

Efficient Development of Event-Driven Systems with Versant Object Database

Dr. Günter Ressel-Herbert

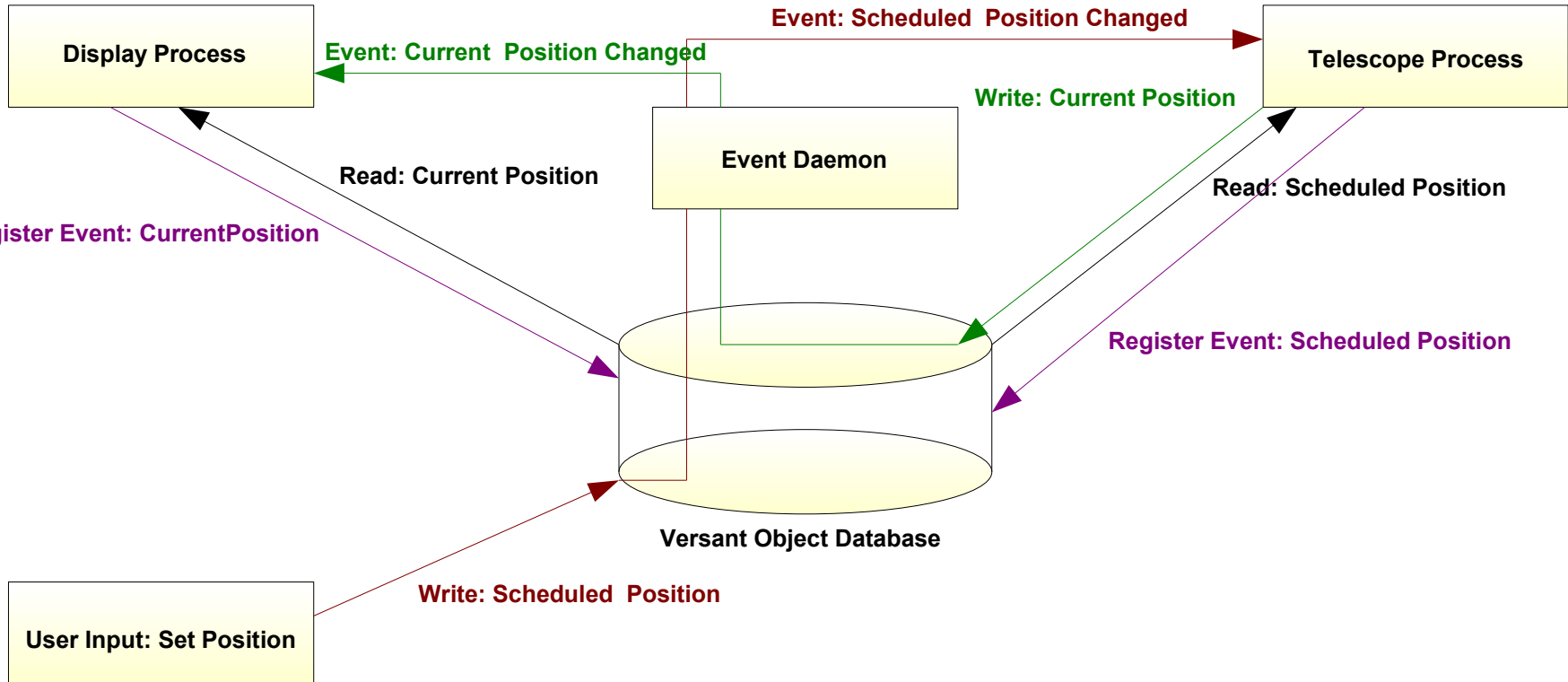
Copyright © Versant Corp. All rights reserved.



