<u>Dimensional Data Modeling</u>

Dimensional Data Modeling Day 3 Lecture
Data Modeling - Graph Databases & Additive Dimensions

Transcript:

0:00
[Music] [Applause] welcome to dimensional data modeling day
0:07
three lecture today we're talking about graph data modeling graph data modeling is significantly different than
0:12
dimensional or relational data modeling in the fact that it is more relationship focused and less entity focused so if
0:19
you're looking at how things are connected that is going to be where graph data modeling truly shines but it
0:24
comes with a tradeoff where you don't really have as much of a schema around the properties so the schema most of the
0:30
graph data modeling is very flexible it just has like a vertex and properties or an edge and properties the schemas are
0:37
very very flimsy and flexible and so I worked a lot with graph data modeling
0:42
when I worked at Netflix we built graphs that had 70 80 different types of things in them so that we can understand how
0:49
everything is connected and you might end up ultimately loading the stuff into like a graph database but in today's
0:54
lecture we're going to be talking about just the data layer and how to build a data agnostic graph data model and I
1:01
hope you enjoy the lecture today and if you want to learn more about how to build graph data models in the cloud with hot Technologies like Iceberg and
1:07
trino definitely check out the data expert Academy in the description below okay so today we're going to be talking
1:14
about a couple things here we're going to be talking about uh additive versus

1:19
non-additive Dimensions uh which is a perplexing little concept uh that I'm
1:25
definitely going to help illustrate here then we also have uh the power of enums
1:31
uh enums and enumerations we we covered a little bit about enums in day one and two remember that like scoring class
1:37
right um I need to mute people I'm getting some Echo I think it was like thar can you uh mute yourself uh anyways
1:45
uh and then we're also going to be talking about when you should use flexible data types like in this case flexible data type is only counts for
1:53
map map is the only one that's flexible struct is not flexible because you have to Define all of the keys and all of the
2:00
data types of everything in the struct and array is also array is kind of flexible because you can like add more
2:05
things to it and but like usually array has to be like an array of string or an array of a specific type and whereas map
2:12
can have like uh keys and values that are all kind of like you can have as many as you want a lot of times map
2:18
string string is a very powerful thing because it's like agnostic and you can create all sorts of interesting data
2:24
with that uh we're going to go deeper into all of these topics so don't worry about it but first off let's let's get
2:29
into the uh to additive versus non- additive Dimensions
2:34
okay additive versus non-additive so this mostly is when
2:41
you're talking about how to count stuff you are able to essentially take
2:49
the sub counts and you can add them all up like all the all if you take all the
2:54
subtotals and sum them up you get the grand total and you get the correct number so like for example in uh the us

3:02
if you take all the one-year-olds plus all the 2-year-olds plus all the threeyear olds plus all the do to all
3:08
the 115 year olds da da da and you add all of them up and like you take and you just and you just know the size of each
3:14
group you know that you can just add all the sizes of each group together and then that is the correct number for um
3:22
uh the grand total but not all dimensions are like this uh for example
3:27
like uh I like this example around cars so say you have the number of Honda
3:32
drivers the number of Honda drivers is not equal to the number of Civic drivers plus Corolla drivers plus a cord drivers
3:40
and um the reason for this is because what if what if I own a Civic and a
3:49
Corolla right then the problem here is I'm going to be double counted because I'll be a Civic driver and a Corolla
3:56
driver I'll be in both buckets so but like you don't want to double count me that I I I'm just one Honda driver I
4:02
shouldn't count as two Honda drivers there whereas like if you slightly tweaked this right in said instead of
4:08
the number of Honda drivers you say the number of Honda cars then you're fine because a car can't be two cars at the
4:14
same time but a driver can be two a driver can drive two different cars at the same time or in the same day and so
4:21
like that's the essence here of additive versus non-additive
4:28
is can can a can can a an entity have two dimensional values at the same time
4:35
over some sort of time frame and that is going to make more sense as we go deeper
4:40
into this okay so the essential nature of additivity a dimension is additive
4:46

