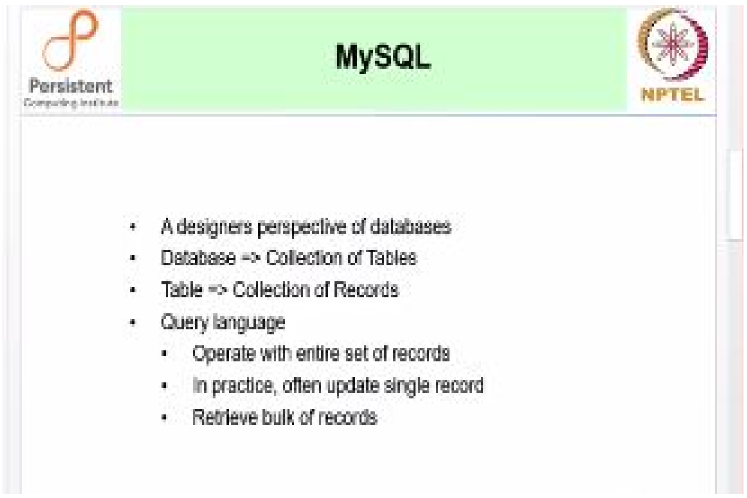
Introduction to Modern Application Development

Prof. Aamod Sane
FLAME University and Persistent Computing Institute
Indian Institute of Technology – Madras

Lecture – 27
Week 09 – Session 1

(Refer Slide Time: 00:16)

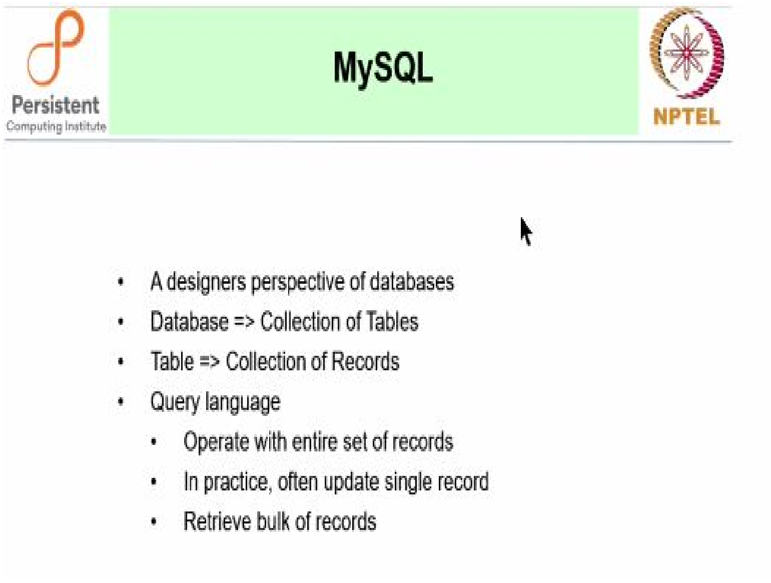


MySQL

- A designers perspective of databases
- Database => Collection of Tables
- Table => Collection of Records
- Query language
  - Operate with entire set of records
  - In practice, often update single record
  - Retrieve bulk of records

Welcome to week 9 of Introduction to modern application development. This week, we will start assembling all the pieces that we have learned up to the last time, before which we learn about MySQL. MySQL is a database management system(relational). We will take a designers perspective while learning about databases.

(Refer Slide Time: 01:07)



MySQL

- A designers perspective of databases
- Database => Collection of Tables
- Table => Collection of Records
- Query language
  - Operate with entire set of records
  - In practice, often update single record
  - Retrieve bulk of records