Agenda

- Introduction
- The Plan: Write a Telescope Control Program
- User Interface
- From the System Perspective:
 - Event Registration
 - Raising an Event
 - Event Processing
 - Event Message Delivery
- Examples

Demo Program Structure



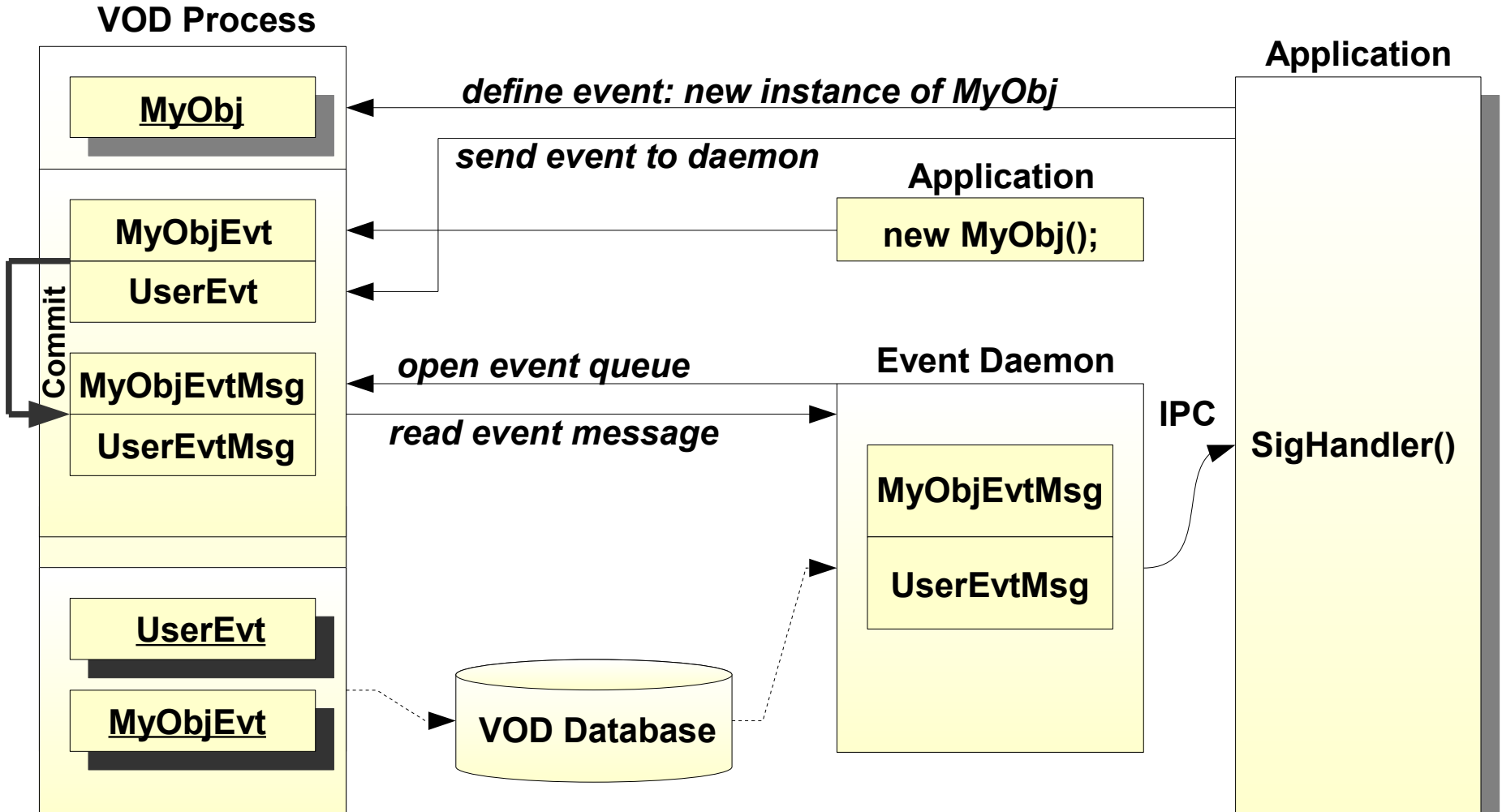


Introduction

Design Rationale and Requirements

- Provide customers with a rich and flexible framework to build powerfull object monitoring applications
- Goes beyond SQL triggers
- Minimum footprint and performance impact
- Event information to survive database restarts
- Maintain performance statistics

Architecture





User Interface

Initialization

- Identify caller as owner of event delivery daemon
- Activate event notification
- Specify event delivery behavior in case of resource shortage



User Interface (cont.)

Event Definition

- Objects to be monitored: can be either class objects or instance objects
- Range of events
 - Object: delete, modify
 - Class instance: create, modify, delete
- Predicate:
 - condition evaluated when event is raised
 - define when the predicate is evaluated (immediate or commit)
- Optional: Definer Data
 - to be used by event delivery daemon



User Interface (cont.)

