# Lecture 4
# Relational Model

# Data Models and Database Design

- Think “logically” when design a database, need some kind of framework to design the database
- Like designing a data structure in some programming language, a data model is like a type system, but is abstract
- Organize the data into tables in the relational data model
- Initially no need to worry about how these tables are implemented

# Relational Model Concepts

- The relational model of data is based on the concept of a **mathematical relation**
- A **relation** is a mathematical concept based on set theory
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations

# Relational Model Concepts

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper: "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970

  *The above paper caused a major revolution in the field of Database management and earned Ted Codd the coveted Turing Award*

# What is a Relational Database?

**Routes**

| RId | RName | Grade | Rating | Height |
|---|---|---|---|---|
| 1 | Last Tango | II | 12 | 100 |
| 2 | Garden Path | I | 2 | 60 |
| 3 | The Sluice | I | 8 | 60 |
| 4 | Picnic | III | 3 | 400 |

**Climbers**

| CId | Cname | Skill | Age |
|---|---|---|---|
| 123 | Edmund | EXP | 80 |
| 214 | Arnold | BEG | 25 |
| 313 | Bridget | EXP | 33 |
| 212 | James | MED | 27 |

**Climbs**

| CId | RId | Date | Duration |
|---|---|---|---|
| 123 | 1 | 10/10/88 | 5 |
| 123 | 3 | 11/08/87 | 1 |
| 313 | 1 | 12/08/89 | 5 |
| 214 | 2 | 08/07/92 | 2 |
| 313 | 3 | 06/07/94 | 3 |

# Can two columns in the same relation have the same name?

- Yes
- No

Total Results: 0

# Can two columns in the same relation have the same name?

# Can two columns in the same relation have the same name?

# Why is the Database Like This?

- Each **route** has an *id*, a *name*, a *grade* (an estimate of the time needed), a *rating* (how difficult it is), and a *height*
- Each **climber** has an *id*, a *name*, a *skill* level and an *age*
- A **climb** records who climbed what route on what *date* and how long it took (*duration*)
- The data values in these tables are all “simple”

# Describing Relations

- Relations are described by a schema which can be expressed in various ways
- A database schema is usually expressed in a data definition language (DDL)

```
Routes(RId:int, RName:string, Grade:string,
       Rating:int, Height:int)
Climbers(CId:int, CNname:string,
         Skill:string, Age:int)
Climbs(CId:int, RId:int, Date:date,
       Duration:int)
```

# Expressing Constraints

- In SQL, constraints are defined as follows:

```
CREATE TABLE Climbers
  (CId INTEGER,
   CName CHAR(20),
   Skill CHAR(4),
   Age INTEGER,
   PRIMARY KEY (CId),
   UNIQUE (CName,Age));
```

```
CREATE TABLE Climbs
  (CId INTEGER,
   RId INTEGER,
   Date DATE,
   Duration INTEGER,
   PRIMARY KEY (CId, RId, Date),
   FOREIGN KEY (CId) REFERENCES
                     Climbers,
   FOREIGN KEY (RId) REFERENCES
                     Routes);
```

# Informal Definition

- RELATION: A table of values
  - A relation is a set of rows
  - A relation is alternatively a set of columns
  - Each row represents a fact that corresponds to a real-world **entity** or **relationship**
  - Each row has an item or set of items that uniquely identifies that row in the table
  - Sometimes row-ids or sequential numbers are assigned to identify the rows in the table
  - Each column is called by its column name or column header or attribute name

# Relational Model Terminology

- Table = relation
- Column headers = attributes
- Row = tuple
- Possible values of each attribute = domain
  - E.g., the domain of **CName** is string, the domain of **Rating** is real
- Relation schema = relation name + attributes + other structure information
  - E.g., keys, other constraints
- Relation instance is the current set of rows for a relation schema
- Database schema = collection of relation schemas

