# Motor de flujo de trabajo, diseño de base de datos

**22 de de abril de, el año 2015**

Nuestra empresa se daba cuenta de que muchos de nuestros procesos internos eran relativamente similares: una persona solicita que algo suceda (por ejemplo, una compra, una aprobación para empezar a construir algo, etc.), luego, una serie de personas aprueban esa solicitud hasta que, por último, el artículo ha sido revisado y aprobado por la persona que necesitaba revisarlo.

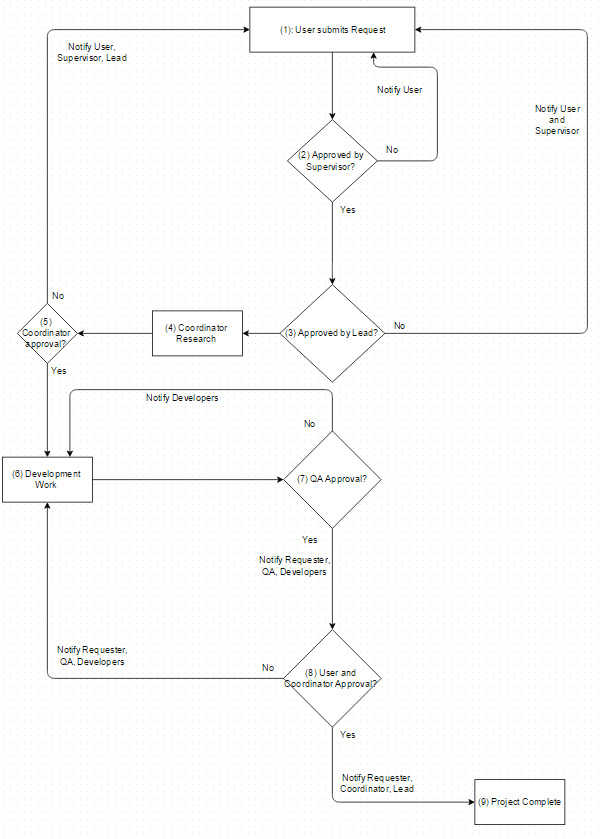
Nuestro equipo (en particular, nuestro grupo de gestión, que tienen que tratar con estas aplicaciones muy similares casi todos los días) comenzó a pensar que, puesto que teníamos tantos procesos de aprobación que eran tan similares, ¿podríamos construir una herramienta con bases de datos para gestionar todos esos procesos?

Me asignaron para investigar estas similitudes, y en el proceso de hacerlo me di cuenta de que estos procesos eran esencialmente máquinas de estados finitos, que tenía una solicitud en un Estado determinado en un momento determinado, y que también define cómo viajar de un estado a otra (transiciones) y lo que tenía que suceder con el fin de invocar ese viaje (acciones).

Sobre la investigación adicional, lo que descubrí fue que estas colecciones de solicitudes, los Estados, transiciones, acciones, etc. tenían un nombre: eran los flujos de trabajo.

## ¿Qué es un flujo de trabajo?

Un flujo de trabajo es una serie de decisiones tomadas por diferentes personas que determinan lo que sucede a una solicitud en concreto según lo que una de esas personas hizo, de acuerdo con un proceso definido y repetible. Un ejemplo de este proceso se muestra en este diagrama de flujo:



### Ejemplo de etapas en un flujo de trabajo

Vamos a ilustrar que el diagrama de flujo de otra manera, mediante la enumeración de los pasos:

1. John Doe envía una solicitud que dice "necesito un sitio que permita a mis empleados registrar su tiempo libre y que puedan revisar rápidamente".
2. John envía la solicitud a su supervisor, Victoria, quien está de acuerdo con él y aprueba la solicitud.
3. Victoria envía la solicitud aprobada a Nate, un desarrollador supervisor, que revisa la solicitud inicial y decide que es una idea viable, él y su equipo puede implementar un sitio web. Aprueba la solicitud.
4. Nate envía la solicitud aprobada a Jenna, el coordinador, quien empieza la investigación (probablemente con la participación de John, Victoria, y Nate) para determinar la cantidad de esfuerzo que esta solicitud requeriría.
5. Si Jenna aprueba la solicitud, se le asigna de nuevo a Nate con la orden de comenzar el desarrollo.
6. Nate y su equipo hacen el trabajo de desarrollo. Este paso puede tardar una cantidad de tiempo variable.
7. Una vez que el trabajo de desarrollo se hace, la solicitud se envía al equipo de control de calidad para su aprobación, liderizado por Charles.
8. Si Charles y su equipo no encuentran problemas de pruebas, se aprueba el proyecto y lo envía a John para su aprobación final (esencialmente todos los equipos están preguntando a John, "esto es lo que querías, ¿verdad?").
9. Si John aprueba el proyecto, se marca completada, y no se pueden tomar más medidas contra esta solicitud.

### Un montón de Similitudes

Tuvimos muchos tipos diferentes de flujos de trabajo en nuestra organización, y el grupo de gestión queríamos programadores para determinar si se podría construir un servicio genérico, con bases de datos que podrían representar a muchos (si no todos) de estos procesos en una base de datos central.

Después de revisar varios de los flujos de trabajo existente en la actualidad, mi equipo y yo descubrimos que todos ellos tenían componentes similares:

* Un **objeto Solicitud (Request)** que se revisa, aprueba o implementa por varias personas.
* Un **conjunto de datos altamente variables** que se asocia a cada solicitud.
* Una **serie de decisiones (llamado Proceso)** que determinan quienes venían después a revisar la solicitud. Estas decisiones se deberán presentar siempre al motor por una persona individual, pero esa persona podría ser una persona relacionada con la solicitud (por ejemplo, el Solicitante o un grupo interesado) o una de un grupo de personas.
* Un **conjunto de notificaciones** que podrían ir a diversos grupos de personas.
* Un pequeño número de personas que estaban **a cargo del propio proceso**.

**El diseño de una base de datos del motor de flujo de trabajo**

En esta serie, vamos a caminar a través del diseño de la base de datos de nuestra aplicación de flujo de trabajo y mostrar cómo se implementó cada parte de la solución, y finalmente la forma en que estaban todos conectados entre sí. Vamos a hacer esto en ocho partes (Este mensaje es parte 1):

* Parte 2: La tabla de proceso y Usuarios
* Parte 3: Solicitud de datos y detalles
* Parte 4: estados y transiciones
* Parte 5: Acciones y Actividades
* Parte 6: Grupos y Metas
* Parte 7: Solicitud de acciones
* Parte 8: Esquema completo y deficiencias

Este no es el único diseño que puede implementar para flujos de trabajo, se ha simplificado de nuestro diseño real implementado. Pero yo creo que poner mis ideas de diseño en un papel (por así decirlo) ayudará un tanto a entender mejor mi diseño y, con suerte, a otros administradores de bases de datos, cuando traten de diseñar una base de datos para sus propios problemas complejos.

**NOTA IMPORTANTE**: No soy un DBA, y este diseño no se ha revisado a fondo por alguien que sea DBA; háganme saber cómo puedo mejorar este. Todas las sugerencias son bienvenidas!

Una nota final: en esta serie estamos mostrando sólo cómo diseñar este motor, en realidad no ponerla en práctica en el código. Lo que esto significa es que no vamos a cubrir ciertos detalles, tales como:

* ¿Cómo los usuarios ingresan a nuestra base de datos? o
* ¿Cómo los correos electrónicos se envían a los destinatarios?

