

# Data Quality Rules: Rules for State-Dependent Objects

## INTRODUCTION

The objects that go through a sequence of states in the course of their life cycle as a result of various events are referred to as *state-dependent objects*. For instance, the life cycle of a job application is a progression from its submission, through pre-screening, to applicant interviews, to possible job offers, and up to an eventual hiring or rejection decision. Employees, insurance claims, and product orders are all examples of state-dependent objects.

State-dependent objects are usually the most important database citizens, and their data are most error-prone. Analyzing quality of the data for state-dependent objects requires a specific set of techniques.

## STATE-TRANSITION MODELS

Consider a career of a loyal Bad Data Corporation employee, Jane Gooding. She was originally hired in 1959 at the age of 26. During the next 19 years, she remained with the company with the exception of two maternity leaves in 1964 and 1967, respectively. In 1978 she quit her job but returned four years later. After 13 more years of hard work, Jane finally retired at the age of 62.

Jane's employment history can be described as a sequence of chronological events, each bringing about a change in her employment status. For instance, a new hire event on 3/13/1959 starts Jane's career as an active employee (status *Active*). Jane's resignation on 5/15/1978 changed her employment status from *Active* to *Terminated*.

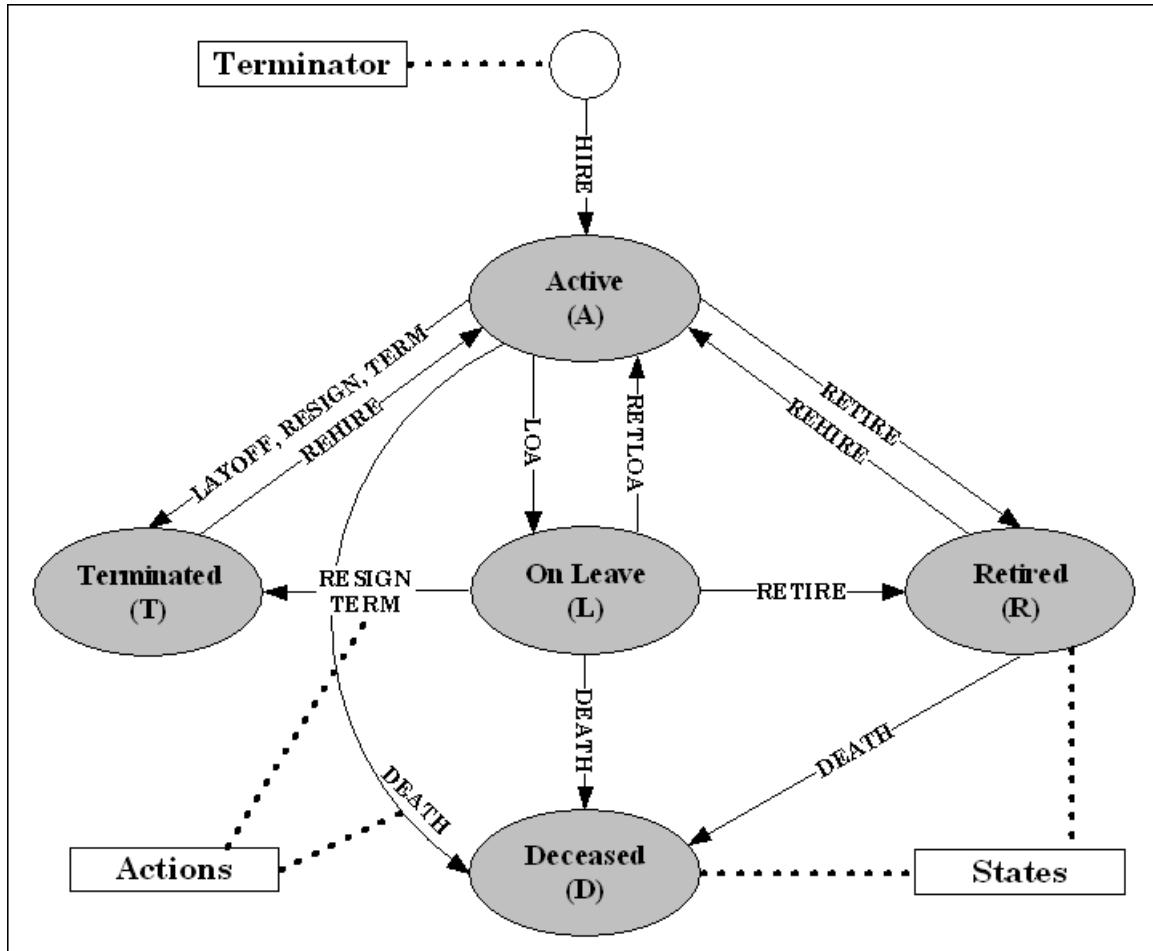
Different employees have different careers. Some take one or more leaves of absence, while others do not. Some are terminated and later rehired; others resign and never come back. Thus, their careers consist of a varying number, sequence, and duration of events and employment statuses. However, not just any combination is allowed. For instance, no employee can be terminated twice in a row without being rehired in between.