Event Processing

- Open event queue
 - Start event message daemon
- Read event message
 - Provide maximum size buffer
 - Read data
 - Evaluate optional definer supplied information
 - Evaluate optional raiser supplied information



User Interface (cont.)

Controlling event notification framework

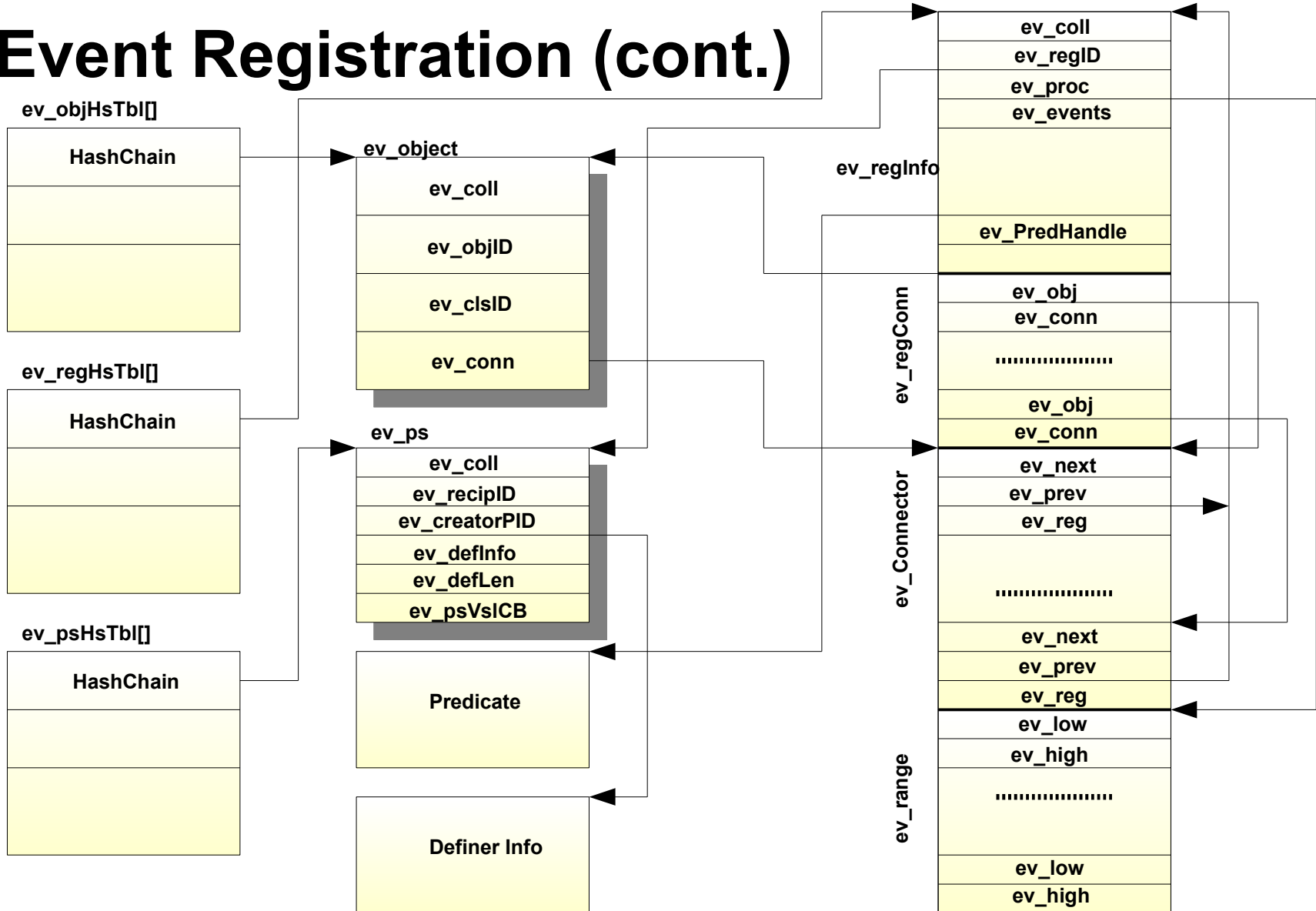
- Send message directly to event daemon
 - Provide event raiser information
 - Provide event definer information
 - No registration required



Event Registration

- Validate user arguments
- Create event registration data structure
 - Write down event range(s), predicate and definer information
 - Hook up with object(s) being monitored (and vice versa)
 - Hook up with registrations monitoring the same object
 - Hook up with event recipient data structure
 - Insert into event registration hash table

Event Registration (cont.)

