Relational Concepts

Anne Denton

Department of Computer Science
North Dakota State University

Outline

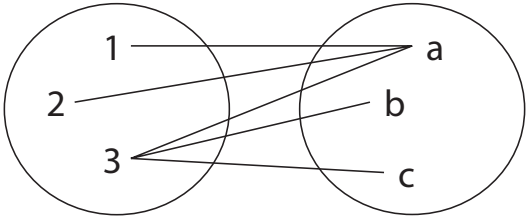
Table of Contents

Brainstorming Question

Do you remember hearing about “relations" in mathematics classes in grade school? It is common that they are on the middle school curriculum — but not in the form in which we will discuss them here.

Relations in Mathematics

- Relations are generalizations of sets
  - Relations relate elements of sets
- Formally, relations are sets of *n*-tuples
  - An *n*-tuple is a sequence of *n* elements that relate *n* set elements

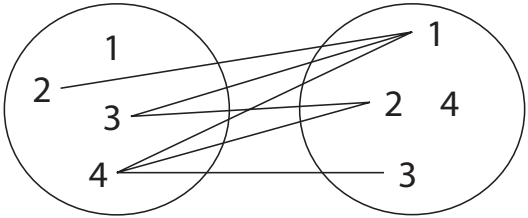


{1,a}
{2,a}
{3,a}
{3,b}
{3,c}

1	a
2	a
3	a
3	b
3	c

Examples

- Examples of relations discussed in grade school
  - “Greater Than"
  - “Equal To"
- See below the “Greater Than" relation over sets {1,2,3,4}



{2,1}
{3,1}
{3,2}
{4,1}
{4,2}
{4,3}

2	1
3	1
3	2
4	1
4	2
4	3

Question 1

How many elements does the "Equal To" relation have over two sets with elements {1,2,3}

1. 1
2. 3
3. 6
4. 9

Table of Contents

From Tables to Relations

- Relations are like tables
  - Rows are records
  - Columns are attributes
- Ordering doesn't matter
  - Relations are sets
  - Records identified by keys, see "constraints"
- Attributes must be simple
  - e.g., don't list multiple telephone numbers in one field!
- Power comes from
  - Queries that combine tables
  - Constraints

Question 2

Is the following representation consistent with expectations for relational databases?

faculty_id	first_name	last_name	courses
248	Anne	Denton	366,765

1. Yes
2. No

Question 3

Is the following representation consistent with expectations for relational databases?

faculty_id	first_name	last_name	course_id
248	Anne	Denton	366
248	Anne	Denton	765

1. Yes, this looks well-designed
2. It solves the problem of multiple elements per field, but seems to create other problems
3. No, it doesn't solve the problem of multiple field elements

Question 4

Is the following representation consistent with expectations for relational databases?

faculty_id	first_name	last_name
248	Anne	Denton

faculty_id	course
248	366
248	765

1. Yes
2. No

Queries and Constraints

faculty_id	first_name	last_name
248	Anne	Denton

faculty_id	course_id
248	366
248	765

- Querying
  - Even reconstructing original table requires “join" query
- Contraints
  - The second table should only contain faculty that exist in the first table

Table of Contents

Simplicity is the Challenge

- What makes relational databases special?
  - Table design not limited to relational databases (e.g. even spreadsheets are like tables)
  - Relational databases represent complex miniworlds with nothing but simple tables
  - Depends on fast join queries!
  - Constraints keep relations consistent with each other
- How can we design tables systematically?
  - Process of breaking tables into simple ones is called “Normalization" (discussed in 2nd half of course)
  - Design processes could have told us that “Faculty" and “Teaching Assignments" are different concepts that should be in different tables (discussed in 1st half of course)

Question 5 (Multiple answers can be correct)

What makes relational databases special in comparison with non-relational ones? Which of the following statements are valid?