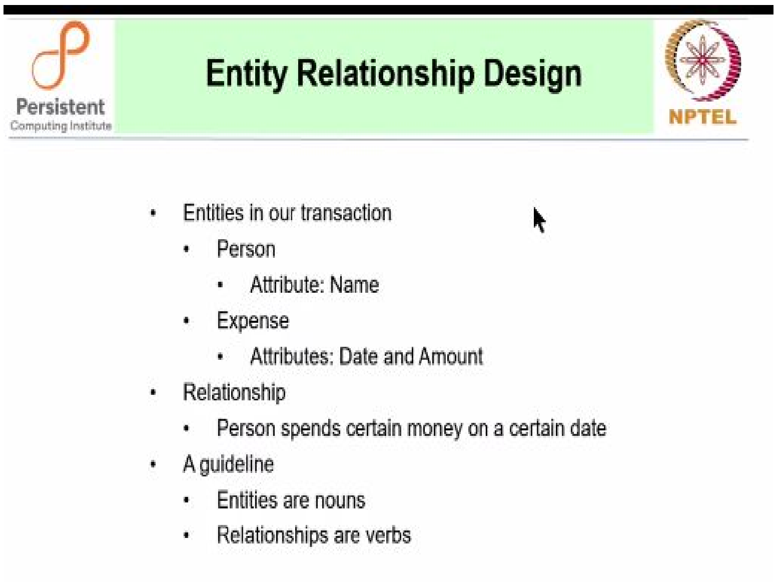
A database is a collection of tables.

In essence a table is no different from a file, with thee following distinction:

1. a file is just a collection of bytes, while a database is analogous to a directory and tables are analogous to files.
2. files can contain just any odd byte that you want to put in, tables have structure, they are collections of records.
3. Just like files are made of lines; lines are made of words, databases are made of structures (example: "struct" in C) with attributes/member variables and their values which get stored in an easily accessible fashion.
4. An odd thing about databases is the language that is used to access them. The query language is a way of operating on the entire set of records at the same time rather than one record at a time. Though updates often go to a single record but retrievals are usually in bulk.

We will only cover those parts of that language that are needed in this course. It is far too big a topic to address in any useful fashion.

(Refer Slide Time: 03:07)



Entity Relationship Design

- Entities in our transaction
  - Person
    - Attribute: Name
  - Expense
    - Attributes: Date and Amount
- Relationship
  - Person spends certain money on a certain date
- A guideline
  - Entities are nouns
  - Relationships are verbs

Design tables and records: Entity Relationship Diagram

Many information processing problems can be thought of as made of two kinds of things:

1. Entities which roughly correspond to nouns.

2. Relationships which correspond to verbs, in ordinary English.

To build a fairshare like application:

1. Entities are usually people, the group of friends, etc and the attribute is their names.
2. The second entity is their expenses; the amounts that they spend and so an expense has two primary features that distinguish it, which is when did it happen and for what amount did it happen.
3. LINK BETWEEN THE ENTITIES: The action of spending. A person is related to a particular expense because "that person spends a certain amount of money on a certain date".

NOTE:

- This above link/ relationship is not what is called a relationship in a database, it is one of the odd terminological things one has to get over with.
- Entities are represented as tables, though sometimes tables can also represent relationships.

(Refer Slide Time: 05:18)

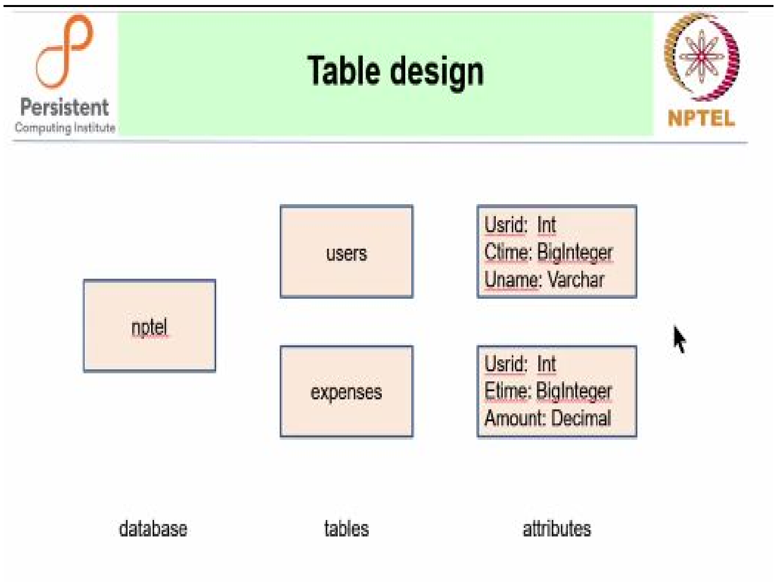


Table design

<u>Database : nptel</u>

<u>Tables:</u>

Users : The users of a fairshare like application (i.e.an entity)

Expenses: Tracks monthly expenditure.

Attributes.

user ID : attribute of a user entity; unique to each user.

Ctime : the time at which a user was created.

Etime: the time at which the expense was created.

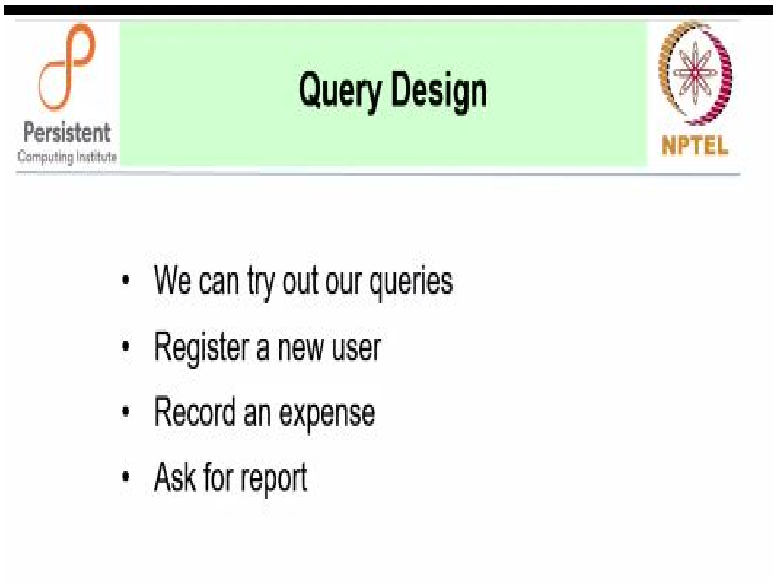
Uname : the name of the user

Amount : The cost.

Observe the type declaration used for attributes in the above figure.

The nptel database is analogous to a directory, “users” is a file and “expense” is a file. The contents of users are basically values of “usrId, Ctime, Uname” and the contents of the expenses table are values,”usrID, Etime and Amount”.

(Refer Slide Time: 06:58)



Query Design

- We can try out our queries
- Register a new user
- Record an expense
- Ask for report

Querying a database:

1. register new user,
2. record an expense
3. ask for a report the way we had planned.

By default, MySQL has a login feature. Users can share access with other users. A similar structure exists inside a database where a database can have certain users and each user, can decide who can access which database, who can access which table, what they can do with their table and so on. So by default, MySQL (here, MySQL in xampp) has a user called root and there is no password for that user (by default), which is pretty dangerous in real databases.The root is the all powerful user, one needs to create a strong password for the root. Since this is just a demo, we are not setting a password for root.

- Login as root: “mysql -u root -p”
- Commands for looking at databases and tables: “show databases” for displaying the names of all the databases created; “create database nptel3” or creating new databases with name of database nptel3;”create user ‘n1’@’localhost’ identified by p1”, creates a user n1 with password p1.
- Where did this creation of the user go? Users are created and stored in the database which itself is called MySQL. (So user information querying commands will be similar to database querying commands).
- To look at any database: “use dbname” command(replace dbname with name of database, say, use nptel3)
- In a database, command “show tables” displays tables.

The database we actually installed in xampp is an open source version of MySQL which Oracle bought. The new open source version is called MariaDB. Nonetheless, you should just think of it as MySQL. So, users are created inside a database which is itself called MySQL and

What are tables made of?

- A table is described by a schema which is like a class or a record.
- For describing the schema of a table tbname:“describe tbname”
- Output: Fields, type of each field, can the field be null, default value, etc.

<u>Users Table in MySQL database:</u>

1. Type the following command: “describe user;”
2. Among the other fields, “host”,”user”and “password” are the fields of interest.
3. “select host,user,password from user” command, to display a table of hosts, users and passwords.If you want to be explicit, you can say mysql.user, makes it a little easier to read without having to rely on the “use” command that we created earlier.
4. The passwords displayed in the table are encrypted versions of the password: the password is in plain text, but in the database,it is stored in this fashion. In some of the

higher security systems, the password is often actually shared in pieces between two or three different users so that they all have to be there before the password becomes accessible.

5. In all the entries of the user, it is attached to a host, so as far as the MySQL database is concerned, only from the localhost, one can interact with the database.