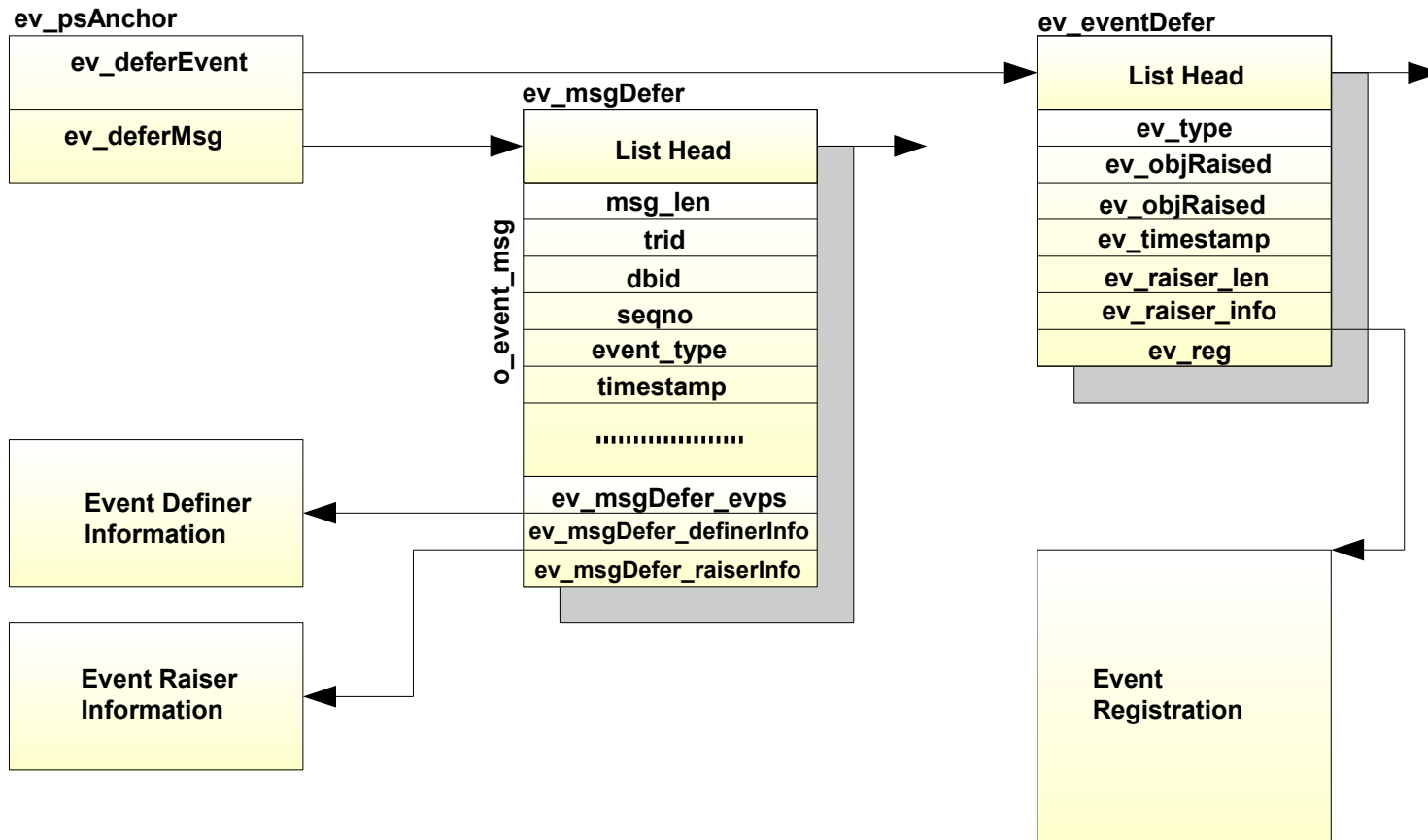




Raising an Event

- Event processing has hooks in VOD kernel to intercept changes of classes and instances of classes
- Current transaction provides context (list heads etc.)
- Check if object is registered for event notification
- If so, build deferred event data structure; enqueue to deferred event list head
- If !EV_EVAL_AT_COMMIT,
 - evaluate event right now against conditions given in registration
 - Create deferred event message data structure; enqueue to deferred event message list head

Raising An Event (cont.)

