The Relational Model theoretical foundation

- Relational Model Concepts
- Characteristics of Relations
- Relational Integrity Constraints
  - Key Constraints
  - Entity Integrity Constraints
  - Referential Integrity Constraints
- Operations

Data Models and database design

- When we design a database we try to think "*logically*", but need some kind of framework in which to design the database.
- It is like designing a data structure in some programming language. You might use arrays, lists, etc. depending on what is available. A data model is like a type system, but is abstract.
- In the relational data model we organize the data into tables. We don't (initially) worry about how these tables are implemented.

Relational Model

- A particular way of structuring data (relations)
- Simple
- Mathematically based
  - Expressions (queries) can be analyzed by DBMS
  - Transformed to equivalent expressions automatically (query optimization)

Basis of Relational Model

- A **RELATION** is a mathematical concept based on the ideas of sets.
- Relational model of data is based on the concept of a **Relation.**
- 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.

Informal Definition

- **RELATION:** A table of values
- A relation may be thought of as a **set of rows.**
- A relation may alternately be though of as a **set of columns.**
- Each row of the relation may be given an identifier.
- Each column typically is called by its column name or column header or attribute name.

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
215	3	06/07/94	3

- 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`).
- Observe that the data values in these tables are all "simple". None of them are complex structures -- like other relations.

Relational Model Terminology

- Table = *relation*.
- Column headers = *attributes*
- Row = *tuple*
- The possible value of each attribute = *domain*
  - E.g., the domain of **`CName`** is `string` and that for **`Rating`** is `real`
- *Relation schema* = name(attributes) + other structure info.,
  - e.g., keys, other constraints.
- *Relation instance* is current set of rows for a relation schema.
- *Database schema* = collection of relation schemas.

FORMAL DEFINITIONS

- A **Relation** may be defined in multiple ways
- The **Schema** of a Relation:

  R (A1, A2, ...An)

  Relation R is defined over **attributes** A1, A2, ...An

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

- CUSTOMER is a *relation* defined over the 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 is 6 digit numbers.
- A ***tuple*** is an ordered set of values
  - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000"> is a tuple belonging to the CUSTOMER relation.
- A relation may be regarded as a set of tuples (rows).

Formal Definition (Cont.)

- A relation may be regarded as a set of tuples
- A relation is formed over the cartesian product of the sets
  - each set has values from a **domain**
  - domain is used in a specific role which is conveyed by the attribute name.
  - E.g., attribute **Cust-name:** the domain of strings of 25 characters.
- Formally,

Given R(A1, A2, ..., An)

r(R) subset-of dom (A1) X dom (A2) X ....X dom(An)

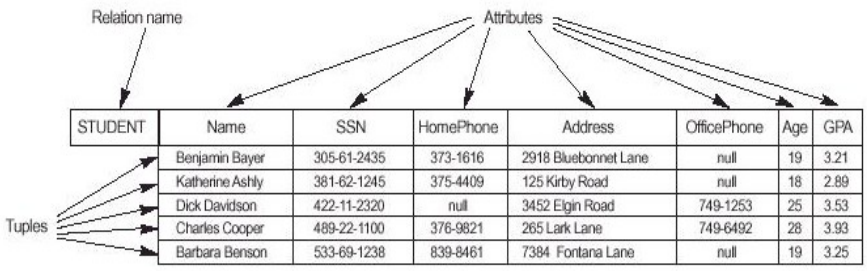
R: schema of the relation (**intension**)

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

Definition Summary

Informal Terms	**Formal Term**
Table	Relation
Column	Attribute/Domain
Values in a column	Domain
Row	Tuple/Instance
Table Definition	Schema of Relation
Populated Table	Extension

Figure 7.1 The attributes and tuples of a relation STUDENT.



