Introduction to the ETL

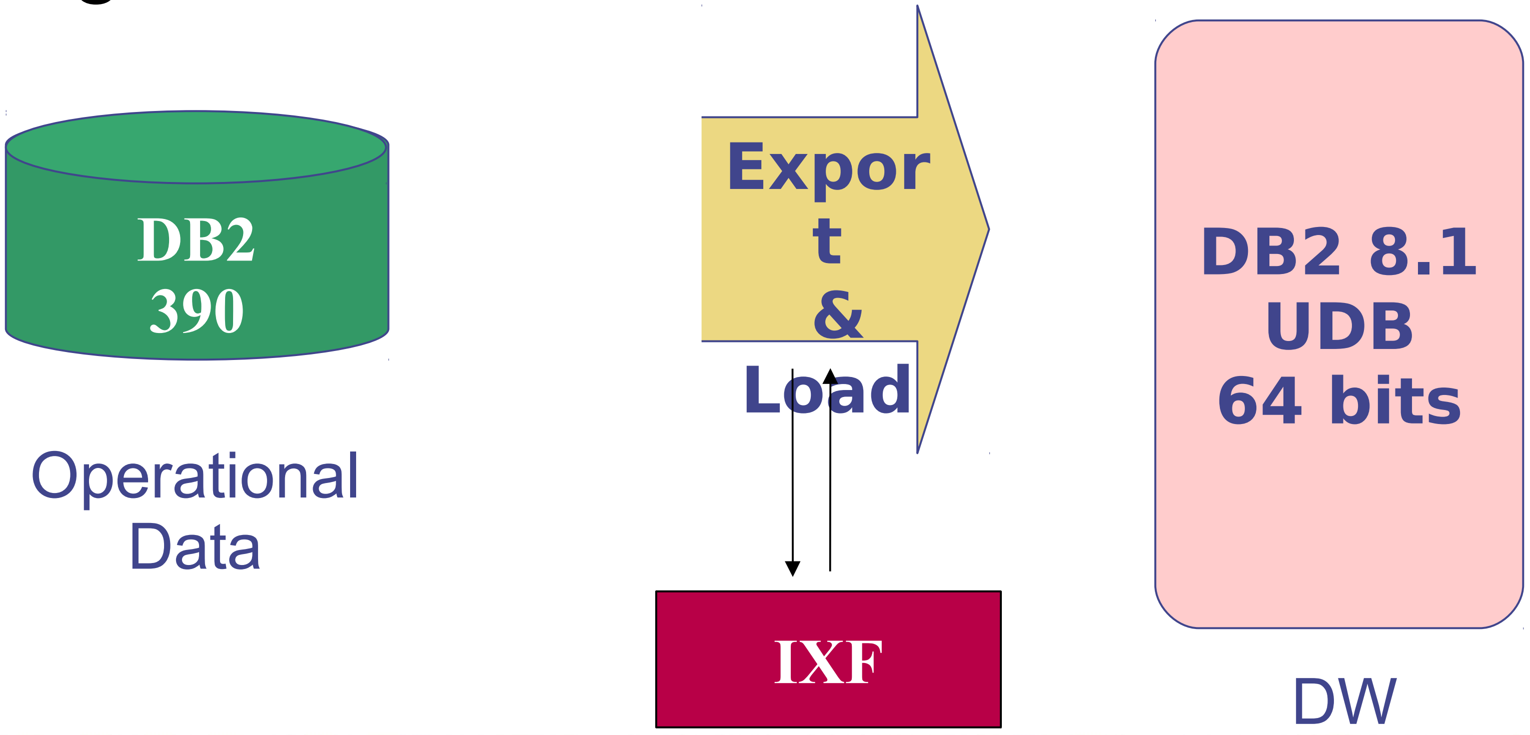
- ETL systems are highly time consuming and the great amounts of data these systems must deal with are increasing constantly.
- Nowadays hardware capabilities and parallel techniques will provide us new ways to increase performance.
- Our goal: To build a simplified DataWareHouse, by feeding a DB2 UDB (DSS) from Operational data located at DB2 Z/OS.

