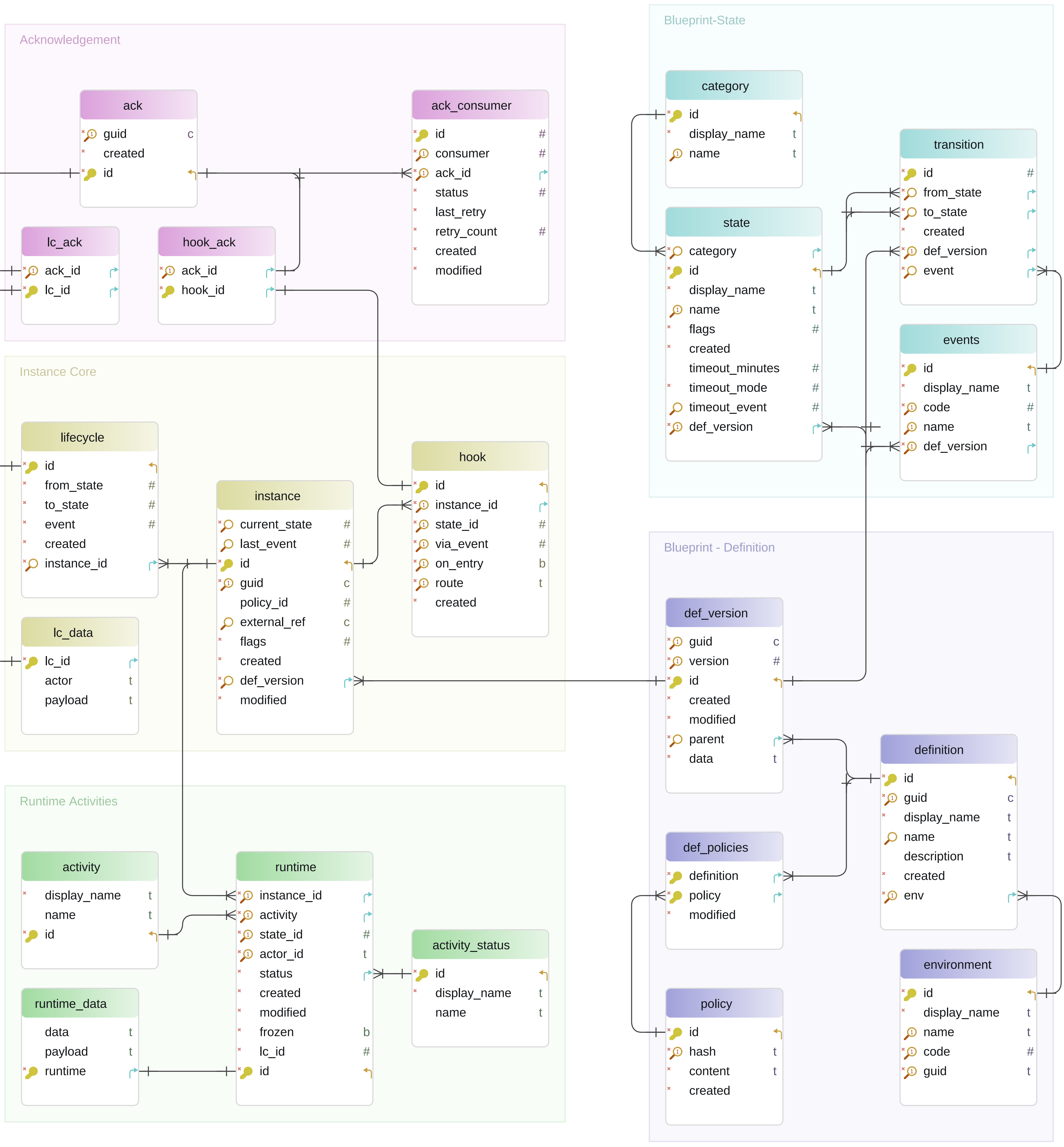
Acknowledgement
ack
guid c
created
id
ack_consumer
id #
consumer #
ack_id
status #
last_retry
retry_count #
created
modified
lc_ack
ack_id
lc_id
hook_ack
ack_id
hook_id
Instance Core
lifecycle
id
from_state #
to_state #
event #
created
instance_id
lc_data
lc_id
actor t
payload t
instance
current_state #
last_event #
id
guid c
policy_id #
external_ref c
flags #
created
def_version
modified
hook
id
instance_id
state_id #
via_event #
on_entry b
route t
created
Runtime Activities
activity
display_name t
name t
id
runtime_data
data t
payload t
runtime
runtime
instance_id
activity
state_id #
actor_id t
status
created
modified
frozen b
lc_id #
id
activity_status
id
display_name t
name t
Blueprint-State
category
id
display_name t
name t
state
category
id
display_name t
name t
flags #
created
timeout_minutes #
timeout_mode #
timeout_event #
def_version
transition
id #
from_state
to_state
created
def_version
event
events
id
display_name t
code #
name t
def_version
Blueprint - Definition
def_version
guid c
version #
id
created
modified
parent
data t
definition
id
guid c
display_name t
name t
description t
created
env
def_policies
definition
policy
modified
policy
id
hash t
content t
created
environment
id
display_name t
name t
code #
guid t

