SQL – Data Definition Language

Carla Teixeira Lopes

Bases de Dados
Mestrado Integrado em Engenharia Informática e Computação, FEUP

Agenda

SQL Introduction

Defining a Relation Schema in SQL

Constraints

SQL

Stands for **S**tructured **Q**uery **L**anguage

Pronounced “sequel”

Supported by all major commercial database systems

Standardized – many features over time

Interactive via GUI or prompt, or embedded in programs

Declarative, based on relational algebra

SQL History

1970 “A Relational Model of Data for Large Shared Data Banks” by Edgar Codd

Early **70's** SEQUEL Developed at IBM by Donald Chamberlin e Raymond Boyce

1979 First commercial version by Relational Software (now Oracle)

1986 SQL-86 and SQL-87. Ratified by ANSI and ISO

1989 SQL-891992 SQL-92. Also known as SQL2

1999 SQL:1999. Also known as SQL3. Includes regular expressions, recursive queries, triggers, non-scalar data types and some object-oriented expressions

2003 SQL:2003 XML support and auto-generated values

2006 SQL:2006 XQuery support

2008 SQL:2008

2011 SQL:2011

SQL is a ...

Data Definition Language (DDL)

- Define relational schemata
- Create/alter/delete tables and their attributes

Data Manipulation Language (DML)

- Insert/delete/modify tuples in tables
- Query one or more tables

Standard

Many standards out there

Database management systems implement something similar, but not identical to the standard for SQL

These slides will try to adhere to the standard as much as possible

- Primarily the SQL2 standard and some constructs from the SQL3 standard

Sometimes we'll talk specifically about SQL as understood by SQLite

Agenda

~~SQL Introduction~~

Defining a Relation Schema in SQL

Constraints

Relations in SQL

Tables

- Kind of relation we deal with ordinarily
- Exists in the database and can be modified as well as queried

Views

We'll see this in another class

- Relations defined by computation
- Not stored, constructed when needed

Temporary tables

- Constructed by the SQL processor during query execution and data modification
- Not stored

Data Types in SQL - Text

CHAR(n)

Stores fixed-length string of up to n characters

Normally, shorter strings are padded by trailing blanks to make n characters

VARCHAR(n)

Stores variable-length string of up to n characters

Data Types in SQL – Numeric values

INT (or INTEGER)

For whole numbers

SHORTINT

Denotes whole numbers but the bits permitted may be less (depends on the implementation)

FLOAT (or REAL)

For floating-point numbers

DECIMAL(n,d)

Values that consist of *n* decimal digits, with the decimal point assumed to be *d* positions from the right

Data Types in SQL – Boolean values

Denotes an attribute whose value is logic

Possible values: TRUE, FALSE and UNKNOWN

Data Types in SQL – Dates and Times

DATE

DATE ‘1948-05-14’

TIME

TIME ‘15:00:02.5’

Two and a half seconds past three o’clock

Essentially character strings of a special form

We may coerce dates and times to string types and do the reverse if the string “makes sense” as a data or time

Different implementations may provide different representations

Storage Classes and Data Types in SQLite

NULL

The value is a NULL value.

INTEGER

The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.

REAL

The value is a floating point value, stored as an 8-byte IEEE floating point number.

TEXT

The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).

BLOB

The value is a binary large object of data, stored exactly as it was input.

Boolean Data Type in SQLite

SQLite does not have a separate Boolean storage class

Boolean values are stored as integers

- 0 (false) and 1 (true)

Dates and Times in SQLite

SQLite does not have a storage class for storing dates and/or times

Dates and times can be stored as:

- TEXT as ISO8601 strings ("YYYY-MM-DD HH:MM:SS.SSS").
- REAL as Julian day numbers, the number of days since noon in Greenwich on November 24, 4714 B.C.
- INTEGER as Unix Time, the number of seconds since 1970-01-01 00:00:00 UTC.

SQLite has built-in date and time functions to convert between formats

Type Affinities in SQLite

To maximize compatibility between SQLite and other database engines, SQLite supports the concept of "type affinity”

The type affinity of a column is the recommended type for data stored in that column

Each column has one of the following type affinities:

TEXT, NUMERIC, INTEGER, REAL, BLOB

Determination of Type Affinities in SQLite

If the declared type contains

- “INT”: INTEGER affinity
- “CHAR”, “CLOB”, “TEXT”: TEXT affinity
- “BLOB” or no type specified: BLOB affinity
- “REAL”, “FLOA”, “DOUB”: REAL affinity
- Otherwise: NUMERIC affinity

Rules should be assessed by the above order.

What is the affinity of a CHARINT declared type?

