Dimensional Data Modeling

Dimensional Data Modeling Day 1 Lecture
Data Modeling - Complex Data Types and Cumulation

Transcript:

0:01
[Music] welcome to dimensional data modeling day
0:07
one this is the very first lecture of the entire boot camp in this lecture we're going to be talking about complex
0:13
data types like struct and array you can think of array as a list in a column and
0:19
you can think of struct as almost like a table within a table so these two concepts are very important when you're
0:26
trying to build more compact data sets when I worked at Airbnb I was able to
0:31
put these things together into an array of struct that modeled the listing
0:36
availability of all airbnbs on the platform and we were able to shrink the data sets by over 95% and the power of
0:45
this can not be understated but it also has very specific use cases because if you're using these complex data types
0:52
they have usability concerns they're harder to query they're harder to work with but they are more compact so
0:58
knowing who your customer is is and knowing who is going to be using this data when you're modeling data in this way is also critically important if you
1:05
want to learn more about how to do this stuff in the cloud I would highly recommend joining the data expert
1:11
academy uh Link in the description below I really hope you enjoy the lecture today all right so what is a dimension
1:18
right A lot of times like when you hear the word Dimension at least like outside of data you think of like 3D or 2D or 4D
1:26

right you have like the three dimensions like x y and z and those things kind of like Define a space of of an area or
1:34
something like that right and that's partially like how this works as well
1:39
like uh in in the data realm uh dimensions are usually attributes of an
1:45
entity that might be my birthday might be my favorite food it might be I don't
1:51
know like the city I live in it might be uh like my name it could be all sorts of
1:57
different things like that there's going to be a lot of different things that Dimensions can refer to and a couple
2:04
things that like I want to like go over here is you have what you have these things called identifier Dimensions
2:11
identifier dimensions are what uniquely identify an entity usually this is like
2:17
a user ID or it might be something like a social security number or it could be
2:22
like I don't know there's all sorts of ways to uniquely identify things like maybe a device ID if you're uh tracking
2:28
your phones that could be another great place where you can see different device IDs that can go on uh then you have uh
2:36
attributes and attributes uh they aren't really part of they they aren't critical
2:41
to the the the identification of that entity but they help you know do other
2:49
sorts of analyses or all sorts of stuff like that and these attributes generally speaking come in two flavors the first
2:56
flavor is slowly changing so I'll give an example of a slowly changing Dimension the a slowly changing
3:04
dimension in this case is going to be like my favorite food right like back
3:09
when I was a a child right uh my my mom would always cook lasagna and I love
3:15

lasagna I thought lasagna was so good and like that was my favorite food for a long time um but then like when I moved
3:21
to the Bay Area I realized that I actually really like spicy food I like I like food that just like slaps me in the
3:27
face you know and it's just like wow like it should be like my my favorite foods now are like a little bit painful to eat because I like that pain I like
3:33
the I like to suffer while eating my food it's very delicious so anyways what I'm saying though is that my favorite
3:39
food changed at one moment it was lasagna and then it was like a spicy curry so and then maybe later on I'm
3:45
going to discover another food and it will just keep changing and keep changing and so what that means is is
3:50
this attribute is time dependent and since it's time dependent uh it's slowly
3:56
changing and like uh that's one type of Dimension and then the other type of
4:02
Dimension is fixed uh a great example of a fixed Dimension is my birthday right
4:08
uh I can't really change my birthday that's like kind of stuck right it is the day that it is I was born at the
4:14
time I was born I can't go back and like change that right there's there's other kind of other fixed Dimensions that are
4:21
the case like for example for a phone right the the manufacturer it's like
4:26
this phone was made by Apple or this phone was made by Samsung like they can't just like go back and destroy the
4:32
phone and like recreate the phone and be like okay this phone is now not made by Apple like it's it's a fixed Dimension
4:39
right you can't change that Dimension it's like set in stone so that's one of the things that's interesting when
4:44
you're modeling Dimensions is knowing whether they're fixed or slowly changing