Extraction, Transformation and Load (ETL)

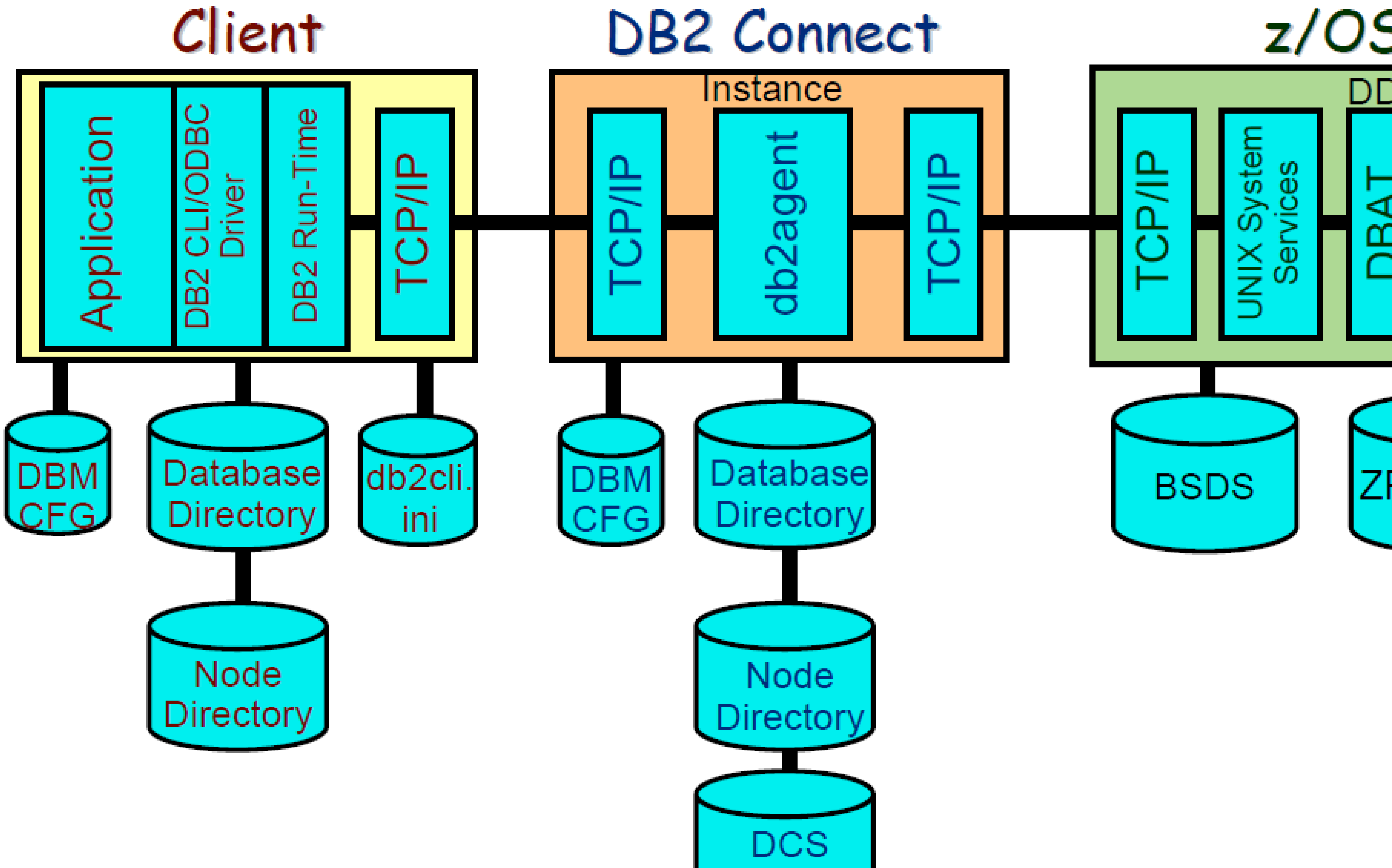
- The Extraction, Transformation and Load (ETL) is a common process in DataWareHouse systems.
- This process can involve huge amount of data which makes it highly time consuming.
- The computing kernel is inherently sequential due to its data dependencies and involves several devices like I/O, network, memory and computing.

