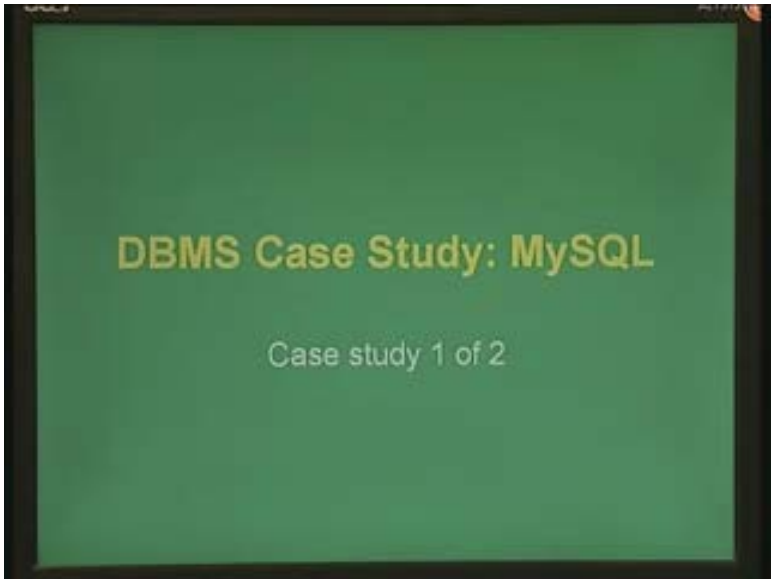
Database Management System
Dr. S. Srinath
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Lecture No. # 32

Case Study: MYSQL

Hello and welcome to another session in database management systems. In this lecture today, we will be looking at a case study of a real world DBMS. We have learned so many concepts in database management systems like especially the relational database, I mean how relational tables are stored, what is the relational algebra and what is the query processing engine do and what are the different kinds of indexing methods and so on and so forth. Let us see how it is actually implemented in some real world context.

In this lecture today, we are taking up a pretty interesting real world database system namely MySQL database system.

(Refer Slide Time: 01:24)



What is very interesting about this database system is that it is a phenomenon or it is an outcome of a very remarkable phenomenon of recent times namely the open source and the free software phenomenon. If it were not for the open source or the free software phenomenon, I would contain that several different innovations that we see in the realm of algorithms and software design would not have been possible. Today it is possible for anyone around the world to be able to freely download this database here which we are going to, this DBMS here which we are going to see today. And you can download it for free of charge that is it's not only you can download it for free for charge, you can also have access to the source code that made up the MySQL database and you can make your own changes to the source code of the DBMS.

And it's completely free in that sense, free as in freedom like which is generally used, which is the term generally used by the free software foundation people who allocate free software around the world.

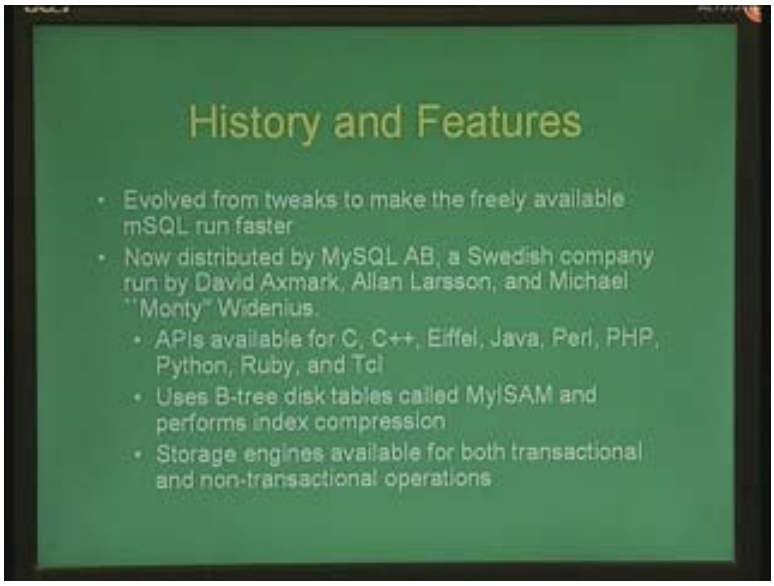
(Refer Slide Time: 03:37)

So let us look at MySQL in more detail. MySQL is a very popular open source database system and it is licensed under what is called as GPL or the GNU public license. Of course looking in to the inter cases of GPL is definitely not within the scope of our lecture here but essentially the gist of GPL essentially says that the software provided is free in the sense that you are free as in freedom that you are free to make any modifications to the source code. You can free to redistribute the source code, you can free to resell the source code if you wish to however you need to also give the same freedom to whoever is going to next use your software and so on.

And there are different varieties of GPL and you should check up what exactly is the specific variety of GPL when you download MySQL. And it's a widely used database management system and it is used in the wide variety of systems like embedded systems to large scale information systems and it's largely written in C and C plus plus and its source code is available like I said and it has been ported to many different operating systems and operating platforms. And the website on which it is available is shown here in the slide namely www.mysql.org.

Let us have a brief look at the history of MySQL, how it came in to being and what are its different features. MySQL is a relatively recent phenomenon in database management systems in fact the genesis of MySQL stems from another small SQL engine called mSQL, small m and SQL as shown in the slide here.

(Refer Slide Time: 04:49)



mSQL or I guess it stood for mini SQL was a simple SQL engine that allowed users to write SQL queries and maintain very simple databases with a small set of tables here. Now some set of people in Sweden actually who try to tweak mSQL, mSQL also was free software and they try to tweak mSQL and out of this endeavor arose a completely new set of database system which is now called MySQL.

Now MySQL has now, the people who formed MySQL has now formed their own company MySQL AB which is a Swedish company and run essentially by the three people who formed this database system in the first phase. And of course there are also commercial versions of MySQL which are sold for a price and which come with lot more support from the MySQL team who can actually install and tune your database system and so on and so forth. But for the free version of the software, there is no support and there is no warranty as such.

MySQL has several different features, there are several APIs or application programming interface that are available for many languages like C, C plus plus, Eiffel, Java, Perl, PHP and so on and so forth. This APIs allow programs written in these languages to directly send SQL commands to MySQL. So you can embed your database system or rather your database client within another application program. It uses B tree based disk tables and MySQL team themselves developed new storage structure is called MyISAM and also performs index compression, so that the shadow of the index on the disk is quite small.

And there are storage engines that are available which support both transactional and non-transactional operations. Remember what is a requirement of a transactional operation essentially the acid semantics, that it has to provide atomicity and isolation and consistency in durability and so on. Durability also in a sense implies recovery that is once a commit has happened, you should be able to, it should be reflected in the database even if there is a crash later on and so on.

MySQL is a multi-threaded engine and if your operating system supports threads at the kernel level, MySQL uses that and if your operating system and your hardware of course support multiple processors, MySQL automatically uses multiple processors.

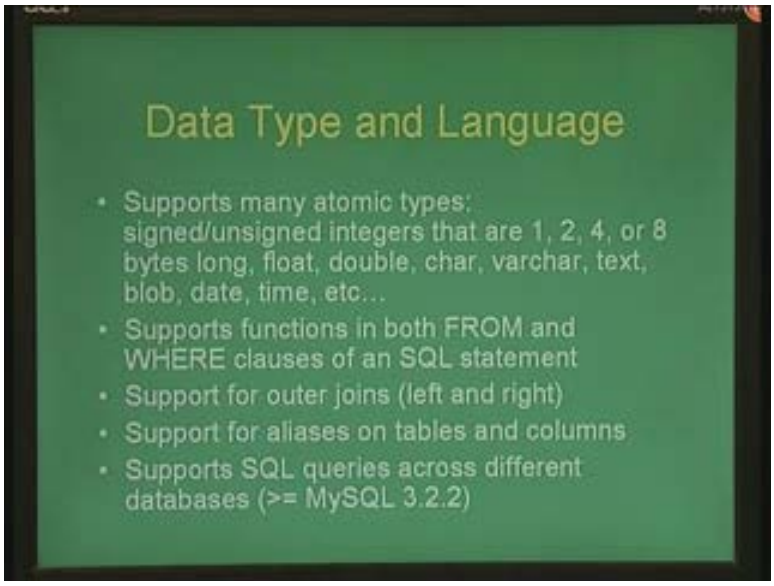
(Refer Slide Time: 08:44)

So it is scalable to many processors on a multi-processor machine and is aware of the fact that there are many processors that are running and you can utilize them. The kind of memory allocation system that it uses is a thread based memory allocation system rather than a process based where each thread which is a light weight process manages its own memory. It also has other nice features like fast joins which is computed by an optimized algorithm which just uses one pass operation over the tables when performing the joins.

