Defining a relation scheme in SQL

Wednesday, April 3, 2019 10:52 AM

- SQL the principal language used to describe and manipulate relational DB. (Structed Query Language)
  - aspects { Data-Definition sublanguage; Data-Manipulation sublanguage }
  - SQL { SQL Lite; SQL Server }
- relations in SQL { tables (stored relations); Views relations defined by computation; Temporary Tables constructed when executing queries }
- Data types
  - TEXT Strings
  - CHAR(n) fixed length string up to n characters } padded by blanks if not enough to n.
  - VARCHAR(n)
  - BIT (n) bit string of length n.
  - BIT VARYING(n) bit string of length up to n. B'...' → '0'b'1' X'...' → heximal → 0 u 9 a u f
  - BOOLEAN TRUE, FALSE, UNKNOWN (1) (0) ($\frac{1}{2}$)
    - AND take smallest value
      OR take largest value
  - INT or INTEGER

SHORTINT

- FLOAT or REAL

  DOUBLE PRECISION

  DECIMAL(n, d) n decimal digits, d position from the right

- DATE 'YYYY-MM-DD'

  TIME 'h:m:s', h,m,s are numbers less than 100, two digits

  can have decimal digits.

- Declaration

```
CREAT TABLE Movies(
    title  CHAR(100),
    year   INT,
    genre  CHAR(30)
);
```

- delete a relation DROP TABLE tablename;
- Modify schema

  ALTER TABLE + tablename { ADD attributename(DATA TYPE)
                          { DROP attribute name

- Null value NULL.
- default values name CHAR(1) DEFAULT 'a'

- Default values gender CHAR(1) DEFAULT ;
- Declaring keys { PRIMARY KEY / UNIQUE

  name data-type PRIMARY KEY

  PRIMARY KEY (name)

- select

  SELECT * or tablename attribute name

  FROM table name

  WHERE clause

- < > (!=)

  = (==)

- comparison of strings

  a < b, bbc > bba, b < bb ...

- Pattern Matching

  s LIKE p % P is pattern, s is string

  s NOT LIKE p

  e.g. WHERE title LIKE 'Star ____';
  (with 4 arbitrary characters)

  LIKE '%U%s%';
  (double quote stands for single quote)

strings whatsoever

- ESCAPE

  any string

  s like '%x%%x%' ESCAPE 'x'

  consider as '%'

- delete tuple

  delete from tablename --delete all

  delete from tablename where

  can't delete where values = NULL

- ORDER BY attribute_name (desc); name1, ..., namen

  output ordered by, by default ascend.

  add desc to ascend.

- Above are SQL queries; Below are SQL commands.

  .header on/off -- output with headers

  .mode column/csv/list

  -- column form/ separated by commas/ separated by "|"

  .width number -- output width of columns

  .read filename -- execute file .quit -quit SQL

- Properties of tables

✓ ∪

- not ordered
- flat (no nested attributes) → no table inside table
  - the relation is in first normal form (json not flat)
- do not prescribe how they are implemented/stored in disk.

  (physical data independence) logical definition of data remains unchanged even when we make changes to implementation.
  - Table implementation
    - { row major: an arrays of objects, easier to manage
      - column major: arrays with the same length, each for a column (more efficient)
    - SQLite used row major.

      Memory used column major.

- ~~Flat~~: when want two tables that're related, create two.

  which implies that there's no nested relations.

first normal form (relations that are flat are in first normal form)

- To use a foreign key

```
CREATE TABLE product(
    pname VARCHAR(20) PRIMARY KEY,
    manufacturer VARCHAR(20) REFERENCES Company);
CREATE TABLE Company(
```

cname varchar(20) PRIMARY KEY);

PRAGMA foreign_keys = ON; -- sqlite don't enforce foreign key by default

- Notice that the manufacturer name should always have the same element with cname.