Fact Data Modeling

Fact Data Modeling Day 1 Lecture

How Meta models Big Volume Event Data

Fundamentals of fact data

Transcript:

0:00
[Music] fact data is the biggest data that
0:07
you'll ever work with in data engineering when I worked at Netflix there was some fact data I worked on that was 2 pedabytes a day of data so
0:14
why is fact data so big compared to dimensional data fact data is every event that a user can use so you know a
0:21
company like Facebook has two billion users every user can do a thousand or 2,000 events every day by all the things
0:28
that they're doing so you do two billion times a you get to trillions of records every day that are generated so when
0:33
you're modeling your fact data you got to be very careful because the volume can cause your Cloud bills to go up a
0:40
lot so in this 5H hour course that I have put my heart and soul into we are going to be covering all of the details
0:47
and the nitty-gritty about how to manage both small volume and large volume fact data so in day one we are going to be
0:54
covering fact data modeling fundamentals like what is a fact like how to model a fact like how to make it so facts that
1:00
join with Dimensions it's a lot of like Kimble data modeling in the lecture and then in the lab we will be covering how
1:07
to build a fact data model on top of the MBA game details table in the postgress
1:12
database like if you don't have postgress set up follow the link in the description below for the setup instructions uh in day two we will be
1:19

covering the blurry line between fact and dimension so when you aggregate a
1:24
fact it can kind of behave like a dimension and it kind of gets blurry of like what's a fact and what's a
1:30
dimension and we're going to be going over that in detail like how to kind of draw the line between these things and
1:35
we'll also be covering the date list data structure which is a very powerful data structure that is used at Facebook
1:41
to model user activity so you can look at like the last 30 days of a user's
1:46
activity as one integer so it compresses the data a lot and it allows Facebook to
1:52
compute monthly active users very very very efficiently and there's a link in the description below for more details
1:58
about the dateless data structure and then in the lab we will be building the dateless data structure and using fancy
2:05
bit operations that are in postgress it's going to be really really exciting and then day three will be a deep dive
2:12
on shuffle and how you can use Shuffle in spark and trino and how you kind of want to minimize it and the way that you
2:18
minimize it is through this thing called A reduced fact so A reduced fact is a way to minimize the volume of all of
2:25
your fact data by preserving the most important bits and by building reduced facts you can supercharge your
2:32
analytical patterns so at Facebook I built a reduced fact framework called the long-term analysis framework and
2:39
what it did was it enabled decade long analyses that used to take weeks for data pipelines to run to now be done in
2:46
hours so the way that that works is by minimizing completely eliminating Shuffle so that allows your pipelines to
2:52
run as fast as possible and then in the lab we will be covering how to build a
2:58

reduced fact framework so that you can minimize and supercharge the analytics at your company so I'm really excited
3:05
for you to check out all this content I love building this content and if you want to learn more about how to build
3:12
this stuff in the cloud with high success Frameworks like trino and Iceberg I highly recommend checking out
3:19
uh the data expert.i Academy in the description below we have all sorts of this content that is completely done in
3:25
the cloud so I hope you enjoy the course I spent a lot of time on this and and uh you know get cracking there will be time
3:32
stamps for every single section in case you want to skip to one or you know go to whatever pieces you want since this
3:37
is a long course so what's a fact like most people think of a fact as like truth something that happened something

Day 1 Lecture (fundamentals of fact data) begins

3:43
that occurred like some sort of uh evidence something like that like a big thing like you think of like an action
3:50
maybe like a user logs into an app you you venmo someone $30 you run a mile on
3:56
your Fitbit even though technically running a mile is uh a little bit more
4:02
dicey in terms of like what is a fact because of the fact that a mile is is is
4:09
more of an aggregation because of the fact that you have all of the steps you can think of each individual step in
4:16
that mile as like the lowest granularity because you can't really break you can't
4:22
really break down a step into any smaller component so that's one of the things to always think about when you're
4:29
thinking about okay what is a fact like it should be something that really can't
4:35
be uh it's like Atomic like you can't break it down into a smaller piece and
4:40

running a mile with your fibit you technically can right you can you can break it down into the individual uh
4:46
steps or individual pieces of that all the way down and so that is something to
4:52
think about uh facts can appear in all sorts of layers and granularities like
4:57
that where you can aggregate up you can aggregate down all sorts of or or you can you can push it back down and like
5:03
be at the atomic level or aggregate up to like something like a mile so uh one
5:09
of the things that's really really really nice about facts though is uh they don't change they like it's like I
5:17
took this step at this time and uh it's not like I can go back and change that
5:22
it's one of the beautiful things you know like I I I I know that's one of the things I I talk about in therapy
5:28
sometimes is uh you can't you can't change the past and and facts are just
5:33
the past like you can't change the fact that you logged into Facebook 30 times today and maybe you need to have better
5:40
boundaries with your phone or something like that but what I'm trying to say here is you don't have to worry about
5:46
that part that was a part of Dimensions Dimensions have this like slowly changing transitive sort of property to
5:53
them that like kind of make them hard to deal with but you don't have to worry about that with facts which is a
5:59
beautiful thing very beautiful thing but facts have their own set of challenges
6:04
and their own set of problems that like honestly like when I when I compare facts and dimensions and I had if I had
6:11
to pick which one is easier like I would say dimensions are slightly easier and uh mostly because of what we're about to
6:18
talk about Let's uh let's go to the next slide okay
6:24

