CO226: Database Systems

The Relational Data Model

Sampath Deegalla
dsdeegalla@pdn.ac.lk

15th July 2014

Relational Model Concepts

- The relational Model of data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.
- 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.

Relational Model Concepts

- The relational Model of data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.
- 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.

Relational Model Concepts

- The relational Model of data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.
- 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.

Relational Model Concepts

- The relational Model of data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.
- 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.

Relational Model Concepts

- The relational Model of data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.
- 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.

INFORMAL DEFINITIONS

- RELATION: A table of values
  - A relation may be thought of as a set of rows (or set of columns).
  - Each row represents a fact that corresponds to a real-world entity or relationship.
  - Each row has a value of an item or set of items that uniquely identifies that row in the table.
  - Each column typically is called by its column name or column header or attribute name.

FORMAL DEFINITIONS

- The Schema of a Relation: $R\,(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- The Schema of a Relation: $R\left(A_1, A_2, \ldots, A_n\right)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- The Schema of a Relation: $R\,(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- The Schema of a Relation: $R\,(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- The Schema of a Relation: $R\,(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- The Schema of a Relation: $R\,(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- The Schema of a Relation: $R\,(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over attributes $A_1, A_2, \ldots, A_n$
- Each attribute A_i is the name of a role played by some domain D in R
- Domain D is denoted by dom(A_i)
- The degree of a relation is the number of attributes of R
- CUSTOMER (CustID, CustName, Address, Phone)

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four 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).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A domain has a logical definition:
  - e.g. "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain may have 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.
  - E.g., Dates have various formats such as month name, date, year or yyyy-mm-dd, or dd mm,yyyy etc.

FORMAL DEFINITIONS

- A domain has a logical definition:
  - e.g. “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain may have 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.
  - E.g., Dates have various formats such as month name, date, year or yyyy-mm-dd, or dd mm,yyyy etc.

FORMAL DEFINITIONS

- A domain has a logical definition:
  - e.g. "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain may have 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.
  - E.g., Dates have various formats such as month name, date, year or yyyy-mm-dd, or dd mm,yyyy etc.

FORMAL DEFINITIONS

- A domain has a logical definition:
    - e.g. "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain may have 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.
    - E.g., Dates have various formats such as month name, date, year or yyyy-mm-dd, or dd mm,yyyy etc.

FORMAL DEFINITIONS

- A domain has a logical definition:
  - e.g. “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain may have 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.
  - E.g., Dates have various formats such as month name, date, year or yyyy-mm-dd, or dd mm,yyyy etc.

FORMAL DEFINITIONS

- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally, Given $R(A_1, A_2, \ldots, A_n)$
  $r(R) \subseteq (\mathrm{dom}(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$
- R: schema of the relation
- r of R: a specific "value" or population of R.

FORMAL DEFINITIONS

- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally, Given $R\left(A_1, A_2, \ldots, A_n\right)$
  $r(R) \subseteq (\mathrm{dom}(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$
- R: schema of the relation
- r of R: a specific "value" or population of R.

FORMAL DEFINITIONS

- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally, Given $R(A_1, A_2, \ldots, A_n)$
  $r(R) \subseteq (\mathrm{dom}(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$
- R: schema of the relation
- r of R: a specific "value" or population of R.

FORMAL DEFINITIONS

- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally, Given $R\,(A_1, A_2, \ldots, A_n)$
  $r(R) \subseteq (\mathrm{dom}(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$
- R: schema of the relation
- r of R: a specific "value" or population of R.

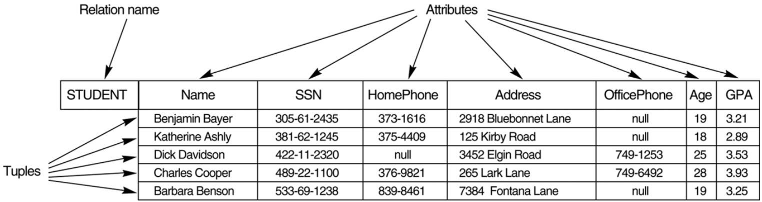
FORMAL DEFINITIONS

- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally, Given $R\,(A_1, A_2, \ldots, A_n)$
  $r(R) \subseteq (\mathrm{dom}(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$
- R: schema of the relation
- r of R: a specific "value" or population of R.

