Normal Form, SQL Constraints & Triggers

Kathleen Durant

CS 3200

Lesson 7

Lecture Outline

- MY SQL Installation
- Introduction to Normal Form
- Constraints
- Triggers
- NULL Values

MY SQL Notes

- One directory for the catalog, which is itself a database.
- Consists of tables specifying privileges – who can access what
  - database-level privileges
  - table-level privileges etc.
- One directory for each user database.
- Each table is represented by three files:
  - one for the per-table metadata
  - one for the data
  - one for any indices on the table

Normal Form: Codd's Objectives

- Free the collection of relations from undesirable insertion, update and deletion **dependencies**
  - Duplicate data in multiple rows
    - Forced to update/delete all copies of a piece of data
    - How do you know you got all copies of it?
- Reduce the need for restructuring the collection of relations
  - Build an extensible design
- Make the relational model more informative to users
  - Cleaner model should be easier to understand
- Make the collection of relations neutral to the query statistics
  - Designed for general purpose querying

Redundancy and Normalization

- Redundant data
  - Can be determined from other data in the database
  - Leads to various problems
    - INSERT anomalies
    - UPDATE anomalies
    - DELETE anomalies
- Normalization aims to reduce redundancy

First Normal Form

- First normal form
  - Tuples in a relation must contain the same number of fields
  - The domain of each attribute is atomic
  - The value of each attribute contains only a single value
  - No attributes are sets
    - No repeating groups

Levels of Normal Form

- Level 1: No repeating entities or group of elements
  - Do not have multiple columns representing the same type of entity
  - Primary key that represents the entity
- Example: Table mother (MotherName varchar(40), child1 varchar(20), child2(varchar(20)…child8 varchar(20))
  - Create 3 tables: Mother, Children and Offspring
    - Offspring links Mother and Children together

1NF vs. Not 1NF

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Mother Id	Mother Name	Child1	Child2	Child3	Child4
1	Elsa	Mary	Alice	NULL	NULL
2	Golda	George	Fred	NULL	NULL
3	Viola	Ava	NULL	NULL	NULL
4	Iris	Kayla	NULL	NULL	NULL
5	Daisy	Harry	NULL	NULL	NULL

- Create Table Mother, Table Offspring and a Table Children
- Link them together via a unique representation (social security number)

Parent Id	Offspring Id
1	11
1	12
2	13
2	14
3	15
4	16
5	17

Mother Id	Mother Name
1	Elsa
2	Golda
3	Viola
4	Iris
5	Daisy

Offspring Id	Offspring Name
11	Mary
12	Alice
13	George
14	Fred
15	Ava
16	Kayla
17	Harry

Benefits

- No duplicated data
- Beneficial when you want to extend your database by adding more concepts
- Example: Say you now want to model the father relationship ?
- With the not 1NF solution you are forced to duplicate all of the offspring data in the father relation

Adding the Father Relation

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Mother Id	Mother Name	Child1	Child2	Child3	Child4
1	Elsa	Mary	Alice	NULL	NULL
2	Golda	George	Fred	NULL	NULL
3	Viola	Ava	NULL	NULL	NULL
4	Iris	Kayla	NULL	NULL	NULL
5	Daisy	Harry	NULL	NULL	NULL

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Father Id	Father Name	Child1	Child2	Child3	Child4
21	Sam	Mary	Alice	Fred	NULL
22	Sal	George	NULL	NULL	NULL
23	Hal	Ava	NULL	NULL	NULL
24	Ed	Kayla	NULL	NULL	NULL
25	George	Harry	NULL	NULL	NULL

- Forced to duplicate child data in both mother and father relationship
- Leads to errors in child data during updates and deletions
- Hard to query child data
- Limits schema
  - 5 children?

