



# Optimizacije i transakcije

# Sadržaj

- Optimizacije
  - Indeksi
  - Operacije nad listama
  - *Batch* obrade
  - Optimizacije dobavljanja
  - *Tracing*
- Upravljanje transakcijama i zaključavanje
- Linkovi:
  - Rad sa datotekama i slikama
  - *Import/export* XML dokumenata
  - *Scheduled microflows*
  - Pozivanje Java akcije
  - Debugovanje

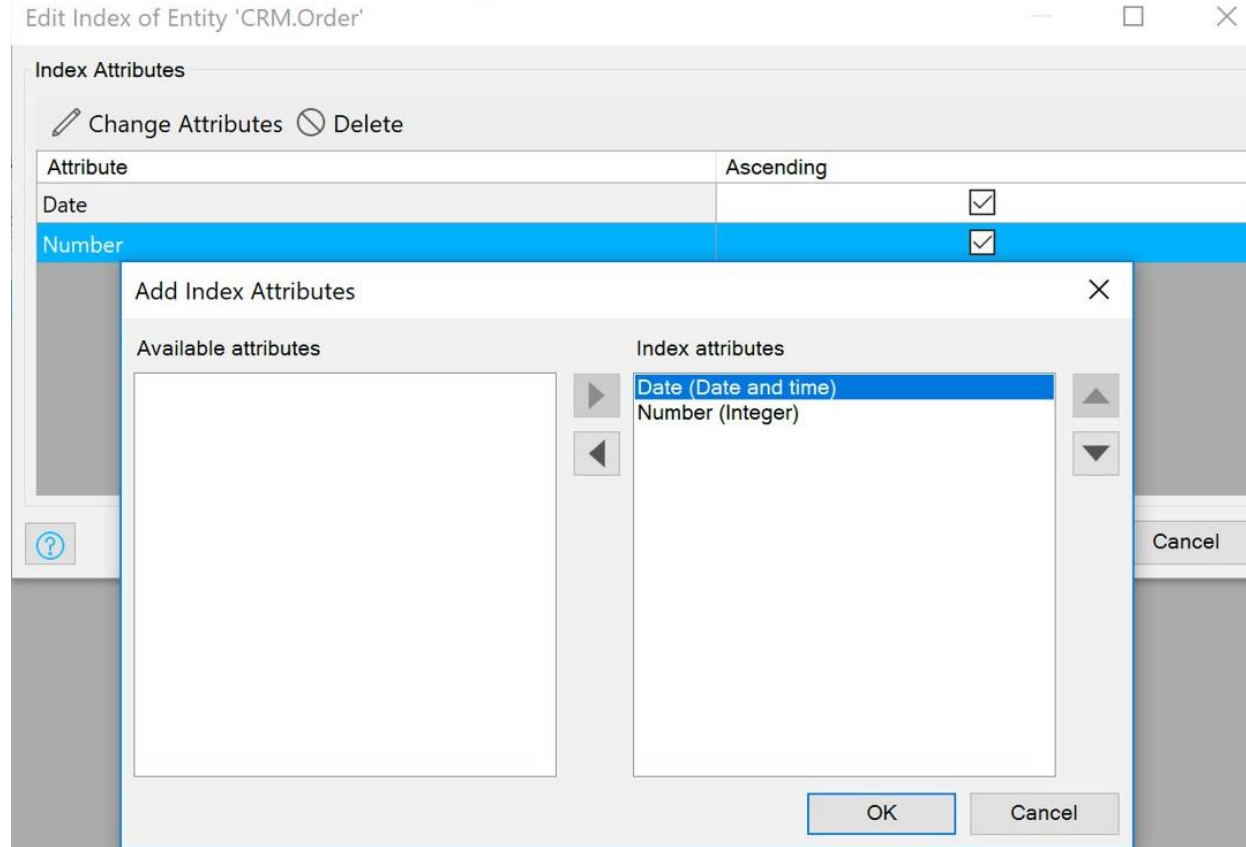
# Optimizacije - indeksi

- Indeksi su lista atributa za koje se u bazi podataka kreiraju indeksi
  - Može ih biti više, i mogu biti višekolonski
  - Kreiraju se samo nad *persistable* entitetima
  - Poboljšavaju performanse upita
- 
- Indeksi su uređeni - ako se kreira indeks nad više kolona, bitan je redosled tih kolona. Redosled bi trebalo da odgovara redosledu obeležja po kojima se vrši pretraga, kako bi se dobilo na performansama.
  - Ako je pretraga po jednom obeležju, na ubrzanju se dobija samo ako je to za indeks prvo obeležje.

# Optimizacije - indeksi

- Uvođenje indeksa usporava upis i brisanje objekata iz baze, jer se indeksi u bazi moraju osvežiti
- O njihovom uvođenju treba dobro promisliti – uvesti ih tek kada je potrebno ubrzanje performansi
- Napomene:
  - Koristiti indekse kada je obeležje bar donekle jedinstveno
  - Indeksi se ne stavljaju na attribute tipa string sa varijabilnom dužinom
  - Od string operacija prilikom pretrage, ima smisla koristiti samo startsWith

# Optimizacije - indeksi

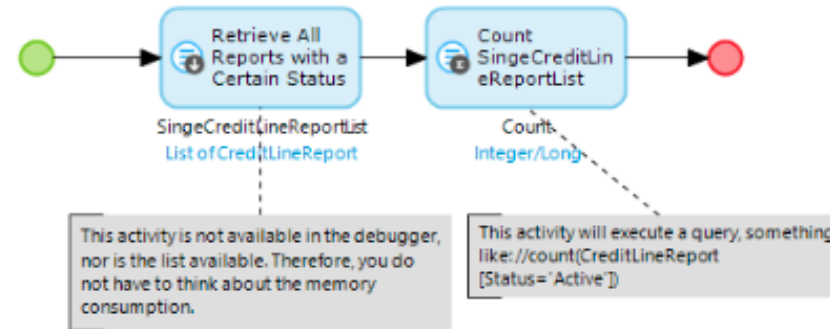


Kreiranje indeksa

# Optimizacije – operacije nad listama

- Mendix pruža podršku za agregaciju podataka u listi (sum, avg, minimum...), preko aktivnosti Aggregate List
- Kako bi se smanjilo opterećenje na RAM memoriju, Mendix podržava optimizovano izvršavanje ovakvih agregacionih upita ka bazi
- Kada se podaci dobavljeni iz baze preko Retrieve aktivnosti koriste samo za ulaz u takvu agregacionu akciju, i uz to se dobavljaju svi podaci koji zadovoljavaju XPath izraz, platforma može da spoji ove dve aktivnosti u jednu akciju