DEFINITION SUMMARY

Informal Terms	**Formal Terms**
Table	Relation
Column	Attribute/Domain
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

CHARACTERISTICS OF RELATIONS

- Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, \ldots, A_n)$ and the values in $t =< v_1, v_2, \ldots, v_n >$ to be ordered.
- Values in a tuple : All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.

CHARACTERISTICS OF RELATIONS

- Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, \ldots, A_n)$ and the values in $t =< v_1, v_2, \ldots, v_n >$ to be ordered.
- Values in a tuple : All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.

CHARACTERISTICS OF RELATIONS

- Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, \ldots, A_n)$ and the values in $t =< v_1, v_2, \ldots, v_n >$ to be ordered.
- Values in a tuple : All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.

CHARACTERISTICS OF RELATIONS

- Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple): We will consider the attributes in $R\,(A_1, A_2, \ldots, A_n)$ and the values in $t =< v_1, v_2, \ldots, v_n >$ to be ordered.
- Values in a tuple : All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.

CHARACTERISTICS OF RELATIONS

- Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, \ldots, A_n)$ and the values in $t = < v_1, v_2, \ldots, v_n >$ to be ordered.
- Values in a tuple : All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.

CHARACTERISTICS OF RELATIONS

- Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, \ldots, A_n)$ and the values in $t =< v_1, v_2, \ldots, v_n >$ to be ordered.
- Values in a tuple : All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.

CHARACTERISTICS OF RELATIONS

Notation:
We refer to component values of a tuple t by $t[A_i] = v_i$ (the value of attribute A_i for tuple t).
Similarly, $t[A_u, A_v, \ldots, A_w]$ refers to the subtuple of t containing the values of attributes $A_u, A_v, \ldots, A_w$, respectively.

Relational Model Notation

- Relation schema R of degree n
    - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
    - R.A
    - e.g. STUDENT(Name, Ssn, ...)
    - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Relation schema R of degree n
  - $R(A_1, A_2, \ldots, A_n)$
- Q,R,S for relation names
- q,r,s for relation states
- t,u,v for tuples and tuples in a relation as $t_1, t_2, \ldots, t_m$
- Attribute A
  - R.A
  - e.g. STUDENT(Name, Ssn, ...)
  - Attributes STUDENT.Name, STUDENT.Ssn, ...

Relational Model Notation

- Tuple t in a relation r(R)
    - $t =< v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
    - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
        - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
        - tuple for 'Barbara': $t =<' BarbaraBenson',' 533 - 69 - 1238',' 839 - 8461',' 7384 FontanaLane', NULL, 19, 3.25 >$
        - $t[Name] =<' BarbaraBenson' >$
        - $t[Ssn, Gpa, Age] =<' 533 - 69 - 1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t =< v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t[A_i] and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t =<' BarbaraBenson',' 533 - 69 - 1238',' 839 - 8461',' 7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] =<' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] =<' 533 - 69 - 1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t = < v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t = <' BarbaraBenson', '533 - 69 - 1238', '839 - 8461', '7384 FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] = <' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] = <' 533 - 69 - 1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t = < v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t =<' BarbaraBenson', '533-69-1238', '839-8461', '7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] =<' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] =<' 533-69-1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t =< v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t[A_i] and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t =<' BarbaraBenson','533-69-1238','839-8461','7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] =<' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] =<' 533-69-1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t = < v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t = <' BarbaraBenson',' 533 - 69 - 1238',' 839 - 8461',' 7384 FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] = <' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] = <' 533 - 69 - 1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t =< v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t[A_i] and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t =<' BarbaraBenson',' 533-69-1238',' 839-8461',' 7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] =<' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] =<' 533-69-1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t = < v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t = <' BarbaraBenson',' 533-69-1238',' 839-8461',' 7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] = <' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] = <' 533-69-1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t =< v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t =<' BarbaraBenson',' 533-69-1238',' 839-8461',' 7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] =<' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] =<' 533-69-1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t = < v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t = <' BarbaraBenson',' 533 - 69 - 1238',' 839 - 8461',' 7384 FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] = <' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] = <' 533 - 69 - 1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t =< v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t =<' BarbaraBenson',' 533-69-1238',' 839-8461',' 7384 FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] =<' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] =<' 533-69-1238', 19, 3.25 >$

