Introduction to Database Systems: CS312 Constraints and Integrity Constraints

Oliver Bonham-Carter

16 Sept 2020

Integrity Constraints

- The CONSTRAINTS enforce conditions to restrict attributes to contain a *correct* type of data while inserting or updating or deleting.
- Integrity constraints provide a mechanism for ensuring that data conforms to guidelines specified by the database administrator.

Integrity Constraints

Common Constraints

- **NOT NULL**: To ensure that no NULL values are allowed
- **DEFAULT**: When none is specified, this constraint provides a default value for a column.
- **UNIQUE**: To ensure that all values of an attribute are different
- **PRIMARY KEY**: Uniquely identifies each row/record in a database table.
  - Ensure that a link exists between two tables.
- **CHECK**: Ensures that all attribute values satisfy specified conditions

Simple NULL constraint demo

Spot the integrity constraint's influence

```
drop table company;
create table company(
   Id text NOT NULL,
   Name text NOT NULL);
```

```
/*Good insert command: complete tuple allowed*/
INSERT INTO company VALUES("COM1","T S LTD.");
SELECT * FROM company;
```

```
/*Good insert command: Empty spaces are allowed*/
INSERT INTO company VALUES("COM1","");
```

```
/*Bad insert command: NULL is not allowed*/
INSERT INTO company VALUES("COM1",NULL);
```

ALLEGHENY COLLEGE

Simple DEFAULT constraint demo

Place predetermined value to a column when no value given

Spot the integrity constraint's influence

```
drop table COMPANY;
CREATE TABLE COMPANY(
   ID INT PRIMARY KEY     NOT NULL,
   NAME TEXT    NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR(50),
   SALARY REAL DEFAULT 50000.00);
```

```
/*Good insert command: complete tuple allowed*/
INSERT INTO COMPANY
VALUES (12, "JAMES", 25, "10, bd Rue du fleur",100000);
```

```
/* Missing entry for SALARY*/
INSERT INTO COMPANY (ID, Name, AGE, ADDRESS)
VALUES (221, "Sherlock", 25, "10, bd Rue du fleur");
```

Simple UNIQUE constraint demo

Prevents two records from having identical values in columns

Spot the integrity constraint's influence

```
/*Create table*/
drop table COMPANY;
CREATE TABLE COMPANY(
   ID INT PRIMARY KEY NOT NULL,
   NAME TEXT NOT NULL,
   AGE INT NOT NULL UNIQUE,
   ADDRESS CHAR(50),
   SALARY REAL DEFAULT 50000.00 );
```

```
/*Good insert command: complete tuple allowed*/
INSERT INTO COMPANY
VALUES  (221, "Sherlock", 25, "10, bd Rue du fleur",100000);
```

Age is not unique

```
SELECT * FROM company;
 /* try to reinsert same values again.*/
```

Simple PRIMARY KEY constraint demo

Only one primary key in a table; UNIQUE Identifiers

Spot the integrity constraint's influence

```
/*Create table*/
drop table COMPANY;
CREATE TABLE COMPANY(
   ID INT PRIMARY KEY NOT NULL,
   NAME TEXT NOT NULL,
   AGE INT NOT NULL ,
   ADDRESS CHAR(50),
   SALARY REAL DEFAULT 50000.00 );
```

```
/*Good insert command: complete tuple allowed*/
INSERT INTO COMPANY
VALUES  (221, "Sherlock", 25, "10, bd Rue du fleur",100000);
```

Key not unique failure

```
SELECT * FROM company;
 /* try to reinsert same values again.*/
```

Simple CHECK constraint demo

Only one primary key in a table; UNIQUE Identifiers

Spot the integrity constraint's influence

```
CREATE TABLE COMPANY(
   ID INT PRIMARY KEY     NOT NULL,
   NAME TEXT NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR(50),
   SALARY REAL CHECK(SALARY > 0));
```

```
/*Good insert command: complete tuple allowed*/
INSERT INTO COMPANY VALUES
 (221, "Sherlock", 25, "10, bd Rue du fleur",100000);
```

CHECK failure

```
INSERT INTO COMPANY VALUES
 (2211, "Sherlock", 25, "10, bd Rue du fleur",-10);
```

Other Types of Constraints

Define variables by data type!

- **INT**: Integer (a finite subset of the integers that is machine dependent).
- **TEXT**: String
- **CHAR(n)**: String of length n (*more below on this*)
- **numeric(p,n)**: Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point. (This allows for number comparisons using operators.)
- **real, double precision**: Floating point and double precision floating point numbers, with machine dependent precision.
- **float(n)**: Floating point number, with user specified precision of at least n digits.

- **NOT NULL**: Ensure that a value is placed here or reject the insertion.

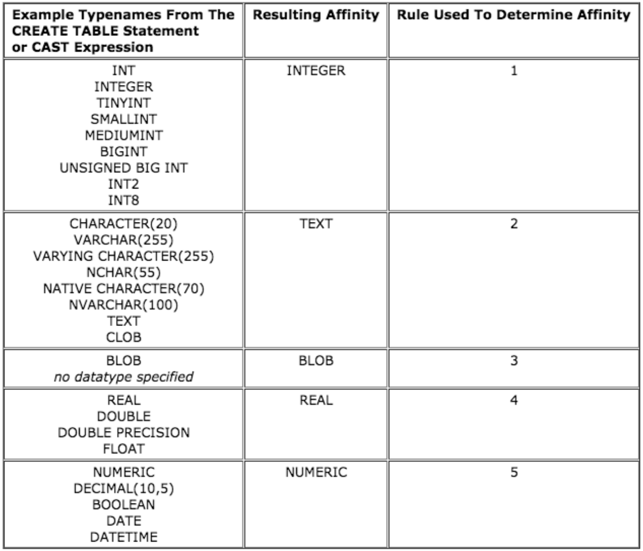
Constraints

Define variables by data type!