We saw several different kinds of join techniques merge join, hash join and so on, so which essentially forms the gist of the kind of algorithms that go into these techniques here that computes fast joints. And whenever temporary tables are required, say virtual tables or tables as a result of a query in a nested SQL query and so on, they are implemented using in memory hash tables and they are extremely fast to access and so on.

MySQL server has these double features that it can run as a standalone server as well as it can run as an embedded server that is server that is actually embedded within some other application. So your application actually runs as the server and it looks as though your application is supporting database system in itself.

(Refer Slide Time: 11:27)

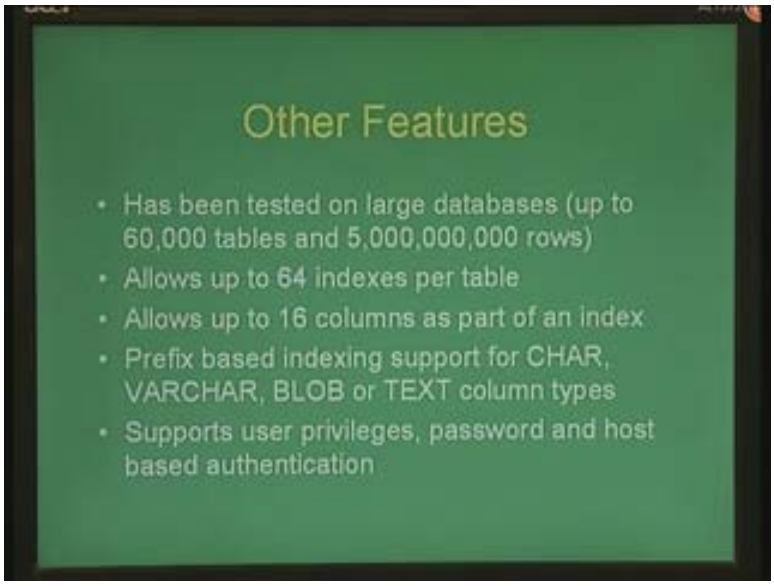


MySQL supports several different data types, several different atomic data types. We saw some kinds of data types when we are talking about SQL for example integers and date type and character strings and characters and enumer or the set data type where you provide the enumerated set of values or the set of all possible values that a particular data element can take and so on.

Similarly in MySQL, there are several different atomic types that are supported, signed and unsigned integers and there are even different lengths of integers they are called as tiny int, small int, integers and big int and so on and so forth. So there could be 1 byte or 2 bytes or 4 or 8 bytes long and there are floating point numbers, double precision numbers, characters, varchar essentially is a string which can, variable number of characters and text and blob is a binary large object which could be something like any media object or audio visual file or anything like that date, time and so on. It even supports variable length records, your record length need not be fixed you can have a combination of fixed and variable length records within a single database. And MySQL supports functions in both from and where clauses of an SQL statement.

Remember an SQL statement has three different parts where you say select something from something else where the condition. So you say select and you say set of field names or attribute names and which says what to select but here in fact you can say select and give a function. So you can say select average of the values of this lets say a total marks taken by or scored by students and so on.

(Refer Slide Time: 17:17)



So you can say select average of marks from student where marks equal to whatever and so on, where mark is greater than 60% or whatever and so on. So you can have functions where average here is a function that's actually in the from clause, not exactly in the where clause. We saw in the session on SQL that functions being supported in the from clause is only a recent addition to the SQL standard and it was not primarily in the initial SQL standards. It also supports, MySQL also supports outer joins both left and right outer joins. Remember what is an outer join, when you are joining two different tables, you basically join let's say you are computing an equijoin, so you basically join based on the value of the certain attribute.

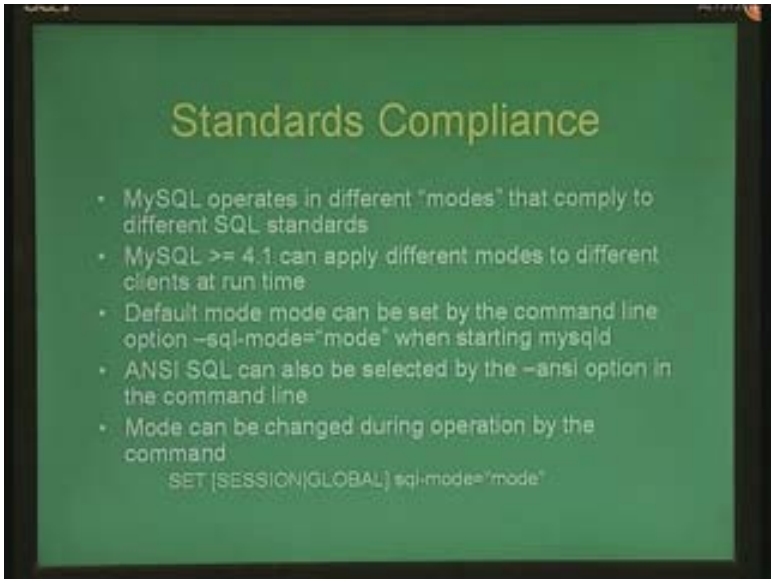
Therefore join happens where, join happens by matching values different values that this attributes takes between the two tables. Now it might so happen that for a particular value in the first table, there may be no corresponding value in the second table. Such records are generally ignored if you are computing an inner join or a usual conventional join operator. But in the outer join operator, you compute even those you don't throw away such records and embed them with nulls essentially.

So this example essentially is an example of left outer join and the corresponding or the dual operation of that were a given attribute in the second table has no corresponding attribute in the first table would form a right outer join. MySQL also supports aliases on tables and columns, so you can refer to the same table and column using different names. And in recent versions of MySQL, especially in MySQL version 3.2.2 and later you can actually provide an SQL query across different databases itself. So where, what is a database here? Databases essentially is a set of tables that are stored under one heading saying this is the student database, this is an employee database and this is the some other database and so on. Here you can actually provide, you can actually give SQL queries that can span across different databases within a single query.

There are also many features like scalability and performance issues and so on. MySQL has been tested on very large databases up to databases which have up to 60000 tables and 5 billion rows in the table. It can allow up to 64 indexes per table that is you can index up to 64 attributes in a table and up to 16 columns as part of an index. Essentially, 16 columns as part of the key that makes with the index. So your key can be up to 16 attributes long and it also supports prefix based indexing. What is a prefix based indexing? Essentially on data types like varchar which is a string or text, you might want to ask queries like show me all students whose name starts with some pra or whatever something like that. So you might want to search for similar strings or you might want to search for prefix match strings and so on. So such kinds of search can be efficiently supported using prefix based indexes rather than indexing on the complete string as such.

MySQL also supports user privileges and password based authentication by users and also host based authentication. So if the same user or if an authenticated user logs in from a different host, here she would have go through a different set of authentication privileges. Here she may not be allowed access from different host and only from particular host as such.

(Refer Slide Time: 17:29)



Now what kinds of standards thus MySQL comply to? Note that when we talk about standards, we don't mean something that is set in time. When we say SQL standard, the standard called SQL itself has been evolving over time. They almost every several years or so new features are added to the SQL standard and which becomes the new version of the standard. In addition to this evolution in standards, MySQL or any other database system would itself be evolving that is when the DBMS is implemented there would be a need that is felt for some more features to be included and so on. So there is a different set of evolution that is happening as part of the database itself or as part of the DBMS itself.

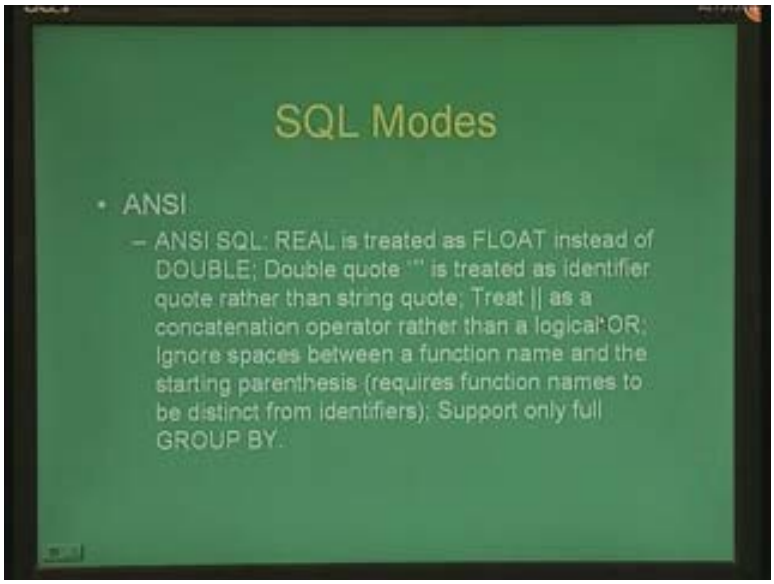
So there are two different evolutionary streams, one of the evolution of the standard and one of the evolution of the DBMS itself. And when we say it supports or it complies to a particular standard, we should be clear about which version of the DBMS supports which version of the standard because both are evolving in its own different trajectory. So MySQL actually has several different modes or it can operate in several different modes that can support several different standards in SQL.