so facts are uh more challenging because there's a lot more of them if you think
6:31
about it like let's go back to the the number of steps that you take every day
6:37
so if you have 10,000 steps a day that's a lot more data than you as a person
6:44
right then so it's like instead of having one t one row of data for you as
6:50
a person you have 10,000 rows of data for every step you took
6:55
so uh there's a lot more data so and it will scale like way more right and like
7:01
that's an extreme example uh where like I would say the more of the general rule
7:08
is there's going to be between 10 and 100x uh the data that you would see uh
7:13
compared to your dimensions and the the big the big way that you can kind of figure out how big the uh fact data is
7:20
going to balloon is based on uh how many actions are taken per Dimension like for
7:29
example when I worked at Facebook uh we sent about 25 to 30 notifications a day
7:35
and so what that meant was from the two billion users that balloons to 50
7:43
billion notifications and that's I know that's just a lot of notifications and I know that a lot of people like turned off
7:50
their notifications and all sorts of stuff like that because it is like just an excessive amount of notifications and we're going to be talking quite a bit
7:56
about these notifications and how they worked and like kind of some of the challenges that I ran into with this
8:02
very bulky fact data uh another big thing that can be important with fact
8:07
data that is less important with dimensional data is you need context for
8:13
Effective analysis for example um say we sent a
8:19
notification well like that fact in isolation is pretty terrible it doesn't
8:26
really provide any insight but say we had say we sent a notification and then 20 minutes later you clicked on it and

8:33
then 5 minutes later you bought something and we have that funnel of like sent to clicked to bought and we
8:41
have that like funnel analysis and then we have all three of those facts in a in a line and then that goes like the first
8:49
one being like Oh we send a notification in isolation that is worthless but if you have those three facts together then
8:54
you can have a $50 billion business because that's all that's literally all Google and Facebook are are conversion
9:01
funnels where it's like hey you saw a you saw a result you clicked on that
9:06
result and you took an action and they they are they have optimized that really
9:11
really strongly I mean I don't know if y'all have seen the where Google ran this experiment where they tried out 40
9:19
different shades of blue for the links on their website and then one shade of
9:24
blue uh caused people to convert and purchase a little bit more than the other ones
9:29
and that was that made them like like I think like 10 or like some like 10 or hundred million some some weird number
9:36
of millions of dollars for just like freaking changing the colors of blue and so like one of the things to think about
9:43
like when you're working with fact data is what other data do we need or
9:49
Dimensions do we need to make this fact data more valuable like for example
9:57
another way to think about it is say we don't have that conversion step that purchase step but we have the we have the sent and clicked so we have like the
10:04
clickthrough rate um but what if we had another dimension we brought in the user Dimension and we now we can see like oh
10:12
users in Canada have a better click-through rate than users in the US and it's like why and that can kind of
10:20
like kind of slowly build up kind of intuitions and have you ask more deeper

10:27
questions for your fact data so that can be a big thing another really common and
10:34
annoying thing about fact data is that duplicates are going to
10:39
be way more common way way way more common and uh duplicates can be caused
10:45
by a couple different like angles of things like maybe uh the software
10:50
engineering team they push out a bug in the logger so every time someone clicks
10:56
it actually logs to records and that's more of a data quality error but you can
11:02
also have duplicates that are genuine like for example in notifications you
11:07
can have it where you send a notification you click on it like in in
11:13
in hour one and then you click on it again in hour 7even and those are both
11:19
genuine actions that we want to log but we don't want to count that as two cuz
11:24
then like we could have it where it's like oh our click-through rate is 200% which like doesn't even make sense like
11:31
you need to be able to dedup a lot of these records uh when you're working with fact data because if you don't DD
11:39
like your metrics are going to look really weird and they're not going to be like what you expect them to be and that
11:46
is actually probably one of the most challenging parts of working with fact data is this kind of D duplication step
11:54
that can really um cause a lot of pain and frustration so yeah yeah let's let's
11:59
go to the next slide how does fact data modeling work well you can think about it in two ways
12:06
on one side you have uh uh there's normalized facts versus denormalized
12:12
facts and they both are very important and I've seen uh both of them actually
12:20
be very powerful and I've seen both of them cause a lot of problems uh in both
12:27