1. Data are organized as one big table
2. Only simple elements allowed in tables
3. Fast evaluation of queries that combine tables
4. Importance of constraints for keeping multiple tables consistent

Join Queries

Combine

faculty_id	**first_name**	**last_name**
248	Anne	Denton

and

faculty_id	**course_id**
248	366
248	765

to give the result

faculty_id	**first_name**	**last_name**	**course_id**
248	Anne	Denton	366
248	Anne	Denton	765

- Typically involve combining rows of two tables based on equality of “join attribute" (**faculty_id**)
- Notice that original tables avoid unnecessarily redundancy

Introduction to Constraints

faculty_id	first_name	last_name
248	Anne	Denton

faculty_id	course_id
248	366
248	765

- Foreign key
  - The two tables assume that for each **faculty_id** in the second table, there is an entry in the first table
- Primary key
  - Combining tables would not work if multiple faculty had the same **faculty_id**
- Other key and domain constraints discussed later

Challenge Questions

Try to represent the following data structures as relations

- Tree
- Linked list
- Graph
- Array
- Associative array

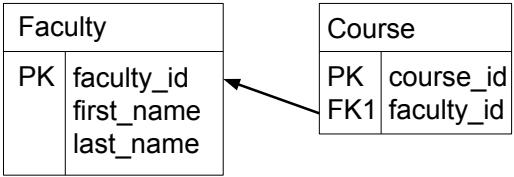
Table of Contents

Consider again the two basic relations

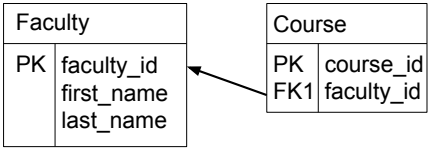
faculty_id	first_name	last_name
248	Anne	Denton

faculty_id	course_id
248	366
248	765

Later we will draw the table design as follows

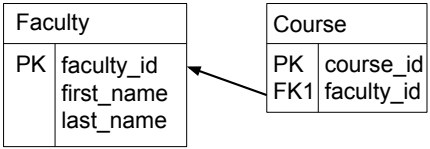


- The following only contains “schema" information, i.e. it does not include actual records



- It does include
  - Attributes
  - Primary keys
  - Foreign keys

- Richness of the model



- It does not allow for
  - Many-to-many relationships
  - Complex types of attributes
- Choice of primary key
  - Today primary keys typically don't have real-world meaning
- $\Rightarrow$ We will treat relational model as result of converting from Entity-Relationship model

Table of Contents

Success of the Relational Model

- Why is the relational model so successful?
- Because the mathematics behind it is very simple
  - Querying straightforward
  - Allows efficient query optimization
  - Not necessarily because it is the most intuitive
    - Hierarchical representations may be more intuitive
    - We may have to think about how to convert a hierarchical representation to a relational one
- A hierarchy can always be represented as a relation

History

- Introduced by Edward F. (Ted) Codd of IBM Research in 1970
  - Codd, E. "A Relational Model for Large Shared Data Banks" CACM, 13:6, June 1970
  - `https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf`
  - Won Turing Award in 1981
  - Turing Award somewhat like Nobel Prize for Computer Scientists

Question 6

Why is mathematics used more heavily for arguing about relational databases than earlier hierarchical ones?

1. Because relations are so simple that it is possible to use mathematics
2. Because relations are so difficult that it is necessary to use mathematics

Table of Contents

Formal Notation

- Relational schema
  - Name of relation
  - Attributes
  - $R(A_1, A_2, \ldots, A_n)$
  - Degree of relation is number of attributes (n in the above relation)
- In practice there are multiple graphical representation that are often used and will be introduced later

Schema vs. Instance

- Schema information
  - Specifies design (intension)
  - Attributes
  - Data types
- Database instance or database state
  - Set of *n*-tuples (extension)
  - Includes all records

Notation Across the Literature

Relational Algebra	Microsoft Access	Database Literature
relation	table	file
tuple	row	record
attribute	attribute (column)	field

Table of Contents

Differences Between Database Tables and Relations

- Relations cannot have duplicates since they are sets
  - Database tables can have duplicates
  - Database tables typically have a Primary Key constraint that does not allow duplicates
- Infinite sets cannot be represented in tables
- In databases the table defines the relation
  - If the content of a mathematical relation can be computed, e.g. “greater than", there is no need to create a database table
- In theoretical computer science relations are typically binary, i.e., sets of ordered pairs of set elements
  - Database tables often have many columns
  - While database tables can represent the relationship between two sets, other common uses include storing of tabular information, e.g., first name, last name

Question 7 (Multiple answers can be correct)

Consider the equality relation over two sets, each containing the numbers 1 through 7. Which of the following statements is correct?