For example MySQL versions greater than 4.1 can apply different modes to different clients itself that is let's say client one connects to MySQL and says I want to use the ANSI standard of SQL. And second client connects to the same MySQL server that is running and says I want to use the IBM db2 standard of SQL and not the ANSI standard SQL and so on.

So the default mode of operation for a client can be set with this option, it is shown here (Refer Slide Time: 19:27) minus minus SQL mode equal to a given mode, when we are starting mysqld that is for the server rather not for the client. That is you can set up, this is the default standard or this is the default mode of operation for MySQL.

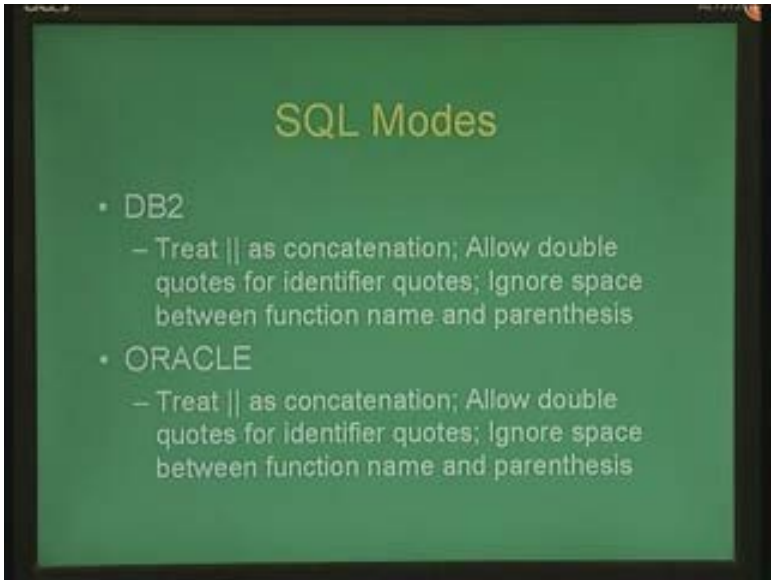
And ANSI SQL can also be selected using the minus minus ANSI option, when you are starting up mysqld that is in the command line itself. And you can set the mode for any particular client at any time using the set command that is set SQL mode equal to some particular mode. So what are the different modes that it supports? The ANSI mode for example where set mode equal to ANSI. So when mode is set to ANSI, there are certain implications in the way MySQL treats given SQL query. For example a real suppose a attribute is named as real then it is by default treat it as double, that is double precision real number by MySQL but in ANSI standard SQL it is treated as a floating point number, the single precision floating point number.

(Refer Slide Time: 22:00)



And similarly you can use double quotes for identifiers rather than strings. That is when you are identifying column name, you can actually use double quotes rather than when you are representing the string data which is part of a column. Similarly this double pipe operator that is shown here is treated as a concatenation operator rather than the logical or operator, the two vertical lines that you see here (Refer Slide Time: 21:14). And when I give a function let us say average, it ignores spaces between a function and the first parenthesis. Suppose this is the function here, now it ignores all the spaces between the function and the first parenthesis here.

(Refer Slide Time: 23:17)

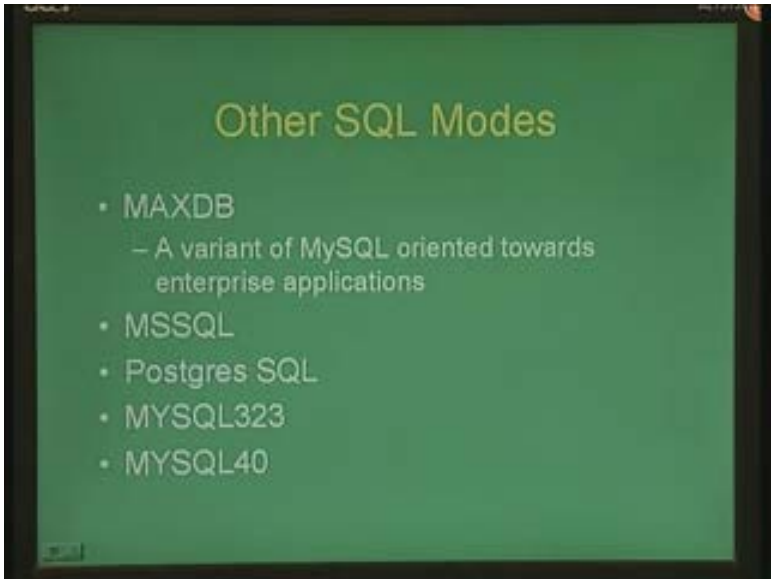


Now what is the implication of this? the implication of this is that I cannot use the same function name as part of a let us say a column name or a variable name or something like that because this would clash with the function name because there is no way to, there is no way for the SQL server to distinguish whether this is a function call or a variable name or anything or a column name during parsing. So by default MySQL does not ignore this thing that is you should not have a space between a function name and the first parentheses so that function names can be repeated or function names need not be unique that is you can use the same function name for your column names or variable names.

Similarly there are other SQL modes for example you can say mode equal to db2 or mode equal to oracle which supports some kinds of parsing, which has some kinds of parsing implications that are compatible with the way IBM db2 works or the way oracle database work. For example both of this modes have more or less the same implication for example you have to treat double pipes as a concatenation operator rather than a logical or operator and allow double quotes for identifier quotes and ignore space between function name and parentheses that is there is you cannot use the same names between the function and a column name.

There also other SQL modes, we shall not go into the details here but let us just enumerate what are all the different SQL modes.

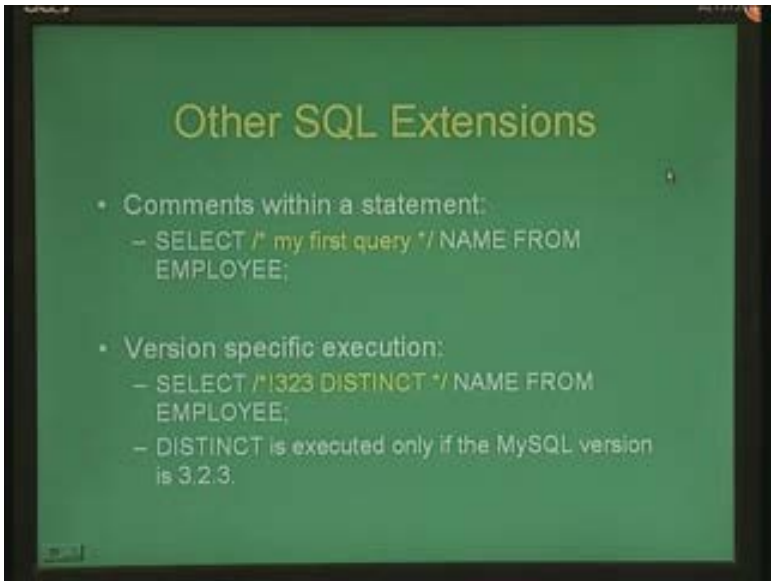
(Refer Slide Time: 23:31)



You can set mode equal to MAXDB, MAXDB is the variant of MySQL which is primarily oriented towards enterprise application. So it's a very, it's a transactional, it supports mainly transactional semantics and you can build large enterprise applications around the MAXDB. And then you can set mode equal to MSSQL which is Microsoft SQL or postgres SQL and MySQL different version like 3.2.3 or MySQL 40.0 and so on. There are other SQL extensions that MySQL supports which are different from the ANSI standard. These include the following that are shown here (Refer Slide Time: 24:12).

For example you can insert a comment c like comment, c like comment means one which starts with slash star and ends with star slash. You can include a comment within a SQL query, anywhere within a SQL query. Similarly you can perform some kind of macro operations or pre-processing operations from which you can bring about selective execution of an SQL query depending on which version of MySQL server this query is being executed on. For example look at this statement here, it says select and within comments it says exclamation mark or bank 3 2 3 distinct name from employee. Now what it means is that if my SQL server or MySQL server which is executing this query is of version 3.2.3, then you execute whatever is there within this comment otherwise comment out this, that is remove this commented part from the query.