Goal

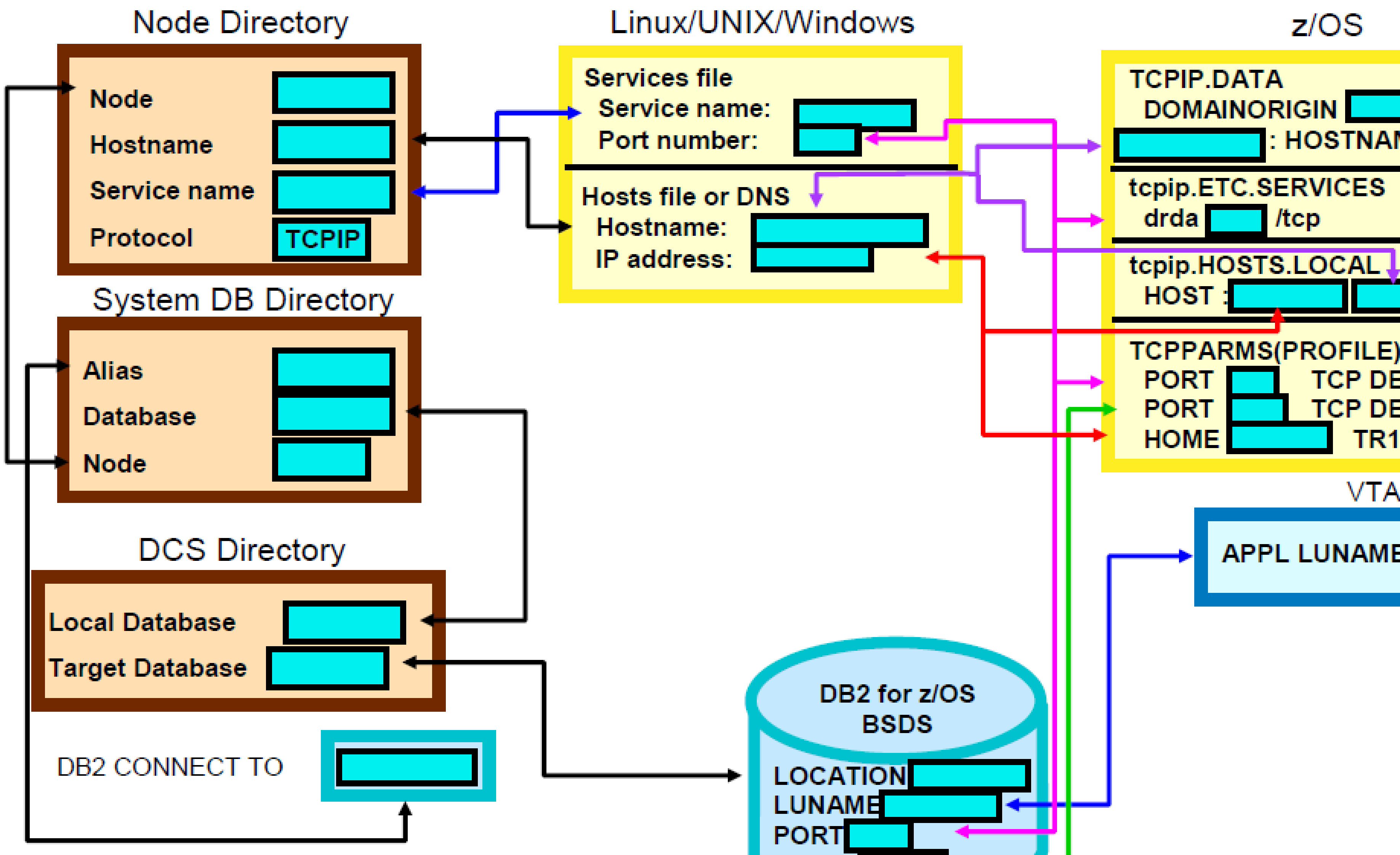
- Problem: To feed up DW (DB2/oracle open) from operational data (DB2 z/os) during batch window



