Topic 3: Relational Data Model

Ts. Chng Chern Wei

Relational Data Model

The relational model is the theoretical basis of relational databases, which is a technique or way of structuring data using **relations / tables**, which are grid-like mathematical structures consisting of columns and rows.

Relational Data Model

In the relational model, all data is logically structured within relations, i.e., **tables**. Each relation has a name and is formed from named attributes or columns of data. Each tuple/row holds one value per attribute. The greatest strength of the relational model is the simple logical structure that it forms.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

Student	ID	Course	Name	Grade	Year	Instructor	Pre-Course
Brown	1	MATH90	Mathemat c I	B	07	James	DATA12
Brown	1	DATA12	Data strc	C	06	Diana	-
Brown	1						
JM	2		Mathemat c II				

Relational Model Concepts

The relational model represents the database as a collection of ***relations***.

A row is called a ***tuple***,
a column header is called an ***attribute***,
and the table is called a ***relation***.
The data type describing the types of values that can appear in each column is represented by a ***domain*** of possible values.

Domains, Attributes, Tuples, and Relations

A **domain** *D* is a set of **atomic** values.
By **atomic** we mean that each value in the domain is indivisible.
A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain, to help in interpreting its values.

- **Usa_phone_numbers**. The set of ten-digit phone numbers valid in the United States.
- **Local_phone_numbers**. The set of seven-digit phone numbers valid within a particular area code in the United States. The use of local phone numbers is quickly becoming obsolete, being replaced by standard ten-digit numbers.
- **Social_security_numbers**. The set of valid nine-digit Social Security numbers. (This is a unique identifier assigned to each person in the United States for employment, tax, and benefits purposes.)
- **Names**: The set of character strings that represent names of persons.

Domains, Attributes, Tuples, and Relations

A **data type** or **format** is also specified for each domain. For example, the data type for the domain **Usa_phone_numbers** can be declared as a character string of the form (*ddd*)*ddd-dddd*, where each *d* is a numeric (decimal) digit and the first three digits form a valid telephone area code.

A **domain** is thus given a **name**, **data type**, and **format**.
Additional information for interpreting the values of a domain can also be given; for example, a numeric domain such as **Person_weights** should have the units of measurement, such as pounds or kilograms.

Domains, Attributes, Tuples, and Relations

A **relation schema** *R*, denoted by $R(A_1, A_2, ..., A_n)$, is made up of a relation name *R* and a list of attributes, $A_1, A_2, ..., A_n$.
Each **attribute** A_i is the name of a role played by some domain *D* in the relation schema *R*.
D is called the **domain** of A_i and is denoted by **dom**($\mathbf{A}_i$). A relation schema is used to *describe* a relation; *R* is called the **name** of this relation.
The **degree** of a relation is the number of attributes *n* of its relation schema.

Domains, Attributes, Tuples, and Relations

A relation of degree seven, which stores information about university students, would contain seven attributes describing each student. as follows:

STUDENT*(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)*
Using the data type of each attribute, the definition is sometimes written as:
STUDENT*(****Name****: string, ****Ssn****: string, ****Home_phone****: string, ****Address****: string,*
Office_phone*: string, ****Age****: integer, ****Gpa****: real)*
(date , datetime, character, boolean, etc)

Domains, Attributes, Tuples, and Relations

A **relation** (or **relation state**) r of the relation schema $R(A_1, A_2, ..., A_n)$, also denoted by $r(R)$, is a set of n-tuples $r = \{t_1, t_2, ..., t_m\}$.
{Benjamin Bayer, Chong cha Kim, Dick Davidson …. }

Each ***n*-tuple** t is an ordered list of n values $t = <v_1, v_2, ..., v_n>$, where each value v_i, $1 \leq i \leq n$, is an element of dom(A_i) or is a special NULL value.
The ith value in tuple t, which corresponds to the attribute A_i, is referred to as $t[A_i]$ or $t.A_i$ (or $t[i]$ if we use the positional notation).

Domains, Attributes, Tuples, and Relations

A **relation** (or **relation state**) r of the relation schema $R(A_1, A_2, ..., A_n)$, also denoted by $r(R)$, is a set of n-tuples $r = \{t_1, t_2, ..., t_m\}$. Each ***n*-tuple** t is an ordered list of n values $t = <v_1, v_2, ..., v_n>$, where each value v_i, $1 \leq i \leq n$, is an element of dom(A_i) or is a special NULL value.