¡Con esas advertencias vamos a explorar cómo diseñar una base de datos para una aplicación de motor de flujo de trabajo, a partir de la tabla de procesos y usuarios!

# Motor de flujo de trabajo, diseño de base de datos parte 2: la tabla de procesos y usuarios

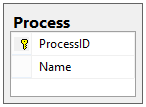
**22 de de abril de, el año 2015**

Antes de que podamos diseñar cualquier otra cosa en esta base de datos para motor de flujo de trabajo, en primer lugar hay que definir qué es exactamente lo que hace un Proceso, así como quiénes son nuestros usuarios y cuáles de ellos pueden modificar el proceso en sí mismo.

## El proceso

Un **proceso** es el **conjunto de todos los otros datos que son exclusivos de un grupo de usuarios y cómo quieren que sus solicitudes sean aprobadas**. En nuestro diseño, un proceso es la infraestructura que se utiliza para definir y asociar la mayoría de la información en esta base de datos.

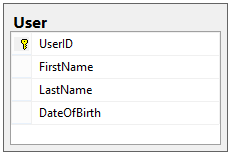
Nuestra tabla de proceso es muy simple, ya que es sólo un ID y un nombre:



Esta tabla, aunque es muy simple, es el punto central de referencia para el resto del diseño; la mayoría de las tablas en esta base de datos (por ejemplo, los Estados, las transiciones, las solicitudes, etc.) tendrán que estar relacionadas a esta tabla, ya sea directa o indirectamente.

## Usuarios

También necesitamos una tabla para enumerar los **usuarios** que pueden acceder a esta aplicación; esto será una tabla de búsqueda que tiene todos nuestros usuarios. Se verá así:

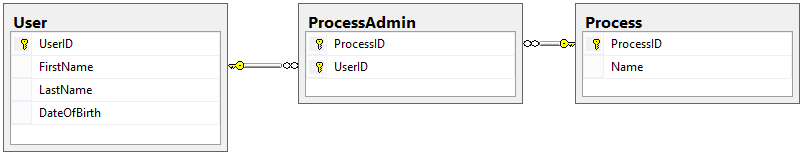


En realidad, no importa cómo los usuarios se agregan a esta tabla; sólo nos importa que exista aquí y que esta tabla actuará como un punto central de acceso a los datos del usuario.

Tal vez se pregunte por qué los usuarios no son específicos de Procesos. Se utilizó este diseño con el fin de permitir a las personas tener puestos de trabajo en varios procesos diferentes sin necesidad de almacenar datos de usuario en múltiples lugares.

## Los administradores de procesos

También queremos permitir un pequeño grupo de usuarios que pueden cambiar el proceso en sí mismo; estas personas se les llama **Administradores de proceso**. La tabla de administradores de proceso es sólo una tabla de relación de muchos a muchos entre el Proceso y el usuario:



## ¿Qué vamos de lograr?

Hemos creado la base sobre la que el motor de flujo de trabajo se fundamenta mediante la creación de las tablas de proceso y los usuarios, y establecimos Administradores de proceso que serán capaces de modificar el proceso en sí mismo.

Pero esto es sólo el comienzo de nuestro viaje. En la próxima entrega, hablaremos de lo que realmente hace una petición cuando hablamos de Solicitud Detalles y datos.

# Motor de flujo de trabajo, diseño de base de datos Parte 3: Detalle de la información y de datos

**22 de de abril de, el año 2015**

Ahora tenemos la información básica del proceso definida, por lo que podemos empezar a abordar las tablas y las relaciones de lo que una solicitud se compone exactamente.

## Partes de una Solicitud

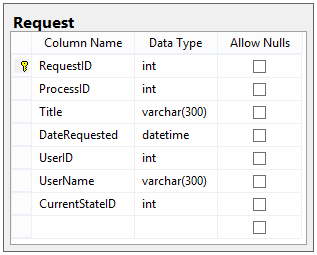
En nuestro motor de flujo de trabajo, una solicitud consta de las siguientes partes básicas:

* **Información Básica**: Un título, una fecha de creación, crear un usuario y una identificación del estado actual.
* **Datos**: Un conjunto altamente variable de los datos que se refiere a una petición individual.
* **Las partes interesadas**: un conjunto de usuarios que se van a recibir actualizaciones periódicas acerca de la solicitud.
* **Archivos**: Los archivos físicos que se relacionan a una petición individual.
* **Notas**: Todas las notas introducidas por los usuarios pertenecientes a una petición individual.
* **Demanda de acciones**: Las acciones que se pueden realizar en cualquier momento dado una petición.

Nos ocuparemos de solicitar acciones en la parte 7 de esta serie .Por ahora, vamos a definir lo que constituye un objeto de solicitud, a partir de la tabla Solicitud básica.

### Fundamentos de solicitud

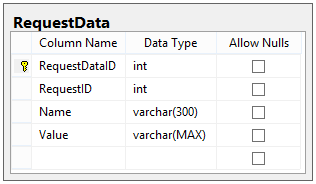
Las solicitudes son exclusivas de Procesos; una solicitud sólo podrá existir en un solo proceso. Una solicitud básica sólo necesita un pedazo de información del usuario (un título) y el resto de la información básica de la solicitud proviene de cómo el proceso se presenta. Nuestra tabla de solicitud se verá así:



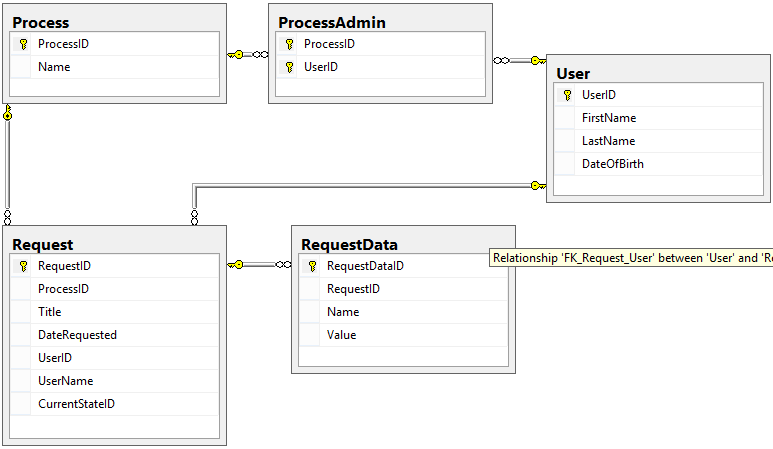
* **Título**: El título de la solicitud.
* **CurrentStateID**: El ID del Estado de la solicitud en que se encuentra actualmente.

### Solicitud de datos

Una vez que tenemos la información básica de la solicitud, seguimos teniendo una forma de almacenar datos que no encajan en el esquema de la tabla de solicitudes. Es muy probable que **cada proceso tenga que almacenar información diferente acerca de sus solicitudes**, por lo que necesitamos un diseño de tabla que sea lo suficientemente flexible como para permitir muchos tipos de datos. Todo lo que realmente sabemos acerca de cada pieza de datos es que tendrá un valor, y necesitamos una manera de determinar cuál es el valor que representa. Por lo tanto, diseñamos la tabla para que sea **un conjunto apareado de nombre y valor**, y nuestro diseño se verá así:



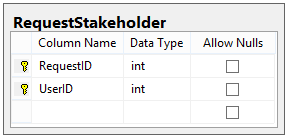
Con Request y RequestData definidos, nuestro diseño completo (incluyendo los cuadros de la parte 2) luce así:



Todavía tenemos tres tablas adicionales para completar lo que comprende una Solicitud. Primero que nada son, las partes interesadas.

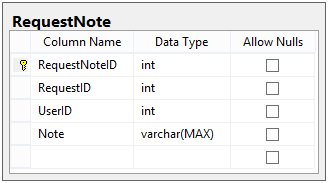
### Las partes interesadas

Una **de las partes interesadas** es una persona que debe recibir actualizaciones periódicas sobre el estado de una solicitud determinada. Esencialmente son usuarios que tienen un interés en el resultado de la solicitud. Lo que esto significa es simplemente que hay una relación *muchos a muchos* entre el Usuario y Solicitud, por lo que la tabla es la siguiente:

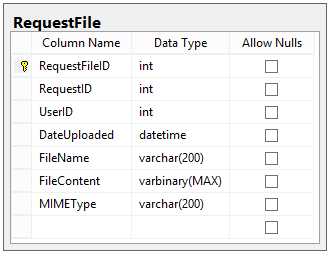


### Notas y Archivos

Queremos permitir a los usuarios añadir notas a una solicitud, de manera que puedan registrar la información adicional no contenida en la solicitud de datos o de los Estados. Nuestra tabla de Notas se verá así:

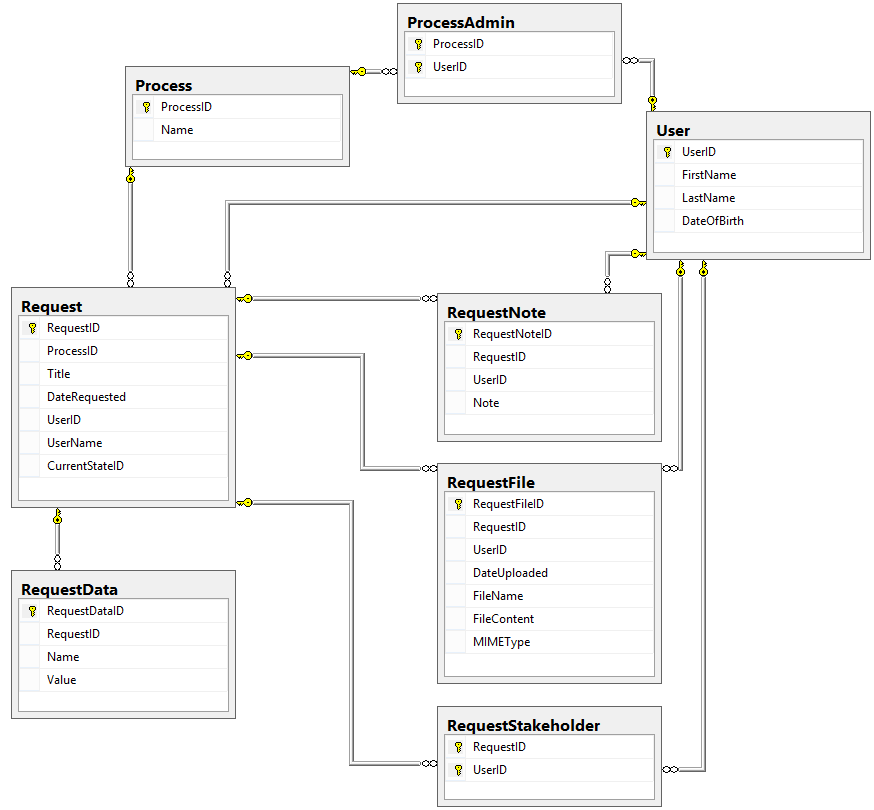


Por último, queremos permitir a los usuarios subir archivos que están relacionados directamente con solicitudes dentro de la base de datos (una solución más flexible puede ser almacenar archivos en un servidor de archivos, pero por simplicidad vamos a almacenar los archivos de la base de datos para este diseño). Necesitamos almacenar un nombre de archivo, tipo MIME, y el contenido real del archivo en esta tabla. La tabla RequestFile se verá así:



### ¿Qué logramos?

Después de implementar la información de la solicitud, nuestro diseño completo es el siguiente:



Al completar las tablas de peticiones, hemos dado forma a los datos que realmente van a ir a través de los procesos de homologación indicadas en esta base de datos. Pero, ¿qué son exactamente estos procesos de aprobación, y cómo podemos definirlos? En la Parte 4 de esta serie, vamos a empezar a diseñar cómo el proceso en sí mismo está estructurado mediante el diseño de tablas de **estados y transiciones**.

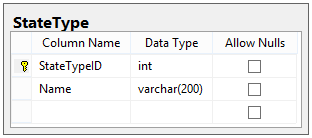
# Motor de flujo de trabajo, diseño de base de datos Parte 4: estados y transiciones

**22 de de abril de, el año 2015**

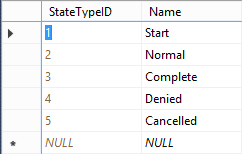
Ahora que tenemos las tablas de proceso y solicitud creados, podemos empezar la construcción de tablas para el proceso en sí. En esta parte, diseñaremos las tablas que mantendrán los diferentes Estados en los que puede estar una Solicitud como parte de un Proceso, y también vamos a diseñar las tablas que muestran cómo llegar de un Estado a otro (que se llaman transiciones). Primero, sin embargo, necesitamos una tabla que muestre los diferentes tipos de Estados que pueden existir. ¡Empecemos!

## Tipos de Estado

Un **tipo de Estado** es una categorización de los Estados individuales. En nuestro diseño, es una lista inmutable, por lo que en el código probablemente utilice una enumeración para representarla. Ya que estamos haciendo este diseño de base de datos totalmente normalizado, vamos a incluir este conjunto de datos en forma de tabla, con la siguiente estructura:



Debido a que no quieren tener ningún Tipos de Estados definidos por el usuario, no hay relación al proceso para esta tabla. Estamos tratando a esta tabla como inalterable, y siempre tendrá los siguientes valores:



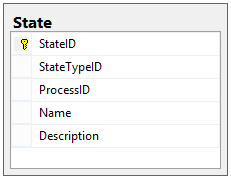
Aquí está el razonamiento para cada uno de estos tipos:

* **Start (Iniciada)**: Sólo debe ser uno por proceso. Este estado es el estado en el que se coloca una nueva solicitud cuando se crea.
* **Normal (Normal)**: Un estado regular sin ninguna designación especial.
* **Complete (Completada)**: Un estado que significa que cualquier solicitud en este estado ha finlizado normalmente.
* **Denied (Negada)**: Un estado que significa que cualquier solicitud en este estado se ha negado (por ejemplo, no se inició y no será trabajado).
* **Cancelled (Cancelada)**: Un estado que significa que cualquier solicitud en este estado ha sido cancelada (por ejemplo, el trabajo fue iniciado pero nunca se terminó).

Cada Estado debe tener exactamente uno de estos tipos de Estado. Pero, ¿qué hace un Estado?

## Estado

Un **Estado** es **una posición en el proceso que una solicitud puede tener en un momento dado**. Estado son exclusivos de los procesos, y cada estado tiene un nombre, una descripción y un tipo. Nuestra tabla de estado es el siguiente:

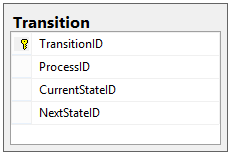


Debemos recordar, sin embargo, que cada proceso se supone que representa un diagrama de flujo, y para ello tenemos que ser capaces de mover las solicitudes entre los Estados. Podemos hacerlo mediante el diseño de tablas para las transiciones.

