SOFTENG351 Notes 2017

Theodore Oswandi

June 26, 2017

1 Fundamentals of Database Systems

1.1 General Information

Database large integrated collection of data.
Contains [Entities, Relationships]

DBMS (Database Management System):
software package to store and manage databases

Database System : DBMS with database

DBMS and uses

- store large amounts of information
- code for queries
- protect from inconsistencies and crashes
- security
- concurrent access

1.2 Why Databases

Need to shift from computation to storage of large amounts of information

Accomodate for changes in:
Variety: types of data
Velocity: movement of data
Veracity: uncertainty of data
Volume: amount of data

Structures/Models Need to have a model to describe data, and a schema used to give an abstract description of the data model

1.3 Levels of Abstraction

Views: describe how data seen
Logical Schema: how data structures organised (variable types)
Physical Schema: how files structured
Data Definition Language: How to define database schema
Data Manipulation: how to update values in database
Query Language: used to access data

1.4 Data Independence

Logical Data Independence

- external handling separate from logical organisation
- mappings change, not external schema
- applications only see external schema

Physical Data Independence

- changes to physical schema doesn't affect logical layer
- abstract from DBMS storage organisation
- can perform optimisation/tuning

1.5 Concurrency Control

- many users have to be able to access information at the smae time and make updates without negatively affecting database
- don't want to access disk lots. It is slow and inefficient
- let multiple users access and keep data consistent
- let users feel like they're the only ones using system

2 Relational Model of Data

2.1 General Information

- is logical model of data
- distinguish between data syntax and semantics
- simple and powerful
- sql based off this

2.2 Simple approach

- use tuples to store data
- relations are sets of these tuples
- tables to represent sets of data
- properties (columns) are called attributes
- attributes associated with domains (variable types)

2.3 Relational Schemata

Use of attributes creates relation schema such as:
MOVIE(title: *string*, production_year: *number*)

Relation Schema provide abstract description of tuples in relation

Database Schema is set S of relational schemata. Is basically the set of all tables and their attributes

2.4 Keys

Are used to uniquely identify tuples over all data in a given table.
They are used to restrict number of database instances, to something more realistic and identify objects efficiently

Superkey over relation schema is a subset of attributes that satisfies this uniqueness property
Key is a minimal superkey, is key if no other superkeys exist for R

Foreign Key: is a key used to index values from other values. Used to make reference between relational schemata.
- ensures referential integrity
- no need to copy info from other tables
- need to ensure that [x,y] $\subseteq$ [x,y] and not [y,x] (Order matters)

Example
MOVIE(title: string, production_year: number, director_id: number)
with key [title, production_year]

DIRECTOR(id: number, name: string)
with key [id]

with foreign key: MOVIE[director_id] $\subseteq$ DIRECTOR[id]

2.5 Integrity Constraints

- Db schema should be meaningful, and satisfy all constraints
- should stay true to keys, and foreign keys
- constraints should interact with each other correctly
- should process queries and update efficiently
- should do this and make as few comprimises as possible

3 SQL as Data Definition and Manipulation Language

3.1 General Information

- **S**tructured **E**nglish **QUE**ry **L**anguage
- used as a standardised language in relational Db systems
- is query, data definition, and data management language
- names not case sensitive

3.2 Keywords

CREATE used to create tables. [CREATE TABLE $< tablename >< attributespecs >$]

Attributes : defined as [$< attribute\_name >< domain >$]

NOT NULL : used to ensure that attribute is not null.

DROP TABLE : removes relation schemata from Db system

ALTER TABLE : change existing relation schemata

CHARACTER and **VARCHAR** are fixed or variable length strings. Variable length string are $VARCHAR(32)$

NATIONAL CHARACTER : string characters from other languages

INTEGER and **SMALLINT** used for integers

NUMERIC and **DECIMAL** used for fixed point numbers

FLOAT, REAL and **DOUBLE PRECISION** used for floating point numbers

DATE used for dates (year, month, day)

TIME used for times (hour, minute, second)

3.3 Null and Duplicate Tuples

NULL used to insert tuple where some of the information is not known. This is not good as it can create inconsistencies in information. SQL does this through use of a null marker

Duplicate Tuples are when another tuple exist with same values. NULL can make this confusing.
- these duplicated really expensive to remove
- relations do not contain these tuples

3.4 SQL Semantics

Table Schema set of attributes

Table over R set of partial tuples

These relations are idealised with no duplicates

3.5 Constraints

- can add uniqueness parameters like primary keys on table
- NOT NULL makes sure that partial tuples don't exist
- **UNIQUE** $< attribute\_list >$ ensures that values for attributes exist only once
- **PRIMARY KEY** used on tables that are also UNIQUE, and ensures that NOT NULL applied to it too
- $FOREIGNKEY < attributelist > REFERENCES < tablename >< attributelist >$ used to create foreign key dependencies
- $CONSTRAINT < name >< constraint >$ used to give names to constraints

3.6 Referential Actions

Are used to specify what happens if tuples updated or deleted

SQL provies:

- **CASCADE** forces refercing tuples to be deleted
- **SET NULL** sets referencing tuples to NULL
- **SET DEFAULT** sets the tuple values to specified default
- **NO ACTION** does nothing