Default Diagram

Table ack

Idx	Name	Data Type
* Unq	guid	CHAR(36) DEFAULT uuid()
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP
* Pk	id	BIGINT AUTO_INCREMENT

Indexes

Type	Name	On
Pk	pk_ack	id
Unq	unq_ack	guid

Options

engine=InnoDB

Table ack_consumer

Idx	Name	Data Type	Description
* Pk	id	BIGINT AUTO_INCREMENT	
* Unq	consumer	BIGINT DEFAULT 0	
* Unq	ack_id	BIGINT	

*	status	INT DEFAULT 1	Flag:
			1 = Pending //We have sent to the application.. We dont' know what happened.
			2 = Delivered // Application has received and may have probably stored it in a database.
			3 = Processed (Idempotent) //Action has been taken by the client on the received information.
			4 = Failed

*	last_retry	DATETIME DEFAULT CURRENT_TIMESTAMP	
*	retry_count	INT DEFAULT 0	
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	
*	modified	DATETIME ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP	

Indexes

Type	Name	On	Description
Pk	pk_ack_log	id	
Unq	unq_ack_consumer	ack_id, consumer	

Foreign Keys

Type	Name	On	Description
	fk_ack_consumer_ack (ack_id) ref ack (id)		

Options

engine=InnoDB

Table activity

Idx	Name	Data Type	Description
These are minor applicatoin managed activies which the statemachine doens't have any awareness about.. like, send_email, firstreview, escalatedreview, finalcheck, etc..			
*	display_name	VARCHAR(140)	

Table activity

	name	VARCHAR(140) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT	
* Pk	id	INT AUTO_INCREMENT	

Indexes

Type	Name	On	Description
Pk	pk_runtime_activity	id	

Options

engine=InnoDB

Table activity_status

Idx	Name	Data Type	Description

These are all execution or activity status, which WorkFlow engine has no visibility about. Like, 'Pending''Completed'"approved','Rejected'"Returned'.. Reason is, we dont know what kind of state each runtime activity might follow.. For instance, one of the runtime activity can have 'Approved','Rejected' state.. another can only have 'Sent'"Pendin' (like, email delivery)

Idx	Name	Data Type	Description
* Pk	id	INT AUTO_INCREMENT	
*	display_name	VARCHAR(120)	
	name	VARCHAR(120) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT	

Indexes

Type	Name	On	Description
Pk	pk_ext_state	id	

Options

engine=InnoDB

Table category

Idx	Name	Data Type
* Pk	id	INT AUTO_INCREMENT
*	display_name	VARCHAR(120)
Unq	name	VARCHAR(120) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT

Indexes

Type	Name	On
Pk	pk_tbl	id
Unq	unq_category	name

Options

engine=InnoDB

Table def_policies

Idx	Name	Data Type
* Pk	definition	INT
* Pk	policy	INT
*	modified	DATETIME DEFAULT CURRENT_TIMESTAMP

Indexes

Type	Name	On
Pk	pk_definition_policy	definition, policy

Foreign Keys

Table def_policies

Type	Name	On
	fk_def_policies_definition (definition) ref definition (id)	
	fk_def_policies_policy (policy) ref policy (id)	

Options

engine=InnoDB

Table def_version

Idx	Name	Data Type
* Unq	guid	CHAR(36) DEFAULT uuid()
* Unq	version	INT DEFAULT 1
* Pk	id	INT AUTO_INCREMENT
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP
*	modified	DATETIME ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP
* Unq	parent	INT
*	data	LONGTEXT

Indexes

Type	Name	On
Unq	unq_def_version	parent, version
	fk_def_version_definition	parent
Unq	unq_def_version_0	guid
Pk	pk_def_version	id

Foreign Keys

Type	Name	On
	fk_def_version_definition (parent) ref definition (id)	

Constraints

	Name	Definition
	cns_def_version	`version` > 0
	cns_def_version_0	json_valid(`data`)

Options

AUTO_INCREMENT 1990

Table definition

Idx	Name	Data Type	Description
* Pk	id	INT AUTO_INCREMENT	it should be a code provided by the user.
* Unq	guid	CHAR(36) DEFAULT uuid()	
*	display_name	VARCHAR(200)	
Unq	name	VARCHAR(200) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT	
	description	TEXT	
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	
* Unq	env	INT DEFAULT 0	

Indexes

Type	Name	On	Description
Pk	pk_definition	id	

Table definition

Unq	unq_definition	env, name	
Unq	unq_definition_0	guid	
	idx_definition	name	

Foreign Keys

Type	Name	On	Description
	fk_definition_environment (env) ref environment (id)		

Options

AUTO_INCREMENT 1998

Table environment

Idx	Name	Data Type	Description

environment code
//Doesnt' need to be like dev/prod/test.. It can be an work-group environmetn as well..

//like preq-app (is one environment), so all preq-app (wherever it runs, local, production etc) will be able to read definitions.

//we can even extend it as , preq-app-dev, preq-app-prod etc.

Idx	Name	Data Type	Description
* Pk	id	INT AUTO_INCREMENT	
*	display_name	VARCHAR(120)	
Unq	name	VARCHAR(120) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT	
* Unq	code	INT	
* Unq	guid	VARCHAR(42) DEFAULT uuid()	

Indexes

Type	Name	On	Description
Pk	pk_environment	id	
Unq	unq_environment	code	
Unq	unq_environment_0	name	
Unq	unq_environment_1	guid	

Options

engine=InnoDB

Table events

Idx	Name	Data Type
* Pk	id	INT AUTO_INCREMENT
*	display_name	VARCHAR(120)
* Unq	code	INT
Unq	name	VARCHAR(120) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT
* Unq	def_version	INT

Indexes

Type	Name	On
Pk	pk_events	id
Unq	unq_events	def_version, code, name
Unq	unq_events_0	def_version, code

Foreign Keys

Type	Name	On
	fk_events_def_version (def_version) ref def_version (id)	

Table events

Options

engine=InnoDB

Table hook

Idx	Name	Data Type	Description
hooks are raised based on policy.. we just check the policy and then raise these hooks			
* Pk	id	BIGINT AUTO_INCREMENT	
* Unq	instance_id	BIGINT	
* Unq	state_id	INT	
* Unq	via_event	INT	
* Unq	on_entry	BIT DEFAULT 1	by default, the hooks are for entry.. we can also, setup on leave. 0 - on leaving 1 - on entry
* Unq	route	VARCHAR(180)	event or the route name that needs to be triggered or hooked.
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	