1NF with Father Relation

Father Id	Father Name
21	Sam
22	Sal
23	Hal
24	Ed
25	George

Mother Id	Mother Name
1	Elsa
2	Golda
3	Viola
4	Iris
5	Daisy

Parent Id	Offspring Id
1	11
1	12
2	13
2	14
3	15
4	16
5	17
21	11
21	12
21	14
22	13
23	15
24	16
25	17

OneDegree Table contains Mapping between parent and offspring

Offspring Id	Offspring Name
11	Mary
12	Alice
13	George
14	Fred
15	Ava
16	Kayla
17	Harry

Second normal form

- Schema must be in first normal form
  - You have eliminated group sets
  - Every tuple has a unique key
- Each field not in the primary key provides a fact about the entity represented via the (entire) primary key
  - The primary key must be minimal – no extra fields thrown in
  - No partial dependency on part of the primary key
  - Only applies to composite primary key
- Helps you identify a relation that may represent more than one entity
- All fields must be functionally dependent on the complete primary key

Example 2NF vs. Not 2NF

1st Normal Form but NOT 2ndNORMAL FORM				
Mother Id	First Name	Last Name	Hospital	Hospital Address
1	Elsa	General	BIDMC	Boston
2	Golda	Major	MGH	Boston
3	Viola	Funt	TMC	Cambridge
4	Iris	Batter	BIDMC	Brighton
5	Daisy	Mae	Mayo	Allston

2nd NORMAL FORM			
Mother Id	First Name	Last Name	Hospital Id
1	Elsa	General	1
2	Golda	Major	2
3	Viola	Funt	3
4	Iris	Batter	1
5	Daisy	Mae	4

	2nd NORMAL FORM	
Hospital ID	Hospital	Hospital Address
1	BIDMC	Boston
2	MGH	Boston
3	TMC	Cambridge
4	Mayo	Allston

3rd Normal Form

- No dependencies between 2 non-key attributes
- Bill Kent: Every non-key attribute must provide a fact about the key, the whole key and nothing but the key

Example 3NF vs. Not 3NF

1st Normal Form but NOT 2ndNORMAL FORM				
Mother Id	First Name	Last Name	Hospital	Hospital Address
1	Elsa	General	BIDMC	Boston
2	Golda	Major	MGH	Boston
3	Viola	Funt	TMC	Cambridge
4	Iris	Batter	BIDMC	Brighton
5	Daisy	Mae	Mayo	Allston

2nd NORMAL FORM			
Mother Id	First Name	Last Name	Hospital Id
1	Elsa	General	1
2	Golda	Major	2
3	Viola	Funt	3
4	Iris	Batter	1
5	Daisy	Mae	4

	2nd NORMAL FORM	
Hospital ID	Hospital	Hospital Address
1	BIDMC	Boston
2	MGH	Boston
3	TMC	Cambridge
4	Mayo	Allston

Normal Form Tips

- Review your attributes in your tables and ensure that they are facts about the complete key and only the complete key
- No duplicating groups in a table
- Split many to many relationships up into 2 many to 1 relationships by identifying the relation that maps them together

Example

- Students takes Courses M-to-M relationship
  - Many students to a Course
  - Many courses to a Student
- Represent using 2 M-to-1 relationships
  - Students has an Enrollment M-to-1
  - Enrollment in a Class 1-to-M

Student Table
StudentID

Enrollment
StudentId, ClassId

Class Table
ClassID

NULL and Missing Data

Missing data values in relations

- Allowing fields to have no value allows us to model the real world as well as mathematics
- We need to be able to semantically represent the NAN or a value that does not have a value (yet) or a value that does exist we just do not know it
- Missing data is prevalent in many fields of study – the majority of data is missing
  - Goal: get a good representation of what is not there given the minority that is provided
  - Many statistical and data mining techniques defined to deal with missing data

NULLS in SQL

- NULL is a placeholder for missing or unknown value of an attribute.
  - It is not itself a value, therefore it has no data type.
- Codd proposed to distinguish two kinds of NULLs:
- A-marks: data Applicable but not known (for example, someone's age)
- I-marks: data is Inapplicable (telephone number for someone who does not have a telephone, or spouse's name for someone who is not married)

NULL and its impact on SQL

- SQL allows field values not to have a value
  - Sometimes the field's value will not be known until later or it is inapplicable
  - Example: Later: (e.g., a rating has not been assigned) or Inapplicable (e.g., no spouse's name).
  - SQL provides a special value NULL for such situations.
- Presence of NULL complicates many issues:
  - Special operators needed to check if value is (not) NULL.
  - Is rating>8 true or false for rating=NULL? What about AND, OR and NOT connectives?
- We need a 3-valued logic (true, false and unknown).
- Semantics of 3-valued logic must be defined consistently.
  - WHERE clause eliminates rows that do not evaluate to true.

Problems with NULLs

- Defining selection operation: if we check tuples for some property like Mark > 40 and for some tuple Mark is NULL, do we include it?
- Defining intersection or difference of two relations: are two tuples <John, NULL> and <John,NULL> the same or not?
- Additional problems for SQL: do we treat NULLs as duplicates?
- Do we include them in count, sum, average and if yes, how? How do arithmetic operations behave when an argument is NULL?

Solutions to NULL: three values in Logic

- Use three-valued logic instead of classical two-valued logic to evaluate conditions.
- When there are no NULLs around, conditions evaluate to true or false, but if a null is involved, a condition will evaluate to the third value (‘undefined’, or ‘unknown’).
- This is the idea behind testing conditions in WHERE clause of SQL SELECT: only tuples where the condition evaluates to true are returned.

3-VALUED LOGIC

X	Y	X AND Y	X OR Y	NOT X
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	UNKNOWN	UNKNOWN	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
UNKNOWN	TRUE	UNKNOWN	TRUE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKOWN	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	UNKNOWN
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	UKNOWN	FALSE	UNKNOWN	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

FALSE = 0, TRUE = 1, UNKNOWN =1/2 NOT(X) = 1-X,
AND(X,Y) =MIN(X,Y), OR(X,Y) = MAX(X,Y)

SQL: NULLs in conditions

- Select SID from Sailor where rating > 5
- Execution: rating > 5 evaluates to 'unknown' on the last tuple

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	Guppy	5	35.0
58	Rusty	NULL	35.0

SID	Sname	Rating	Age
28	Yuppy	9	35.0

SQL: NULLs in conditions

- Select SID from Sailor where rating > 5 **OR** Name = 'Rusty'
- Execution: rating > 5 evaluates to true on the last tuple

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	Guppy	5	35.0
58	Rusty	NULL	35.0

SID	Sname	Rating	Age
28	Yuppy	9	35.0
58	Rusty	NULL	35.0

SQL: NULLs in Arithmetic

- Select SID, Rating * 10 as NewRating from Sailor
- Arithmetic operations applied to NULL result in NULLs

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	Guppy	5	35.0
58	Rusty	NULL	35.0

SID	NewRating
28	90
31	30
44	50
58	NULL

SQL with NULLS: Aggregates

- Select avg(Rating) as AVG,
  COUNT(Rating) as NUM,
  COUNT(*) as ALLNUM,
  SUM(Salary) as SUM from Sailors

  - AVG = 5.67
  - NUM = 3
  - ALLNUM = 4
  - SUM = 17

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	Guppy	5	35.0
58	Rusty	NULL	35.0

Outer Joins

- When we take the join of two relations we match up tuples which share values
  - Some tuples have no match are ‘lost’
  - These are called dangles
- Outer joins include dangles in the result set and use NULLs to fill in the blanks
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

Alternative Solution: Default Values to Express Loss of Data

- Default values are an alternative to the use of NULLs
  - If a value is not known a particular placeholder value – the default is used
  - Actual values within the domain type so no need for 3 value-logic
  - Default values can provide more meaning than NULLs
    - None
    - Unknown
    - Not supplied
    - Not applicable

Default Value Example

- Default values are
  - ???? For Name
  - -1 for Rating and Age
- Hopefully no one has a name of ???? and rating and age cannot really be = -1 so can identify your default values
- What about
- Update Sailors

  set age = age +1?

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	????	5	-1
58	Rusty	-1	35.0

Problems with default values

- They are real values in the domain of the variable
  - They can be updated like any other field value
  - You need to use a value that will not appear in any other circumstances
  - They may not be interpreted correctly
  - You need compatibility in the domains
    - You can't have a string such as 'unknown' stored in an integer field
- You may want to just use NULL

NULL support in SQL

- SQL allows you to INSERT NULLs
  - Example: UPDATE Sailors set rating = NULL where Name = ‘Mark’
- Separate function to test for NULL
  - Example: SELECT Name from Sailor where rating IS NOT NULL
  - Example: SELECT Name from Sailor

    where rating IS NULL

NULL or Default Value

- Which method to use?
- Default values should not be used when they might be confused with ‘real’ values
- NULLs can (and often are) used where the other approaches seem inappropriate

Triggers

- Similar to Integrity constaints

Integrity Constraints

- An IC describes conditions that every legal instance of a relation must satisfy.
  - Inserts, deletes, updates that violate IC's are disallowed.
  - Can be used to ensure application semantics (e.g., sid is a key), or prevent inconsistencies (e.g., sname has to be a string, age must be < 200)
  - Types of IC's: Domain constraints, primary key constraints, foreign key constraints, general constraints.
- Domain constraints: Field values must be of right type. This is always enforced.

