Techniques For Improved Batch Designs

Session 660

Greg Moll
Texas Instruments

Agenda

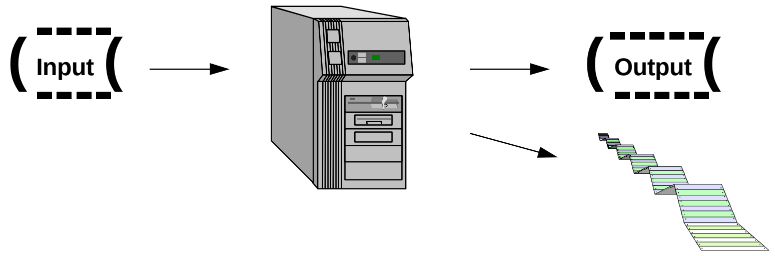
- Define Batch Processing
- Techniques of Batch Processing
- Designing for Parallel Processing
- Alternatives for Input/Output
- Describe Strategies and Infrastructure Components of Batch Processing
  – Logic Errors
  – Checkpoint
  – Restartability

Session Objectives

- Enhance batch processing techniques
- Compare and contrast alternative design methods
- Improve software quality
- Improve developer productivity

What is Batch Processing?

- Volumes of data that are processed with no human intervention
- Can be accomplished by batch or on-line "No display" Procedures



Typical Uses

- Interface
- Conversion
- Service-related processing, e.g., Billing
- Distributed data
- Data retrieval for reports

Design System Structure

- Define procedures (batch or on-line)
- Dialog flows to indicate procedure step flow
- Define input/output
- Define reuse requirements

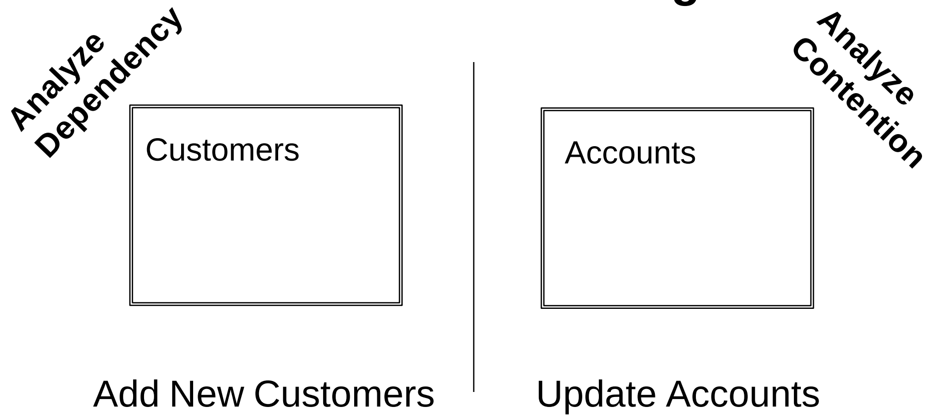
Batch Processing Techniques

- Method determined by procedure definition
- Batch
  - Supported by COBOL only
  - Completely accomplished within Composer
  - Conforms to MVS batch theory
- On-line no display
  - Supported by any language
  - Terminal-dependent/terminal-independent
  - More flexible, e.g., any type of dialog flow

Think about Parallel Designs

- Run multiple units of work simultaneously
- Best opportunity for throughput of work
- Also aids in better usage of development resources
- Two types:
  - Horizontal partitioning
  - Vertical partitioning

Vertical Partitioning



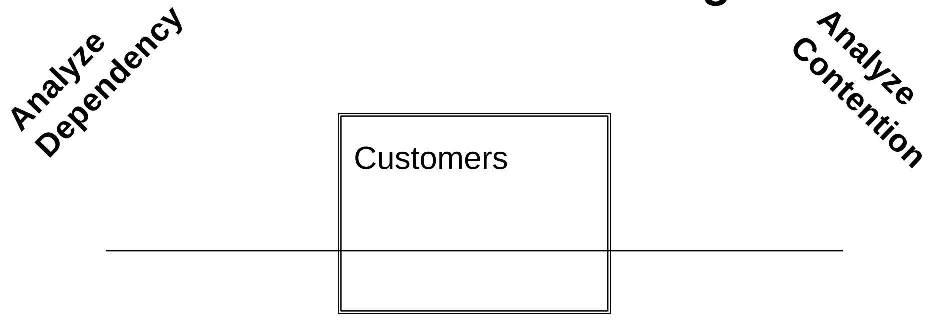
Add New Customers

Update Customers

Delete Customers

Update Accounts

Horizontal Partitioning



1 - 1000000
Customer Updates

1000001 - 2000000
Customer Updates

Alternatives for Input/Output

- Designer-added entity type
  - Load transactions
  - Increased I/O cost (locking, logging, index)
- Data sent through dialog flow
  - Import – 32K limit
  - Export – 32K limit
- Structures available within environment
  - External Action Block
  - DL/I not available within TSO Testing Facility

Advantages/Disadvantages for Input/Output

- Designer-added entity type
  - Ease of usage
  - Increased overhead
- Data sent through dialog flow
  - 32K limit
- Structures available within environment
  - Efficiency
  - Portability
  - Expertise

Reuse Requirements

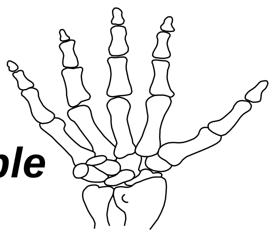
- Batch design strategy and reusable components
  - Shorten development lifecycle
  - Quality
  - Maintainability
- Examples
  - Handling Errors
  - Checkpoint
  - Restart
  - Logging

Skeleton

- Example action diagrams or external programs
- Support the structure “ How”
- Recommend working example

Read RESTART
...