## Transiciones

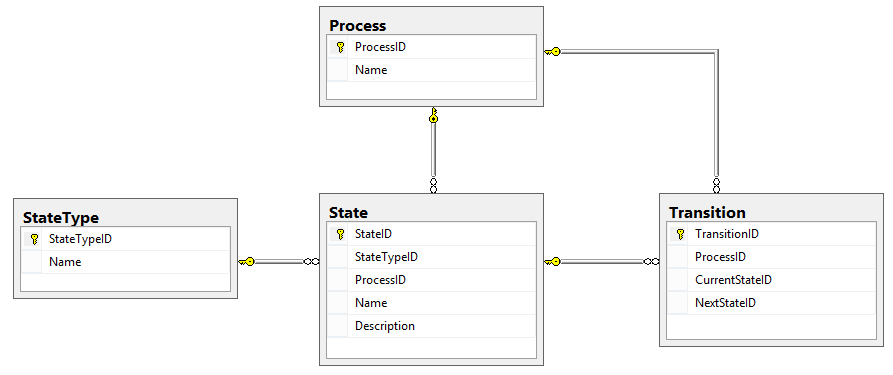
Si un desarrollador principal aprueba una solicitud, y ahora debe ir al equipo coordinador de la investigación, ¿cómo podemos diseñar nuestros datos con el fin de representar que ese movimiento es posible? Creamos una transición.

Una **transición** es **un camino entre dos Estados que muestra cómo una solicitud puede viajar entre ellas**. Las transiciones son exclusivas de los procesos, y por lo tanto una transición está compuesta de una clave principal, un identificador de proceso, un estado actual y un estado siguiente:



### ¿Qué logramos?

Nuestro diseño de base de datos actual (mostrando sólo los Estados, transiciones, y las tablas de proceso) se ve algo como esto:



En esta publicación, dimos forma a los bloques de construcción del Proceso definiendo los Estados donde puede existir una Solicitud y las Transiciones entre esos Estados.

Todavía nos queda un problema pendiente, sin embargo: ¿cómo invocamos nuestras transiciones? ¿Cómo podemos realmente hacer que una solicitud pase de un estado a otro y lo que debería ocurrir cada vez que vamos a un nuevo Estado (o seguir una nueva Transición) para una determinada solicitud? Todo eso y mucho más en el próximo, Parte 5 de esta aventura, que trata sobre **las acciones y actividades**.

# Motor de flujo de trabajo, diseño de base de datos Parte 5: Acciones y Actividades

**22 de de abril de, el año 2015**

Después de haber definido nuestra infraestructura de procesos, nuestra estructura de solicitud, y nuestros estados y transiciones, ahora podemos empezar a diseñar tablas para lo que un usuario puede realmente hacer a una solicitud en este motor. Para ello, definiremos dos términos:

* **Acciones**: Cosas que un usuario puede realizar en una solicitud.
* **Actividades**: Las cosas que resultan de una Solicitud al trasladarse a un Estado determinado o después de una transición en particular.

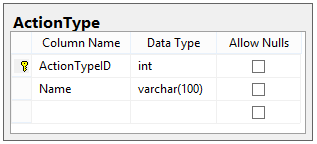
Vamos a empezar por la creación de acciones y de tipos de acción.

## Acciones

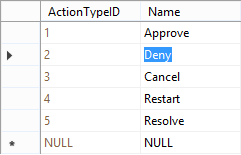
Las acciones son cosas que un usuario puede realizar a la Solicitud.

Digamos que tenemos una solicitud para construir una nueva tienda de comestibles, y que la solicitud incluye la dirección donde se debe construir el desarrollo. La persona que está a cargo de aprobar nueva construcción de tiendas, John, echa un vistazo a la solicitud y decide que, sí, es una buena idea para construir una tienda aquí. Cuando presente la aprobación de la solicitud, debe hacer que la Solicitud pase al siguiente estado. John ha presentado una acción.

Dado que no queremos permitir un número infinito de tipos de acciones que se pueden realizar, vamos a agrupar acciones a través de una tabla de ActionType que tiene este aspecto:



Al igual que la tabla StateType, esta tabla es independiente del proceso y se considerará estática. Para nuestra base de datos, vamos a utilizar los siguientes tipos de acciones:

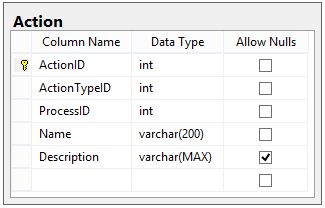


¿Por qué utilizamos este tipo de acción?

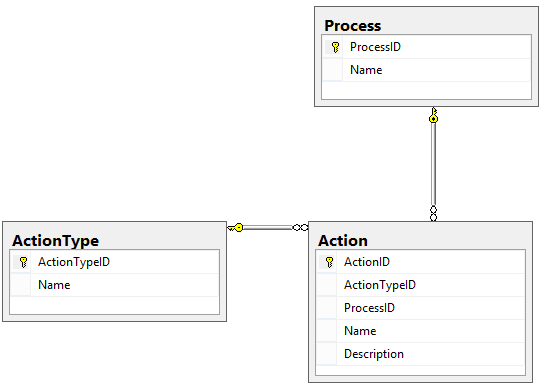
* **Approve (Aprobada)**: la acción sugiere que la solicitud debe pasar al siguiente estado.
* **Deny (Denegada)**: La acción sugiere que la solicitud debe pasar al estado anterior.
* **Cancel (Cancelada)**: La acción sugiere que la solicitud debe pasar al estado Cancelado en el proceso.
* **Restart (Volver a empezar)**: La acción sugiere que la solicitud se traslade de nuevo al estado de inicio en el proceso.
* **Resolve (Resuelta)**: la acción sugiere que la solicitud sea movida al estado Completado.

La razón por la que decimos que la persona "sugiere" que la solicitud se mueva es que **queremos permitir que un proceso requiera múltiples acciones para invocar una transición**. Es posible que una solicitud necesite que sucedan varias cosas antes de que pueda continuar en el proceso, y queremos permitir ese escenario.

Ahora necesitamos la tabla para las propias acciones. Acciones son exclusivas de los procesos, y cada uno tiene un ActionType, por lo que la tabla se verá así:



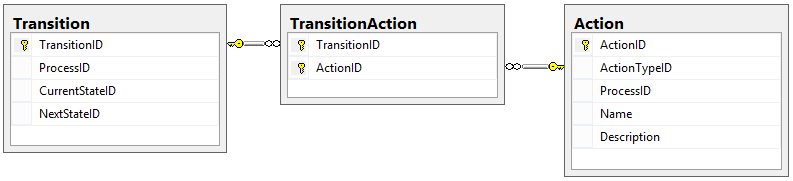
Nuestro diseño de ActionTypes y acciones es el siguiente:



## Acciones de transición

Ahora que hemos definido qué acciones se pueden realizar, tenemos que ser más específicos: ¿qué acciones se realizan para una transición en particular?

