Logic and Databases

Jeff Davis

LAIKA, Inc.

Relational Model

- Relational Model is just logic combined with set theory for the purpose of data management.
- Each relation has a predicate
- For every set of arguments that would make that predicate true, there exists an associated tuple in the relation with those values.

Predicates and Relations

- predicate means: a truth-valued function
- Let P(name, pop) = “There exists a city named [name] with population [pop].”
  - Evaluated by a human
- P(“Portland, OR”, 5e5) -> true
  - Appears in relation
- P(“Portland, OR”, 5e1) -> false
  - Doesn't appear in relation

Natural Language

- Relational operators don't just *do* something, they *mean* something.
- That meaning can be expressed in natural language
- Useful for translating business rules to schema design and queries

JOIN is AND

- Relation R1 with predicate “Company [C] makes part [P].”
- Relation R2 with predicate “Part [P] requires material [M].”
- The relation (R1 JOIN R2) has predicate “Company [C] makes part [P] and part [P] requires material [M].”

UNION is OR

- Relation R1 with predicate “A Student has name [N] and address [A].”
- Relation R2 with predicate “A Teacher has name [N] and address [A].”
- Relation (R1 UNION R2) has predicate “A Student has name [N] and address [A] or a teacher has name [N] and address [A].”

Projection is Existential Operator

- If you have a relation with the predicate “Company [C] is at location [L].”
- The predicate when you “project away” L (e.g. “SELECT C FROM R”) is: “There exists some location L such that company [C] is located at L”.

Data Migration

- Extract enough of the meaning from the source system that the information can be accurately translated for the target system.
  - Extract context-insensitive, unambiguous meaning
  - Test assumptions

Meaning of an Attribute

- Attributes can have meaning on their own, but it may be more complex and less useful than it may seem.
- Many simple predicates are simple ANDed lists: “Company [C] has budget [B] and location [L].”
- Consider a predicate like “During year [Y] company [C] had budget [B].”
- Use the entire predicate to express the meaning of the data.

Context Sensitivity

- Context Sensitive – depends on implementation details unrelated to business rules:
  - “Request #123 is in state 'pending'.”
  - In this case, 'pending' does not have a specific meaning to the business.
- Context Insensitive – meaning directly applies to business rules:
  - “Request #123 has been approved by the purchasing department, but has not arrived yet.”

Test Assumptions

- Helps eliminate ambiguity and reduce context sensitivity
- Test whether 'pending' means that the product has been purchased and has not arrived yet:
  - SELECT 'assumption is wrong!',* FROM request WHERE state='pending' and request_id NOT IN (SELECT request_id FROM purchase)

Test Assumptions (Cont.)

- If that query returns tuples, that means that our assumption – that 'pending' means 'purchased but has not arrived' – is false, and needs to be re-examined.
- Beware: The previous query has a hidden problem. If there are purchases with a request_id = NULL, the NOT IN (...) will never return TRUE, and thus will never return tuples, even if the assumption above has numerous contradictions!

Limitations

- Testing assumptions by looking at the data has limited effectiveness.
  - Data set might not exhaust all of the edge cases
  - If contradictions remain, the target system might misbehave in subtle ways
  - Preferably, both the source and target system are well documented.
  - If not – uh.. – call the person who entered the data and ask them what it means?

NULLs

- False-like when the WHERE predicate evaluates to NULL
- Truth-like when a constraint (e.g. UNIQUE, FK, CHECK) evaluates to NULL
- “Unknown” and also “no value” (a.k.a. “not applicable”)
- Fewer tautologies, e.g.: X OR (NOT X) may not evaluate to TRUE. Beware of relying on these tautologies implicitly!

Example

```
test=# select sum(column1) from (values (1),(NULL)) t;
 sum
-----
   1
(1 row)

test=# select (1 + NULL) as plus;
 plus
------

(1 row)
```

Unknown

- A third truth value, introducing 3-value logic (3VL)
- Operations that treat NULL as unknown:
  - IN (...)
    - Using NOT IN (...) in a WHERE clause can be particularly tricky. If any NULLs exist among the values, NOT IN (...) will never return true!
  - Functions
  - operators

No Value

- "Not Applicable"
- Operations that treat NULL as "No Value"
  - Aggregate functions
  - OUTER JOIN
- Even if you never design your database to represent "no value" as NULL, you still encounter the "no value" version of NULL.

Don't Mix NULLs

- Avoid mixing “Unknown” and “No Value” NULLs.
- Find unprofitable customers: WRONG:
  - SELECT name FROM customer LEFT JOIN orders ON (orders.customer_id = customer.customer_id) GROUP BY name HAVING SUM(price) < 100.00
  - SUM(price) will return a “No Value” NULL for customers who don't buy anything, but “<” assumes that it's an “Unknown” NULL and the entire predicate evaluates to NULL (false-like).

Separate NULLs

- Instead, the best approach is to eliminate “no value” NULLs before using them in an operation expecting “unknown” NULLs (and vice versa).
- Find unprofitable customers: RIGHT:
  - SELECT name FROM customer LEFT JOIN orders ON (orders.customer_id = customer.customer_id) GROUP BY name HAVING SUM(COALESCE(price,0)) < 100.00

Conclusion

- Try to translate natural language to SQL and SQL to natural language, so that you can tie queries directly to business questions.
- Use an iterative process to clean up data before migrating it to a new system: make assumptions, test the assumptions, revise the assumptions, and repeat.
- Avoid the pitfalls of SQL NULLs