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 model - practice

- Foundation of most Database Management Systems
- SQL language is a programming language to express constructs of formal model

Relational Database Definitions

1. A relation is a set of tuples over specified domains
   - $R$ subset of $D_1 \times D_2 \times D_3 \times \ldots \times D_k$ (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 Database: Terminology

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

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 of the schema

Each $d_i$ of a tuple $=(d_1, d_2, d_3, \ldots, d_k)$ 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

```
<table>
<thead>
<tr>
<th>elements</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>attribute 1</td>
</tr>
<tr>
<td>b</td>
<td>attribute 2</td>
</tr>
<tr>
<td>c</td>
<td>attribute 3</td>
</tr>
</tbody>
</table>
```
Translating ER model to relational

- Domains $\rightarrow$ domains
- Entity $\rightarrow$ relation
- Relationship $\rightarrow$ one* or more relations
  - come back to
- Constraints $\rightarrow$ constraints BUT
  - Not all ER constraints expressible in basic relational model

Relational model is FLAT – no hierarchy!

Our ER Example $\rightarrow$ Relational schema

For entities, get relations:
- books: (title, ISBN#, edition, date)
- authors: (name, gender, birth date, place of birth, date of death)
- publishers: (name, country, address)

Need declare domains:
  - e.g. title: string
  - Same def: candidate keys, primary key, superkeys

Our ER Example $\rightarrow$ Relational schema

For relationships:
- ER published by: (books, publishers, in print)
  - becomes
  - published by: (ISBN#, publisher_name, in print)
  - key constraint on entity books in relationship published by $\rightarrow$
    - A book has at most one publisher
- ER written by: (books, authors)
  - becomes
  - written by: (ISBN#, author_name, birth date, place of birth)

Our ER Example $\rightarrow$ Relational schema

Because ER key constraint on entity books in relationship published by
- Can fold relation published by into relation books:
  - books: (title, ISBN#, edition, date, pub_name, in print)

What if some books not published?
  - i.e. entity books not totally participate in relationship published by

Our ER Example $\rightarrow$ Relational schema

- books:
  - (title, ISBN#, edition, date, pub_name, in print)

What if some books not published?
  - i.e. entity books not totally participate in relationship published by

Must allow values of attributes
  - pub_name and in print to be null

Translating ER model to relational

General conclusion:
- Relationship $\rightarrow$ one zero or more relations
Translating ER model to relational

- Get flat set of relations
- But relations are interrelated
  - Bring together primary keys of different relations to build new relation
  - Captures ER relationship
- How capture this in relational model?
  
Foreign key constraints

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)
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

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

Summary of board example: with Copies as weak entity

Relational model:
PU branches: (br_name, librarian, hours)
Copies: (ISBN#, call#, copy#, condition, br_name)
  (call#, copy#) is a foreign key referencing Copies
  br_name is a foreign key referencing PU branches

Summary of board example: with Copies as strong entity

Relational model:
PU branches: (br_name, librarian, hours)
Copies: (ISBN#, call#, copy#, condition)
  ISBN# is a foreign key referencing Books
  call# is a foreign key referencing Copies
  br_name is a foreign key referencing PU branches

Summary of board example: Alternative with Copies as strong entity

Relational model:
PU branches: (br_name, librarian, hours)
Copies: (ISBN#, call#, copy#, condition)
  ISBN# is a foreign key referencing Books
  call# is a foreign key referencing Copies
  br_name is a foreign key referencing PU branches

NEW
Board example: Total participation of Copies?

Copies: (ISBN#, call #, copy #, condition, br_name)

br_name not null
isbn# not null

isbn# is a foreign key referencing Books
br_name is a foreign key referencing PU branches

capture total participation in PU book and PU holding because PU book, PU holding represented within Copies versus PU holding (call #, copy #) is a foreign key referencing Copies
br_name is a foreign key referencing PU branches

br_name "not null" would not capture that every (call #, copy #) value is in a PU holding pair

Board example: Total participation of PU branches?

Can't get constraint applied to all PU branch tuples without being part of PU branch relation

⇒ Total participation of PU branches in PU holding not representable in pure relational definition

Basic Paradigm

• Each entity becomes a relation
• Relationship becomes
  \[ R: \{ \text{list of attributes forming key of Entity 1 (denote } L_1), \text{list of attributes forming key of Entity 2 (denote } L_2), \ldots, \text{list of attributes forming key of Entity } k, \text{Attribute } R_{A_1}, \ldots, \text{Attribute } R_{A_m}, L_1 \text{ is a foreign key referencing Entity 1, } \ldots, L_k \text{ is a foreign key referencing Entity } k \} \]

What about constraints on relationships?

• Key constraint:
  – Simplifies key of corresponding relation
  – Allows folding of relation into key entity

• Total participation constraint:
  – In general, cannot represent in purely relational definition:
    • Domain specification
    • Keys of relations
    • Foreign keys
    • "not null"s

Note primary key

For Basic Paradigm (binary relationship)

• Each entity becomes a relation with same attributes
• Relationship becomes
  \[ R: \{ \text{list of attributes, } (a_1, \ldots, a_p, b_1, \ldots, b_q, r_1, \ldots, r_m} \}
  \text{Note primary key}
  \text{constraint on } a_i \text{ and } b_j \text{ in tuple of } R \text{ make up } R \text{’s primary key} \]

Clarifying null values and foreign keys
When one entity (e.g. Entity A) has key constraint and fold R into it

- Entity B becomes a relation with same attributes
- Relationship R becomes part of relation for Entity A:
  \[ A: \{(a_1, \ldots, a_p, x_1, \ldots, x_u, b_1, \ldots, b_q, r_1, \ldots, r_m)\} \]

  \( (b_1, \ldots, b_q) \) is a foreign key referencing B

  now need to allow null values for \( b_1, \ldots, b_q \) in A
  not every entity in A is related to an entity in B

When have key constraint and total participation and fold R in

- Entity B becomes a relation with same attributes
- Relationship R becomes part of relation for Entity A:
  \[ A: \{(a_1, \ldots, a_p, x_1, \ldots, x_u, b_1, \ldots, b_q, r_1, \ldots, r_m)\} \]

  \( (b_1, \ldots, b_q) \) is a foreign key referencing B

  now prohibit null values for \( b_1, \ldots, b_q \) in A
  every entity in A is related to an entity in B

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

- 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

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

Actions for board example?

PU branches: (br_name, librarian, hours)
Copies: (ISBN#, copy#, condition, purchase date, br_name)
   br_name not null
   ISBN# is a foreign key referencing books
   br_name is a foreign key referencing PU branches

Questions:

What about constraints not expressible in ER model?

• Value-based constraints?
• General functional constraints?

In relational model:
• Declaring and enforcing these depend on use of database language
• Use query semantics to check