# Optimizacije – operacije nad listama



Dve aktivnosti – spajanje u jednu akciju

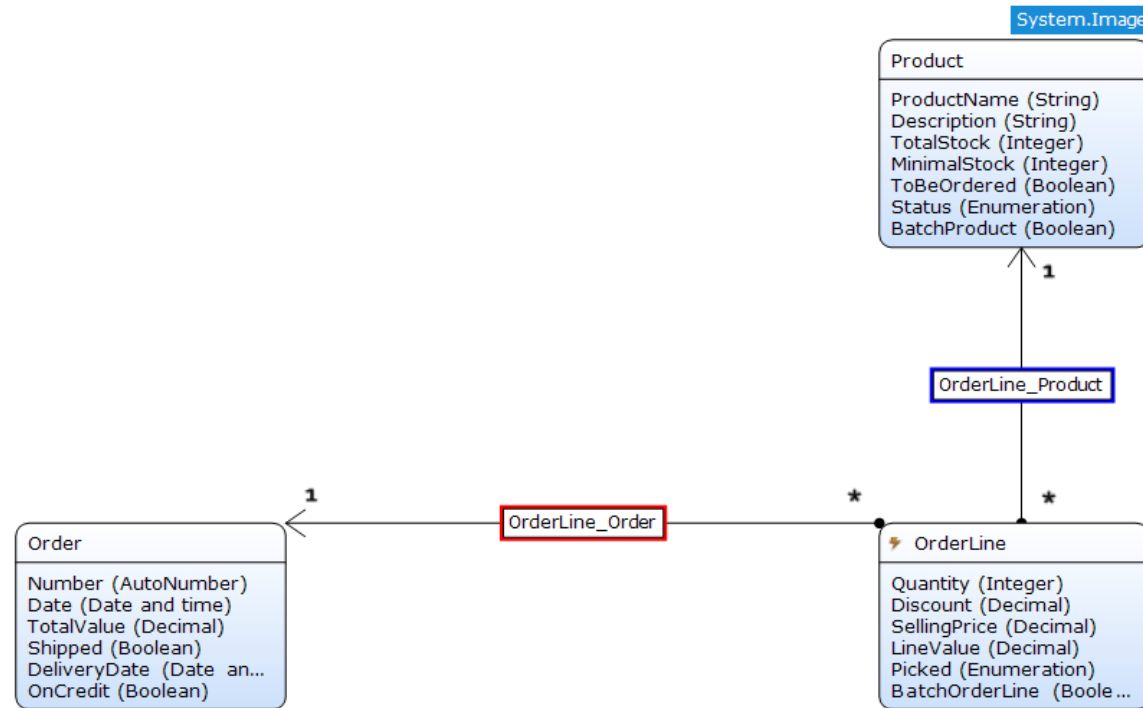
- Redovi iz tabele se nikad neće dobiti u RAM
- Lista ne sme biti upotrebljena ni u jednoj više aktivnosti, jer Mendix tada neće odraditi optimizaciju

# Optimizacije – dobavljanje podataka

- By Association ili From Database – koja varijanta ima više smisla sa stanovišta performansi?
- Ako se podaci nalaze u RAM-u, By Association
- A ako se ne nalaze? Možemo izazvati N+1 problem!



