

## Re: Many-to-many relationships

---

*Source:*

<http://www.tech-archive.net/Archive/Access/microsoft.public.access.gettingstarted/2005-04/msg00192.html>

---

- *From:* "Ed Warren" <[ewarren@xxxxxxxxxxxxxxxx](mailto:ewarren@xxxxxxxxxxxxxxxx)>
  - *Date:* Mon, 4 Apr 2005 13:43:57 -0500
- 

I yield (this is becoming a number of angels dancing on a pin-head discussion).

Please help Al, as he proceeds to grope with the M:M relationships.

Ed Warren.

"BruceM" <[BruceM@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:BruceM@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)> wrote in message [news:873CC145-001E-4746-ABA5-05C0A117B109@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:873CC145-001E-4746-ABA5-05C0A117B109@xxxxxxxxxxxxxxxxxxxxxxxx)

- > An Access primary key is unique. That is what makes it a primary key.
- > Regarding your point about an extra field adding considerably to the size
- > of
- > a large database project, for purposes of this forum I tend to assume that
- > people are working on relatively small projects. Somebody developing a
- > database to store terabytes of information or hundreds of millions of
- > records
- > is probably past the point of needing to ask about many-to-many
- > relationships.
- > I have noticed there is a long-running debate about a separate PK field
- > rather than a "natural" PK based on data in the record. All I will say on
- > that topic is that either approach is probably valid. I find that
- > autonumbers (or such things as Employee IDs or invoice numbers) work well,
- > have the advantage of not leaving me to wonder whether I have satisfied
- > the
- > "uniqueness" requirement, and are simple to implement.
- > Presumably each record in the junction table would contain additional
- > information such as year, model, color, etc., they are not really
- > repeating
- > records. A family with two children could well contain repeating
- > information
- > in some fields of a Children table, but that doesn't mean they are
- > repeating
- > records.
- > One other point has to do with cascading deletes. If you remove a person
- > from the database (I would be inclined to use a query to filter them out
- > of
- > the recordset rather than deleting their records) then it makes sense to
- > also

## Re: Many-to-many relationships

> eliminate related records from the junction table. However, if you delete  
> a  
> record from the vehicle table and have cascade delete set up you would be  
> eliminating from the database that anybody had ever owned that type of  
> vehicle. That may not be what you intended.  
> I should probably study more about database theory so that I can jump into  
> discussions about first normal form and all that, but for now I am at the  
> point of asking "What if somebody owns two Fords?". Please don't get me  
> wrong. I think it's great that you have put so much time and thought into  
> your responses. That I can navigate databases now is due in large part to  
> people like yourself who were generous with their time and knowledge when  
> I  
> was getting started. I am just suggesting alternative thoughts on some  
> particular points. It is meant as a discussion, not a criticism or  
> anything  
> negative. I sincerely hope the spirit of my remarks is coming across as  
> intended.  
> "Ed Warren" wrote:  
>  
>> Sometimes one must keep things simple. (Required in this case to try to  
>> explain M:M relationships.  
>>  
>> I come from a bit not stored is a bit saved background.  
>>  
>> Not saving an additional field saves a few bits or couple of Bytes and  
>> over  
>> a large database them little bits/bytes can add up in search time and  
>> file  
>> size. They can get important in designs storing a several of terabytes of  
>> data with a few 100 million records.  
>>  
>> Yes I could add a unique Key field, and that would be appropriate if what  
>> I  
>> wanted was to know how many jeeps one owned, but then I may want to add a  
>> field to track the year of each, and/or I may want to put the number of  
>> jeeps in another field e.g.  
>> or maybe my VehicleType includes the model/year or ..... or.....  
>> or.....  
>>  
>> PeopleID |VehicleTypeID| Year|numberowned  
>> or  
>> PeopleID |VehicleTypeID| numberowned  
>> or  
>> UniqueKey |PeopleID| VehicleTypeID  
>> or  
>> .....  
>>  
>> Using a 'Unique Primary Key' is appropriate when it is required and  
>> inappropriate when not.  
>>  
>> I'm no expert with regard to the theory of database normalization, but

