Python SQL

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Relational Database**  Before discussing how Python integrates with databases via SQL, we will discuss some background database concepts.  **Database** An integrated collection of data (usually relational tables) which model a real world enterprise.  **Relation** A table with rows (database records) and columns (attributes).  **Primary Key** A specially designated unique key used to access a particular database record.   * A Student record is identified by a ABC123 ID. * A Course Offering is identified by a combination of Course Number, Section Number, and Semester. * An Employee record has an ID as its primary key.   **SQL S**tructured **Q**uery **L**anguage is the most popular language for accessing data. It has three categories of language statements:  **Data Definition Language** (DDL)  create/drop tables, indexes, and views  **Data Control Language**  grant/revoke privileges to administer databases  **Data Manipulation Language** (DML)  perform operations (Select, Insert, Delete, Update) on tables and views of tables.  **DBMS Database Management System** is a software package for defining, controlling, and manipulating a database. | Employee Relation (Relational Table)   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Id | Name | NamePrefix | Salary | Dept | | 1111 | Benz, Rob | Dr | 125000 | CS | | 2222 | Nard, Mae |  | 75000 | CS | | 3333 | Benz, Kay | Dr | 180000 | CS | | 4444 | Toast, Melba |  | 50000 | IS | | 5555 | Mecium, Perry |  | 42000 | IS | | 6666 | Clid, U. | Dr | 60000 | MAT |   Id, Name, NamePrefix, Salary, and Dept are column names.  Department Relation (Relational Table)   |  |  |  | | --- | --- | --- | | Dept | DeptTitle | MgrId | | CS | Computer Science | 3333 | | IS | Info Systems | 4444 | | MAT | Mathematics | 6666 | |
| **SQLite**  **SQLite** is a very simple database management system which has several advantages:   * Easy to install (see at right). * Simple code library and database file which doesn't require setting up a database server. * Python provides a module for connecting with it. * Provides an interactive shell window which can be used to execute DDL and DML SQL statements. | You can download a copy of SQLite at [www.sqlite.org/download.html](http://www.sqlite.org/download.html)  Select software labeled with "shell" for your OS.  You can use the SQLIte shell by invoking the sqlite3 executable (sqlite3.exe on windows).  Some important commands within the SQLite shell:  .open *fileName*.db  opens the specified database file  .exit  exit the SQLite shell.  .read *fileName*.sql  reads the specified file containing SQL statements |
| **SQL statements**  The SQL statements can be executed from the SQLite shell or within Python. We will see how to execute SQL in Python shortly.  Create a table:  create table *tableName* (*columnList*);  The *columnList* is a list of column names separated by commas. If a single column is the primary key, it is followed by the PRIMARY KEY phrase. If there are multiple columns in the primary key, specify an extra clause PRIMARY KEY (*primaryColumns*) after listing all the columns in the table.  Insert data into a table:  insert into *tableName* (*columnList*)  values (*valueList*); | Create the tables for our examples:  create table Employee (Id PRIMARY KEY, Name, NamePrefix,  Salary, Dept);  create table Department(Dept PRIMARY KEY, DeptTitle, MgrId);  Populate those tables with data:  insert into Employee (Id, Name, NamePrefix, Salary, Dept)  values('1111', 'Benz, Rob', 'Dr', 125000, 'CS' );  insert into Employee (Id, Name, Salary, Dept)  values('2222', 'Nard, Mae', 75000, 'CS' );  insert into Employee (Id, Name, NamePrefix, Salary, Dept)  values('3333', 'Benz, Kay', 'Dr', 180000, 'CS' );  insert into Employee (Id, Name, Salary, Dept)  values('4444', 'Toast, Melba', 50000, 'IS' );  insert into Employee (Id, Name, Salary, Dept)  values('5555', 'Mecium, Perry', 42000, 'IS' );  insert into Employee (Id, Name, NamePrefix, Salary, Dept)  values('6666', 'Clid, U.', 'Dr', 60000, 'MAT' );  insert into Department (Dept, DeptTitle, MgrId)  values('CS', 'Computer Science', '3333');  insert into Department (Dept, DeptTitle, MgrId)  values('IS', 'Info Systems', '4444');  insert into Department (Dept, DeptTitle, MgrId)  values('MAT', 'Mathematics', '6666'); |