NOTE: In reality, databases usually sit by themselves on a separate machine. The machines can be clients that are explicitly declared. So when web servers that are part of the application contact these clients, they also use the databases.

ADMINISTRATIVE USER:

For a particular application; we can create a database (such as the database nptel), create multiple users, we can distinguish between a user of the database whereas somebody who is an administrative user of the database. Typically the administrator logs in and sets up the core structures of the database whereas an ordinary user can often be a servlet which contacts the database in order to deal with the content of the tables.

A user, by default, when you create it, in MySQL is unrestricted in many ways. They are not as powerful as the root, but in general a user by default. Restricting access for a user can be done.

For example, if a servlet program is contacting a database. The servlet has to be able to access certain fields but may not be able to update certain information in the database.Like permissions in an Android phone etc to do explicit things like read your contacts or be able to read photos and so on, there is a distinction to be made among what you can do to a table in a database and a "grant" statement is used to do this.

Example:

1. Create users n1 and n2.(Some commands :"drop user", "delete user")
2. Login as n1.
3. Execute"show grants 'n1'@'localhost'". Outputs: GRANT USAGE. This basically means all you can do is look at the existing databases. It allows them to work with a

special database called “test”, the only database that a default user can use.

(MySQL by default the command line does not show you which user you are logged in as and currently we are logged in as the user: root)

PRO TIP: Login as root, do whatever it is you want to do as root and immediately log out,for security reasons.

To display current user in prompt: “prompt MariaDB \u[\d]>”.

To make this change permanent , we can update it in config files, etc., but a simpler method of setting this change is by using environment variables instead.

set “MySQL_PS1= MariaDB \u [\d]>”

On Unix the syntax might be different, but the idea is to use an environment variable to remember the command properly.

Giving Grants to Users:

1. “grant all on nptel3.*” to ‘n1’@’localhost’;
2. show grants for ‘n1’@’localhost’;
3. use nptel3.*;
4. create table xyz(f1 int primary key);
5. show tables;
6. describe xyz;

The above exercise demonstrates the effect of “grant” command.

- ❏ Line1 requests user n1 to be granted all privileges of nptel3.
- ❏ Line2 requests to display grants for n1 (The output says that all privileges of nptel3 are granted).
- ❏ Line3 requests to access database nptel3.
- ❏ Line4 tries out one of the privileges granted, i.e. permission to create new tables to the existing database nptel3. This table has a field f1 of integer data type. Primary key field uniquely identifies each record in the xyz table.(Output: table xyz created successfully)
- ❏ Line5 displays the tables currently part of the database nptel3. We can see that the

table xyz is listed.

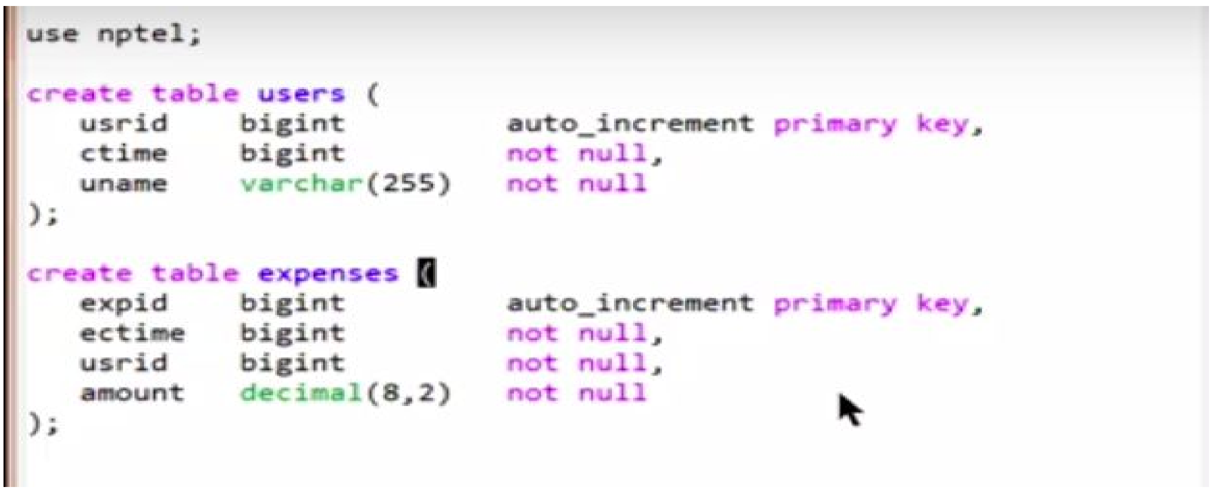
❑ Line6 requests to display the fields of the table.

