ITSE333A ABAP

Lesson 6 : Specialties for ERP software

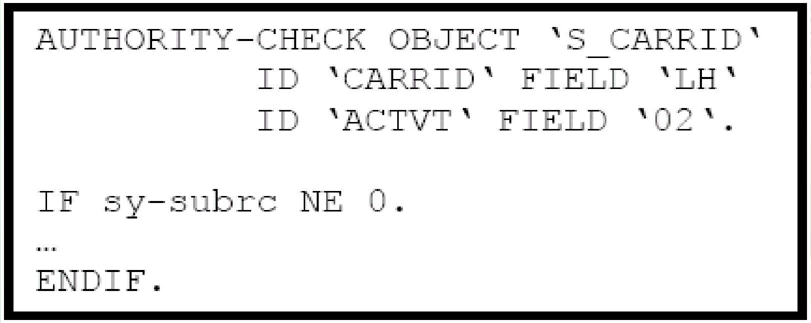
Anthony D. Aquino

Agenda

1. Authorizations
2. Lock objects
3. Logical units of work
4. Updater

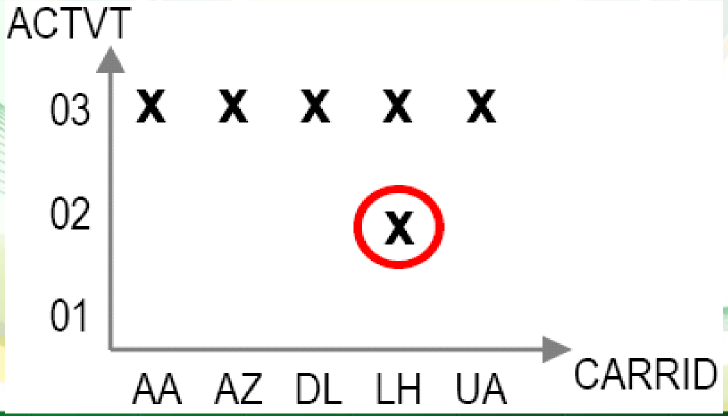
Authorizations

- Before starting a program user authorizations should always be checked
- But: authorizations are not checked by the system
- Every ABAP program has to check authorizations by itself
- Different authorization levels: READ (01), CHANGE (02), DELETE (03) for every data set
- Transaction: SU03



```
AUTHORITY-CHECK OBJECT 'S_CARRID'
          ID 'CARRID' FIELD 'LH'
          ID 'ACTVT' FIELD '02'.

IF sy-subrc NE 0.
…
ENDIF.
```



Locking concept

- What happens when several users want to work on the same data set? → data inconsistency
- Lock objects prevent simultaneous changes on data set
- Beside the lock concept of the database, SAP implements its own lock concept
- SAP does not use the database’s lock concept
- Deadlock:
  - May occur when two programs wait for each other to lock data set
  - Rare

Locking concept

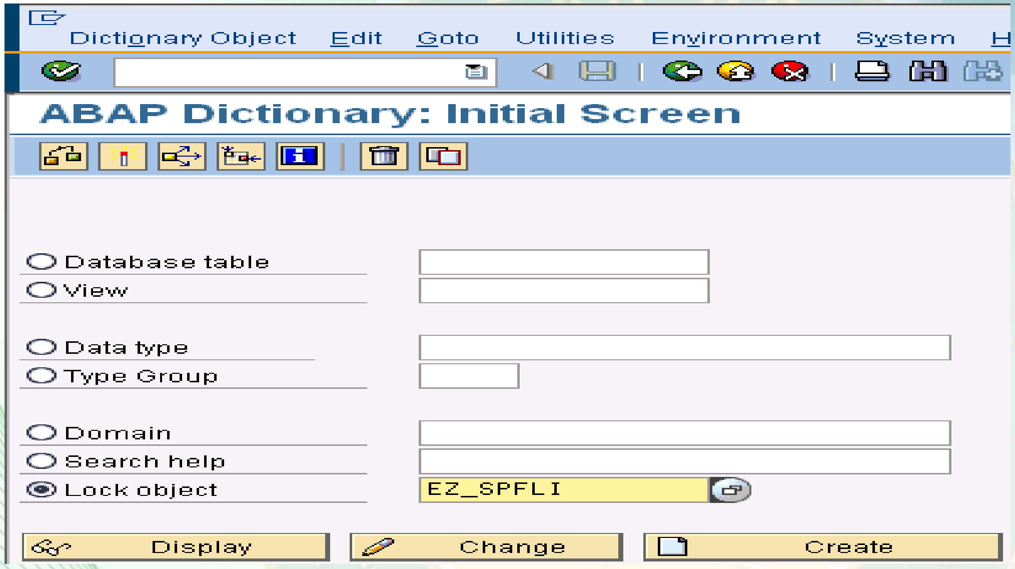
Modes:

1. Read lock = shared lock
   - Locks data set for reading
   - Several shared locks may exist on one data set at the same time
   - Prevents the SAP system from creating an exclusive lock on a data set
2. Write lock = exclusive lock
   - Locks data set for writing/changing
   - No shared locks, only one is allowed at the same time
   - First come, first serve

Locking concept

Creation of a lock object:

- Creation can be done in data dictionary (**SE11**)
- One lock object per database table
- Lock objects are generated automatically by using the primary key
- There is only one lock object dealing with all lock modes



ABAP Dictionary: Initial Screen

○ Database table

○ View

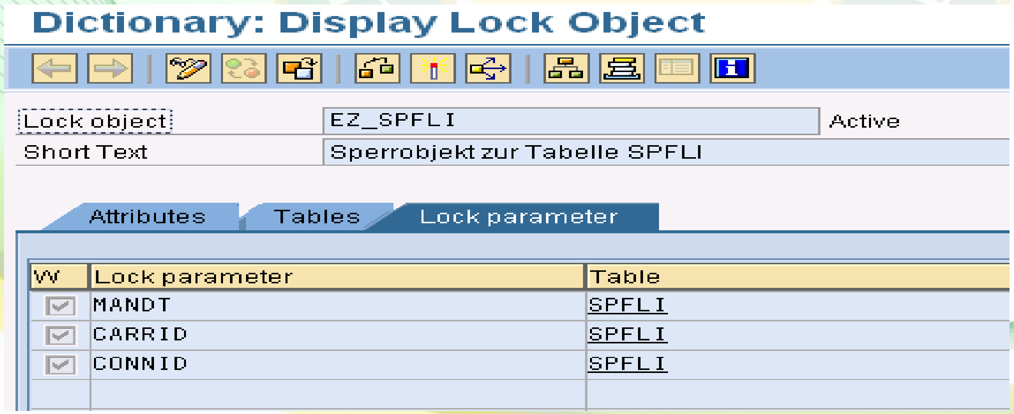