General Constraints

- Allows you to define a constraint beyond key or unique fileds.
  - Can use queries to express constraint.
  - Constraints can be named uses the CHECK keyword

- CREATE TABLE Sailors (sid INTEGER, sname CHAR(10), rating INTEGER, age REAL, PRIMARY KEY (sid), CHECK (rating >= 1 AND rating <= 10)

- CREATE TABLE Reserves (sname CHAR(10), bid INTEGER, day DATE,PRIMARY KEY (bid,day), CONSTRAINT noInterlakeRes CHECK (`Interlake' <> (SELECT B.bname FROM Boats B WHERE B.bid=bid)))

Constraints over multiple tables

- Create a contraint such that: *Number of boats plus number of sailors is < 100*
- CREATE ASSERTION smallClub CHECK ((SELECT COUNT (S.sid) FROM Sailors S) +(SELECT COUNT (B.bid) FROM Boats B) < 100)

Triggers

- Trigger: procedure that starts automatically if specified changes occur to the DBMS
- A trigger has three parts:
- Event
  - Change to the database that activates the trigger
- Condition
  - Query or test that is run when the trigger is activated
- Action
  - Procedure that is executed when the trigger is activated and its condition is true

Trigger Options

- **Event** can be insert, delete, or update on DB table
  - Condition can be a true/false statement
    - All employee salaries are less than $100K
- **Condition** can be a query
  - Interpreted as true if and only if answer set is not empty
- **Action** can perform DB queries and updates that depend on
  - Answers to query in condition part
  - Old and new values of tuples modified by the statement that activated the trigger
  - Action can also contain data-definition commands, e.g., create new tables

When to Fire the Trigger

- Triggers can be executed once per modified record or once per activating statement
  - Row-level trigger versus a Statement Level Trigger
  - Trigger looking at the set of records that are modified versus the actual individual values of the old and the new values
- Should trigger action be executed before or after the statement that activated the trigger?
  - Consider triggers on insertions
    - Trigger that initializes a variable for counting how many new tuples are inserted: execute **trigger before insertion**
    - Trigger that updates this count variable for each inserted tuple: **execute after each tuple is inserted** (might need to examine values of tuple to determine action)
    - Trigger can also be run **in place of the action**

Trigger Example

- CREATE TRIGGER YoungSailorUpdate
  AFTER INSERT ON SAILORS
  REFERENCING **NEW** TABLE NewSailors
  FOR EACH STATEMENT
  INSERT
  INTO YoungSailors(sid, name, age, rating)
  SELECT sid, name, age, rating
  FROM NewSailors N
  WHERE N.age <= 18

Trigger has acess to **NEW** and **OLD** values

Trouble with Triggers

- Action can trigger multiple triggers
  - Execution order is arbitrary
- Challenge: Trigger action can fire other triggers
  - Very difficult to reason about what exactly will happen
    - Trigger can fire “itself” again
  - Unintended effects possible
- Many religious wars on triggers evil vs. not evil
  - Analogous to the gun control debate in society
    - Triggers do not corrupt databases people who write triggers do
- Example: Triggers defined to monitor Stock prices
  - Once multiple triggers are activated can’t shut off
  - Sit back and watch the world’s economic system collapse
- Introducing Triggers leads you to deductive databases
  - Need rule analysis tools that allow you to deduce truths about the data

MY SQL limits the user of triggers

- Triggers not introduced until 5.0
- Not activated for foreign key actions
- No triggers on the mysql system database
- Active triggers are not notified when the meta data of the table is changed while it is running
- No recursive triggers
- Triggers cannot modify/alter the table that is already being used
  - For example the table that triggered it

Summary

- NULL for unknown field values brings many complications to a DBMS
  - However, unknown values are part of the real world
- SQL allows specification of rich integrity constraints
  - Define constraints across tables
- Triggers respond to changes in the database
  - Strength: Very Powerful
  - Weakness: Very Powerful