cases like in the lab to today we're going to be showing how denormalized facts are causing issues but uh in a
12:36
slide here in a second we're going to be talking about how normalized facts can actually cause issues so what is the
12:42
difference between normalized and denormalized well so imagine instead of
12:47
when you when you have your record of like Zach logged in at this time but it
12:52
wouldn't be Zach it would be like user ID 17 logged in at this date and so you
12:58
would have that as a record but maybe uh for your analyses you want to not do a
13:05
join so instead you would say Zach 29-year-old male uh who lives in
13:12
California made this uh logged in at this time so then you bring it you can
13:18
bring in some of the other dimensions from like the other actors or objects in the fact data and that can make it so
13:26
you can just do group by and you don't have to do a join and that can make things
13:31
faster but it can also obviously make things duplicated because then it's like
13:37
okay what if I have like 50 facts a day then we just copy
13:43
my we copy that 29-year-old male in California 50 times that doesn't sound very very
13:50
efficient but sometimes that is the better option and we're going to go a little bit deeper into like when that is
13:56
a better option and when uh you want to go more the normalized route the key thing to remember here is
14:05
normalization like the smaller the scale the better normalization is going to be
14:10
as your option because when you have normalized facts you remove all the duplicates and you increase the data
14:17
integrity and you don't have to worry about restating stuff so that's where
14:22
normalization especially at the smaller scale normalization is going to be a big win um and so there is a trade-off

14:31
between these two and they both work pretty well and so yeah let's let's let's go a little bit deeper into this
14:37
so one of the things that I think people get wrong here is uh raw logs and fact
14:43
data are not the same logging and fact data are like inextricably linked
14:49
they're almost married where like if you don't have logging right it's very hard to get fact data right um so how are
14:57
they different i' would say a big difference is raw logs are usually owned by people who don't necessarily have
15:03
really strong data skills I know at uh Facebook when I work there the raw logs
15:09
are owned by software Engineers who mostly are responsible for working with online systems and keeping the server
15:17
running and they don't really care quite as much about the actual log data so
15:23
that's where you can have a lot of impact as a data engineer is you can work with software Engineers to get the
15:31
data logged in the correct format because that will make your life in the
15:36
fact data modeling layer a lot nicer so that's one of the the biggest problems right with fact data is you usually have
15:44
ugly schemas uh that are from the online system and you have other you know it
15:51
could contain duplicates and other quality errors the raw logs are don't have any quality
15:57
guarantees um the fact data is going to have longer retention nice column names
16:02
quality guarantees it's going to be just a lot better like the trust in the fact data should be in order of magnitude
16:09
higher than the Trust In The Raw logs and that is one of the biggest things that you can do as a data engineer is
16:16
being able to convert those raw logs into highly trusted fact data you can think of like facts from
16:23
this angle of uh think of this as a there's a few here you have who where

16:29
how what uh oh that's that should be who where when what and how sorry that that
16:36
that there's a double how in there I need to fix that okay so um who fields are usually pushed uh as IDs you can
16:43
think of this as user IDs this could be uh surface IDs this could be like
16:48
imagine like a Tesla has a car and they're driving down the road there's the ID for you as the user there's the
16:55
ID for the car like the car ID and you have have uh also you have maybe an
17:00
identifier for the operating system all sorts of things that like help you identify who is part of the event and
17:07
then you have where where in this case can be a couple different things this
17:12
could be a location like a country or a city or a state but this could also be where in the app like are you on the
17:20
homepage are you on the profile page are you on like where in the app uh so this
17:26
can be uh very similar uh where it it it can be modeled with
17:32
IDs as well uh but a lot of times they it's not modeled with IDs it's model is
17:37
just like like the like the pages are like back SL where or it's like the country or things like that um how and
17:46
where a lot of the times in uh fact data modeling are very similar uh because
17:52
we're in this virtual world and so it's like he used an iPhone to make this click so is is is the iPhone the where
17:59
or the how cuz it was on the iPhone but I would say the iPhone in this case
18:05
would be the how and uh the where is going to be the where on the web page or
18:12
in the app or what button they clicked that is more of the where and the and
18:17
the how is going to be more of like the methodology and that you could think of
18:22
uh this is one where it gets close between who and how are also um

18:28
interested like in the Tesla example you could think of like uh the the Tesla car
18:34
as more modeled as a how as opposed to a who But It's tricky because it is like
18:40
some there's ownership there because this person owns that car so it's like it's kind of in the middle so definitely
18:46
uh those two are kind of really closely linked um then you have uh we have the
18:51
other ones that we need to talk about this is going to be the event like the actual thing that happened so like
18:58
notifications there was a funnel of events right you have like sent or you have generated which is like a
19:04
notification that could be sent and you have the notification that is sent you have the notification that is delivered
19:09
to the device then you have the notification that's clicked on and then you have the notification that's converted into an action Downstream
19:16
whether that be like a like a comment a share a purchase something like that and uh you can think of all there's all
19:22
sorts of What fields like someone made a transaction someone took a step someone
19:27
uh did some sort of event someone threw a party there's all sorts of different uh events that can happen that are like
19:34
what and uh think of those what events usually as atomic and uh because you
19:41
could think of an event that's like a little bit higher level like for example you have like I threw Burning Man as a
19:48
party and that's like that is true that that's like an event and there was probably like but like that like is more
19:56
of like a group of events because that's like a weeklong thing but you could think of it as like a a shorter level of
20:01
like I registered with the Bureau of Land Management to be okay to get the
20:06

