Databases

CS 210: Data Management for Data Science

Constraints

We can add constraints to ensure columns don’t contain null values:

```
CREATE TABLE students (name VARCHAR(20) not null,
                      major CHAR(5) not null);
```

One big table

OrderId	Item	Customer	CustPhone	CustAddress
1	Xmas socks	Bob	123-4567	12 Main St.
2	Reindeer onesie	Bob	123-4567	12 Main St.
3	Toaster	Alice	456-1234	24 2nd St.
4	Bag of coal	Bob	123-4567	12 Main St.

Multiple tables

OrderId	Item	Customer
1	Xmas socks	Bob
2	Reindeer onesie	Bob
3	Toaster	Alice
4	Bag of coal	Bob

Customer	CustPhone	CustAddress
Bob	123-4567	12 Main St.
Alice	456-1234	24 2nd St.

Nobel data

The nobel data set has fields of year, category, and laureates, where laureates has fields:

- id
- first name
- surname
- motivation
- share

Nobel tables

We can create separate tables to reduce duplication:

yearcat

- year
- category

contribution

- motivation

laureates

- first/last name
- share

Keys

To connect the tables, we use keys:

yearcat

- id
- year
- category

contribution

- id
- motivation

laureates

- first/last name
- share
- yearcat id
- motivation id

Keys

The id in each table is a primary key.

- unique value
- not null
- at most one per table

A reference to another table’s primary key is a foreign key.

Primary keys

To specify a primary key:

```
CREATE TABLE yearcat (id SMALLINT AUTO_INCREMENT PRIMARY KEY,
                      year YEAR NOT NULL,
                      category CHAR(10) NOT NULL);
```

Foreign keys

To specify a foreign key:

```
CREATE TABLE laureate (
        fname VARCHAR(80) NOT NULL,
        lname VARCHAR(40),
        share TINYINT NOT NULL,
        year_cat_id SMALLINT NOT NULL,
        motiv_id SMALLINT NOT NULL,
        FOREIGN KEY (year_cat_id) REFERENCES yearcat(id),
        FOREIGN KEY (motiv_id) REFERENCES contribution(id));
```

Inspecting tables

To see the table schema:

```
DESC laureate;
SHOW COLUMNS FROM laureate;
```

For more details on keys:

```
SHOW CREATE TABLE laureate;
```

Connecting from python

First, install `mysql-connector-python`.

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="bob",
  passwd="bobpass",
  database="nobels")

# ... do stuff ...

mydb.close()
```

Connecting from python

```python
import mysql.connector
from mysql.connector import errorcode

try:
    mydb = mysql.connector.connect(...)
except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print("Access denied")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print("No such database")
    else:
        print(err)
```

Cursor

To interact with the database, create a `cursor` object:

```
cursor = mydb.cursor()

# ... do stuff ...

cursor.close()
```

Cursor

Some methods/properties:

- `execute(operation, params)`: run an operation
- `executemany(operation, params)`: run multiple operations
- `fetchall()`: get all rows
- `rowcount`: number of rows affected
- `lastrowid`: value generated by `AUTO_INCREMENT`

Execute

```
stmt = "INSERT INTO students (name, major) VALUES (%s, %s)"
data = ('Bob', 'CS')
cursor.execute(stmt, data)
```

Executemany

```
stmt = "INSERT INTO students (name, major) VALUES (%s, %s)"
data = [ ('Alice', 'CS'),
         ('Bob', 'Math'),
         ('Carol', 'Music') ]
cursor.executemany(stmt, data)
```

Committing

To make any changes persist, we must commit them:

```
mydb.commit()
```

Joins

```
SELECT fname, lname, category
FROM laureate, yearcat
WHERE yearcat.year=2010 AND laureate.year_cat_id = yearcat.id;
```