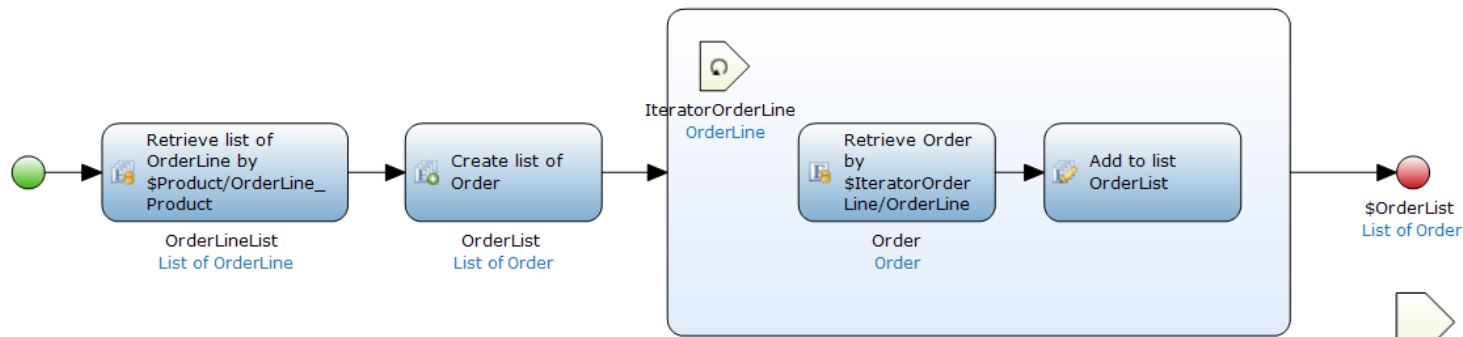
# Optimizacije – dobavljanje podataka



- Dobavljanje svih narudžbi za određeni proizvod

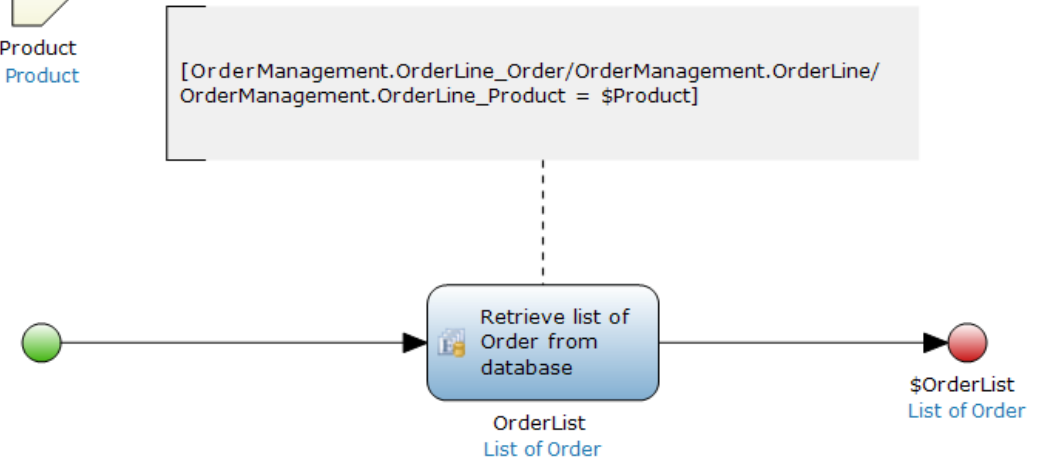
# Optimizacije – dobavljanje podataka

Product  
Product



Dobavljanje po asocijaciji kroz petlju

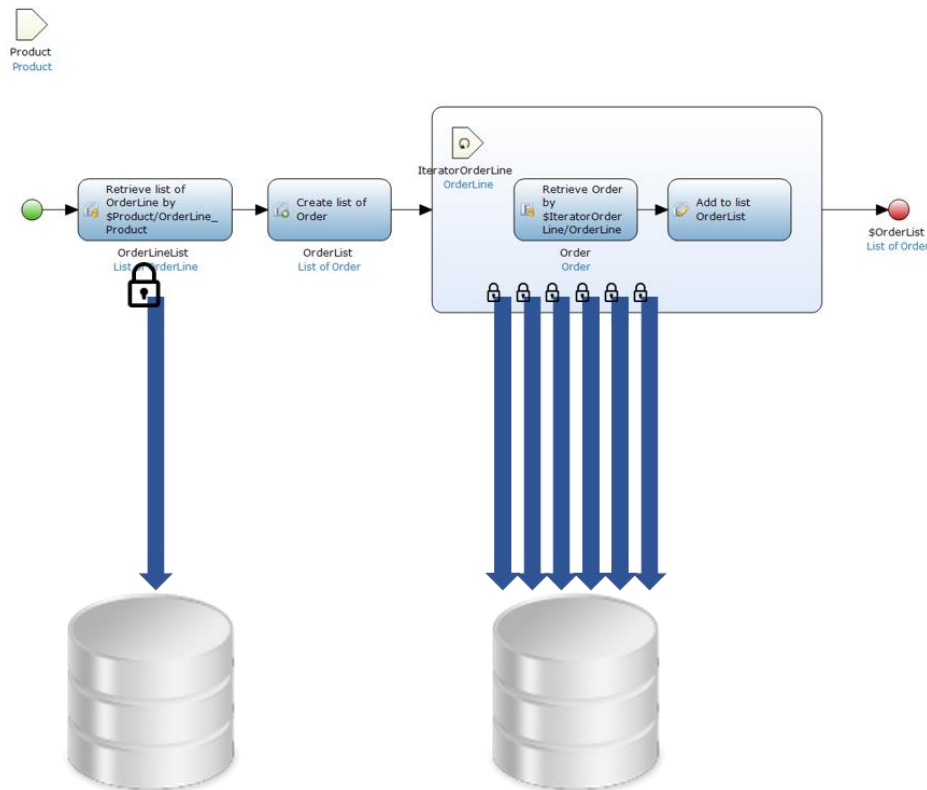
Product  
Product



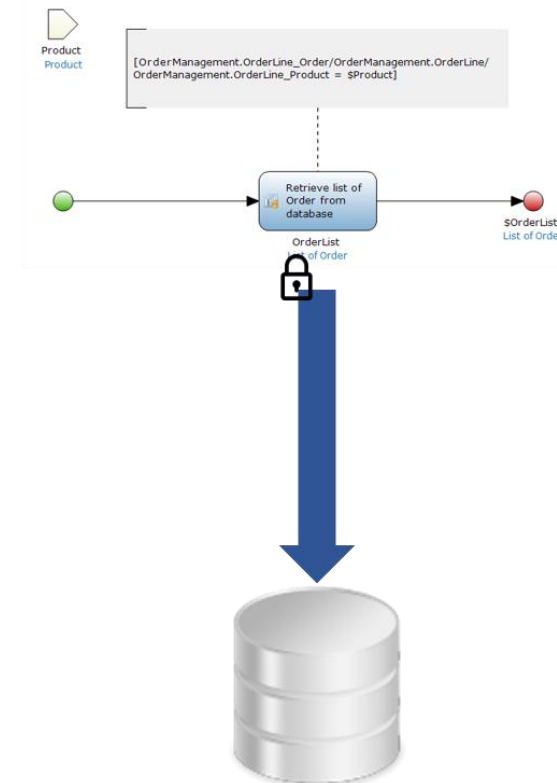
Dobavljanje iz baze podataka

# Optimizacije – dobavljanje podataka

## By association



## From database

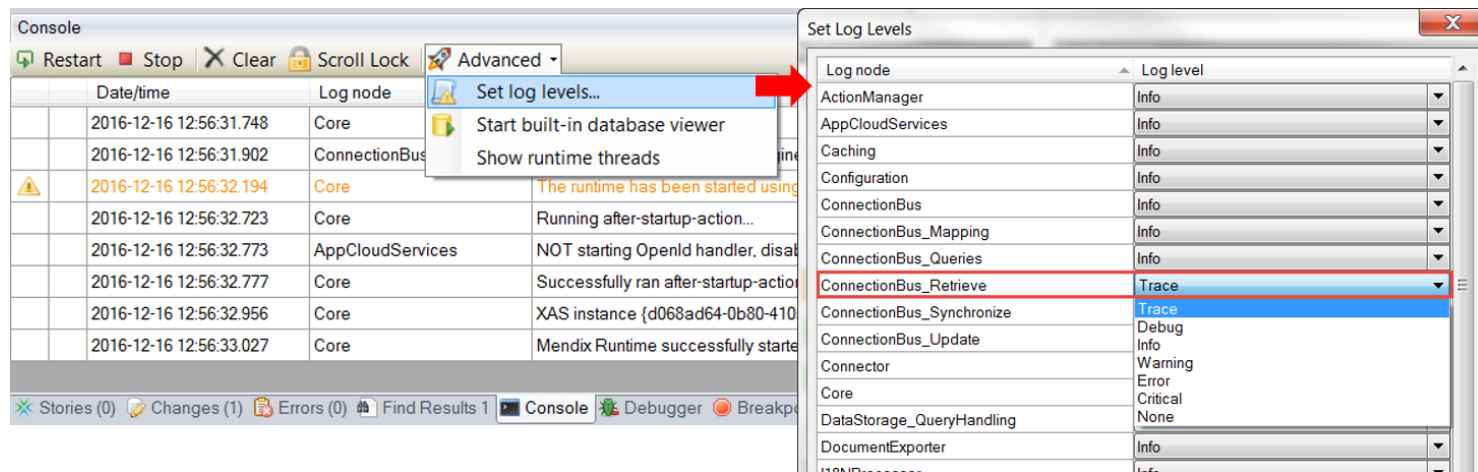


Rezultujući upiti ka bazi podataka

- Ako se podaci ne nalaze u RAM-u, kao što je ovde slučaj, imamo N+1 upita ka bazi

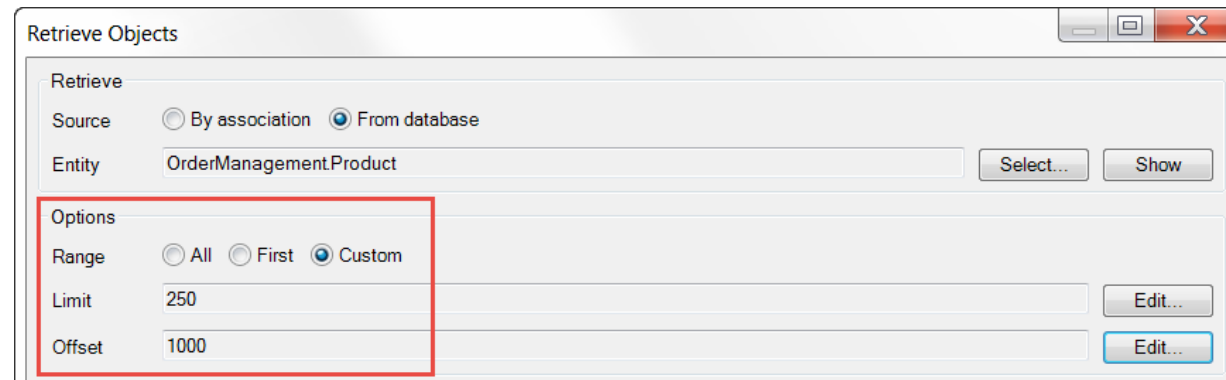
# Optimizacija – dobavljanje podataka

- Mendix omogućava praćenje komunikacije sa bazom - Tracing



# Optimizacije – opšte preporuke

- Operacije nad podacima prepustiti bazi podataka kad god je to moguće, optimizovana je za to
- Operacije nad velikim skupovima podataka raditi paginirano, što Retrieve From database i omogućava.