okay to to throw this party that's more Atomic right and you can think of like all the kind of atomic level things and
20:13
then those higher level components are like okay those are more aggregations of
20:18
because like you think of burning man is like the all of those events that are happening at one time they can kind of
20:24
all aggregate up into the whole party or like it's like a it's it's like a dimension almost at that point because
20:30
it's like okay all of these events are tied to this one thing and it's like a
20:35
kind of an aggregated view of one uh kind of set of facts and so but we're
20:42
trying to talk about modeling individual facts so remember that the atomicity of
20:48
facts is also really important and then this last piece is also critical for
20:54
facts is the when uh the when is going to be almost always modeled as a
20:59
timestamp or a date usually a time stamp and it says like okay I clicked on this
21:05
notification at this moment in time and so that is critical for facts like
21:11
without the when like you don't know where to put it and and like some things
21:16
to be careful here about is that you really do want to make sure that all of your devices are logging uh the same uh
21:23
time zone and this should be all be logging UTC time zone because a lot of
21:29
times like when companies are moving to being more mobile first where they do a
21:34
lot of the logging on the client side they do client side logging then the client side logging happens most of the
21:41
time in the time zone that the client is in which is bad for uh when it gets
21:47
pushed to the server so you want to make sure uh in The Client app that they are
21:52
logging things in UTC not in whatever the client time zone is so that when it
21:58

gets push to the server all of your time stamps are in the same time zone so that you can easily know like that like when
22:05
they actually happened and you're not it's not all shifted because of weird time zone problems so that's a key thing
22:11
to remember when a company a lot of companies are moving that way because it is better to do things uh client side
22:17
logging because client side logging is higher Fidelity than server side logging is because you get all the interactions
22:24
that a user takes not just the ones that uh cause a web request to happen so
22:30
that's a big thing to remember when you're thinking about the when of of
22:36
client side logging and and and fact data modeling okay so more things about facts
22:44
uh fact data sets should have quality guarantees um some some of the key ones
22:50
that I really like for fact data is uh no duplicates or like that's one of the key
22:56
things is like can you crunch all the data and duplicate it down so that like when an analyst is looking at stuff it's
23:02
all kind of flattened out uh you also have uh fields that need to be there
23:07
like for example that event field like the what field and the when field the
23:13
what and when field should always be not null they should always be there like if
23:18
they're not there like what like you can't even you can't even analyze that data if it's null it doesn't make sense
23:24
it's bogus so like you should definitely always and then like really who the who feel should also have some sort of
23:30
notnull um fact data should generally be smaller than raw logs uh raw logs a lot
23:37
of times will log a lot of extra stuff that you don't actually need that they might need for like diagnosis or for
23:44
like other like the software engineer needs it to like understand what's going on in the server and like that's not

23:50
something that you need to know to understand what's going on in the business and so a lot of times you'll end up modeling that stuff away um yeah
23:57
and fact data should par out hard to understand columns like a lot of the times the software Engineers will pass
24:03
you a a column that is like a string but it's actually a Json string that's just
24:10
a blob of like nastiness and it's like really hard to even query it and so fact
24:17
data should not have very many of those fact data should for the most part just be like strings and integers and decimal
24:23
numbers and enumerations and it shouldn't really be um
24:29
uh as much of those complex data types there can be some complex data types like for example uh when I worked at
24:35
Facebook one of the complex data types we had was just an array of string which was this notification was part of these
24:42
experiment groups and that was a really powerful way to um like easily kind of
24:47
slice and dice the notifications conversion rates based on the experiments that they were in so it's
24:53
not always the case that you shouldn't have complex data types but like you definitely shouldn't have like these massive Blobs of Json that's just like
25:01
painful and that's like your analysts are going to be so sad if you just like give them the raw logs and say parse the
25:07
Json yourself like that's like literally your job as a data engineer is you should be making data sets that are
25:14
really fun and delightful to use and if you're not doing that then yeah you're
25:19
not being a good data engineer so definitely uh make sure that those things are things that you are considering when you're doing fact data
25:25
modeling now I want to talk a little bit about uh in in one of those previous slides I
25:31
talked about normalization versus denormalization and so one of the questions with fact data modeling is

