Modeling Workflows

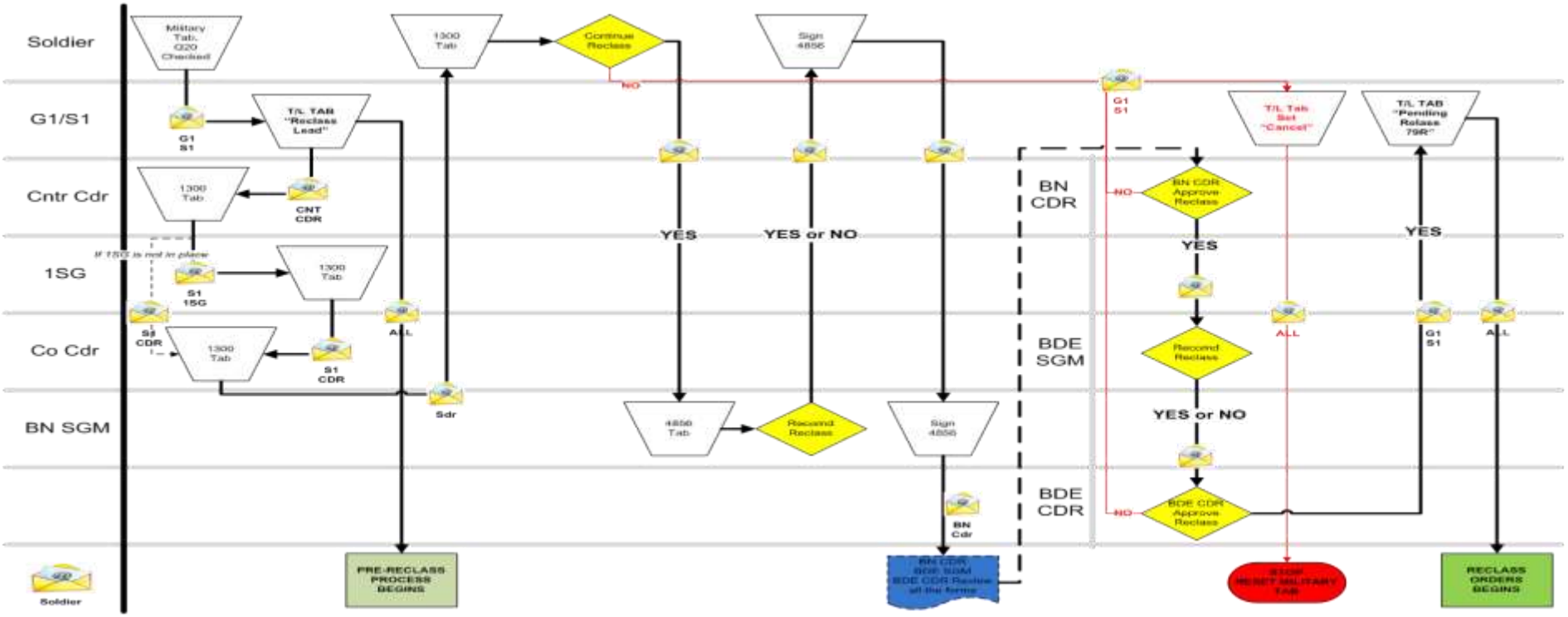
Using Event Driven Deterministic Finite Automata (DFA)

Speaker Introduction

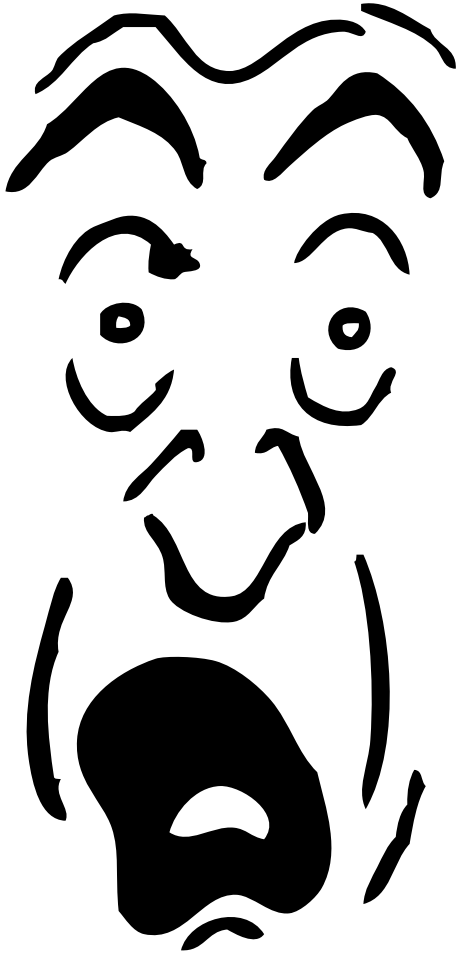
- Richard Roda's linked in profile: http://www.linkedin.com/in/richardroda
- Over 15 years of IT experience.
- Sr. Software Engineer for Hewlett Packard for the Army at Ft. Knox
- Headquarters Support Structure application Technical Lead.
- Certifications: Security+, ITILv3 Foundation
- BA Business with minor in Computer Science from Warren Wilson College