The screenshot shows a 'Retrieve Objects' dialog box with the following settings:

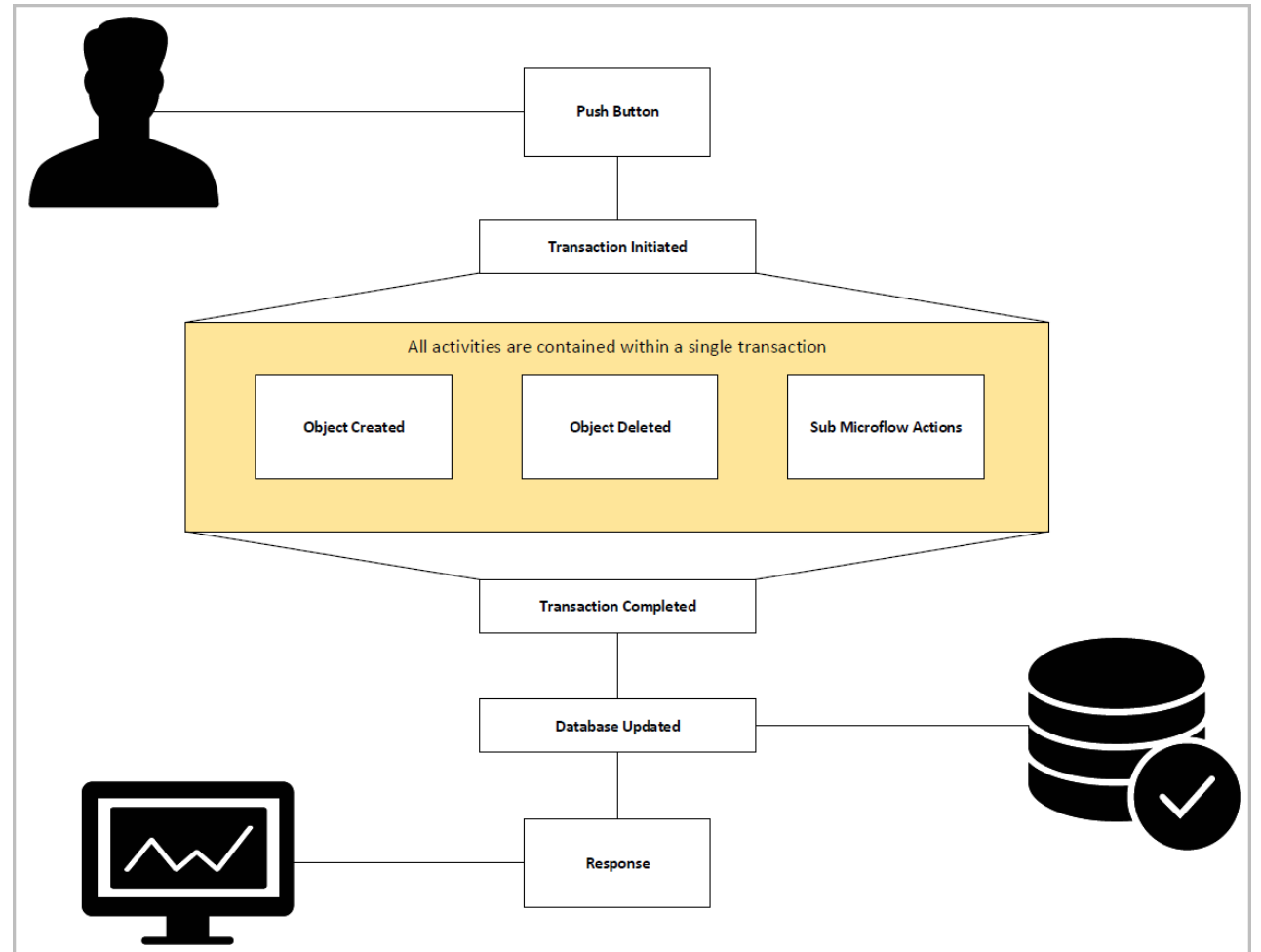
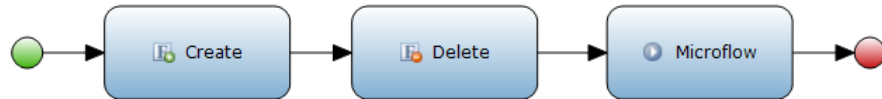
- Retrieve**
  - Source: ☐ By association ☒ From database
  - Entity: OrderManagementProduct (with 'Select...' and 'Show' buttons)
- Options** (highlighted with a red box)
  - Range: ☐ All ☐ First ☒ Custom
  - Limit: 250 (with 'Edit...' button)
  - Offset: 1000 (with 'Edit...' button)

# Upravljanje transakcijama

- Mendix Runtime je *engine* za izvršavanje zasnovan na transakcijama
- Sve aktivnosti u okviru transakcije su izolovane dok god se transakcija ne završi
- Mendix podrazumevano koristi *Read Committed* nivo izolacije
- Podrazumevano ponašanje je takvo da se svaki *microflow* izvršava u zasebnoj transakciji. Ako se ne specificira drugačije, ili se cela takva transakcija izvrši, ili se ništa ne izvrši nad bazom
- *Sub-microflow* procesi se podrazumevano izvršavaju u okviru iste transakcije u okviru koje su pozvani
- Ako se u okviru transakcije desi greška, podrazumevano sve izmene se poništavaju (*rollback*), i korisniku se prikazuje greška

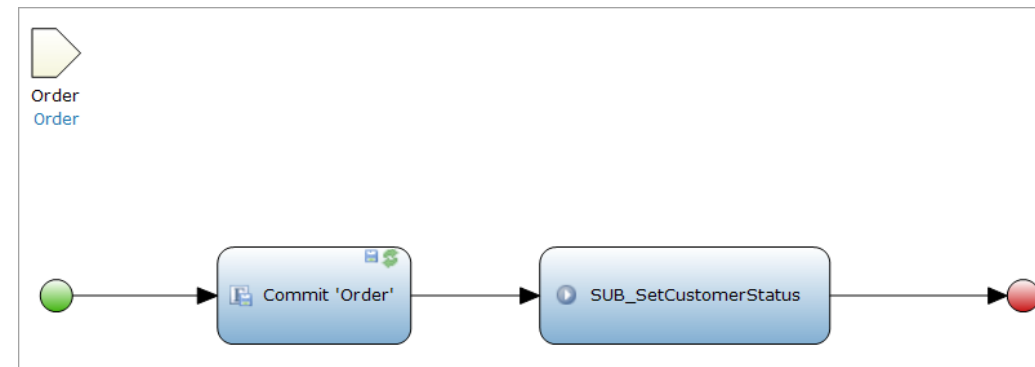
# Upravljanje transakcijama

- Podrazumevano ponašanje:

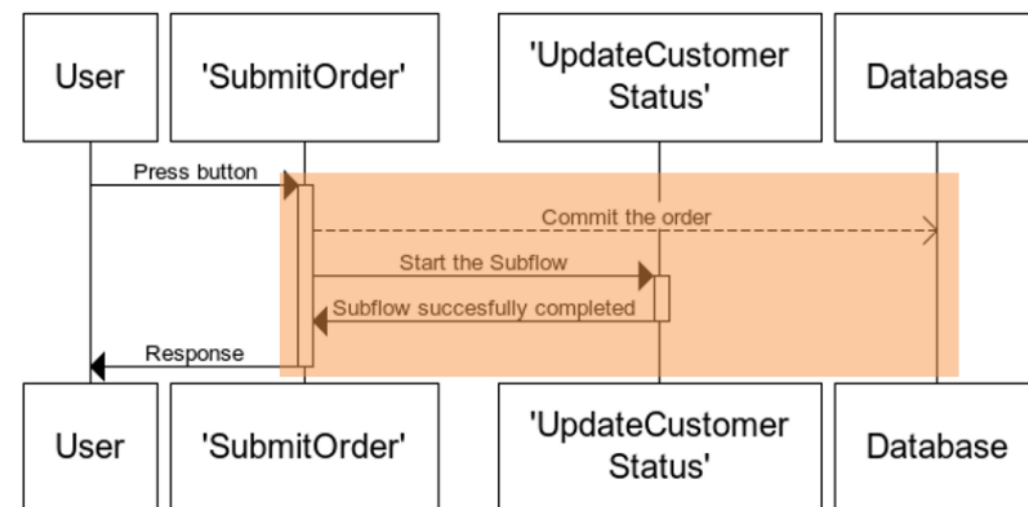


# Upravljanje transakcijama

- Primer: Čuva se narudžba. Korisniku se menja status ako dostigne određenu sumu u narudžbama (sa *silver* na *gold*).
- Šta ako se prilikom poziva *sub-microflow*-a desi greška (*error*)? Podrazumevano, sve se poništava, i korisniku se prikazuje greška!



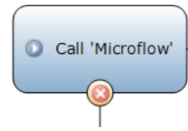
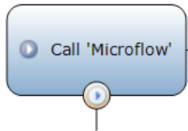
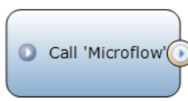
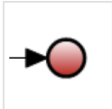

## Basic Transaction





# Upravljanje transakcijama – upravljanje greškama

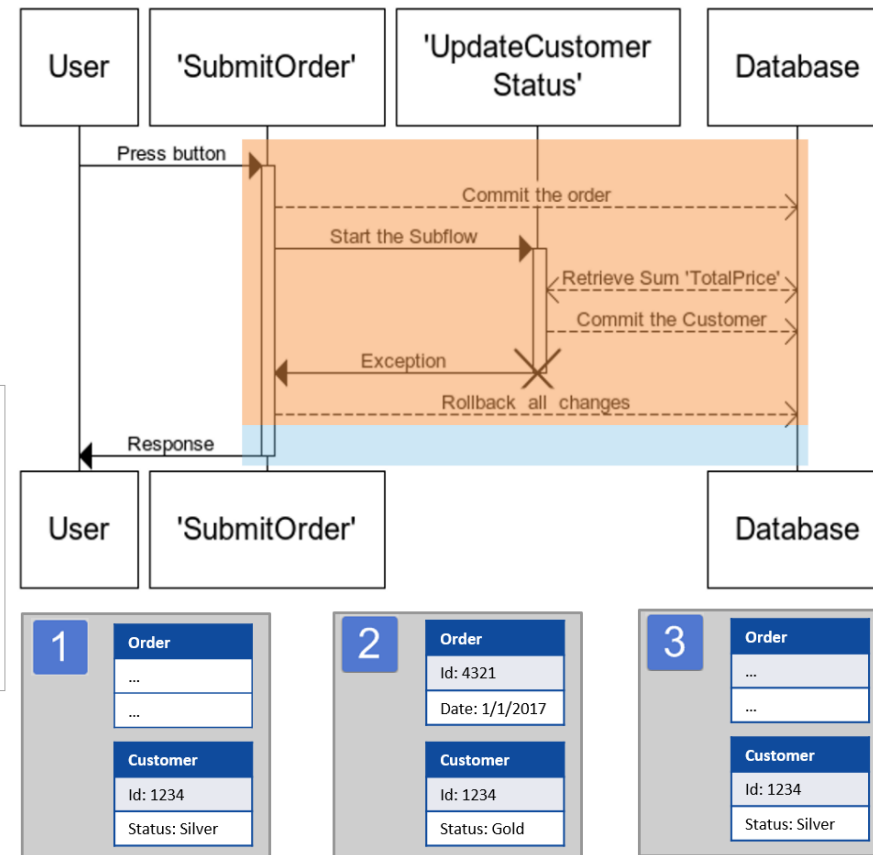
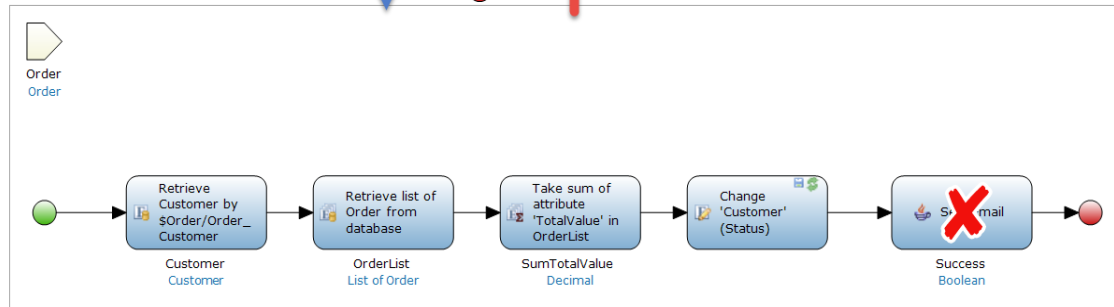
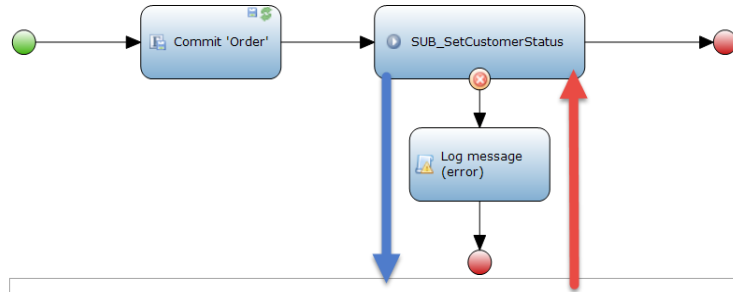
- Izmena podrazumevanog načina rukovanja greškom, pomoću sledećih elemenata (prva tri su rukovaoci greškama, a druga dva su događaji)

Type	Image	Description
Custom with Rollback		Everything that happened prior to the error will be rolled back. A new transaction will be initiated, and only the changes executed in the error handler flow will be executed.
Custom without Rollback		Any action taken inside the sub-microflow will be reverted. Everything that happened before the error will be preserved, and the microflow will continue using the custom error handler flow.
Continue		Any action taken inside the microflow will be reverted. Everything that happened before the error will be kept, and the microflow will continue as if nothing happened. <i>Note: You should only use this in more complicated combinations of multiple error handlers. You want to make sure you at least log the error message. If it breaks, you must be able to trace it.</i>
End Event		This is the end of the microflow transaction and all actions are executed at the end of the main microflow.
Error End Event		This re-throws the error to all the parent microflows after executing the custom activities. The error handling on the activity calling this microflow determines how the transaction is processed further.

