This is a response to a question from **dzhu** on StackOverflow:

How to reference groups of records in relational databases

Response to Update 2

> AccessInfo has two columns: SpecID and AccessibleTo.
>
> The value of AccessibleTo is a CSV composed of UserIDs and GroupIDs.

CSVs break Second Normal Form, we don't allow that because it leads to horrendous code, for insertion, update, ordering, etc. Of course the main problem is duplication and lack of Integrity over the components in the list.

> And the CSV may contain hardcoded strings like 'Author'.

That is a second or compounded form, a second breach of 2NF. The imbeciles have dipped their toes into EAV.

In order for me to complete the model, please provide a list of all such tag-value pairs.

> This is a simplified version of an application I have seen. The actual security logic is far more complicated than this.

It might be better if you can provide all the requirements, then I can provide a more detailed model, and you will learn more.

- Note that it is a famous trick that the "teachers", the frauds (obviously not you or your team, you are the victims of the fraud) use, they provide a simple model to prove or disprove a proposal. ***Anything***, right or wrong, truth or falsity, can be proved or disproved using a simple example. They use this technique to attack the truth, and to justify insanity.
- That is why I always give full detail, and my case study models have 20 or more tables with complex relationships.

  > I am not trying to change the table structures (the senior developers won't take such a risky advice anyway, they are very careful about changing existing code)

That is because they are senior imbeciles, totally absent of the education that is required for the job. They do not realise that a Relational database would make all their "code" obsolete.

Frauds and imbeciles stress the fear factor, honest technicians stress the truth factor. You are seeking truth, hence you will learn much, things that they are not open to learning.

- First they have implemented spreadsheets as "tables". Spreadsheets, *by definition*, are the flattened **view** of Normalised tables, what the developer looks at when they are obsessed with the data *values*. They are *not* tables. That perspective, and their slavish habit of stamping an `ID` column on every spreadsheet, before the tables and keys have been determined, guarantees that the data will *not* be modelled, and that a massive Record Filing System will be implemented. No Relational Integrity, power or speed.
- Second, as a consequence, they have to implement massive and complex code. And they will not realise that the entire lot of it is garbage, necessary only because they do no thave a Relational Database.
- They don't even know the cause of their horrendous cause, that it is their own mistakes in their primitive pre-1970's ISAM record Filing System.
- Once the data is Normalised, in Relational form, we obtain full Relational integrity, power, and speed, and the code required is simple.

  > The performance suffers when retrieving a large number of specs (not large really, hundreds of specs) with security enabled.

Yes, the heap of garbage smells, and the more load you put on it, the more it smells. That (exponential degradation per linear load) is actually a classic indicator of a non-relational filing system. Whereas with a Relational system, performance maintains a linear curve as the population grows.

Record Filing Systems are at least two orders of magnitude slower than the equivalent Relational database. Of course, if it is a bad RFS, it will be even slower. A CSV in a single field (not "column", please, RDbs don't have CSVs, RFSs do) that has to be searched with:

```
AccessibleTo LIKE "%foo%"
```

will cause a tablescan.

And then they have to search multiple times, once for each tag-value pair.

If they can code Dynamic SQL, fine, they can search the string once, one tablescan, otherwise, they will code multiple searches, multiple tablescans. I am not categorising them as imbeciles without reason.

> How can we design a Relational Model based on these requirements? I just want to know how to develop a sound database structure if I were to develop it from scratch. I want to know the right way to build high-quality software.

Sorry, but I can't give you a tutorial on Relational Database Design. Especially not on SO. You will have to please deal with one question and answer at a time.

- If you find my posts of value, you can learn a lot from my other Answers. Go to my Profile page, look for any database Question that interest you, and read my Answers.

Solution

Based on the details given, here is a Data Model. Note that there are questions outstanding, this is something to get you started, as a discussion platform.

- There is one redundancy, which I will remove when details are clarified.
- Likewise, I will provide the Predicates when the questions are closed.

Specification 1 Data Model

In order for you to fully understand the solution given, you need to understand IDEF1X data models and their Notation. Every line; circle; notch; tick; crows foot; the square vs round corners; the solid vs dashed lines, means something very specific.

IDEF1X Notation

Predicate

If you would like to understand this important part of Data Modelling exercise, please visit **this Answer**, scroll down to Predicate, and read that section.