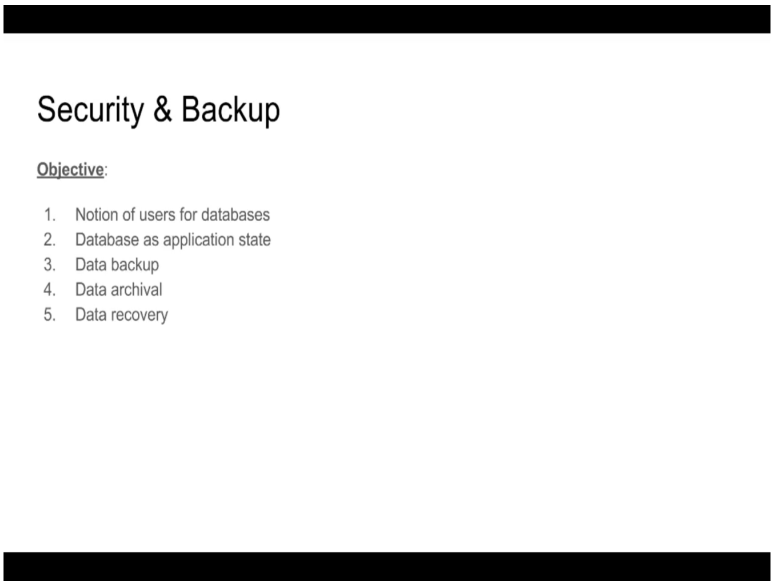
Introduction to Morden Application Development

Dr. Gaurav Raina
Prof. Tanmai Gopal
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 10
Lecture – 19
Database security, backup & recovery

(Refer Slide Time: 00:06)

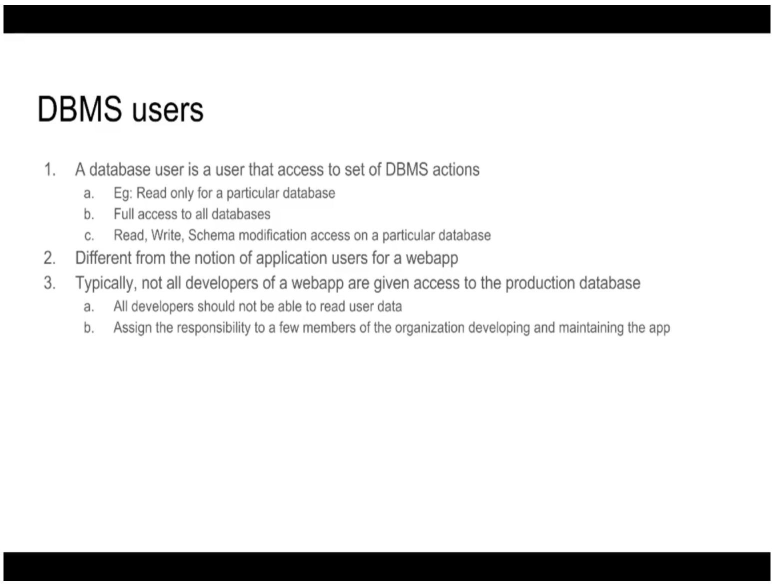


Security & Backup

Objective:

1. Notion of users for databases
2. Database as application state
3. Data backup
4. Data archival
5. Data recovery

(Refer Slide Time: 00:07)