Characteristics of Relations

- **Ordering of tuples in a relation r(R)**:
  - Not ordered (like a set)
- **Ordering of attributes in a relation schema R** (and of values within each tuple):
  - the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be *ordered*
  - Not important

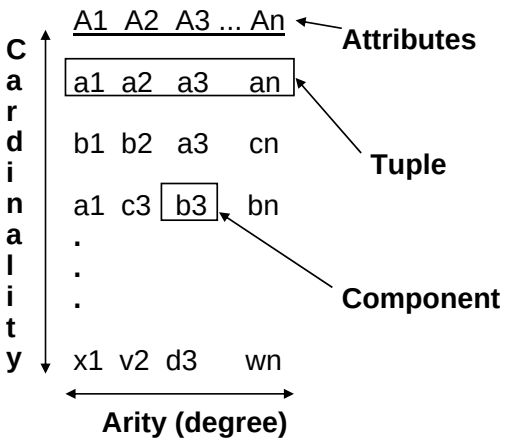
- **Values in a tuple** : *atomic* (indivisible)
  - **Null** value: unknown or inapplicable values to certain tuples
- Notation
  - t[Ai] = vi (the value of attribute Ai for tuple t).
  - t[Au, Av, ..., Aw] refers to the tuple of t containing the values of attributes Au, Av, ..., Aw, respectively.

Figure 7.2 The relation STUDENT from Figure 7.1, with a different order of tuples.

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

Relational Data Model

Relation as table
- Rows = tuples
- Names of columns = attributes
- Set of attribute names = schema
  - REL (A1,A2,...,An)



Set theoretic
- Domain — set of values
  - like a data type
- Cartesian product (or product)
  - $D1 \times D2 \times ... \times Dn$
  - n-tuples (V1,V2,...,Vn)
  - s.t., $V1 \in D1$, $V2 \in D2$,...,$Vn \in Dn$
- Relation-subset of cartesian product
  - of one or more domains
  - FINITE only; empty set allowed
- Tuples = members of a relation inst.
- Arity (degree) = number of domains
- Components = values in a tuple
- Domains — corresp. with attributes
- Cardinality = number of tuples

Relation Instance

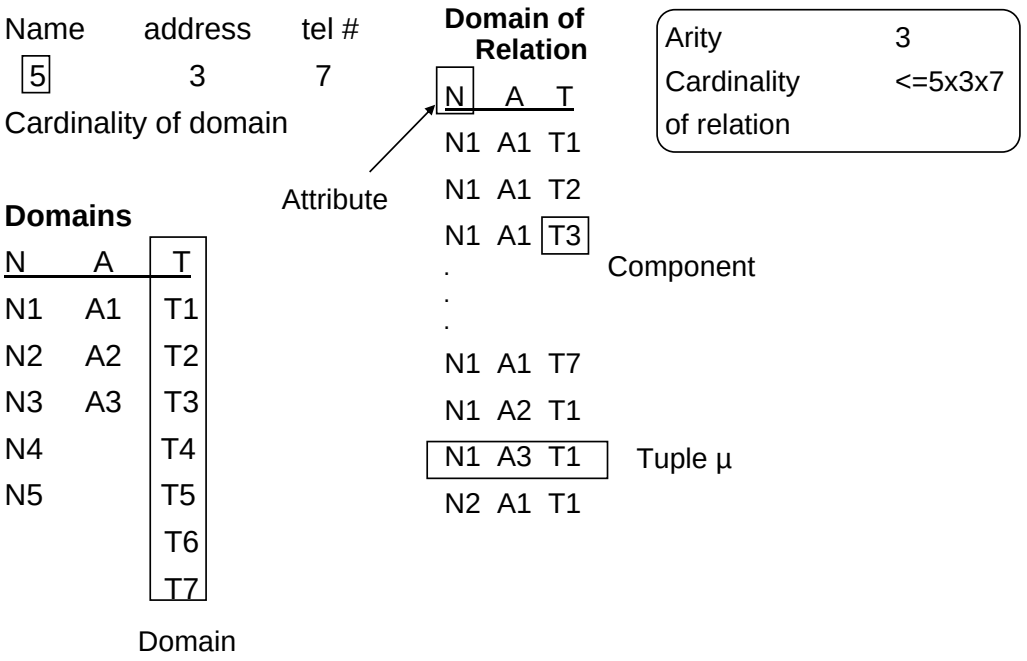
- Relation is a set of tuples
  - Tuple ordering immaterial
  - No duplicates
  - Cardinality of relation = number of tuples
- All tuples in a relation have the same structure; constructed from the same set of attributes
  - Attributes named (ordering immaterial)
  - Value of an attribute drawn from the attribute's domain
  - Arity (Degree) = number of attributes