In order to design data quality rules for state-dependent objects, we need a model distinguishing valid from invalid life cycles. *State-transition models* achieve this objective and describe constraints on the life cycle of state-dependent objects through two key concepts: state and action.

- **State** is a unique set of circumstances in which an object may exist. At any point in its life, the object must be in one and only one state. The states that identify possible beginning and ending points of the object's life cycle are called *terminators*.
- **Action** is a unique event that results in a change of state. An action may have conditions that must be satisfied before it can take place (action pre-conditions) or after it is completed successfully (action post-conditions).

The diagram below depicts a simple state-transition model for the object EMPLOYEE. The five shaded ovals represent valid object states: *Active* (A),

*Terminated (T), On Leave (L), Retired (R), and Deceased (D).* At any point in time, each object must be in exactly one of these states. Further, each object must begin the life cycle in terminator state *Active* as illustrated by the white circle on the top of the diagram. A more comprehensive state-transition diagram would also depict pre-conditions and post-conditions for various actions.



## PROFILING STATE-TRANSITION MODELS

State-transition models are a source of a wealth of important data quality rules. Unfortunately, these models are often unavailable or unreliable. The problem can be easily mitigated, as state-transition models can be reverse-engineered quite easily from the data and available metadata through analysis and profiling.

Understanding the nature of the object from the business perspective is a good starting point. A business user will quickly tell you what states the object can take, which of the states are valid terminators, what actions can apply to the object in each state, and what is the impact of each action on the object state. This information can be used to build a preliminary state-transition diagram.

However, data does not always follow common sense. As is the case with most other types of models and metadata, the only way to get them right is to use data profiling.

**State-transition model profiling** is a collection of techniques for analysis of the lifecycle of state-dependent objects that provides actual information about the order and duration of states and actions. Combining the results of the data profiling with the information obtained from business users will yield the correct state-transition model.

State-transition model profiling is generally more complex than regular attribute profiling. While I have not seen tools that specifically address this important area of data profiling, some existing tools can be cajoled into providing necessary information with little maneuvering. The profiling can also be done using some advanced queries and data manipulation techniques.

## BASIC RULES FOR STATE-DEPENDENT OBJECTS

Many data quality rules can be derived directly from the state-transition models. Here are the key categories:

- A **state domain constraint** limits the set of allowed states to only those shown in the state-transition model. Invalid states are usually typos inside otherwise valid records. The true state can often be deduced based on the action value.
- An **action domain constraint** limits the set of allowed actions to only those shown in the state-transition model. Invalid actions are usually typos inside otherwise valid records. The true action can often be deduced based on the state value.
- A **terminator domain constraint** limits the set of allowed terminators, specifically states in which an object can start and end its life cycle. Invalid terminators often are a symptom of missing records at the beginning of the life cycle.
- **State-transition constraints** limit state changes to those allowed by the state-transition model. For example, a person who is already terminated cannot be terminated again without being rehired in between. Invalid state-transitions often signify a missing action.
- **State-action constraints** require that each action is consistent with the change in the object state. For instance, when the record with *Active* state is followed by the record with *On Leave* state, the action recorded along with the second record must be *LOA*.
- **Continuity rules** prohibit gaps and overlaps in state-transition history. In other words, they require that the effective date of each state record must immediately follow the end date of the previous state record.
- **Duration rules** put a constraint on the maximum and/or minimum length of time an object can stay in any specific state. The simplest form of the duration rule is the **zero-length rule**, which requires the length of