Relational Model Notation

- Tuple t in a relation r(R)
  - $t = < v_1, v_2, \ldots, v_n >$ where v_i is the value correspoinding to attribute A_i
  - Both t$[A_i]$ and $t.A_i$ referes to v_i in t for attribute A_i
    - e.g. Student relation: STUDENT(Name, Ssn, Homephone, Address, Officephone, Age, Gpa)
    - tuple for 'Barbara': $t = <' BarbaraBenson',' 533 - 69 - 1238',' 839 - 8461',' 7384FontanaLane', NULL, 19, 3.25 >$
    - $t[Name] = <' BarbaraBenson' >$
    - $t[Ssn, Gpa, Age] = <' 533 - 69 - 1238', 19, 3.25 >$

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation instances. There are three main types of constraints:
  1. Key constraints
  2. Entity integrity constraints
  3. Referential integrity constraints

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A “minimal” superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.
  Example: The CAR relation schema:
  `CAR(State, Reg#, SerialNo, Make, Model, Year)`
  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:
  `CAR(State, Reg#, SerialNo, Make, Model, Year)`
  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A “minimal” superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- Superkey of R : 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. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- Key of R : A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

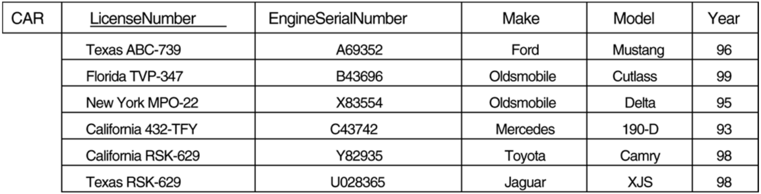
  Example: The CAR relation schema:

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

  has two keys Key1 = {`State, Reg#`}, Key2 = {`SerialNo`}, which are also superkeys. {`SerialNo, Make`} is a super key but not a key.

Key Constraints

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key. The primary key attributes are underlined.



CAR	LicenseNumber	EngineSerialNumber	Make	Model	Year
	Texas ABC-739	A69352	Ford	Mustang	96
	Florida TVP-347	B43696	Oldsmobile	Cutlass	99
	New York MPO-22	X83554	Oldsmobile	Delta	95
	California 432-TFY	C43742	Mercedes	190-D	93
	California RSK-629	Y82935	Toyota	Camry	98
	Texas RSK-629	U028365	Jaguar	XJS	98

- The CAR relation, with two candidate keys: LicenseNumber and EngineSerialNumber

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Entity Integrity

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

$$S = \{R_1, R_2, \ldots, R_n\}$$

- 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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note:Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- A constraint involving <u>two</u> relations
- Used to specify a <u>relationship</u> among tuples in two relations : the referencing relation and the referenced relation.
- Tuples in the <u>referencing relation</u> R_1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the <u>referenced relation</u> R_2. A tuple t_1 in R_1 is said to reference a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a <u>directed arc</u> from $R_1.FK$ to R_2.

Referential Integrity

- A constraint involving two relations
- Used to specify a relationship among tuples in two relations : the referencing relation and the referenced relation.
- Tuples in the referencing relation R_1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R_2. A tuple t_1 in R_1 is said to reference a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.FK$ to R_2.

Referential Integrity

- A constraint involving two relations
- Used to specify a relationship among tuples in two relations : the referencing relation and the referenced relation.
- Tuples in the referencing relation R_1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R_2. A tuple t_1 in R_1 is said to reference a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.FK$ to R_2.

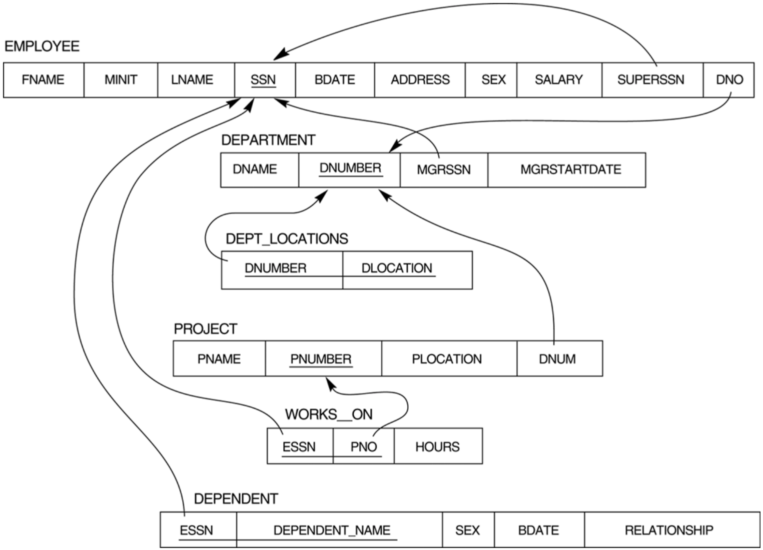
Referential Integrity

- A constraint involving two relations
- Used to specify a relationship among tuples in two relations : the referencing relation and the referenced relation.
- Tuples in the referencing relation R_1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R_2. A tuple t_1 in R_1 is said to reference a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.FK$ to R_2.