# Upravljanje transakcijama – upravljanje greškama

- Rukovanje greškom - Custom **with** rollback

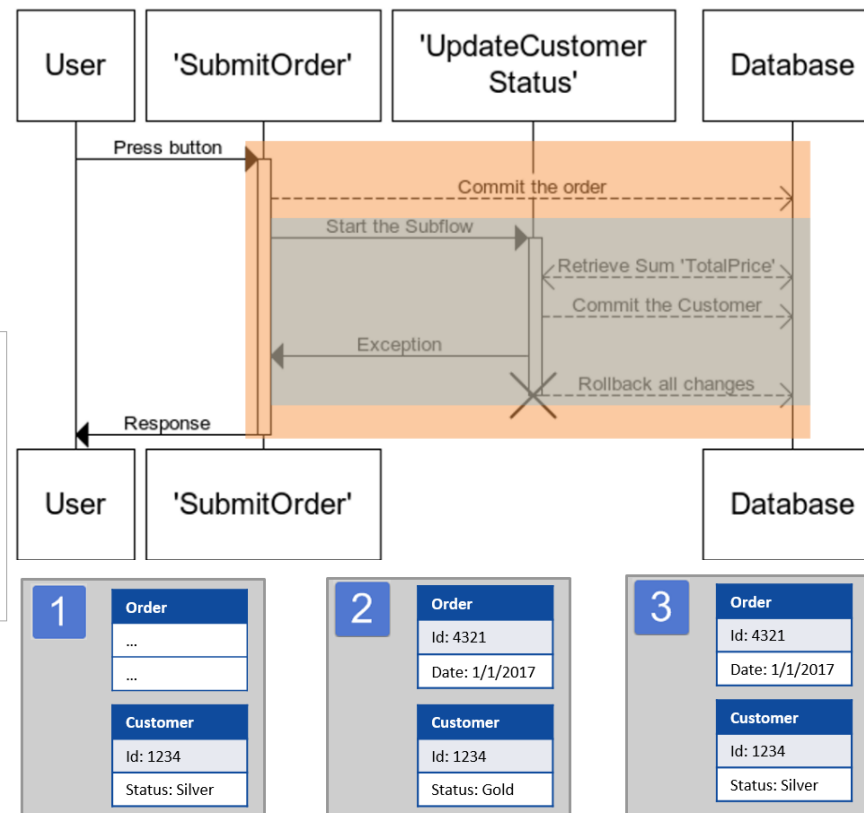
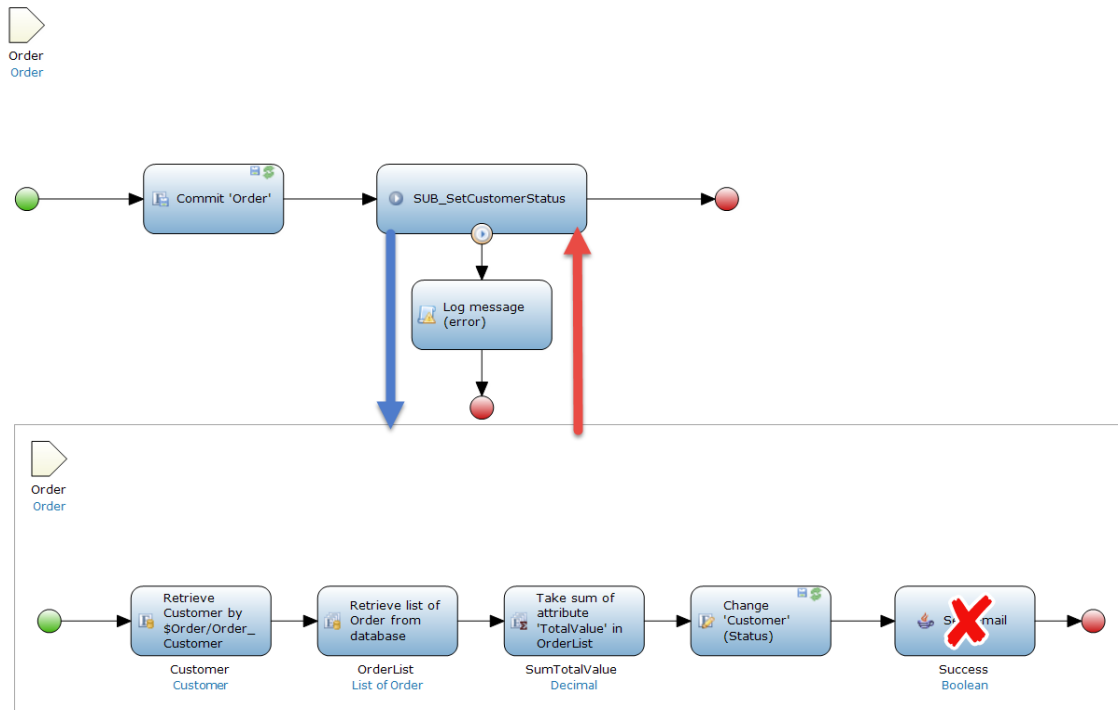
Order  
Order



- Sve se poništava, ali se prilagođava rukovanje greškom. Na primeru se samo izvršava logovanje, ali nakon hvatanja greške, moguće je i komunicirati sa bazom, u okviru nove transakcije.

# Upravljanje transakcijama – upravljanje greškama

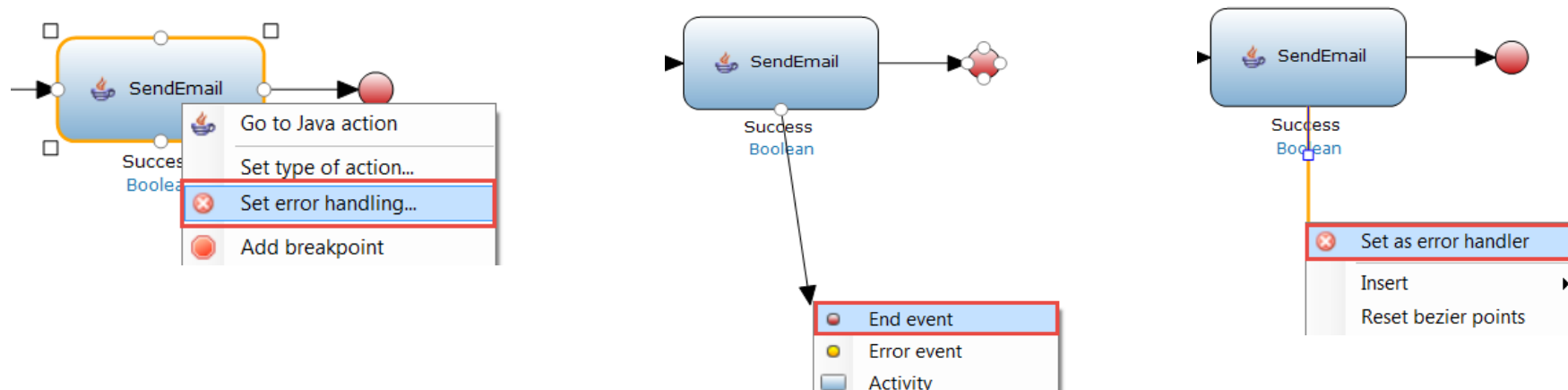
- Rukovanje greškom - Custom **without** rollback



- Pre poziva sub-microflow-a se pravi *save point*! Samo izmene u okviru njega se poništavaju, dok se narudžba čuva. Kao i u prethodnom primeru, nakon dešavanja greške, izvršava se prilagođeno rukovanje. *Continue* način se isto ponaša, samo se ne rukuje greškom.

