Chapter 9

Integrity

Topics in this Chapter

- Predicates and Propositions
- Internal vs. External Predicates
- Correctness vs. Consistency
- A Constraint Classification Scheme
- Keys
- Triggers
- SQL Facilities

Integrity

- Originally focused on keys, the theory of integrity has evolved to focus on constraints in general
- An integrity constraint is a boolean expression associated with a database that is required to evaluate at all times to true
- An integrity constraint can regarded as a formal expression of a business rule
- Constraints may be *a priori* or *a posteriori*

A priori vs a posteriori constraints

- Type implies an *a priori* constraint
- Since every attribute of every relvar is of some type, the collection of types is *a priori* for the relvar
- Business rule constraints – of the sort that are represented by uniqueness and value constraints, are a posteriori, that is, a result of decisions after the fact

Constraint Example

- CONSTRAINT SC1
  FORALL SX (SX.STATUS $\geq$ 1
  AND SX.STATUS $\leq$ 100) ;
- Represents a business decision about acceptable values, as opposed to the type of STATUS (be it INTEGER or STATUS), which represents the set of possible values

Key Constraints

- The general formulation for a relvar constraint is as follows: If a certain tuple appears in a certain relvar then that tuple satisfies a certain condition
- This applies equally to value constraints, uniqueness constraints and key constraints
- More generally, if certain tuples appear in certain relvars then those tuples satisfy a certain condition

Key Constraint Formal Definition – Beginning

- FORALL x# ∈ S#, xn ∈ NAME,
  xt ∈ INTEGER, xc ∈ CHAR,
  y# ∈ S#, yn ∈ NAME,
  yt ∈ INTEGER, yc ∈ CHAR

  (IF { S# x#, SNAME xn, STATUS xt,
  CITY xc} ∈ S AND
  { S# y#, SNAME yn, STATUS yt,
  CITY yc} ∈ S

Key Constraint Formal Definition – Conclusion

- THEN (IF x# = y#
  THEN xn = yn, AND
  xt = yt, AND
  xc = yc))
- This expresses that S# is a superkey, and possibly a candidate key

Predicates and Propositions

- An expression is a predicate
- Its variables are parameters to the predicate
- When we instantiate the variables, we are passing arguments to the predicate, and so turning it into:
- A proposition, which is either true or false
- A constraint is an expression, and therefore a predicate, which is checked by passing it arguments, and testing the proposition

The Golden Rule

- A relvar predicate is the conjunction (logical AND) of all constraints associated with any of its components
- Golden Rule: No update operation must ever assign to any relvar a value that causes its relvar predicate to evaluate to false
- A database predicate is the conjunction of all predicates of its relvars
- A database predicate is also golden

Checking the Constraints

- Constraints should be checked before attempting any insert or update or delete
- This is equally true from an implementation perspective and from the model
- To do otherwise is inefficient, and violates the Golden Rule

Internal vs. External Predicates – The Closed World Assumption

- Internal predicates are those understood and enforced by the system
- External predicates are those understood and implemented by the user
- Internal predicates should reflect external predicates
- If an otherwise valid tuple does not appear in a relvar, its corresponding proposition is false: The Closed World Assumption

Correctness vs. Consistency

- The Closed World Assumption is logically valid, but is unenforceable by the system
- External predicates are not understood by the system; therefore the system can enforce consistency, but not truth
- The external predicate for a relvar is its intended interpretation
- Thus a database may be populated by valid but false propositions

A Constraint Classification Scheme

- Constraints can apply to a database, a relvar, an attribute, or a type
- Type constraints check format and values immediately
- Attribute constraints are inherited from those of the declared type
- Relvar and database constraints inherit constraints from attributes and add business rule constraints in addition

Transition Constraints

- Transition constraints can apply to a database or a relvar, but not an attribute, or a type
- Transition constraints constrain certain actions, for example forbidding an update to change a status from "married" to "never married"

Keys – Topics

- Candidate Keys
- Superkeys
- Primary Keys
- Alternate Keys
- Foreign Keys
- Referential Integrity
- Referential Actions

Keys – Candidate Keys

- Let K be a set of attributes of relvar R. Then K is a candidate key for R if and only if it has both of the following properties:
- Uniqueness: No legal value of R ever contains two distinct tuples with the same value for k
- Irreducibility: No proper subset of K has the uniqueness property

Keys – Superkeys

- If SK is a superkey for relvar R and A is an attribute of R then SK implies A
- A superkey has the uniqueness property: no two tuples will have the same value
- But it does not have the irreducibility property: it can contain a subset that has the uniqueness property
- A superkey can contain subset superkeys; a candidate key cannot

Keys – Primary Keys and Alternate Keys

- If a relvar has two or more candidate keys, one must be chosen to be the primary key
- The others are then designated alternate keys
- This choice is logically arbitrary
- Logically, candidate keys are of paramount importance; choosing the primary key is ancillary

Keys – Foreign Keys

- Loosely, a foreign key is a set of attributes of some relvar R2 whose values are required to match values of some candidate key of some relvar R1
- R1 and R2 may be the same relvar, in the case of a recursive constraint, for example employee and manager
- Since relvars can be both referenced and referencing, the database contains referential paths connecting relvars in chains

Keys – Foreign Keys – Referential Integrity

- The database must not contain any unmatched foreign key values
- Originally foreign keys were defined in terms of the primary key in the referenced relvar, but this qualification is superfluous, although perhaps desirable in practice
- The relationship in one directional; it is not a requirement that every referenced candidate key value appear in the foreign keys of the referrer

Keys – Foreign Keys – Referential Actions

- DELETE may not violate the referential integrity constraint
- RESTRICT limits the action of the DELETE to just those tuples that do not have referring tuples in another relvar
- CASCADE broadcasts the DELETE to include any tuples that reference the affected tuples
- UPDATE requires similar behavior

Triggers

- Triggers were used to implement constraints in the days when the database software didn't
- They are used more now for auditing, or to carry out corresponding actions beyond those handled via referential integrity
- They should be avoided if possible, because they quickly become hard to manage: triggers can become chained, and ultimately may become recursive unintentionally

SQL Facilities

- SQL does not support type constraints, nor attribute constraints, nor relvar constraints, nor database constraints
- SQL supports base table constraints, which are a superset of a subset of relvar and database constraints
- SQL supports, for base tables:

  PRIMARY KEY, UNIQUE,

  FOREIGN KEY, CHECK, ASSERTION

Deferred Checking

- SQL constraint verification can be DEFERRABLE or NOT DEFERRABLE
- NOT DEFERRABLE means the check will be immediate
- DEFERRABLE offers the option of

  SET IMMEDIATE

SQL Trigger Syntax

- CREATE TRIGGER <trigger name>
  <BEFORE or AFTER> <event> ON
  <base table name>
  [REFERENCING <naming commalist>]
  [FOR EACH <row or statement>]
  [WHEN <bool exp>]
  <action> ;