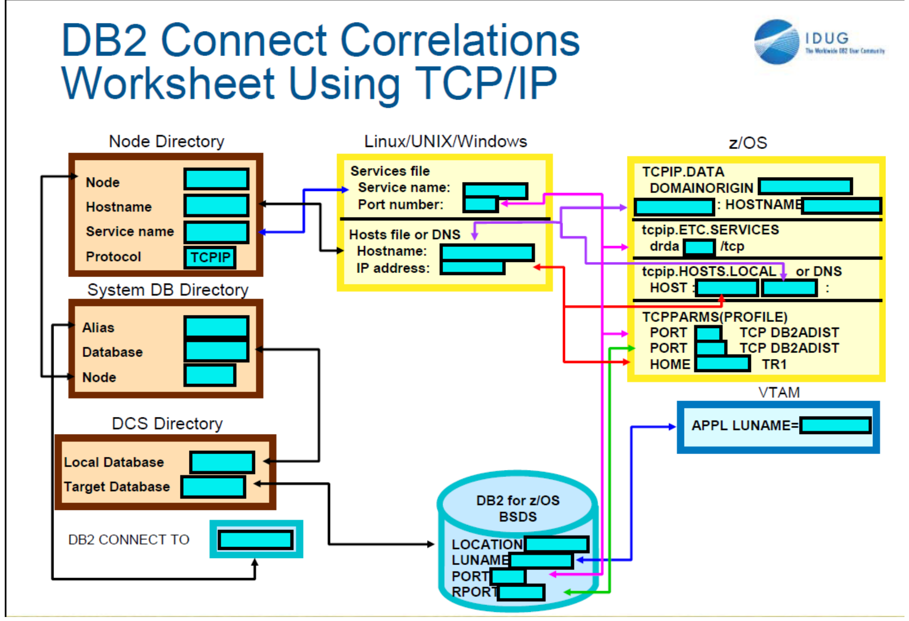
Distributed Database Environment – TCP/IP



