# 成绩表：

编号 学号 科目编号 成绩

select \*from 成绩表，查询的过程是逐条查询的，也就是一条一条查询的，从第一条成功后才查询第二条

查特定的一门科目的最高分值的前两条记录

select \*from 成绩表 where 科目编号 = 1 order by 成绩 desc limit 2;

查询每一门科目的最高分前两条：

select \*from 成绩表 as A where 学号 in (select 学号 from 成绩表 AS B where B.科目编号=A.科目编号 order by 成绩 desc limit 2)

# TCP三次握手

Client server

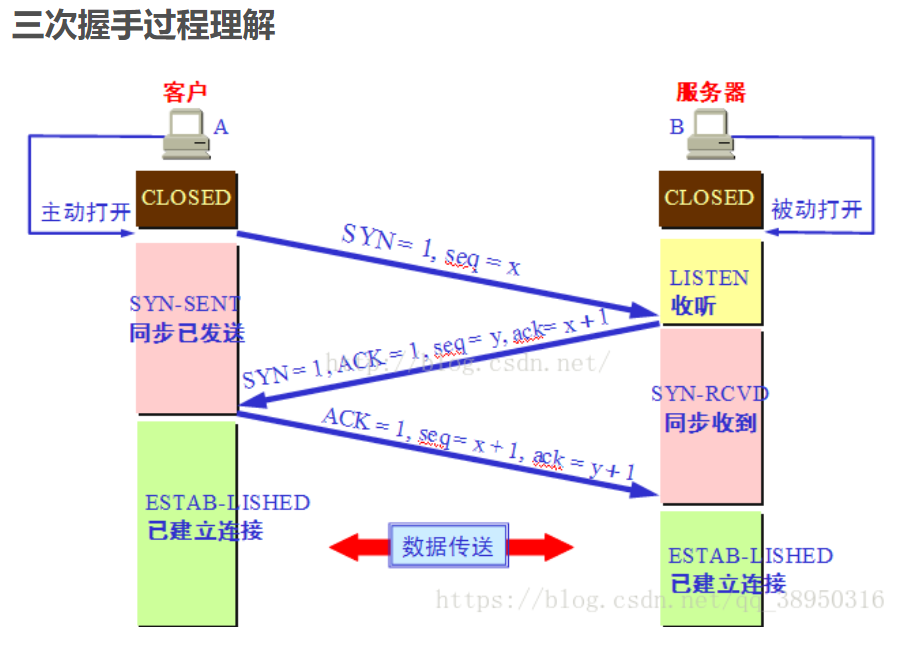
Sendsync

Send ack

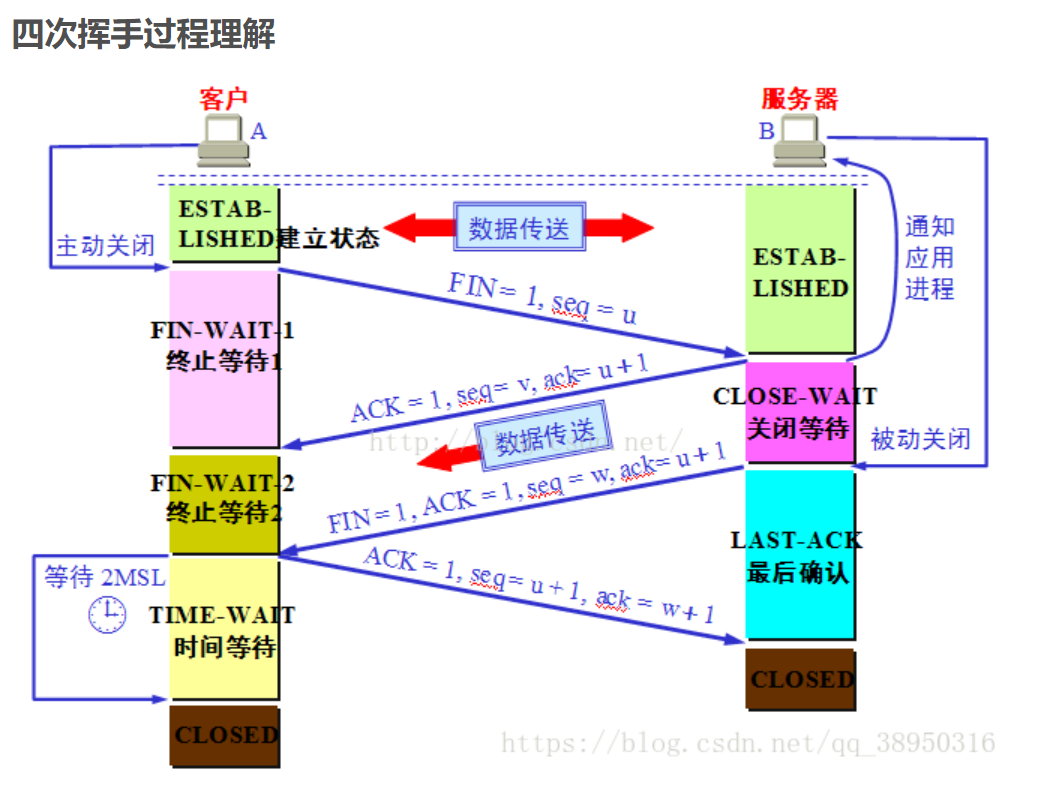
Ack(established) (established)

为什么需要三次：

客户端发出去的第一个连接请求由于某些原因在网络节点中滞留了导致延迟，直到连接释放的某个时间点才到达服务端，这是一个早已失效的报文，但是此时服务端仍然认为这是客户端的建立连接请求第一次握手，于是服务端回应了客户端，第二次握手，连接就建立了，但是此时客户端并没有任何数据要发送，而服务端还在傻傻的等候佳音，造成很大的资源浪费。所以需要第三次握手，只有客户端再次回应一下，就可以避免这种情况。



# 四次挥手



【问题1】为什么连接的时候是三次握手，关闭的时候却是四次握手？

答：因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

【问题2】为什么TIME\_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态？

答：虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可以最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME\_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

# 高并发怎么处理

集群

批量读取，减少访问数据库

缓存

Sq语句优化

# Kafka

kafka 不同 topic 的 consumer 如果用的 group id 名字一样的情况下，其中任意一个 topic 的 consumer 重新上下线都会造成剩余所有的 consumer 产生 reblance 行为。