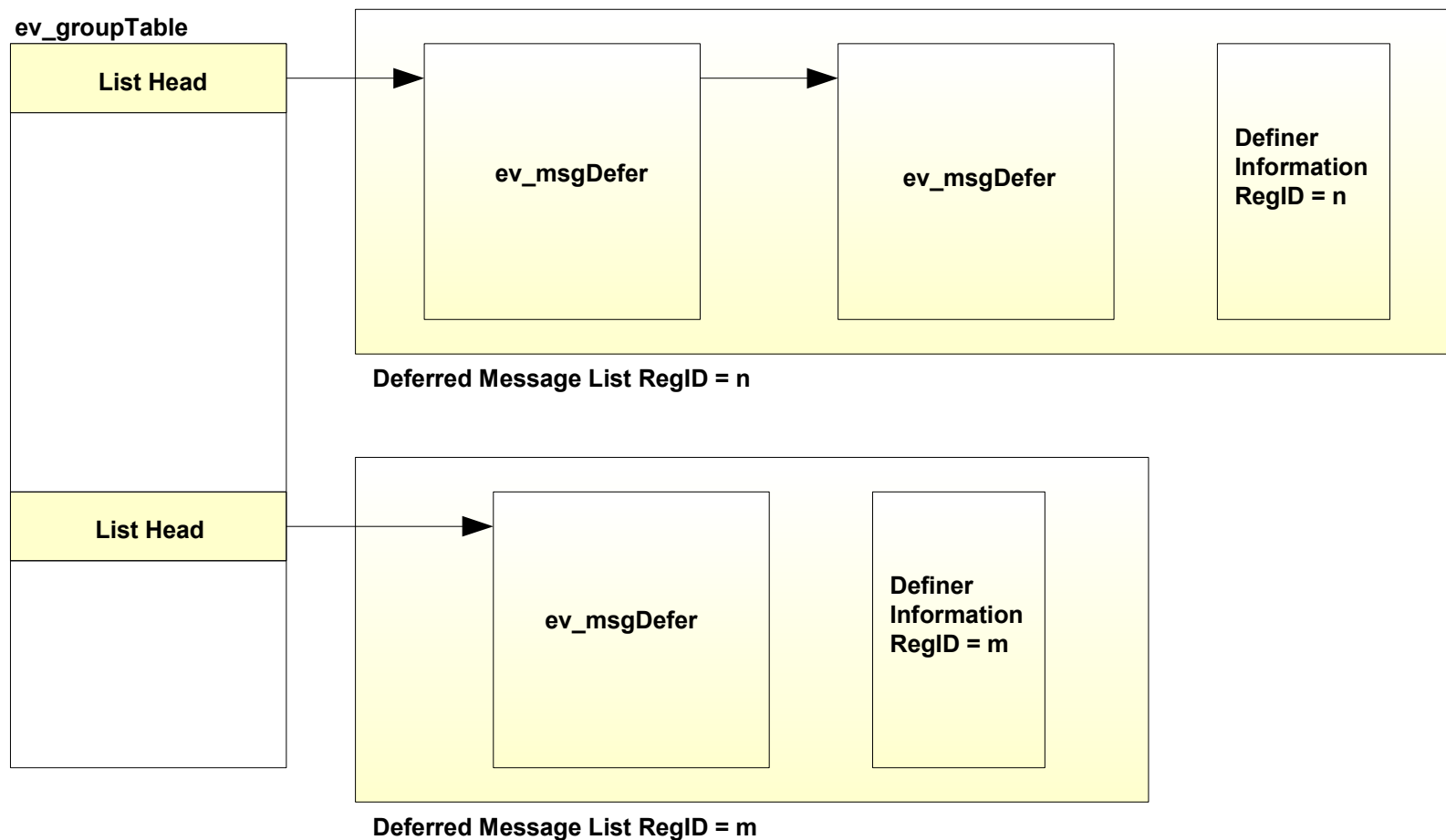




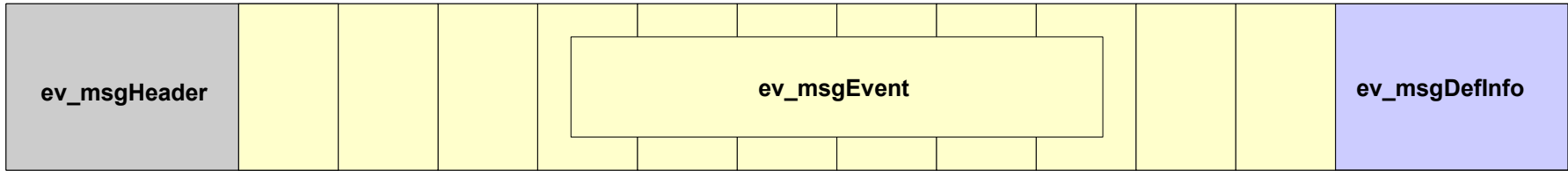
Event Processing

- Takes place when current transaction commits
- If `EV_EVAL_AT_COMMIT`,
 - evaluate event right now against conditions given in registration
 - Create deferred event message data structure; enqueue to deferred event message list head
- Sort deferred event messages according to registration
 - Group table provides per registration list head for event messages; they share the same definer information