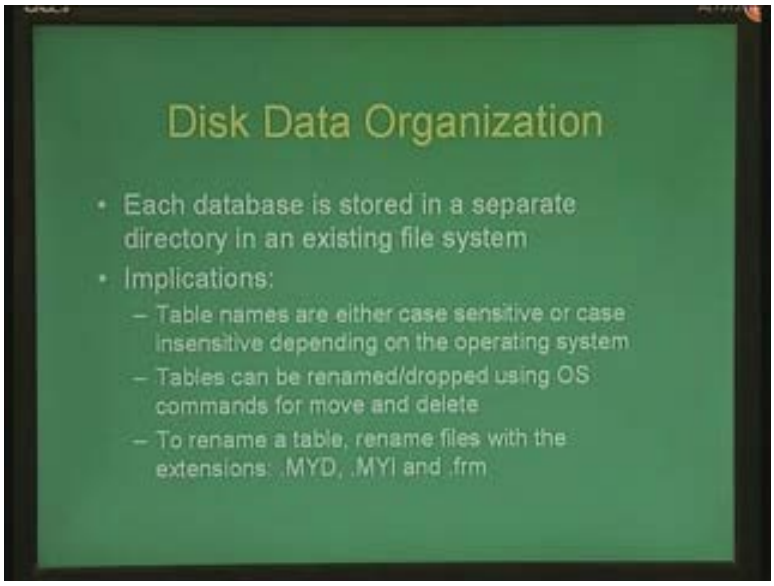
(Refer Slide Time: 24:52)



Therefore if I am running it on let us say MySQL server 4 then the outcome would be select name from employee and only in 3.2.3 it would say select distinct name from employee. Now why would one want to do something like that because there are certain kinds of nuances or certain kinds of changes in the SQL semantics between different versions of MySQL and you can write a single SQL query that can work on any version of MySQL by selectively masking or enabling certain parts of the query.

Let us look at how data is organized on the disk in MySQL. MySQL does not create its own file system, in fact it uses the file system whatever file system is provided by the operating system. And each database is stored in a separate directory in an existing file system. What are the implications of doing this, that is what are the implications of using an existing file system and the fact that you are using, you are keeping one directory per database.

(Refer Slide Time: 26:32)

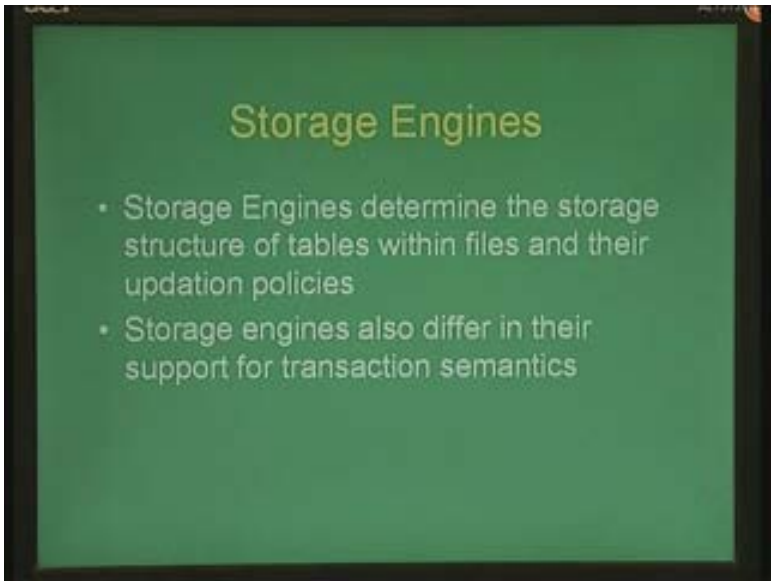


Some of the implications are shown here that is the names of tables can be either case sensitive or case insensitive depending on the operating system. For example if you are using a Unix based system or a Linux based system, it would be case sensitive. So if you say name with n as capital, this would be different from name with all small letters, so they would be two different tables having these two different names. However on an operating system like windows it is case insensitive, where both of them would point to the same table. And tables can be renamed and dropped using operating system commands not necessarily with in MySQL.

For example you can just say move or rename a given file and then the table actually gets renamed. so as shown here to rename a particular table, you just have to rename files with the following extensions .MYD, .MYI and .frm. For example if you have a table called employee you will see that and in a database called employees database, you will see that under the directory called employees database you will have employee.MYD, employee.MYI and employee.frm.

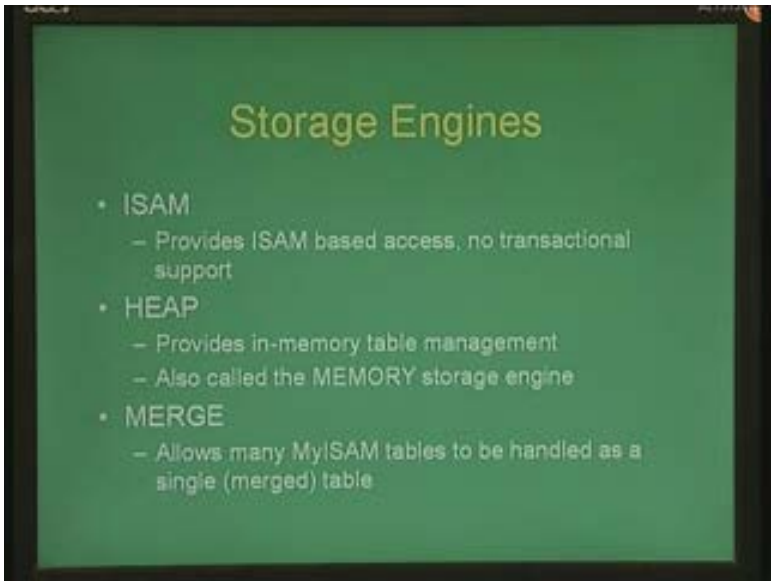
now suppose you want to change employee to something else say contractor whatever, you just have to change or you just have to move these files to some other files names using operating system commands and this change will be automatically reflected within the database itself. What kinds of storage engines does MySQL use and what is the storage engine in the first place.

(Refer Slide Time: 28:39)



A storage engine essentially is the engine that determines how tables are organized within files and what is the updation policy and so on.

(Refer Slide Time: 29:33)

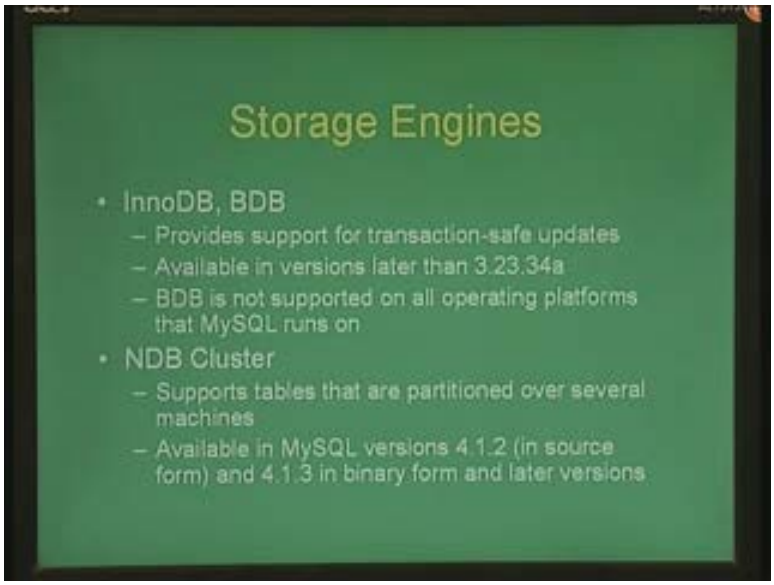


And storage engines also differ with respect to their support for transaction semantics that is does it support atomic updates, does it support isolated updates, does it support recovery and so on and so forth. The different storage engines that MySQL supports, the primary storage engine which many of the earlier versions of MySQL where shift with was the ISAM storage engine. As we saw earlier it is called the MyISAM storage engine. ISAM essentially stands for indexed sequential access mode and provides indexed

sequential access and does not provide any transactional support. There are also other kinds of storage engines like the heap storage engine which is used for managing tables that are in memory that is temporary tables or virtual tables and so on. This is also called the memory storage engine.