4:50
obviously fixed dimensions are easy they are just one value that never change so like they are a lot easier to model
4:56
slowly changing dimensions are uh kind of a or of all of the dimensional data
5:03
modeling problems they're probably the the most annoying to deal with so
5:08
keeping in mind that those are the two different types of dimensions and we're going to be going deeper into this this is this this uh whole lecture gets way
5:16
deep so let's get going okay so we're going to be covering six topics today and some of these
5:23
topics we are going to be doing in the lab as well as we go more into that kind
5:30
of postgress stuff I hope uh most of y'all have that set up if not like we're going to have like another uh time
5:36
tomorrow to get people set up because we really want you to get set up on postgress at some point in the next
5:42
couple days so that you because we're going to be using the same postgress stuff for next week as well so if you
5:47
can't query stuff in that postgress instance that's going to be a big blocker for you to get a lot of value out of the first two weeks here um so
5:56
first thing knowing your data customer you have to have empathy right we're going to be going going over that we're going to talk about oltp versus oap like
6:04
it's transactional versus analytical processing then we have cumulative table design massively important we're going
6:10
to be going going over cumulative table design in quite a lot of detail in the lab today uh then we have the
6:16
compactness versus usability tradeoff which is an important consideration when you are doing your data modeling
6:24
exercises uh then we have a very fun one here called the temporal cardinality explosion
6:30

uh which can definitely happen sometimes when you're modeling uh Dimensions that have a temporal component to them we
6:37
will go into that and how there's a v various different ways that you can model that and then the very last thing
6:42
we're going to talk about is run length en coding uh run length encoding is one of the most powerful ways to compress
6:49
your data in the Big Data World especially if you're using what's called Park file format which is the standard
6:55
across industry now so we will be going deeper into each one of these and uh I
7:01
hope yall enjoy this presentation as we go deeper into this all right one of the first things
7:07
you got to know about when you are modeling your data
7:12
is who who Who's using this like who who's going to be actually trying to get
7:18
insights from this like this is the part of data modeling this is one of the
7:23
things where I think that data modeling is like one of the will be one of the hardest things for uh chat GPT or llms
7:29
to figure out because of the fact that it has this kind of exercise in empathy of understanding okay am I delivering
7:35
data to an analyst to another engineer to a model to a customer to an executive
7:40
like what is this data and how are we going to be moving it around right and how are we going to be defining this
7:46
data and so we're going to go into each one of these real quick so with data analysts and data scientists this is
7:54
where you want to have data that is pretty easy like you should have mostly uh decimal numbers uh maybe string data
8:03
types it should be very like flat like all the columns should be very easy to
8:08

work with you can just easily sum or average or do whatever you want to do with these columns you don't want to
8:14
like make the data analyst or data scientists job hard with the these data sets so that's going to be the important
8:21
thing when you are building analytical data sets and a lot of times another name for that is like called like an
8:26
olap Cube and we'll be going a little bit deeper into that in another slide
8:32
sometimes like your Downstream consumers are actually other data Engineers that are going to take your data set and then
8:38
join it with other data sets to make other data sets so this is the case especially as you get further up the
8:45
ladder in data engineering you will start to develop this stuff called
8:50
Master data and master data is data that other data engineers and other people in the company depend on and then there's
8:57
like many many pipelines that read from your data and produce other data sets
9:02
and the these data sets are going to be different and we are going to be producing one of these Master data sets
9:08
in the lab today and a big thing to remember here is that using nested types like structs and arrays is okay don't
9:16
worry about it like because that's the whole idea you're working with data engineers and they're going to be able
9:22
to understand how to work with your stru and a race because that's part of their job is they need they understand how to
9:28
work with comp data types um obviously there's other types of uh consumers uh
9:35
another big consumer here is a machine learning model um machine learning models generally speaking they want the
9:43
identifier and then a bunch of flattened decimal columns like the feature value

9:49
columns they're usually but they might be categorical values too but usually they they want it to be pretty flat but
9:54
it depends on the model some models can have can you can feed them more complex data structures but most models they
10:00
want just like identifier and then flat primitive types that's what they want so
10:06
it's kind of similar to what data analysts and data scientists expect but you probably don't have to be as concise
10:12
about the naming conventions of things because a lot of times the model doesn't care what the name of the column is and
10:19
then obviously the last one is customers customers you shouldn't that you you know you never want to give like a
10:25
customer like an Excel spreadsheet you want uh the customers to mostly be uh uh
10:34
looking at charts and looking at uh geometric patterns and you really they
10:39
you shouldn't be serving your customer like data Mo data modeling stuff right that's not obviously there are some
10:46
small edge cases there like if you're working in some sort of analytical product and you know that your customer is also an analytical person then maybe
10:53
there is some cases where you can do that but generally speaking you should give them a chart that's easy to read
10:59
that has a lot of good annotations in it and if if you have that that will really
11:04
fit the needs right that's one of the most important things to remember when you're doing data modeling throughout
11:11
all of these exercises that we're going to be doing over the next two weeks is you have to understand how the data is
11:18
being used because if you don't like you're going to mess up you're going to mess up you're going to cause a bunch of
11:24