$t = <$ (Name, Dick Davidson),(Ssn, 422-11-2320),(Home_phone, NULL),(Address, 3452 Elgin Road), (Office_phone, (817)749-1253),(Age, 25),(Gpa, 3.53)>

Relation Name

STUDENT

Attributes

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Characteristics of Relations

1. Ordering of Tuples in a Relation.

A relation is defined as a *set* of tuples.

A relation is **not sensitive** to the ordering of tuples.

However, in a file, records are physically stored on disk (or in memory), so there always is an order among the records.

This ordering indicates first, second, *i*th, and last records in the file. Similarly, when we display a relation as a table, the rows are displayed in a certain order.

Characteristics of Relations

2. Ordering of Values within a Tuple and an Alternative Definition of a Relation. an *n*-tuple is an *ordered list* of *n* values, so the ordering of values in a tuple—and hence of attributes in a relation schema—is important. However, at a more abstract level, the order of attributes and their values is *not* that important as long as the correspondence between attributes and values is maintained.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Characteristics of Relations

3. Values and NULLs in the Tuples.

Each value in a tuple is an **atomic** value; that is, composite and multivalued attributes are not allowed (**flat relational model**).

NULL values, used to represent the values of attributes that may be unknown or may not apply to a tuple.

In general, we can have several meanings for NULL values, such as ***value unknown***, ***value*** exists but is ***not available***, or ***attribute does not apply*** to this tuple *(*also known as ***value undefined****).*

Characteristics of Relations

4. Interpretation (Meaning) of a Relation.

The relation schema can be interpreted as a declaration or a type of **assertion**. For example, the schema of the **STUDENT** relation asserts that, in general, a student entity has a *Name, Ssn,Home_phone, Address, Office_phone, Age,* and *Gpa*.

Each tuple in the relation can then be interpreted as a **fact** or a particular instance of the assertion. For example, the first tuple asserts the fact that there is a STUDENT whose **Name** is *Benjamin Bayer*, **Ssn** is *305-61-2435*, **Age** is *19*, and so on.

Relational Model Notation

A relation schema *R* of degree *n* is denoted by *R*(*A*1, *A*2, ..., *An*).

STUDENT*(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)*

STUDENT*(**Name**: string, **IC**: 001234134444, **Home_phone**: string, **Address**: string, **Office_phone**: string, **Age**: integer, **Gpa**: real)*

An attribute *A* can be qualified with the relation name *R* to which it belongs by using the dot notation *R.A*,

STUDENT*.Name* or ***STUDENT****.Age*, ***STUDENT****.Address*

Relational Model Notation

The letters *t*, *u*, *v* denote tuples

An *n*-tuple *t* in a relation *r*(*R*) is denoted by *t* = <*v*1, *v*2, ..., *vn*>,

where *vi* is the value corresponding to attribute *Ai*.

t *= <'Barbara Benson','533-69-1238','(817)839-8461',*

'7384 Fontana Lane', NULL, 19, 3.25>

t[Name] *= <'Barbara Benson'>*

t[Ssn, Gpa, Age] *= <'533-69-1238', 3.25, 19>*

Relational Database Schemas

A **relational database schema** *S* is a set of relation schemas S = {R_1, R_2, ..., R_m} and a set of **integrity constraints (IC)**.

A **relational database state** DB of *S* is a set of relation states DB = {r_1, r_2, ..., r_m} such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.

COMPANY = {EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, DEPENDENT}.

Relational Database 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 3.5
Schema diagram for COMPANY relational database schema.

Relational Model Constraints

Relational Integrity constraints is referred to conditions which must be present for a valid relation.

There are many types of integrity constraints. Constraints on the Relational database management system is mostly divided into three main categories are:

1. **Domain constraints**
2. **Unique Key constraints**
3. **Constraints on Null**
4. **Entity Integrity Constraint**
5. **Referential integrity Constraints**

1. Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types **integers, real numbers, characters, booleans, variable length strings**, etc.

2. Unique Key Constraints

NO DUPLICATE Primary KEY
An attribute that can uniquely identify a tuple in a relation is called the **key** of the table.

The value of the attribute for different tuples in the relation has to be unique.
Every relation should have at least **one** or **one set of attributes** which defines a tuple uniquely.

2. Unique Key Constraints

A relation may have more than one key, called a **candidate key**.
For example, the **CAR** relation has two candidate keys: ***License_number*** and ***Engine_serial_number***. It is common to designate one of the candidate keys as the **primary key** of the relation. This is the candidate key whose values are used to *identify* tuples in the relation.