DB2 Connect Correlations Worksheet Using TCP/IP

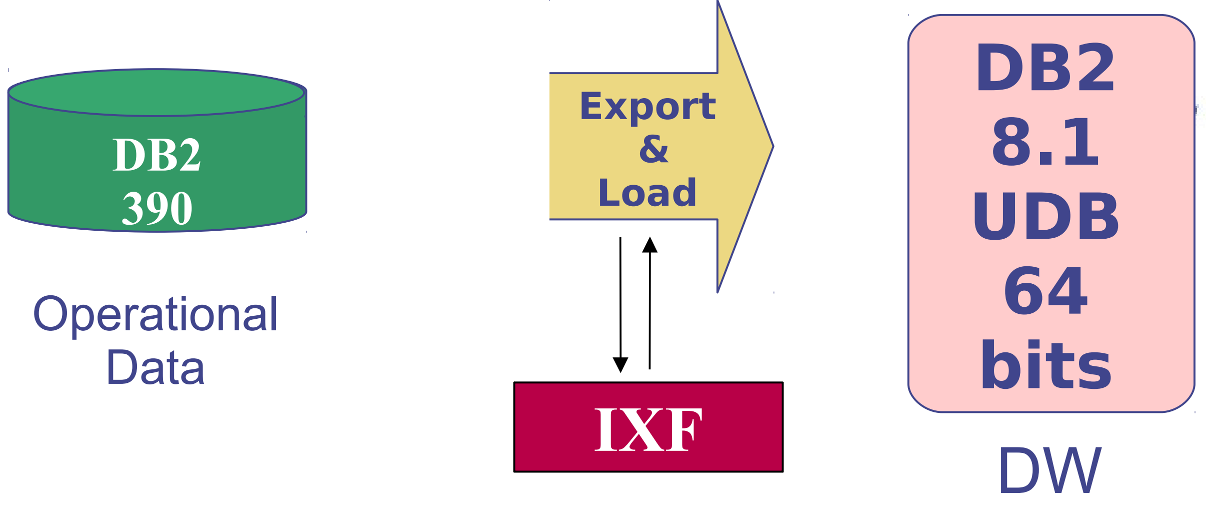


Script



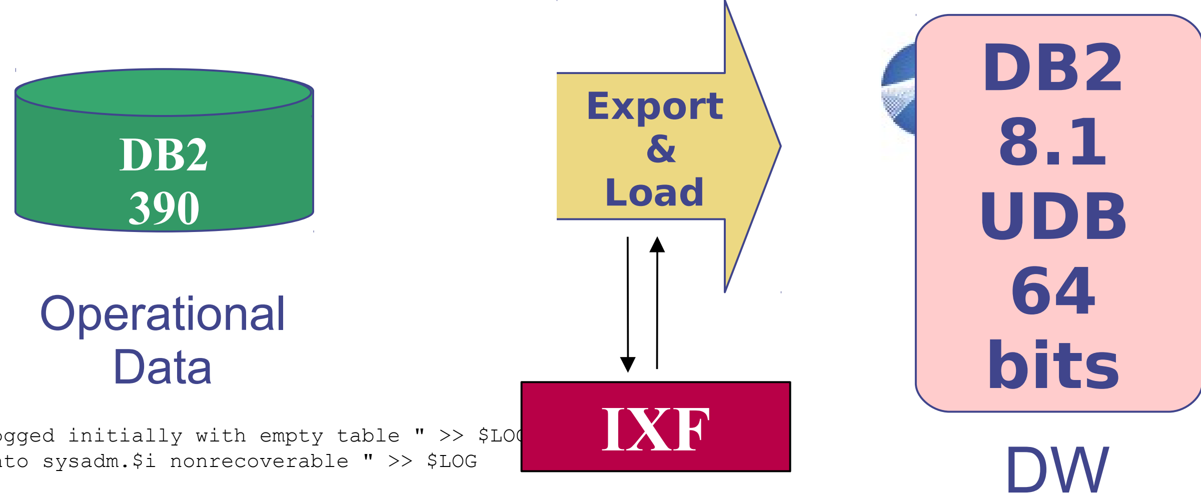
```
#!/bin/bash

NODE=NP390
SERVER=p390.uv.es
PORT=446
ZOSDB="S390LOC"
DBALIAS=P390  # To be changed for each user.

echo "db2 uncatalog node $NODE"
db2 uncatalog node $NODE

echo "db2 catalog tcpip node $NODE remote $SERVER server $PORT ostype OS390"
db2 catalog tcpip node $NODE remote $SERVER server $PORT ostype OS390

echo "db2 uncatalog dcs database $ZOSDB"
db2 uncatalog database dcs $ZOSDB

echo "db2 uncatalog database  $DBALIAS"
db2 uncatalog database  $DBALIAS

echo "db2 catalog dcs database $ZOSDB  as $ZOSDB"
db2 catalog dcs database $ZOSDB  as $ZOSDB

echo "db2 catalog database $ZOSDB  as $DBALIAS at node $NODE authentication dcs"
db2 catalog database $ZOSDB  as $DBALIAS at node $NODE authentication dcs

db2 terminate
```

Script



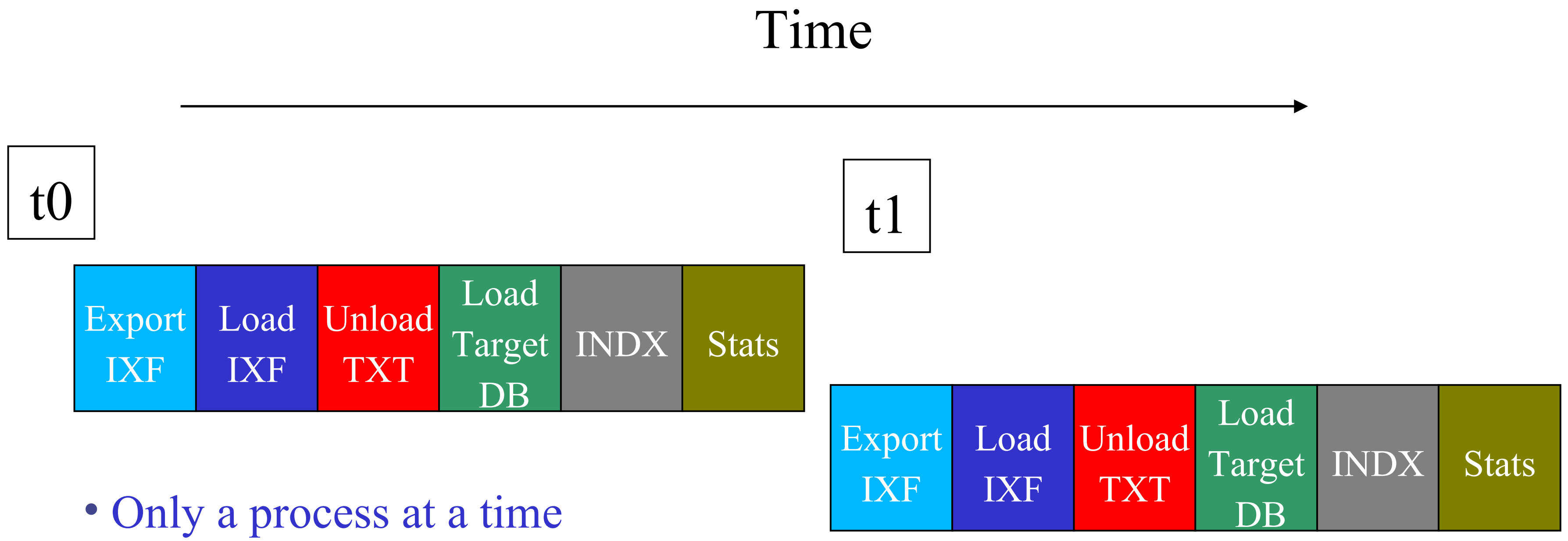
```
#!/bin/bash

replica_db2zos_schema() {

        SCHEMA=DSN8710
        export SCHEMA
        connect $DATABASE $USER $PASSWORD
