**Important Notes**

**System & Database Admin**

**Week 1 - Intro**

**Goals of Relational Database** **–** Physical Data Independence, Query Optimization, Authorization, Distributed Independence, Concurrency

*Physical Data Independence* – Data is presented to the user logically. The user has no knowledge of how the data is stored. We can change the underlying storage mechanism without affecting the database

*Query Optimization* – Users retrieve data from the Database using SQL. Many ways to retrieve data. Some versions run faster than others

*Authorization & Authentication* – Databases are intended to share data. Not everyone should have the same access (Principle of Least Privilege). Each relational system provides tools for Authorization & Authentication.

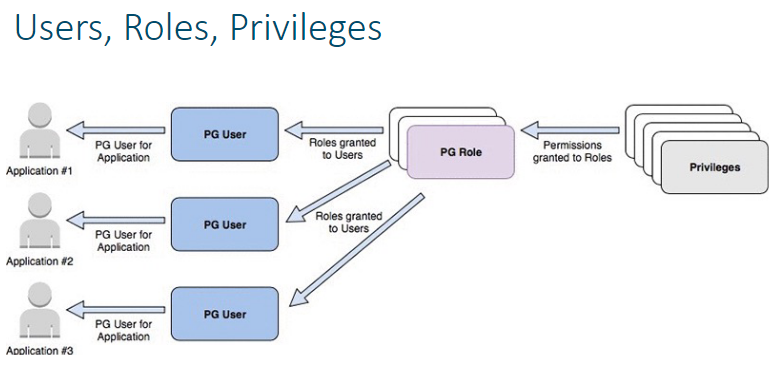
*Distributed Independence* – Database systems can be made up of multiple computers on a network. It is the job of the system to act like only a single computer is acting as the database server.

*Concurrency* – Database should allow multiple users to access data at the same time. Should however act as though a single user is on the system. Prevents users queries from interfering with each other

Responsibilities of a Database Admin

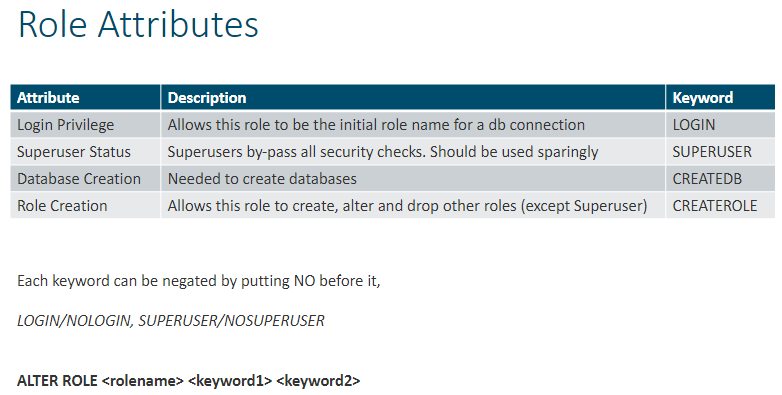
* Determine system requirements for hardware and software
* Install and configure OS and database
* Carry out audits to prevent unauthorized access
* Optimise the database through indexes, partitioning etc

