Designing a Workflow Engine Database Part 3: Request Details and Data

exceptionnotfound.net/designing-a-workflow-engine-database-part-3-request-details-and-data

April 22, 2015

This is Part 3 of an eight-part series describing how to design a database for a Workflow Engine. Click here for Part 1

We now have our basic Process information defined, so we can start tackling the tables and relationships for exactly what a Request is comprised of.

Parts of a Request

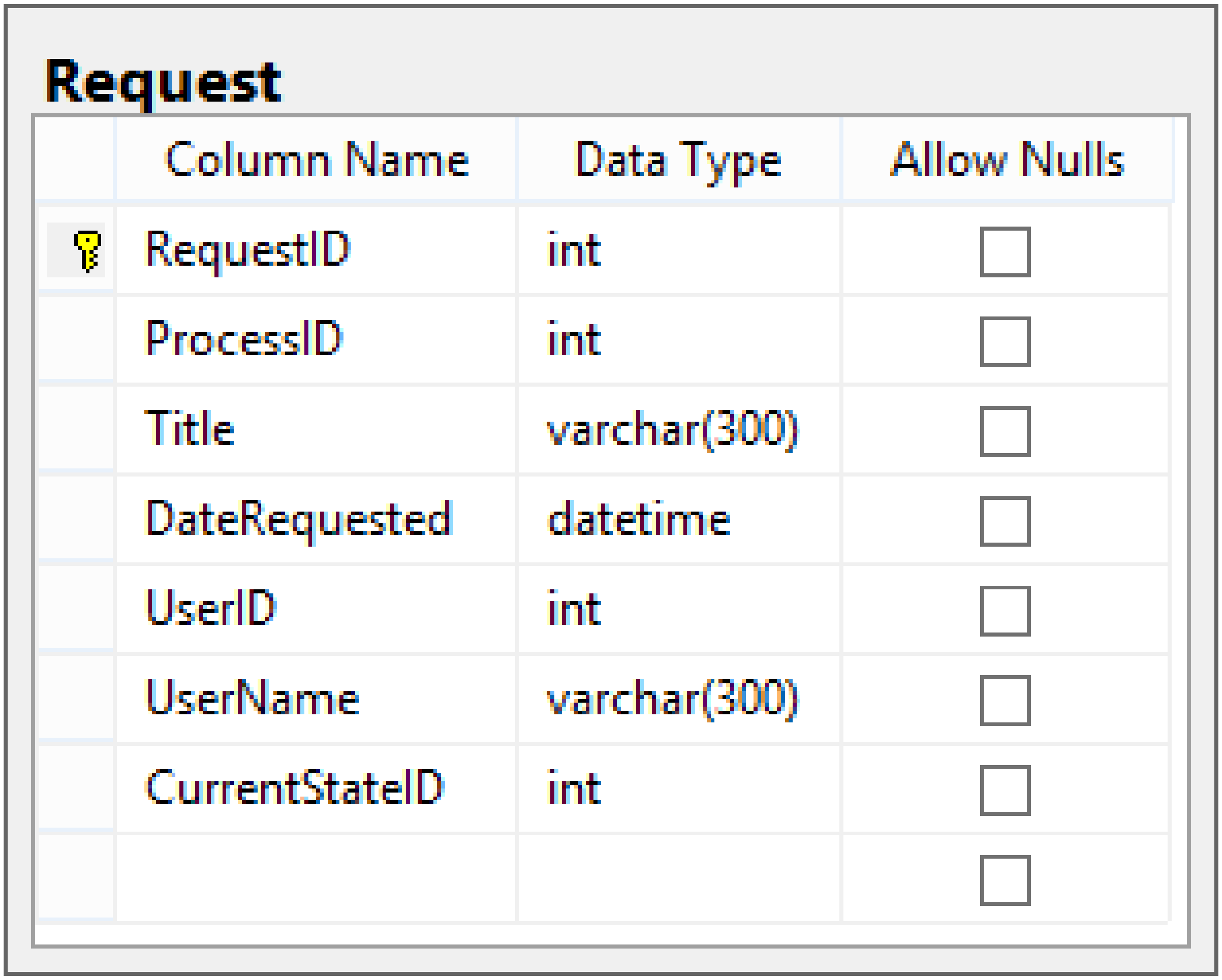
In our workflow engine, a Request has these basic parts:

- **Basic Info**: A title, a create date, a create user, and a current state ID.
- **Data**: A highly-variable set of data that pertains to an individual Request.
- **Stakeholders**: A set of Users that are to receive periodic updates about the Request.
- **Files**: Any physical files that relate to an individual Request.
- **Notes**: Any notes entered by Users pertaining to an individual Request.
- **Request Actions**: The Actions that can be performed at any given time upon a Request.