| **SQL statements - continued**  Deleting (dropping) an entire table:  drop table *tableName*;  Deleting rows in a table:  delete from *tableName*;  delete from *tableName* where *condition*;  Updating rows in a table:  update *tableName* set *varValueAssignments*;  update *tableName* set *varValueAssignments*  where *condition*; | drop table Employee; -- deletes the entire table  delete from Employee; -- deletes all rows in the Employee table  delete from Employee where Dept = 'CS'; -- deletes all emp in CS  -- give everyone a 10% raise  update Employee set salary = salary \* 1.10; |
| **SQL statements - continued**  Selecting particular columns of each row in a table:  select *columnList* from *tableName*;  Selecting every column of each row in a table:  select *\** from *tableName*;  Selecting rows conditionally:  select *columnList* from *tableName*  where *condition*;  Joining two tables to get a list of columns from either/both tables:  select *columnList* from *table1, table2*  where *table1.column1 = table2.column2*;  If joining by multiple columns, additionally specify:  and *table1.column3 = table2.column4*; | sqlite> select Id, Name from Employee;  1111|Benz, Rob  2222|Nard, Mae  3333|Benz, Kay  4444|Toast, Melba  5555|Mecium, Perry  6666|Clid, U.  sqlite> select \* from Employee;  1111|Benz, Rob|Dr|125000|CS  2222|Nard, Mae||75000|CS  3333|Benz, Kay|Dr|180000|CS  4444|Toast, Melba||50000|IS  5555|Mecium, Perry||42000|IS  6666|Clid, U.|Dr|60000|MAT  sqlite> select Id, Name from Employee  ...> where Dept = 'CS';  1111|Benz, Rob  2222|Nard, Mae  3333|Benz, Kay  sqlite> select Employee.Id, Department.DeptTitle  ...> from Employee, Department  ...> where Employee.Dept = Department.Dept;  1111|Computer Science  2222|Computer Science  3333|Computer Science  4444|Info Systems  5555|Info Systems  6666|Mathematics  sqlite> |
| **Python using sqlite3**  **import sqlite3**  imports the sqlite3 module  **db = sqlite3.connect**(*dbname.*db)  connects to the specified database file  **cur = db.cursor()**  creates a Python database cursor for accessing and  manipulating data  **cur.execute(*sqlStatement*)**  executes the specified SQL statement, passed as a string without a semicolon. For select statements, it prepares for execution an does the execution with the fetchone or fetchall methods.  ***resList* = cur.fetchall()**  executes the specified SQL statement and returns the resulting list of tuples.  ***resOne* = cur.fetchone()**  fetches the next row resulting from executing a select statement, returning either a tuple or None.  **cur.close()**  closes the cursor  **db.commit()**  commits insert, update, and delete changes to the database.  **db.rollback()**  rollbacks any changes since the last commit or connect to the database.  **db.close()**  closes the database connection. If you db.close() without doing a db.commit(), it will implicitly do a db.rollback(). | # dbRaise.py  # Program to give each employee in CS dept a 10% raise  import sqlite3  import pprint  db = sqlite3.connect("Employee.db")  cur = db.cursor()  # Execute a query to show all employees  cur.execute("select Id, Name, Salary from Employee")  resList = cur.fetchall()  pprint.pprint(resList)  # Give each employee in CS dept a 10% raise  cur.execute("update Employee set salary = salary \* 1.10 " +  " where Dept = 'CS'")  cur.close()  # Show the resulting employees  cur = db.cursor()  # Execute a query to show all employees  cur.execute("select Id, Name, Salary from Employee")  resList = cur.fetchall()  print("Resulting employees")  pprint.pprint(resList)  # commit the changes  db.commit()  # close the connection  db.close()  **[('1111', 'Benz, Rob', 125000),**  **('2222', 'Nard, Mae', 75000),**  **('3333', 'Benz, Kay', 180000),**  **('4444', 'Toast, Melba', 50000),**  **('5555', 'Mecium, Perry', 42000),**  **('6666', 'Clid, U.', 60000)]**  **Resulting employees**  **[('1111', 'Benz, Rob', 137500.0),**  **('2222', 'Nard, Mae', 82500.0),**  **('3333', 'Benz, Kay', 198000.00000000003),**  **('4444', 'Toast, Melba', 50000),**  **('5555', 'Mecium, Perry', 42000),**  **('6666', 'Clid, U.', 60000)]** |