RDBMS & SQL

Documentation: Google Sqlite and Python Library Reference

What is a DBMS?

- An organized store of data
  - Why not use files?
  - Why not just shelve?

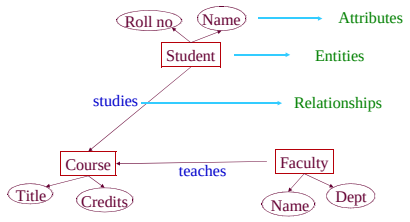
Facilities Provided

- Fast access – indices
- Recover from failures
- Protect from unauthorized access
- Multiple users
- Transaction concept – ACID properties

Entity Relationship Model

- Any data that we want to store on a computer can be modelled in terms of entities and relationships between those entities
- Examples
  - Students, courses, faculty
  - Company, employees, staff, managers, customers
  - People, addresses
  - Songs, titles, artists, composers
  - Bank accounts, customers

E-R Diagram



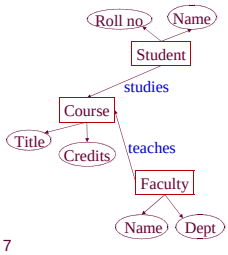
Conversion to Tables

- Students, courses, faculty
- Possibilities:
  - One table with columns:
    - student, course, faculty
  - Two tables
    - student, course
    - course, faculty
  - Several tables
    - For each course, a table containing all students of that course
    - Course, faculty

E-R Diagram to Tables

- One table for each entity, with attributes as columns – entity tables
- One table for each relationship, with pointers to rows in entity tables as columns – relationship tables

Example



- Student
  - Roll-no, Name
- Course
  - Title, Credits
- Faculty
  - Name, Dept
- Studies
  - Student_id, Course_id
- Teaches
  - Faculty_id, Course_id

Designing Databases

- Normalization
- Several (may be 100) normal forms
- 5 normal forms – covered intuitively
- Covered more formally in DB course
- 3 normal forms – most popular

First Normal Form (1NF)

- First normal form – The table must be flat.
  - All rows must contain the same number of columns.
  - No column can contain repeating elements.

1NF – Number of Columns

Title	Field1	Field2	Field3
Book1	Author1		
Book2	Author2		
Book3	Author31	Author32	Author33
Book4	Author4		
Book5	Author51	Author52	

1NF – Repeating Elements

Title	Field1
Book1	Author1
Book2	Author2
Book3	Author31, Author32, Author33
Book4	Author4
Book5	Author51, Author52

1NF – Compliant

Title	Author
Book1	Author1
Book2	Author2
Book3	Author31
Book3	Author32
Book3	Author33
Book4	Author4
Book5	Author51
Book5	Author52

Second Normal Form (2NF)

- All non-key columns must be dependent on the whole key and not on just one attribute of the key
- Implied requirements
  - Key column established
  - All non-key columns describe the key column entry

2NF – Key and Non-key Columns

Title	Author	Author_address
Book1	Author1	City1, …
Book2	Author2	City2, …
Book3	Author31	City1, …
Book3	Author32	City3, …
Book3	Author33	City3, …
Book4	Author4	City4, …
Book5	Author51	City2, …
Book5	Author52	City5, …

2NF – Compliant: Choices

Author	Address
Author1	City1, …
Author2	City2, …
Author31	City1, …
Author32	City3, …
Author33	City3, …
Author4	City4, …
Author51	City2, …
Author52	City5

Author	Title
Author1	Book1
Author2	Book2
Author31	Book3
Author32	Book3
Author33	Book3
Author4	Book4
Author51	Book5
Author52	Book5

Third Normal Form (3NF)

- No non-key attribute can be dependent on another non-key attribute
- BCNF: No attribute can be dependent on a non-key attribute
- A table is meant to maintain relationship between non-key columns and the key column

3NF Violation Examples

- PINCODE based on City of author address
- Total price of order entry based on quantity and unit price column (calculated value)
- Solutions
  - Have a separate table relating city and pincode
  - Databases support calculated columns in forms and reports

Fifth Normal Form (5NF)

- Any table that has been divided into multiple tables must be capable of being reconstructed to its exact original structure by one or more joins.

Review

- 1NF = Flat, Column Count Equal, No Repeating Elements in a Column
- 2NF = Non-key columns related to key columns
- 3NF = No interdependent non-key columns
- BCNF
- 5NF = Reconstruction must be possible

SQL Practice

SQL Summary: Create/Modify Databases

- create database university;
- use university;
- create table students (rollno integer, name char(30), bday date);
- insert into students (rollno, name) values (200301001, 'Ram');
- alter table students add column btime time;
- update students set bday=19831002, btime=213000 where rollno=200301001;
- delete from students where rollno>200301001;
- drop table students;

SQL Summary: Query Databases

- show tables;
- show columns from students;
- select * from students;
- select rollno, name from students where bday >= 19831001;
- select courses.name from students, courses, studies where students.name = 'Ram' and students.rollno = studies.rollno and studies.courseid = courses.courseid;
- select avg(bday) from students;
  # also: sum, max, min, count, count(distinct)
- select avg(bday) from students group by rollno % 3;
- select * from students order by rollno;
- select avg(bday) from students group by rollno % 3 having count(rollno % 3) > 10;

Problems

students (rollno, name, email, bday)
courses (id, name)
studies (rollno, courseid, marks)
faculty (id, name, dept)
teaches (facultyid, courseid)

- Display rollnos and emails of all students.
- Display rollnos and emails of students born between 1st Aug 1980 and 21st Jan 1983.
- Display rollnos and emails of all students sorted by their names.
- Display rollnos of students who study "maths".
- Display rollnos of students who study some subject under "Dr.Shankar".
- Display average marks of students for each subject.
- Display average marks of students for each subject which has atleast 30 students studying it.

Python DB API

```
conn = db.connect('user=x dbname=y')
curs = conn.cursor() # get cursor object
conn.commit()
conn.close()
curs.execute('sql string')
curs.fetchone() # returns sequence (e.g. tuple)
curs.fetchall() # returns seq of seqs
curs.dictfetchone()
curs.dictfetchall()
```

Example

```
>>> import MySQLdb
>>> conn=MySQLdb.connect('user=x dbname=y')
>>> curs = conn.cursor( )
>>> curs.execute('select * from messages')
>>> curs.fetchall( )
[ ]
>>> #for i in curs: print i
```