expensive mistakes and that is going to cost the business money or you know cost
11:30
a lot of time or whatever there's going to be a lot of issues if you uh don't think about the downstream users so yeah
11:37
let's go to the next slide okay
11:43
so there's three types of data modeling uh for the most part when I think about it and they kind of exist on a Continuum
11:50
and keeping in mind here we're talking about dimensional data modeling uh fact data modeling is slightly different and
11:56
we will talk about that later but so you have oltp which is your online transaction
12:02
processing uh these are going to be uh mostly outside of the realm of data
12:08
engineering this is how software Engineers do their data modeling to make
12:14
their online systems run as quickly as possible this is where you're going to run into things like third normal form
12:21
and minimizing data duplication and you're going to have all sorts of like Linker tables and primary keys and for
12:29
forign keys and constraints and there's all these like things that go on in these places like and you're a lot of
12:36
the time in the in this world you're going to be working in like I don't know MySQL or postgress or something like
12:43
that and you're going to have these constraints and you're going to need to do a lot of joins to get all the data
12:48
that you want um on the other side you have olap or um online analytical
12:54
processing this is the most common uh data model that data Engineers use and this is
13:02
mostly used for like you're not you're optimizing for something completely different like we're not talk like we we
13:07

don't care as much about data duplication we mostly care about can we run a query that's fast and we don't
13:13
have to join a bunch of things together because when you're doing like a big join between two tables that can be very
13:19
slow and that's where if you model the data the right way then maybe you don't even ever need to do the join and you
13:25
can just group right and because olap is always going to be looking at the
13:31
population it's looking at the entire data set or or a subset of the entire
13:36
data set but usually like a good chunk of the data set whereas oltp is looking
13:42
at like one user and one person right it's it it it's already filtered all the way down to just like one entity so
13:49
that's why like these two ways of data modeling are different and they have their benefits and their strengths and
13:55
their risks and they uh they both work but like if you try to do them uh if you
14:01
try to model like your transactional systems like your analytical systems or the other way around you're going to
14:07
have a bad time and that's why you kind of have to have this middle ground and like I call you know this Middle Ground
14:13
is called like the master data middle ground and uh in this case you still are
14:19
working in the area where you want to have your entities duped and you want to
14:25
have uh very complete definitions for your enti ities and that's where the
14:32
master data kind of sits in the middle right so you have your transactional data that is like very uh kind of split
14:38
apart into a bunch of different tables then you have the master data layer and then you have the olap layer and those
14:45
those layers kind of like all work together and if you model uh like if you

14:50
have a master data table that you model as an olap table you're going to have some problems as well and so we're going
14:57
to be working um in the lab today on like how to model A Certain Master data table that will kind of combine some of
15:03
these things together and I think y'all will like it so there's a Continuum here
15:09
we're going to keep talking about this but first let's go back to mismatching needs right so in the case of uh say say
15:18
you are going the other way where you don't like you you have a transactional system and you model it like an
15:24
analytical system so you optimize for the um the population but you really
15:29
only need the single row your symptoms there are going to be your online app is going to be slow like
15:37
because it's going to be pulling in all of this data that it probably doesn't need because it's um it's modeled where
15:44
it has all these extra columns like that are probably duplicated across tables and that can be a very painful thing to
15:50
see but on the other side right if you do it the other way where you just take your uh transactional system and dump it
15:57
to the lake and then you're trying to model uh your analytics like your
16:04
transactions the pain that happens there is different the pain that happens there is what in Joins where you're going to
16:10
have a lot of joins that happen and these joins uh will if you have to do all these joins for every single
16:16
analytical query that's going to just be very expensive and you're going to have a lot of Shuffle and a lot of pain and a
16:22
lot of frustration and so that's where having good Master data can be a very very critical role for where you can