Relation Instance (Example)

Id	Name	Address	Status
1111111	John	123 main	freshman
2345678	Mary	456 cedar	sophmore
4433322	Art	77 so. 3rd	senior
7654321	Pat	88 no. 4th	sophmore

Student

Relation: Example



Relation Schema

- Relation name
- Attribute names and domains
- Integrity constraints - e.g.,:
  - The values of a particular attribute in all tuples are unique
  - The values of a particular attribute in all tuples are greater than 0
- Default values

Relational Database

- **Finite** set of relations
- Each relation consists of a schema and an instance
- Database schema = set of relation schemas (and other things)
- Database instance = set of (corresponding) relation instances

Integrity Constraints

- Part of schema
- Restriction on state (or sequence of states) of data base
- Enforced by DBMS
- Intra-relational - involve only one relation
  - Part of relation schema
  - e.g., all Ids are unique
- Inter-relational - involve several relations
  - Part of relation schema or database schema

Types of Integrity Constraints

- Static - limitation on state of database
  - Syntactic (structural)
    - e.g., all values in a column must be unique for Id
  - Semantic (involve meaning of attributes)
    - e.g., cannot register for more than 9 credits
- Dynamic - limitation on sequence of database states (supported by some DBMSs, but not in original SQL standard)
    - e.g., cannot raise salary by more than 5%

Integrity Constraints

- Constraints: ***conditions*** that must hold on ***all*** valid relation instances.
- Tree main constrains:
  - **Key** constraints (single relation)
  - **entity integrity** constraints (single relation)
  - **referential integrity** constraints (two relations)

Key Constrains

- **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 t1 and t2 in r(R),
    t1[SK] <> t2[SK].
- **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:

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

- Two **candidate keys**:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
  - Are Key1 and Key2 superkeys?
- {SerialNo, Make}, key or superkey?
- **Primary key**: If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are *underlined*.

Entity Integrity

- **Relational Database Schema**: A set S of relation schemas that belong to the same database.
  - S is the *name* of the **database**.

    S = {R1, R2, ..., Rn}
- **Entity Integrity:**The ***primary key*** *attributes* PK of each relation schema R in S **cannot** have **null** values in any tuple of r(R).
  - Why?

Entity Integrity

- **Relational Database Schema**: A set S of relation schemas that belong to the same database.
  - S is the *name* of the **database**.

    S = {R1, R2, ..., Rn}
- **Entity Integrity:**The ***primary key*** *attributes* PK of each relation schema R in S **cannot** have **null** values in any tuple of r(R).
  - primary key values are used to *identify* the individual tuples.
    - t[PK] <> null for any tuple t in r(R)

Entity Integrity

- Other attributes (Non PK) of R may be similarly constrained to disallow null values
- E.g., Name, Address

Referential Integrity
(Foreign Key Constrains)

- A constraint involving *two* relations
- specify a *relationship* among tuples in two relations:
  - the **referencing relation(R1)** and the **referenced relation (R2)**
  - **FK(foreign key** attributes) of R1 reference **PK** of R2
- displayed in a relational database schema as a directed arc from R1.FK to R2.PK

