Lesson 3: Diving Deeper into the Relational Database Model

Quick Note

I will post this on Brighspace and put the dates in the syllabus as well.

- Your midterm will be on Monday, March 10.
- Your final will be on Monday, May 12.
- My goal is not to trick you, but rather to ensure your taking away the key concepts.

Characteristics of a Relational Table

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (tuple) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute and each column has a distinct name
4	Each intersection of a row and column represnets a single data value
5	All values in a column must conform to the same data format.
6	Each column has specific range of values known as the attribute domain.
7	The orders of the rows are immatrial to the DBMS.
8	Each table must have an attribute or comibination of attributes that uniquely identifies each row.

What does this mean?

- Let's look at an example.

Keys a Deeper Dive

- A **key** consists of one or more attributes of a table (aka relation).
  - Keys are based on determination: The value of A can be used to look up the value of B. OR If someone knows my Employee ID, they can look up my First Name.
  - Relationships based on determination are known as **Functional Dependence**.
  - Let's use our same example.

Functional Dependence Notation

Breaking Down the Notation
player_id → first_name
DETERMINANT → DEPENDENT
The player_id functionally determines first_name
The first_name is functionally dependent on player_id

Full Functional Dependency

- This occurs if the entire collection of attributes in the determinant is necessary for the relationship.
    - In other words, if we remove the/an attribute from the determinant, determination holds.
- Let's look at an example again!

Types of Keys

- Remember that Functional Dependence is the bases of keys (that's how we got here)
- A **primary key** is an attribute or combination of attributes that uniquely identifies any given row.
    - Every table must have a primary key!
    - Recall the discussion of constraints; primary keys have both NOT NULL and unique constraints. This characteristic is also known as **entity integrity**.

Types of Keys (Continued)

- **Superkeys** uniquely identify (or functionally determine) the attributes of any row.
- **Composite keys** are keys that are made up of more than one attribute.
  - In this circumstance these attributes are referred to as **key attributes**.

Types of Keys (Continued)

- **Candidate keys** a set of one or more attributes that could be used to uniquely identify a row in a table
    - They should be minimal in that they contain as few attributes as possible.
    - They should be irreducible, in that you cannot remove an attribute and maintain superkey status.
    - In practice attributes like first name, last name and email addresses typically don't work out.
    - Sometimes the names of things in general aren't great, because names can change.

Types of Keys (Continued)

- A **foreign key** is a primary key of one table that has been placed in another to create a common attribute.
    - Foreign keys ensure **referential integrity**, that is every reference to an entity instance (row) to another entity instance (row) is valid.
    - Remember how we talked about naming conventions!

Types of Keys (Continued)

- **Secondary keys** are used specifically for retrieving data.
  - They may not yield unique result sets, but they are more intuitive.
  - Example: You're at the grocery store and you realize you forgot Shopper's Club Card. You want a sale item, so you enter your phone number.

Null Flashback

- Recall that I referenced NULLs when I spoke of data types.
  - A NULL is not itself a datatype, but it represents a possible value for every data type
- They can signify any of the following scenarios (not exhausitve)
  - An unknown value
  - A missing value (that is known)
  - A scenario where something is not applicable
- Columns with NULL values can never be a candidate key

Integrity Rules and Relational Database Keys

- Review tables 3.3 and 3.4 in your book!

Relational Algebra

- This is based on Set Theory and Predicate Logic
- A **relvar** is a [var]iable that holds a [rel]ation.
  - The algebra makes more sense with this term, but remember that a relation is equivalent to table.
- Relational operators have the property of closure.
  - Relational algebra using one or more relations (tables) results in another relation (table).

Select

- Uses only a single table as input (unary).
- Returns all rows or any rows that satisfy a particular condition.
- It subsets rows, not attributes.

Project

- Also uses only a single table as input (unary).
- Returns all rows, but only the defined attributes.
- It subsets attributes, not rows.

Union

- Uses two tables.
- Table must have the same set of attribute characteristics (think data types).
    - Known as being union compatible
- Keeps all rows for boths tables, but excludes duplicates.

Intersect

- Uses two tables.
- The tables must be union compatible.
- Keeps only rows that are in common between both tables.

Difference

- Uses two tables.
- The tables must be union-compatible.
- Keeps only rows that are not in common between both tables.

Product

- Uses two tables.
- Yields all possible pairs of rows from two tables.
- Can also be known as a cartesian product.

Join

- Uses two (or more) tables, but two in our example for simplicity.
- Enable tables to combined using a common attribute or set of attributes.

Join Continued

- Natural Join
  - The key here is the steps. To perform a natural join you use:
    a. Product
    b. Join
    c. Project
- Equiijoin
  - Joins based on equality
- Theta Join
  - Joins based on inequality comparison <, >, <=, >= or !=

Join Continued

- Inner Join
    - Keep only matching records.
- Outer Join
    - Left [Outer] Join
        - Keep all records in the left table
    - Right [Outer] Join
        - Keep all records in the right table

Divided

- Uses two tables.
- Enables questions to be answered about how one set of data is associated with a set of values of another.

Relational Algebra Reference Table

Symbol	Operator
σ	SELECT
π	PROJECT
$\cup$	UNION
$\cap$	INTERSECT
$-$	DIFFERENCE
$\times$	PRODUCT
$\bowtie$	JOIN
$\div$	PRODUCT

Deeper Dive Into Relationships

1:1 & 1:M

- These are fairly straightfoward, so let's just use our example!

M:N

- Composite Entity are used to implement a many to many relationship in a relational database
    - These could be referred to as bridge tables, associative tables or junction tables.
- Linking Table

Homework

- Read chapters 3 and 4