Course Name:Business Intelligence and Analytics
Professor Name:Prof. Saji.K.Mathew
Department Name:Department of Management Studies
Institute Name: Indian Institute of Technology Madras
Week:02
Lecture:08

ONLINE TRANSACTION PROCESSING | BI&A

Now, since we are dealing with designing databases, designing relational databases to store and manipulate data in a structured way, let us take an example of a, say a transaction, a purchase transaction. And so, in a B2B context, business to business context. And let us try to imagine and then ask certain questions and see how a database will be created for enabling the purchase transaction in a B2B context and then sort of understand the steps involved. So, who are the stakeholders involved in a purchase transaction? That is the first question to ask. Who or who, what? Or what are the objects about which data needs to be captured in the transaction? It is another way of asking that question.

What are the things, what are the objects or who are involved, what is involved or what is the sort of entities we call, you know, in design level we call them entities. So in other words, what are the entities? When we ask this question, we can imagine that a business to business transaction starts with a customer, is not it? Every business has a customer, if a business does not have a customer, there is no business. So we, let us use some representations. So, let me draw some boxes.

So, here comes a customer, there is a customer. So, the label is a customer, a singular noun. And this is an entity that I have represented, an entity. So customer is an entity. And what does customer do? Customer places an order.

Therefore, there is something called order in a transaction process. Customer places an order. And what else you can think of in a transaction? So, order is something that is placed by a customer, but order has some things in it. Order has certain content in it. What does an order talk about? And when you imagine that, you can see that an order will definitely contains, will contain item.

Order is about what you ordered, that is the major part of an order. So, order we can say, the business language can be order contains items. Now quickly, let us look at each of these entities, customer, order and item. When you look at these three entities closely, you can easily imagine that customer will have certain attributes, customer will have certain understandable attributes, which are related to this particular context. We include

attributes of a customer, a customer which are relevant in this particular context.

So, for a customer, we know that a customer should, if it is a B2B, a customer is a business entity. So there has to be some name, some address, telephone number, email, GST, or whatever you want to add, which is relevant for a business transaction. There may be a lot of other data about an organization or a customer, which is not relevant to you, you do not have to include. So, it is about entity you can see, the thought process in designing a database is you identify entities. Entity is a complex data element, which consists of multiple attributes; multiple, but related attributes.

And that together define a customer. So, for example, if you instead of customer, I change it to a student, what would be the attributes of a student? A student has a name, a student has a address, email, cell phone, and so on and so forth. But that is from an academic point of view. Suppose the student is getting enrolled in a gym, then what are the relevant attributes, the student's name, email, identification, all that is fine. Along with that, they may also ask you for your height, your weight, and other dimensions of your body, etc.

But you do not have to enter it in a business context like this. So we say relevant attributes. So, and similarly, when you look through each of the entity and try to enter, what are the relevant attributes. So, order, so order will have definitely a date. And when you think of order, you see that there will be, in an order there will be multiple items, is not it? Item 1, description, item 2, description, and so on.