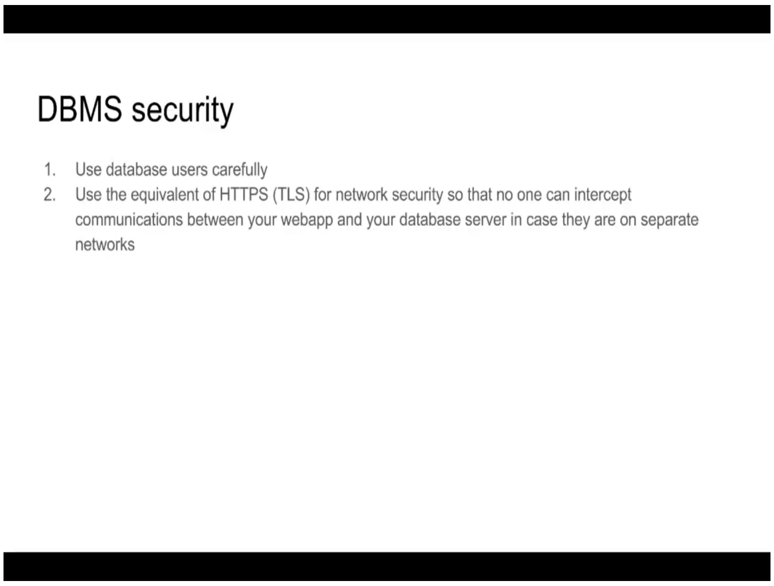
DBMS users

1. A database user is a user that access to set of DBMS actions
   a. Eg: Read only for a particular database
   b. Full access to all databases
   c. Read, Write, Schema modification access on a particular database
2. Different from the notion of application users for a webapp
3. Typically, not all developers of a webapp are given access to the production database
   a. All developers should not be able to read user data
   b. Assign the responsibility to a few members of the organization developing and maintaining the app

Hi all; welcome to module 10, in this module we will talk about database security back and recovery. The first thing that we are going to try to understand is a concept of DBMS users. So, a database user is a user that gets access to a certain set of DBMS actions for example, in the practical modules that we did last week when you going to the I mat console and clicked on create DB credentials user name and the password was generated that allowed to access a specific database and the database was also named after the user name. So, permissions to create data to modify schema inside the database to update data to create indexes, to update indexes etcetera were all privileges that were given to that user on that particular database only right.

Similarly, you can actually create more users have more specific privileges on 1 or more databases. It is important to understand that a database user is very different from the notion of an application user for the web app for example, in my case I use the username coco 98 to login to a database called coco 98 and perform my actions there; however, in my database I had the data models for the user table and the articles table and the user table contained users that had nothing to with the user coco 98. So, application users are different from database users. The reason why database users are an important concept provided by most mature DBMS systems today is because typically in an application not all developers are given access to the production database only a few key members of your organisation will have the access to the database credentials which will allow your web app code to talk to your database.

(Refer Slide Time: 01:47)

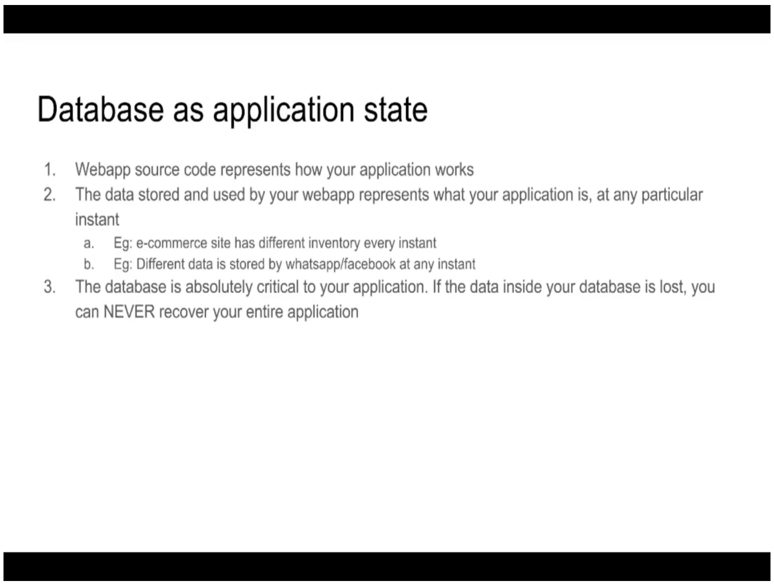


DBMS security

1. Use database users carefully
2. Use the equivalent of HTTPS (TLS) for network security so that no one can intercept communications between your webapp and your database server in case they are on separate networks

So, the way to ensure appropriate security of your databases that are managed by DBMS is to use database users carefully. So, we should create appropriate roles for members with an organisations and appropriate access credentials for the different piece of software that access the database we should also use the equivalent of HTTPS which is called TLS for network level security. So, that nobody can intercept the connections between the web app and the database server.