- Combine all lists into one event message
- Enqueue event message to message queue

Event Message Processing (cont.)



Anatomy of An Event Message



msg_len
msg_id
msg_version
msg_flags
msg_trid
msg_db
msg_numevent

len
type
objRaised
objDefined
objTimestamp
raiserInfoLen
raiserInfoOffset
raiserInfo

regID
DefInfoLen
definerInfo

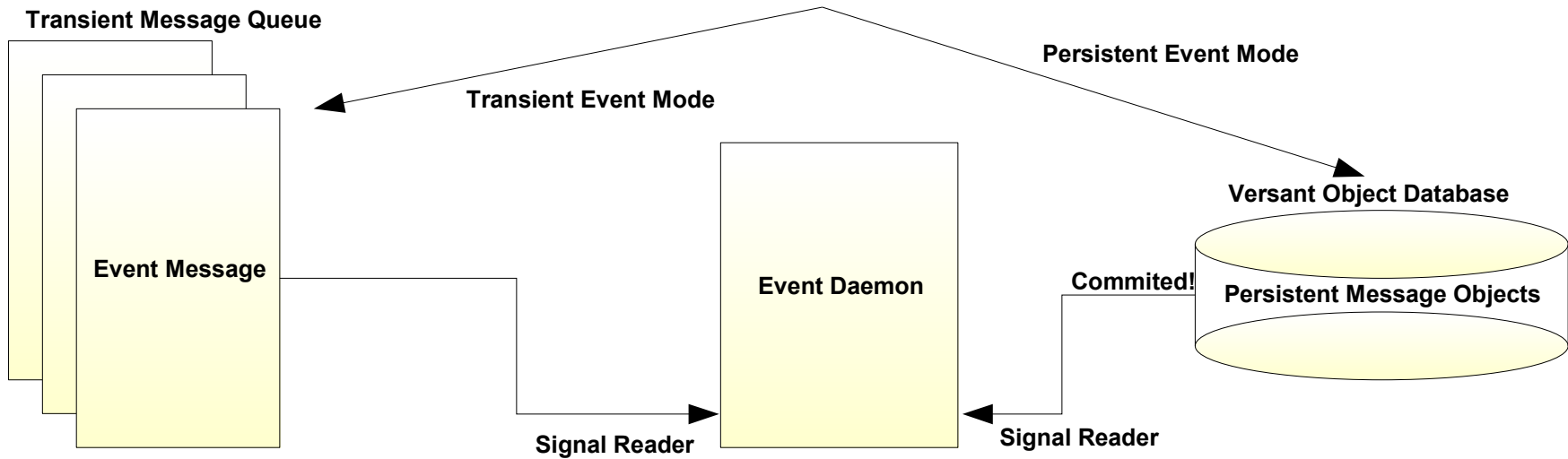
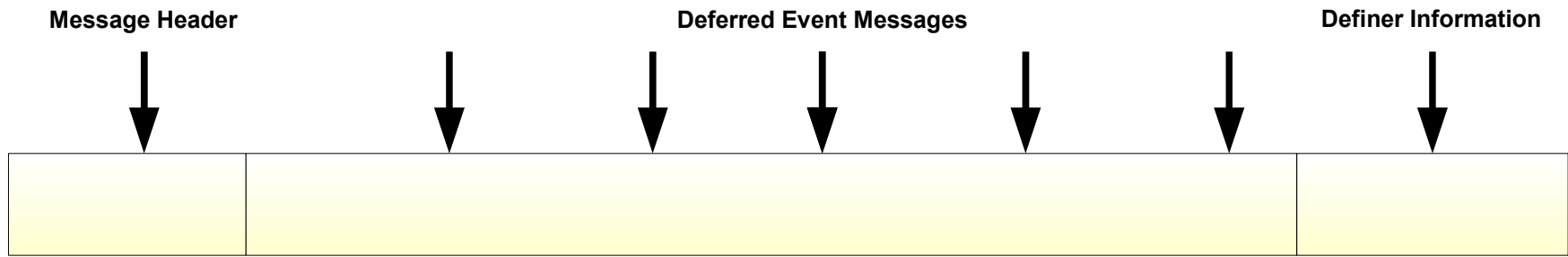