- Integer

Type Affinities in SQLite

Text affinity

- Storage classes: NULL, TEXT or BLOB
- Numerical data is converted into TEXT when inserted into a column with text affinity

Numeric affinity

- All storage classes
- Text data is converted into INTEGER or REAL if conversion is lossless and to TEXT otherwise

Integer affinity

- Similar to the numeric affinity

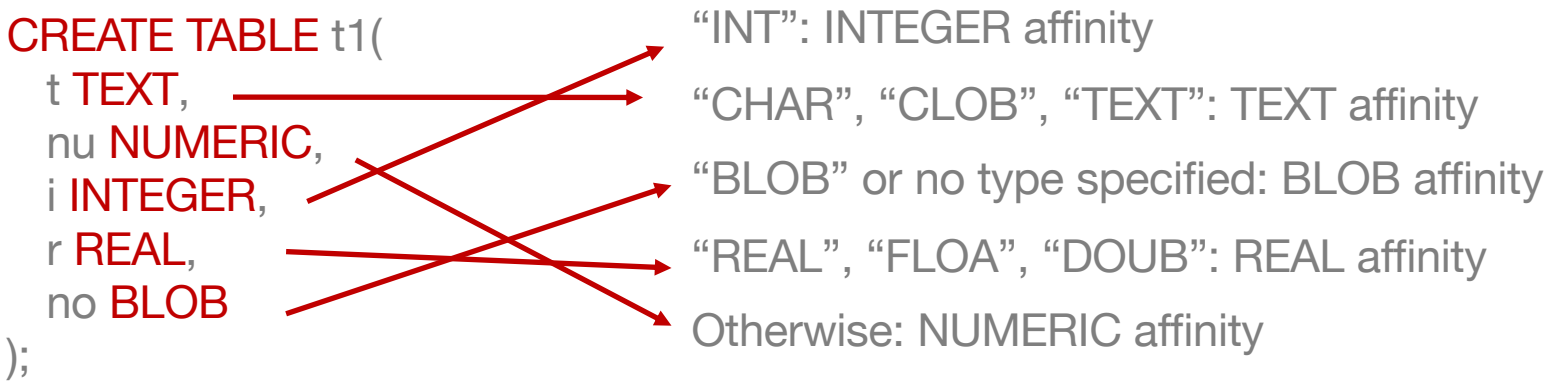
Real affinity

- Similar to numeric affinity but forcing integer values into floating point representation

BLOB affinity

- No attempt to coerce data from one storage class into another

Determination of Type Affinities in SQLite



INSERT INTO t1 VALUES('500.0', '500.0', '500.0', '500.0', '500.0');
text | integer | integer | real | text

INSERT INTO t1 VALUES(500.0, 500.0, 500.0, 500.0, 500.0);
text | integer | integer | real | real

INSERT INTO t1 VALUES(500, 500, 500, 500, 500);
text | integer | integer | real | integer

BLOBs are always stored as BLOBs regardless of column affinity

NULLs are also unaffected by affinity

Simple Table Declarations

```
CREATE TABLE <table_name> (
  <column_name> <data_type>,
  <column_name> <data_type>,
  ...
  <column_name> <data_type>
);
```

Example

```
CREATE TABLE MovieStar (
        id          INTEGER,
        name        CHAR(30),
        address     VARCHAR(255),
        gender      CHAR(1),
        birthdate   DATE
);
```

Modifying Relation Schemas

To remove an entire table and all its tuples

DROP TABLE <table_name>;

To modify the schema of an existing relation

ALTER TABLE <table_name> **ADD** <column_name> <data_type>;

ALTER TABLE <table_name> **DROP** <column_name>;

Default values

For each column we can define its default value

```
CREATE TABLE <table_name> (
<column_name> <data_type> DEFAULT <default_value>,
...
<column_name> <data_type>
);
```

The default default value is NULL

Example

```
CREATE TABLE MovieStar (
        id          INTEGER,
        name        CHAR(30),
        address     VARCHAR(255),
        gender      CHAR(1) DEFAULT '?',
        birthdate   DATE
);

ALTER TABLE MovieStar ADD phone CHAR(16) DEFAULT
'unlisted';
```

Kahoot time!

Any doubts?

Readings

Jeffrey Ullman, Jennifer Widom, A first course in Database Systems 3rd Edition

- Section 2.3 – Defining a Relation Schema in SQL
- Section 2.5 – Constraints on Relations
- Section 7.1 – Keys and Foreign Keys
- Section 7.2 – Constraints on Attributes and Tuples
- Section 7.3 – Modification if Constraints
- Section 7.4 - Assertions