(Refer Slide Time: 02:14)



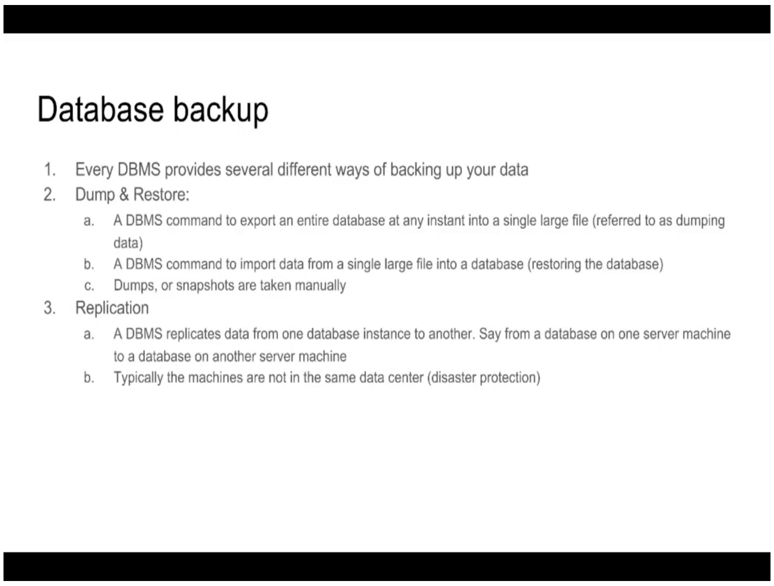
Database as application state

1. Webapp source code represents how your application works
2. The data stored and used by your webapp represents what your application is, at any particular instant
    a. Eg: e-commerce site has different inventory every instant
    b. Eg: Different data is stored by whatsapp/facebook at any instant
3. The database is absolutely critical to your application. If the data inside your database is lost, you can NEVER recover your entire application

Now, the database is essentially the state of the entire application the web app source code of our application represents how the application works, but the database actually represents what the application is at any particular instance. So, for example, in e-commerce like has a different amount of stock at any instance.

So, what the state of the e commerce site is represented only by it is database if we lose the information inside the database our entire application actually cannot be recovered in case we lose portions of our source code it might be possible for us to recover our source code or to rewrite our source code, but in case data that has being dynamically created by the activities of our web app or the activities of our users are lost. The entire state of the application as it was is also lost.

(Refer Slide Time: 02:57)