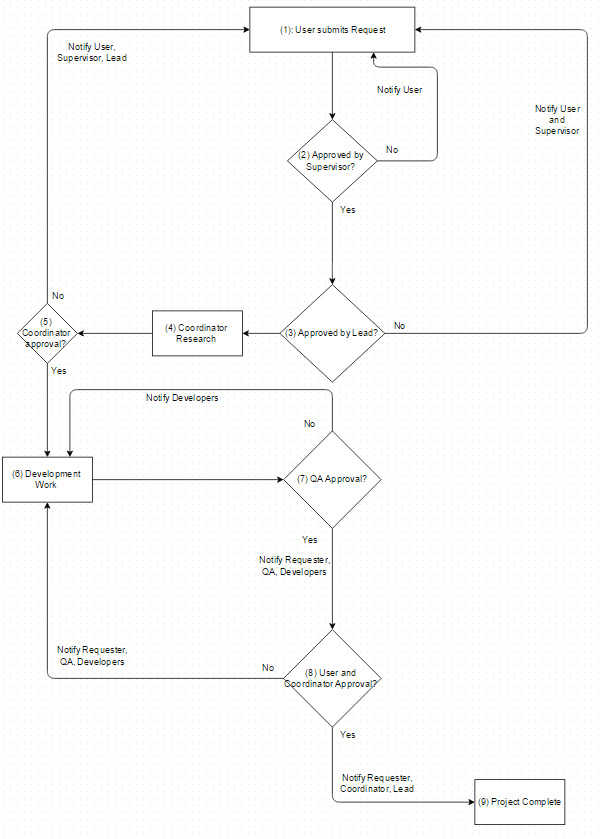
La relación entre la transición y la acción es de muchos a muchos:



## Actividades

**Las actividades** son cosas que pueden ocurrir como resultado de que una solicitud entre en un Estado o después de una transición.

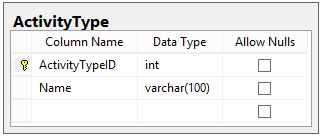
Por ejemplo, vamos a ver el diagrama de la parte 1 de nuevo.



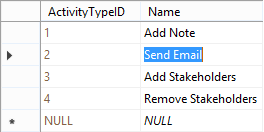
En el paso 3 de este diagrama de flujo, es posible que desee agregar al líder como parte interesada en una solicitud, por lo que él/ella recibirá correos electrónicos automáticos sobre el estado de la solicitud. Sin embargo, si el líder rechaza la solicitud vamos a querer notificar al solicitante, pero si aprueba la solicitud hay que notificar a los coordinadores.

En otras palabras, en este ejemplo la adición de un grupo de interés es una actividad que queremos que suceda cuando una petición llega a un cierto estado, y el envío de correo electrónico es una actividad que queremos que suceda cuando se sigue una cierta transición. Tenemos que diseñar para ambos escenarios.

En primer lugar, necesitamos saber qué tipo de acciones podemos hacer. Esta tabla es igual que StateType y ActionType no es única para el proceso y puede considerarse estática. Aquí está el diseño de la tabla ActivityType:



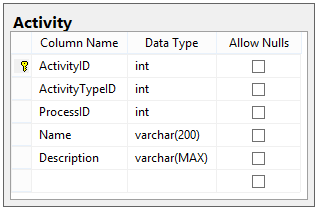
Al igual que ActionType, los valores para ActivityType son estáticos. Vamos a utilizar los siguientes valores:



* **Add Note (Añadir Nota)**: especifica que hay que añadir automáticamente una nota a una solicitud.
* **Send Email (Enviar por correo electrónico)**: especifica que hay que enviar un correo electrónico a uno o más destinatarios.
* **Add Stakeholders (Añadir partes interesadas)**: especifica que hay que añadir una o más personas como partes interesadas en esta solicitud.
* **Remove Stackholders (Retire las partes interesadas)**: especifica que hay que eliminar uno o más grupos de interés de esta solicitud.

Se podría definir bastantes más tipos de ActivityTypes, pero por ahora sólo tendremos que usar esos cuatro.

La última cosa que necesitamos hacer es diseñar nuestra tabla de Actividad, que se parece mucho a la tabla de Acción:

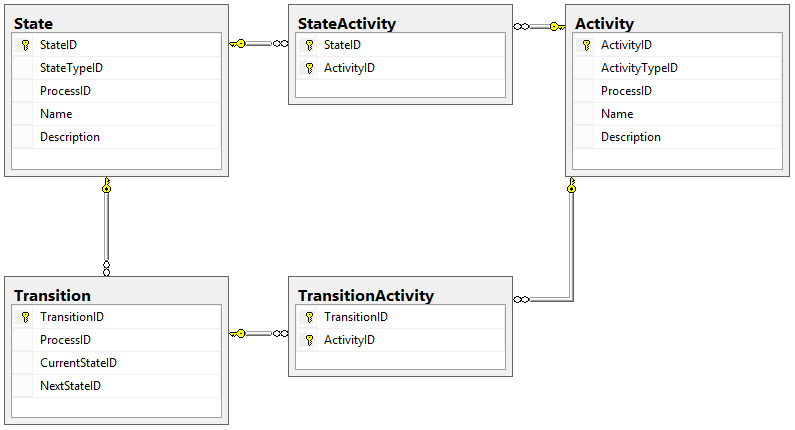


## Actividades de Estado y Transición

Una vez que tenemos la tabla base de Actividad definida, podemos empezar a diseñar cómo las actividades se asocian a estados y transiciones. A modo de recordatorio, queremos ser capaces de poner en marcha actividades en dos situaciones:

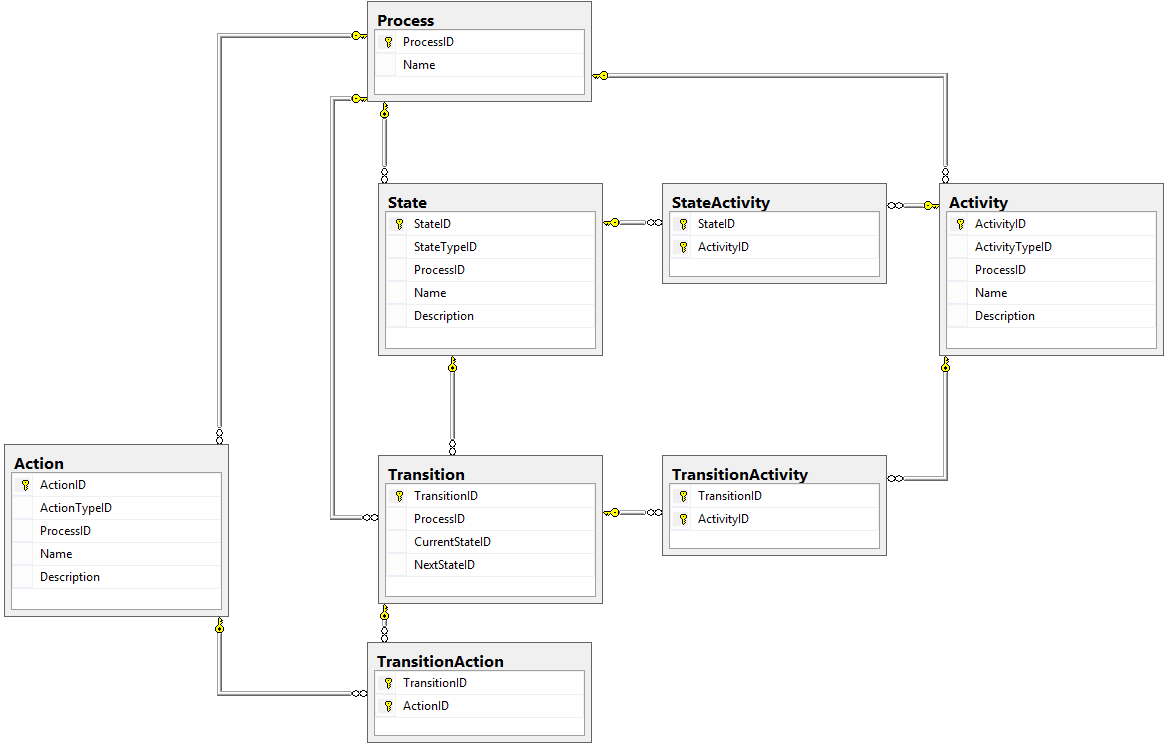
* Cuando la solicitud entra en un Estado
* Cuando la solicitud se produce después de una transición

Esto significa que todavía tenemos que asociar Actividades con los estados y transiciones, así:



### ¿Qué logramos?

Nuestro esquema de base de datos (que muestra el proceso, los Estados, transiciones, acciones y actividades) es el siguiente:



En esta publicación, demostramos cómo podemos almacenar qué acciones pueden ser realizadas por los usuarios, y qué tipo de actividades se iniciaron por ciertos Estados o transiciones. En esencia, mostramos lo que los usuarios pueden hacer con las solicitudes, y qué pasa con los usuarios como consecuencia de ello.