That is the basic functionality of this grant function

SELECTIVE GRANT:

1. grant select, insert on nptel.* to ‘n2’@’localhost’
2. show grants for ‘n2’@’localhost’
3. Login back as: mysql -u -n2 p2
4. use nptel;
5. create table xyz;

This table is not created as n2 doesn’t have privileges to create tables.



```
use nptel;

create table users (
    usrid     bigint          auto_increment primary key,
    ctime     bigint          not null,
    uname     varchar(255)    not null
);

create table expenses (
    expid     bigint          auto_increment primary key,
    ectime    bigint          not null,
    usrid     bigint          not null,
    amount    decimal(8,2)    not null
);
```

To design a database for fair-share applications, we create tables using the above commands in database ‘nptel’.

Note:

1. `bigInt` is 8 bytes, so 64 bits
2. `auto-increment`: This means that every time you insert a new record/row to the table, automatically generate a value for this field, which is one larger than the previous row’s value of this field.
3. uname is a 255 character username
4. Expense has the ID of the expense, the time when it was created, which user created it and how much was the amount.

The pair of creation time and user ID describes the user. But user ID alone can be used alone in this scenario for practical purposes as auto increment will ensure each userID is unique, but there are situations where auto-increment may not be possible to implement. Example: user IDs may come from a completely different data set.

Populating the tables:

```
select * from xyz;
```

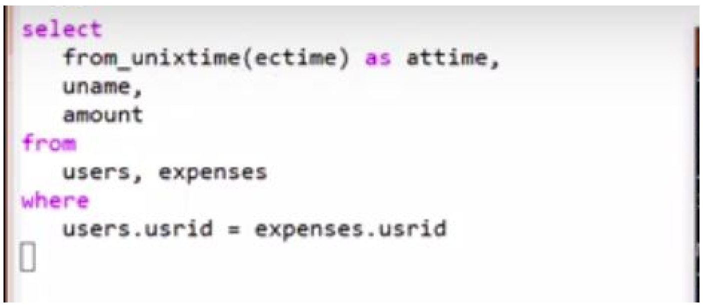
Displays the contents(rows/records of the table xyz). The table is empty right now as we have not added any rows yet.

```
insert into users values ( 0,123,'f1');
```

This populates users table with fields for the row :userID = 0, ctime=123, uname= f1.

REMARK : This syntax is called SQL syntax.

Querying the database:



```
select
   from_unixtime(ectime) as attime,
   uname,
   amount
from
   users, expenses
where
   users.usrid = expenses.usrid
```

Other than maintaining the table, this structured style of writing helps to fetch specific data from multiple tables using queries. The above figure is an example of a query.

Note:

A. The time we are storing in the above figure is just an integer. This number is a Unix timestamp. A unix timestamp is the number of milliseconds from January 1970 (

which is an arbitrarily chosen starting point), but it is a convention now.

B. Databases have native examples of date and time. Expert database users can sometimes prefer to have the database supported types rather than external types

C. The above example is a query like this since what we normally want is not just the expenses and the users information.

D. WHAT DOES THE QUERY REQUEST?

   1) This means pick rows in which the user ID in the users table and the user ID in the expenses table is equal.
   2) Return "join" of users and expenses that satisfies 1. (combine each row of the users table with each row of expense table. "jon" comes from joining the tables.)
   3) Take the field values of expense creation time of the joined table -> convert it into a normal timestamp from Unix time(call it 'attime')-> Output the converted field values of time in a column and two other columns of joined table, uname and amount.

Essence of the query: Which user was responsible for which amount at which time (user name used rather than the ID of the user to identify the user in the table returned by the query).

The final web app will send queries like this, from the server to this database management system and get the results.

In this session we learnt how to manage login and maintain databases, to create the tables and how it is that we intend to use it for our specific application. **(Video Ends: 38:26)**