Motivation

- Customer wants over 25 long running complex workflows like this one implemented.

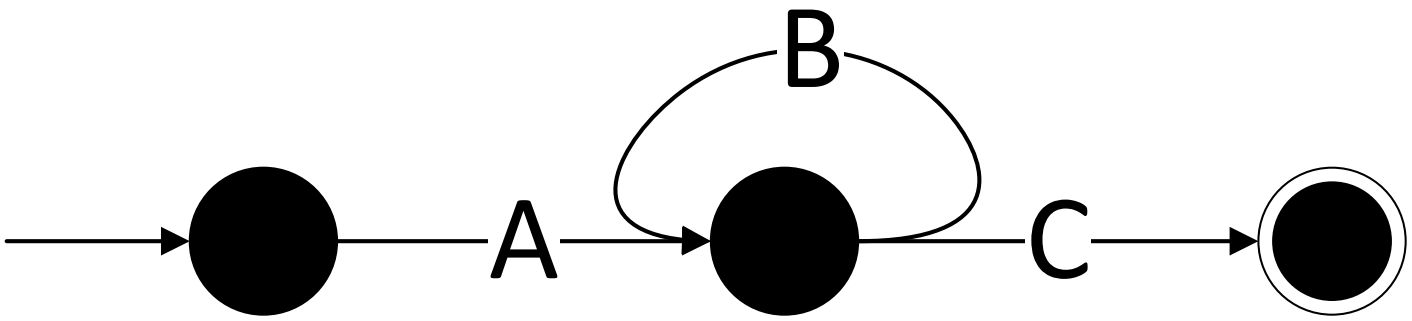


My Initial Reaction



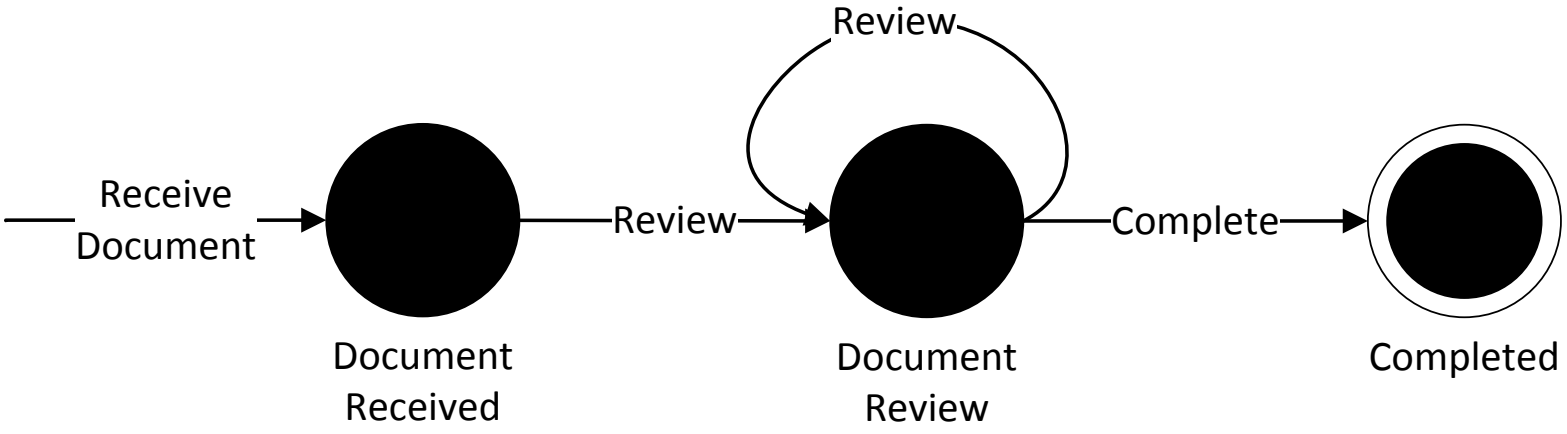
Finite State Machine (FSM)

- FSM is a mathematical construct that processes symbols using states and vertices
- A Deterministic Finite Automaton (DFA) is a FSM where any symbol has 0 or 1 vertices.
- Example: Classical DFA that accepts AB*C (A followed by 0 or more Bs followed by C).