        init_db2zos_tables
        full_ixf_export
        disconnect
        connect  $REPLICA
        full_ixf_import
        db2_runstats
        disconnect
}

init_db2zos_tables(){

        TABLES_LOG=$LOG/tables.log
         db2 SELECT NAME FROM SYSIBM.SYSTABLES WHERE CREATOR=\'$SCHEMA\' and \(TYPE=\'T\'\) ORDER BY NAME | grep -v  "^NAME" | grep -v "^\-\-\-" > $TABLES_LOG
        TABLES=`cat $TABLES_LOG | grep "^[A-Z].*" `
        export TABLES
        echo $TABLES
}

# Export all tables to ixf
full_ixf_export(){
    for T in $TABLES ; do
       echo "Exporting table: $T to ixf format"
       # db2 code
db2 << EOF > $LOG/$T.ixf_export
export to $FILES/$T.ixf OF IXF MESSAGES $LOG/$T.log select * from $SCHEMA.$T
quit
EOF
    done
}

# Import all tables from ixf
full_ixf_import(){
    for T in $TABLES ; do
       SCHEMA=            # TO BE CHANGED FOR EACH USER
       echo "Importing table: "$T
       # db2 code
       db2 drop table $SCHEMA.$T
       db2 import from $FILES/$T.ixf OF IXF CREATE INTO $SCHEMA.$T

    done
}
```