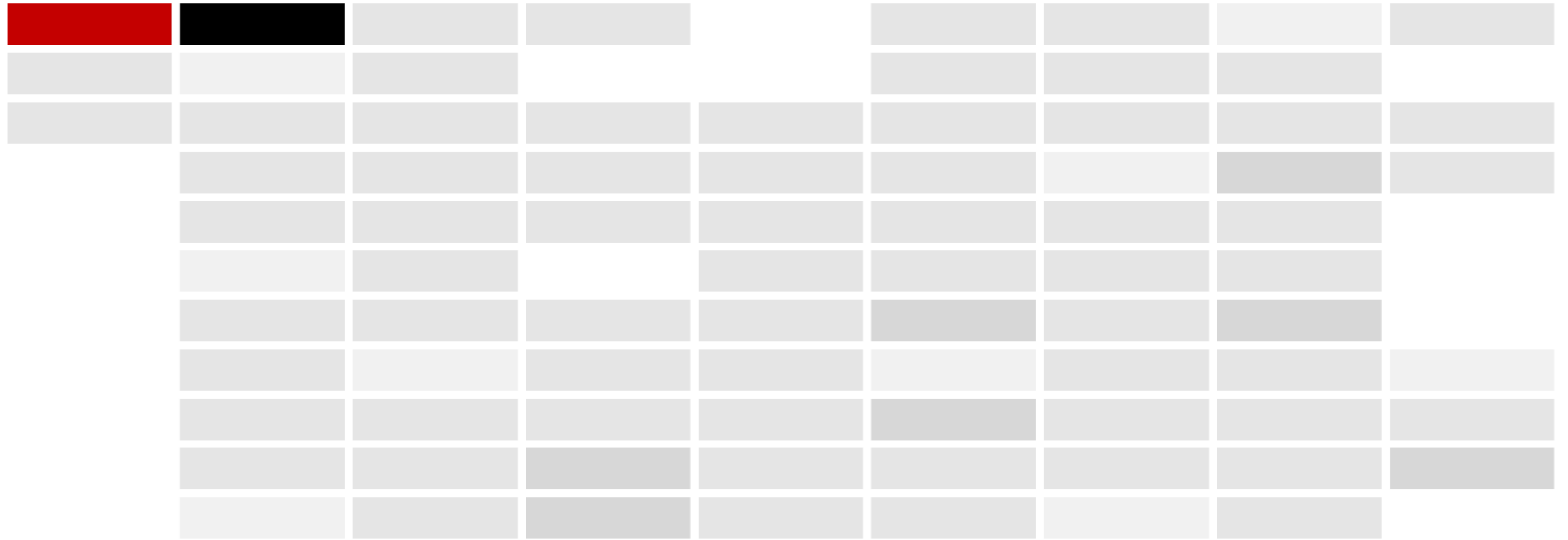
Event Message Delivery

- Two types of message queues:
 - Transient message queue for memory resident events
 - Persistent message queue for event *objects*
- Transient events:
 - Check size of message queue against upper limit to avoid memory exhaustion
- Persistent events:
 - Transform events into event objects and commit to disk
 - You can really watch them with db2tty etc...

Event Message Delivery (cont.)

Transaction Committed





THINK OUTSIDE THE GRID.
NON-SQUARE DATA MANAGEMENT.