16:30
match your needs to the right spot and like you're not trying to force one
16:35
layer into the other layer and you have this middle ground that gives you essentially uh the agility to go
16:43
wherever you want right you and that's where Master data and that completeness is really really powerful um so here's
16:51
the Continuum that I was talking about so at most companies they have
16:57
production data because they have like an app of some kind like for example I worked at Airbnb for a while they had an
17:03
app and uh inside the app like you have all these datas around like hosts and
17:08
guests and listings and prices and availability settings and all sorts of
17:14
different like you know Dimensions right and all those dimensions are modeled in
17:19
a transactional way and what one one of my jobs was was to take all of those
17:25
transactional datas and merge them together to build m Master data that made it easy to understand pricing and
17:32
availability at Airbnb so that's where uh I built more like Master data which is like taking all of these kind of
17:39
production data sets that are like you know I I remember like for that for that pipeline it was like 40 there was like
17:45
40 transactional data sets that all came together to create this master data set which was one table right so that's a
17:53
that could be a big difference like where instead of having to query imagine like if we didn't have the master data
17:58
layer there and then like for the analysts it was like oh you want to learn about uh pricing okay just join
18:05
these 40 tables together and uh good luck right and so that's where you can
18:10

have a lot of value as uh as a data engineer and um but it wasn't an olap
18:16
Cube either because of the fact that like it was still normalized and we're going to talk a little bit more about
18:22
that in uh another slide where it was still duped and normalized but it still had a it was very complete
18:29
so then you have olap cubes right which is where you kind of flatten the data out and you might have multiple rows per
18:36
entity at this point and that's when you can then Aggregate and do like group
18:41
buys and like figure out like oh buy country by age group by you know device
18:48
operating system or you can do all like uh olap cubes is the area of like
18:53
another word that is often used in the analytic spaces slice and dice and if
18:58
you could do slice and dice then that is where you're going to really live a lot
19:03
is this olap Cube olap cube is the the space that data analysts and da data
19:10
scientists they love it when you give them an olap Cube because you can they will be able to do whatever analysis
19:16
they want very quickly and then uh you have one more layer here which is metrics which is essentially you take
19:21
the olap Cube and you aggregate it even further down and then you have like one number right which you could think about
19:28
it this way it's like you take the 40 transactional tables you have the master data that describes price and
19:33
availability and then the olap cube is uh a flattened uh table that describes
19:40
all the listings and all their prices and then you can imagine aggregating that all the way up where then you have
19:45
one number which might be like the average listing price for all of Airbnb and then you you go from 40 tables to

19:51
one number right and it's kind of like where you distill the data where it's like you have like all this data that's
19:57
like uh all over the place and it's all messy and then then you com then you put Humpty Dumpty together that's what uh
20:04
the master data does and then then you kind of split them out into as pieces like but but now they're all like they
20:10
make more sense and that's the olap Cube and then you smash those pieces together and that's how you get the metrics so
20:17
those are kind of the like the four layers of data modeling that are very common in uh data engineering and like
20:24
if you can understand each layer here in dimensional data modeling you will have a much better time as a data
20:32
engineer because this is a very common pattern that you'll see over and over and over again so yeah let's go to the
20:38
next slide this Continuum like like definitely try to let this sink in and come back to this
20:44
um so we're going to shift gears here a little bit we're going to talk about cumulative table design so one of the
20:52
things that can happen when you're building Master data is uh some days not
20:59
every user is going to show up because maybe they're only active once a week or something like that but the master data
21:06
should still hold on to that user you should have a complete history that's what cumulative is all about so if you
21:12
have a cumulative table what it's all about is holding on to all of the
21:18
dimensions that ever existed right up until a point you might need to filter out users that are like deleted or like
21:25
hibernated or something like that and and we're going to talk talk more about that we're going to we're definitely going to see like how that can be uh

