COS 597A:
Principles of
Database and Information Systems

Relational model

Relational Database Definitions

1. A relation is a set of tuples over specified domains
   - R subset of D_1 X D_2 X D_3 X … D_k (k-ary)
   - Each D_i is a declared domain
   - Domains atomic
     - *types* of programming languages
2. A relational database is a set of relations and possibly constraints among the relations

Relational model

- A formal (mathematical) model to represent
  - objects (data/information),
  - relationships between objects
  - Constraints on objects and relationships
  - Queries about information
- Well-founded on mathematical principles :
  - Precise semantics of constraints and queries
  - Can prove equivalence of different ways to express queries

Relational Database: Terminology

Schema for a relation:
1. Relation name
2. Domain (type) of each component
   i.e. declare D_i s

Equivalent:
- Instance of a scheme
- Table

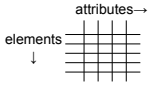
Term "relation" is used to refer to a schema and a particular instance – disambiguate by context

Relational Database: More Terminology

Each D_i of a schema is referred to as a component or attribute or field or column of the schema

Each d_i of a tuple = (d_1, d_2, d_3, … d_k) is referred to as component or attribute or field of the tuple

Each tuple of a relation is also referred to as an element or row of the relation



Example

books: (title, ISBN#, edition, date)

publishers: (name, country, address)

authors:
(name, gender, birth date, place of birth, date of death)

Need declare domains:
e.g. title: string

Constraints

Identifying elements

Key: a minimal set of attributes whose values uniquely identify each element in a relation

Candidate Key: any key

Primary key: a candidate key defined to be primary by person who defines relation

Superkey: any set of attributes that contains a candidate key

Denote primary key by underlining attributes

books: (title, ISBN#, edition, date)

publishers: (name, country, address)

authors:
(name, gender, birth date, place of birth, date of death)

Need declare domains:
e.g. title: string

Constraints on elements

- Declaring a candidate key constrains values of attributes
- Example: ISBN# as key
  - No book without an ISBN#
  - No two books with same ISBN#

But there are relationships between authors, books and publishers

How represent?

Our Example

books are published by publishers:

published by: (*ISBN#*, *publisher_name*, in print)

books are written by authors:

written by:
(*ISBN#*, *author_name*, *birth date*, *place of birth*)

Alternative

• If each book must have exactly one publisher, then:

published by: (*isbn#, publisher_name*, in print)

• Instead put info from **published_by** in **books**:

books:
(title, isbn#, edition, date, publisher_name, in print)

Null values

What if some books in relation **books** not published?

- Want no entry in *publisher_name* and *in print*
- Add value null to domain to represent.
- Attributes of candidate keys cannot have null values.

Foreign keys

- ISBN# in **books** is related to the ISBN# in **written by** and **published by**
  - a specific ISBN# value in one relation refers to the same book as the ISBN# in the other relation
- name, birth date, place of birth in **authors** is related to author_name, birth date, place of birth in **published by**

How represent?

Foreign key constraint

- Specify that a set of attributes in schema for one relation form a primary key for a specific other relation
  - "other relation" is referred to or referenced by first relation

R1: (attrib1, attrib2, attrib3, attrib4, attrib5)

R1 refers to/references R2

R2: (attrib1, attrib2, attrib3, attrib4)

Foreign Keys for Our Example

published by: (*isbn#, publisher_name*, in print)
isbn# is a foreign key referencing *books*
Primary key of *books* understood
Publisher_name is a foreign key referencing *publishers.name*

written by:
(*isbn#, author_name, birth date, place of birth*)
isbn# is a foreign key referencing *books;*
(author_name, birth date, place of birth) is a foreign key referencing *authors*

Enforcing relational constraints

- Constraints must be satisfied at all times
- What happens when tuples in relations change?
- Action of changing a relation not part of basic relational model
- Database language implementing model enforces

Enforcement in SQL

SQL commands changing relations:
INSERT, DELETE, UPDATE

- Domain constraints
  - Don't allow attribute value not in domain
    INSERT or UPDATE fails
- "Not null" constraints
  - Special case of domain constraints

Enforcement in SQL

- Candidate key constraints
  - Can have other candidate keys declared as well as primary key
  - Don't allow 2nd tuple with same key value
    INSERT or UPDATE fails
  - Implicit "not null" for attributes in a key
    INSERT or UPDATE fails

Enforcement in SQL

- Foreign key constraints

Suppose Y denotes a set of attributes of relation B that reference the primary key of relation A.

- Don't allow tuple into B if no tuple in A with matching values for Y
  INSERT or UPDATE fails

Enforcement in SQL

Foreign key constraints continued
- suppose want to remove a tuple in A
- Suppose there is a tuple in B with matching values for Y

Choices (in SQL):
1. Disallow deletion from A
   DELETE or UPDATE fails

Enforcement in SQL

Choices (in SQL) continued:
2. Ripple effect (CASCADE):
   - Remove tuple from A and all tuples from B with matching values for Y
   - DELETE or UPDATE in A causes DELETE in B
3. Substitute value
   - Put "null" (if Y not part of candidate key for B) or other default value for Y in B
   - DELETE or UPDATE in A causes UPDATE in B

Example?

Books: (title, ISBN#, edition, date)

PU branches: (br_name, librarian, hours)

Copies: (ISBN#, copy#, condition, br_name)
br_name not null
isbn# is a foreign key referencing ***books***
br_name is a foreign key referencing ***PU branches***

Other Constraints of Interest

- Domain attribute constraints
  - Need to test values of attributes not simply membership properties in sets
  - Example:
    Attribute *NJ driver*: yes/no flag
    Attribute *age*: number
    Constraint "if age <17 then *NJ driver* == "no"

Other Constraints of Interest, cont.

- Functional constraints

Example:

relation **person** with 6 attributes:
first name, last name, street address, state, area code, 7-digit phone number.

Constraint:

if *area code* of person 1 = *area code* of person 2
then *state* of person 1 = *state* of person 2

Equivalently, *area code* determines *state*

Functional Constraints

General form:

Let A and B be subsets of attributes for a relation
For any tuples e_j and e_k of the relation:

If the values of attributes in set A for tuple e_j equal the values of attributes in set A for tuple e_k

Then the values of attributes in set B for tuple e_j equal the values of attributes in set B for tuple e_k

Functional Constraints Example

More complicated example:

customer relation with 8 attributes:
height, weight, arm length, leg length, jacket size, pant size, shirt size, color preference

Constraints:
Height, weight, arm length determine *shirt size*
Height, weight, leg length determine *pant size*

Enforcing Other Constraints

- Value-based constraints?
- General functional constraints?

In relational model:

- Not expressed in formal relational model
- Declaring and enforcing these depend on use of database language
- Use query semantics to check