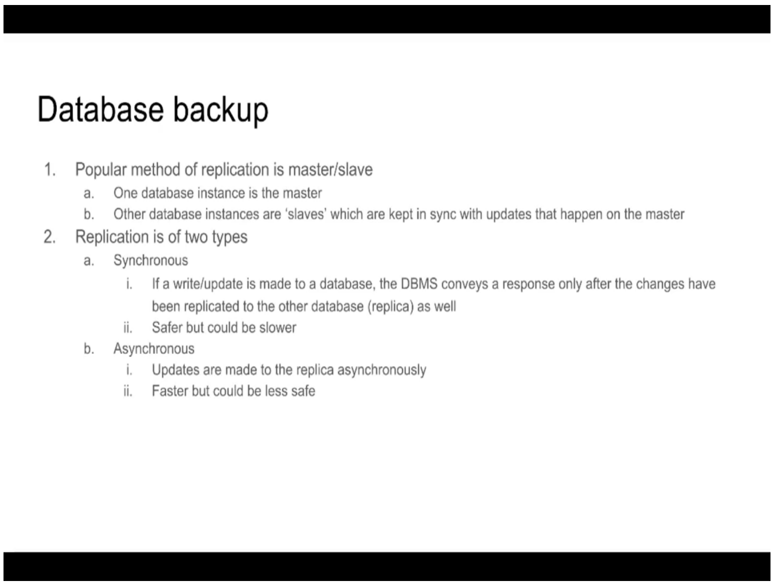
Database backup

1. Every DBMS provides several different ways of backing up your data
2. Dump & Restore:
   a. A DBMS command to export an entire database at any instant into a single large file (referred to as dumping data)
   b. A DBMS command to import data from a single large file into a database (restoring the database)
   c. Dumps, or snapshots are taken manually
3. Replication
   a. A DBMS replicates data from one database instance to another. Say from a database on one server machine to a database on another server machine
   b. Typically the machines are not in the same data center (disaster protection)

So, it is extremely important to start backing up our database and every DBMS provides a several different ways of backing up data the most common method is dumping and restoring data. So, DBMS is provided commands to export in a entire database into a single file or into a set of single files and this is referred to as dumping data.

Another DBMS command allows to import this set of files or single file and use that to create or populate an existing database. So, a backup strategy is to take dumps or take snapshots at the database manually at regular intervals and then in case of emergency situation when a restore is required the database is restored from this dump or from this snapshot. This obviously, means that the last state of the application that we have is when the snapshot was taken and all the information that was captured in the database after the snapshot was taken is lost. Another style of back up which is more optimated is replication DBMS replicates data from one database instance to another say we have one instance of a DBMS which has a database on server machine and other DBMS another database instance on another sever machine modern DBMS has allowed a particular configuration that allows data to be replicated from one database to another.

(Refer Slide Time: 04:17)



Database backup

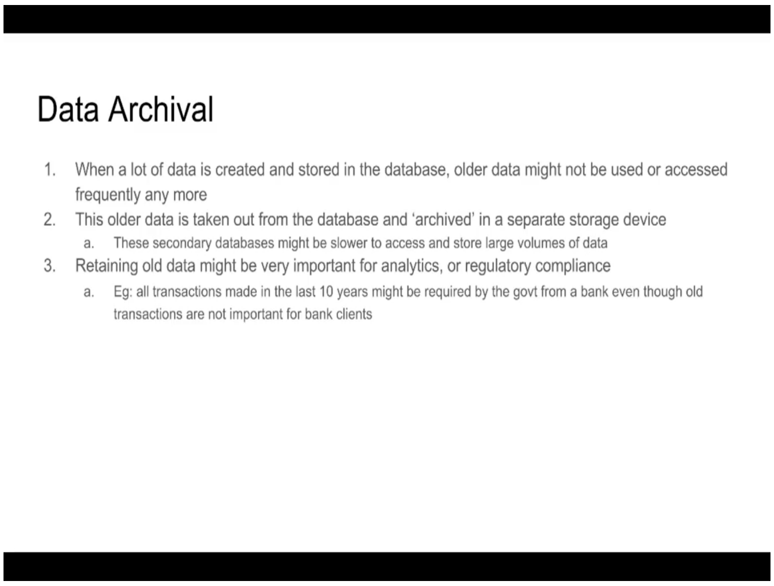
1. Popular method of replication is master/slave
    a. One database instance is the master
    b. Other database instances are 'slaves' which are kept in sync with updates that happen on the master
2. Replication is of two types
    a. Synchronous
        i. If a write/update is made to a database, the DBMS conveys a response only after the changes have been replicated to the other database (replica) as well
        ii. Safer but could be slower
    b. Asynchronous
        i. Updates are made to the replica asynchronously
        ii. Faster but could be less safe

So, typically these machines are also not kept in the same data centre for a disaster protection a popular method of replication is master slave one database instance is called the master and the other database instance is called slave. So, slaves are kept in sync with updates that happen on the master. All the activity happens on the master database so the web app communicates with the master database and all of these updates are propagated to the slave databases.