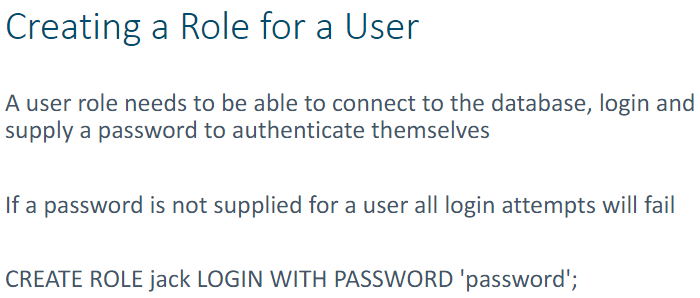
**Week 2 – Security in the Database**

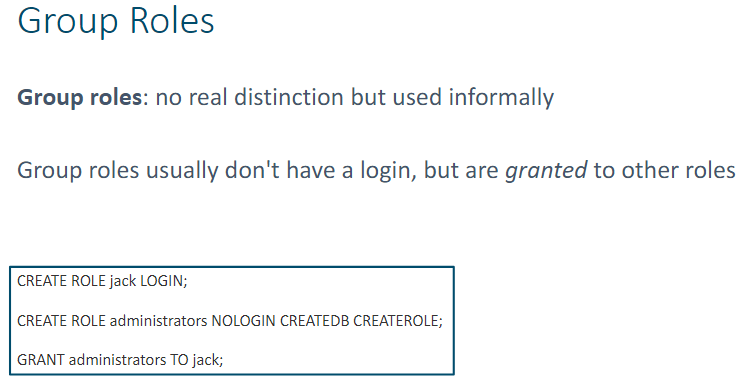
****

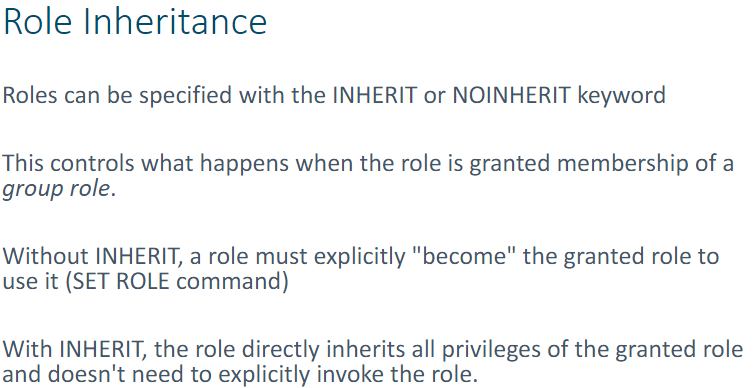
****

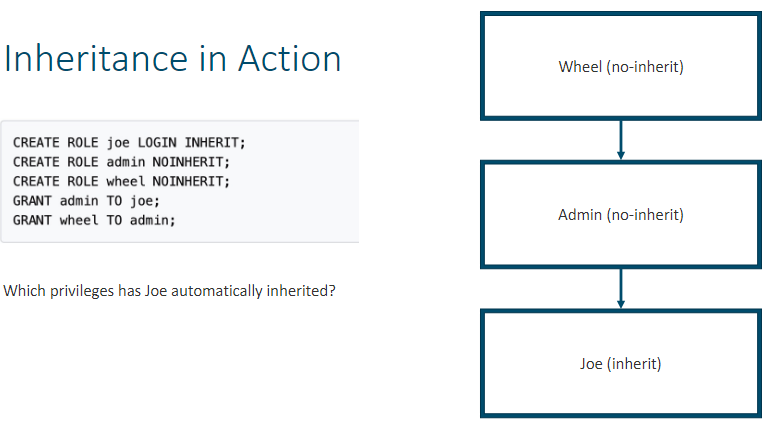
A fresh installation to Postgres comes with a single login-enabled role WITH super-user privileges











Dropping Roles

* A role may not be dropped while it has ownership of any database objects
* The object ownership may be manually reassigned one step at a time
* REASSIGN OWNED command let you transfer ownership of all objects

**Database, Schemas & Objects**

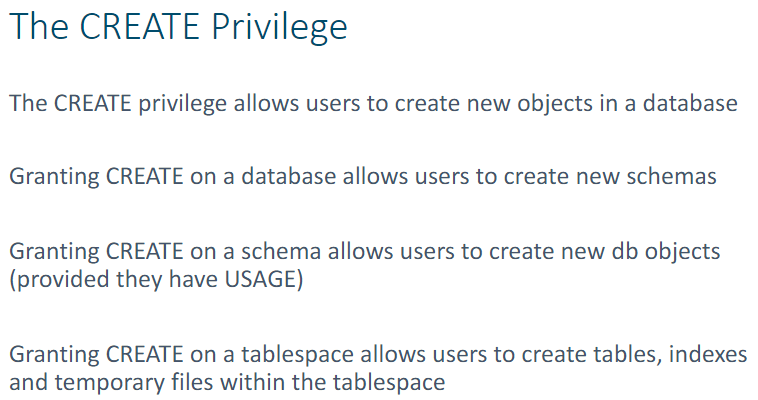
* A database consists of objects: tables, views, schemas, functions etc
* Each database has an owner