# Schema Definition

- The **schema** of a relation: $R(A_1, A_2, \ldots A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots A_n$
  - R: Name
  - Degree: # of attributes *n*
  - Cardinality: # of tuples (rows)

# Example

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

- CUSTOMER is a relation defined over four **attributes** Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values
  - E.g., the domain of Cust-id could be 6 digit numbers
- A **tuple** is an ordered set of values
  - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
- A relation may be regarded as a set of tuples (rows)

# Formal Definition

- The relation is a subset of the Cartesian product of the domains
  - Domain is used in a specific role conveyed by the attribute name
  - E.g., attribute Cust-name is defined over the domain of strings of 26 characters. The role these strings play in the CUSTOMER relation is that of the name of customers
- Formally,

  Given $R(A_1, A_2, ....., A_n)$

  $r(R) \subseteq dom(A_1) \times dom(A_2) \times ....X\ dom(A_n)$

  R: relation schema (**intension)**

  r(R): a specific "value" or population of R (**extension**)

# Example

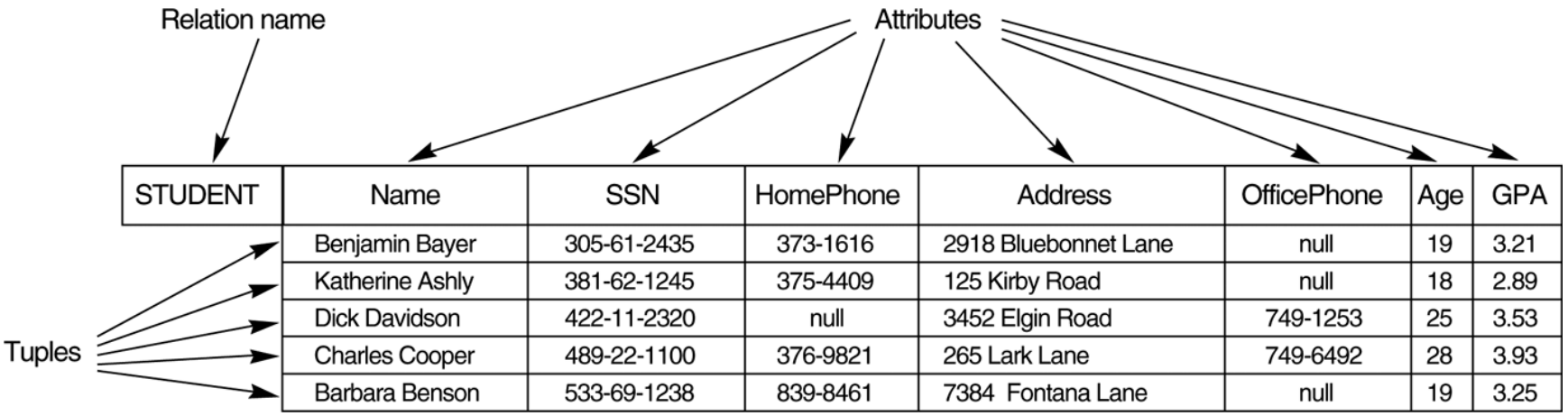
- $D_1 = \{0,1\}$
- $D_2 = \{a, b, c\}$
- $r(R) \subseteq D_1 \text{ X } D_2$
- For example: r(R) = {<0,a> , <0,b> , <1,c> }

  is one possible “state” or “population” or “extension” r of the relation R, defined over domains $D_1$ and $D_2$

# Definition Summary

| Informal Terms | Formal Terms |
|---|---|
| Table | Relation |
| Column name | Attribute |
| Row | Tuple |
| Values in a column | Domain |
| Table Definition | Schema of a Relation |
| Populated Table | Extension |

# Example




| STUDENT | Name | SSN | HomePhone | Address | OfficePhone | Age | GPA |
|---|---|---|---|---|---|---|---|
| | Benjamin Bayer | 305-61-2435 | 373-1616 | 2918 Bluebonnet Lane | null | 19 | 3.21 |
| | Katherine Ashly | 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 |
| | Charles Cooper | 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 |