25:37
when should we like when should we bring dimensions in like instead of just like the ID but we want to actually bring in
25:43
some of the values so I want to talk about this uh Network logs pipeline I'll
25:49
give you a link to this as well that you guys can read more about it so uh one of the pipelines I worked on at Netflix was
25:56
this network logs pipeline and this is the biggest pipeline I ever worked on in my entire career uh this pipeline uh was
26:04
you know it's like it it did about two pedabytes of data every single day two pedabytes of brand new data every single
26:11
day and one of the things that we had uh and what it was was the The Source data
26:17
set was just every Network request that Netflix receives just every single one
26:23
and so that's why it's so much data and what we wanted to do with this data set was we wanted to see how the
26:29
microservices were talking to each other because I don't know if you know but Netflix is actually uh like very famous
26:36
for like uh pioneering the microservice architecture and the microservice architecture is really great uh for uh
26:45
making your development faster and making your uh uh responsibilities kind
26:51
of split out and like you can really do really great things with microservices uh but for security
26:58
they're actually kind of a mess because it's like okay if you have instead of having one app that you can secure if
27:04
you have 3,000 apps think about like all the ways that things can get hacked if you have 3,000 apps well there's just a
27:11
lot more things to worry about and so what we were trying to do was we were trying to figure out like okay what are
27:17
all of the apps and how do they talk to each other and like so then we can see like okay if this app talks to this app
27:23

we know if this app gets hacked we know all the other apps that it talks to and uh the only way that we could do that is
27:29
by looking at all the network traffic and that was like insane but like anyways how it worked was we take this
27:37
network traffic and we join it with this uh small database of IP address data
27:43
that we were able to um come up with which was um like just all of the IP
27:50
addresses for Netflix's microservice architecture and it worked it actually
27:56
worked and the only reason that worked is because of the fact that that data
28:02
set was small enough to do a broadcast join and a broadcast join and Spark for
28:07
the people who don't know is a way that you can do a join that's very efficient but the only way it works is if one side
28:14
of the join is small and by small I mean like less than like five or six gigabytes if it's more than five or six
28:20
gigabytes then spark is going to have a really hard time doing a broadcast join
28:26
um but one of the things that happened was because we were like oh this pipeline needs to be upgraded and um we
28:32
needed it to go to IPv6 instead of ipv4 and we needed to bring in a lot more IP
28:38
addresses and then we realized like this isn't going to work anymore
28:43
this join is no longer going to work we will not be able to figure out like based on IP address what the app is uh
28:50
so that then we can because after that join what we did was like we figured out what the app was and then we aggregated
28:56
down to see like oh this app talks to this app this app talks to this app right because it's just ip1 to ip2 you
29:02
just have like two IP addresses and you just do that join but like if you can't broadcast join it then we I mean we
29:09

tried we we literally tried to have it run without broadcast join and do a shuffle joint and like uh the cost
29:16
ballooned the cost ballooned like over 10 times and so we were like wow okay so
29:22
how do we solve this problem like how do we solve this fact data modeling problem because like we can't just like give up
29:28
our security questions um if because we want to upgrade the IPv6 and so what was
29:35
the solution here to solving this like very difficult problem even though like we initially did solve it in like kind
29:42
of a more denormalized way of solving it so what we needed to do was we didn't
29:49
need to do that join that like what we needed was instead of doing that we needed all of the apps to just log their
29:57
app every time time that a network request happened we just needed to log which app was what which app it was
30:03
receiving so then like instead of having to do this join we would that the app
30:09
would just be in the network request data itself so that was like really crazy
30:15
that we realized that that's what we needed to do was so what we needed people to do was adopt this thing called
30:21
a sidecar proxy which what it did was every app could just adopt this sidecar
30:26
proxy so every time it received a request it would log the app that it was
30:32
and was coming from so that would give us the app name and we wouldn't need to do a join anymore and so we would have
30:39
uh but now you'll see the fact data is now denormalized though because the IP
30:45
address is the identifier and so um bringing in that app information that string
30:51
information uh actually does denormalize the data because that is something that you can answer with a join but since the
30:57
data is so large and that join just becomes unmanageable you need to denormalize it
31:03

and log it ahead of time and so every time that the data is logged so then you don't have to do the join at all and
31:11
that made this pipeline more efficient I mean it was also a massive massive massive massive pain to do because then
31:18
we had to go and talk to 3,000 app owners and be like yo can you install this sidecar proxy and we had to like
31:23
get everyone on board and then uh and then get all new apps on board as as well and that was like so many
31:30
conversations right and that's just another great example of where a lot of the times the impact that you have as a
31:36
data engineer is not riding a pipeline and it's not optimizing a pipeline it's
31:43
actually going upstream and solving the problem at the source in this case the problem was we aren't logging the app on
31:50
our Network requests and we're trying to walk around it by doing these joins and
31:56
like trying to essentially solve putting a Band-Aid on the solution because we're like wow we really need this information
32:02
but like we're going to solve it through this kind of a crazy join way and when we realized oh wow we can change how uh
32:10
the the whole company manages this data and looks at security and that was you
32:16
know what we ended up doing even though like as Engineers we're like when we think about that we're like which would
32:21
you rather do own a pipeline that processes 100 terabytes an hour and does a join or have 3,000 conversations with
32:29
app owners uh I mean I I I think that a lot of data Engineers are going to pick
32:34
the the 100 terabyte an hour pipeline because it's like that's a lot of conversations and like some what what
32:40
what if people say no and you have to like uh talk with them and understand like why they don't want to adopt what
32:46

you want them to do and you have to have like debate and persuasion and all this kind of stuff like that so but that's
32:53
that's that's a beautiful part of being a data engineer is that is being able to help people understand the data
33:00
practices that they need to be adopting in order to make the company better and more secure so this is like for me this
33:07
is like one of the more like bigger impacts of things that I worked on in my career that I noticed was like wow this
33:14
is a a great way to solve this problem so yeah denormalization wins in this
33:19
case right so that just shows you that like sometimes denormalization is actually the solution to large scale
33:25
problems and not the cause of large scale problems okay so we talked a lot a lot
33:33
about logging so um how does logging fit into fact data uh logging should give
33:40
you all the like pretty much all the columns that you need except for maybe some of the dimensional columns and
33:46
those honestly you probably shouldn't even be in the data table to begin with you should just have the IDS and then people can join in on those IDs to get
33:54
the dimensions that they need uh that could be a very big important thing uh
34:00
remember this is all in collaboration with online system Engineers they are going to be the ones who are going to be
34:06
knowing a lot more about the event generation like when those events are actually being created in the app so
34:14
they're going to have a lot more context on that like as a data engineer you probably won't have as much context on that um another big thing is is like
34:21
don't log everything uh log only what you really need because a lot of times these raw logs can be very expensive
34:27
expensive and they can cost a lot of money in the cloud and so a lot of times
34:33
uh companies can fall victim to this idea that like we need to log stuff like

34:38
just in case um and so that's something that you want to not do that's something that is
34:46
very uh an anti- pattern and against kind of the efficiency ethos of data
34:52
engineering where data engineering is kind of like we're going to give you all the answers to all of your questions in
34:57
the most efficient way possible so that's a big thing uh the last thing I
35:02
want to talk about with logging here is conformance uh so when you are logging
35:08
your data there should be some sort of contract or schema or shared sort of
35:14
vision for things and this can be tricky because of the fact that like for example at Airbnb they like for a long
35:21
time they had a lot of their app in uh Ruby and we did all of our Dev in Scala
35:28
and it's like Ruby and Scala don't uh like you can't really just import their Ruby libraries into your Scala libraries
35:35
it doesn't work so what you need is a middle layer between the two so between
35:41
Ruby and Scala they shared some of the schema so for example like I worked in pricing and availability and I worked on
35:50
uh like some of the shared schemas of like okay what describes a price at Airbnb and what describes uh you know
35:56
what's available and these things were defined in what's called a thrift schema so Thrift is
36:03
actually a a specification that is language agnostic and it's a way to
36:09
describe schema and to describe data and functions as well and data Engineers
36:15
mostly just use the schema part I don't really use the function part as much but like it's mostly like schema and data
36:21
you can describe that stuff and it's in a in in a way that is shared so the Ruby
36:28
code and the Scola code will both reference the same schema for price and
36:33
so I'll know if like the Ruby code is if they add another column to their price

36:40
data like I will my my schema will also have that new column and hopefully like
36:47
my there'll be a test there that will break my code so that the Ruby team can't push their code until they talk to
36:55
me and they say like Hey we're adding this new column that's going to change the calculations so you need to go and
37:01
update your code as well so that like I can then go update my test and I can make sure my pipeline isn't going to
37:07
produce junk data just because the online teams decide to change how they calculate price so that can be a
37:14
beautiful way to have an integration between what the online service teams
37:20
are doing like however they're calculating the things in the app and how you are calculating your things in
37:27
the pipeline because if you don't have this shared schema and this shared way of talking about things those things are
37:34
destined to drift and they will slowly and slowly kind of fall apart from each
37:39
other and that's where having this kind of shared knowledge and this shared talking can be a very very beautiful way
37:47
to keep everyone on the same page and keep everyone kind of working towards the same kind of shared
37:54
Vision let's talk about uh some of the other things that is super important uh
38:00
to think about when you're dealing with high volume fact data
38:06
um sometimes like the solution is to not to not work with it like to actually
38:12
just filter it down and not work with all of the data uh that can be a very
38:17
powerful way to solve these problems so for example one of the things I worked
38:23
on at Netflix was this thing called infrastructure impact which was a metric that like showed if
38:30
an AB test caused a higher S3 bill or a higher AWS Bill than like another uh