Event Driven Deterministic Finite Automaton (DFA)

- Uses events instead of symbols.
- Commonly used in embedded systems and networking to track state.
- Example DFA that models a review process that requires one or more reviewers.



Why Store and Manipulate the DFA in a Database?

- Design advantages.
  - Platform and language agnostic.
  - Implicitly keeps historical data
  - Handles long running workflows with many event sources.
- Runtime Advantages (Scalability)
  - Number of states does not make program larger.
  - Direct database operation avoids network overhead.

Design Advantage: Platform and Language Agnostic

- As long as the client can connect and use the database, it will work.
- In my current project, we have both our Java web project and DataStage ETL (Extract, Transform and Load) tool manipulating the DFA through its stored procedure interface.
- Other sources could be added, such as web services or ESB (Enterprise Service Bus) calls.

Design Advantage: Historical Persistence

- DFA collects and make available historical data
- Enables analytics reports to identify slow points and pain points in workflows.
- Workflow may be long running between events (not session bound). Days, weeks, months or years may pass between events.
- Events may be sent to the workflow from many sources.

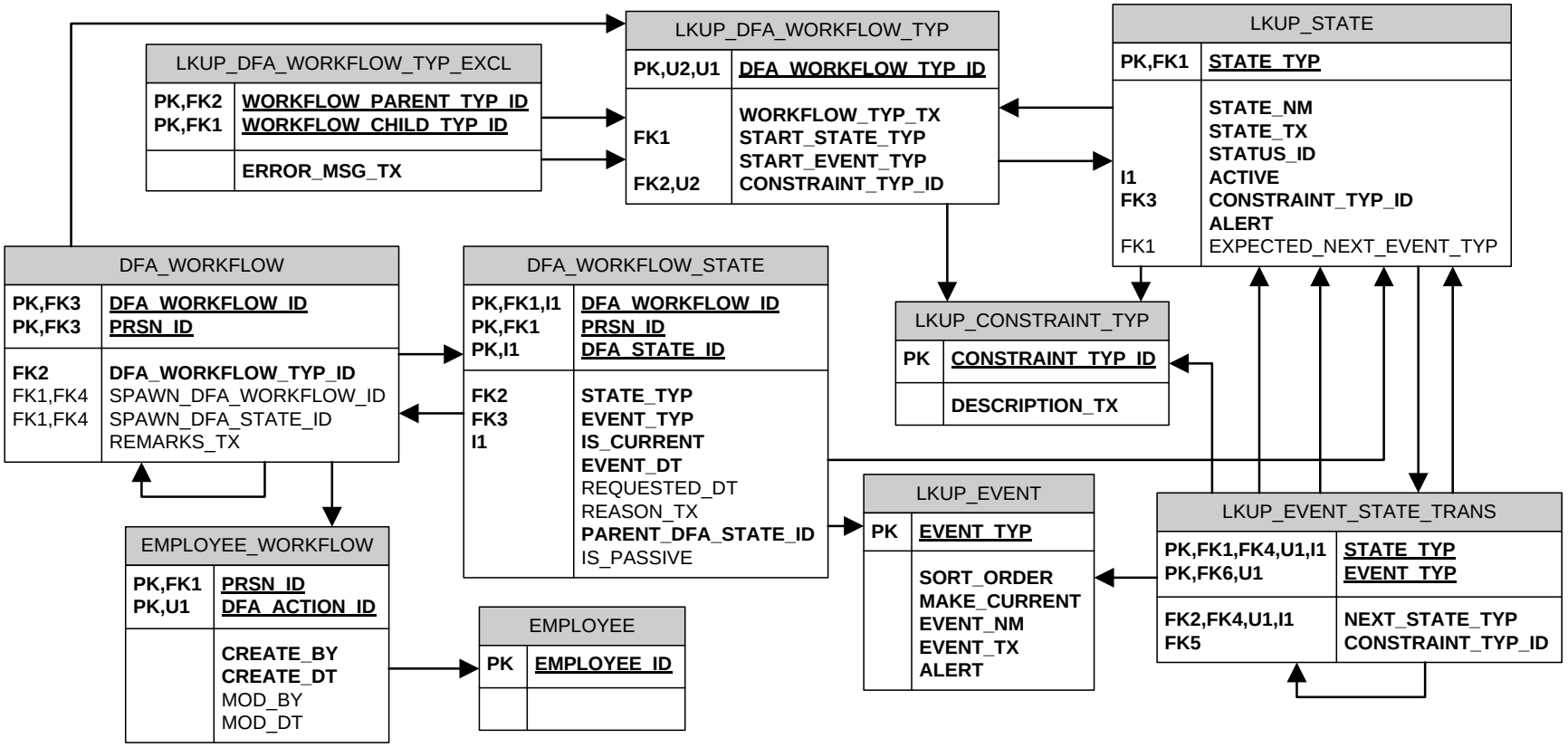
Database DFA Scalability

- A large number of states and transitions may be manipulated by a relatively small program. Adding states and transitions does not increase program size.