○ Data type

○ Type Group

○ Domain

○ Search help

● Lock object EZ_SPFLI



Dictionary: Display Lock Object

Lock object EZ_SPFLI Active

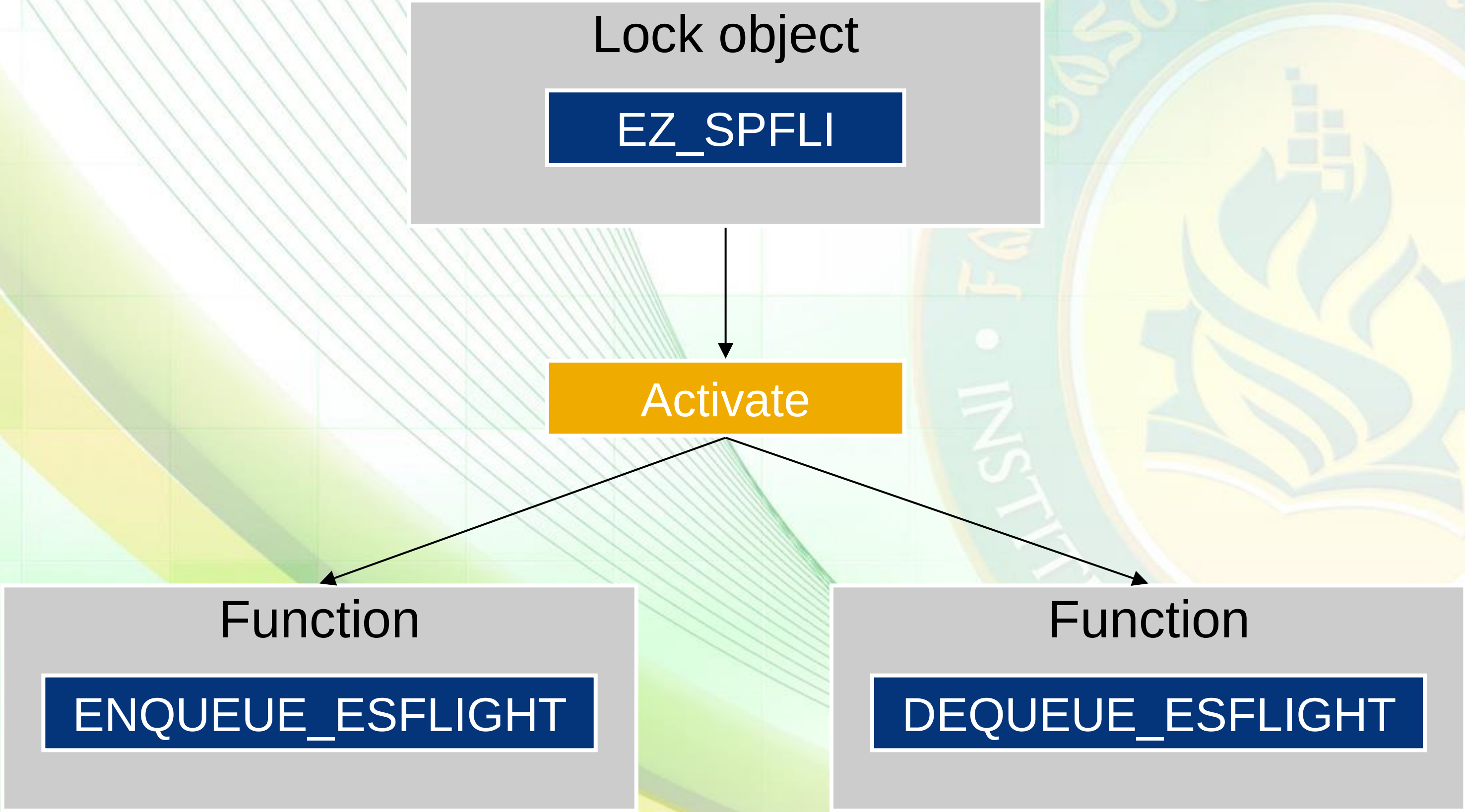
Short Text Sperrobjekt zur Tabelle SPFLI

W	Lock parameter	Table
☑	MANDT	SPFLI
☑	CARRID	SPFLI
☑	CONNID	SPFLI

Usage of lock objects

- Two function modules are generated automatically when creating a lock object:
  - Enqueue_<lock object> → lock data set
  - Dequeue_<lock object> → unlock data set
- Before writing/reading data set → lock data set
- After writing/reading data set → unlock data set
- In case of an error → unlock data set
- Use button ‘PATTERN’ to avoid misspellings
- Locks can be viewed and deleted in transaction **SM12**

