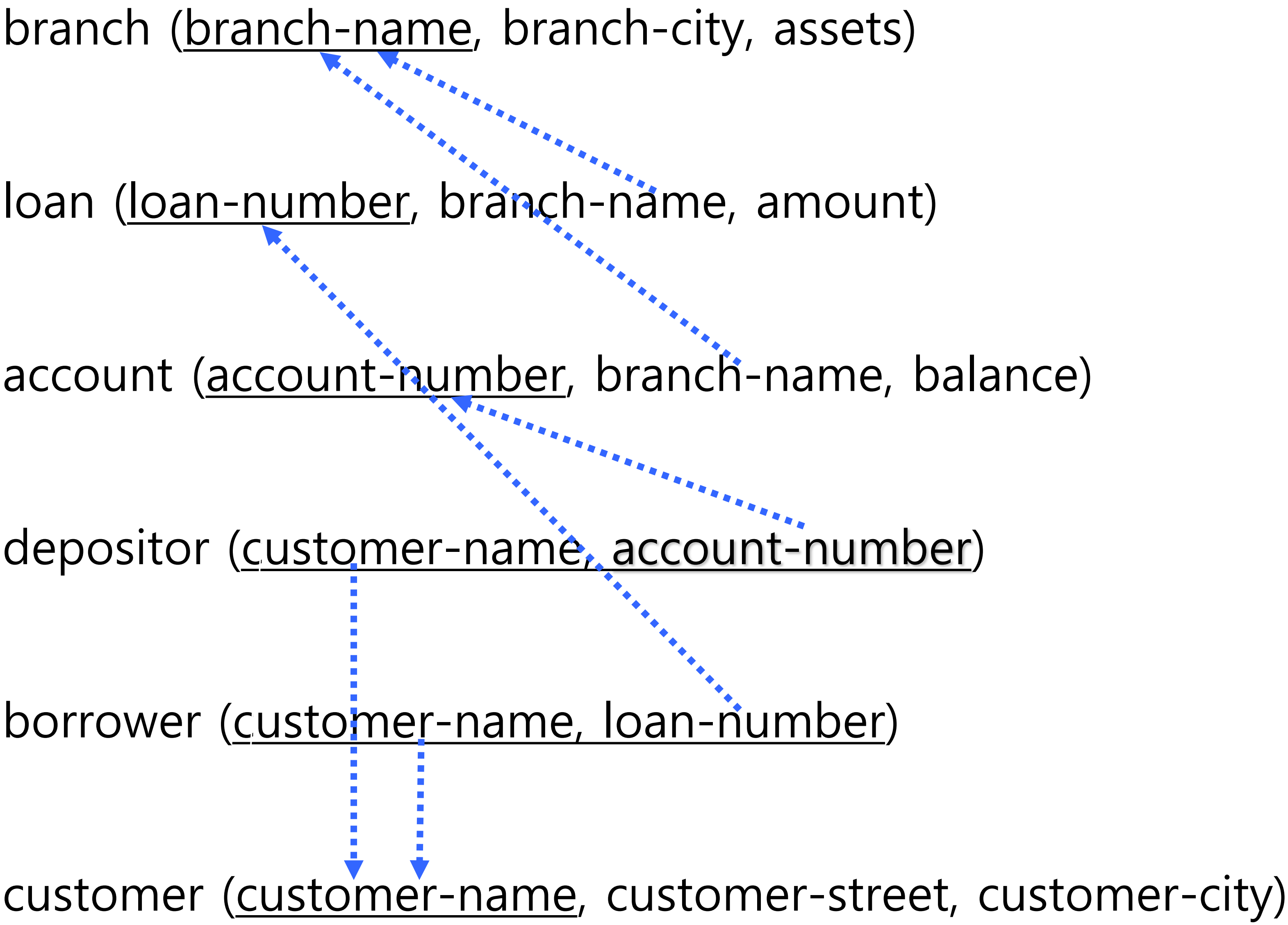
ACCOUNT (account-number, branch-name, balance)

DEPOSITOR (customer-name, account-number)

BORROWER (customer-name, loan-number)

CUSTOMER (customer-name, customer-street, customer-city)

Referential Integrity: Exercise



branch (branch-name, branch-city, assets)

loan (loan-number, branch-name, amount)

account (account-number, branch-name, balance)

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

customer (customer-name, customer-street, customer-city)

Referential Integrity: Exercise

- Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

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)

- Draw a relational schema diagram specifying the foreign keys for this schema.