21:33
used um uh so what we're going to do is like so cumulative tables are all about
21:38
holding on to history and if you're not holding on to history then like that's not cumulative right and so cumulative
21:45
has two the way that cumulative works is you have two data frames or two data
21:51
tables you have today's data table and yesterday's data table and those two data tables come together and You full
21:59
outer join them and the reason why you full outer join is because uh there might be data in yesterday that isn't in
22:06
today and there might be data in today that isn't in yesterday so then you can kind of merge these together so that you
22:12
can uh get the whole set and then uh then you need to coales the values
22:17
because uh they might match or they might not match uh depending on uh like
22:23
if they if they exist in both tables or not and uh then you do this coals and
22:30
and then that's what how you hang on to all of history and if this doesn't make a lot of sense yet don't worry we're
22:35
going to be covering this in the lab uh one of the big places where I saw this
22:41
in my career was at Facebook doing growth analytics so they had a table called Dim All users and what that table
22:48
was was it looked at every user's activity every day so that we could see who was daily monthly weekly active and
22:56
we can like pull in all those users and we can like look at them and we can pull that history forward every day with like
23:01
full outer joins and it's a very powerful pattern um another place that
23:07
this happens is uh State transition tracking like for example this is
23:13

another thing that I learned a lot at Facebook was they had this thing called growth accounting so how growth
23:18
accounting worked was you have today and yesterday right so if a user was active
23:24
yesterday but they're not active today they're churned right they churn and but
23:30
say they were um not active yesterday and then today they were active then that's uh resurrected but then you have
23:37
other ones where maybe they didn't exist yesterday and then now they're active today and that's new right and there's
23:43
all sorts of different like transitions that can happen from today yesterday to today and that can be um modeled and
23:52
those patterns are something that we are covering um in very uh very intense
23:59
detail in the analytics track uh like we have a a week on applying analytical patterns so definitely stay tuned for
24:07
that if you are in that track uh it should be a pretty great time this whole
24:12
cumulative table design is very common to for creating Master data that's like so you see this Dim All users table that
24:18
I was talking about like that table at Facebook had 10,000 Downstream pipelines
24:25
right so it was it it was used not just for like managing you know the immediate
24:30
growth analytics but it was also used as the source of Truth for all user data
24:37
like all users in the in the company and then it's like if you needed to join to the user table you use Dim All users
24:42
like and that's what everybody used right and that's like where you can get a lot of value out of these Master data
24:48
tables is that way so yeah but yeah like definitely uh you should only be using two data frames uh for doing this
24:54

cumulative table design because the columns in those two data frames should be essentially the same they should have
25:01
like all the same columns where one is just yesterday's data and one is today's data right and so that's how you can
25:07
keep building this stuff up and so that's how uh I would definitely recommend uh like doing this design
25:13
there's more uh in the next slide here so there is uh here's like the kind of
25:22
diagram for how a cumulative table design works right so you have today and yesterday so today is just like we're
25:29
only looking at the data for today and then yesterday might be uh all of like
25:34
the cumulated history up until that point or it might be null right yesterday might be null if you're just
25:40
starting if you if you're starting cumulation today then yesterday will be null right and it will just have no no
25:47
values obviously like how many users you hold on to and like if you're going to hold on to users for a long time and
25:53
then like they're stale and you have all this data like you know Facebook has a couple billion users billion active
25:59
users but the number of inactive users they have right is like tens of billions
26:04
right the number of people who have churned on Facebook who have made an account and left right so you don't want to like and a lot of those users are
26:10
never coming back so they're not like really relevant to your analysis so there is a point in this cumulative
26:16
table design where you do want to like like get rid of people and kind of prune them out like there's got to be like
26:21
some sort of line and I think at Facebook like they pick like if the user hasn't been active in 180 days kick them
26:27

out of the table right and that's essentially like but that all depends on the company and the business and the
26:33
requirements of like what your company is looking for so like that like and how to make this efficient like and how when
26:39
you're pulling all all of history forward because that means that generally speaking this table gets bigger and bigger and bigger every
26:45
single day right that's the general trend of this but like uh you're totally right that there needs to be other
26:51
things involved here like filtration things or something like that if like otherwise these tables can get kind of
26:57
unwelded definitely so one of the things about this right is we do full outer join and
27:03
then we coales the IDS right uh so because yesterday and today we'll have a user ID um they might not one might be
27:10
null the other they both might exist um if someone was uh in both tables they
27:16
might only exist in one or they might only exist in the other and that's where you coales and that's how you you uh you
27:23
get back to just one ID and we'll be going over all of this in the lab so don't worry about it too much much then
27:28
we need to compute the cumulation metrics right the thing that's nice about this is if you were holding on to
27:34
all of history you can see things like okay how long has it been since they were last active right we could be like
27:40
oh it's been seven or eight days since they were last active great and then you can increment it by one if they still aren't active and you can do all sorts
27:46
of interesting cumulative metrics uh when you have these tables and these they're on the same row uh and you can
27:53
do all sorts of really cool things like that and also you have this collect functionality that you can do where you

