Backend Software Developer

Programming

1. Fail fast simply means that it will throw an exception when it identifies some change in the collection object while iterating through the collection.

Say for example, if we have an Array list with some list of elements and you have Iterator to traverse these elements and at the same time, if you are trying to insert or delete the elements, it will throw a concurrent modification exception.

It should be mitigated using fail safe. When it comes to fail safe, it will not throw any exception, even when collection is modified at the time of iteraring through the collection.This is because they will work on clone of collection instead of working on the collection directly.

Say for Example: When Iterator is applied on CopyOnWriteArrayList and Concurrent HashMap , the iterator is fail safe

2. Exception is an error event that can happen during the execution of a program and disrupts it's normal flow. Exception can arise from different kind of situations such as wrong data entered by user, hardware failure, network connection failure etc.

Whenever any error occurs while executing a statement, an exception object is created and then the application tries to find exception handler to handle the exception. If suitable exception handler is found then the exception object is passed to the handler code to process the exception, known as catching the exception. If no handler is found then application throws the exception to runtime environment and the application terminates the program.

3. It should be done reading the files in smaller parts, sort these and write them to temporrary files. Then you read two of them sequentially again and merge them to a bigger temporary file and so on. If there is only one left you have your file sorted. Basically that's the Megresort algorithm performed on external files. It scales quite well with aribitrary large files but causes some extra file I/O

4.RESTful (Representational state transfer) API programming is writing web applications in any programming language by following 5 basic software architectural style principles:

- Resource (data, information).

- Unique global identifier (all resources are unique identified by URI).

- Uniform interface - use simple and standard interface (HTTP).

- Representation - all communication is done by representation (e.g. XML/JSON)

- Stateless (every request happens in complete isolation, it's easier to cache and load-balance),

In other words we are writing simple point-to-point network applications over HTTP which uses verbs such as GET, POST, PUT or DELETE by implementing RESTful architecture which proposes standardization of the interface each “resource” exposes. It is nothing that using current features of the web in a simple and effective way (highly successful, proven and distributed architecture). It is an alternative to more complex mechanisms like SOAP, CORBA and RPC.

RESTful programming conforms to Web architecture design and, if properly implemented, it allows you to take the full advantage of scalable Web infrastructure

Database

1. A page is the most basic element of storage in SQL Server.

Of the 8192 bytes on a page, approx. 8060 are available to you as a user. If we can manage to fit our data rows onto the page nicely, they'll take up a lot less storage.

If our data row e.g. is 4100 bytes long, only a single row will be stored on a page (and the rest of the page - 3960 bytes - is wasted space). The important point is: those pages aren't just relevant on disk, but also in SQL Server main memory --> we want to try to avoid large areas of space that cannot hold any useful information on a page.

2. Splitting a table into two tables containing different columns is required in order to address database design changes, business requirements or even adding domain restrictions retroactively (for example, isolating currencies in a separate table and enforcing referential integrity via a foreign key to ensure that only valid currencies can be stored in the database)

For instance, in scenarios where our table contains large, but rarely used fields, moving them to a separate table will increase performance as the frequently used data will be stored in a much smaller table, and the rarely used data will only be looked up when required. The impact on performance caused by the occasional joining will be compensated

just by having SQL Server look up the data that's used more often in a table which requires less disk space. In general terms, JOIN costs CPU time whereas large, flat tables cost disk space and disk access time. From a performance point of view, the disk is still orders of magnitude slower than memory and since large tables usually increase the number of logical reads, sacrificing some CPU cycles on that extra JOIN might significantly improve the overall database performance.

The table was performing great with less than 10'000 entries. Now that the table has 1'000'000 entries, the query is performing very poorly. Describe how you would go about improving the performance.

3. There is no difference: both querys are completely equivalent but using different versions of SQL syntax. The database engine will handle them in exactly the same way.

The explicit join is the preferred syntax these days. It was introduced in the SQL-92 standard and is supported by pretty much every SQL-style query engine.

The query against a view that joins the same table is usually referred to as an implicit join. It is an older syntax but still supported by most SQL based query engines and is still in the standard (so is considered to be correct even though the newer syntax is preferred). It quickly fell out of favour with the introduction and wide support of explicit joins as these are usually clearer and easier to read & maintain (though the older syntax could be said to have the advantage of being more concise).

4. A stored procedure is a group of SQL statements that has been created and stored in the database. A stored procedure will accept input parameters so that a single procedure can be used over the network by several clients using different input data. A stored procedures will reduce network traffic and increase the performance. If we modify a stored procedure all the clients will get the updated stored procedure.

I used stored procs in 1 of 3 scenarios:

Speed When speed is of the utmost importance, stored procedures provide an excellent method

Complexity When I'm updating several tables and the code logic might change down the road, I can update the stored proc and avoid a recompile. Stored procedures are an excellent black box method for updating lots of data in a single stroke.

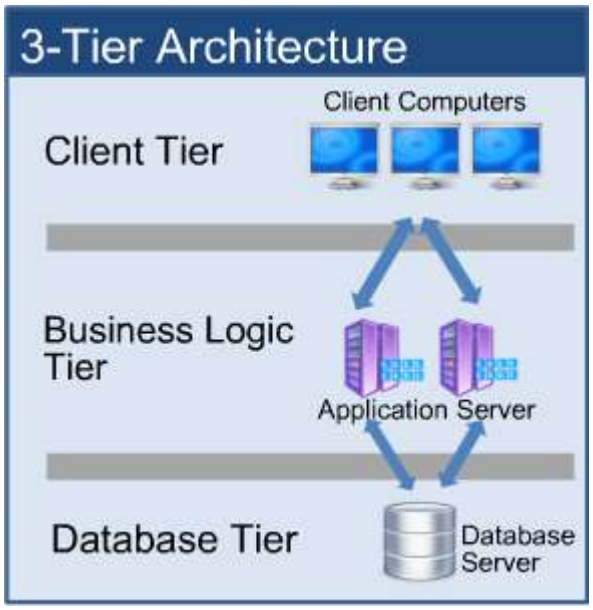
Transactions When I'm working an insert, delete or update that spans multiple tables. I wrap the whole thing in a transaction. If there is an error, it's very easy to roll back the transaction and throw an error to avoid data corruption.

The bottom 2 are very do-able in code. However, stored procedures provide an black-box method of working when complex and transaction level operations are important. Otherwise, stick with code level database operations.

Security used to be one of the reasons. However, with LINQ and other ORMs out there, code level DAL operations are much more secure than they've been in the past. Stored procs ARE secure but so are ORMs like LINQ.

Security

1. 3-Tier Architecture is the most secure and scalable solution. As client traffic is increased we can add up as many middle tiers needed to ensure performance. Three Tier architecture is also more secure because the middle layer is protecting the database tier. We need to protect database tier from direct access and need to be placed it in trusted zone and it should be only accept connections from application servers.



2. Symmetric encryption is used to share information between a set of people that all shall have access to it. Furthermore symmetric encryption is nice because it is easier to understand (less likely to mess it up) and the algorithms tend to be faster.

Asymmetric encryption is used when a large number of subsets of people shall be able to share information. Furthermore asymmetric cryptography can be used in reverse to sign documents. This is especially interesting because it allows people to certify that a public key belongs to a certain person.

In a set of 10 people, allowing every pair of people to communicate securely, requires 45 unique keys (9+8+7+6+5+4+3+2+1). And now think about of the internet instead of the small set of 10 people. It's obvious that this cannot be handled with symmetric keys.

In the real world both types are often combined. An asymmetric approach is used to confirm the identity of a communication partner and to transmit something that will result in a symmetric key. This symmetric key is then used to for performant encryption of the actual data

3. OAuth is a protocol with which a 3-party app can access your data stored in another website without your account and password. For a more official definition, refer to the Wiki or specification.

Here is a use case demo:

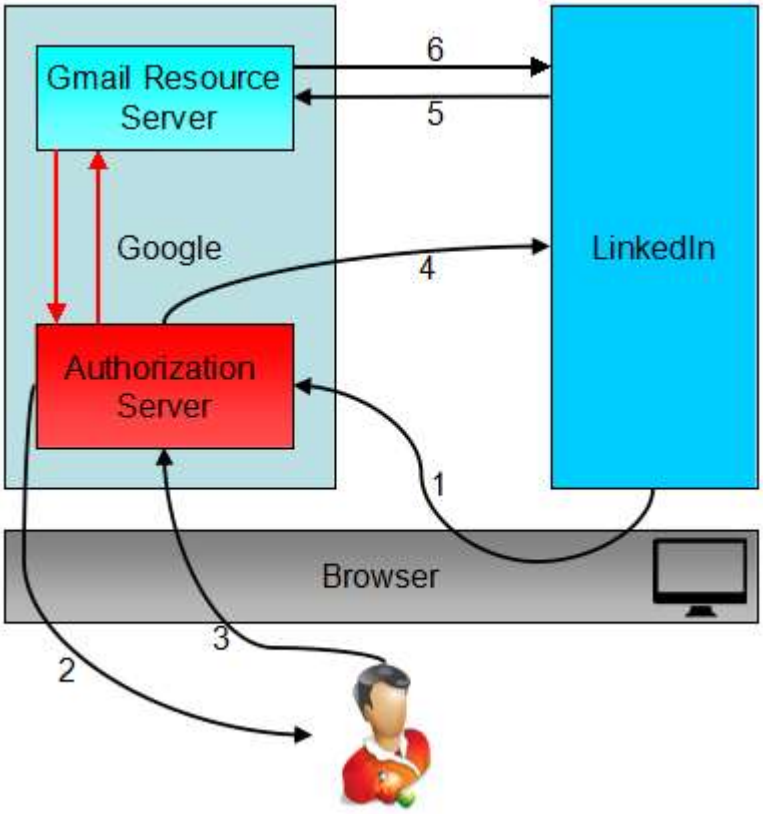
1. I login to LinkedIn and want to connect some friends who are in may Gmail contacts. LinkedIn supports this, so I click this button:

2. A web page pops up, and it shows the Gmail login page, when I enter my account and password:

3. Gmail then shows a consent page where I click "Accept":

4. Now LinkedIn can access my contacts in Gmail:

Below is a flowchart of the example above:



Step 1: LinkedIn requests a token from Gmail's Authorization Server.

Step 2: The Gmail authorization server authenticates the resource owner and shows the user the consent page. (the user needs to login to Gmail if they are not already logged-in)

Step 3: User grants the request for LinkedIn to access the Gmail data.

Step 4: the Gmail authorization server responds back with an access token.

Step 5: LinkedIn calls the Gmail API with this access token.

Step 6: The Gmail resource server returns your contacts if the access token is valid. (The token will be verified by the Gmail resource server)

Networking / Server

1. If we want to ensure the data is stored in our load balancer, then we have to terminate the SSL on the load balancer. This means that we have to install the SSL certificate into load balancer.

Another solution is to configure the load balancer to use source IP stickiness for SSL (HTTPS).

A 3rd solution would be to keep the sessions in a common database (e.g. memcached, SQL database).

2. Load testing is a type of non-functional testing. A load test is type of software testing which is conducted to understand the behavior of the application under a specific expected load.

Load testing is performed to determine a system's behavior under both normal and at peak conditions.

It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time.

Load testing can be done under controlled lab conditions to compare the capabilities of different systems or to accurately measure the capabilities of a single system.

Load testing involves simulating real-life user load for the target application. It helps you determine how your application behaves when multiple users hits it simultaneously.

Load testing differs from stress testing, which evaluates the extent to which a system keeps working when subjected to extreme work loads or when some of its hardware or software has been compromised.

The primary goal of load testing is to define the maximum amount of work a system can handle without significant performance degradation.

how would you go about performing a load test to the scale of 100'000 clients? >>

With a external Load Testing Tool such as HTTPERF. For instance:

httperf --hog --server=192.168.122.10 --wsess=100000,5,2 --rate 1000 --timeout 5

3. In short term:

1. The first thing to check is whether or not my own network connection is experiencing problems. Although I may be able to connect to some other sites, I may find that I cannot connect to certain sites.

2. If my connection is fine, check web monitoring software or services, which should have notified us when there was a web connection problem.

3. If those indicate our server is down, check data center or web host's website for current network status. They may have already been aware of the problem and have posted some information about it.

4. Next, try connecting to your server via SSH. If I can get in through SSH, it means that a service, such as Apache or MySQL, may just need restarting.

5. If SSH is not accessible, server may need a reboot. Follow the normal procedure to either automatically reboot or contact a support person to manually reboot server.

6. In the rare case that even rebooting will not resolve the issue, some data centers will setup KVM remote console control so that I can fix whatever network issues your server is having. If I find it to be some type of problem with operating system I cannot fix, they may have to re-image the machine and/or restore backups.

In large term:

I would try to understand what caused the problem (looking logs and so on) and apply an optimizacion (can be in code, network, hardware etc) so that it does not repeat itself