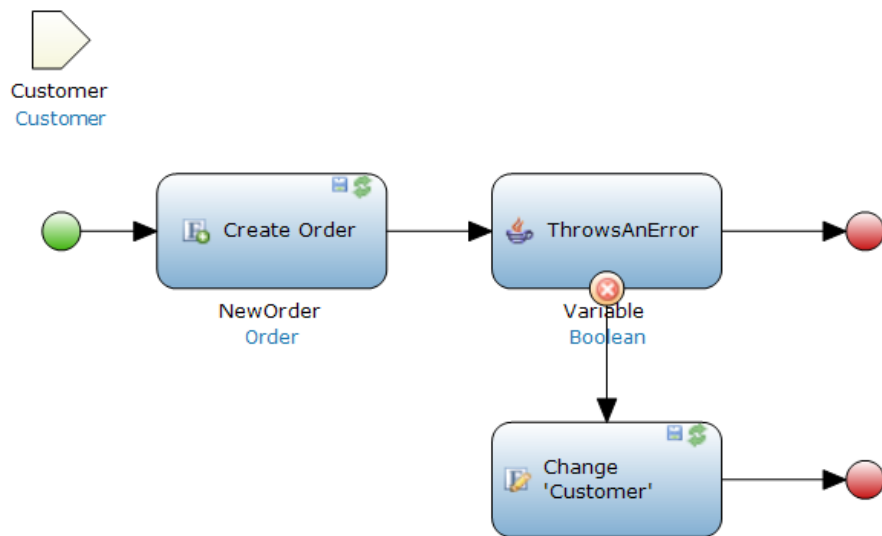
# Upravljanje transakcijama – upravljanje greškama

- Nije omogućeno stavljanje sva tri načina rukovanja greškama nad svim komponentama. Zavisí od konteksta – nad *loop* samo *rollback* i *continue*, poziv *microflow-a* omogućava sva tri...
- Postavljanje rukovaoca greškom (*error handler-a*):

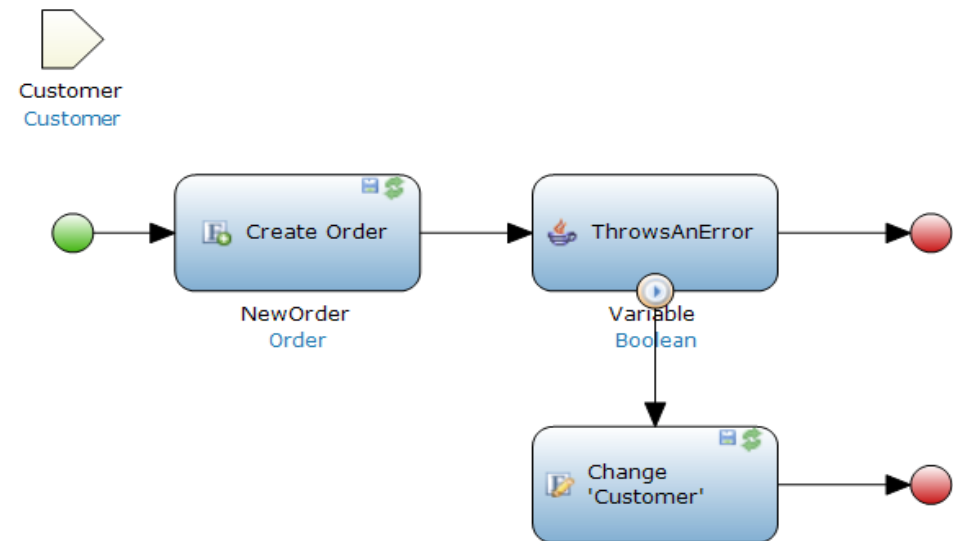


# Upravljanje transakcijama – upravljanje greškama

- Šta će biti komitovano?