Todavía tenemos un pedazo que falta de todo esto, sin embargo: en realidad exactamente ¿quién puede realizar las acciones o recibir las actividades? Vamos a responder a esa pregunta en la, Parte 6 de esta serie, donde discutiremos **Grupos y Objetivos**.

# Motor de flujo de trabajo, diseño de base de datos Parte 6: Grupos y Metas

**22 de de abril de, el año 2015**

Ahora tenemos la mayor parte de los cuadros de proceso definidos, pero todavía faltan algunas cosas.

Una de esas cosas es exactamente una definición para que pueda realizar o recibir acciones Actividades; vamos a llamar a esta **Objetivos**.

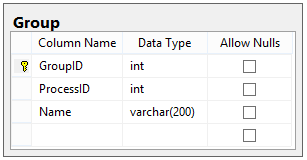
La segunda pieza que necesitamos es **Grupos** o colecciones de las personas que realizan un trabajo similar o relacionado en este proceso. En nuestro diseño, queremos permitir que un grupo sea también un objetivo. ¡Vamos a ver cómo podemos diseñar estas tablas!

## Grupos

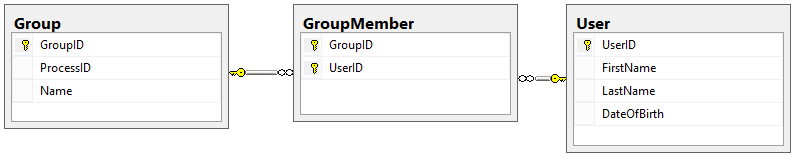
Un grupo es **un conjunto de usuarios que realizan funciones relacionadas**.

Por ejemplo, podríamos tener un grupo llamado Los programadores que realmente escriben código o un grupo llamado Los ejecutivos que aprueban los proyectos para el desarrollo. Debido a que estamos construyendo un motor de flujo de trabajo genérico, queremos asegurarnos de que cada proceso puede tener sus propios grupos.

La tabla del grupo es bastante simple:



También necesitamos una tabla para representar a los usuarios que están en un grupo determinado, que será otra tabla de muchos a muchos. Esta tabla se llama GroupMember:



## Objetivos

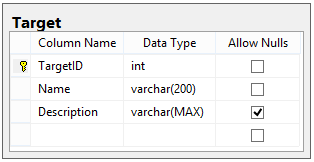
Recuerde que este sistema está orientado hacia las personas; sólo las personas pueden hacer solicitudes. Todavía necesitamos una manera de asociar qué personas (o Grupos) pueden realizar Acciones y recibir Actividades.

Probablemente el problema es que tenga que especificar que sólo el solicitante de una solicitud determinada puede enviar esa solicitud a su supervisor(a), o cuales de estas personas necesitan recibir un correo electrónico cuando una Solicitud alcanza un determinado Estado. ¿Cómo podemos diseñar esto de manera flexible en quién puede realizar la acción o recibir la actividad, implementando una pocas reglas que debe seguir el motor?

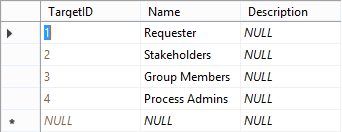
Podemos lograr esto mediante la creación de **Objetivos**. Un objetivo es **un conjunto de representaciones generalizadas de una persona que tienen funciones específicas relativas a una solicitud o de proceso**. Utilizamos los siguientes objetivos:

* Solicitud Creador (Solicitante)
* Las partes interesadas de solicitud
* Miembros del grupo
* Los administradores de procesos

Nuestra Tabla de destino tiene el siguiente aspecto:



Debido a que esta es otra tabla estática (como StateType, ActionType, y ActivityType), no esperamos que los datos cambien. Aquí están los datos que va a utilizar para este diseño:



## Los objetivos de acción y los objetivos de actividad

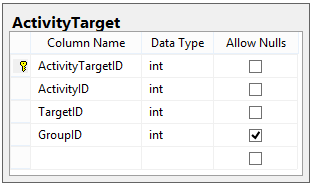
La tabla de Objetivos no hace nada a menos que podamos relacionarlo con otras tablas que realmente pueden utilizar los Objetivos. Queremos usar Objetivos en dos escenarios:   
\* Como las personas pueden realizar acciones \* Como las personas pueden recibir Actividades

Vamos a diseñar un Objetivo Acción primero. Por cada acción que pueda ser enviada al motor, es necesario definir quién puede enviar una acción para que pueda ser considerada válida. Después de todo, no queremos que el conserje Kevin apruebe la construcción de una nueva tienda de comestibles, ya que no es su responsabilidad. Con esto en mente, nuestro diseño de la tabla es el siguiente:

La tabla ActionTarget, mostrando ActionTargetID, ActionID, TargetId y GroupID opcional

Debemos incluir ID de grupo, porque si nuestro objetivo es un grupo, tenemos que especificar qué grupo puede realizar la acción.

Ahora vamos a hablar de Objetivos Actividad. Dependiendo del tipo de actividad, los Objetivos para esa actividad podrían recibir un correo electrónico, o ser añadidos a la lista de actores para una solicitud, etc. Debido a que hemos definido una lista central de Objetivos, la tabla ActivityTarget se asemeja mucho a la tabla ActionTarget:

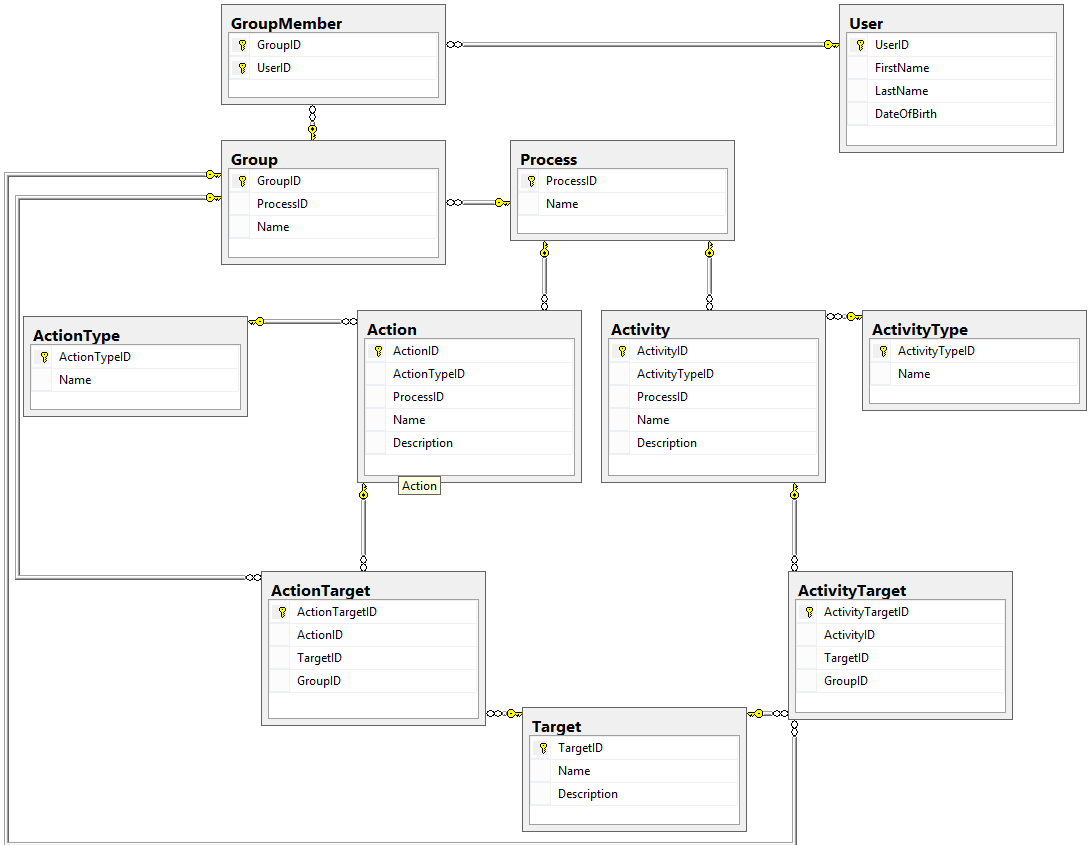


**NOTA IMPORTANTE**: Cuando se utiliza un grupo como destino, la forma en que el sistema interpreta esta relación es diferente.

* Si el grupo es un Objetivo Acción, entonces, cualquier miembro del grupo puede realizar la acción para que sea válida.
* Si el grupo es un Objetivo de la Actividad, entonces, todos los miembros del grupo reciben la actividad (por ejemplo, todos en el grupo recibe un correo electrónico).

### ¿Qué logramos?

Nuestro diseño de Acciones, Actividades, Grupos y Objetivos ahora se ve así:



En esta parte, hemos definido exactamente quién podría llevar a cabo Acciones y recibir Actividades mediante la creación de Objetivos, y dimos a nuestro motor más de flexibilidad mediante la creación de Grupos de usuarios que podían hacer cada uno lo mismo que los demás.

En la siguiente parte de esta serie, se llega al corazón real del sistema. Vamos a mostrar cómo un Solicitante individual pueden realizar un seguimiento de las Acciones que le corresponde, y vamos a ver cómo podemos utilizar esa lista para determinar qué Transición debe seguir la Solicitud. El siguiente es la parte 7 de esta serie, **solicitar acciones**.

# Diseño de una base de datos para motor de flujo de trabajo Parte 7: Solicitar acciones

**22 de de abril de, el año 2015**

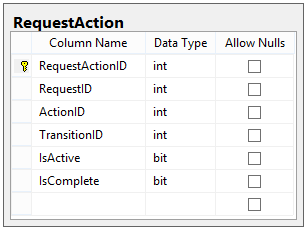
Toda la infraestructura que hemos construido hasta ahora ha conducido a este momento. Por fin, podemos construir la pieza final de nuestro esquema: la tabla Acciones de la Solicitud.

## Acciones de Solicitud

Así que ahora tenemos acciones que los usuarios pueden llevar a cabo para invocar transiciones. Obviamente, no podemos permitir cualquier acción que debe realizarse en contra de la solicitud; sólo debe permitirse las acciones que necesitamos.

## La Tabla

Veamos el esquema de la tabla RequestActions primero, y luego mostramos la forma en que se utilizaría en realidad:



Esas últimas dos columnas (IsActive y isComplete) son muy importantes para la ejecución real de esta tabla.

Así es como vamos a utilizar esta tabla.

1. Cuando una solicitud entra en un Estado, obtenemos todas las transiciones salientes de ese estado. Para cada acción en esas transiciones, añadimos una entrada en RequestAction, con IsActive = 1 y IsCompleted = 0.
2. Un usuario puede enviar una Acción en cualquier momento. Cada acción presentada consiste en un ActionType, un RequestID, y un UserID.
3. Cuando se presente una acción, comprobamos los RequestActions para la Solicitud especificada. Si la Acción enviada coincide con una de las Solicitudes Activas (donde IsActive = 1), cambiamos IsActive= 0 y IsCompleted = 1.
4. Después de marcar la acción presentada como completada, comprobamos todas las acciones de transición en dicha solicitud. Si todos los RequestActions se marcan como Completado, entonces desactivamos todas las acciones restantes (mediante el establecimiento de IsActive = 0, por ejemplo, en todas las acciones de Transiciones que coincidian).

### Tutorial de ejemplo

Vamos a ver si podemos ver cómo esto funciona mediante la introducción de algunos datos de la muestra.

**USUARIOS** : Jane (ID 1), Tom (ID 2), Gary (ID 3)

**GRUPOS** : Ejecutivos (ID 1), incluye Tom y Gary

**ESTADO** : A (Tipo: Inicio), B (Tipo: Normal), C (Tipo: Denied)

**TRANSICIONES** : A -> B (ID 1), A -> C (ID 2), B -> C (ID 3)

**ACCIONES DE TRANSICIÓN** : A -> B: Aprobado por Solicitante (ID 1) y aprobado por los ejecutivos (ID 2)

A -> C: Denegado por ejecutivos (ID 3)

B -> C: Denegado por el Solicitante (ID 4)

Digamos que Jane crea una solicitud, que de inmediato se coloca en el Estado A.

En este punto, el sistema busca todas las transiciones salientes desde el Estado A y se encuentra a dos de ellos, las transiciones 1 y el 2. A continuación, carga los datos siguientes en RequestActions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RequestID** | **ActionID** | **TransitionID** | **Está activo** | **Está completo** |
| 1 | 1 | 1 | SÍ | NO |
| 1 | 2 | 1 | SÍ | NO |
| 1 | 3 | 2 | SÍ | NO |

Ahora, la solicitud permanece en su estado actual, a la espera de una acción que se presentará.

Jane somete esta acción:

Recurso   
ID de usuario: 1

ActionType: Aprobar

Solicitud ID: 1

(se lee que "Usuario 1 aprueba la petición 1")

Dado que la acción coincide con el primer requestAction en la tabla, se marca como completada:

| **RequestID** | **ActionID** | **TransitionID** | **Está activo** | **Está completo** |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | **NO** | **SÍ** |
| 1 | 2 | 1 | SÍ | NO |
| 1 | 3 | 2 | SÍ | NO |

En este punto, no hacemos coincidir todas las acciones para una transición, por lo que no pasa nada a la solicitud; permanece en el Estado A.