Performance Issues



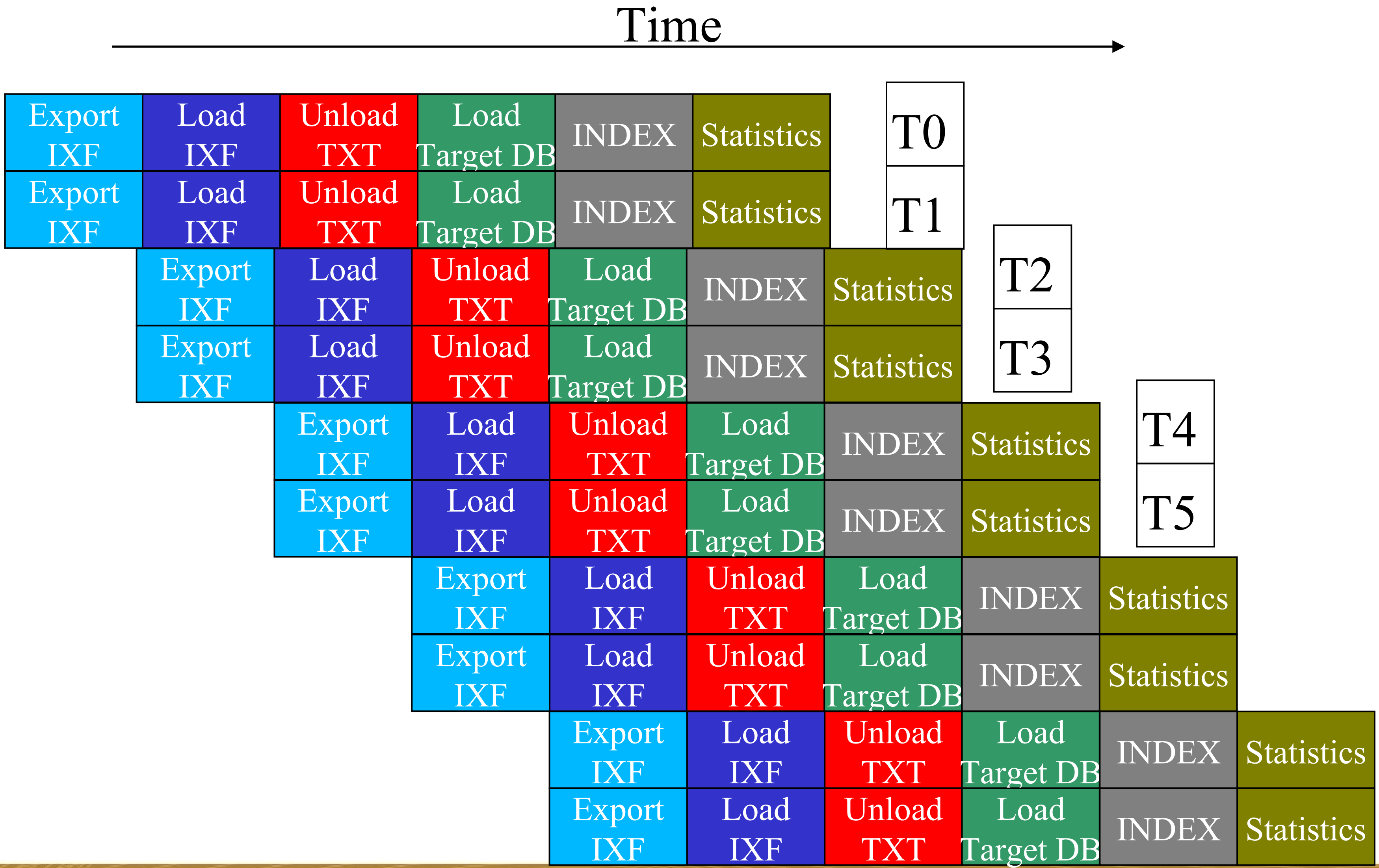
```
#
#       FAST TABLE DROP
#       -------------------
#       echo "db2 alter table sysadm.$i activate not logged initially with empty table " >> $LOG
#       db2 "load from /tmp/dummy.txt of del replace into sysadm.$i nonrecoverable " >> $LOG
#       drop table sysadm.$i
#
#       LOGGING
#       ---------
#       CREATE TABLE XXX ....... IN ${DB2TBS} INDEX IN INDX NOT LOGGED INITIALLY"
#
#       TABLE CREATION
#       --------------
#       Delay table droping
#       echo "Renaming table $T to $AUX. Don't loose time dropping"
#       db2 "rename table $SCHEMA.$T to $AUX " >> $LOG/$T.db2_drop_table.log
#
#       LOCKING
#       --------
#
#       for tables with n. rows < 100000 -> import (No catalog locking )
#       for tables with n. rows > 100000 -> load
#
#       FAST LOAD
#       -------------
#       LOADSQL=load from $FILES/$T.ixf OF IXF INSERT INTO $SCHEMA.$T NONRECOVERABLE CPU_PARALLELISM 2 DISK_PARALLELISM 2 ALLOW READ ACCESS
#       LOADSQL="import from $FILES/$T.ixf OF IXF INSERT INTO $SCHEMA.$T"
#       LOADLOG=$LOG/$T.import.log
#       if [ "${tsize[$s]}" ] && [ ${tsize[$s]} -gt 100000 ] ; then
#                LOADSQL="load from $FILES/$T.ixf OF IXF INSERT INTO $SCHEMA.$T NONRECOVERABLE DATA BUFFER 10000  ALLOW READ ACCESS"
#
#       TABLESPACE AVAILABILITY DURING LOAD
#       ---------------------------------------
#       If a load operation is aborted, it remains at the same access level that was specified when the load operation was issued.
#       So, if a load operation in ALLOW NO ACCESS mode aborts, the table data is inaccessible until a load terminate or a load restart is issued.
#       If a load operation in ALLOW READ ACCESS mode aborts, the pre-loaded table data is still accessible for read access.
```

Sequential Version of ETL



- Only a process at a time
- Only 2 information bundles processed in this period of time
- Nevertheless, each stage only consumes a determinte type of resources:
  - Export IXF -> Net , Load, Unload -> I/O, Index + Statistics -> CPU
  - While the data is downloaded from operational systems (Export IXF), mainly remote CPUs & network bandwidth is consumed
- So, we are wasting resources & time

Applying Pipelining to ETL (III)


Time
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
Export IXF | Load IXF | Unload TXT | Load Target DB | INDEX | Statistics
T0
T1
T2
T3
T4
T5