Re: Many-to-many relationships

Re: Many-to-many relationships

>> best  
>> I can recall you must remove any 'repeating' rows, to get to third normal  
>> form.  
>> Your design would produce 'repeating rows', and while convenient and  
>> simple  
>> this deviates from the strict standard. When required, I certainly do  
>> this  
>> rather than build yet another table to tie stuff together).  
>>  
>> E.g.  
>> Key PeopleID VehicleID  
>> 1 1 2  
>> 2 1 2  
>> 3 1 2  
>> 4 1 3  
>>  
>> Note rows 1,2,3 are repeating rows.  
>>  
>>  
>> See all this is already too much for my feeble mind to grasp, gotta go  
>> have  
>> another cuppa coffee to wake up! ;>  
>>  
>>  
>> Ed Warren.  
>>  
>>  
>>  
>>  
>> "BruceM" <BruceM@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
>> [news:A68C022F-DD70-446A-98D1-BC3B8966BA03@xxxxxxxxxxxxxxxxxxxx](mailto:news:A68C022F-DD70-446A-98D1-BC3B8966BA03@xxxxxxxxxxxxxxxxxxxx)  
>> > You would run into problems with that combined primary key if one  
>> > person  
>> > owns  
>> > two of a particular type of vehicle. Given that some people are very  
>> > dedicated to a particular type of vehicle it is certainly possible.  
>> > The  
>> > combined PK is not guaranteed to be unique, and therefore is a  
>> > questionable  
>> > choice as a PK.  
>> >  
>> > "Ed Warren" wrote:  
>> >  
>> >> Al  
>> >> It ain't all that hard, just sounds like it.  
>> >>  
>> >> Example  
>> >>  
>> >> You have Lots of People  
>> >> You have Lots of Vehicle Types  
>> >>

## Re: Many-to-many relationships

```
>>> Each 'people' can own many vehicletypes
>>> Each 'vehicletype' can be owned by many people.
>>>
>>> so we have People Many --> Many VehicleTypes
>>>
>>> To can't do M:M relationships directly, we can only do 1:M
>>> relationships
>>>
>>> So we build an new table (People_VehicleType)
>>> with two columns
>>>
>>> PeopleID VehicleTypeID We make the combination PeopleID;
>>> VehicleTypeID the Primary key
>>>
>>> We go to relationships and add the People, VehicleTypes, and
>>> People_VehicleType tables
>>>
>>> Relationship
>>> People --> People_VehicleType ( 1:M on PeopleID enforce relationship
>>> integrity, auto update, autodelete)
>>> Vehicletypes --> People_VehicleType (1:m on VehicleTypeID enforce
>>> relationship integrity, auto update, autodelete)
>>>
>>> Now when you delete a people their related records are deleted from
>>> the
>>> table People_VehicleType
>>> ditto for the VehicleType
>>>
>>> Form Setup:
>>>
>>> Forms:
>>>
>>> frmPeople (based on the People Table), display a single form
>>> frmVehicleTypes (based on the the VehicleType table), display a single
>>> VehicleType
>>> frm People_VehicleTypes (continious records)
>>> Two ComboBox fields
>>> 1. cboPeople (bound to PeopleID) (lookup data from query
>>> based
>>> on People with col1(id), col2 Display:[LastName] & ", " & [FirstName]
>>> &
>>> " "
>>> & [Address], bound column: 1, Number of columns: 2, Column Widths:
>>> "0";"2")
>>>
>>> This should give you a comboBox bound to the PeopleID but displays the
>>> Person's last name, first name, and address in the window for lookup.
>>>
>>>
>>> 2. cboVehicleType (bound to VehicleTypeID)
>>> (lookup data from query based on People with col1(id), col2
```

Re: Many-to-many relationships

>>> Display:[VehicleTypeDescription] ,bound column: 1, Number of columns:  
>>> 2,  
>>> Column Widths: "0";"2")  
>>>  
>>> This should give you a comboBox bound to the VehicleTypeID but  
>>> displays  
>>> the  
>>> Description of the vehicle in the window.  
>>>  
>>> Almost there:  
>>>  
>>> Now decide if you are going to enter vehicleTypes for people or People  
>>> for  
>>> vehicleType or either  
>>>  
>>> To handle vehicle type for people:  
>>>  
>>> Open People form  
>>>  
>>> Add frm People\_VehicleTypes as a subform: Parent (PeopleID) child  
>>> (PeopleID)  
>>>  
>>> Now for a selected person you can add the types of vehicles they own  
>>>  
>>> Hope this helps  
>>>  
>>> Ed Warren  
>>>  
>>>  
>>> "Al Williams" <AlWilliams@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
>>> [news:80A7FE8B-7D2F-4D66-B99D-C7B67E437F54@xxxxxxxxxxxxxxxxxxxx](mailto:news:80A7FE8B-7D2F-4D66-B99D-C7B67E437F54@xxxxxxxxxxxxxxxxxxxx)  
>>> > Access 2002: I need to read a really good discussion, with  
>>> > examples,  
>>> > about  
>>> > implementing many-to-many relationships. What issues need to be  
>>> > addressed  
>>> > when setting up table relationships – referential integrity and  
>>> > cascaded  
>>> > updates. How compound foreign keys are updated by Access – both  
>>> > automatically and manually (if possible). What queries result in  
>>> > updateable  
>>> > fields and what don't. How to setup forms to easily create, edit  
>>> > and  
>>> > view  
>>> > many-to-many relationships. Preferably, more than one way to do the  
>>> > implementation would be discussed. I've spent a lot of time  
>>> > thinking  
>>> > and  
>>> > trying to read about how to do it but haven't found enough  
>>> > information  
>>> > to