# More Relational Model Notations

- $t[A_i] = v_i$ (the value of attribute $A_i$ for tuple t)
- $t[A_u, A_v, ..., A_w]$: the tuple of t containing the values of attributes $A_u, A_v, ..., A_w$

# Characteristics of Relations

- **Ordering of tuples in a relation r(R)**:
  - Not ordered (like a set)
- **Ordering of attributes in a relation schema R**
  - Not important
  - However, the correspondence between the attributes in $R(A_1, A_2, ..., A_n)$ and the values in $t=<v_1, v_2, ..., v_n>$ is maintained

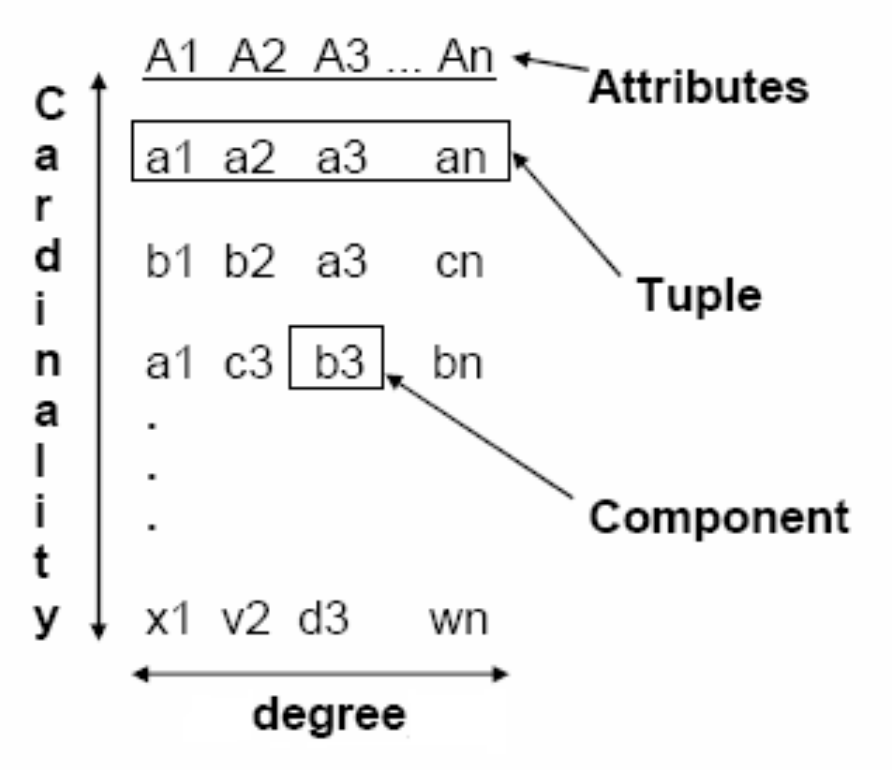
# Characteristics of Relations (Cont.)

- **Values in a tuple**:
  - A set of *atomic* values
  - Each value in the domain is indivisible
  - Composite and multi-valued attributes are not allowed
  - **null** value: unknown or inapplicable to certain tuples

# 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 |

# Relational Data Model



# Relational Model Constraints

- Constraints are *conditions* that must hold on **all** valid relation instances
- Three main constraints:
  1. **Key** constraints (single relation)
  2. **Entity integrity** constraints (single relation)
  3. **Referential integrity** constraints (two relations)

# Key Constraints

- **Key constraints**: No two tuples can have the same value for the key
- **Superkey**: A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK
  - For any distinct tuples $t_1$ and $t_2$ in r(R), $t_1[SK] \neq t_2[SK]$
- **Candidate Key**: A "minimal" superkey
  - A superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey

# Example 1

CAR(State, Reg#, SerialNo, Make, Model, Year)