38:37
experience like test and control like the test group oh like we're going to see 2% higher AWS costs or whatever so
38:44
but the way we looked at that was we did a sample of network requests and we only need needed to do like a 1% sample of
38:50
the network requests and that solved our problems and we were able to build pipelines a lot faster different than
38:56
the security problem because in security you can't sample because you have to know all of the possible like you have
39:02
that needle in the Hy stack problem right where it's like you can't sample so sometimes sampling doesn't work right
39:07
in like especially in cases like security or like where you where you have like those very low probability
39:12
events that you need to capture then uh sampling is going to essentially be a show it's going not work at all but for
39:19
a lot of metrics and a lot of directionality things you really are going to it's probably going to give you
39:26
almost the exact same uh number or the exact same metric but it just uses a 100 times less compute
39:32
and 100 times Less storage so in a lot of ways it's more ideal uh sampling
39:38
works because of this concept called The Law of large numbers which is as more as you have more and more examples and more
39:45
and more rows of data it approaches more of a gaussian distribution and so the as
39:50
you get more and more data you get a diminishing return on what that that distribution looks like and Ian the lob
39:57
large number says 30 right if you have about 30 data points then like you start to approach that uh normality
40:04
distribution and so that is a very powerful technique that you can use especially when you're working with
40:11
metrics that are like mostly used to kind of gauge directionality of things

40:16
but not like when you need to know the specific row data because if you know need to know the specific row data
40:22
that's when sampling is not going to work um another great example of things
40:27
that you can do with fact data especially if it's high volume is going to be bucketing where in these cases you
40:35
can bucket things on like user ID or you can bucket things on like whatever the
40:40
actor usually you bucket on The Who The Who IDs and um the cool thing about that
40:46
is like if you need to join on that column then you can join on that column and then it like you don't have to
40:52
shuffle everything you don't have to shuffle across the entire data set you just have to you just join within the
40:59
buckets and then uh it can be a lot faster so you can do bucket joins that is something that we are going to cover
41:05
in week five uh for the infrastructure track um how to do bucket joins in spark
41:11
uh and it helps minimize uh Shuffle uh and if you go to a sorted merge bucket
41:18
join which is an SMB join then you can actually do joins without Shuffle at all because in those cases you have like
41:25
both both buckets are line up and they're sorted so it's like a zipper so they like come together and then it just
41:31
Zips down and you don't have to uh there's no there's no Shuffle that needs to happen at all and that was a big uh
41:37
SMB joins was a very powerful technique that I used at Facebook to really make things more efficient it's not quite as
41:45
powerful or as used nowadays because spark kind of uh shuffling in spark is
41:50
like a lot nicer than shuffling And Hive was so but it still is a technique that
41:55
I see uh in in practice and I even saw at Airbnb so it's not it's not gone yet
42:01

though so bucketing that's the key thing to remember bucketing can be a really great way if you have a very high volume
42:07
uh fact data set bucketing is going to be a really powerful way that you can handle uh that
42:14
data another great question is like retention uh so so you have high volume
42:19
high volume fact data uh you can't just hold on to it forever dimensional data you can kind of hold on to as long as
42:25
you legally you legally are allowed to um one of the things that I noticed uh
42:32
for fact data in big Tech was uh the whales which are any fact data tables
42:39
that were greater than 100 terabytes they had very short retention usually it was like a week maybe two weeks and
42:46
those are like the whale tables which were like you know those are only like maybe 2% or 3% of all the data tables
42:52
and then uh and then it was like if it was less than 10 terabytes like big Tech
42:57
it was like okay like doesn't matter it's uh the only thing that mattered was like if
43:04
it got anony anonymized or not and that was because they only wanted they cared about the legal risk but from the
43:10
efficiency perspective they didn't really care um obviously uh depending on the company and the budget of things
43:17
these rules might change maybe it's uh at your company instead of 10 and 100 it's 1 and 10 or it's 100 gigs and 1
43:26
terabyte or whatever there's like usually there's an order of magnitude between like where you're like we need
43:31
to make this more efficient and we need to reduce the retention on this because the cost outweighs the benefit and
43:38
that's a big thing that like as a data engineer that you should be thinking about is what is the ROI for holding on
43:43
to more data and because you'll get push back from data scientists all the time because they're going to be like oh uh

43:50
we we need this data we need longer retention on this data just in case and
43:56
that just in case rarely comes like every time I hear that just in case like I don't know it's it's almost always
44:02
that's almost always a cop out and that's like not like very rarely actually something that they need to
44:07
worry about um so yeah retention think definitely thinking about that when
44:12
you're dealing with high volume data this is the last bit of this presentation and then we'll take a break
44:19
so um I talked about how D duplicating fact data is uh challenging uh because
44:26
you can have General actual duplicates of data and um but one of the other
44:31
things about it is like you have to think about the time frame like I know in notifications right where it's like I
44:37
imagine it this way like Facebook sends you a notification you click on it today but then you also click on it in a year
44:44
because the notification is still in your tray and then you click on it in a year do we care about that duplicate
44:50
like like do we honestly genuinely care about that duplicate probably not so there's going to be kind of a frame that
44:57
you have to like care about duplicates of some some time frame where those
45:03
matter uh but there's also a long tail where they don't matter and so that's a
45:08
big thing to think about is how like what is the distribution there of duplicates and you can even look at that
45:15
like you can do some analysis on the data set itself to see like okay where
45:20
is the the big chunk and like what is the way to do this like I found um
45:27
one of the things that can happen with these things like so I worked on D duplicating the notification event data set at Facebook and when I got there uh
45:36