The other candidate keys are designated as **unique keys**, and are not underlined.

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

3. Constraints on NULL Values

A constraint on attributes specifies whether NULL values are not permitted.

For example, if every **STUDENT** tuple must have a valid, non-NULL value for the **Name** attribute, then Name of **STUDENT** is constrained to be NOT NULL.

4. Entity Integrity Constraint

The **entity integrity constraint** states that no **primary key** value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples.

5. Referential Integrity

Referential integrity constraints is base on the concept of **Foreign Keys**.

A foreign key is an important attribute of a relation which should be referred to in other relationships.

Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

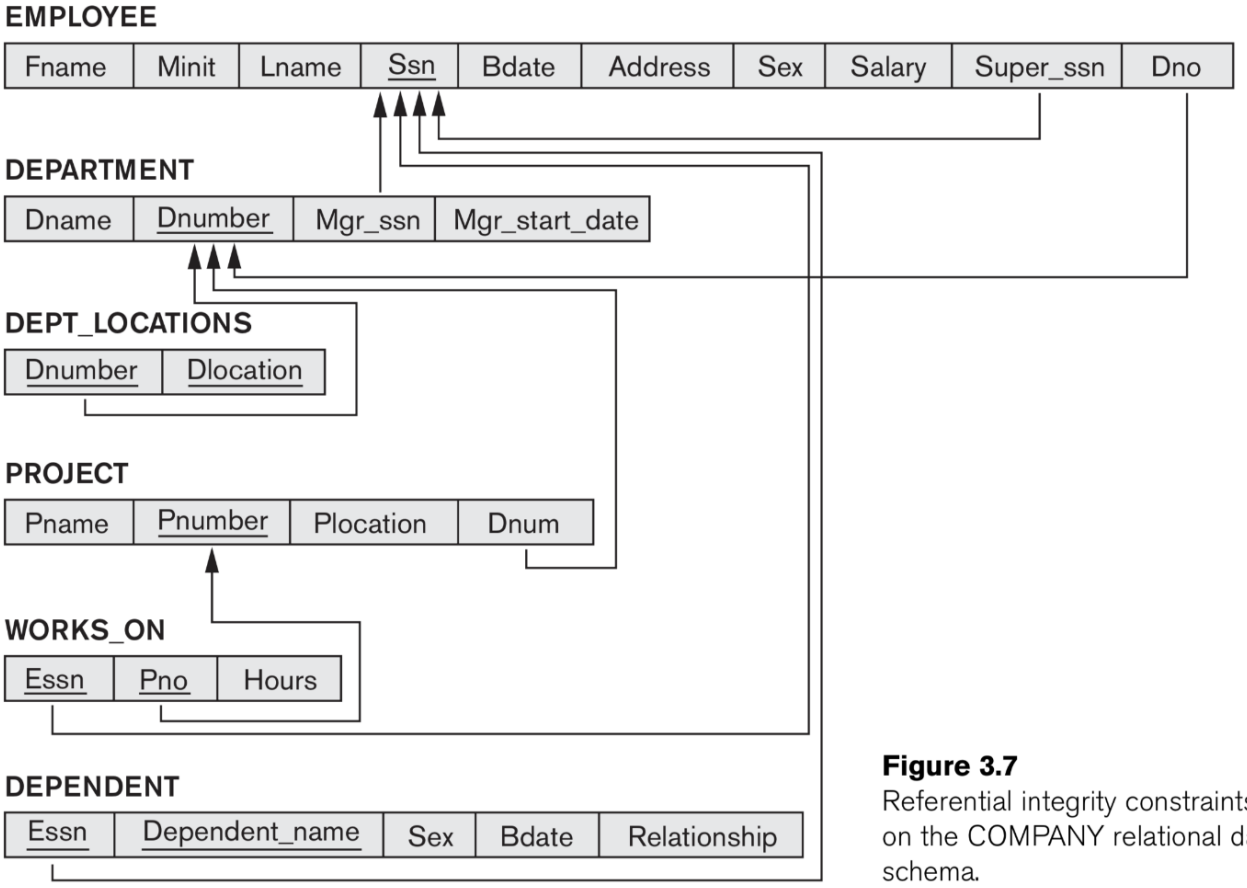


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

6. Semantic integrity constraints (Application)

a large class of general constraints, ***semantic integrity constraints,*** which may have to be specified and enforced on a relational database.