- Two candidate keys:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
  - Are Key1 and Key2 superkeys?
- {SerialNo, Make}: candidate key or superkey?
- **Primary key**: If a relation has several candidate keys, one is chosen arbitrarily to be the primary key
  - The primary key is underlined
  - Implies “not null”

# Example 2

- The CAR relation, with two candidate keys: License-number and Engine_serial_number

**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 |

# Entity Integrity

- **Relational Database Schema**: A set of relation schemas that belong to the same database
  - S = {$R_1$, $R_2$, ... $R_n$}, S is the name of the **database**
- **Entity Integrity**: The PK (primary key) attributes of each relation schema R in S cannot have null values in any tuple of r(R)
  - t[PK] ≠ null for any tuple t in r(R)
  - Why?
- Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key

# Referential Integrity

- If a tuple in $R_1$ refers to $R_2$, it must refer to an **existing** tuple in $R_2$
- E.g., Dno in every Employee tuple must match Dnumber value of some tuple in the Department relation

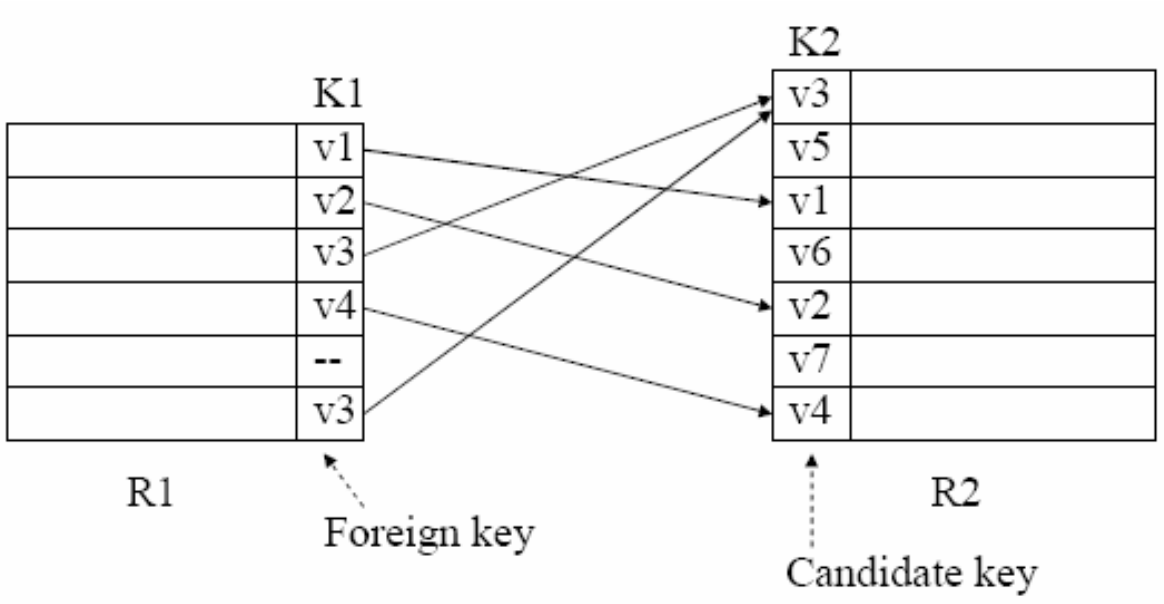
# Foreign Key Constraints

- A foreign key constraint mostly involves *two* relations
- Specify a *relationship* among tuples in two relations:
  - The **referencing relation** ($R_1$) and the **referenced relation** ($R_2$)
  - FK (**foreign key** attributes) of $R_1$ reference PK (primary key attributes) or candidate key of $R_2$
- Displayed in a relational database schema as a directed arc from $R_1$.FK to $R_2$.PK or $R_2$.CK

# Foreign Key

- A set of attributes in $R_1$ is a foreign key of $R_1$ that references relation $R_2$ if it satisfies the following two rules:
  - The attributes in FK have the same domain(s) as the primary key (PK) attributes or candidate key of $R_2$
  - A value in the foreign key column (or columns) of the the **referencing relation** $R_1$ can be either:
    1. a value of an existing primary key or candidate key value in the **referenced relation** $R_2$
    2. is null
- In case (2), the FK in $R_1$ should not be a part of its own primary key

# Example




- If no row exists in R2– violation of referential integrity
- Not all rows in R2 need to be referenced
- Value of a foreign key might not be specified
- Names of K1 and K2 need not to be the same

# Does foreign key need to be unique?

Yes

No

Total Results: 0

# Does foreign key need to be unique?

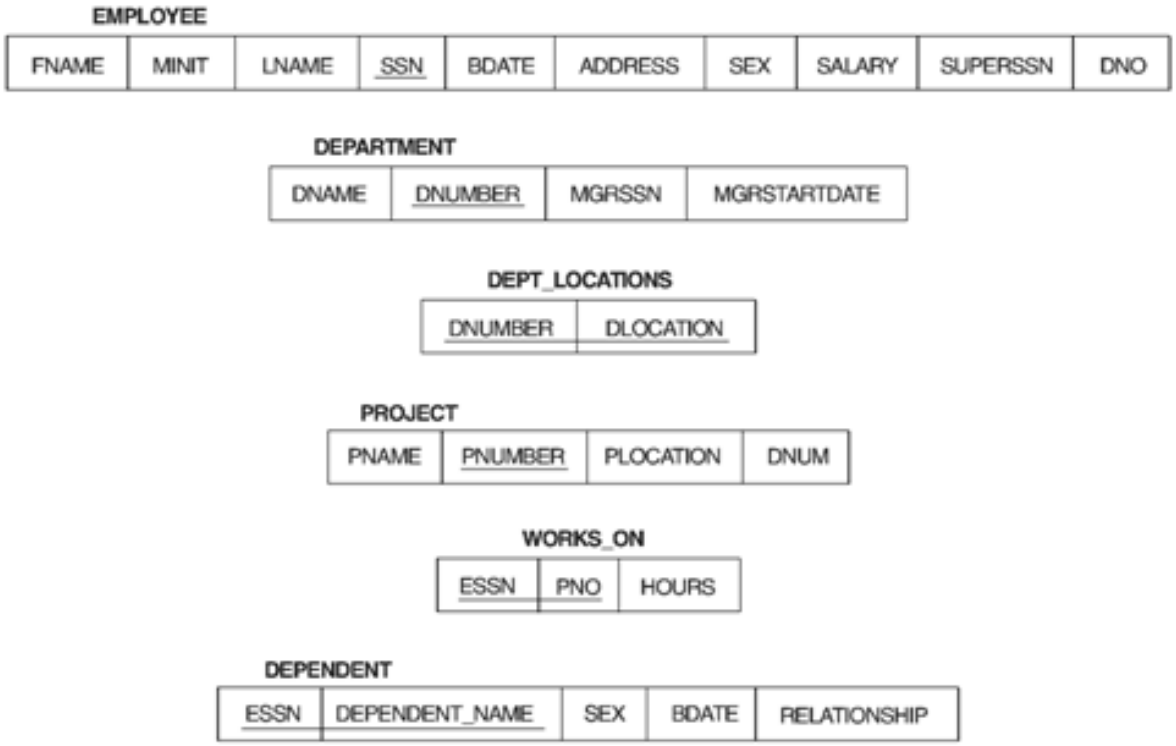
# Does foreign key need to be unique?