So, now, in case something goes wrong on the master database we can always recover from the slave this is the advantage that in case something goes wrong we can recover our data, we can recover all most all of our data from these slave replica of our database there are 2 kinds of replication the synchronous and asynchronous replication a synchronous replication is when updates are transmitted from the master to a slave and only once the updates have actually been propagated to the slave as well there is the master actually save that an update has been committed. So, this is a very safe method of replication, but it can be slower because we are not just updating one database every time we update the master we are updating the master and the slave database.

Asynchronous replication is when updates are made to the master, but the master does not wait to propagate those updates to the slave replica the master response immediately and. So, updates to the slave replica are made asynchronously this is faster because the master response the master database responds to updates much faster, but it is potentially less safe than synchronous replication.

(Refer Slide Time: 05:34)

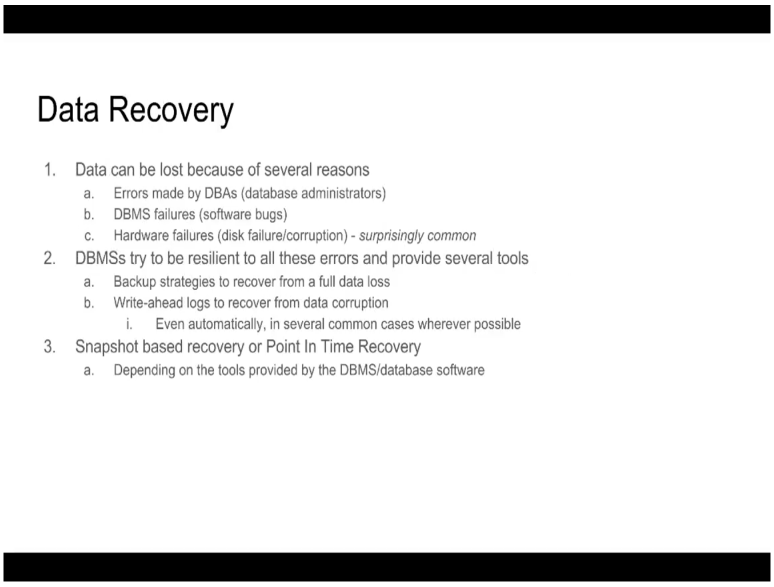


Data Archival

1. When a lot of data is created and stored in the database, older data might not be used or accessed frequently any more
2. This older data is taken out from the database and 'archived' in a separate storage device
   a. These secondary databases might be slower to access and store large volumes of data
3. Retaining old data might be very important for analytics, or regulatory compliance
   a. Eg: all transactions made in the last 10 years might be required by the govt from a bank even though old transactions are not important for bank clients

Another concept is the concept of data archival in applications where a lot of data is created some of the older data might not be very useful or might not be accessed very often. In this case old data is actually taken out from the database and archived in a separate storage device and these secondary databases actually might be different kinds of databases or might be or different kinds of hardware that are slower to access and are able to store larger volumes of data. It is important to archive this old data and not just delete the old data because the old data might be important for analytics purposes or for regulatory compliance purposes.

(Refer Slide Time: 06:09).



Data Recovery

1. Data can be lost because of several reasons
   a. Errors made by DBAs (database administrators)
   b. DBMS failures (software bugs)
   c. Hardware failures (disk failure/corruption) - *surprisingly common*
2. DBMSs try to be resilient to all these errors and provide several tools
   a. Backup strategies to recover from a full data loss
   b. Write-ahead logs to recover from data corruption
      i. Even automatically, in several common cases wherever possible
3. Snapshot based recovery or Point In Time Recovery
   a. Depending on the tools provided by the DBMS/database software