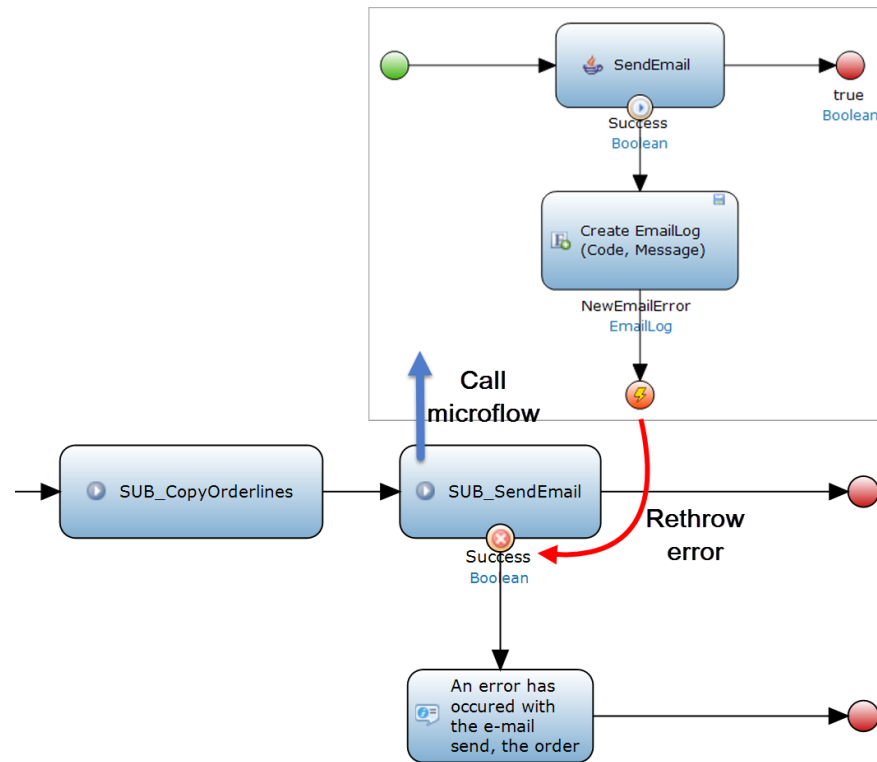
Primer 1



Primer 2

# Upravljanje transakcijama – upravljanje greškama

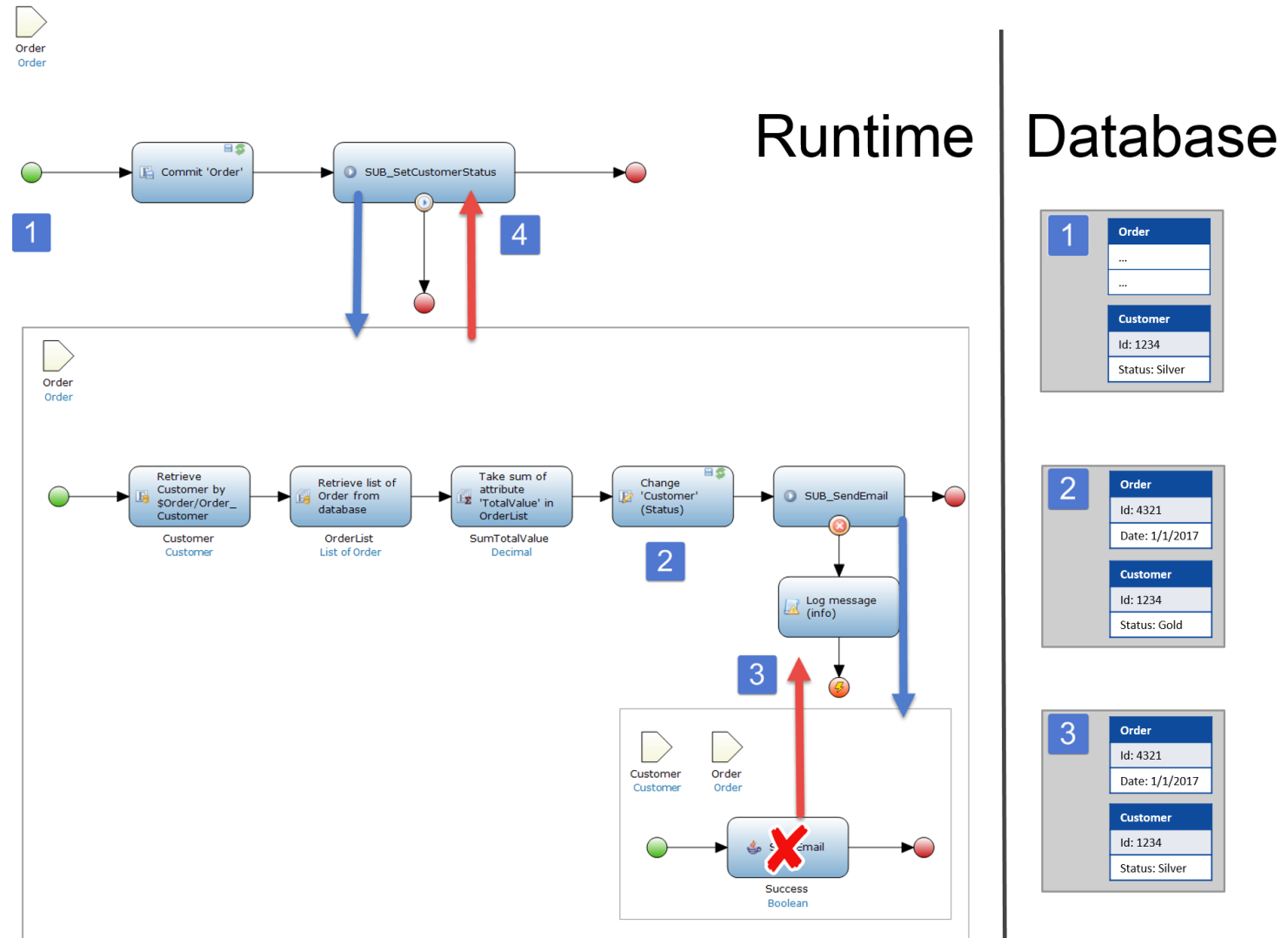
- Moguće je propagirati grešku



Ponovo bacanje pomoću error event-a

# Upravljanje transakcijama – upravljanje greškama

- Primer:



Poništavanje izmena u *sub-microflow*-u

# Upravljanje transakcijama – upravljanje greškama

- Životni vek transakcija

Microflow action	Database action
Any database action (retrieve or commit)	Begin transaction (if not present)
Before action with error handling <i>Custom without rollback</i>	Start save point
After action with error handling <i>Custom without rollback</i>	Release save point
Error (custom) with rollback	Rollback transaction
Error on custom without rollback	Rollback save point
Before commit action	Start save point
After commit action	Release save point
End request	End transaction

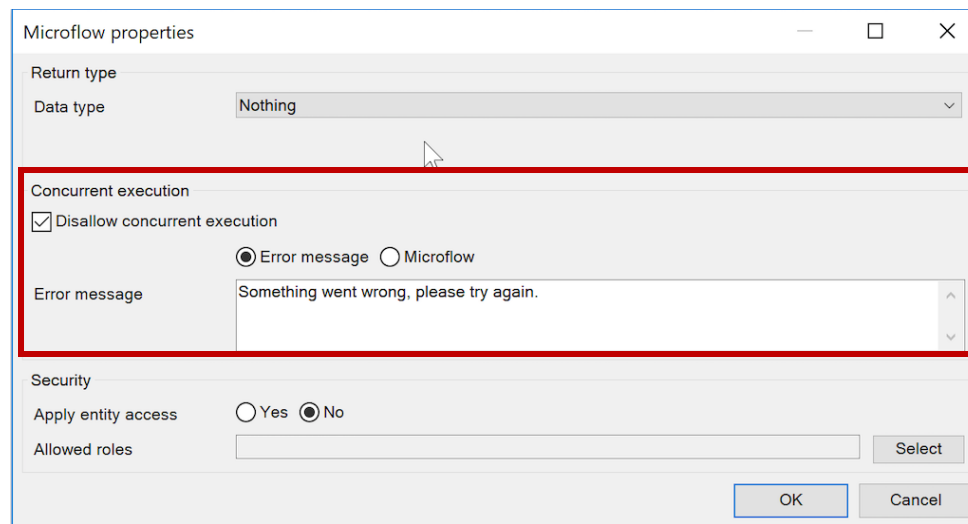


# Upravljanje transakcijama

- Prilikom izmene podataka, Mendix zaključava objekte dok traje transakcija
- Ako je jedna transakcija zaključala objekat, sve ostale transakcije moraju da sačekaju da se ta transakcija završi, prilikom čega se navedeni objekat automatski otključava
- Ovo ne sprečava da dva korisnika istovremeno menjaju isti objekat. Kako bismo rukovali takvom situacijom, potrebni su nam drugi mehanizmi!

# Upravljanje transakcijama – zaključavanje

- Mendix ne nudi podršku za optimističko i pesimističko zaključavanje
- Jedno od mogućih rešenja jeste da se onemogući konkurentno izvršavanje nekog *microflow*-a:



The screenshot shows the 'Microflow properties' dialog box. The 'Return type' section has 'Data type' set to 'Nothing'. The 'Concurrent execution' section is highlighted with a red rectangle and contains the following settings:

- ☒ Disallow concurrent execution
- ☒ Error message ☐ Microflow
- Error message: Something went wrong, please try again.

The 'Security' section at the bottom shows 'Apply entity access' set to 'No' and an empty 'Allowed roles' field with a 'Select' button. 'OK' and 'Cancel' buttons are at the bottom right.

- Za pesimistički pristup zaključavanju, moguće je iskoristiti i sledeći modul sa *App Store*-a:  
<https://marketplace.mendix.com/link/component/109405>

# Linkovi:

<https://docs.mendix.com/howto8/data-models/working-with-images-and-files>

<https://docs.mendix.com/refguide/scheduled-events/>

<https://docs.mendix.com/howto8/logic-business-rules/extending-your-application-with-custom-java>

<https://docs.mendix.com/howto8/monitoring-troubleshooting/debug-microflows/>

<https://docs.mendix.com/howto8/monitoring-troubleshooting/debug-microflows-remotely/>

Hvala na pažnji!