Examples: *the salary of an employee should not exceed the salary of the employee's supervisor* and *the maximum number of hours an employee can work on all projects per week is 56*.
Such constraints can be specified and enforced within the application programs that update the database, or by using a general-purpose **constraint specification language**.

Database States

A database state that does not obey all the integrity constraints is called an **invalid state**.

A state that satisfies all the constraints in the defined set of integrity constraints IC is called a **valid state.**

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

INSERT Operations (**Domain constraints)**

The **Insert** operation provides a list of attribute values for a new tuple *t* that is to be inserted into a relation *R*.

Domain constraints can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type.
Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.
Result: This insertion violates the domain constraint because 1960 is not the complete date format.

INSERT Operations (**Key constraints**)

The **Insert** operation provides a list of attribute values for a new tuple *t* that is to be inserted into a relation *R*.

Key constraints can be violated if a key value in the new tuple *t* already exists in another tuple in the relation *r*(*R*).

Insert <'James', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

INSERT Operations (**Entity integrity**)

The **Insert** operation provides a list of attribute values for a new tuple *t* that is to be inserted into a relation *R*.

Entity integrity can be violated if any part of the primary key of the new tuple *t* is NULL.
Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.
Result: This insertion violates the **entity integrity constraint** (NULL for the primary key Ssn), so it is rejected.

INSERT Operations (**Referential integrity**)

The **Insert** operation provides a list of attribute values for a new tuple *t* that is to be inserted into a relation *R*.

Referential integrity can be violated if the value of any foreign key in *t* refers to a tuple that does not exist in the referenced relation.

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

DELETE Operation (**Referential integrity**)

The **Delete** operation can violate only **referential integrity**. This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database.

- ***Operation***:
  Delete the WORKS_ON tuple with Essn = ‘999887777’ and Pno = 10.
  Result: This deletion is acceptable and deletes exactly one tuple.
- ***Operation***:
  Delete the EMPLOYEE tuple with Ssn = ‘999887777’.
  Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

UPDATE Operation (**Referential integrity**)

The **Update** (or **Modify**) operation is used to change the values of one or more attributes in a tuple (or tuples) of some relation *R*. It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified.

■ ***Operation***:

Update the salary of the EMPLOYEE tuple with Ssn = ‘999887777’ to 28000.

Result: Acceptable.

■ ***Operation***:

Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 7.

Result: Unacceptable, because it violates referential integrity.

UPDATE Operation

Operation:
Update the Ssn of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘987654321’.
Result: Unacceptable, because it violates primary **key constraint** by repeating a value that already exists as a primary key in another tuple; it violates **referential integrity constraints** because there are other relations that refer to the existing value of Ssn.

The **Transaction** Concept

A **transaction** is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database. At the end of the transaction, it must leave the database in a valid or consistent state that satisfies all the constraints specified on the database schema.

A single transaction may involve any number of retrieval operations, and any number of update operations. These retrievals and updates will together form an atomic unit of work against the database.

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

Exercise 1

Suppose that each of the following Update operations is applied directly to the database state shown in Figure 3.6. Discuss *all* integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.

a. Insert <‘Robert’, ‘F’, ‘Scott’, ‘943775543’, ‘1972-06-21’, ‘2365 Newcastle Rd, Bellaire, TX’, M, 58000, ‘888665555’, 1> into EMPLOYEE.
b. Insert <‘ProductA’, 4, ‘Bellaire’, 2> into PROJECT.
c. Insert <‘Production’, 4, ‘943775543’, ‘2007-10-01’> into DEPARTMENT.
d. Insert <‘677678989’, NULL, ‘40.0’> into WORKS_ON.
e. Insert <‘453453453’,‘John’,‘M’,‘1990-12-12’,‘spouse’> into DEPENDENT.
f. Delete the WORKS_ON tuples with Essn = ‘333445555’.
g. Delete the EMPLOYEE tuple with Ssn = ‘987654321’.
h. Delete the PROJECT tuple with Pname = ‘ProductX’.
i. Modify the Mgr_ssn and Mgr_start_date of the DEPARTMENT tuple withDnumber = 5 to ‘123456789’ and ‘2007-10-01’, respectively.
j. Modify the Super_ssn attribute of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘943775543’.
k. Modify the Hours attribute of the WORKS_ON tuple with Essn = ‘999887777’ and Pno = 10 to ‘5.0’.

