

Account Number: A Pattern

William C. Wake, Virginia Tech
wakew@cs.vt.edu
October 31, 1994

Pattern

If:

- Information (an Account) is associated with a Person.
- Two different Persons might have the same Name.
- Traversal is commonly from Name to Account.

Then:

Assign each Person a unique Account Number.

Problem

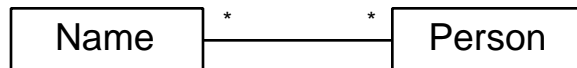
A single name may correspond to many people. A person has many names they may use, depending on the situation. We want to examine or update Account information for a particular Person. Given their name, how do we find the corresponding Account object?

Forces

- We often want to go from a person's name to the corresponding Account. For example, when you go to a doctor's office, they'll ask for your name so they can locate your medical records.
- Many people share the same name. Almost any American phone book will list more than one John Smith.
- A person may have many valid names. For example:
 - formal name on legal documents,
 - everyday name,
 - maiden name,
 - nickname,
 - pen name,
 - stage name,
 - and many others.

Depending on the situation, a person may use any of these.

- Thus, names and people are in a many-to-many relationship:

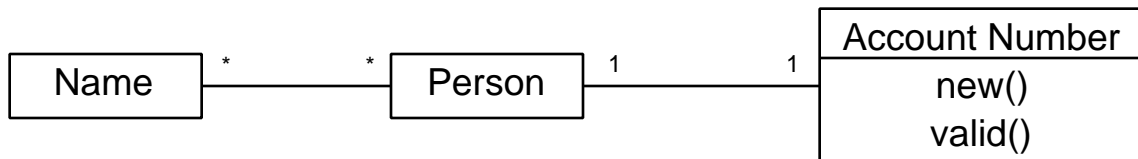


- Not only are names insufficient to identify a person, no other combination of attributes of the Person object is guaranteed unique. For example, two people named John Smith might have the same birthday, live at the same address, and share a telephone. Even where attributes sufficient for identification do exist, it may be socially infeasible or too expensive to use them: people are not willing to have their DNA sampled just to obtain a library card.

Solution

Rather than depending on the name or other attributes, we introduce an artificial name: an account number.

Assign exactly one account number to each person, and assign each account number at most once.



- **Name:** the object encapsulating a name.
- **Person:** the object representing a person. In addition to being related to a set of Names, the Person object is in one-to-one correspondence to an AccountNumber object.
- **AccountNumber:** the generated account number. The AccountNumber class must be able to issue new numbers, and validate whether a purported AccountNumber is in fact a valid one.

Collaborations

- Given a name, we can identify a set of “matching” people. If there is more than one person in that set, we can use the Account Number (and possibly other attributes of the Persons) to select among them.
- Alternatively, given a valid Account Number, there is exactly one corresponding person, and we can verify the name through the Person object.

Implementation

- *Numbers or not?* It’s not necessary that account numbers be strictly numeric.
- *Fixed- or variable-length?* Account numbers are often of fixed length. This is convenient for software (and hardware!) that must work with the numbers. However, it limits expansion, and makes the choice of “how many digits” an important up-front decision. Some numbering

schemes don't have this restriction. For example, the ISO object identifier numbering scheme [Rose] uses a hierarchical numbering scheme of essentially unlimited depth.

- *Unstructured or structured numbers?* One way to issue account numbers is to have a central issuer maintain a counter, starting at 1 (or some other convenient number), and issuing successive numbers. This is an unstructured number: there is no relationship between the number and the account or the issuer.

An alternative is to develop structured numbers. One or more digits are reserved for particular issuers or attributes of the account. Each issuer is responsible for a certain range of numbers. For example, credit card numbers are 16 digits long; the first digit indicates the type of credit card (say MasterCard or Visa). This lets each issuer choose their numbers independently. It has the disadvantage that the account numbers will tend to be larger to enable easy encoding of the fields.

- *How many digits?* If an account number has a fixed length, it should accommodate the maximum potential number of accounts. Particularly with fixed-length structured numbers, you should try to ensure that one group won't run out while another has numbers to spare.

Consider the case of (United States) telephone numbers [Townson]. (This is an example of assigning numbers, but not an example of the Account Number pattern.) Telephone numbers are structured as a 3-digit area code and a 7-digit local number. In principle, this allows 10 billion numbers, and that would seem sufficient: enough to give every person in the world two phone numbers. However, because of the way numbers are assigned, only about a billion numbers are actually available. (Plans are in place to expand this to about 6 billion numbers.)

Unfortunately, there are only 10 million numbers within each area code. Large metropolitan areas approach this many occupants. These areas run out of numbers, while smaller areas may have numbers to spare. Because area codes are assigned to geographic regions, a region with extras can't share them with a region that needs them.

Things get worse: many people need more than one phone number (perhaps one for home and one for work). The phone system was originally designed for people, but fax machines and computers need telephone numbers as well. Companies need phone numbers not tied to any particular person.

Thus, demand for telephone numbers has outstripped the supply. Ten billion numbers seems like a lot, until you start restricting them, structuring them, and providing multiple numbers for each person. Changing the numbering scheme has been expensive and time-consuming.

Variations

- *One name per person.* We're often not interested in recording all the names for a person; a single "formal" name may be sufficient. This will make Name:Person a one-to-many relationship. This reduces storage and management requirements, but doesn't affect the fundamental problem of shared names.

When restricting people to one name each, we may let someone change their name, but will not normally issue a new Account Number.

- *Attributes of name relationships.* The Name:Person relationship can be very complicated: the relation may have attributes indicating when the name is valid, the type of name, and so on.
- *Relationships between names.* This pattern has described the situation where the various names are related to a person, but hasn't considered names' relationships to other names. Thus, while we might recognize *Jack* as a possible nickname for a person named *John*, not every *John* will use that nickname. However, in some situations we are more interested in names themselves. If we were developing a database of possibly related names, *John* and *Jack* might be linked together independently of any particular person.

Examples

- *Accounting systems for businesses.* Most business account systems (for things like utilities, credit cards, etc.) have used the technique of issuing account numbers for people rather than trying to locate their record based solely on name.
- *Keys in databases.* In a database, each record is identified by a key unique to that record: a combination of one or more attributes of the record. The non-key attributes correspond to Name, the record itself to Person, and the key to AccountNumber.
- *Social Security Number.* In the United States, most citizens have a Social Security Number, issued by the government. No one may legally have more than one number. However, this identification number is susceptible to misuse—either by intentional fraud, or by unintentional transposition or copying errors. There is no check digit in the coding of these numbers, so almost any 9-digit number is potentially valid. [Hibbert]
- *International Standard Book Number.* Many publishers use ISBNs to identify their books, as titles aren't unique. This is a 10-digit number, the last digit of which is a check digit [Hamming, pp. 33-34]. (The check digit allows one to detect transpositions of adjacent digits, single substitutions, and several other errors.) Here, the title corresponds to Name, the book to Person, and the ISBN to AccountNumber.

References

[Hamming] Hamming, Richard W. *Coding and Information Theory*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

[Hibbert] Hibbert, Chris. "Social Security Number FAQ", posted to Usenet newsgroup *news.answers*, May 3, 1994.

[Rose] Rose, Marshall T. *The Open Book*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.

[Townson] Townson, Pat, et al. "How numbers are assigned", March 28, 1990, in *ftp://lcs.mit.edu/telecom-archives/areacodes/how.numbers.are.assigned*.

Acknowledgements

I'd like to acknowledge discussions with and comments from: my advisor, Dr. