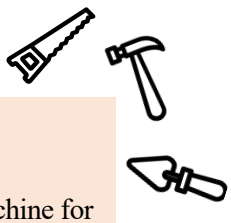
Dear Data, tell me a story ...

by Rachel Brabender and Oliver Clasen

Welcome to the world of data. Nowadays, modern computer systems (e.g. your mobile phone, car and small robots) are collecting tons of data on what is happening in their environment. Such data provides much information about the people and their passions. Companies have already realized the power of such data by analyzing them. Now we want to explain to you how this data can tell a story.

Let's think of a handsome example.

Since the rental store has to be aware of which of his inventories are currently lended, they save all necessary information in a database.



You are a passionate

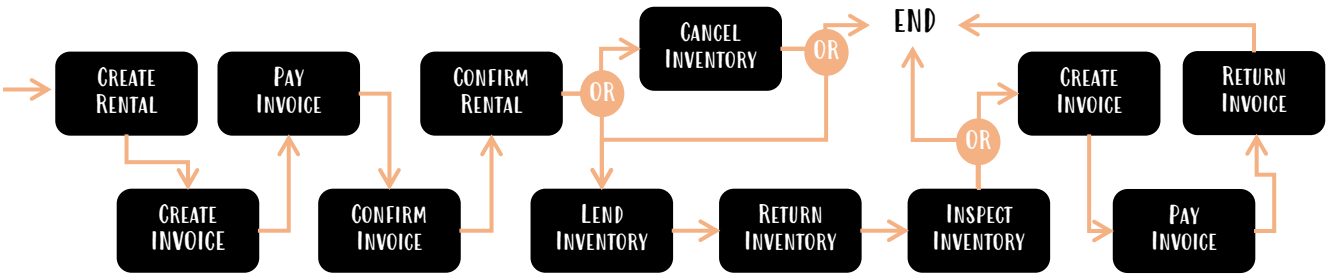
DIY ENTHUSIAST

and in need of a grinding machine for your homemade table project. Recently, you have noticed that not far away from your place, there is a rental store where you could probably borrow it for a few hours.

Perhaps, you can imagine how much data is generated if you consider the amount of inventories the store or even the business chain may offer. This data is saved in very big information systems and waits to be used. Potential application scenarios are analyzing all rentals, on which inventories are from high demand, so that the store can be retrofitted onto the popular ones. As a result, this could improve the image of the store because of the higher availability of popular machines and enhances the customer satisfaction.

Before digging deeper into the usage of such data, let's have a look on the rental workflow. It could look like this:

1) When the customer **creates a rental** order involving the several inventories, it is also considered as the starting point of the rental.
2) Based on all recently created rental orders, the store will **create invoices**.
3) The customer must **pay his open invoices**.
4) The store receive your payment, and will **confirm the invoice**, and
5) will **confirm the rental** order.
6) Now the customer can **lend** his **requested inventories**,
7) or probably **cancel an inventory** from his rental order.
8) If the customer has picked up all **requested inventories, he will return** them at a later point in time.
9) When the inventories are returned, the store will **inspect** them,
10) since they might have suffered from damage.
11) In the case of a damage, the store will **create a new invoice**,
12) which the **customer has to pay**, so that the store can **confirm** it.



A well-organized store is aware of the customers rental requests for specific inventories, and knows in which step of the rental process the customer currently is. Therefore, we store such information about performed activities in form of adding or updating information related to an object (e.g. rental, inventory, invoice, customer, staff or inspection) in the database.

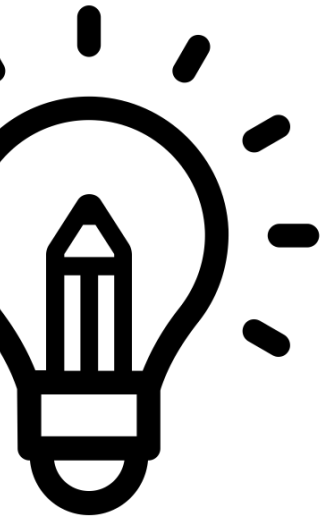
When thinking about the process, please be aware of the following restrictions:

- Each of these actvities do not involve all objects at the same time. As for example, create rental and pay invoice doesn't involve a staff member and inspect inventory doesn't consider the customer and is performed by a staff member.
- Be aware that a rental can involve more than one inventory at the same time. An invoice is always created for one rental.
- You can deal with multiple rentals (in form of picking them up all together from the store) and multiple invoices (in form of paying them all together) at the same time. The store e.g. creates invoices for all created rentals, confirms all received payments/rentals and inspect multiple inventories of the different rentals at the same time.

As you can see, the rental procedure is far more complex than it seemed in the beginning. Furthermore, inventories are not only just rented or not, they might be already requested by a customer, so that we could not reserve or hand it out to someone else for the moment. A common technique for structuring the information is to create an own table within the database for each object type of that process. The complex dependencies between our objects (a customer creates a rental which involves multiple inventories, and the store creates an invoice for each rental) is also visible in our database. For realizing such dependencies, we make a reference within an inventory entry to the belonging rental. The disadvantage is that information, such as about the customers, rental requests and related invoices, is distributed over different database tables.

Today, one of the biggest challenge when describing a process flow is that one object can be involved in different activities and one activity can include several objects.

HOW COULD WE GET AN IDEA OF THE PROCEDURE FOR A RENTAL, AN INVENTORY, A STAFF MEMBER, THE CUSTOMER OR EVEN OF THE INVOICES?



A baseline approach for deriving a process flow from a database is to consider the perspectives of the object types. Therefore, we collect all activities for an object from the considered type, where it is involved and sort the activities by their timestamp.

Inspired by that approach, we thought on how the process flow would look, if we consider multiple object types at the same time instead of one. Thereby, you could easily derive where two objects are interacting with one another. Here, you would collect all activities, where both objects are involved at the same time.

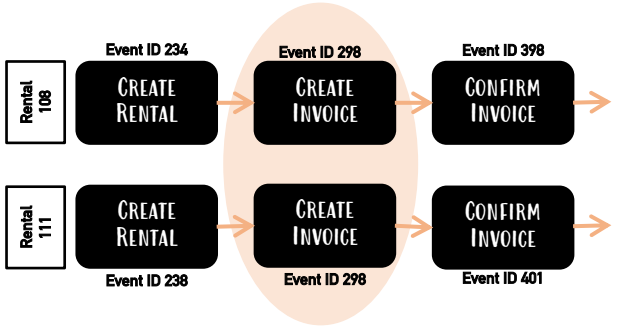
SO SMART THESE IDEAS MAY APPEAR,
THEY ALSO BRING PROBLEMS WITH THEM.

And here, we realize, that there will be no perspective, which comprises all activities since not all objects are always included within the performance of an activitiy at the same time. It only gives the possibility to understand how these objects behave. But there is no perspective which comprises all activities. Considering multiple object types at the same time, it visualizes the interactions between the object types and thereby generates a specific view on the process. Here, it helps to understand the complex dependencies between the objects.

As you remember, multiple different objects may be involved in the same activity. When collecting the activities for each object instance, there will be some activities which will appear more than once for each object instance and are separately saved (this behavior is called convergence). As an example, I have two rental orders, and the store sends for both rentals the invoices at the same time. So both rentals are passing that activity together, but this information isn't easily to access anymore in the end. Furthermore, if our rental involves multiple inventories, we can probably detect, that the inventories are returned one by one. The occurrence of the same activity instance within the same procedure is called divergence. If you have only a little knowledge about the procedures within a company, this might be confusing.

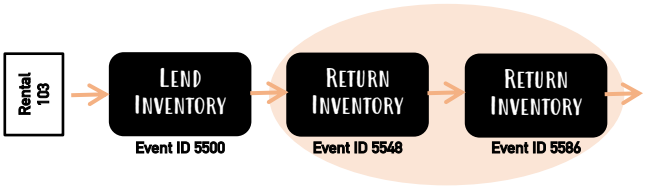
Convergence

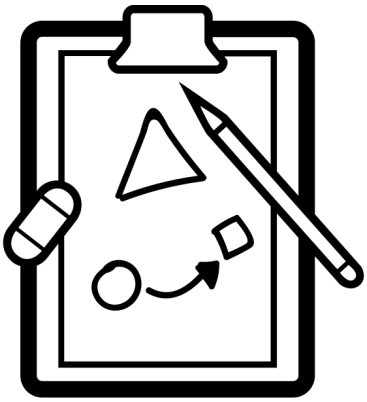
= the same performed activity appears in different object type perspectives



Divergence

= the same activity type appears multiple times in the same object type perspective

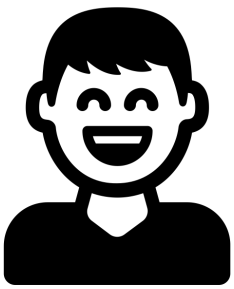




When you decide to visualize the perspectives of the object types, you will never be able to recover the complete procedure as we have discussed earlier. The moment you choose a perspective (object type) you have already accepted, that you will lose information about the other involved objects which can't be solved. That's why people are trying to create approaches to easily visualize such complex behavior of objects while to minimize convergence and divergence problems as much as possible.

We are within a conflict between completeness and clearness.

As you can see, the world of data is very deep and complex. But there are many ways to bring structure into it e.g by modelling and analyzing the renting procedures. Here, each object tells a different story.



End.