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

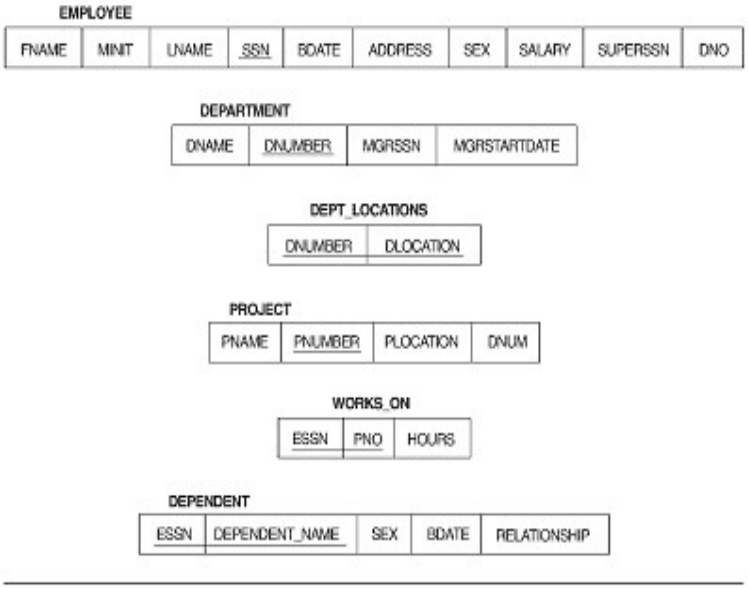


Figure 7.6 One possible relational database state corresponding to the COMPANY schema.

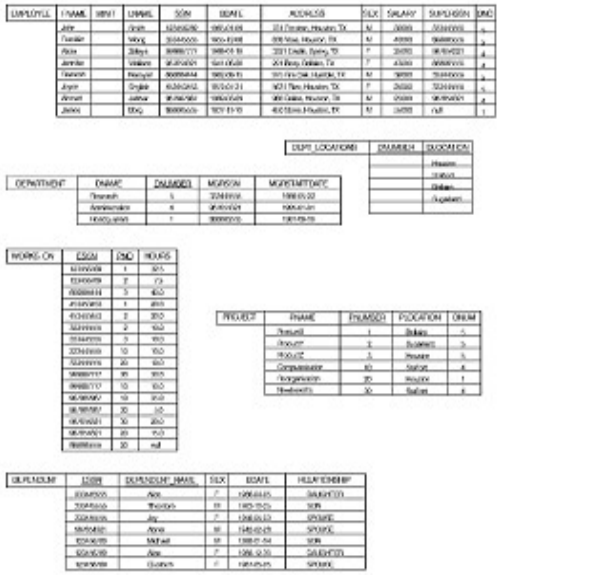
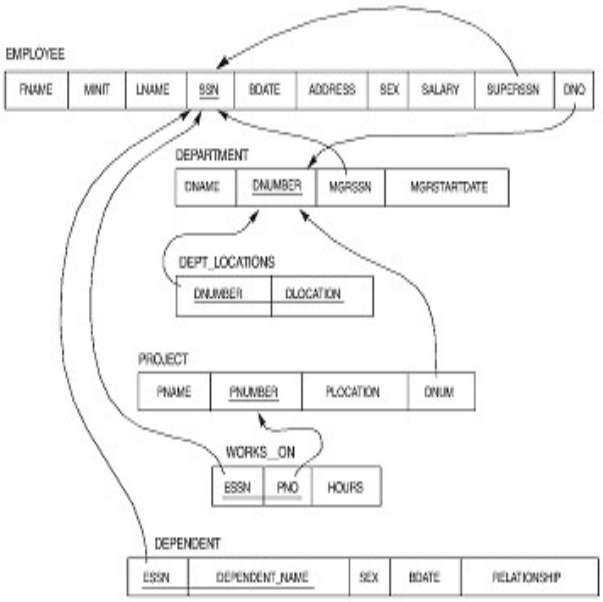


Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



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

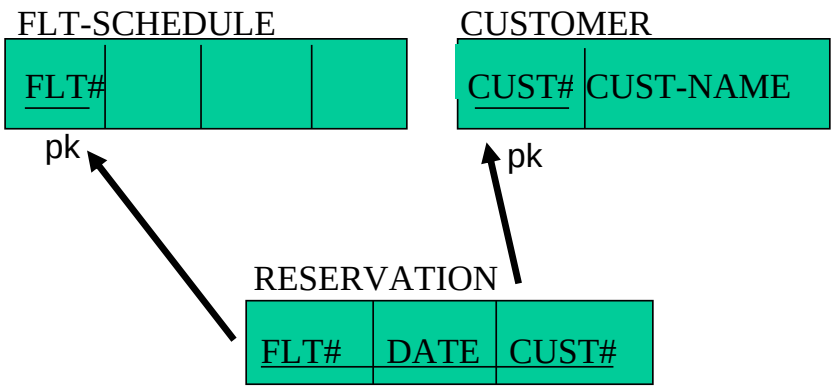
plane#	plane-type	total-#seats

CUSTOMER

cust#	first	middle	last	phone#	street	city	state	zip

RESERVATION

flt#	date	cust#	seat#	check-in-status	ticket#



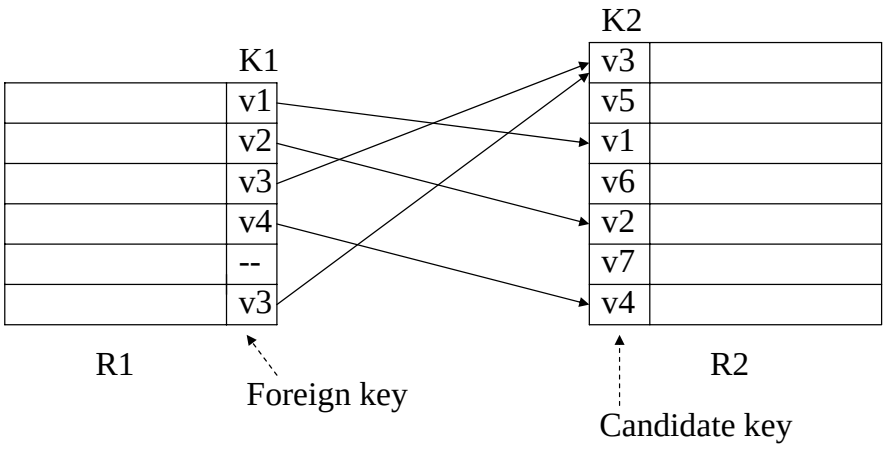
Database Schema (Example)

- Student (Id: INT, Name: STRING, Address: STRING, Status: STRING)
- Professor (Id: INT, Name: STRING, DeptId: DEPTS)
- Course (DeptId: DEPTS, CrsName: STRING, CrsCode: COURSES)
- Transcript (CrsCode: COURSES, StudId: INT, Grade: GRADES, Semester: SEMESTERS)
- Department(DeptId: DEPTS, Name: STRING)

Foreign Key Constraint(Cont.)

- **Referential integrity** - attribute named in one relation must correspond to tuple(s) in another that describes the item
  - Transcript (CrsCode) references Course(CrsCode)
  - Professor(DeptId) references Department(DeptId)
- K1 is a **foreign key** of R1 referring to K2 in R2
  - if v is a value of K1, there is a *unique* tuple of R2 in which K2 has value v
  - This is a special case of referential integrity: K2 must be a candidate key of R2 (CrsCode is a key of Course)
  - If no row exists in R2 -- violation of referential integrity
  - Not all rows of R2 need to be referenced. Relationship is not symmetric (some course might not be taught)
  - Value of a foreign key might not be specified (DeptId column of some professor might be null)

Foreign Key Constraint (Example)



Foreign Key (con't)

- Names of K1 and K2 need not be the same.
  - With tables:

    Teaching(CrsCode: COURSES, Sem: SEMESTERS, ProfId: INT)
    Professor(Id: INT, Name: STRING, DeptId: DEPTS)

  ProfId attribute of Teaching references Id attribute of Professor
- R1 and R2 need not be distinct.
  - Employee(Id:INT, MgrId:INT, ….)
    - Employee(MgrId) references Employee(Id)
  - Every manager is also an employee and hence has a unique row in Employee

Foreign Key (con't)

- Foreign key might consist of several columns
  - (CrsCode, Semester) of Transcript references (CrsCode, Sem) of Teaching
- R1(A1, …An) references R2(B1, …Bn)
  - There exists a 1:1 relationship between A1,…An and B1,…Bn
  - Ai and Bi have same domains (although not necessarily the same names)
  - B1,…Bn is a candidate key of R2