Example usage: FOREIGN KEY(title, year) REFERENCES MOVIE(title, year) ON DELETE **CASCADE** will delete director tuple if movie deleted

FOREIGN KEY(title, year) REFERENCES MOVIE(title, year) ON DELETE **SET NULL** will keep tuple in director but won't know what he directed if movie deleted

3.7 CREATE statement

- **CREATE ASSERTION** ¡name¿ CHECK defines integrity constraints
- **CREATE DOMAIN** define domains and check clause
- **CREATE VIEW** $< name >$ **AS** $< query >$ define views

- **CREATE GLOBAL TEMPORARY TABLE** makes a table for current SQL session
- **CREATE LOCAL TEMPORARY TABLE** makes table for current module or session

3.8 Use as Data Manipulation Language

Allow insert, delete or update

- INSERT INTO $< tablename >$ VALUES $< tuple >$
- INSERT INTO $< tablename >< attributelist >$ VALUES $< tuples >$
- INSERT INTO $< tablename >< query >$
- DELETE FROM $< tablename >$ WHERE $< condition >$
- UPDATE $< tablename >$ SET $< value - assignment - list >$ WHERE $< conditions >$

4 Relational Query Languages: Algebra

4.1 Query Languages

- allow access and retrieval of data
- foundation based on logic, and allows for optimisation
- are not programming languages
- not intended for complex claculations
- easy efficient access to lots of data

Query Language must have

- formal language with
- input schema
- output schema
- query mapping

Equivalent Queries

- same input schema
- same output schema
- same query mapping
- if L $\sqsubseteq$ L' and L' $\sqsubseteq$ L

Dominant, if Q1 dominates Q2

- if L $\sqsubseteq$ L'

4.2 Relational Algebra

- useful for representing execution plan
- easily translated to SQL
- A is set of possible relations
- Set of Operations:
  - Attribute Selection σ
    returns value where A in MOVIE is same as B in MOVIE
    $[\sigma_{A=B}(MOVIE)]$
  - Constant Selection σ
    returns vlaue where A is 3 in MOVIE
    $[\sigma_{A=3}(MOVIE)]$
  - Projection π
    returns title and date of MOVIE
    $[\pi_{title,date}(MOVIE)]$

- Renaming $\delta \mapsto$
  renames attribute in MOVIE
  [$\delta_{title \mapsto title'}(MOVIE)$]
- Union $\bigcup$
  takes 2 arguments and returns set of tuples contained in either result, removing duplicates
  [$\sigma_{A=2}(MOVIE) \bigcup \sigma_{A=3}(MOVIE)$]
- Difference $-$
  returns negation
  [$-\sigma_{A=3}(MOVIE)$]
- Natural Join $\bowtie$
  Joins two tables and return tuples that match in both tables
  [$MOVIE \bowtie DIRECTOR$]

Redundant Operations

- Intersection $\bigcap$
  because can use DeMorgans to produce with $-and \bigcup$
- Cross Product $\times$
  because creates $N \times M$ tuples which is not space efficient
- Division $\div$
  because it uses cross product to generate all those that match

$\pi \sigma \bowtie \delta \bigcap \bigcup \bigvee \bigwedge \div \times - \in \mapsto$

4.3 Query Language $\mathcal{L}_{ALG}$ of Relational Algebra

wtf is this

4.4 Incremental Query Formulation

Can use multiple Queries to break up calculation to make it more readable. Ths will also reduce the amount of copying needed as you can use previous results in subsequent calculations.

5 Relational Query Language: Calculus

5.1 General

- let users describe what they want
- simple to write query and translate to SQL
- sometimes relational algebra can get convulated
- based on first order logic
- if safe then can automatically be transformed to SQL style
- **Tuple Relational Calculus TRC** variables represent tuples
- **Domain Relational Calculus DRC** variables represent values in domain

5.2 Object Properties

have variables or may set its value in query variable values must be in its domain can make complex properties by using negation, conjunction and exisential quantification remove brackets if doesn't increase ambiguity

5.3 Shortcuts

- **Inequation** $A \neq B$ equivalent to $\neg A = B$
- **Disjunction** DeMorgan's: $A \vee B$ shortcuts $\neg(\neg A \wedge B)$
- **Universal Quantification** DeMorgan's: $\forall x(A)$ shortcuts $\neg \exists x(\neg A)$

- **Implication** $A \Rightarrow B$ shortcuts $\neg A \vee B$
- **Equivalence** $A \Leftrightarrow B$ shortcuts $(A \Rightarrow B) \wedge (B \Rightarrow A)$
- **Successive Exestential Quantification** $\exists x_1(\exists x_2(A))$ same as $\exists x_1, x_2(A)$
- **Successive Univeral Quantification** $\forall x_1(\forall x_2(A))$ same as $\forall x_1, x_2(A)$

5.4 Free variables

5.5 Domain Relational Calculus

5.6 Tuple Relational Calculus

5.7 Safe Range Normal Form

$\neq \Leftrightarrow \Leftarrow \Rightarrow \forall \exists \neg x \vee \wedge$