that was just like one query and it was like it took like nine hours to run in Hive and it was very painful because it
45:42
was Master data that a lot of other data sets relied on and so we wanted to reduce the latency of that data because
45:49
it was like it was available like at like way way late and then all these other pipelines could only start after 9
45:55
hours because they were waiting on this data so what are some options for uh
46:01
making or reducing the latency of these dding so the two big ones are going to
46:06
be uh streaming and microbatch is so microbatch might be like on an hourly
46:12
basis and streaming obviously is going to be on like a even lower basis than that so we're going to go a little bit
46:19
more into details on like how I actually duplicated uh this very large volume data um streaming allows you to capture
46:27
the most duplicates in a very efficient manner because you can essentially capture the duplicates on whatever
46:33
window you want uh and you just like say okay we saw this notification ID and we're going to hold on to it and then if
46:40
we see any duplicates over the next like 15 20 30 minutes or whatever then we can
46:46
capture most of them that way because uh you know a large majority of the duplicates happen in a short window
46:53
after the first event and so you can capture a large majority of the duplicates with streaming in a very uh
47:00
small window one of the things that was interesting about this for the requirements that I had in notifications
47:07
though was that we needed to hold on to every notification for the whole day and
47:14
we needed to D duplicate throughout the entire day not just in like a 30 minute
47:19
or an hour long kind of window and one of the things that caused was there was
47:25

just so much memory used so much memory and like actually streaming didn't work
47:31
I tried I tried I I tried doing streaming stuff on this for like six weeks and I could not get it to work and
47:36
I was like wow this is this like an unsolvable problem like how do we get past this problem in notifications and
47:43
um so but streaming especially if the volume's less because you know keeping in mind that my problem I was working on
47:49
was like 50 billion records a day so like if if your problem's less than that you're probably going to be in a better
47:55
spot we're streaming is going to probably be great for your use case especially like if your duplication
48:01
window is smaller then uh streaming it can be like almost a slam dunk and it can be the one of the simplest Solutions
48:08
so I'm not saying just because streaming didn't work for me that streaming wouldn't work for you it could
48:14
definitely work for you in doing these kind of D duplication of facts this was
48:19
how I actually pulled this off uh of D duplicating um the Facebook notification
48:25
data so used this thing called hourly microbatch ddop and so what this did was instead of
48:33
taking nine hours it was actually available one hour after midnight so and
48:38
that was um great we were able to really uh solve this problem but like what does
48:44
this pattern even look like like it's probably uh kind of messy so we're going
48:49
to go over each piece of this real quick so the first step in hourly
48:55
microbatch DD is you get all the data for a specific hour and then you
49:00
aggregate down right so in this case uh you would have your um product ID event
49:07
type and this would be for a given hour like for one hour you collect all of the
49:12
data for one hour first right and you do that for every hour all the way from hour zero to hour 23 you just get all of

49:20
them together and you group them and that's how that's the first step and what this group bu does is it eliminates
49:26
Ates duplicates within an hour so within that one within the hour after this group buy
49:33
all the duplicates are eliminated so that's step one then step two what you do is you do
49:42
a full outer join between hours zero and one or hours uh two and three or hours
49:49
uh four and five etc etc and then what this full outer join does is again it
49:56
makes it so that it eliminates duplicates that are across hours so if there's a duplicate that happened in
50:03
like an hour zero and an hour one this full outer join would get rid of that
50:09
duplicate so that you could be able to have uh uh like a d duped two hours duplicate
50:18
or a d duped like two hours event stream I mean so then what you do is these all
50:24
come together and it branches like a tree so let let me kind of show you what I mean by that so this is kind of a
50:31
diagram so what this does is like you have you wait for your hour and then you
50:36
do that group by that's what this DD hours one through eight dot dot dot and then you do the merge which is that full
50:42
outer join query and then after hours one and two merge and hours three and four merge then these two merge
50:49
togethers to do hours one and four and then you keep merging keep merging keep merging until finally you get down to
50:55
where you have hours one and and 16 and ours 17 and 24 and then they merge together and then you have your final
51:01
daily duped data set and yeah this like was a lot of work but it's a it's this
51:07
way is a very resilient way to dup fact data because like it's still batch but
51:13
like it also uh handles data in a very it will dup data in a very low latency
51:19

way so definitely I highly recommend checking this out because this works at pretty much an arbitrary scale
51:26
uh yeah so so today's lab what we're going to do is we're going to be uh looking at some
51:37
of our different kind of uh values here so we we are working with the MBA data
51:44
set again where we we have games and game details we're going to be looking at things like teams and games and
51:50
players a couple different things like that and we're going to be building out our fact data sets that will will be
51:56
using a lot of these data congratulations on getting to the end of the first lecture in the fact data modeling course if you are watching on
52:03
the platform make sure to switch over to the next one in the links so that you can get credit for every lecture in lab
52:09
here you don't want to just keep watching completely thank you so much for getting here and make sure to like comment and subscribe so today we're