

Now the other type of indexes that we have the option of creating are **global secondary indexes and with global secondary indexes, you can specify a new partition key and a new sort key.** So this gives you the option of having some radically different perspectives on the data. Now, at the time of writing you can actually create 20 GSIs per table. So that's an initial limit that's set by AWS but that is something that can be increased by logging a support ticket. GSIs are easy to create. We can go to an existing table, click on "Indexes," "Create an Index" and then just specify the alternative partition and, if appropriate, sort keys. Now, with global secondary indexes, you don't actually have to specify a sort key at all but I do want to do it for this example. I want to create two global secondary indexes. The first one is I want to create a global secondary index called game scores. I want to be able to get all of the scores for a particular title. I can't do that with either the original table or the local secondary index that I created. Now, in order to get all of the scores for a particular title, what I'm going to do is I'm going to set the partition key to be game and then for the sort key, I'm going to pick a date and time and for index name, I'm going to call it game scores. Now, global secondary indexes are in some way separated from the table that they're linked with so you can allocate individual read capacity and write capacity units to the global secondary index itself. Any settings that you set on the table are independent from the GSI. You can specify read and write capacity units, or you can apply auto scaling read capacity and auto scaling write capacity. You've got that flexibility and that's important because a lot of the time GSIs have a completely different access pattern than the table that they relate to. So I'm going to go ahead and create this game scores global secondary index. Now global secondary indexes do not share the data with the table. That's the reason that they have their own performance settings and the data that's stored in the table is actually replicated asynchronously to the global secondary index. It's a separate set of data. So there is the potential that the index will lag behind data updates to the table. What it also means is **you can't perform strongly consistent reads on the global secondary index. It has to be eventually consistent and as well as that first GSI,** I'm going to create an additional one. This time, the partition key is going to be game. I'm going to add another sort key, and the sort key is going to be high score. So this GSI would allow me to query all of the data in the table looking for a particular title and then limiting it based on the score that the user has for that title. So I can use it to list the top score of the game, the highest score, the lowest score, or a range of scores. So this will be a really useful index to give me a different perspective on the data. So I'm going to the index topscores, I can accept the rest of the defaults and hit "Create." So that's finished creating I can click on "Items" I'll just do a UI refresh just to make sure it's picked up all of those indexes, change the dropdown to query, and then let's say that I wanted to list all the scores for a particular title using the top scores GSI. Well, I could do that. I could change the dropdown, select top scores, and then for the partition key, change it to Beat Saber, click on "Start Search" and that would show all of the scores in the table data set, which have a game title of Beat Saber. Now because high score is this sort key, I could also perform a subset selection out of those really easily and really efficiently using the query operation. So this illustrates the power of global and local secondary indexes. They essentially allow you to pre compute other access patterns so you generally use indexes when you know you've got a different type of access pattern. In this case, we've got our game and score table, and the normal access pattern is looking for a particular user ID and then either showing all of that user's scores or limiting it based on a particular title. The local secondary index that I created allows me for a particular user ID to show all scores or limit it based on date range and in the global secondary indexes that I created specifically the top scores one allow me to show all scores for a particular title or to limit that based on a particular high score. So showing the highest score for a particular title. So we've got some real flexibility by using secondary indexes. Now the important things to remember for the exam: local secondary indexes you're currently limited to five, they share the performance with the main table, and you can only create them at the time of creating the table. Global secondary index is currently have a limit of 20. That is increasable using a support ticket and you can create them after the fact so after the table. Global secondary indexes allow you to specify alternative partitions and sort keys, whereas local secondary indexes keep the partition key but allow an alternative source key.

Now, one final thing to be aware of about indexes, and that's something called **projection or projected attributes. Indexes can have either just the keys, all of the attributes of the main table, or only certain attributes, and whatever you decide here will be the data that's in the index. Now the reason why this is important is that if you choose to only project some of the attributes into the index from the main table, then when you're reading from that index because you're reading in terms of items, you can be reading smaller amounts of data and so it can be used to improve performance but what you need to realize is that if you ever need to read attributes from an index and those attributes or not in that index, if they're non projected attributes, then you take a huge performance penalty because it has to fetch those attributes from the main table.** So it's an important thing to understand. It's probably beyond the scope of what you need for the associate level but keep in mind when you're creating indexes, you need to pay special attention to which attributes you project.