Exercise 2

Consider the following six relations for an order-processing database application in a company:

CUSTOMER(Cust#, Cname, City)

ORDER(Order#, Odate, Cust#, Ord_amt)

ORDER_ITEM(Order#, Item#, Qty)

ITEM(Item#, Unit_price)

SHIPMENT(Order#, Warehouse#, Ship_date)

WAREHOUSE(Warehouse#, City)

Here, ***Ord_amt*** refers to total dollar amount of an order; ***Odate*** is the date the order was placed; and ***Ship_date*** is the date an order (or part of an order) is shipped from the warehouse. Assume that an order can be shipped from several warehouses.

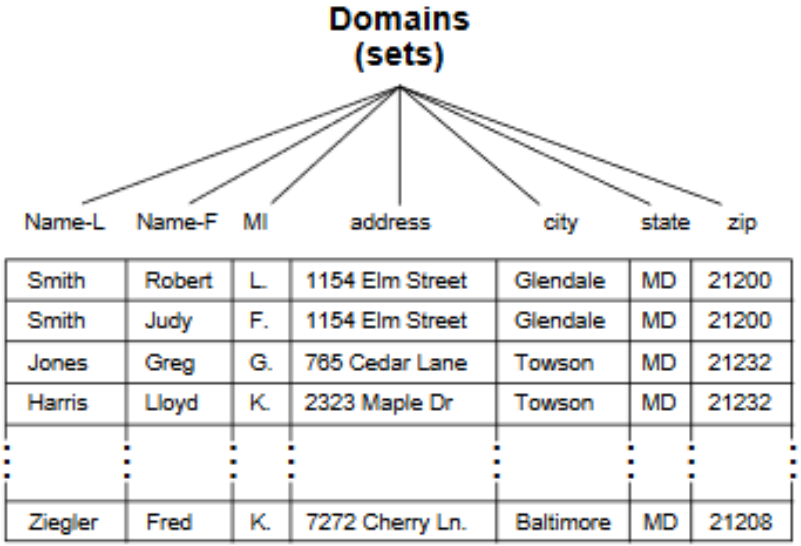
Specify the foreign keys for this schema, stating any assumptions you make.

What other constraints can you think of for this database?

Relations as a Database

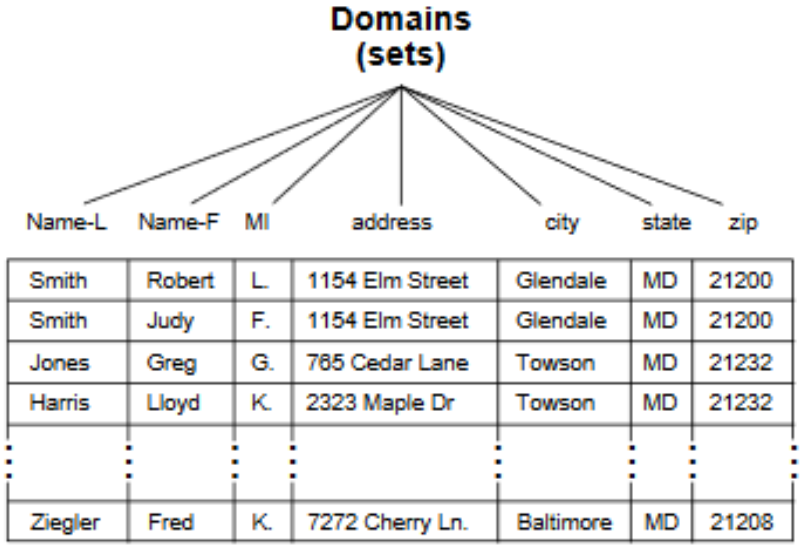
The business data file resembles a relation in a number of ways. The tabular file itself corresponds to a relation.

Each column, or attribute, in the file corresponds to a particular set and all of the values from a particular column come from the same domain, or set. Each row, or record, in the file corresponds to a tuple.