- An interactive program only has to load information about the current state.
- An event source only needs to send its event.
- A stored procedure processes events sent to it directly on the database without network overhead.
- For a given set of states and transitions, the next state for a given event is computed in constant O(1) time.

Theoretical Graph Model

- All of the DFA states exist in a graph of states with vertices labelled with events.
- Each DFA state may connect to zero or more states (including itself).
- A state uses an event no more than once.
- A workflow defines a start state.
- Closure: the state, its events, vertices and neighbors, and the neighbors closure.
- Workflow is defined by its start state's closure.

Database DFA Schema



LKUP_EVENT

- The elements in this table define the events.
- Defines the default display name (or resource key) and if the event is passive.
- When an event is passive, the resulting state does not become the current state.
- Passive events are useful for features such as leaving comments.

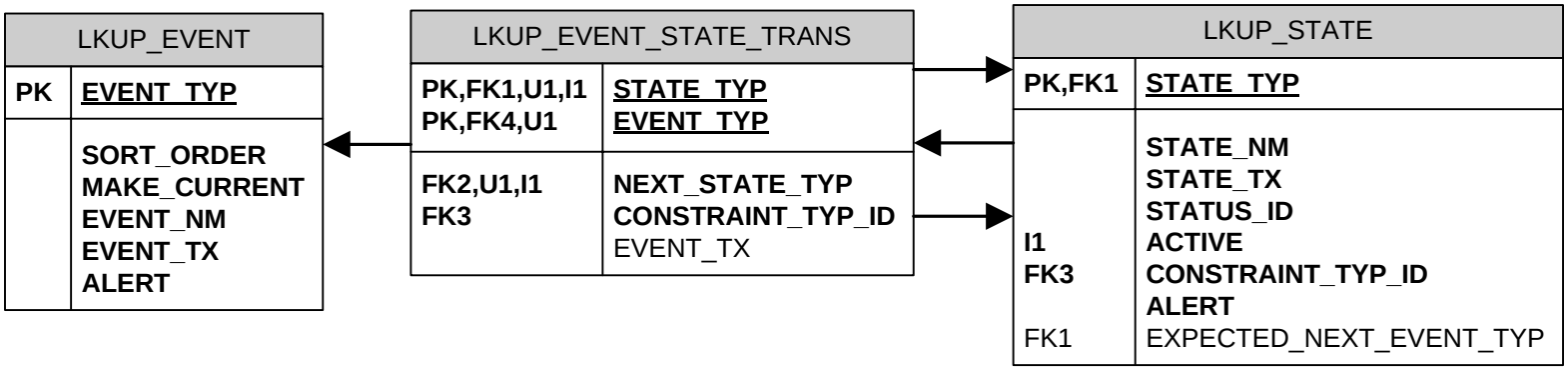
Event Sources

- Direct User Input to apply an event (action) to a specific workflow. Example: Manager approves pay raise.
- Application updates. Example: Employee salary update results in a SALARY INCREASE event.
- Services (SOA). Example: A request to a notification system may be sent due to a salary change resulting in a EMPLOYEE NOTIFY event.
- Extract, Transform and Load (ETL) system.
- Another workflow, especially a parent workflow.