There is an other kind of storage engine called merge storage engine which can treat several ISAM tables as a single merge table that is it can efficiently merge different tables that have been created using the MyISAM storage engine. And in recent versions of MySQL there are storage engines like InnoDB, BDB which provides support for transaction safe updates that means it provide support for acid semantics. They are available in versions only later than 3.23.34a. So to be very safe and also you should also be aware of the fact that in these versions or if this is the first version in which this storage engine was released then it would be available only in source code form and not in pre compile binaries.

(Refer Slide Time: 30:45)

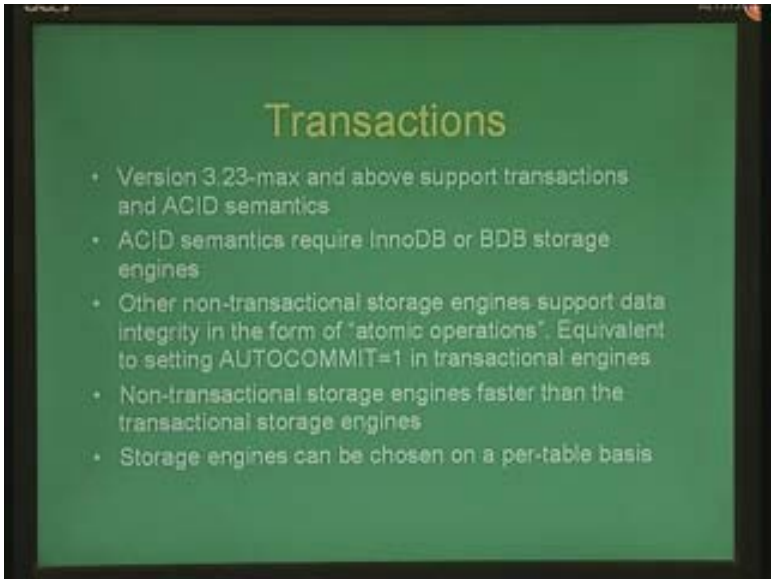


Therefore if you want to use transaction semantics on MySQL, it would make a lot of sense to download a later version something like MySQL 4 or MySQL 5 which is the latest as of today. And the BDB storage engine is not supported on all operating platforms and so you have to ensure that for whatever platform that you are using, the BDB storage engine is actually supported.

There is also the NDB clusters storage engine where you can implement MySQL over a computational cluster. What is a computational cluster? It is a collection of different machines on a local area network which can provide what is called as the single system interface. That is which can provide the combined power or processing power of all the processors of all the machines.

Now NDB clusters is a storage engine that exploits this capability of a cluster, so it can support tables that are actually physically partitioned over different machines in a cluster. And of course NDB cluster is also very recent storage engine that is supported by MySQL and it is available in versions 4.1.2 and later on. And like we had noted above that in 4.1.2 it's only available in source form that is you have to compile this engine yourself but in later versions, the binary forms or pre compiled versions are also available.

(Refer Slide Time: 32:34)



What about transaction semantics in MySQL? Transaction semantics have been supported in MySQL only after version 3.23 max and above and so on but before that MySQL actually supported what are called as atomic updates. We will look into atomic updates shortly and what are the relative advantages and disadvantages of atomic updates versus acid semantics and so on. But if you want specific acid semantics, suppose you are using your MySQL engine for let us say performing some debit credit updates in any kind of whether banking or any kind of money transaction that is going on which has to be recorded on this.

It's better to use MySQL which supports the Inno DB or the BDB storage engines because they have support for full transaction semantics. And other non-transactional storage engines like the earlier versions of MySQL and so on support what are called as atomic updates or atomic operations. And this can be simulated in a Inno DB for example by setting auto commit equal to 1 that is commit is automatic and it's not explicitly stated in the SQL query.

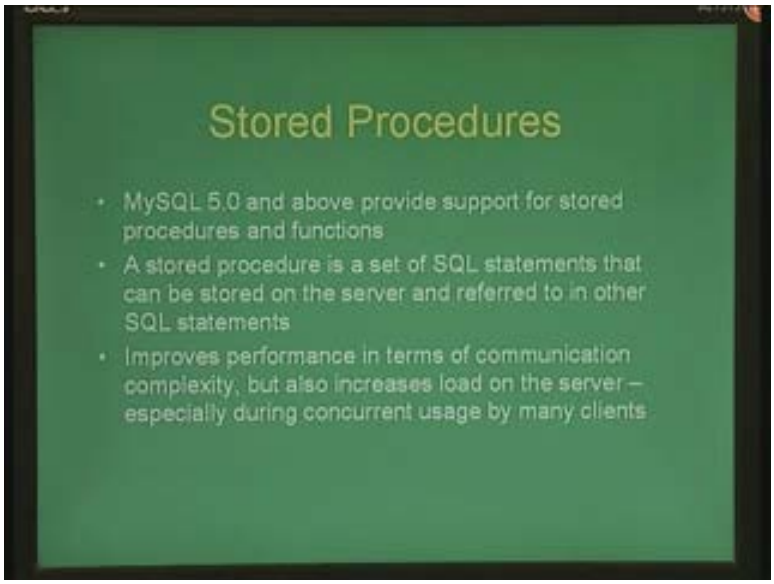
These non-transactional atomic updates are actually faster than the transactional storage engines and because of course you don't have to worry about locking and isolation and other isolation based semantics nor do you have to worry about logging and so on where the first was locking and the second is logging and logging in order to support log based

recovery mechanisms that is in order to be able to support durability aspect of acid semantics.

Now because all of this are not required and only atomic updates are required, it's much faster. and in many cases this is all that is required that is suppose you know that there are no concurrent operation that are happening or suppose you are relatively sure of the uptime of your DBMS then it makes much more sense to use non transactional storage engines which support only atomic updates because they are many times faster than acid, I mean storage engines which support acid updates. And a very interesting feature of MySQL is that you can use storage engines on a per table basis. So you are actually save only for this table use Inno DB and for every other tables use let us say MyISAM and so on. So that you use atomic updates and only for the critical table, you say that I want full acid semantics.

The next topic that we will see are stored procedures. What are stored procedures? stored procedures is essentially a set of SQL statements that can be actually be stored on the server and some kinds of SQL statements which are used pretty often in different queries. So you don't have to keep writing it as part of the query again and again. And you can actually refer to these stored procedures as part of a given query. There are two kinds of, two ways in which you can store SQL statements as what are called as stored procedures and stored functions. A procedure essentially performs a set of operations and may or may not return a value or may even return multiple values when it is called.

(Refer Slide Time: 36:11)

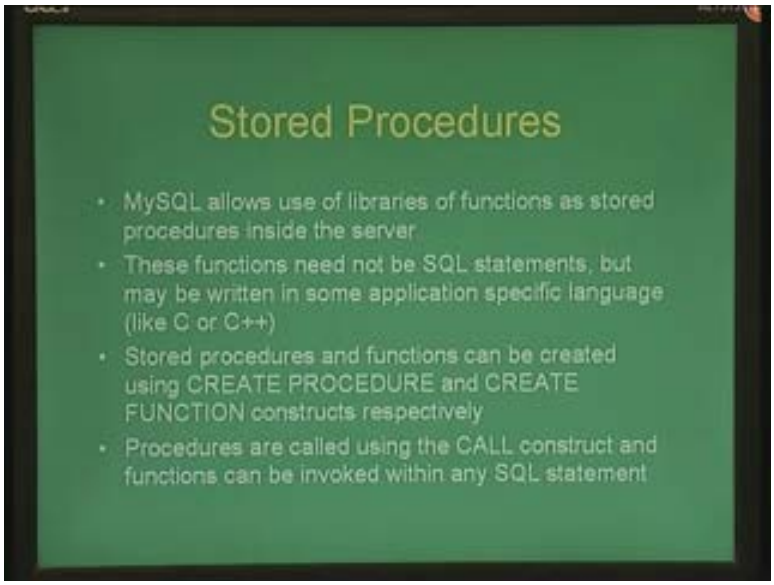


But a function has a specific semantics that is it can be embedded within any SQL statement and it returns back exactly one value as part of its execution. So stored procedures support is a very recent phenomenon in MySQL and it is supported in MySQL versions 5.0 and above and the latest version is actually as of now is actually 5.0 and whatever later versions would still support stored procedures.

And a stored procedures as you know increases the performance in terms of communication complexity, especially if you are using a large number of stored procedures in your query, you can save on a large amount of communication complexity between the client and the server. so you don't have to send all the query from the client to the server for execution but the flip side of stored procedure is that if several different clients are connecting to the server and calling several different stored procedures then the sever gets a huge load in trying to execute all this different stored procedures.