Schemas and Tablespaces

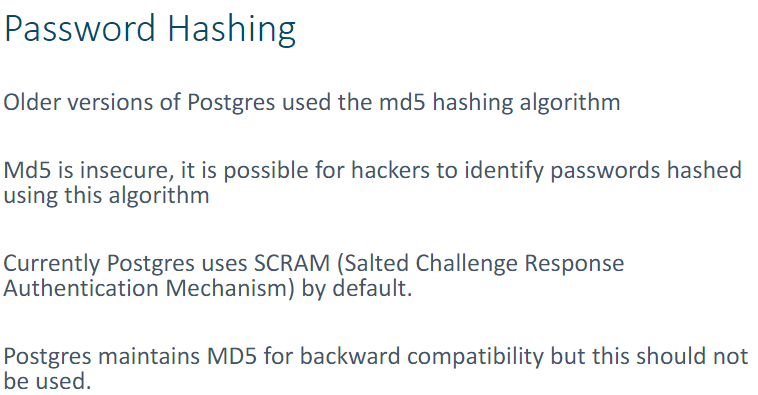
* Schemas provide a logical separation of database objects
* Tablespaces provide a physical separation of objects. Tablespaces correspond to files in the OS which hold the data in tables and indexes

Minimum Privileges

* In order to connect to a database, a role needs the CONNECT privilege GRANT CONNECT ON <database>TO <role>

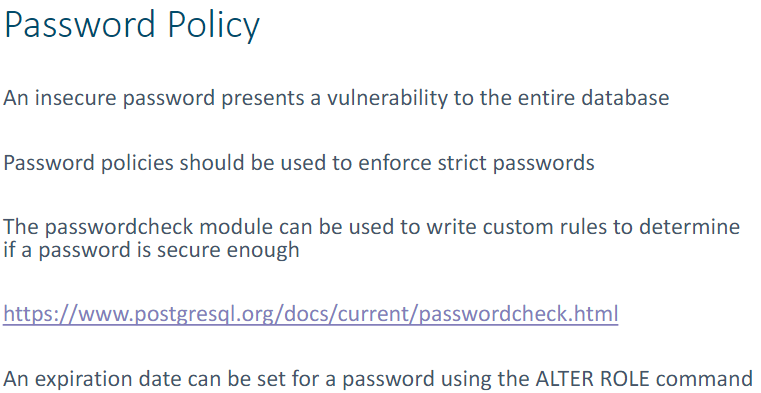


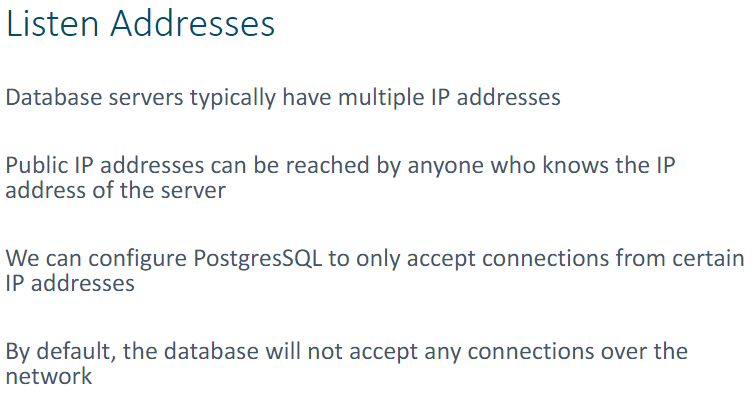
**Additional Security Considerations**

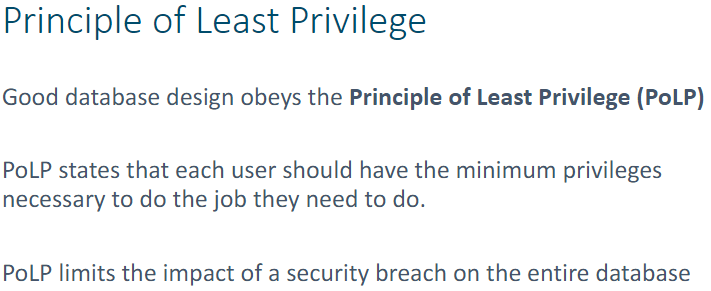


Other Encryption Options

* Password Encryption
* Column-Specific Encryption
* Data Partition Encryption
* Encrypting Across the Network
* Two-Way Encryption
* Client-Side Encryption







**Week 3 – Triggers, Auditing & Logging**

**Principles of Information Security** **–** Information Security is the confidentiality, integrity and availability of information.

