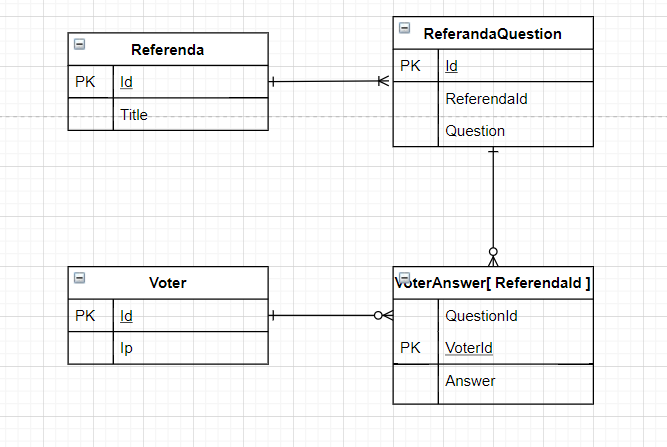
### Voting



* VoterAnswer[ ReferendaId] consist of multiple tables based on ReferendaId. Which means each referenda which have a VoterAnswer table.
* VoterAnswer[ ReferendaId].Answer is byte with 0 is ‘no’, 1 is ‘yes’
* To get the result of a question:

SELECT SUM(Answer) AS VoteYes, Count(\*) AS TotalVote from VoterAnswer[ ReferendaId] WHERE QuentionId = x

### Disk Management

* ‘disk utilization is high’ means disk almost full
* Solution:
* Check which folder consume most hard disk space [ df -h ]
* Check which process generated the files [ ps -ef | grep ‘’ ]
  + Stop the process or check any abnormal root cause
* Archive the files to external drive

### Git

* Checkout A, create A’ branch, fix the comment then commit and push
* Checkout A’, create B’ branch, merge with B, then commit and push

B and B’ will have different hash, because B’ is checkout from different branch.

### Concatenation

String is immutable and final in Java and every modification in String creates a new String object. StringBuffer is mutable means you can modify a StringBuffer object once you created it without creating any new object.

Internally "+" operation is implemented using either StringBuffer or StringBuilder

when concatenate local variables, the performance are similar; but when concatenate array of objects (Bulk Operation), StringBuilder is better because it does not created a myriad of temporary String objects to perform the concatenation

Source: <https://redfin.engineering/java-string-concatenation-which-way-is-best-8f590a7d22a8>

### Unix tools

ls -ltr | grep -l -r -w -i 'aardvark' /usr/local

### Code

###### Use ExecutorService pool:

class Helper implements Runnable {

private Socket socket;

Helper(Socket socket) {

this.socket = socket;

}

@Override

public void run() {

}

}

final class RequestHandler {

private ExecutorService executor = Executors.newFixedThreadPool(20);

private RequestHandler(int port) throws IOException {

try (ServerSocket server = new ServerSocket(port)) {

executor.execute(new Helper(server.accept()));

}

}

public static RequestHandler newInstance() throws IOException {

return new RequestHandler(0);

}

}