time spent in each state to be greater than zero. Violations of this rule create time warps common in science fiction and leading to well-known time travel paradoxes. Occasionally, the duration rules will apply to the cumulative amount of time the object can spend in a particular state.

## ADVANCED RULES FOR STATE-DEPENDENT OBJECTS

Object states and actions are often too complex to be adequately described by a single code. Rather, additional action- or state-specific attributes must be stored. Numerous data quality rules can be designed to validate such attributes. These rules fall into two broad categories:

- **Action-specific attribute constraints** enforce that action-specific attributes are populated consistently with the actions. Consider for example the lifecycle of the state-dependent object *ORDER*. When a product is shipped, the package tracking number must be recorded along with the action. The data quality rule for this event will require a valid tracking number to accompany *SHIPPED* action. Further, when a payment is received it might be necessary to store the form of payment (and the amount in case of a partial payment). The data quality rule will enforce that a valid payment code (and payment amount) accompanies *PAYMENT RECEIVED* action.
- For many state-dependent objects, it is necessary to track some state-specific data. For example, an active employee can be designated as full-time or part-time and regular or temporary. These attributes are not action-specific since they can change without any state-altering action, i.e. employees can freely change from part-time to full-time and vice versa but still remain in the same state *Active* from the perspective of the state-transition model. At the same time, these attributes are only applicable to objects in certain state(s). An employee can go from part-time to full-time employment while in states *Active* and *On Leave*, but not in any other state. **State-specific attribute constraints** enforce that state-specific attributes are populated consistently with the object state.

State-specific and action-specific attribute constraints, while rather simple in concept, can take many different forms when applied to the real data. It is important to recognize that the form of these data quality rules will depend on the way the data for the state-dependent object is organized in the database.

Most actions can only take place under certain unique circumstances. Action pre-conditions and action post-conditions verify these circumstances:

- **Action pre-conditions** are the conditions that must be satisfied before an action can take place. For example, an employee may not be hired without her job application being previously considered and approved. Thus, a pre-condition for the *HIRE* action in the employment state-transition model is presence of a job application history record for the same person with state *Job Offer Accepted* and an effective date shortly before the effective date of the *HIRE* event. Another example of a pre-condition is that an action *DEFAULTED* can only be applied to state-

dependent object LOAN when payment history shows payment sufficiently in arrears.

- **Action post-conditions** are the conditions that must be satisfied after the action is successfully completed. For instance, upon retirement the employee must have post-retirement benefits calculated and entered in the appropriate tables of the HR system. Thus, a post-condition for the *RETIRE* event in the employment state-transition model is the presence of an employee benefits record for the same employee with all necessary attributes filled in and effective dated by the retirement date.

Action pre-conditions and post-conditions are very common in state-transition models, yet rarely fully understood. In order to identify these constraints, we must analyze the objects from the business perspective and interview business users. On the other hand, these more complex data quality rules will often find many otherwise hidden, but critical, data quality errors.

### **SUMMARY**

State-dependent objects go through a sequence of states in the course of their life cycle as a result of various events. Data for the state-dependent objects is very common in real world databases and is also most error-prone. Various data quality rules can be implemented to validate such data. Some of these rules are rather simple, while others can be quite complex and vary significantly depending on the data structure. In all cases, data quality rules for state-dependent objects are key to successful data quality assessment, since data for such objects is typically very important and yet contains numerous “hidden” errors.