# Foreign Key (Cont.)

- Foreign key can refer to its own relation
- For example: SuperSSN is a FK that refers to SSN of Employee relation itself

# Example 1

- Schema diagram for the COMPANY relational database schema (the primary keys are underlined)



# Example 1 (Cont.)

- One possible relational database state corresponding to the COMPANY 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**

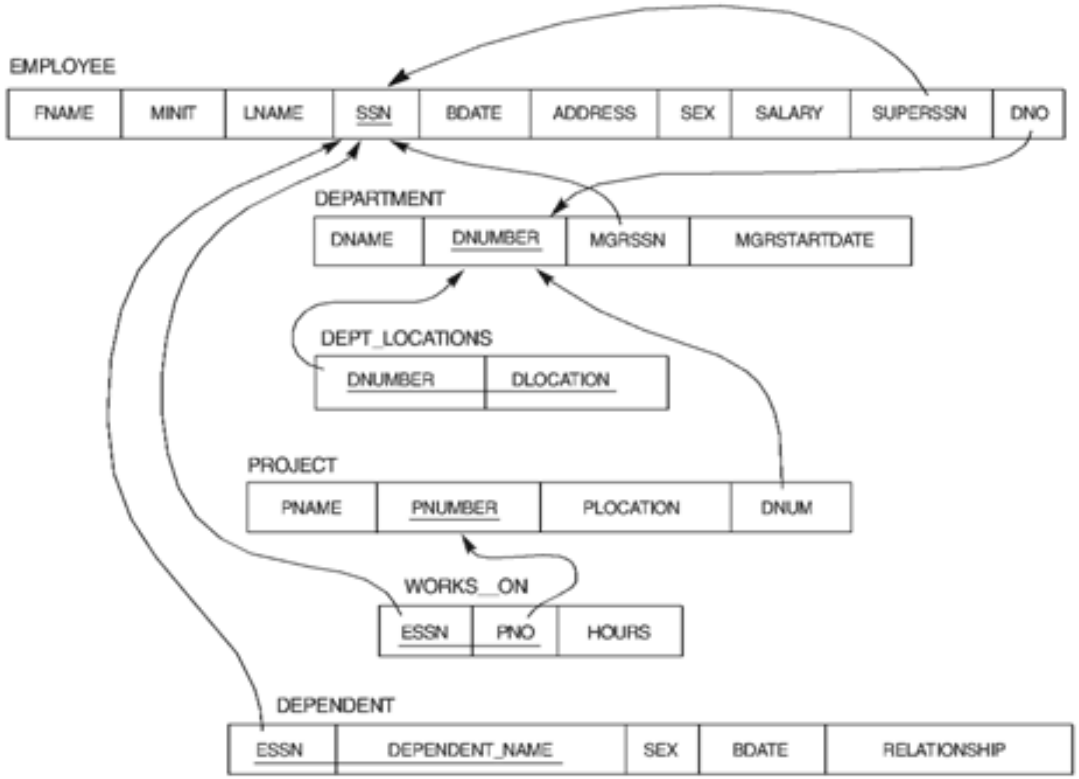
| Pname | Pnumber | Plocation | Dnum |
|---|---|---|---|
| ProductX | 1 | Bellaire | 5 |
| ProductY | 2 | Sugarland | 5 |
| ProductZ | 3 | Houston | 5 |
| Computerization | 10 | Stafford | 4 |
| Reorganization | 20 | Houston | 1 |
| Newbenefits | 30 | Stafford | 4 |