And MySQL also allows libraries or a function to be used as a stored procedure within the server. That is these libraries need not actually be SQL statements but they could be pre compiled libraries written in some other application specific language like C or C plus plus. And stored procedures are created using the CREATE PROCEDURE construct in SQL and functions can be created using the CREATE FUNCTION constructs and procedures are invoked using the CALL constructs, you can say CALL procedure 1 or procedure 2 and so on, while functions can be embedded within any SQL statement like we saw select average of whatever or select anything and so on.

(Refer Slide Time: 37:41)

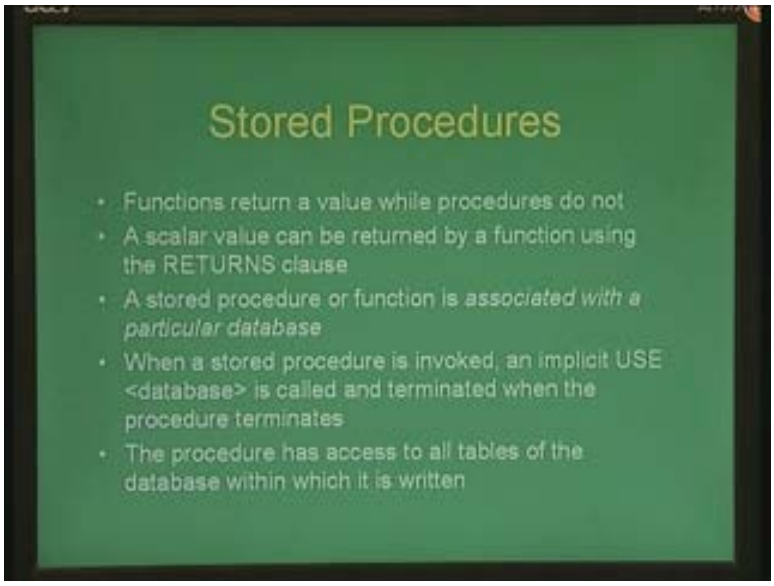


So you can actually embed function within an SQL statement, however the function has to return one specific value. So functions return a value while procedures need not, it's not actually do not but it need not return a value. In fact a procedure can actually return multiple values through the input parameters itself and a function is distinguished from a procedure by this RETURNS statement or the RETURNS clause. So function says returns this thing, returns a particular value when you are declaring a stored function. And a stored procedure or a function is associated with the specific database. What are the implications of saying that a stored procedure is associated with a specific database?

See you can say for example I have an employee database and there is a particular kind of query that is invoked on this database for example what is the average, what is the

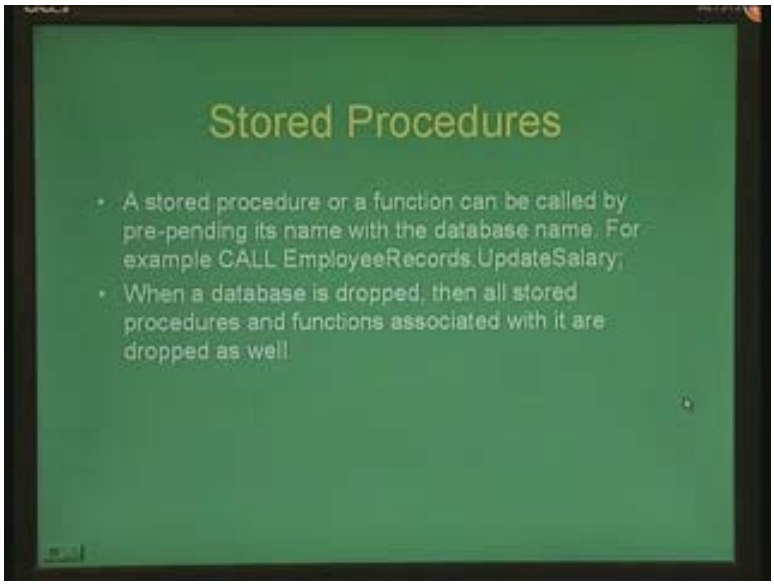
average working hours per week in this month and so on. Now you can actually store that as a stored procedure in the employee's database. Now let us say you are using some other database, let us say salaries database or some other completely different database itself not some other table but you can still use or invoke a stored procedure on the employee's database. Note that in SQL, in MySQL now you can actually give an SQL statement that spans over multiple databases.

(Refer Slide Time: 38:54)



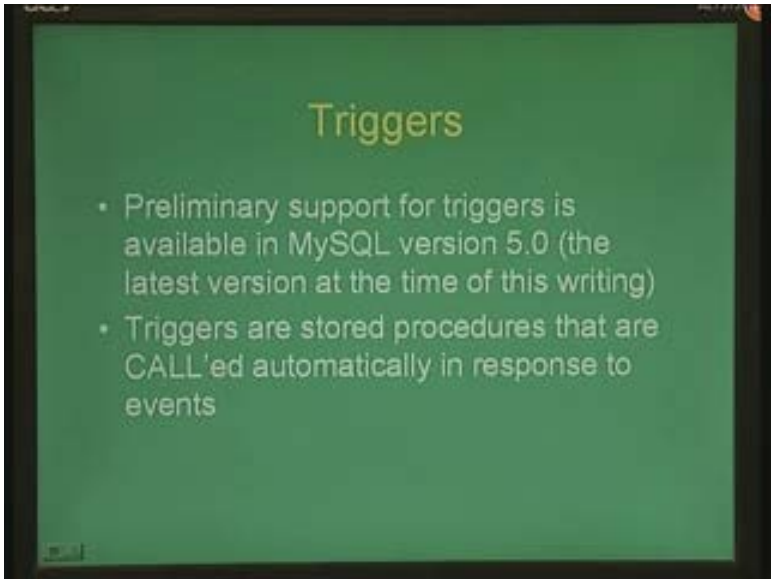
So when you invoke a stored procedure on a given database and implicit command called USE database is called. USE database is basically something like connect to the database and start using tables in this database.

(Refer Slide Time: 41:31)



And this USE statement is terminated when the procedure terminates and the procedure now has access to all tables of the database within which it is written. And of course you can call stored procedure of a specific database by prepending its name with a database name. For example here you say CALL EmployeeRecords.UpdateSalary where UpdateSalary is a procedure that performs updates on salary and employee records is the database within which this stored procedure is stored. And another important implication of the fact that stored procedures are associated with the database is that when a database is dropped or it is deleted then all stored procedures and functions that are associated with it are also dropped.

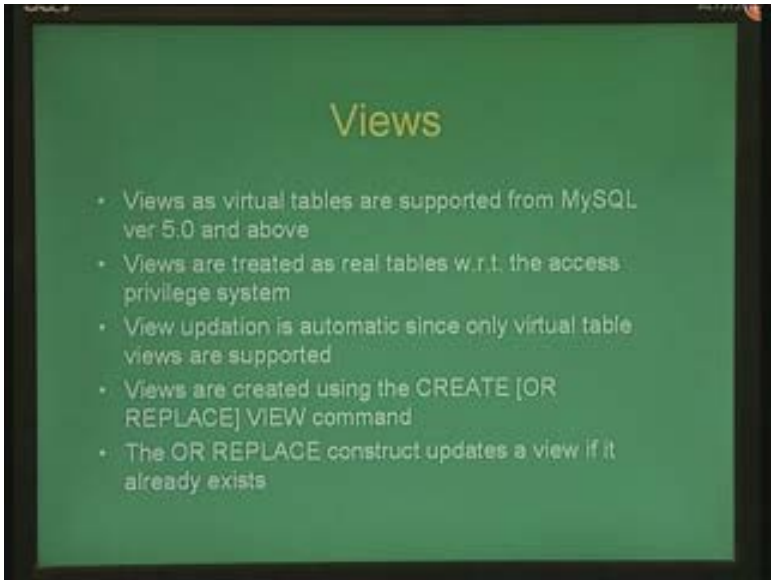
(Refer Slide Time: 41:37)



Support for triggers. Triggers, as you know we covered a session on constraints and triggers. Triggers as you know are essentially special kind of stored procedures that is they are written using ECA rules as we saw in one of the previous sessions. That is they are triggered automatically when an event happens and a given conditions is true. Now support for triggers is not very comprehensive in MySQL as of now at least. That is there is only preliminary support for triggers that is available in MySQL version 5.0. And triggers are essentially stored procedures and which are actually called automatically in response to certain events.

