When we declared our table in the ABAP dictionary, we had to use key fields, to uniquely identify every record in the table.

Table. Internal tables are a little bit different. As well as identifying unique records, you can also specify a non-unique key. This allows us to store any number of non-unique records.

We can define, different types of internal tables too.

So, first, we have standard tables. Now, a standard table, gives us the option of accessing the records, using a table key or an index.

When we access records using a key in a standard table, the larger the internal table is, the longer it will take to read the records we want, and therefore, we have the option of using a table index.

Standard tables don't give the option of defining a unique key. Which resorts in the possibility of having identical lines appearing many times throughout your table.

Now, as I just mentioned previously, we don't have to have unique records in our table. So, a standard table can hold, any number of records, that have identical lines. Now, although accessing records using a table key, can take a fair amount of time, for the standard table with lots of records, there is a flip side, in that a standard table can be filled with records, very, very quickly. Because the system doesn't have to check the table key for any duplicate records.

Now, we come to sorted tables. Now, with a sorted table, we can define a unique key. This means we can force all records to be unique in the internal table. Thus, removing any duplication of records. And like a standard table, we can access records, of a sorted table, using the table key, or using the table index. Another commonality with standard tables, is that you can use the table key to find records, and with standard tables, I mentioned this can be quite slow, the bigger your table gets, well the same sort of thing applies with sorted tables.

But I must say the performance is a lot better. Now, it is preferable to use a sorted table over a standard table. One because of the increased speed in accessing records, but also because a sorted table, sorts your records into a specific sequence. It gives you a performance increase, when accessing the data.

Now, the final internal table I want to discuss, is a hashed table. In that you do not access the records of a hash table, using an index. You will only use a unique key. Hashed tables are normally the preferred internal table when it comes to speed. And I recommend you use hash tables whenever you think you're going to have large internal tables.

See, these types of tables, use what's known as a special hash algorithm, to ensure the response times of reading records, are maintained, no matter how many rows of data, your table contains. They really are super-fast.