Name-L	Name-F	MI	address	city	state	zip
Smith	Robert	L.	1154 Elm Street	Glendale	MD	21200
Smith	Judy	F.	1154 Elm Street	Glendale	MD	21200
Jones	Greg	G.	765 Cedar Lane	Towson	MD	21232
Harris	Lloyd	K.	2323 Maple Dr	Towson	MD	21232
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ziegler	Fred	K.	7272 Cherry Ln.	Baltimore	MD	21208

Relations as a Database



Name-L	Name-F	MI	address	city	state	zip
Smith	Robert	L.	1154 Elm Street	Glendale	MD	21200
Smith	Judy	F.	1154 Elm Street	Glendale	MD	21200
Jones	Greg	G.	765 Cedar Lane	Towson	MD	21232
Harris	Lloyd	K.	2323 Maple Dr	Towson	MD	21232
Ziegler	Fred	K.	7272 Cherry Ln.	Baltimore	MD	21208

If such a file is to be genuinely interchangeable with a relation, certain constraints must be met:

- every tuple must be unique
- every attribute within a tuple must be single-valued
- in all tuples, the values for the same attribute must come from the same domain or set
- no attributes should be null

An essential attribute of a relation is that every tuple must be unique.

This means that the values present in some individual attribute (or set of attributes) must always provide enough information to allow a unique identification of every tuple in the relation.

In a relational database, these identifying values are known as **key values** or just as the **key**.

Sometimes more than one key could be defined for given table. For example, in the table below (which represents, perhaps, a patient record file), several columns might serve as a **key**.

Either patient number (assigned by the hospital) or social security number (brought with the patient) are possibilities.

Any attribute or set of attributes that might possibly serve as a key is known as a **candidate key**. Keys that involve only one attribute are known as **simple keys**. Keys that involve more than one attribute are **composite keys**.

patient #	SS #	Last Name	address	birth date
P-64122	123-45-6789	Smith	123 Main Street	10 MAY 44
P-75642	001-32-6873	Pedersen	1700 Cedar Barn Way	31 MAR 59
P-70875	444-44-5555	Wilson	1321 North South St	7 AUG 90
P-79543	555-12-1212	Grant	808 Farragut Avenue	1 DEC 68
⋮	⋮	⋮	⋮	⋮
P-71536	888-88-8888	MacPherson	1617 Pennsylvania Ave	11 APR 60

In designing a database, one of the candidate keys for each relation must be chosen to be the **primary key** for that table. Choosing primary keys is a crucial task in database design. If keys need to be redesignated, the entire system may have to be redone.

Primary keys can never be null and should never be changed.

patient #	SS #	Last Name	address	birth date
P-64122	123-45-6789	Smith	123 Main Street	10 MAY 44
P-75642	001-32-6873	Pedersen	1700 Cedar Barn Way	31 MAR 59
P-70875	444-44-5555	Wilson	1321 North South St	7 AUG 90
P-79543	555-12-1212	Grant	808 Farragut Avenue	1 DEC 68
⋮	⋮	⋮	⋮	⋮
P-71536	888-88-8888	MacPherson	1617 Pennsylvania Ave	11 APR 60

A **binary relation** (i.e., a subset of a Cartesian product of two sets) could be presented in a computer system as two-column tabular file, with one member from the first set named in the first column of each record and one member of the second set in the second column.

For example, a binary relation could be used to provide unique three-letter identifiers for academic departments. Additional relations could be used to give more information about individual departments or individual faculty members.

ZOL	Zoology
PSD	Political Science
CPS	Computer Science
HIS	History
⋮	⋮
ACC	Accounting

ZOL	Zoology	Room 203	Natural Science Bldg	355 4640
CPS	Computer Science	Room 714A	Wells Hall	355 5210
BSP	Biological Science	Room 141	Natural Science Bldg	353 4610
CEM	Chemistry	Room 320	Chemistry Bldg	355 9175
⋮	⋮	⋮	⋮	⋮
PSD	Political Science	Room 303	South Kedzie Hall	355 6590

Whenever the values in an attribute column in one table “point to” primary keys in another (or the same) table, the attribute column is said to be a **foreign key**.

Columns containing foreign keys are subject to an integrity constraint: any value present as a foreign key must also be present as a primary key

ZOL	Zoology
PSD	Political Science
CPS	Computer Science
HIS	History
⋮	⋮
ACC	Accounting

ZOL	Room 203	Natural Science Bldg	355 4640
CPS	Room 714A	Wells Hall	355 5210
BSP	Room 141	Natural Science Bldg	353 4610
CEM	Room 320	Chemistry Bldg	355 9175
⋮	⋮	⋮	⋮
PSD	Room 303	South Kedzie Hall	355 6590

Q&A?

Thank You!

Reference:

Abraham Silberschatz (2019), Database System Concepts. McGraw-Hill Education

Carlos Coronel et., al (2017) Database Systems: Design, Implementation, & Management, 12th Edn. Cengage Learning