27:58
can keep adding new things to like an array of values or uh of different
28:03
changes computed stuff we'll be going over a lot of that in the lab like if this doesn't make quite a lot of sense
28:10
quite yet and then at the end of that you have the cumulated output and the accumulated output of today will become
28:18
yesterday's input like so for tomorrow it will be yesterday's input right
28:24
essentially feeds back into to this and so that can be a very that's how yesterday is defined um and we'll be
28:31
going over the kind of the pros and cons here for the cumulative table design so cumulative table design let's talk about
28:38
the um uh strengths so you get this amazing ability to do historical analysis and
28:46
you don't even need a group bu because you can collect all of the history of a user as they keep going on you can see
28:52
like when they were last active or you can have like their last 30 days of data in one row and it's just an array and
28:59
then you can see exactly like oh like they were active 8 days ago right and you don't have to use Group by because
29:06
it's all in the table it's just a select so it you can do a massively scalable uh
29:13
queries when you use this and it's on historical data which a lot of times
29:18
that those queries can be very very slow because you like say you're like oh has this user been active in the last 90
29:24
days if you look at the daily data you're going to have to query all 90 days of data and that's going to be a
29:30
very expensive query and you're going to have to group by and there's going to be Shuffle so that's going to be a very
29:35

slow query if you query all the daily data but if you move to this cumulative table design you just select from the
29:41
latest date and you can get all this data directly and obviously I was talking about the transition analysis
29:47
that's like that churned resurrected like active today inactive yesterday like uh all those different ways that
29:53
users can transition from one state to another and you can easily do those analys IES with cumulative table design
30:00
now let's talk about the drawbacks because this is not I'm not saying that this is like the the Silver Bullet solution to all of your dimensional
30:06
problems because it's totally not uh one of the things is is like because it relies on yesterday's data when you
30:13
backfill you can only backfill um each day after each other you have to do um today then tomorrow then the next day
30:20
then the next day which is not with with daily data you can run all of them at the same time and you can back fill them
30:26
all in parallel and so that one of the things that can be very painful about the cumulative table design is that
30:32
since it relies on yesterday's data you can't back fill in parallel so that's
30:38
going to be a very painful thing when you're looking at and then obviously uh pii can get kind of messy where like if
30:44
you uh you have to filter out those users you have to be intentional about removing the users that have been
30:49
deleted or inactive for too long so you have to bring in another table a lot of times to like filter those users out be
30:55
like oh this user was deleted so we have to we got to remove them from the table and so that like that can be a mess as
31:01
well like handling that kind of pii mess okay so let's talk about the compactness

31:06
versus usability tradeoff so there is a trade-off here where um
31:13
and a lot of this goes back to like oltp and olap and the differences there but the most usable tables are
31:20
straightforward they have identifier for the dimension and then they might have like uh some strings or some uh decimal
31:27
numbers or some some integers and it's very easy to use wear and group by item awesome great and then on the flip side
31:33
you have the most compact tables the most compact tables are going to have like maybe the identifier and then just
31:38
a Big Blob of bites because what they're trying to do is they're trying to compact the data down as much as they
31:44
can and they use like compression codecs and you can't even read the data at all like you have to use a you have to
31:50
decompress the data and decode it in order to do that and uh for example at
31:55
Airbnb when I when I was working there in like pricing and availability the um
32:00
in the app they actually send you down that like for the calendar the availability calendar they actually send
32:06
you down um a very compressed data set that has to be decoded in the app and
32:12
the reason why they do that is they want to minimize the amount of IO for you when you request that data because the
32:18
network uh IO is going to be one of the most expensive pieces there and then your app can just decode it and then
32:25
display the calendar to you but from an analytical purpose it's very bad because you have to decode the calendar and then
32:32
you're going to the if if the analyst has to do that decode every single time
32:37

