SOFTENG 351: Fundamentals of Database Systems - Guide to Test 1

Michael Kemp

April 12, 2017

Contents

1 Basics of the relational model of data

1.1 The relational model of data

- *relations* are sets of *tuples* often represented as a table
- *attributes* are the column titles of a relation
- for each attribute we assign a *domain* which is a universal set containing all possible values(like a string; $dom(A) = string$
- *tuples* are the rows of a relation and all have the same structure in a relation
- if there is no value for an attribute then the value is *null*
- *relation schema* are a finite set R where attributes are A and each attribute $A \in R$ has a domain $dom(A)$
- relation schema can be written $R = \{A_1, A_2...A_n\}$ or $R(A_1, A_2...A_n)$ or $R(A_1 : dom(A_1)...A_n : dom(A_n)$
- All *R-tuples* (a tuple in a relation schema) are an element t of the Cartesian product of the domains of all the attributes $t \in A_1 \times A_2 \times ...A_n$ because each attribute's value is bound to it's respective domain.
- R*-relations* are a finite set r of R-tuples thus $r \subseteq dom(A_1) \times ...dom(A_n)$
- R-tuples can be written with their values $t = (A_1 : v_1...A_n : v_n)$
- A *database-schema* is a finite set S of relation schemata
- An S-database I consists of one R-relation for $I(R)$ for each relation R in S ($I = \{I(R)|R \in S\}$)
- Having duplicates in a database is normally useless so we have *keys* to ensure a uniqueness over an attribute or a combination of
- *superkey* over a relation schema R
  - finite, non-empty subset $K \subseteq R$
  - is *satisfied* if an R-relation r only has tuples with a unique combination of values for each attribute in the superkey.
  - A *Key* is a superkey if there is no subset which is also a satisfied superkey
  - A *foreign key* is when all of the combination of values of attributes(in the foreign key) is in the set of the table which defines the foreign key($[A_1..A_n] \subseteq S[A_1..A_n]$). Also the same S can not be referenced twice.

1.2 SQL as a data definition and query language

this whole section is not in the test

- is a DDL (Data Definition Language) and a DML (Data Manipulation Language)
- names are not case sensitive
- DDL
  - CREATE TABLE $< tablename >< attribute1domain, ... >$ attributes can be specified as NOT NULL
  - DROP TABLE $< tablename >$
  - ALTER TABLE followed by other stuff
- Domains
  - CHARACTER: Character strings of set length
  - VARCHAR: Character strings up to set length
  - NATIONAL CHARACTER

- INTEGER: Signed 32-bit integer
- SMALLINT: Signed 16-bit integer
- NUMERIC = DECIMAL: Numeric values with definable precision and scale
- FLOAT: Approximate numeric values with definable precision(up to 64)
- REAL: Approximate numeric values up to 64 precision
- DOUBLE PRECISION: Double a REAL
- BIT = BOOLEAN: TRUE, FALSE, false or true
- BIT VARYING: Bit strings
- DATE: YYYY-MM-DD but can use single digits for month and date
- TIME: HH:MM:SS with optional nano seconds and seconds up to (including) 61.999999

- to insert a tuple to a relation all values must be known thus null exists if it doesn't exist or is no yet known
- *duplicate tuples* are tuples where both values are the same for every attribute
- duplicate tuples are not allowed in relations but IRL it's allowed because it's computational expensive to manage.
- A table is X-total if all it's tuples are X-total
- Constraints
  - NOT NULL means A must be A-total
  -

I'll come back to this

2 Query Languages

2.1 Relational algebra

- A is the set of possible relations
- Partial operations on A take either 1 or 2 relations as input and produce another relation as output
- $\#r$ is the set of attributes of a relation r
- Operations:
  - Attribute selection: $\sigma_{A=B}$ produces a relation where tuples have the same value in attribute A as in B
  - Constant selection: $\sigma_{A=c}$ Same as attribute but against a constant c instead of an attribute
  - Projection: $\pi_{A_1,...A_n}(r)$ Takes a relation r and returns another with only the specified attributes $A_1, ...A_n$
  - Renaming: $\delta_{oldname \mapsto newname}$ Changes the name of an attribute without changing the relation
  - Union: $r_1 \cup r_2$ Relations with the same set of attributes $\#r_1 = \#r_2$ form a relation with tuples from both
  - Difference: $r_1 - r_2$ Relations with the same set of attributes $\#r_1 = \#r_2$ form a relation with the tuples from the first relation r_1 that aren't in the second relation r_2
  - Join (Natural): $r_1 \bowtie r_2$ Joins tuples in both relations where attributes that are in both all have the same values. If there is more than one match then the cross product is given
- Redundant Operations:
  - Intersection: $r_1 \cap r_2$ Returns a relation where tuples are in both relations r_1, r_2 and have the same set of attributes $\#r_1 = \#r_2$
  - Cross-product: $r_1 \times r_2$ where both relation have no common attributes the produced relation is every possible pair of tuples
  - Division: $r_1 \div r_2$ the set of attributes in r_2 must be a superset of r_1. Where there is there is attribute values for every tuple in r_2. It's hard to explain and much easier to show. `http://www.mathcs.emory.edu/~cheung/Courses/377/Syllabus/4-RelAlg/division.html`

2.2 Relational calculus

- Tuple Relational Calculus is where variables represent tuples
- Domain Relation Calculus (DRC) is where variables represent individual values in the domains D_i
- We focus on DRC
- Objects(terms): are placeholders(variables) and values(constants)
- I will come back to this
- $\mathcal{F}$ is the set of all formulae
- Shortcuts:
  - Inequation: $t \neq t' \Leftrightarrow \neg(t = t')$
  - Disjunction: $\psi \vee \phi \Leftrightarrow \neg(\neg\psi \wedge \neg\phi)$
  - Universal quantification: $\forall x(\phi) \Leftrightarrow \neg\exists x(\neg\phi)$
  - Implication: $\phi \Rightarrow \psi \Leftrightarrow \neg\phi \vee \psi$
  - Equivalence: $(\psi \Leftrightarrow \phi) \Leftrightarrow (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$
  - Successive quantification: $\forall x_1(\forall x_2(...(\psi))) \Leftrightarrow \forall x_1, x_2, ...(\psi)$ same with $\exists$
- Free variables are those not bound by a quantifier
- I'll come back to this....right?.....right?
- Domain Relational Calculus
  - Every query Q in the language of DRC $\mathcal{L}_{DRC}$ is expressed in the form $Q = \{(x_1, ..., x_n)|\varphi\}$
  - query mapping on a database is expressed as $\{(listofvariables)|Q\}$
- Safe Range Normal Form (SRNF)
  - Doesn't change the query; just makes it safe
  - Elimination of shortcuts: remove all universal quantification, implication and equivalence
  - Bounded renaming: change names of bound variables so there is non both bound and free
  - Shift Negation: replace sub-formula until negation is only in front of quantification and atoms (down to atoms and remove double negation)
  - Shift disjunction: no ands of ors; use de morgan's
  - Omit parentheses: remove all unnecessary

tell the author to stop procrastinating...

2.3 SQL

not in the test

3 Database design

3.1 Entity-Relationship modelling

not in the test