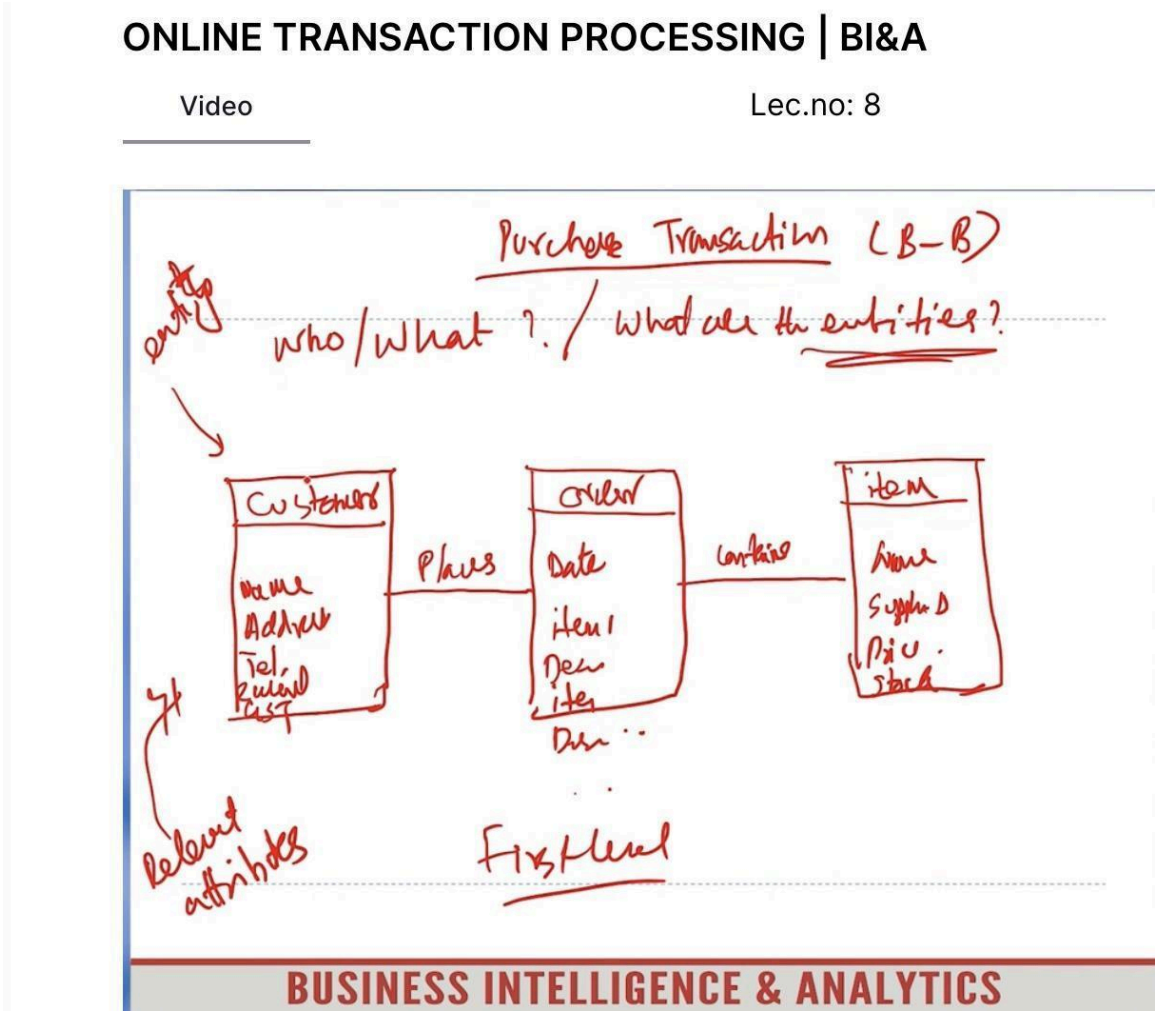
So, you do not feel very comfortable because some things are repeating here. So, but let us leave it there. So, when it comes to item, you know that item has a name, item has a supplier. So, there may be a supplier ID or something that tells us that there should be a supplier entity. We are not involving that here.

And what other attributes of the item, like the price of the item, stock, how many items are there in the inventory and so on. So, you can add relevant items there or different relevant attributes there. Now, so this is the first level, I will call this as the first level of database design. This is the thing I want to emphasize, you will learn a little more about databases in the next session. But here, how are we thinking to build databases, we are having a very structured approach.

The structure is that in a particular context, we are trying to define what are the main things about which data needs to be stored. So, we are thinking in terms of entities. So, then logically, we think we see that customer is involved, order is involved, item is involved. You can see that these are entities, entities meaning complex data elements. So,

entities consists of attributes.

So, these are multi, entities are multi-attribute things. It is difficult to find an equivalent word for entity because it is defined at a very fundamental level. So, these entities are discovered and their relationships are discovered. And now, we talked about data integrity. So, we are missing something which should be added to ensure that each record is unique.



Suppose there is a customer, a new customer getting added to a business. That customer should be uniquely defined, that customer's identity should be unique, that customer should not be confused with another customer. So we cannot use name to define, uniquely define a customer, you cannot even use address, telephone number could be shared and email and GST number etc could be unique, but they are sort of, they have a different purpose. So, typically what is done to make a particular record unique is to add

a unique identifier. So in this case, we add another field or another field called customer ID and this will be known as the primary key or PK, primary key of this. For order, it can be a purchase order number or just order number, item can be barcode or item ID.

So if a barcode is used, so then barcode is the primary key. So we can see that we have added a unique identifier to each entity, ensuring that data entity will be ensured from the entity integrity point of view. Now, we also talked about relational integrity. Now, look at this relationship between different entities. Customer is related to order, order is related to item. We have described what is the business relation, but we can define that in a much better way.

For example, when we fill the order attributes, we missed something, which is very important for an order. That is who placed the order, who placed the order, who placed the order, a customer should have placed the order. So that customer has an ID. Now, you can see that I am adding an attribute in order table, which is also present in the customer table. I can extend this to capture this also.

Now, this particular attribute I added to the order table, which is borrowed from the customer table is known as a foreign key or FK, foreign key. It is foreign key because it is donated by an external entity. Now, what has happened now, you can see that since I added customer ID, it is now related to a table. I am saying here that for every order, there is a unique customer or an order comes from a unique customer. So there is one customer who has placed the order, a customer can place many orders.

It is a one to n relationship, a customer can place many orders. So, you cannot put a unique order ID in customer table, but you can put a unique customer ID in an order table. And so therefore, this particular relationship has a foreign key involved. And therefore, now you can see how the relational integrity is ensured. For example, if this customer record is deleted from the customer table and order, a new order when you place this you can see as a master table, so or a master entity.

So, if a customer's data is removed, then you can no more place an order with that customer's ID because that customer ID is deleted from the master table. So, there is a referential integrity that is ensured here. So, foreign keys are actually carriers of relationships or a relationship between two entities is bonded through a foreign key, you can see it here. So, I have defined something known as the cardinality or the level of relationship between two entities here.

This is known as cardinality. And now, if you look at the relation between order and item, can you have a similar cardinality? For a given order, how many items can be

there? An order can contain multiple items. For a given item, how many orders can be there? An item can be present in multiple orders. So, therefore, this is a kind of n to n relationship. And therefore, in database design, we say that this relationship should be resolved. And for resolving this kind of relationship, you will introduce a third entity called an associative entity.

And this can be called an order item table. And this will have the keys of both the tables, order number and item number called a concatenated key or a combined key. And then, instead of relating the tables here, you will relate, both the tables will relate to each other through a third entity, which is having the keys of both the tables. The idea here is that order number and item number, when you combine them, what are the unique features or what are the unique attributes for a given order and given item? You can easily imagine a quantity or a price or that kind of information pertains to a given order and a given item. So therefore, the quantity and all aspects that are uniquely related to a combination of both the keys will be entered in the third table.

In a way, we are actually trying to normalize these tables. We found that the item description, item quantity description etc. are repeating here. Essentially, we are not including them in the order table.

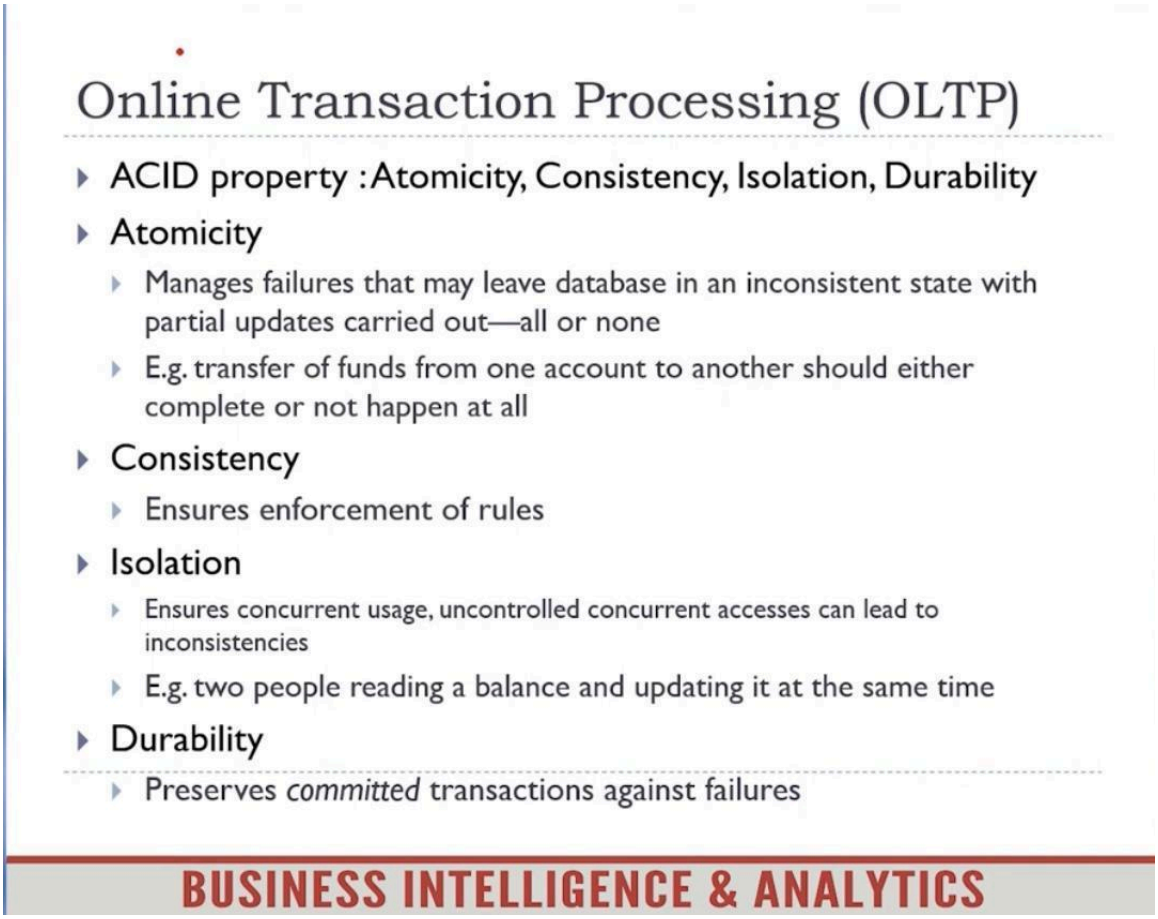
This is not the place for it. We are actually normalizing the table. You will get to know a little more about normalization in the next session. So, I am not repeating it here. But you remove it, item details are entered in the item table already. And item master is referred to in the order item table.

So, you are trying to remove redundancies. How is redundancy removed? On the first instance, when a customer places an order, the order contains the order details like the order date and any other information that direct shipping, shipping date, for example, a date has multiple characteristics here. One is the order date, other is the shipping date and so on. So, all that will be in the order table. But order table also has the data who place the order, customer ID, but does it mention customer ID followed by customer name, customer address, telephone, email, GST, etc.

It does not have to come in the order table, because order table with a key customer ID is already referring to that information from the customer master or customer table. It is already entered once in the table. But think of a scenario, if you are using a spreadsheet to store transaction data, every time you enter a customer ID, you have to enter the customer data in terms of address, phone number etc. And then in, enter the details of the purchase order. Suppose the same customer comes after say 1 month, you will be doing the same way, you will be adding the customer name, customer address etc.,if

you are doing it in a very rudimentary manual way. You can see that that kind of redundancy of repeating information would happen if you are not structuring data like this. So in summary, the idea of relational databases is to structure large volumes of data into a few entities, which contains a coherent set of attributes, which define together, is related to a key called a unique identifier. And each entity within a schema, we can call it a transaction schema, a schema, schema, this is a schema, because it is a set of related tables. So in a schema, each table is having a relationship with one or more other tables. So, customer is related to order through a key customer ID, which is a foreign key, ensuring referential integrity.

And it also ensures that there is no redundant or repeating attributes within a table. So, this is managed in a relational design of databases. So, that is the concept. So, that is the thought process through which data bases, relational databases are defined or set up. So, we will learn a little more about data bases and subsequently about querying databases very soon.



So, we have seen so far that relational databases are structured and they store data in a way that enables retrieval of data, very easy and intuitive. Now, we have also seen that relational databases reduces redundancy. So, they reduce redundancy through a process known as normalization. So, there are different steps in normalization. It starts with reducing or removing repeating items from entity and then also ensuring that all attributes are fully related to the primary key or there is no partial dependency, but there is full dependency of each attribute to own the primary key. So, the first normal form, the second normal form and the third normal form, Boyce Codd normal form etc you will see in the next session separately, when we discuss database management and SQL queries.

So, we move on from here to get an overview of databases, particularly databases used in the online context is what is described here in this slide. So, the OLTP or online transaction processing systems are databases which empower online transactions, in terms of data storage and also other unique requirements for online processing. Think of online reservations, for example, if you are using the IRCTC website, to reserve train travel from point A to point B. So, suppose you are one person who is trying to access that particular leg of travel from A to B, but you are sitting in, say in Chennai or in Delhi or in Hyderabad and you are trying to access the IRCTC database, essentially to see if a ticket is available or if a vacancy is available and if so, you are trying to reserve that vacancy, but they along with you, there may be others who may be in different locations, but trying to access the same resource, for the same date, same leg, same resource is being accessed by others.

So, that is online multi-user environment. So, it is important for the databases to enforce certain rules that there is no conflict among multiple users when same resource is accessed. So, you need to have those properties for the database, only then they can enable online transactions. So, in order to do that, there is a set of four important characteristics that is known as ACID together. So, ACID is a very dangerous liquid, but here in ACID as a property for online transactions is a very positive set of attributes. So, atomicity, consistency, isolation and durability-these are the four characteristics that is required for online transaction processing databases.

Atomicity ensures that it manages failures well. For example, if there is a failure, suppose your system crashed or there is a power failure etc, when you are making a reservation and suppose you made, you were making a payment at the end, but during that step, there was a crash.Then as a user, you get into a mode of anxiety because you worry if you lost your money, but you did not get the reservation done. But the atomicity property will ensure that either a transaction is complete or it is not complete, there is no intermediate level at which it will leave you. So, a failed transaction is a failed

transaction, you get the money back because the transaction was not complete. So, the database ensures at the, because of the atomicity property that there is no transaction that is left in the middle, but either it is failed or it is successful.

And the second property is consistency, where it ensures certain rules that you can configure as a database expert or a user of database, you can say, when a date is captured, it has to be captured in a date format.

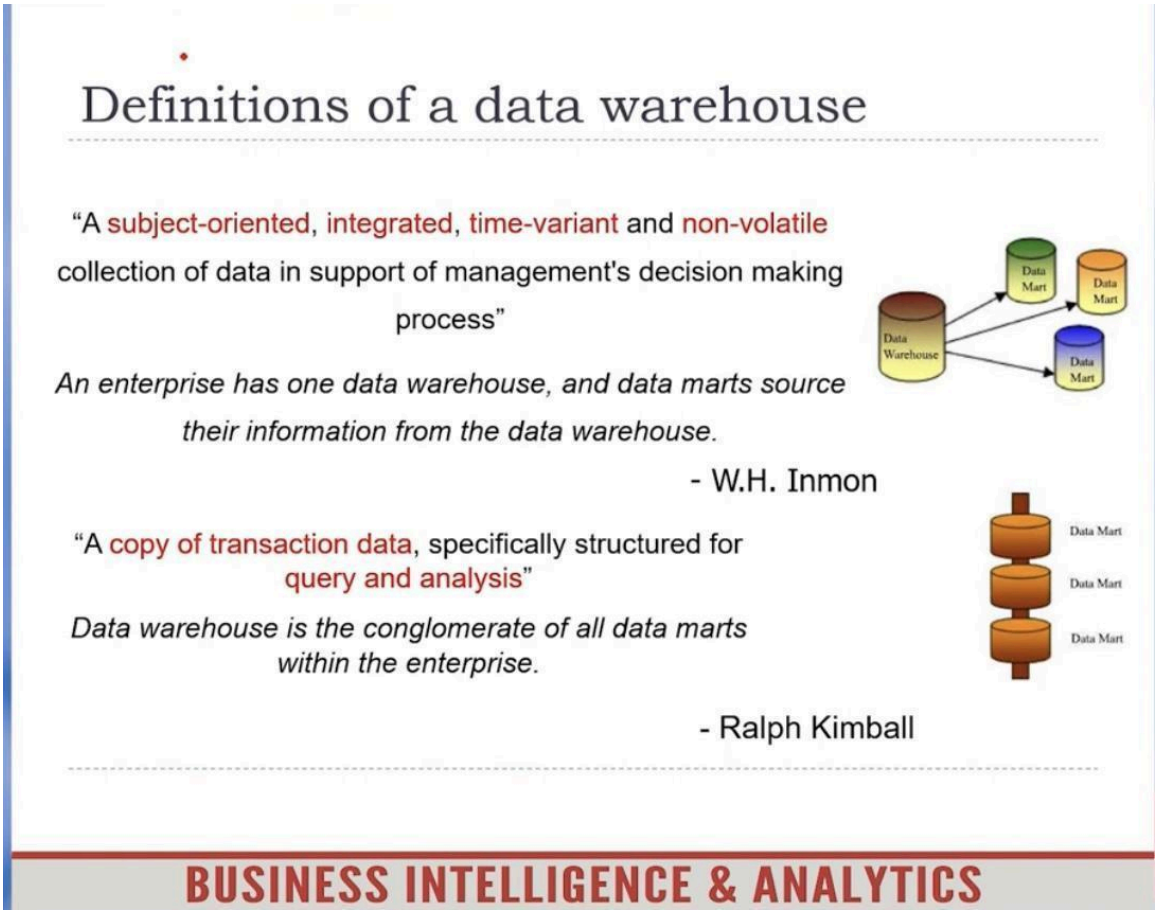
If you enter date in a currency format, it is not accepted. So, these kinds of rules about data and data types and other rules can be enforced by a database. And that also ensures integrity of data. And then there is isolation, this is what I discussed in the beginning, that for multiple users, when they access the same resource, database either logs that particular record, so that it is not accessed by others, or it manages this kind of problems in different ways, locking is not the only option, there are other options too, but that property is known as isolation.

And the last property in asset property is durability, which where if a committed transaction or a completed transaction is completed and committed against all failures.

So, and therefore, if you have booked a ticket, you know, there is no way that it can be reversed, it is committed transaction. So, databases ensures that nobody tampers with the records that have been created and these properties of the online transaction processing databases ensures it. So OLTP, whenever you hear OLTP, imagine a database that facilitate online transactions. Now, we come to the next important component that is a part of the infrastructure for analytics, which is the data warehouse. Databases or OLTPs are for transactions, they are not for analysis.

And it is always a good principle to be followed that you do not mix data analysis and queries with data transactions or business transaction. Business transactions use OLTP for data analysis have a separate database called data warehouse. So, as we saw in the architecture of business intelligence, data warehouse is a central store, which acquires data from different sources, different types of databases, depending on the nature of your business and depending on the nature of your IT, but an organization has one data warehouse, and that is the central idea. Although it can be debated, but a data warehouse is what is commonly understood.

And that is a central asset to an organization. And data warehouse also has certain properties in one philosophy or one architecture. Data warehouse design can be done mainly by two approaches. One is the Inmon’s approach, other is the Kimball's approach.



But Inmon, W.H. Inmon is known as the father of data warehouse and he defined data warehouse as a subject oriented, integrated, time variant and non-volatile collection of data in support of management's decision making process. So, data warehouse is not for transactions, it is to enable decision making. And it is one central entity and it has four features, subject oriented, meaning each data object is stored subject wise, customer is a subject, order is a subject, item is a subject, supplier is a subject and so on. So, it is also having a table structure or a subject oriented approach and entity structure in the design.And it is integrated. So, databases may be multiple and could be distributed, but data warehouse integrates data from multiple sources into one store or one storage mechanism. And that is the second characteristic.

And third is, it is time variant. It is not static, but it is a dynamic data storage system. For example, customer transactions, when it is captured by POS systems in different parts of a country but there can be one data center and one data warehouse which will receive a predefined, relevant and summarized data or filtered data for permanent storage into a data warehouse on a daily basis. All that raw data need not be stored in a data warehouse, but you may capture, for example, the recency, frequency, monetary value,

bizocity score, these are the relevant data that you need to maintain in a data warehouse for analysis.That will be periodically, it is a batch processing typically and frequency is to be defined by the, at the time of designing the data warehouse. That is why it is called time variant.

And the fourth characteristic is that it is non-volatile. What does, you know, you know, volatile memories and non-volatile memories. A RAM is a volatile memory, when you shut down the computer, what is stored in the random access memory is lost, whereas a disk, an external disk where you store your data, that is a non-volatile memory because that data is not deleted when you remove the power connection.

So, a data warehouse is non-volatile in the sense, it is not erased. It is a history of the organization. It is like your memory, you do not want your memory to be erased, although some bitter memories you want to erase, but they do not go off. So, it is like the memory of the organization. So, data warehouse is the memory of the organization and it is non-volatile. But a transaction database like the OLTP is not non-volatile in that sense.

You may delete data records after a period of time because you do not want to store all the raw data forever. So, there will be some frequency at which such data will be removed. It also depends on sometimes regulations, you need to retain certain transaction records like the call detail data for a period of time, but it is not non-volatile. So, subject oriented, integrated, time variant and non-volatile. And the particular architecture that is shown here is actually about the data warehouse which is central.

And you also see something known as a data mart as part of this representation. There are different data marts. As I explained earlier, data mart in this particular architecture is a subset of the data warehouse, but not just a subset. It is a subset in the sense it will have certain tables or certain views of a data warehouse contained in it, but it may also have, that is the part of the data warehouse part, but it may also add external data. Some external data like the census data, like certain survey data or whatever is relevant, fundamental environmental data and so on, you know, econometric data.

So, all this depends on what kind of analytical requirements you have. So, that is a data warehouse. So, marketing, data mart. So, marketing may have a data mart, finance may have a data mart, operations may have a data mart and so on. So, it is a more specialized data set for the purpose of analytics pertaining to that particular unit, that particular function.

So, that is how data marts are defined. So, this is Inmon's architecture, but there is a competing architecture defined by Ralph Kimball and he, according to Ralph Kimball, data marts, there are only data marts, a copy of transaction data, specifically structured

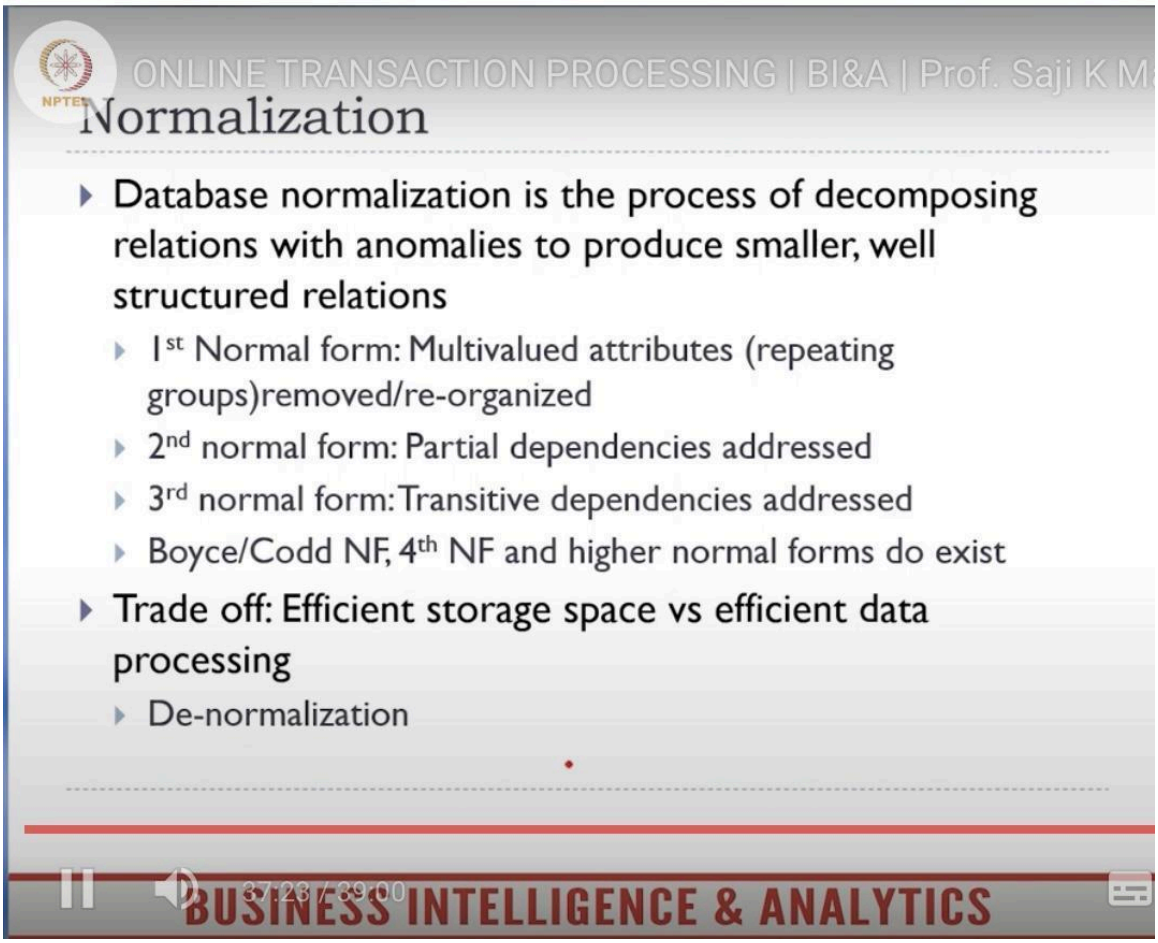
for query and analysis. And you can see that there are different units in an organization, it is more like a federal approach, federated approach. This is more centralized. If you are, it also depends on your IT architecture, certain organizations have a very centralized IT architecture, certain organizations have federated architecture where each business unit is independent to acquire and deploy its own information systems, its own databases, but then, I am sorry, you build your own data marts, each unit builds its own data mart from its OLTP or databases. And then an organization as a whole connects these data marts through a bus, through a means, when the data marts are connected, then it becomes, the connected data marts becomes a data warehouse.

As a whole, when the data marts are integrated, it becomes a data warehouse. So, this is more like a bottom up approach. You build data marts and then build data warehouse. Whereas in Inmon's representation, you first build the data warehouse, and then divide that into data marts, it is a top down approach. But these are two approaches to building data warehouses. And keep in mind, the purpose of a data warehouse and the data marts is not business transaction.

It is not part of an application that conducts transactions for your business, but it is a part of the analytics program of your business. So, I do not have to spend much time on these slides, because I have explained what each of this means. What subject oriented means is data warehouse is organized into tables, subject wise, subjects meaning a customer, product, sales, etc. You can imagine the corresponding tables.

Data warehouse is integrated that it carries data from multiple sources. It could be your email server, it could be your transaction server. Or if you have flat files, still have legacy systems. So, there may be wrappers that convert flat files into database format. And you may be actually connecting with those data sources as well. Then data warehouses, time invariant, and it is generally not deleted, depending on the policy.

And or it may actually carry data for a long period. That is understanding. And then data warehouses, sorry, data. Here we discuss data warehouses time variant, and therefore, it is constantly updated. And here we say that not it is non-volatile, in the sense data is not deleted, but data is stored for longer period of time. And we also define a data mart as a smaller, more focused or specialized data warehouse or it is a mini warehouse. Data mart typically reflects the business rules of a specific business unit within an enterprise.



It could be a business function as well. So, normalization is something that you are going to see in more detail in the next session. So, there we will appreciate what is the first normal form, second normal form, third normal form, and fourth normal form, Boyce Codd normal form and so on. And always keep in mind that there is a trade-off between storage space and efficient data processing. What is the trade-off? The purpose of normalization is to reduce redundancy, to reduce redundancy of storage.

So, it actually makes storage more efficient. But when you have to query, if there is some redundancy, the queries can be faster. Otherwise, you will have, you know, inefficient queries or more time required to process queries. So, that is a trade-off between normalization and query efficiency. And therefore in, you will see when we discuss OLAP, there is denormalization often done in certain schemas to make queries more efficient, queries more efficient because they are getting results faster is more important than redundancy.

So, we say some redundancy is fine so long as queries are responded to quite fast. So, that is a trade-off between query and normalization.