1. Since the relation is infinite, it cannot be represented in a database table
2. The relation could, in principle, be represented in a database table
3. If the relation was represented as a table, the table would have at least two columns
4. Querying this relation is the easiest way to establish equality between values in database tables

Question 8

Consider a database table that contains first name, last name, and age of a person. Which of the following statements is correct?

1. This table does not relate different sets
2. The table could be viewed as a relation over the valid sets of first names, last names, and ages, that contains those persons who are represented by the database
3. This table cannot be viewed as a relation because there is no ordering to the persons
4. This table would violate the expectations of relations if there were two 22-year-old persons with name John Smith to be represented

Table of Contents

Types of Keys

- A superkey is set of attributes that uniquely defines a tuple
- A key is a minimal superkey
- $\Rightarrow$ you cannot remove an attribute from a key without destroying the uniqueness property
- If a relation has more than one key, each of them is called Candidate Key
- One of them is designated to be Primary Key, PK
- All others are called Alternate Keys, AK

Question 9 (Multiple answers can be correct)

Consider a database table that has 3 attributes: The course identifier (e.g. “CSci 366" or “CSci 765"), its name, and the number of credit hours. Assume that no two courses have the same course identifier or the same name. Which of the following is a superkey of the relation?

1. The course identifier
2. The number of credit hours
3. The course identifier and course name together
4. All attributes together

Question 10 (Multiple answers can be correct)

Consider a database table that has 3 attributes: The course identifier (e.g. “CSci 366" or “CSci 765"), its name, and the number of credit hours. Assume that no two courses have the same course identifier or the same name. What would mathematically qualify as a key of the relation?

1. The course identifier
2. The number of credit hours
3. The course identifier and course name together
4. All attributes together

Key Constraint

- No two tuples with the same Primary Key allowed
- The Key Constraint refers explicitly to the Primary Key
  - For Alternate Keys, the constraint is called uniqueness constraint

Entity Integrity Constraint

- The Primary Key must not be null
- You do not need to specify this explicitly
- There is no way to allow a Primary Key to be null
- Having a Primary Key that is null would make join queries meaningless
- Whether Alternate Keys can be null depends on the DBMS

Primary Key Choice

- Notation: PK
- Must be unique for each entity
- Must be available for each entity, or there must a strategy for assigning dummy values when it isn't available
- Should not pose security risks
- Should not consume a lot of storage (for speed and storage reasons)
- Should not change later
- Often, if not always, it is best to create a new surrogate key without real world meaning

The Case for Using a Surrogate Key

- The PK will be replicated in many places as Foreign Key, which causes problems for most real-world attributes
  - Causes storage and speed problems for large types like
  - Causes problems if PK were to ever change
  - Causes problems for potentially sensitive attributes
- For real-world quantities there are usually cases where the value is not known or does not exist, and using dummy values tends to be inelegant
- Most of the reasons for using a real-world attribute are not compelling
  - Records can be searched based on any type of information not just the key
  - Duplicate entries in non-primary key attributes can be prevented through the uniqueness constraint (Alternate Key, or AK)

Question 11 (Multiple answers can be correct)

Consider a database table that has 3 attributes: The course identifier (e.g. “CSci 366" or “CSci 765"), its name, and the number of credit hours. What to you think would be a suitable Primary Key choice?

1. The course identifier should almost certainly be used
2. The course identifier could be used, but may not be a good choice because the course number can be changed
3. The course name would be best because it is most descriptive
4. An additional surrogate key is probably the best choice
5. All attributes together are best as key

Table of Contents

Foreign Keys

- A Foreign Key, FK, is an attribute, or set of attributes, that
  - has the same domain as the PK of another relation
  - references the PK of another other relation
- Referential Integrity Constraint
  - The FK either references an existing PK or is null
- Notation: An arrow is drawn from the FK to the PK that it references

Question 12 (Multiple answers can be correct)

Consider a database table of courses, and another database table of sections of those courses. Assume that a course can have multiple sections but a section is a section of only one course. Which of the following statements are valid?