We will deal with Request Actions in Part 7 of this series. For now, let's define what makes up a Request object, starting with the basic Request table.

Request Basics

Requests are unique to Processes; a Request may only exist in a single Process. A basic Request only needs one piece of information from the User (a title) and the rest of the Request's basic information comes from how the Process is laid out. Our Request table will look like this:



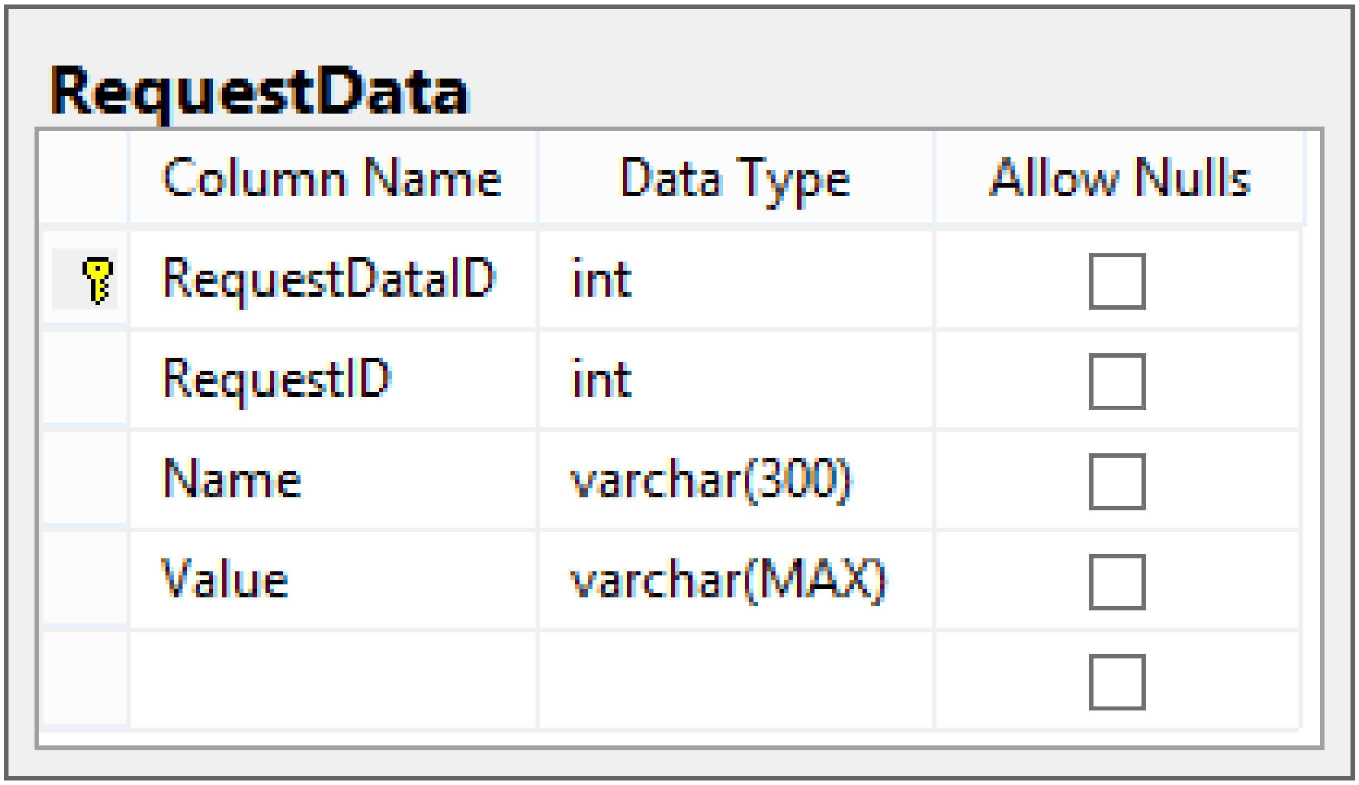
Request

	Column Name	Data Type	Allow Nulls
🔑	RequestID	int	☐
	ProcessID	int	☐
	Title	varchar(300)	☐
	DateRequested	datetime	☐
	UserID	int	☐
	UserName	varchar(300)	☐
	CurrentStateID	int	☐
			☐