over a specific time window if and only if the grain of data over that window
4:52
can only ever be one value at a time this is like I know this sounds very like mathematical very like I even got
4:58
the if and only if in there you know the bu conditional statement so um for example like say we go back to the Honda
5:05
example so on a small enough time scale you could say that the number of Honda
5:12
drivers is equal to the number of like Civic plus Corolla plus that right
5:17
because imagine if you looked at just one a slice of one second right and then
5:22
in one second like I can only be The Driver of one car I'm either going to be a a Honda or a Corolla driver or a Civic
5:30
driver but I'm not going to be both Because unless I'm like able to like switch cars in one second which is
5:36
probably not going to happen so uh on a small enough time scale uh that
5:42
Dimension is additive but once you get it to a big enough time scale it no longer is additive and so that's a big
5:48
thing to remember here is like how these Dimensions interact with time that is a
5:54
big deal on like how how to uh think about the nature of these as you
5:59
Aggregate and roll these up so that's a like this problem comes into a place in
6:05
a couple places um one of the one of the key things that oh yeah it's on this slide cool so um how does additivity
6:11
help so how additivity helps is if you have the subtotals and you know if you
6:17
know the dimension is additive and you have the subtotals you can just add them up and you don't have to use count
6:24
distinct cuz if you have to like if you're trying to do it uh the the grand total and the the dimension isn't
6:31

additive then you have to go back right you have to essentially do a count distinct over the entire data set as
6:37
opposed to the subtotal data set you have to essentially go back you have to go down a layer you can't you can't have
6:43
that partially aggregated data you have to go all the way back to like the row level data to get the grand total
6:49
because you don't know how much of the data is being overlapped between like you know the the Civic and Corolla
6:54
drivers is it one is it two like how much how much overlap are we talking and so one of the things that's interesting
7:01
here though is this really only applies if you're aggregating with a count
7:07
because if you're aggregating a sum like for example if you instead you're saying the number of miles driven by Honda
7:16
drivers then you're fine because you can say the number of miles driven by Honda drivers is equal to the number of miles
7:22
driven by Civic drivers plus the number of miles driven by Corolla drivers and it all adds up it all makes sense right
7:30
because I can't drive a mile in two cars at the same time unless I'm like I mean
7:36
unless I'm like freaking Superman and I'm like like I'm like steering one car with my feet and like steering the other
7:41
car with my hands right and doing some crazy like I I I mean for all intense of purposes we're going to say
7:46
that that's not possible right so really additivity and this this nature only
7:52
really matters for count but because it matters for count
7:58
it also matters for other types of metrics as well right and the big other
8:03
type of metric that it applies to besides um uh count metrics is when you
8:09

have a ratio metric right so imagine you have uh the number of miles driven per
8:17
driver then you're back to a metric is non-additive right and you're going to
8:23
have to like deal with it right if you compose these metrics together then you're still back into the nasty
8:30
additive versus non-additive thing where you have to worry about count distinct versus not count
8:35
distinct so remember the key thing here
8:40
when uh you're looking at these Dimensions to know if they're additive or not is can a user be two of them at
8:47
the same time in usually like like a rule of thumb here is like can a user be two of these at the same time in a given
8:53
day and if they can then it's not additive and you have to deal with it spec you have to deal with it you have
8:59
to do deal with it differently you can't do like the partial aggregations like you can with other dimensions good news
9:05
is is like most dimensions are additive uh like I would say there's only a couple usually these dimensions
9:12
are going to be like a like I like to think of it as like a tool right it's like a a users tool right another great
9:18
example is like you can't say the number of users on your app is equal to the number of iPhone plus Android users
9:24
because of the fact that there's some people out there that are on both iPhone and Android and so it's like you see how
9:29
that's like a a user's tool just like a car is a a user's tool and you can it's
9:35
something that a user uses and you can aggregate along that Dimension and that's why you get two values so that's
9:42

definitely something to think about when you are uh doing these aggregations and like I like this was such a big problem
9:48
at Facebook that like we actually built Frameworks around this so that people didn't have to worry about this problem
9:54
anymore and that like if they wanted to do a non-additive aggregation they can do it uh like efficiently through our
10:01
framework and that was uh we built this thing called Milky Way and like yeah that was a painful little process that I
10:07
noticed that I uh like this problem will show up so definitely just like do more
10:13
research on this we're not going to cover much more about this in the lab but this is an important topic in terms
10:19
of analytics and growths especially like user counting uh in in growth analytics
10:24
this is a very important topic to to consider all right so we're going to shift gears kind of completely here so
10:32
we have um enums right so we covered enum in the last class remember the NBA
10:39
scoring class like uh like Star good um average bad like we had those kind of
10:45
different uh enumerated fields that we were working with uh there and one of
10:51
the things that uh you'll notice with enumerations is there is a limit like to
10:57
how like like how much you want enumerate like and I love country country is always a great example
11:03
because country is one where like uh some Junior data Engineers might be like yeah we should throw country in an enum
11:09
and it's like n dog that's too many that's too many that's like 200 like I
11:15
always think of Like A good rule of thumb here is like is it less than 50 right if it's less than 50 it's probably
11:21
a good idea right uh that's where um we're going to be covering how enums