* Confidentiality – Prevents unauthorized access of information
* Integrity – Protects the information from unintended/unauthorized alteration, modification or deletion
* Availability – Information readily accessible to authorized users

**Auditing** - What?

* An audit is an *inspection*
* It shows us what has happened in the system
* IT audit – examines controls within an IT group
* Monitoring and recording of database actions
  + Individual actions
  + Successful and failed activities
  + Combinations and failed activities

When & where?

* Audit times and locations should be chosen in accordance with the objectives of the audit

Who?

* Audit should be run by an independent author but will need the assistant of System Admin and Database Admin

How?

* Identify the audit topic – Set the standard – Collect the data – Analyze the data – Implement Change
* I.S.C.A.I (I Saw Carl Add Ingredients)

What it provides?

* Provides user (all levels) accountability
* Data leakage protection strategy
  + Lost/Stolen devices
  + Separation of personal/corporate data
* Ability to trace actions
  + Data sharing
  + Data access by whom
  + Regulatory compliance

Auditing Windows Systems

* Set up GPOs
* Look at logs:
  + System/Security/App logs

Logging vs Auditing

* Log files are key to the auditing process
* It tells us what was done by whom and when
* Logs are useful outside of auditing too (debugging and performance tuning)

In Postgres

* Automatic auditing can be enabled (stderror is logged by default)
* Easiest solution to log all SQL commands
* Database triggers for fully customizable auditing
* pgAudit for more control

Configuring Auditing

* Done in the postgresql.conf file

Disadvantage of Auditing

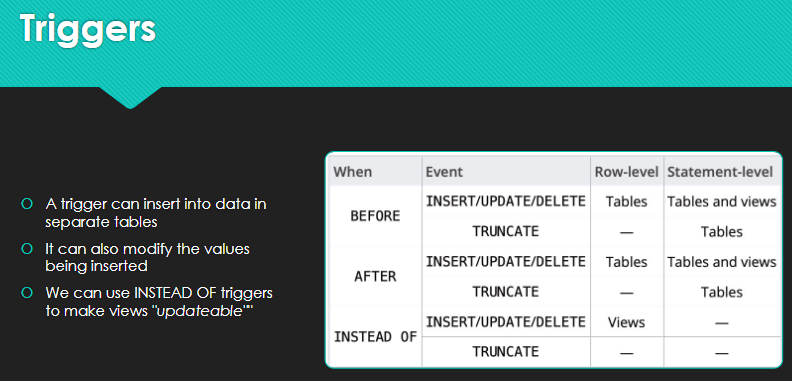
* File is WAAAY to big
* You need to save it in a secure location
* Can slow down system

PgAudit Types

* Session Auditing – Audits all actions by any user
* Object Auditing – Audits only actions affecting certain database objects

Event Auditing

* Uses database triggers
* Triggers can be run, before, after or instead of a SQL statement
* Can abort statement if desired
* Can be written in many languages, full control of what you want to do



BEFORE Triggers

* Can run before a statement is run on the target object
* Can raise exceptions, preventing the operation and altering the user to the error
* Can modify values by changing properties
* A single statement may trigger multiple triggers

INSTEAD OF Triggers

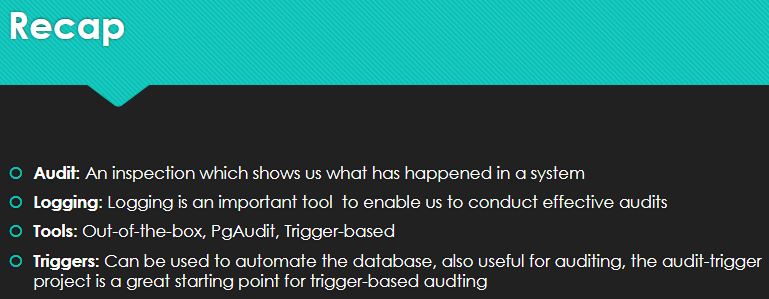
* Applied to views
* When a user tries to update, delete or insert from view the INSTEAD trigger will be fired

AFTER Triggers

* Fired after a statement has been completed
* Cannot modify values
* Can abort the operations by raising exceptions
* Multiple AFTER triggers can be set off by a single statement

Disadvantages of Logging/Auditing

* Increases the workload on the database system
* Trigger-based auditing adds complexity, and a lot of bugs can be introduced
* Too much auditing can make the output files unnecessarily big and difficult to parse