- **Title**: The title of the request.
- **CurrentStateID**: The ID of the State the Request is currently in.

Request Data

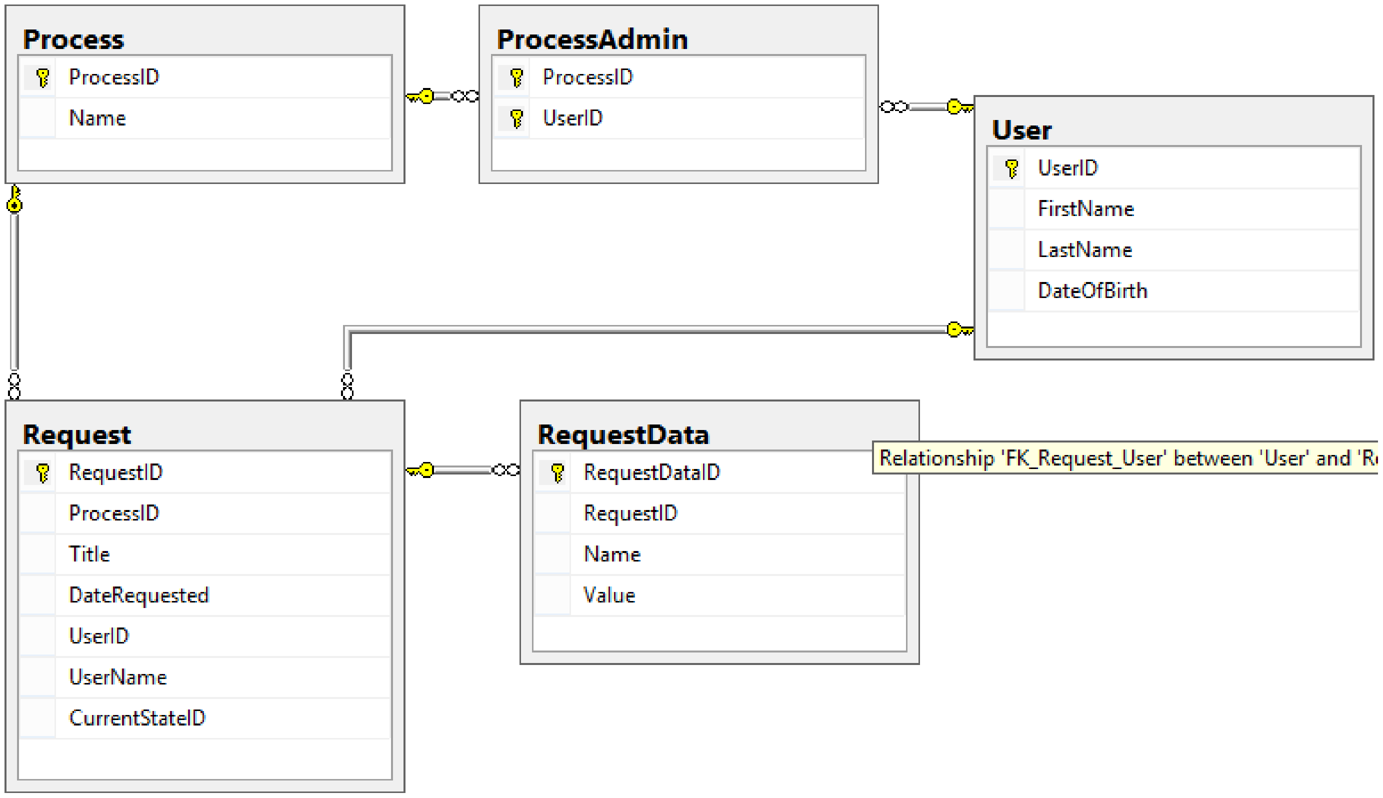
Once we've got the Request basic info down, we still a way to store data that doesn't fit in the Request table's schema. It's very likely that **each Process will need to store different information about their Requests**, so we need a table design that's flexible enough to allow for many kinds of data. All we really know about each piece of data is that it will have a value, and we need a way to determine what the value represents. Hence, we design the table to be **a set of name-value pairs**, and our design will look like this:



RequestData

	Column Name	Data Type	Allow Nulls
🔑	RequestDataID	int	☐
	RequestID	int	☐
	Name	varchar(300)	☐
	Value	varchar(MAX)	☐
			☐

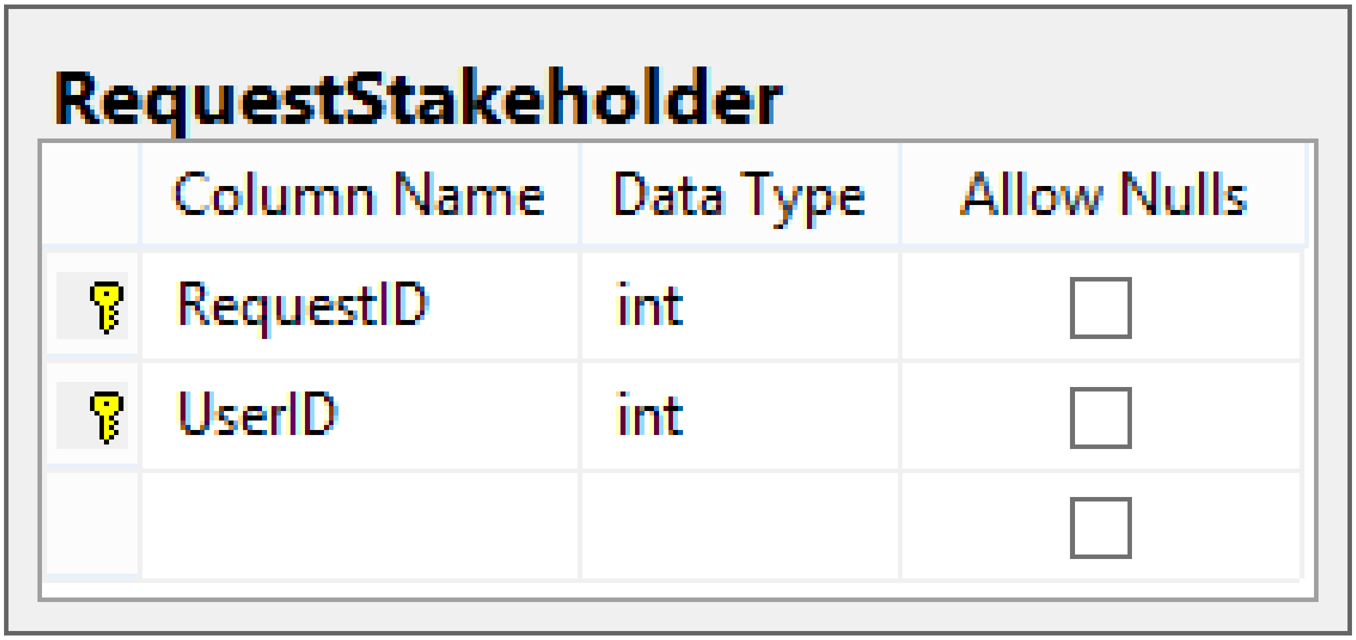
With Request and RequestData defined, our complete design (including the tables from Part 2) looks like this:



We still need three additional tables to round out what comprises a Request. First up is Stakeholders.