11:26
could be very powerful in different data architecture in this um class as well let's go to the next slide so why should
11:33
you use enums so a couple reasons one is you get built-in data quality like for
11:39
free essentially because if you model a field as an enum if you get a value that
11:44
doesn't the fit in the enum then the pipeline fails usually the pipeline will
11:51
fail because it's like uh we're trying to cast this value to this enum and it doesn't exist so you get very good data
11:57
quality in your pipeline like like automatically um other things that you can get is you can get built-in static
12:04
Fields cuz sometimes there's fields of an enum that are just static uh that you
12:11
need to know about right uh for example like in unit economics uh like I had an
12:19
enum for all of the different line items and one of the static Fields was was this a revenue or a cost and that is
12:27
just like okay each line item fees are a revenue right and coupons are a cost and
12:32
there's like all sorts of different like uh like like static fields that you can also apply to enums and the good thing
12:39
about those is you can bring those in in a more efficient way than even like uh
12:44
like a like a join like a broadcast join because of the fact that they're just like in the code already and it's like
12:50
you can just bring them you can ship you can ship the enum and all of the static Fields with your code so like it can be
12:57
very very powerful way to do it and another big thing is is you get built-in documentation because now if you if you
13:03
have a list and you know what all the possible values of that list can be that
13:10

is another really great thing because that shows people like hey I know what are the possible values here and I know
13:16
what I can do and um whereas like if it's a string that you don't get that right you don't get the the possible
13:22
values when it's a string and so um that's another great thing you get is built-in documentation so enums can
13:28
really uplevel your data quality uh keeping in mind that you don't want to just throw it on anything like you want
13:34
to make it so it's like if it's something that has 50 or less uh possible values that's like a good rule
13:41
um like for example one of the enums that we had at Facebook was uh in notifications there was um we had this
13:49
thing we had this notion of a channel a channel was like how we sent someone a notification which was like either SMS
13:56
email or push and uh there was like one more there's also one that was like logged out push which is like a
14:01
different Channel cuz some people um like in De in the developing World they actually share phones and so they get
14:08
like Facebook actual things set up where you can get notifications when you're logged out for a different user uh
14:15
because uh if you have someone who's like sharing a phone and that actually like really helped increase engagement
14:21
uh um in notific or like in for Facebook in the developing world but like when
14:27
you think about that from like the United States perspective you're like who that's crazy that's like a crazy uh like we would never do that in the US
14:33
right that sounds like a privacy concern or something like that but like different countries have different expectations there especially like when they're like sharing a phone but uh so
14:40
anyways enums enums are great and they can model things really cool um so one

14:45
of the amazing things that can happen is with enums is they can be a good uh
14:51
value for your sub partitions so if say like in notifications again like when I
14:58
was processing notifications like you don't want to process all the notifications together because it's like a lot so instead of just partitioning on
15:06
date we partitioned on date and uh channel so there was also the channel column there which was and that that's
15:13
why enums are so powerful because if you have the exhaustive list of all of the
15:18
possible values for a partition then you can build a pipeline that covers
15:24
everything and it waits and actually gets all the possible values and it doesn't miss any data and that could be
15:30
another really great thing to remember when you're like working with the stuff so when I was like doing like my
15:35
notifications ddop work uh that that was the case where we we essentially dded
15:41
notifications by channel so that we could do them more in parallel as opposed to having to ddop all the
15:46
notifications together and that can be a very powerful thing especially if you move up a layer and you are pushing this
15:54
into uh like the logging layer as opposed to in your ET L's because then
16:00
what you can do is if y'all have ever heard of thrift Thrift is a very powerful thing that you can use so
16:06
Thrift is a way to manage schema in your logging as well as your etls and you can
16:12
share the schema across the board and you can share enums you can share all sorts of different things through Thrift
16:19
and uh that that's a very powerful way so that like the logging and the pipeline are on the same page in terms
16:27