Ahora Tom presenta esta acción:

Recurso   
ID de usuario: 2

ActionType: Aprobar

Solicitud ID: 1

(Usuario 2 aprueba la solicitud 1)

Después de que la acción coincide, la tabla de solicitar acciones (ActionRequest) ahora se ve así:

| **RequestID** | **ActionID** | **TransitionID** | **Está activo** | **Está completo** |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | NO | SÍ |
| 1 | 2 | 1 | **NO** | **SÍ** |
| 1 | 3 | 2 | SÍ | NO |

Nótese que ambas Acciones para la Transición 1 se completan, ahora pasamos a la Transición 1 y mover la solicitud al Estado siguiente, que es el Estado B. Después de movernos al Estado B, cargamos las acciones para las transiciones de ese Estado y desactivamos las acciones previas; nuestra tabla RequestActions se ve así:

| **RequestID** | **ActionID** | **TransitionID** | **Está activo** | **Está completo** |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | NO | SÍ |
| 1 | 2 | 1 | NO | SÍ |
| 1 | 3 | 2 | **NO** | NO |
| 1 | 4 | 3 | SÍ | NO |

De esta manera, podemos mantener un registro de todas las acciones que se han realizado, que podrían haber sido realizadas, y que aún están a la espera de realizar, en la misma tabla.

### ¿Qué logramos?

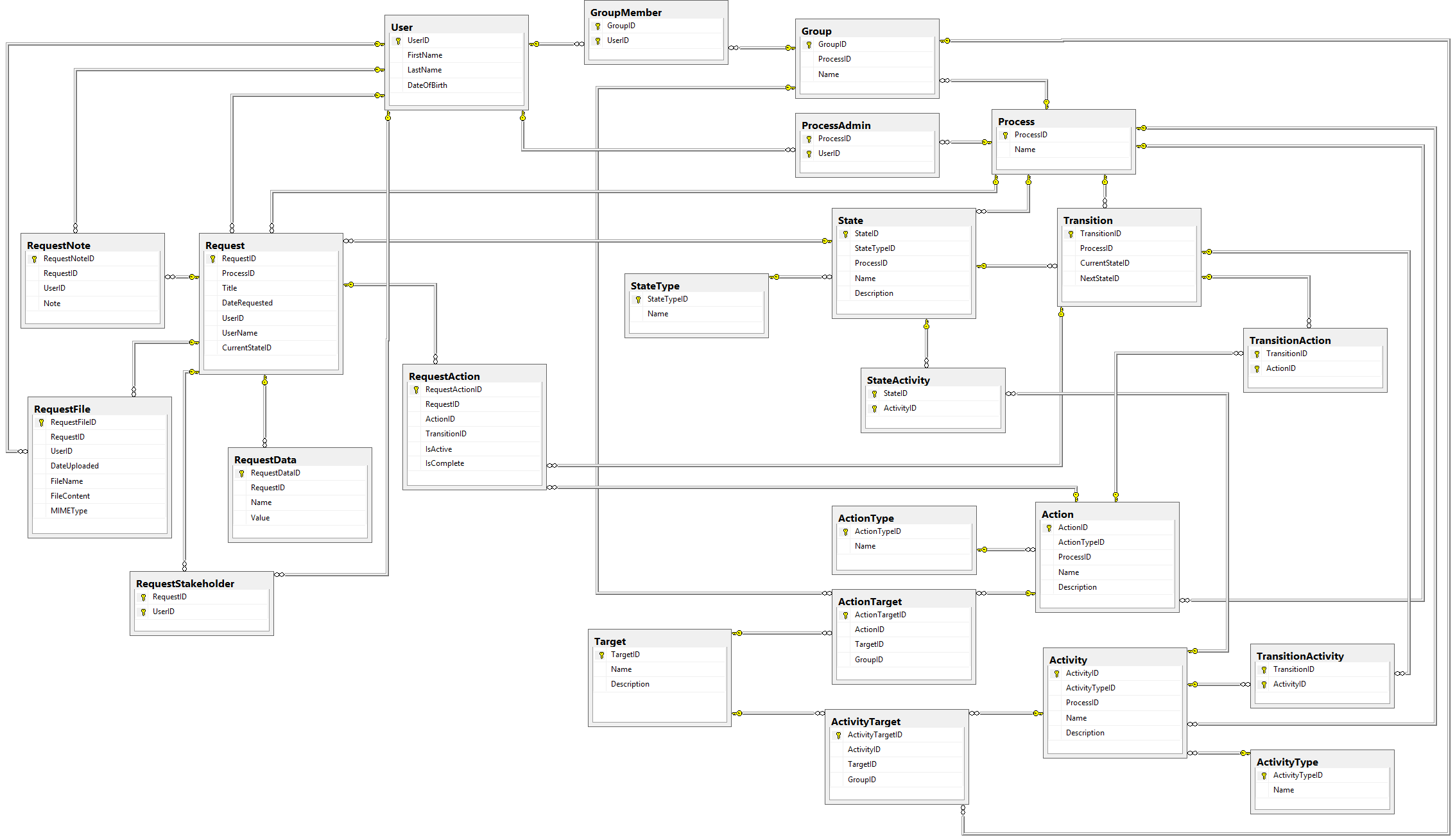
En este escrito, se implementó la última pieza de nuestra estructura: la tabla de Acciones de Solicitud. Esta tabla almacena todas las acciones que se pueden hacer contra una determinada solicitud, y es la fuerza impulsora de cómo funciona realmente este motor.

Todavía hay una última parte de la que hablaremos, que en realidad es la construcción y demostración del proceso que se ejecuta en este motor. La parte final de nuestra saga es la parte 8, donde vamos a discutir el **esquema completo y deficiencias** de nuestro diseño de la base de datos del motor de flujo de trabajo.

# Diseño de una base de datos para motor de flujo de trabajo Parte 8: Esquema completo y deficiencias

**22 de de abril de, el año 2015**

Ahora que hemos completado el diseño de base de datos para nuestro motor de flujo de trabajo, por lo que vamos a echar un vistazo al diagrama de base de datos final:



!Esto es un montón de tablas! Pero estamos tratando de diseñar un motor genérico, uno que se puede utilizar para muchos procesos diferentes, y como tal la complejidad de este diseño refleja la complejidad de los requisitos.

### Las deficiencias de este diseño

Hay algunas deficiencias de este diseño:

* Actualmente, no hay forma de decir "Permanezca en este estado durante X días; después de X días, pasar a otro estado." No podemos hacer que las Solicitudes se muevan por el Proceso según el tiempo transcurrido.
* No tenemos un procedimiento para manejar que un Estado o transición deje de existir (Es decir, aunque se supone que los Administradores de Proceso pueden cambiar el Proceso en sí, ¿qué sucede cuando hacen eso?
* La tabla RequestFiles almacena los archivos reales en la propia tabla; una forma más eficaz para hacer esto sería la de almacenar los archivos en un almacén de archivos en algún lugar y sólo mantener las referencias a ellos en esta tabla.
* Actualmente, no tenemos una manera de indicar que un individuo específico tiene que realizar una acción o recibir una Actividad.

Dicho esto, la mayoría de estas deficiencias son bastante fáciles de implementar; resolvimos algunos de ellos para el motor de flujo de trabajo de nuestra empresa.

Recuerde: **esto es sólo un esquema de base de datos**. El diseño no tiene en cuenta cómo este servicio se implementa en el código.

¡Uf! ¡Fueron una gran cantidad de pasos! Pero terminamos con un **genérico, extensible, reutilizable diseño de base de datos del motor de flujo de trabajo** que puede manejar una gran variedad de procesos.

¡Gracias por leer! Si has encontrado esta serie útil (o sólo tiene que gritar a mí por algo que incorrectamente) que me haga saber en los comentarios.

Feliz Codificación!

#### [Matthew Jones](https://www.exceptionnotfound.net/author/matthew-jones/)

Soy desarrollador en ASP.NET y Microsoft que me gusta enseñar y programar.

*Phoenix, AZ*[*http://www.exceptionnotfound.net/about-me*](http://www.exceptionnotfound.net/about-me)