From a Project -

Problem -

Project had a complex problem of do a lot of comparisons before finding winning ad. Application allows multiple targets to be selected by advertiser and bid on videos. Where targets are categorized in to groups, each having options to select, user can choose any number of groups and multiple targets from any group. Current implementation provides total 6 groups - Group 1 having 2 options, Group 2 having 7 options, group 3 having 5 options, group 4 having 4 options, group 5 having 9 options, and group 6 having 18 options. Group can be treated as 'and' in condition checking and options within group as 'or' in condition checking. There are 3 major problems - 1st Database design to keep advertiser selected targets as it need to be fast to find/load/compare advertiser data with input request? 2nd imagine code to check all these combination to find winning ad for given input? How many if/else statements code will have? 3rd one is groups and options are expected to change in near future. How much change in code is required if we need to add/delete group or options?

Solution used -

project used bit signature (1 and 0s ) to store and compare these combinations. For each group there is one integer (allowing max 31 options can be part of a group, 1st bit is reserved for unspecified case, where either advertiser or input request don't have value for that group), We had total 6 integer variables each representing a group. When an option is selected respective bit at that position of integer is set (right to left, 1st bit is right most). Solution deal with above problems –

1. Storage of advertiser targets is simple to store, we just need to store these integers (Currently single column having all integers separated by comma (,) is used)
2. Comparison is also faster and clean, input request is converted to bit signature with same number of groups, and then & operation is performed between input request same group to advertisers stored group (one by one, where highest bid value advertiser is on top) and if that value is non-zero, that group comparison is set it as passed. When all groups matching passes, advertiser campaign is the winning one and his ad is eligible to be shown.
3. Current implementation also allows groups/options addition removal possible by changing values in configuration file, restarting application and repopulating data.

Think of following code to do comparison, which will remains same no matter how many targets added/removed, given data has been populated with changes –

int targetGroups []; // advertiser data, groups/options, total are set using configuration file

public Boolean match (TargetBitSignature adRequestSignature){

if (targetGroups == null || adRequestSignature == null

|| adRequestSignature.targetGroups == null || targetGroups.length < 1

|| adRequestSignature.targetGroups.length < 1

|| targetGroups.length != adRequestSignature.targetGroups.length) {

return false;

}

boolean matched= true;

for (int i = 0; i < groups.length ; i++) {

if (!targetGroups[i] & adRequestSignature.targetGroups[i] == 0 )

matched = false;

break;

}

return matched;

}