Generation of lock objects



Lock object

EZ_SPFLI

Activate

Function

ENQUEUE_ESFLIGHT

Function

DEQUEUE_ESFLIGHT

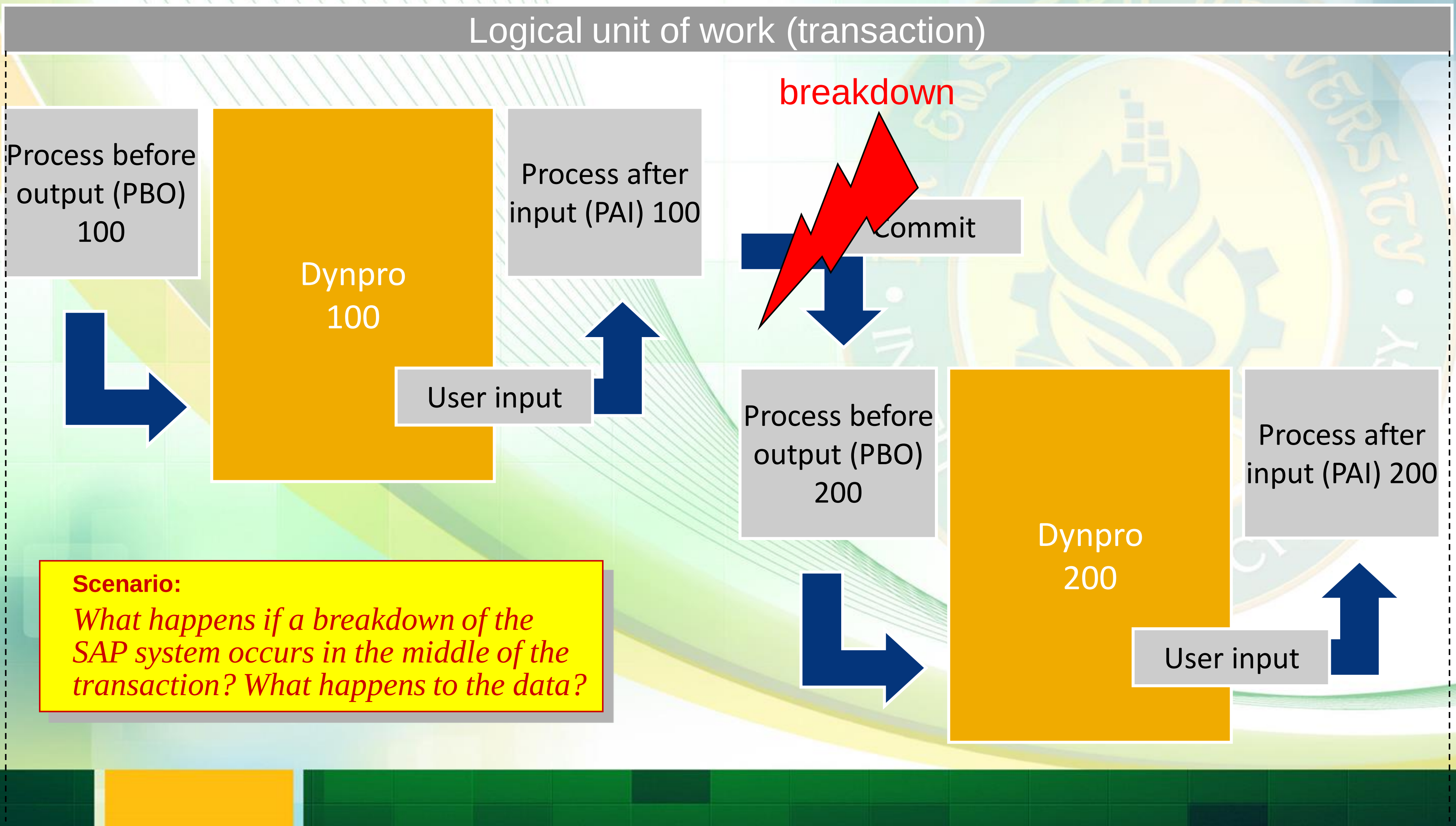
→ Lock data set

→ Unlock data set

ENQUEUE parameter‘s

Parameter	Value	Meaning
mode_<tabname>	‘S‘ ‘X‘	Read lock (shared) Exclusive lock (not shared)
<lock parameter>	<value>	Field values used to lock the data set
_wait	SPACE ‘X‘	If foreign lock, no new attempt If foreign lock, new attempt
_collect	SPACE ‘X’	Lock without local lock container Lock with local lock container

Logical unit of work (LUW)


Logical unit of work (transaction)
breakdown
Process before output (PBO) 100
Dynpro 100
User input
Process after input (PAI) 100
Commit
Process before output (PBO) 200
Dynpro 200
User input
Process after input (PAI) 200

Scenario:

What happens if a breakdown of the SAP system occurs in the middle of the transaction? What happens to the data?