MySQL also supports views, as we saw view can be created either as a virtual table or as the materialized view. That is view can be stored as a query or can be actually physically materialized depending on what kind of application that you are using. In transactional applications views are usually virtual tables whereas let us say in data warehousing and other analytical applications, views are actually materialized. MySQL supports views as virtual tables and they are supported from MySQL version 5.0 and above.

(Refer Slide Time: 42:31)

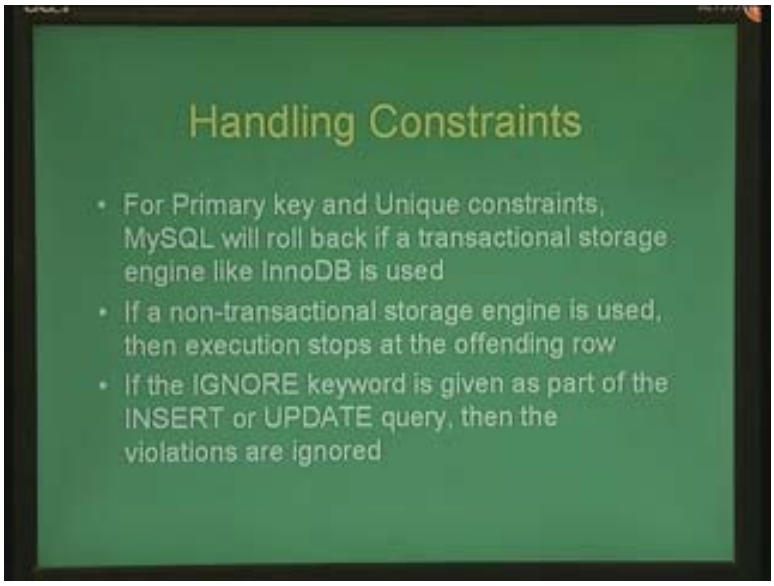


And they are treated as real tables for all practical purposes that means each table has what is called as an access privilege system that is which user has what privileges for a particular table. For example you can say this user has a read privilege but no write privilege or this user has a read privilege write privilege but no drop privilege that is you can read and write to the table but he cannot drop the table, he cannot delete the table and so on.

And view updation is automatic because only virtual views or virtual tables are supported. And views are created using CREATE VIEW command or there is also a variant of this which says CREATE or REPLACE VIEW. So if the OR REPLACE construct is present then a view would be updated if another view of the same name exists. And how does MySQL handle constraints and of course triggers and so on?

In MySQL the notion of triggers or calling stored procedures is separate from that of handling constraints. The kinds of constraints for example data specific constraints like say primary key constraint or unique constraint, if they say violation on the primary key constraint essentially that primary key has to be unique and it can't be null.

(Refer Slide Time: 44:20)

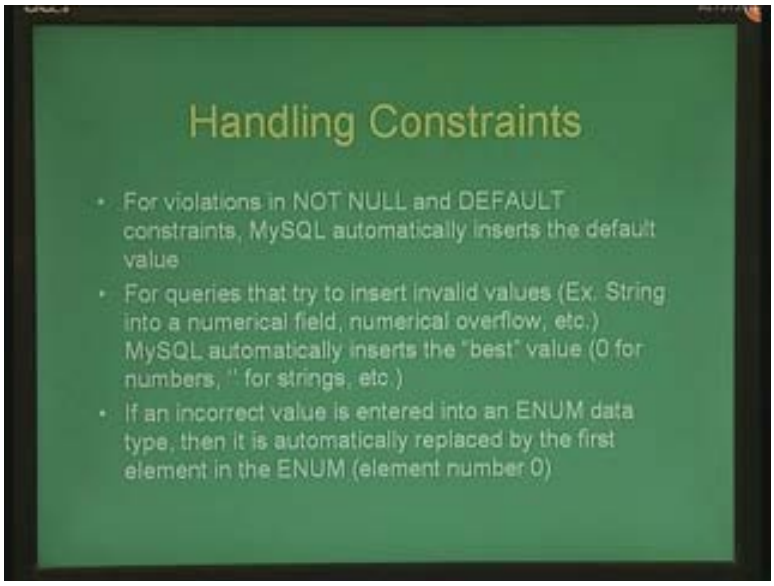


So if there is a violation then MySQL will roll back, if a transactional storage engine is used that is it just performs the transaction rollback and if a non-transactional engine is used then it just stops execution on the offending row that is it does update the row and from there on it does not update any other row in the engine. So therefore it is not an atomic operation that is it could have updated some rows prior to it but it will stop execution at that point in a non-transactional storage engine. And many of this insert or update queries also support a keyword called ignore essentially that means that constraint violations are ignored.

Similarly if there are violations in NOT NULL constructs and say default constructs, MySQL automatically inserts the default value. for example if you say that some attribute should be not null and the default value is say zero then whenever that attribute value is null then zero is inserted automatically. Now the reason that MySQL does not stop in this case is that you will be able to catch such violations only after the query parsing stage has been finished. And it would make it quite in efficient to stop execution at this point and get back to the user for a different query. So MySQL automatically, because it already has a default value it automatically inserts the default value.

And similarly for queries that try to insert invalid values, for example if it tries to insert string into a numerical field or some kind of a numerical overflow happens during insertion and so on. MySQL automatically inserts the best value that is zero for numbers or null string for strings and so on.

(Refer Slide Time: 45:32)



And similarly in an enumerated data type, what is an enumerated data type where you say this data type can or this column can take only these set of values. So one can say the column called gender can only take m or f or column called grade let us say, can only take values a b c or d and so on. So if an invalid value is entered in to an enumerated type then it is automatically replaced by the first element in the enumerated type that is the element number zero. Let us see now how you can actually download and install MySQL on a Linux based environment and get it running.

Now of course in order to download MySQL, we already saw where to visit in the internet www.mysql.org and from that you can actually download MySQL on to your machine. Now there are several different sources and binary versions of MySQL that are available and the pre compiled binaries that are available for different linux distribution. So there are pre compiled binaries for say SUSE Linux or Mandrax Linux or Red Hat Linux and DBAN and so on so forth.

(Refer Slide Time: 48:15)

The easiest way of installation is to use an appropriate binary RPM package. RPM essentially is the Red Hat package manager which is the default way in which the packages are managed across different Linux distributions and binary versions of MySQL are complied with static option which uses static libraries. So it's quite stable that is there is very little possibility or no possibility of linking violations or run time error occurring after the MySQL server has started running.

(Refer Slide Time: 48:51)

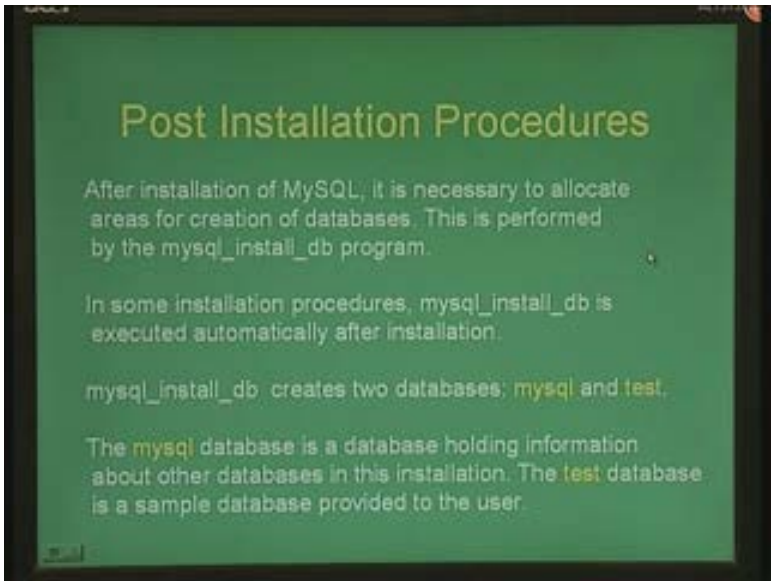
So when you install MySQL on your system, you need to install at least MySQL server and MySQL client packages in order to get a complete functional DBMS on your system.

That is you should have a both server part and a client part. You should also install this package called MySQL shared compat for backward compatibility especially if you are upgrading to a newer version of MySQL. MySQL servers are by default installed in to the in to this directory var lib MySQL, so after you install you can actually visit the directory and see what all it is installed and a new user called MySQL is also created and which is the owner of the MySQL demon or the MySQL server that is running and it's also possible to make MySQL server start up automatically whenever you boot up your machine.