of what the enum values are because because sometimes the enum values will change right you'll add one right or you
16:32
might even remove one right and depending on the the case and the circumstances so that's a very powerful
16:38
thing to think about uh this other big thing I I'll share these slides after this but like uh I came up with this
16:45
design called the little book of pipelines that is a very uh it leverages
16:50
this concept of enumerations uh to a great extent and
16:55
this design I've used a bunch of times I use this design at Netflix to do like the asset inventory stuff and at Airbnb
17:02
to work on unit economics essentially like this enumeration little book of pipelines uh pattern if you ever have a
17:12
a a pipeline that you're working on that really pushes that V the variety V where
17:17
there's like wow like this pipeline has 50 Upstream data sets like how do I manage all of these data sets that's
17:23
when you want to group those data sets into an an enumerated group so then you have like uh all the groups that could
17:30
possibly run and then that is how you keep track of all the possible values of
17:35
stuff and that is uh what the little book of pipelines actually ends up doing and uh we're going to talk about that
17:42
more on the next slide here let's talk about how this works um so you have uh a
17:49
source all these different sources here so imagine like these s like there was not just three you can have n number of
17:55
sources this essentially Works uh kind of infinitely so what this does is you
18:01
have an enumerated uh value right um that has some sort of
18:07
uh um uh set of values in here like for example at Airbnb there was like maybe

18:14
fees and coupons and um infrastructure cost and all the different things that impact profitability and those all go
18:21
down the the side here and you have all the source data that you get that from and then how it works is this enum gets
18:28
pushed into these Source functions all it gets shared with all the source functions and then these Source
18:34
functions what they do is they map the data to a shared schema and the way they do that is this shared logic ETL will
18:41
call The Source function and then it will map it over into the shared schema
18:47
and then after that the little book of enums also has the data quality checks for that enumerated value because your
18:53
data quality even though now the schema is shared that doesn't necessarily mean that the data quality check should be
18:59
shared because like what is anomalous for fees and what is anomalous for coupons is probably different and so
19:07
that can be a very big difference that you can think about and that's where this little book can be very powerful
19:12
because it what it stores is it stores all the enumerated values all the data quality checks and then but then it
19:18
makes it so that like even if you have a shared schema you can have customized data quality checks on each partition
19:24
depending on whatever's in that little book and then after the data quality checks pass then you have your final
19:30
subp partitioned output and this partitioned output will be have like it'll have a date partition and then it
19:36
will have a subpartition which is this enumerated value whether that be like uh like you know fees coupons or something
19:43

like that like some sort of enumerated list of things that could possibly be in here so this pattern works at like
19:50
especially if you have a massively large ETL that like covers a lot of different topics and like and they all need to
19:57
come together into one shared schema this is a very very powerful design that
20:03
I've noticed that really makes it easy to keep track of things because then if you need a new source you just add another value to the enum done easy
20:11
right and then and then you get your DQ checks and then you just have to add the source function and then everything else will just work and then it all is very
20:18
cohesive and integrated and I've noticed that like when you implement this pattern what happens is the the quality
20:26
of these pipelines goes up dramatically and you get uh and you also get this
20:31
documentation for free because now people can be like oh what are the possible partitions in here and you just
20:37
query the little book here and you're done right and it's pretty great um so how that little book actually is
20:42
generated right so it's usually going to be an En an enumeration defined in either like python or Scola but then
20:50
what you you have a job that essentially turns that enumerated uh list into a
20:55
table and that table will be tiny it'll maybe maybe maybe like 20 or 30 rows or 40 rows however many values in the enum
21:02
you have and then that's how you can share it between like your DQ checks and your Source functions where in the
21:09
source functions you pass it as Python and then in the D the DQ checks you pass
21:14
it as a table and you join and that's how you get like your thresholds for like week over week month over month
21:20

stuff like that that uh and keep in mind that like uh there's a lot going on with this pattern and so like I highly
21:26
recommend that y'all check out this example there's a something it's it's a it's an open source uh library that I um
21:32
or open source like GitHub repo that I I shared that y'all can check out after this um presentation uh so let's uh go
21:42
into this a little bit more so like what are the use cases for this little book of enums so I've seen it happen a couple
21:49
different times right earlier I was talking about uh Airbnb they have unit economics where it has all these
21:55
enumerated Fields fees coupons credits insurance and infrastructure cost taxes uh referral credit there's so many more
22:02
and then uh Netflix also with the infrastructure graph that I worked on in that case you have like applications
22:07
databases servers codebases cicd drops you just have this like big list of things that you need to put together and
22:14
that's kind of the idea behind this this pipeline design is that if you ever have this like I call this like large scale
22:20
like data integration when you need to like just bring tons and tons of data together into one table and then also
22:26
this I saw this at Facebook as well for the family of apps where in that case you have like Oculus Instagram Facebook
22:31
Messenger WhatsApp and probably threads now I don't know if yall are on threads you should definitely follow me on threads I I post some spicy memes on
22:37
threads so uh those are kind of the um use cases right that I've seen in my
22:45
career and they're they've all been very impactful they've all like changed the company in some way or another so
22:51
definitely uh I highly recommend checking this pattern out because it can be very very