that's not going to work and so like I had to build my own tables that were essentially that decoded
32:43
calendar and then there's a middle ground here between the most compact and the most usable which is where you use
32:49
um arrays and maps and structs and you're able to kind of crunch the data down a little bit but it's a little bit
32:56
harder to query and so you can have all three here and they all have their own
33:01
use cases where you can kind of think about the very usable tables are going to be more analytics focused and then
33:07
the very compact tables are going to be more software engineering focused and more like uh production data focused so
33:14
those are going to be the kind of the big uh differences for the compactness versus usability trade-off and who the
33:20
consumers are so the most compact is all about online systems like if you're serving an
33:26
app that has thousands or millions of users then you want to minimize data as much as possible that's going to be a
33:33
very important thing um and then on uh the middle ground is kind of that Master
33:38
data layer that's where you can really use a little bit more uh complex data
33:43
types like structs and arrays and then most usable is that olap um Cube layer
33:49
that like analysts love to work with and that's going to be your most usable and like each one of these has its own place
33:56
like the one thing I want talk about a little bit here is that middle ground layer if your Downstream consumers or
34:02
other data Engineers who are joining your table and then creating more Downstream data sets that is where
34:09
you're going to uh like really benefit from using these more complex data types
34:14

at least in my experience I've seen that be I've seen that ring true
34:19
um Okay so we've talked a little bit about struct and array and map right so
34:24
let's uh let's talk about the um kind of the trade offs of these right so struct
34:30
is essentially where you that's like almost like a table inside of a table that's the way I like to think of struct
34:36
sometimes because it's you you have a list of of keys and values but they're
34:41
all defined and uh one of the things that's nice about struct is that the the
34:46
keys and values the the data types of the values can be different uh that's one of the things about map that is is
34:53
hard right is for a map the the all the values have to be the same type they
34:58
have to all be the same data type and that can cause all sorts of weird casting problems like if you're using
35:03
map but map has some other things that are nice in the fact that the keys of a map don't have to be determined like
35:11
like you can have n number of keys you can have an infinite number of keys in a map well not technically infinite or
35:16
there's like there's some I think it's like 65,000 Keys it's like some depending on like what uh framework
35:22
you're using it's usually like either 65,000 or 32,000 or some sort of like power of two number of keys otherwise
35:29
like you can't have more than that but like you can have a lot like you have you can have more than you would ever need um and then obviously array array
35:36
is very good for ordered data sets like if they're in some sort of list and then uh inside the array each thing in the
35:44
list does have to be the same data type but that data type could be a map or that data type could be a struct so you
35:50

can actually Nest these or you can have array of struct or array of map and uh you can do all sorts of different kind
35:56
of modeling exercises like that and we uh in in our um lab today we are going
36:02
to do array of struct in the lab today so that will be pretty cool this is uh
36:08
one of the things that I worked on at Airbnb that was like probably one of the most
36:14
uh uh like impactful things I worked on was looking at how you can manage when
36:20
you have a dimension that has a Time component to it like for example like at
36:26
Airbnb uh you have a listing but then a listing has a calendar right and then that calendar has a bunch of nights on
36:32
it and then it's like so then you have what it's called like a listing night which is its own dimension in some ways
36:38
it's like its own entity the Knight itself is an entity and so uh when when
36:43
I was looking at how to model this like uh do you model this as like six million
36:49
listings and then uh or do you model this as two billion nights because we
36:54
needed to look at the next year the next the next year forward and so do you keep
37:00
everything at the listing level or do you um Compact and then you have like listing ID and then an array of night or
37:08
do you uh explode it out and then you have two billion rows where you have listing ID and KN on the same row and
37:17
then uh and there's benefits and trade-offs here that are interesting and
37:22
one of the things that can happen here is like if you use Park the right way
37:30
and everything is sorted these two data sets are actually the exact same size because of the fact
37:37
that uh the listing ID will like uh if if there's 365 Knights but they're all
37:43