Stakeholders

A **Stakeholder** is a person that should receive periodic updates on the status of a given Request. Essentially they are Users who have a stake in the Request's outcome. What this means is simply that there is a many-to-many relationship between User and Request, so that table looks like this:

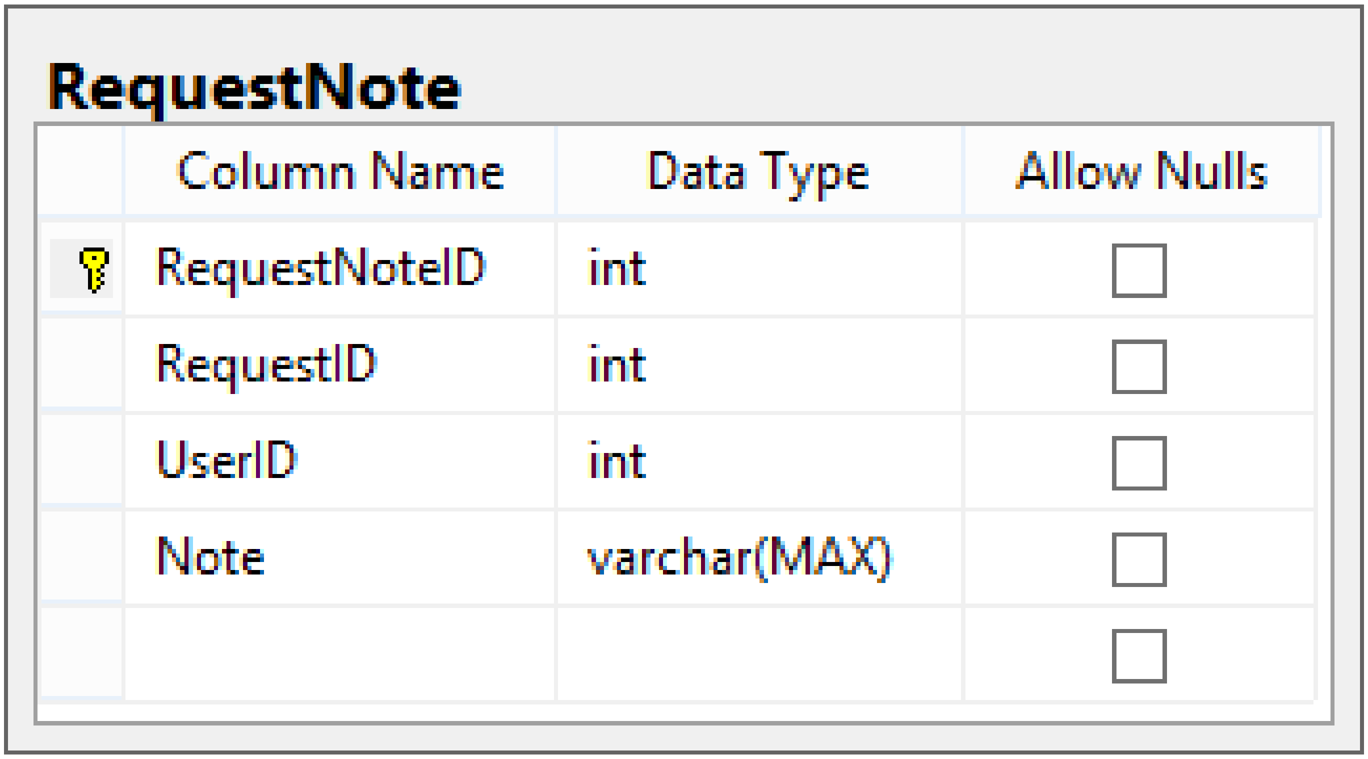


RequestStakeholder

	Column Name	Data Type	Allow Nulls
🔑	RequestID	int	☐
🔑	UserID	int	☐
			☐

Notes and Files

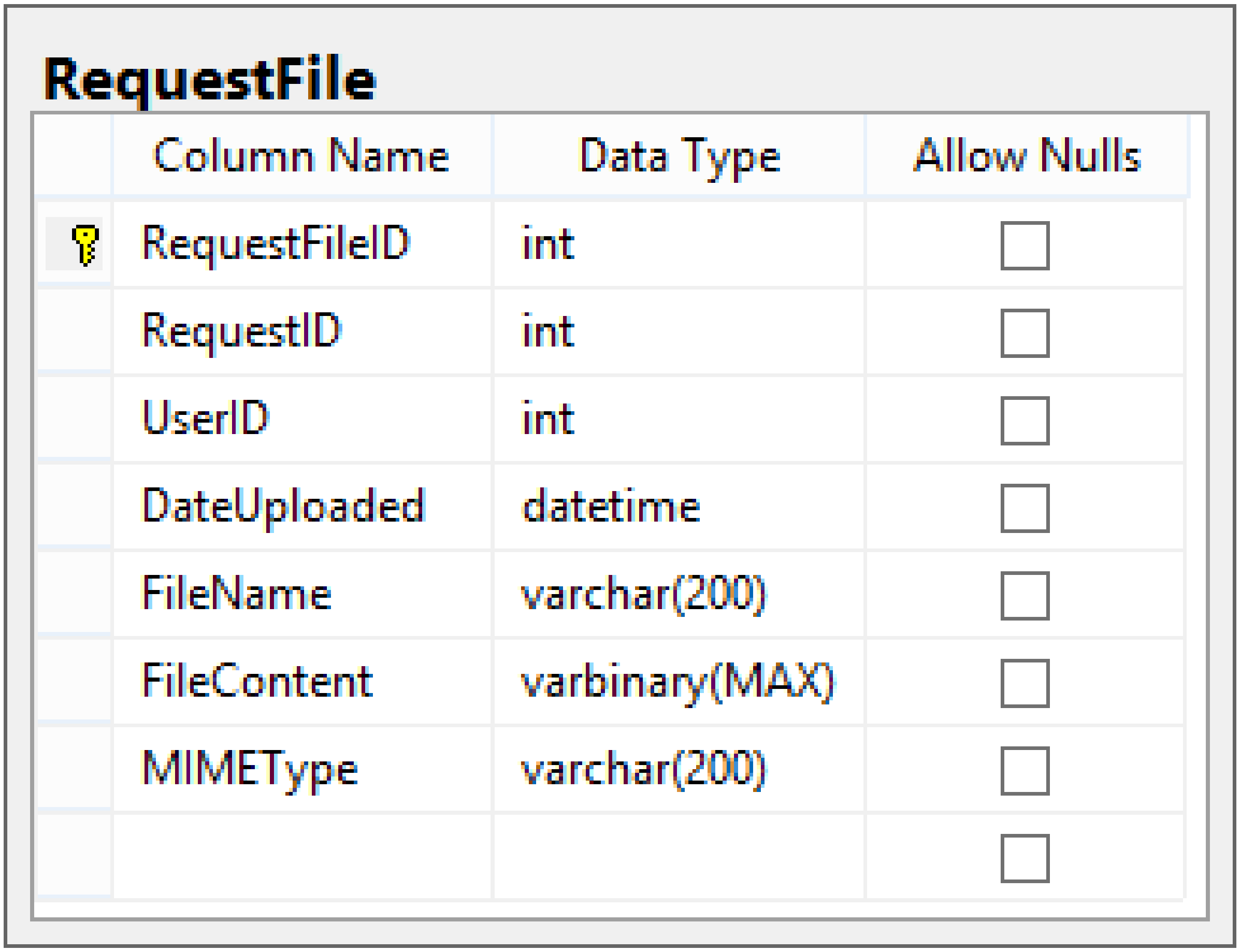
We want to allow for Users to add Notes to a Request, such that they can record additional information not contained in Request Data or the States. Our Notes table looks like this:



RequestNote

	Column Name	Data Type	Allow Nulls
🔑	RequestNoteID	int	☐
	RequestID	int	☐
	UserID	int	☐
	Note	varchar(MAX)	☐
			☐

Finally, we want to allow users to upload Files that are related to Requests directly into this database (a more extensible solution may be to store files on a file server somewhere, but for simplicity's sake we will store files in the database for this design). We need to store a file name, MIME type, and the actual content of the file in this table. The RequestFile table looks like this:

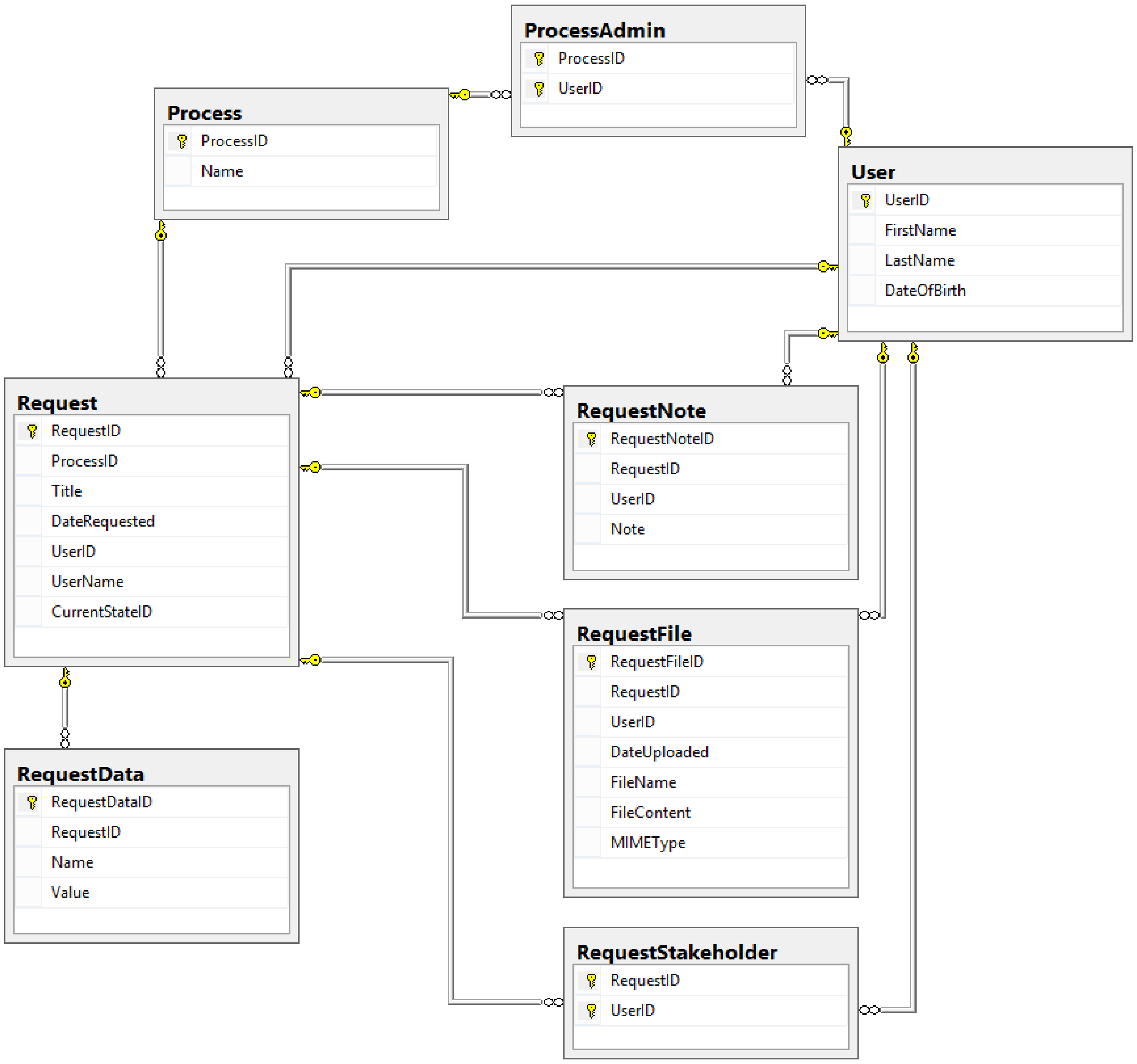


RequestFile

	Column Name	Data Type	Allow Nulls
🔑	RequestFileID	int	☐
	RequestID	int	☐
	UserID	int	☐
	DateUploaded	datetime	☐
	FileName	varchar(200)	☐
	FileContent	varbinary(MAX)	☐
	MIMEType	varchar(200)	☐
			☐

What did we accomplish?

After implementing the Request information, our complete design looks like this:



By completing the Request tables, we have given form to the data that is actually going to go through the approval processes defined in this database. But what exactly are these approval processes, and how can we define them? In Part 4 of this series, we will start designing how the Process itself is structured by designing tables for **States and Transitions**.