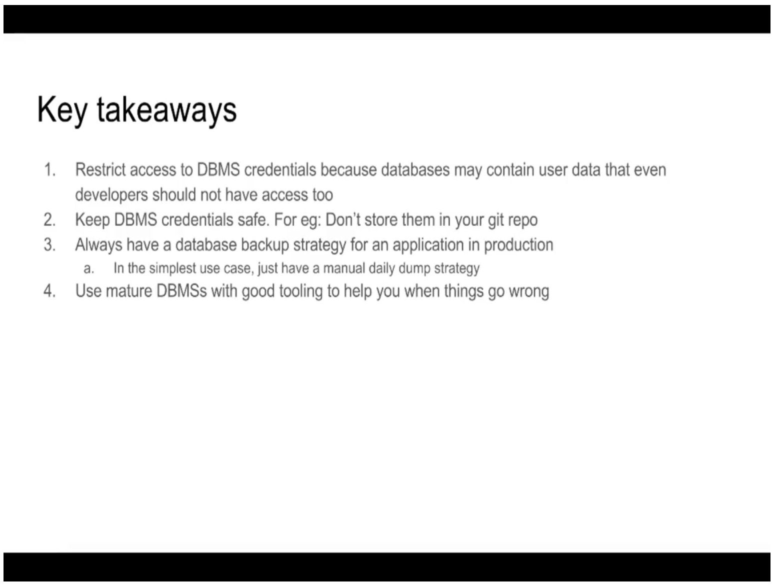
It is important to think about data recovery and what kind of guarantees we want for the data recovery of our application from the very beginning. It is important to think about because data can be lost because of several reasons errors can be made by database administrators or that might be software errors in the DBMS, there might be hardware failures where the disk fails or there is disk corruption and this is actually a surprisingly common failure.

So, to be resilient to these kinds of errors it is important to think about what the recovery characteristics of our applications are how much time do we want to be able to recover our entire database in our application then our application is right. Most DBMS says try to be resilient fairly atomically to most of these errors for example, for example, we discuss that most DBMS provide backup strategies that allow a recovery from full data loss, but in case there is partial data corruption then databases try to recover from it automatically like we discussed transactions in the last module. In case transaction has in case of transaction could not complete entirely the database detects that a few operations have happened which need to be rolled back and the database is able to roll back those operations.

Databases like PostgreSQL use the concept of a write ahead log a write ahead log is essentially a log that is maintained by the DBMS logs all the actions and queries that are made by users into a file and once the data and once the query is logged in to the file the database server actually applies that query on to the database. This is the advantage that in case something goes wrong that in case of query has not being completely processed and applied to the database the database server can read the write ahead log to figure out what the query was and replay the query. Another style of data recovery is called point in time recovery when we discussed our when we discuss the data backup strategy of dumping and restoring we were looking at what is called snapshot base recovery where you were recovering a snapshot of the data from the time the dump the database dump was taken.

A point in time recovery is some more complex kind of recovery when we want to recover the database state as it was at any particular time in the past. So, a point in time recovery method allows us to choose any particular movement from the past and move through all those different queries as they executed and choose an exact particular instant at which we want to reset our database and this can be done only by specialised tools that are provided by specific DBMS and databases softwares.

(Refer Slide Time: 08:35)



Key takeaways

1. Restrict access to DBMS credentials because databases may contain user data that even developers should not have access too
2. Keep DBMS credentials safe. For eg: Don't store them in your git repo
3. Always have a database backup strategy for an application in production
   a. In the simplest use case, just have a manual daily dump strategy
4. Use mature DBMSs with good tooling to help you when things go wrong

The important things to take out of this short module are number one we should be very careful about the database credentials that we generate if we are not careful about these database credentials then people will have access to user data that is stored in database. It is important to keep these DBMS credentials safe for example, these credentials should never be stored in a git repository because in case in open source project or the project source code is open then people can just use the DBMS credentials to login to the database and create all kinds of have a. It is important to have a data backup and a data recovery strategy for an application production in the simplest case just having a manual daily database dump strategy is always a good thing to start with. It is important to also to use mature DBMS's which have large community and which has good tooling to help when things go.