- **char(n)**: Fixed length character string, with user-specified length n.
  - Used to store character string value of fixed length
  - The maximum num of chars (not important to SQLite3)
  - About 50 per cent faster than VARCHAR
- **varchar(n)**: Variable length character strings, with user specified maximum length n.
  - Used to store variable length alphanumeric data
  - The maximum num of chars (not important to SQLite3)
  - Slower than CHAR

Affinity Constraints

A small subset of accepted data types that SQLite will accept



Example Typenames From The CREATE TABLE Statement or CAST Expression	Resulting Affinity	Rule Used To Determine Affinity
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8	INTEGER	1

CHARACTER(20)
VARCHAR(255)
VARYING CHARACTER(255)
NCHAR(55)
NATIVE CHARACTER(70)
NVARCHAR(100)
TEXT
CLOB
	TEXT	2

BLOB *no datatype specified*	BLOB	3
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL	4
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC	5

https://www.sqlite.org/datatype3.html#affinity

Adding Constraints to Create

```
CREATE TABLE relationshipTable (
                    A1 D1,
                    A2 D2,
                      ...,
                    An Dn,
                    (integrity-constraint1),
                    ...,
                    (integrity-constraintk));
```

- relationshipTable is the name of the relationship
- Each A_i is an attribute name in the schema of relation relationshipTable
- D_i is the data type of values in the domain of attribute A_i
  - The D_i constrains the particular type of entry

Unique Constraint in the Code

```
/*Two constraints?*/
DROP TABLE instructor;
CREATE TABLE instructor (
   ID CHAR UNIQUE,
   name VARCHAR NOT NULL,
   dept_name VARCHAR,
   salary VARCHAR
 );
```

```
/******PSSST! Now Add some department information ***************************/
insert into instructor  values ('10211', 'Smith', 'Biology', 66000);

/*Any trouble inserting this next line?*/
insert into instructor  values ('10212', null, 'Biology', 66000);
insert into instructor  values ('10211', 'Franklin', 'Biology', 66000);
```

- NULL and repeating UNIQUE values are not inserted

Defining a Table with Primary a Key?

ID is unique, Salary bound by numbers

```
/*Two constraints?*/
DROP TABLE Employee;
CREATE TABLE Employee (
   ID CHAR PRIMARY KEY,
   name VARCHAR NOT NULL,
   dept_name VARCHAR,
   salary NUMERIC(8,2)
 );
```

```
/******PSSST! Now Add some secret information ***************************/
INSERT INTO Employee VALUES("001","Jimmy", "secretService", 1000000);
INSERT INTO Employee VALUES("002","Stevie", "secretService", 1000000);
INSERT INTO Employee VALUES("003","Frankie", "secretService", 10);
INSERT INTO Employee VALUES("004","Robbie", "secretService", 10A);
   /* Oops! Robbie's salary information has a typographical error*/
INSERT INTO Employee VALUES("004","Robbie", "secretService", 100);
INSERT INTO Employee VALUES("004","Jamie", "secretService", 500);
 /* Error: UNIQUE constraint failed: Employee.ID */
 /* Huh?! */
```

Integrity Constraints in CREATE TABLE

- NOT NULL *(but you already knew that!)*
- **Primary Keys**: A primary key is a column or group of columns used to identify the uniqueness of rows in a table.
    - Each table has one and only one primary key.
- **Foreign Keys**: A column (or columns) that references a column (most often the primary key) of another table.
    - The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted.

AgentsDB: Two Tables, One With a Primary Key

Two Tables, One with Primary Key

Super Keys: Using multiple attributes to enforce unique-ness

```
/* Accepts no redundancy */
DROP TABLE Agents1;
CREATE TABLE Agents1
( last_name VARCHAR NOT NULL,
  first_name VARCHAR NOT NULL,
  address VARCHAR,
  CONSTRAINT agents_pk
   PRIMARY KEY (last_name, first_name)
);

/* Accepts redundancy */
DROP TABLE Agents2;
CREATE TABLE Agents2
( last_name VARCHAR NOT NULL,
  first_name VARCHAR NOT NULL,
  address VARCHAR
);
```

Try Your Insert Twice

Let's try

- Insert agent names into both tables.
- Try the same `INSERT` commands again!
- Which commands work?

```
INSERT INTO Agents1 VALUES ("Bond","James","123 abc street");
INSERT INTO Agents2 VALUES ("Bond","James","123 abc street");
```

Forget-Me-Nots

How is the database set up?

- .tables (The tables are of the DB)
- .schema (How the data is stored in the tables)

What data is stored in each table?

- select * from agentsConst;
- select * from agents;
- (note the '*' for the column wildcard)

Is *James* the plural form of *Jame*?

- There can only be **one** “James Bond”
- The name “James Bond” could not be inserted more than once in our base
- Constraints were in place to ensure *distinguishable* rows

Consider this ...

- Can you build a new database table with two (or more) types of constraints?
- For instance, try to alter an earlier database for which you have the *build* file to recreate it (in case anything goes *dreadfully* wrong)