Indexes

Type	Name	On	Description
Pk	pk_hooks	id	
Unq	unq_hooks	instance_id, state_id, via_event, on_entry, route	

Foreign Keys

Type	Name	On	Description
	fk_hooks_instance (instance_id) ref instance (id)		

Options

engine=InnoDB

Table hook_ack

Idx	Name	Data Type
* Unq	ack_id	BIGINT
* Pk	hook_id	BIGINT

Indexes

Type	Name	On
Pk	pk_hook_ack	hook_id
Unq	unq_hook_ack	ack_id

Foreign Keys

Type	Name	On
	fk_hook_ack_hook (hook_id) ref hook (id)	
	fk_hook_ack_ack (ack_id) ref ack (id)	

Options

engine=InnoDB

Table instance

Idx	Name	Data Type	Description
* Idx	current_state	INT	
Idx	last_event	INT	
* Pk	id	BIGINT AUTO_INCREMENT	
* Unq	guid	CHAR(36) DEFAULT uuid()	

Table instance

	policy_id	INT DEFAULT 0	
Unq	external_ref	CHAR(36)	like external workflow id or submission id or transmittal id.. Expected value is a GUID

*	flags	INT UNSIGNED DEFAULT 0	active =1,
			suspended =2 ,
			completed = 4,
			failed = 8,
			archive = 16

*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	
* Unq	def_version	INT	
*	modified	DATETIME ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP	

Indexes

Type	Name	On	Description
Pk	pk_instance	id	
Unq	unq_instance	guid	
	fk_instance_state	current_state	
	fk_instance_events	last_event	
	fk_instance_def_version	def_version	
Unq	unq_instance_0	def_version, external_ref	
	idx_instance	external_ref	

Foreign Keys

Type	Name	On	Description
	fk_instance_def_version (def_version) ref def_version (id)		

Options

engine=InnoDB

Table lc_ack

Idx	Name	Data Type
* Unq	ack_id	BIGINT
* Pk	lc_id	BIGINT

Indexes

Type	Name	On
Pk	pk_lc_ack	lc_id
Unq	idx_lc_ack	ack_id

Foreign Keys

Type	Name	On
	fk_lc_ack_lifecycle (lc_id) ref lifecycle (id)	
	fk_lc_ack_ack (ack_id) ref ack (id)	

Options

engine=InnoDB

Table lc_data

Idx	Name	Data Type	Description
* Pk	lc_id	BIGINT	
	actor	VARCHAR(60)	
	payload	LONGTEXT	Could be any data that was the result of this transition (which could be later used as a reference or input for other items)

Indexes

Table lc_data

Type	Name	On	Description
Pk	pk_transition_data	lc_id	

Foreign Keys

Type	Name	On	Description
	fk_transition_data_transition_log (lc_id) ref lifecycle (id)		

Options

engine=InnoDB

Table lifecycle

Idx	Name	Data Type	Description
Contains the major states which are controlled by the statemachine			
* Pk	id	BIGINT AUTO_INCREMENT	
*	from_state	INT	
*	to_state	INT	
*	event	INT	
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	
* Idx	instance_id	BIGINT	

Indexes

Type	Name	On	Description
Pk	pk_transition_log	id	
	fk_transition_log_instance	instance_id	

Foreign Keys

Type	Name	On	Description
	fk_transition_log_instance (instance_id) ref instance (id)		

Options

engine=InnoDB

Table policy

Idx	Name	Data Type	Description
* Pk	id	INT AUTO_INCREMENT	
* Unq	hash	VARCHAR(48)	hash of the policy contents (states, attach modes, routes)
*	content	TEXT	supposedly the policy json
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	