**Week 4 – Indexing**

**Query Optimization**

HDD Data Access is Slow because:

* HDD contain moving parts, friction which causes heat. There are no moving parts in SSD maintaining a lower temperature
* Data must be accessed sequentially
* Generally optimized for capacity more than speed

A database index usually holds a single column value. As well as storing the value it stores an offset allowing the database to directly access the records from file

**Read/Write Trade-off**

* An index makes certain read operations significantly faster
* Database now has to maintain an additional set of data
* Indexes make write operations a lot slower as index maintenance is carried out for each insert or update

**Types of Database Index**

Hash Index and B-tree Index

**Hash Indexing** – Get their name from the phrase “Make a hash of something”, i.e. to mess it up. Hash functions do lots of weird operations on their inputs to turn each one into a numeric output. They are cheap and easy to run. We can use this numeric output as a location at which to store our data.

**How it works?**

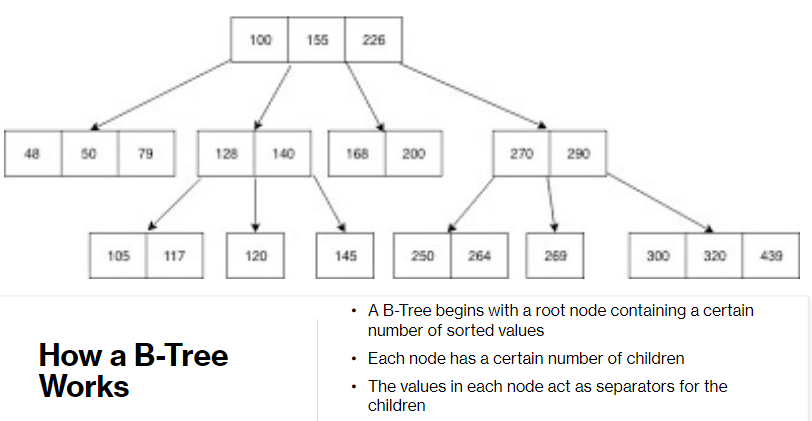
I store my index as an array with let say 100 elements. When I add a new element “John Smith”, I pass it to the hash function and store it in the slot putted by the hash function. If I check for “John Smith” in the index I can run it through the hash function and see immediately where it would be

**Issues**

Hash Collisions

* A hashing function can output any integer. If we reduce this to 100 buckets, we can divide the number by 100 and use the remainder/modulo
* If we have more data than buckets, we end up with collisions

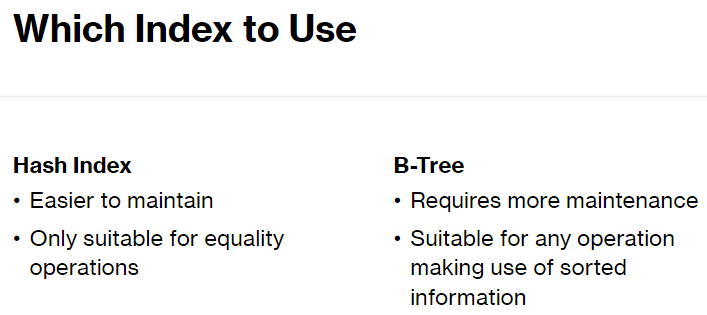
**B-Tree Indexing** – Get their name because it resembles a tree. The B refers to the fact that this data structure is self-balancing. Good way to store and retrieve data

****

**Issues**

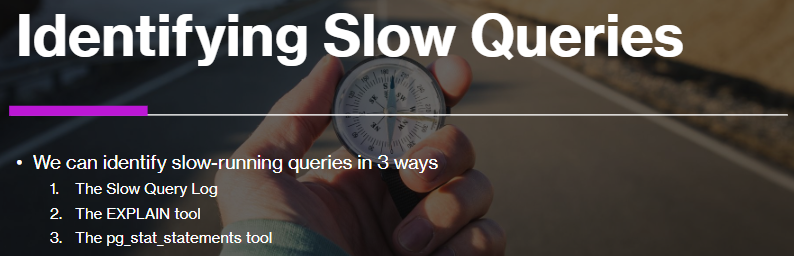
Self-Balancing

* When new data is entered there is a chance that it will overfill a node. When this happens, the node is split up and multiple child nodes are created. This ensures that B-Trees remain optimized and never grow too deep