sorted in in in an order then the listing ID will be compressed down because you'll have 365 IDs in a row and
37:50
that's and that's what run length encoding is all about when you have duplicates of the same value in a column
37:58
they can be smashed up together as like one ENT and then the rest of them can be nullified and just the first value it'll
38:05
be like okay we have this ID 365 times and then the all the rest of them can be
38:11
removed from the data set and that's one of the most powerful ways to compress data with par and uh yeah let's I think
38:19
the next slide is where we talk about that um oh yeah so let's talk a little bit
38:25
about like how this works we have the Badness of denormalized temporal Dimensions so if we have it be at the
38:32
listing Night level if we do a join shuffling is going to mess with the
38:37
Sorting because if if it's at the listing light level and it's all nice and sorted but if we need to do a join
38:44
spark is going to break the sorting and that can like make the compression a lot worse and because then the Run length
38:51
and coating is not going to compress stuff down as much and uh I have an example here in the next slide here so
38:59
here's run length and coding we're going to talk about it is so the big thing about run length and coding is if you
39:05
have a bunch of duplicate data then it gets nullified right so you see all
39:10
these like AC greens uh like how you have like one two three four five like you have five in a row here like you'll
39:17
see like run length en coding what it does is it essentially nullifies all of them except for the first one and then
39:23

it puts a five there and then so then like when you actually store the data you don't have to store all the
39:29
duplicates you just store okay we have this value five times and this can be a
39:34
very powerful way to um compress your data down and especially if you have
39:40
this temporal component to it you see how we have like this season value in this data set here and so you have this
39:47
temporal component that will uh kind of make it so you have a lot of duplicate um um entity
39:54
information so how do you manage that right so like let's talk about how the joins would work if we don't uh if we
40:01
don't use complex data types so imagine if we just have this like flattened out
40:07
table and it's nice and sorted just like we have here we have AC Green all nice and sorted um but then we do a join and
40:14
Spark when we do a join what it will do is we get this table now and it's all mixed up so now you see how AC Green is
40:21
not in the order it's not it doesn't have the same order as it did before right so now we only get one compression
40:30
we only have like this one value that ends up getting compressed and then all like we end up getting all these extra values in here and this can be a very
40:37
very big difference so like for example at Airbnb when I was working like one of the things I noticed about this was that
40:44
this breaking of the Sorting was something that a lot of Engineers didn't even notice because it just happens and
40:51
then uh their output data set is there but it's way bigger and but they don't even really notice it and then there's
40:57
two things there's two ways to solve this problem one way to solve this problem would be like hey if you join on
41:02
this data set make sure to Resort it and I I'm not about that I'm all about like

41:07
you should sort your data once like and then like everyone Downstream should just know how to work with it and that's
41:14
where uh if we keep everything in an array right so imagine instead of having
41:19
all of these uh player names and Seasons broken out like as rows but instead we
41:24
had one row with a player name and then all of the seasons then what we can do is we can
41:31
join on player name works great and then after the join we can explode out the
41:37
seasons array and then it keeps the Sorting because then all of the all of the rows that are associated with that
41:42
player name get kept together after the explode and um if that like doesn't uh
41:49
like make sense uh don't worry we're going to be covering a lot of that in the lab today we're going to be using this exact data set in the lab today and
41:56
covering how we can keep the sorting and how this ends up working together so
42:02
that's essentially how Shuffle can break things and that's where like if you model this data with complex data types
42:09
then you don't have to tell your Downstream consumers oh if you do a join make sure to Resort it because it will
42:16
already be sorted for them they will like in all of their queries it will just work and that is where like that's
42:22
where leveraging this especially for master data can be very very powerful f for your Downstream data Engineers who
42:29
are working on your stuff cuz then then you aren't giving them a landmine to step on of like when they create their
42:35
Downstream data set and it's like a lot bigger than like they were expecting because you modeled your data the right
42:42
way so that they they can't make that mistake and so that's a big thing that I've noticed I've seen this happen a

42:48
couple times in my career now with these like temporal Dimensions that you want to be careful with so that's a big thing
42:54
to remember about spark Shuffle is like how it messes up The Ordering of the data it's a good thing because it makes
43:00
uh the join fast but you got to be careful right and like spark Shuffle is a big thing to to watch out for all
43:07
right everyone so that is it for the lab that's what we have or for the
43:12
presentation today

English (auto-generated)

All

From the series

From Data with Zach

Data science

Software Engineering

Presentations

Related

Recently uploaded