**DEPENDENT**

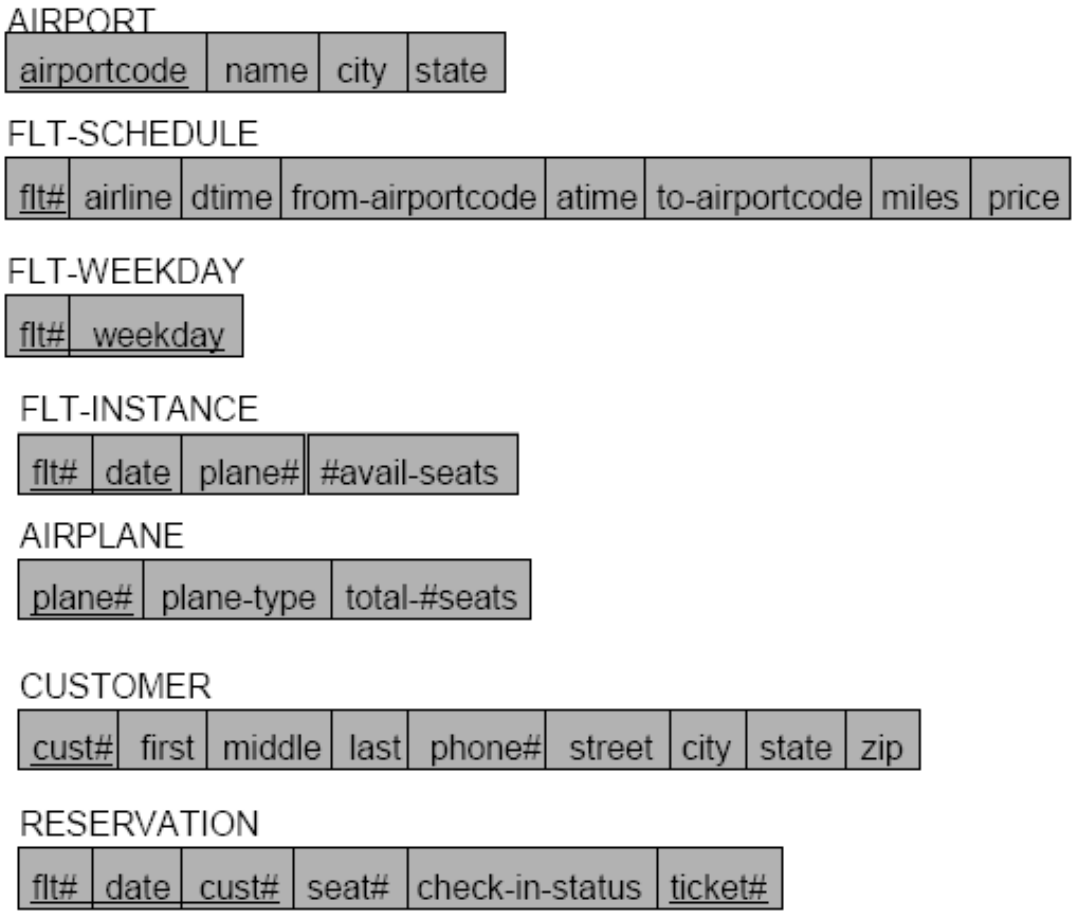
| Essn | Dependent_name | Sex | Bdate | Relationship |
|---|---|---|---|---|
| 333445555 | Alice | F | 1986-04-05 | Daughter |
| 333445555 | Theodore | M | 1983-10-25 | Son |
| 333445555 | Joy | F | 1958-05-03 | Spouse |
| 987654321 | Abner | M | 1942-02-28 | Spouse |
| 123456789 | Michael | M | 1988-01-04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