(Refer Slide Time: 50:03)

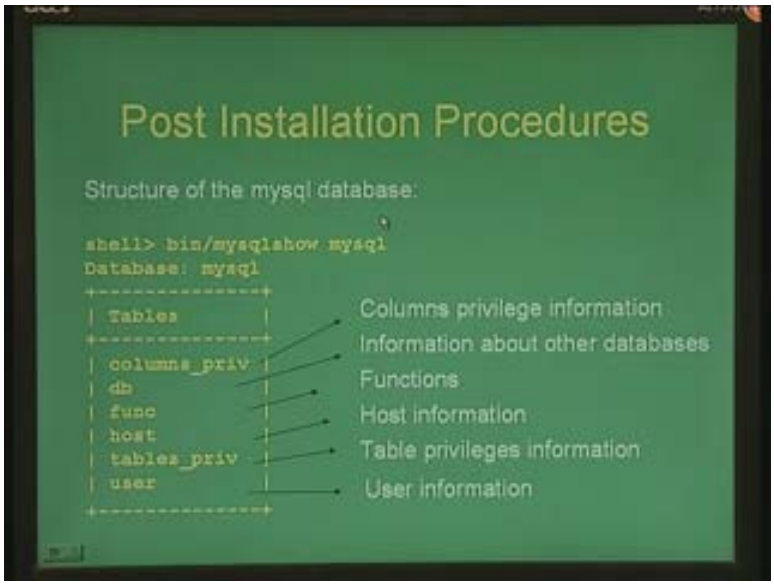
Let us see how we can do all of these things. In order to install MySQL on your machine, you have to use this rpm minus i command or the MySQL or the install command which installs the server and the client and of the appropriate version number and after installation of MySQL it is necessary to allocate certain areas where databases can be created. And this is automatically perform by the MySQL install db program and in some kinds of installation procedures, this is, this program is automatically called after installation. So you don't have to even worry about calling this program. And MySQL install db creates a directory structure under which databases are created where each database as you know is a directory in itself.

(Refer Slide Time: 50:40)



And by default it creates two databases MySQL and test. The MySQL database is the database of databases that is it's a database holding information about other databases in this installation and the test database is a sample database that is provided to the user. So let us look at how MySQL database looks like. Once you start let us say, once you have run your MySQL install db there is something called MySQL show which will show the contents of a given database.

(Refer Slide Time: 51:16)



So when you say MySQL show MySQL, you see that MySQL database has several different tables columns underscore prev db func host tables underscore priv user and so

on. So, which holds different information's about all the databases that are stored on this machine. For example this holds user information which are all the valid users and their passwords and their privileges and so on and so forth. And tables privileges, that is per table privilege, this table is authorized to which user for doing what and so on. Then there are host based privileges and functions that are stored as part of this database and db is information about the other databases that are stored on this machine and this is a column based privilege information.

(Refer Slide Time: 52:46)

And you can, like we had noted earlier you can make MySQL to start and stop automatically or start automatically on boot up, whenever you boot up your machine. For that you need to go to /etc/rc.star where star is either dot 3 or dot 5 or whatever in which you start your computer by default and those files should be changed in order to start up MySQL server by default. And the MySQL server can be stared at any time using this command MySQL.server start and it can be stopped at any time from the shell by MySQL.server and then stop.

(Refer Slide Time: 53:22)

There are other options for MySQL server that can be added in a global configuration file and this is available in /etc/my.cnf and this is the typical global configuration file which is saying which is the database directory, which is the socket under which MySQL is connected, which is the port on which the MySQL is listening and who is the user owning the MySQL and so on.

(Refer Slide Time: 53:41)

And the MySQL client is the client program using which you connect on to the MySQL server. So MySQL client can be started by MySQL minus h host minus u user that is

connect to the server on this host using this user id and then it ask for a password and then you get in to the MySQL prompt.

(Refer Slide Time: 54:17)

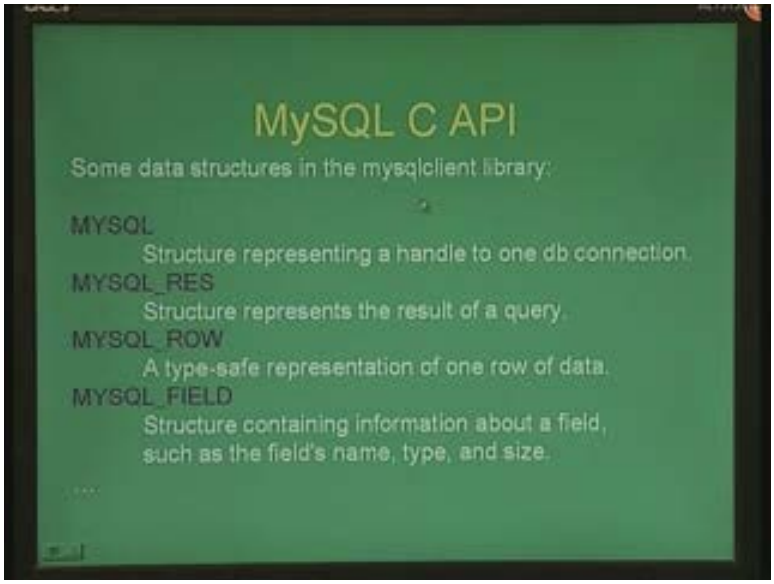
And SQL commands can be issued from the MySQL prompt and it works within a user session that is you can say use particular database which is the same as connect to a database in several other DBMS system. And then you can start issuing SQL commands and you can quit the client using the quit command.

(Refer Slide Time: 54:35)

In addition to SQL using the MySQL client itself, MySQL provides support for several application programming interfaces that is you can embed your MySQL client within another application program like C, C plus plus, perl, java and so on and so forth.

Let us have a brief look at what kinds of C APIs that does MySQL provides. In order to use the C API in your C program, you have to include the MySQL client library and once you include the MySQL client library there are several data structures that are available to your program.

(Refer Slide Time: 54:53)

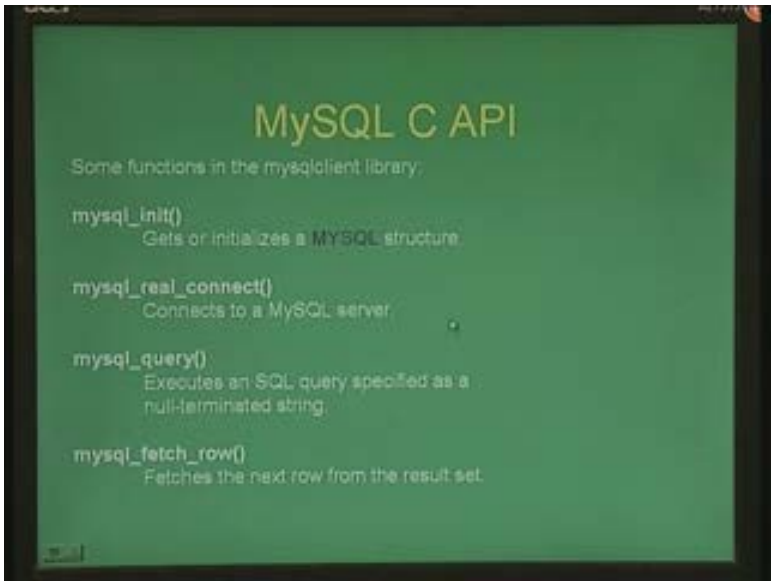


For example there is a data structure called MySQL which is a structure representing, which is the struct, structure is essentially the struct data type in C that represents a handle to a db connection or to a database connection that you have presently opened.

Similarly MySQL underscore RES represents the result of a query and so on and so forth. So there are several different such data structures which you can access. Similarly there are several different functions that you can also use as part of your program. For example you can say MySQL underscore init which gets or initializes the MySQL structure that is it obtains the handle to a data base connection.

Similarly you can say MySQL real connect which actually connects to the server and you can issue a query using MySQL underscore query and once you get the query results, you can say MySQL fetch row which fetches the results of this query in a row by row fashion, so the first row and the next row and so on so forth.

(Refer Slide Time: 55:47)



So using this you can actually embed your SQL semantics or database semantics in to your larger application program itself.

(Refer Slide Time: 56:41)

So let us summarize what we have learnt about MySQL today. So it's a very interesting database, it's a very popular open source DBMS that is the source code is available to you and as a result MySQL has or changes to MySQL has been contributed by several people across the world.

In addition to the MySQL ab people, who are in MySQL ab and it provides several different storage structures and its scalable to number of clients and data sizes and it also has transaction support and limited support for triggers and stored procedures. And best of all, there are several different, it has been ported on several different platforms you have MySQL for window, Mac and Linux and so on. And it has several different APIs for application programming or embedding application programs in to MySQL. So that brings us to the end of this session.