1. The section table could define a FK that reference the PK of the course table
2. The course table could include the PK of the section table as FK
3. A separate table could be created that has the PKs of both the course and the section tables as FKs

Properties of Foreign Keys

- Foreign Keys capture relationship information
- A FK can be null unless a constraint on null is explicitly imposed
- A FK can refer to its own relation (recursion)
- A FK consists of as many attributes as the PK it references
  - Hence the goal of using atomic Primary Keys
- The attributes in the FK may or may not have the same name as the attributes of the PK in the other relation

Question 13 (Multiple answers can be correct)

Consider a database table of sections with a FK that references the course identifier of a course table. What are valid statements about the foreign key?

1. From a database perspective, courses without sections could, in principle, be represented
2. From a database perspective, sections without courses could, in principle, be represented
3. Section records can refer to course numbers for which there is no course entry

Table of Contents

Constraint On Null

- Constraints on null are often imposed for practical reasons, e.g., a record of a person that does not include last name may be useless
- Primary key
  - The primary key must not be null for many reasons
  - The programmer does not have to state this, since entity integrity guaranteed in most or all DBMSs
- Alternate Key, i.e. “unique" attributes
  - May or may not have a uniqueness constraint imposed by default (vendor-dependent), check for the DBMS you are using
- Foreign Key
  - The FK attribute can be null by default
  - Imposing constraint on null can be used to enforce modeling constraints

Problems with Null Values

- Can mean a variety of things
  - Missing
    - There is a value but it isn't provided
    - E.g., a person may not share private information
  - Non-existent
    - There is no value
    - E.g., a person may not have a middle name
  - Not applicable
    - There cannot be a value in the context
    - E.g., a single-family house does not normally have an apartment #
  - Not known
    - It is not known what is the reason for the null value
    - E.g., a middle name could be missing or non-existent
- Waste storage

Domain Constraints

- Not just data type but often also format
- Can be set of values
- Can be range of values
- May included number of significant digits for numbers
  - Keep in mind that many database applications have financial aspects
- May be complex types like dates
- Array data are often better represented through additional tables

Semantic Integrity Constraints

- All constraints that are not otherwise covered
- Depend on the logic of the application itself
- How many of these should be enforced within the DBMS is often debated
  - If it is centrally enforced, it will apply to all applications
  - May make application DBMS-vendor specific
  - May result in inflexibility

Summary of integrity constraints

Key constraint: PK must be unique, i.e., no two tuples with the same PK

Uniqueness constraint: AK must be unique (whether NOT NULL is implied depends on DBMS)

Entity integrity: PK must not be null (typically enforced)

Constraint on null: NOT NULL can be separately enforced for any attribute

Referential integrity: FK must reference the PK value of an existing tuple, or be null (if NOT NULL is intended, it has to be specified separately)

Domain constraints: What type of attribute is allowable

Semantic integrity constraints: Specified through constraint specification statements

Table of Contents

Constraint Violations

- Default response is rejection of action
- All constraints can be violated through record insertions
- Only referential integrity can be violated through record deletions
- Enforcement of constraints can involve
  - Only schema information
  - The relation on which an operation is to be performed
  - More than one relation

Question 14

Which of the following constraints depends only on schema information

1. Entity Integrity Constraint
2. Primary Key Constraint
3. Referential Integrity Constraint

Question 15 (Multiple answers can be correct)

Which of the following constraints can be violated through a record deletion

1. Entity Integrity Constraint
2. Primary Key Constraint
3. Referential Integrity Constraint

Referentially Triggered Actions

- When inserting records violations are common, and the typical response is to ask the user to resubmit
- For deletions, only FKs can lead to violations
  - Cannot delete a record with PK that a FK still references
- Automatic response may be desirable
- Referentially triggered actions
  - Cascading deletion, i.e. also delete records that reference PK
  - Other choices like "set null" will be discussed in the context of SQL
  - Referentially triggered actions can also be used for updates

Question 16 (Multiple answers can be correct)

Consdier a course and a section table, for which a foreign key in the section table references the primary key of the course table. Which of the following actions may trigger a referentially triggered action

1. Deleting a record from the section table for which the corresponding course record still exists
2. Deleting a record from the course table for which the corresponding section record still exists
3. Inserting a record into section table for which no corresponding course record exists

Designing Databases

- Relational model not very rich
- Led to Entity Relationship (ER) Modeling
- ER Models are then mapped to relational schemas