# Example 1 (Cont.)

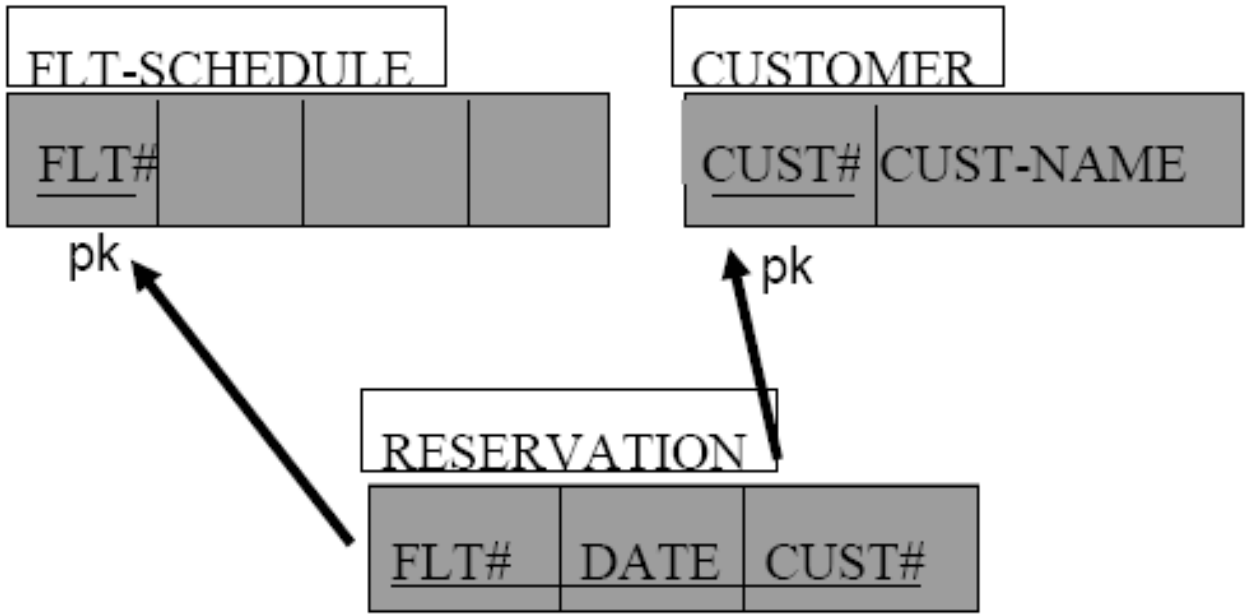
- Foreign key constraints displayed on the COMPANY relational database schema diagram



# Example 2


AIRPORT
airportcode
name
city
state
FLT-SCHEDULE
flt#
airline
dtime
from-airportcode
atime
to-airportcode
miles
price
FLT-WEEKDAY
flt#
weekday
FLT-INSTANCE
flt#
date
plane#
#avail-seats
AIRPLANE
plane#
plane-type
total-#seats
CUSTOMER
cust#
first
middle
last
phone#
street
city
state
zip
RESERVATION
flt#
date
cust#
seat#
check-in-status
ticket#

# Example 2 (Cont.)



# Other Types of Constraints

- Domain constraints: values of each attribute A must be an atomic value from the domain for that attribute, dom(A)
- Semantic Integrity Constraints:
  - Based on application semantics and cannot be expressed in a database schema
  - E.g., “one student cannot register for more than 9 credits”
  - A *constraint specification language* may have to be used to express these constraints
    - SQL-99 uses triggers and assertions

# 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)

**Add foreign key constraints to the schema above**

# Update Operations on Relations

- INSERT a tuple
  - Domain, key, entity, and referential integrity constraints can be violated
- DELETE a tuple
  - Only referential integrity constraint can be violated
- MODIFY a tuple
  - Modify PK: delete one tuple and insert another in its place
  - Modify FK: make sure the new value refers to an existing tuple in the referenced relation (or is null)
  - Neither PK nor FK: only check new value is of the correct domain and data type

# Update Operations on Relations (Cont.)

- 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

# Update Operations on Relations (Cont.)

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (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

# Example

- The instances below satisfy some constraints

**Climbers:**

| CId | CName | Skill | Age |
|---|---|---|---|
| 123 | Edmund | EXP | 80 |
| 214 | Arnold | BEG | 25 |
| 313 | Bridget | EXP | 33 |
| 212 | James | MED | 27 |

**Climbs:**

| CId | RId | Date | Duration |
|---|---|---|---|
| 123 | 1 | 10/10/88 | 5 |
| 123 | 3 | 11/08/87 | 1 |
| 313 | 1 | 12/08/89 | 5 |
| 214 | 2 | 08/07/92 | 2 |
| 313 | 1 | 06/07/94 | 3 |

- Insert (123, Jeremy, MED, 16) into Climbers?
- Insert (456, 2, 09/13/98, 3) into Climbs?
- Delete (313, Bridget, EXP, 33) from Climbers?
- Modify 123 to 456 in Climbers?