LKUP_STATE

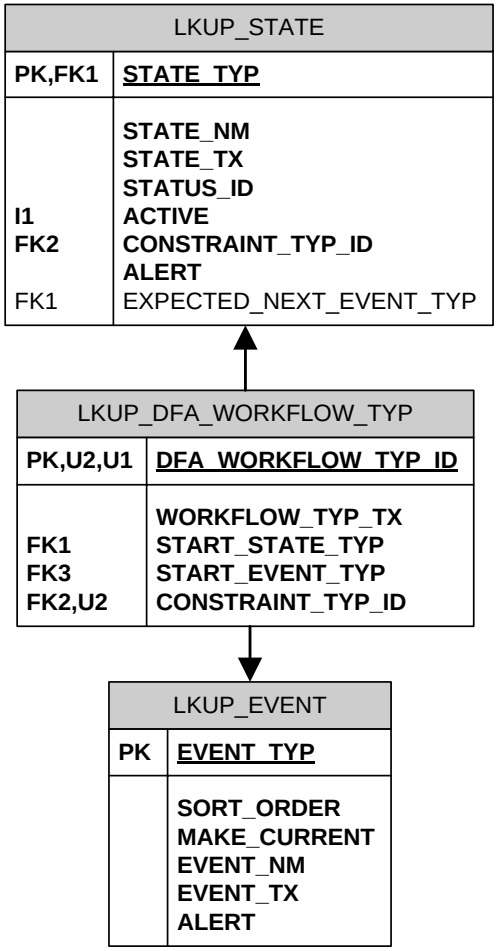
- Defines the DFA states.
- Defines display name (or key).
- May define an expected event.
- Has constraints that must be satisfied to enter.
- May define an alternate state that is tried if constraints are unsatisfied.
- Constraints + alternate states are a way to implement conditional branching in the DFA.

LKUP_STATE_EVENT_TRANS



- Defines the transitions or vertices between the states.
- Is also the foreign key to the expected next event type defined in LKUP_STATE.
- The expected next event corresponds to the “normal” path of a use case.

LKUP_WORKFLOW_TYP



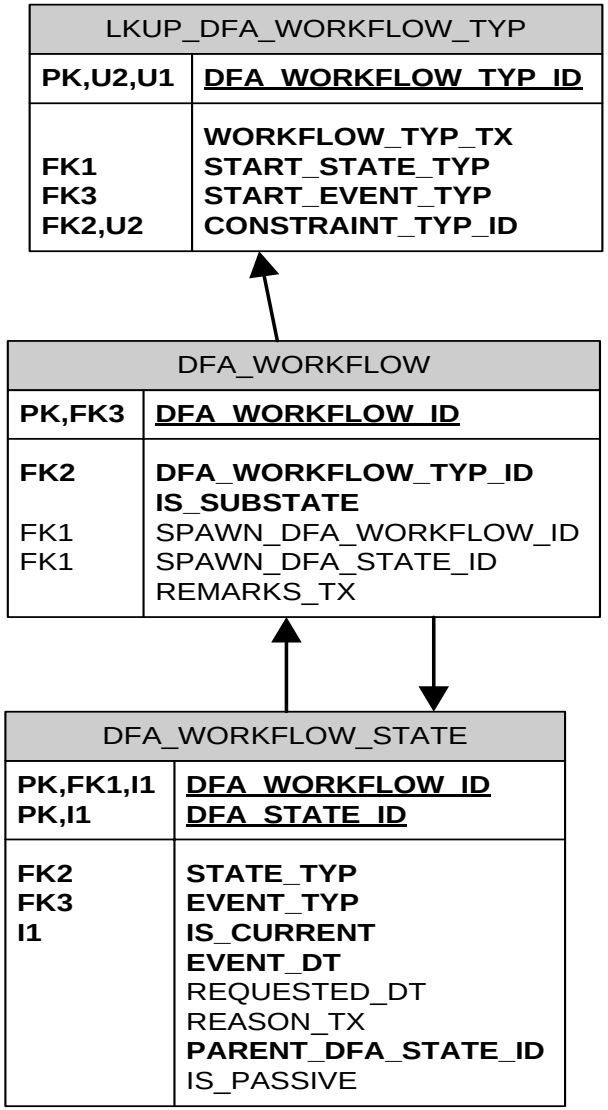
- Defines the start state within the DFA graph.
- Multiple workflows may share the same state.
- The entire workflow is the closure of the DFA graph from the START_EVENT_TYP.

Examples of Workflows That Share States

- Workflows for various employee awards at the same level may share the same workflow for review and approval.
- There could be workflows for both approving a new salary and hiring a new employee. The hiring a new employee could use (transition into) the approving a new salary workflow.

DFA_WORKFLOW

- Has its type (name) defined by LKUP_DFA_WORKFLOW_TYP.
- Has 1..N states, with exactly 1 current state. Start state is defined by LKUP_DFA_WORKFLOW_TYP.
- Can be spawned by a parent workflow state. Spawned workflows are only valid while their spawned state is current.

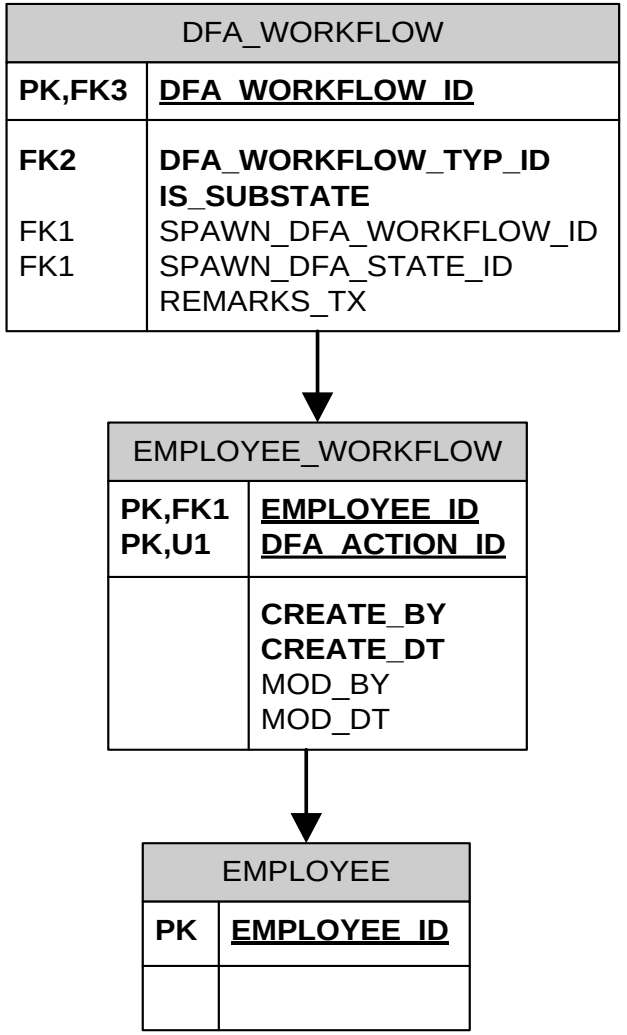


DFA_WORKFLOW_STATE

- Defines workflow states, including the current workflow state.
- Each workflow has 1 current workflow state.
- May spawn DFA sub-workflows. This is used to model parallel paths in workflows.
- Has comments that apply to the particular state.

Entity Binding Table

- Exists outside of the dfa database (schema).
- Associates 1 or more entities to DFA_WORKFLOW.
- For the EMPLOYEE table, the entity binding table is EMPLOYEE_WORKFLOW.
- Allows only 1 employee per DFA action.



Temporary Table DFA_WORKFLOW_ENTITY

- Before invocation of DFA stored procedure, this table is populated with all DFA_WORKFLOW ids associated with the entity.
- After invocation, this table includes any new states and workflows that were created as a result of the DFA operation.

Entity Procedures

1. Check any additional constraints that apply to the entity for the DFA call.
2. Populate DFA_WORKFLOW_ENTITY with all of the WORKFLOW IDs that are bound (associated) to the entity.
3. Execute the DFA stored proc.
4. Check any additional constraints against the result (held in DFA_WORKFLOW_ENTITY).
5. Bind any new DFA_WORKFLOW_IDs found in DFA_WORKFLOW_ENTITY to the entity binding table.

Employee Demo

Demonstration of DFA used to track workflow of hiring a new employee.

Is DFA Workflow the Right Choice?

- Sometimes when you are given a hammer, everything looks like a nail, but sometimes this is not the best choice when…
- An objective does not have a well defined order (which means it isn't actually a workflow…)
- A workflow is data centric as opposed to process centric. Computed/generated columns coupled with triggers work well for this case.

Questions

(and Hopefully Answers)