Indexes

Type	Name	On	Description
Pk	pk_policy	id	
Unq	unq_policy	hash	

Options

engine=InnoDB

Table runtime

Idx	Name	Data Type	Description
Remember, runtime state (or activity track) doesn't need an acknowledgement, because, this infomration itself is managed in application side and we are only receiving it directly from the app itself.. But, there might be some events that we need to raise for each state based on the policy.. those has to be properly acknowledge.d			
* Unq	instance_id	BIGINT	
* Unq	activity	INT	

Table runtime

* Unq	state_id	INT	
* Unq	actor_id	VARCHAR(60) DEFAULT 0	
*	status	INT	
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	
*	modified	DATETIME DEFAULT CURRENT_TIMESTAMP	
*	frozen	BIT DEFAULT 0	For instance, this specific item may have some status, but we might freeze it, meaning, it cannot change status anymore.. unless we unlock the frreeze state..
*	lc_id	BIGINT DEFAULT 0	
* Pk	id	BIGINT AUTO_INCREMENT	

Indexes

Type	Name	On	Description
Pk	pk_micro_log	id	
Unq	unq_execution	instance_id, state_id, activity, actor_id	

Foreign Keys

Type	Name	On	Description
	fk_execution_runtime_state (status) ref activity_status (id)		
	fk_runtime_instance (instance_id) ref instance (id)		
	fk_runtime_activity (activity) ref activity (id)		

Options

engine=InnoDB

Table runtime_data

Idx	Name	Data Type	Description
	data	LONGTEXT	data that needs to be displayed.. For instance, I can send in a json value, which can then be displayed in the UI with property name/value pair.. So that parsing during display can be reduced ..
	payload	LONGTEXT	Some data associate with this transition.. may or may not be present, which can then be reused or used for idempotency.
* Pk	runtime	BIGINT	

Indexes

Type	Name	On	Description
Pk	pk_runtime_data	runtime	

Foreign Keys

Type	Name	On	Description
	fk_runtime_data_runtime (runtime) ref runtime (id)		

Options

engine=InnoDB

Table state

Idx	Name	Data Type	Description
* Idx	category	INT DEFAULT 0	
* Pk	id	INT AUTO_INCREMENT	
*	display_name	VARCHAR(200)	
Unq	name	VARCHAR(200) GENERATED ALWAYS AS (lower(trim(display_name))) PERSISTENT	

Table state

*	flags	INT UNSIGNED DEFAULT 0	none = 0
			is_initial = 1
			is_final = 2
			is_system = 4
			is_error = 8

*	created	DATETIME DEFAULT CURRENT_TIMESTAMP	
	timeout_minutes	INT	in minutes
*	timeout_mode	INT DEFAULT 0	0 = Once 1 = Repeat
Idx	timeout_event	INT	
* Unq	def_version	INT	

Indexes

Type	Name	On	Description
Pk	pk_state	id	
Unq	unq_state	def_version, name	
	fk_state_category	category	
	fk_state_events	timeout_event	

Foreign Keys

Type	Name	On	Description
	fk_state_def_version (def_version) ref def_version (id)		
	fk_state_category (category) ref category (id)		

Options

engine=InnoDB
 AUTO_INCREMENT 2014

Table transition

Idx	Name	Data Type
* Pk	id	INT AUTO_INCREMENT
* Unq	from_state	INT
* Unq	to_state	INT
*	created	DATETIME DEFAULT CURRENT_TIMESTAMP
* Unq	def_version	INT
* Unq	event	INT

Indexes

Type	Name	On
Pk	pk_transition	id
Unq	unq_transition	def_version, from_state, to_state, event
	fk_transition_state	from_state
	fk_transition_state_0	to_state
	fk_transition_events	event

Foreign Keys

Type	Name	On
	fk_transition_state (from_state) ref state (id)	
	fk_transition_state_0 (to_state) ref state (id)	
	fk_transition_def_version (def_version) ref def_version (id)	
	fk_transition_events (event) ref events (id)	

Options

engine=InnoDB