Grab an Example

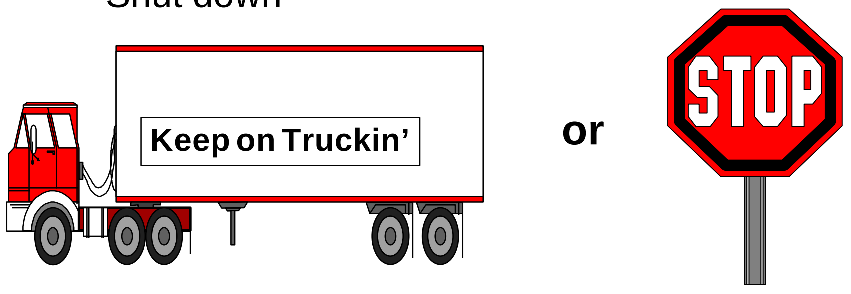


Handle Logic Error – Types

- System
  - Program unable to run
- Runtime
  - Trapped by Composer
  - Under Composer's control
  - TIRTERM
- Logic
  - Under designer's control
  - Level (severity) of error
  - Threshold (count) of error

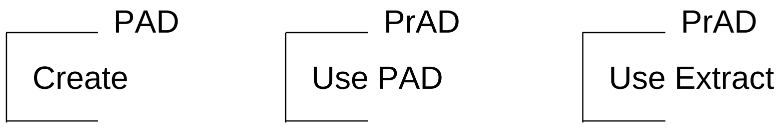
Handle Logic Error

- Zero or one Create Update Delete (CUD) action/process
- Multiple CUD action/Process – Rollback
- Correct with default
- Shut down

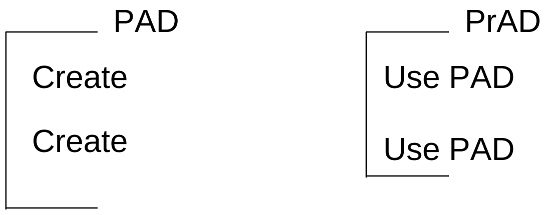


More on Logic Error

- Zero or one CUD action/Process

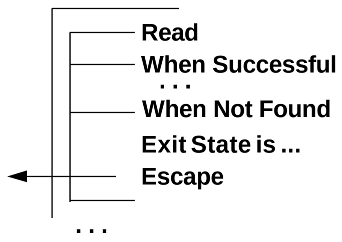


- Multiple CUD action/Process



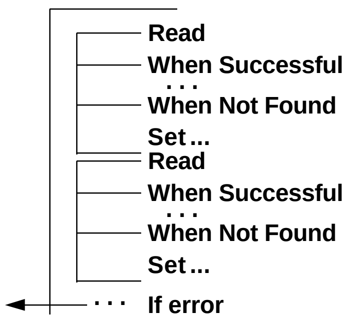
Errors and Action Block Design

- Traditional (Single)
  - On error Escape
  - Single error communicated by Exit State



Errors and Action Block Design

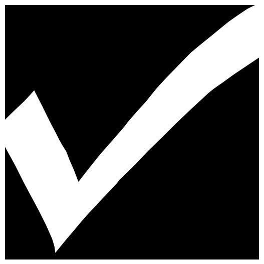
- Multiple
  - On error don't escape,
  - Move CUD to the end
  - Group view of errors
    - » or
  - Duplicate logic
    - » or
  - Validation Action Block
- Error dependency



On Notification of Error ...

- Message to operations
- Beeper
- Mail

Checkpoint



- Protect completed work
- Held locks cost resources
- Unit of work
  - – Count 1000, 5000
  - – Group view size 1 or 2 megs
  - – Time interval
- Threshold the commit
- Use a designer entity type to control

Checkpoint Options

- Self-reference flow
  - – Step must retain position
  - – Commit issued at the end of procedure step
  - – Database cursors are closed
  - – Batch – TIRMSGF trace
- Commit by EAB

Comparison

	Pros	Cons
Self-Referencing	–Portable –Accomplish within Composer –TIRMSGF Trace (Batch)	–Design limitation possible with the 32K view limit –Performance –Extra Logic
EAB	–Flexible designs can achieve performance gains	–Less Portable –Management of code –TIRMSGF Trace (Batch)

Checkpoint Entity Type

- Job name
- Job step name
- Checkpoint amount
- Time interval

Position Commit EAB (without “Cursor withhold”)

- EAB must not be placed within a Read Each

```
While . . .
    Read Each . . .
        . . .
Use Commit EAB
```

Position Commit EAB (with “Cursor withhold”)

- Maintain cursor position on commit
- Significant performance improvement
- Commit EAB must be positioned within READ EACH bracket

```
Read Each . . .
    Use Commit EAB
    . . .
```

Restartability

- Begin job from prior point
- Standard JES2 or JES3 in the job card
  - Self-referencing flow
  - Checkpoint information in TIRIOVF
  - JCL RESTART = parameter “Step Name”
  - Batch-defined procedures only
- Designer Entity Type
- 3rd-party product
- Recover tables and re-run job

Batch – TIRIOVF Data Sent

- May be used for checkpoint repositioning, error logic, restartability
- Trace – USERID.IEF.TIRIOVF
- JCL – temporary
- Default JCL always allocates
- DSORG PS, RECFORMAT FB, LRECL 4096
- CLEANIOF or NO Flow

Restart Entity Type

- Job name
- Job step name
- “N” text restart keys
- “N” numeric keys
- “N” date/time keys
- “N” restart control totals
  - – Records read, written, error...

Make the Move

- Define the strategies
- Build the reusable components
- Gain the productivity

Techniques For Improved Batch Designs

Session 660

Greg Moll
Texas Instruments