**Fill-Factor**

This parameter allows the DBA to specify how much the database will attempt to compress the index. Ranges from 1 to 100. A large number means the index takes up less space in disk but also means it will be difficult to update the index as more rebuilding will be required. The default value is 90



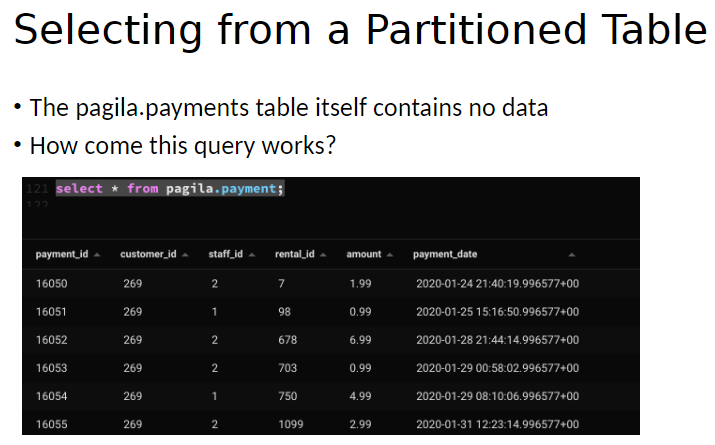
**Week 5 – Partitioning and Tablespaces**

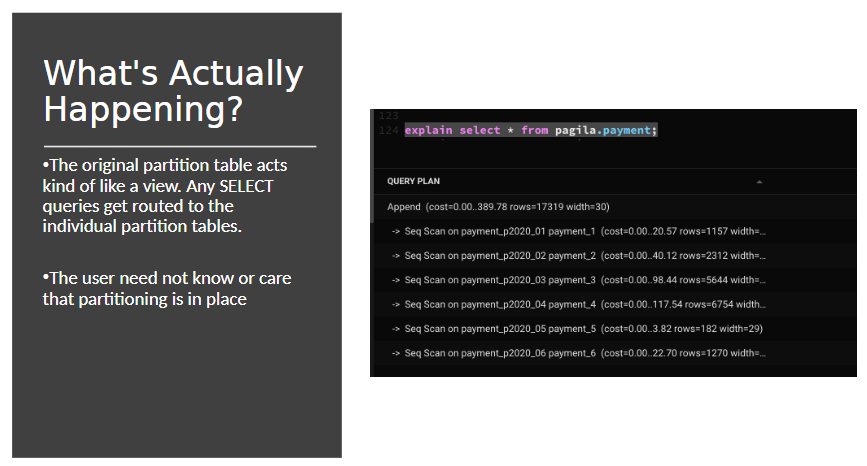
**Partitions**

* Used to break a table into multiple sub-tables
* Improve performance dramatically
* Easier to manage and archive data
* Anatomy of Partitioned Table – A partitioned table (the container) and one or more partitions (the tables holding the data)
* Postgres supports two types of partitioning: list and range
* List Partitioning – Uses a different partition for each value in the partitioning columns
* Range Partitioning – Uses a different partition for each range band in the partitioning columns

Creating a Partitioned Table

* Partitions automatically inherit all the columns, CHECK and NOT NULL constraints from the parent table





Postgres supports declarative partitioning. You declare what you want and let the system look after how it does it

**Partition Maintenance**

*Remove Old Data* – We easily remove old date that is no longer necessary simply DROP the partition. If we want to keep the data but remove it from the partition table, we can DETACH it.

Adding New Partitions – We can add a new partition declaratively. We can also attach an existing table; every row needs to be checked to make sure it does not violate the partition. This locks the entire table

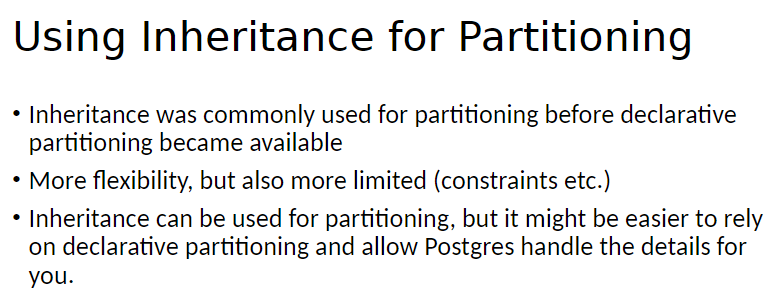
Unique Constraints – Prevents duplication of values of a given value