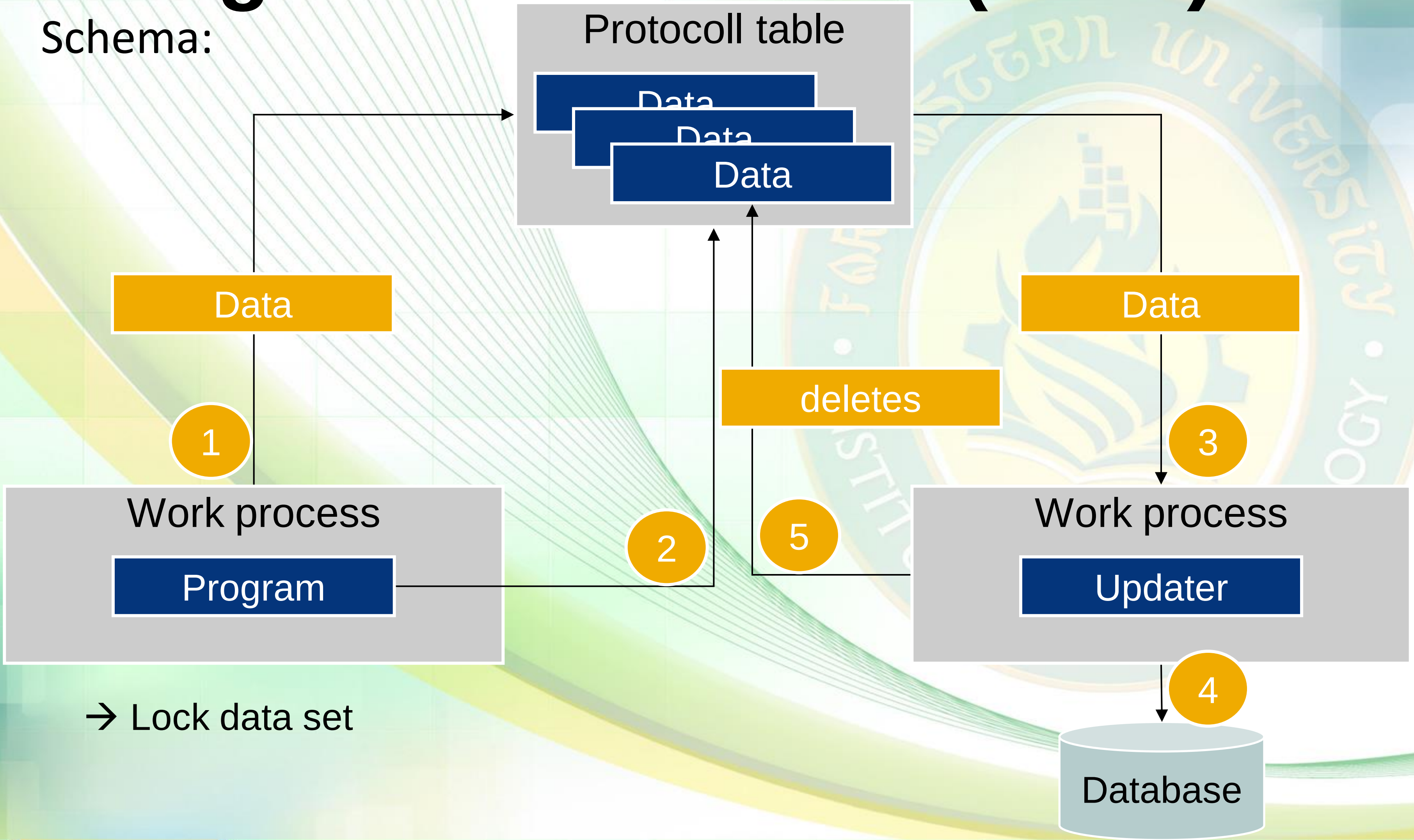
Logical unit of work (LUW)

- Problem:
  - New data sets are transferred into the database by each screen
  - Only if last screen of the program is reached and all data sets are written to the database the data are correct
- Solution
  - Updates to database are written to the updater process
  - Only when last screen of the program is reached, the updater receives a trigger to write data to database (commit) or to dismiss (rollback)

→ ensures data integrity

Logical unit of work (LUW)

- Schema:



→ Lock data set