Referential Integrity

- A constraint involving two relations
- Used to specify a relationship among tuples in two relations : the referencing relation and the referenced relation.
- Tuples in the referencing relation R_1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R_2. A tuple t_1 in R_1 is said to reference a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.FK$ to R_2.

Referential integrity constraints



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 Constraint

The value in the foreign key column(s) FK of the the referencing relation R_1 can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the referenced relation R_2, or
2. a null.

In case (2), the *FK* in R_1 should not be a part of its own primary key.

Referential Integrity Constraint

The value in the foreign key column(s) FK of the the referencing relation R_1 can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the referenced relation R_2, or
2. a null.

In case (2), the *FK* in R_1 should not be a part of its own primary key.

Referential Integrity Constraint

The value in the foreign key column(s) FK of the the referencing relation R_1 can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the referenced relation R_2, or
2. a null.

In case (2), the *FK* in R_1 should not be a part of its own primary key.

Referential Integrity Constraint

The value in the foreign key column(s) FK of the the referencing relation R_1 can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the referenced relation R_2, or
2. a null.

In case (2), the FK in R_1 should not be a part of its own primary key.

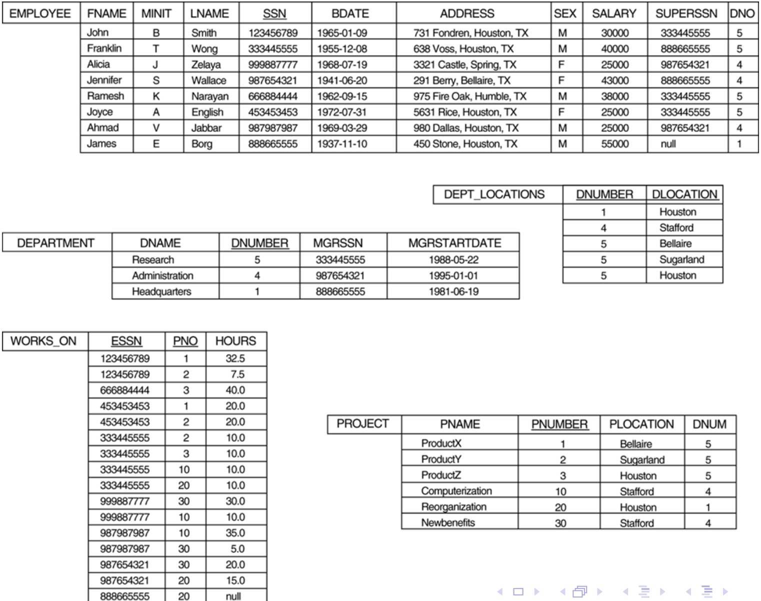
Referential Integrity Constraint

The value in the foreign key column(s) FK of the the referencing relation R_1 can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the referenced relation R_2, or
2. a null.

In case (2), the *FK* in R_1 should not be a part of its own primary key.

One possible database state for the COMPANY relational database schema



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	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	1968-07-19	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

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

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

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
  - The salary of an employee should not exceed the salary of the employee's supervisor
  - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
    - The salary of an employee should not exceed the salary of the employee's supervisor
    - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
  - The salary of an employee should not exceed the salary of the employee's supervisor
  - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
  - The salary of an employee should not exceed the salary of the employee's supervisor
  - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
  - The salary of an employee should not exceed the salary of the employee's supervisor
  - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
  - The salary of an employee should not exceed the salary of the employee's supervisor
  - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
  - The salary of an employee should not exceed the salary of the employee's supervisor
  - 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
- Triggers and assertions can be used to allow for some of these

Other Types of Constraints

- Integrity constraints are defined as part of relational database schema
- General general constraints (sometimes called as semantic integrity constraints) are based on application semantics and cannot be expressed by the data model
- Examples
    - The salary of an employee should not exceed the salary of the employee's supervisor
    - 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
- Triggers and assertions can be used to allow for some of these