**Inheritance**

* Creates a link between these tables
* If I query one table, the other table is queried also
* You can query one table and not the other table using the “ONLY” keyword

**Caveats**

* Unique and Foreign keys are not supported with inheritance



**Tablespaces**

* A tablespace is a database object representing a directory on your filesystem
* Every table has it data stored in a file on disk, the tablespace property determines where that file is stored
* Unless specified all tables created is stored at “PG\_DEFAULT TABLESPACE”

Why Use Tablespaces?

* Allows you to control the disk layout of Postgres
* If the Postgres partition runs out of space a new tablespace on another partition can be used
* Frequently used data can be put on a fast disk

Importance of Tablespace

* Important part of the database if the files are not accessible the database may fail
* Should not be created on removable media but a temporary filesystem

Managing Tablespaces

* A tablespace is created as a symlink in the database root pointing to a directory on the filesystem
* To move = Stop the server, update the symlink in pg\_tblspc, then restart the server
* Make sure files are copied to the new location before restarting the server

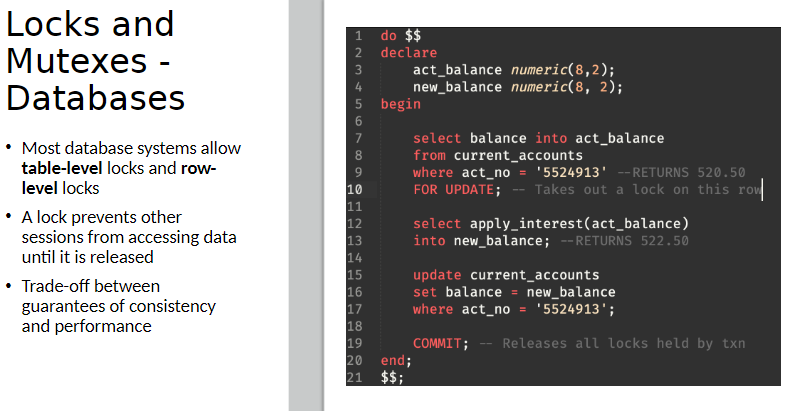
**Week 7 – Transaction and Isolation Levels**

Database allows for multiple users to connect at the same time

Shared concurrent access to resource is one of the oldest problems

**Locks and Mutexes**

* Used to protect data from concurrent access
* Similar to a library system access must be checked in and out
* Boolean variable records if resource is ready to use
* Requesters have to form a queue waiting for the resource to be available

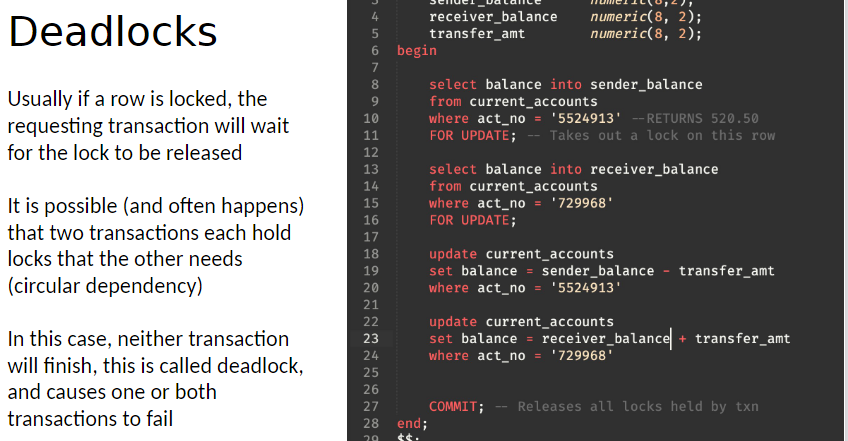


**Issues**

* Waiting for transactions to complete can cause major performance issues
* Bottlenecks quickly build up, like traffic jams

**SQL Lock Types**

* Access Share – Conflicts only with access exclusive (Read Only Queries)
* Share Row Exclusive – Prevents concurrent updates to tables
* Access Exclusive – No other process is allowed access to the table
* (A.S.A)

****