22:57
powerful okay so one of the things that I was talking about in this pattern though if you remember back here where
23:03
we have that shared logic and that shared output what does ends up being is you end up having a pattern where all of
23:10
the all of the data ends up being in the same table um so how do you model data
23:17
from disperate sources into a shared schema So It's Tricky uh and the main
23:24
way that you do this is with what I call a flexible schema because like you don't want to just like bring in all the
23:29
columns from all 40 tables and you just have like one table that has like 500 columns where most of them are null all
23:35
the time because that's like not a very usable table right and you don't want to work with stuff that way that's
23:41
definitely not the way that you want to like bring the data in what you want to do instead is you want to work with what
23:47
I call a flexible schema and this flexible schema is where you're going to
23:52
be leverage a lot of map map data types here it's going to be the big one that you use to do things and a lot of this
24:00
stuff is going to overlap a lot with uh that kind of graph you know that graph database I was talking about because the
24:08
enumerated list of things is very similar to like in uh in the in the
24:13
graph database world we have this thing called a Vertex type right and that is very similar because it's just like an
24:19
enumerated list of types and so these these concepts are integr integrally
24:24
connected and so like let's talk more about flexible SCH SCH
24:30
so uh flexible schema is awesome because if you need to add more things you just
24:37

put it in the map uh which is great you just throw more columns in the map like
24:42
if more columns appear you can just throw them in the map uh that's awesome
24:48
that's a great um and then the map can just get bigger and bigger and bigger uh there is a limit right I think Maps like
24:55
in like spark and that world I think they can have like 65,000 Keys like so
25:00
so there is a limit there actually is like a fundamental limit but like I've never had that many keys in my map I don't I've maybe had like a thousand
25:07
keys in my map at one time but like never like 65,000 that's like a it's an aggressive amount of keys so that's like
25:13
and that limit comes from the like the bite length or some sort of java limit right um so you can manage a lot more
25:20
columns when it's flexible right CU you can just add another key to the map and you don't have to run alter table it's
25:26
great uh you don't have all these n columns because a lot of times like with flexible schemas it's just going to not
25:32
be in the map and so it just doesn't show up um another good thing that I've seen a lot of like companies do is they
25:40
have this column called like other properties where if you want to bring more columns into the table that are
25:46
probably not going to be queried that much so you don't actually want to do the modeling of them because like why bring them in and get all these weird
25:53
questions about like what does this column mean like what they do instead is they just throw it into other properties
25:58
and then they call it a day uh really great um obviously flexible schema if
26:04
like it was all good then we would just be using flexible schema for everything uh it's not all good um a very big
26:12

drawback is compression so if you throw everything into a map uh maps are going to be uh pretty much the worst
26:20
compression of all the data types um the only thing that might be worse is Json
26:25
Json and maps are both really terrible they both are like very huge and they don't they don't squish down very well
26:31
because and the main reason for that is that you got to think about it like where um in the old world like if you
26:38
have a a column header for every column that's great but and then the the column
26:43
header is only listed once but if you put it in the map now the the the header
26:49
is in every single row so and that's stored as data so the header is now data instead of being just part of the schema
26:57
and that is one of the things that that really greatly increases the size of the
27:02
map and the flexible data types so definitely um think about that when
27:08
you're trying to work with this stuff so yeah how is graph data how is graph data
27:13
modeling different graph data modeling is relationship focused it's not entity focused so like uh a lot of these
27:18
columns and uh things like that that you would find in data tables are just freaking not there uh like the I'm I'm
27:27
about to give you the super secret sauce to all graph data modeling I hope yall are ready for this it's it's very very
27:34
easy because every graph database that I've ever I've made like probably three or four graph databases now in my career
27:41
and they all have the same schema always and it's weird it's like spooky how they all have the exact same
27:47
schema so like just if you can remember this schema then like you'll like you you you essentially have your graph data
27:53

modeling skills mastered so um the main thing to remember about
27:59
graph data modeling is that it's not entity focused so we don't care about columns we really don't so really for an
28:06
entity in a graph uh you have three columns you have an identifier you have a type and then you have properties of
28:14
that um of that uh vertex right or that node and so um in this case like in our
28:22
examples today that we're going to be working on we're going to have a type which is like a player type and then the
28:27
identif if will be their name and then in the properties might be like their height or their weight or their draft
28:32
year and things like that so we're definitely going to be working with some of those and we will be uh building that
28:39
out but keeping in mind that like this schema is exactly the same for uh like
28:44
the family of apps right for example like you would have a user ID and then it would be of type user and or it might
28:50
be of type WhatsApp user or type um Instagram user and then you have a graph that maps all of these properties over
28:57
to to each other and how they're connected because maybe this Instagram user actually has a connection to that
29:03
WhatsApp user because they're the same person and there's a relationship between these two nodes and that's what
29:09
we care about we care about the relationships we don't care about the entities themselves as much and that's
29:15
like the whole idea behind graph databases is like it's Shifting the
29:20
focus from how things are to how things are connected and if you can remember
29:26
that you're going to do a lot better in your graph data modeling uh just experiences so yeah let's go to the next

29:34
slide okay so in that case remember um uh for uh vertexes you have that three
29:42
column schema um edges have their own uh
29:47
schema which again is pretty much standard this schema is
29:52
very standard where you have your subject identifier subject type object
29:57
ident ifier object type Edge type and then properties of that relationship
30:04
like for example like um like the edge type the edge type is almost always like
30:09
a a verb like it might be is a or plays with or has a or different things like
30:17
that it's like it's like a very it's like like like a like one or two word like sentence it's a very simple thing
30:24
that like connects things like for example in the lab today we're going to look at like how players in the NBA play
30:30
with each other and against each other so you have like plays with and plays against those are two different uh
30:37
columns that we are going to be generating or two different Edge types that we are going to be generating in today's lab uh and so if you look at the
30:45
different things here like I like to think of subject and object as like like a sentence right where the subject is
30:52
the person doing the thing and then the object is the thing that's being the thing is being done on like
30:58
right like for example we are also going to be creating a relationship between a player and a team so it would be like um
31:04
so it' be like player plays on team and so in that case you would have the ident
31:10
the subject identifier would be a player name the subject type is going to be player the object identifier will be a
31:16
team name the object type is a team and then the edge type is plays on or like

31:23
that's going to be the they play on that team and then the properties might be like how many years did they play on
31:28
that team when did they first start playing on that team things like that right there'll be these different properties of that team object that
31:36
we'll be working with so that's kind of the idea behind um edges so you have
31:42
nodes and you have edges and they are connected right the way they're connected is an an edge takes two nodes
31:50
and links them and it links them through some sort of type some sort of edge type and then they're linked together that's
31:57
like that's the whole point of graph databases is looking at how things are connected so let's look at an example
32:03
here real quick so in uh we're going to be building out this exact graph database in the lab today but you'll see
32:11
um like you have you have like Michael Jordan and John Stockton these are two like kind of famous NBA players who
32:17
played against each other in like the 1990s in the NBA finals and uh they play
32:22
against each other because they're on different teams but then you'll see that like John Stockton is connected to the
32:27
Utah Jazz because he plays on the Utah Jazz and so this is exactly what kind of
32:33
modeling you're going to expect when you are building out graph databases is you
32:40
have uh like you have your subjects you have your objects you have your
32:46
connections and then they might have properties inside of them but this is the idea is like you're really focusing
32:52
on how things are connected and this is how you can build out these kind of
32:57
models and like you can build out really really cool insights from this cuz a lot
33:03

of the stuff like when you are in the other world when you're in that kind of like relational database world and
33:09
you're not in this graph World a lot of the times like the queries that you're trying to do like these graph traversal
33:15
queries they get messy because like you have to join and you have to create all these like crazy joins to like do it you
33:21
have to bring all the vertex data together and do all these like crazy extra scans but like once you get the
33:28
data in this format then you can start traversing the graph and seeing how things are connected in a lot more of an
33:34
efficient way and that was a big thing that like when I was working on Netflix that was a big thing I worked on was
33:39
like how do I get all of these different entities into a singular graph database
33:44
that then people can query and people can like look at and that was uh yeah it was fun it was a really fun problem but
33:51
also a very challenging problem because it just gets super messy but that's kind of the idea behind how graph databases
33:57
kind of work all right so I think that's pretty much uh it for the the slides
34:04
today [Music]

English (auto-generated)

Learning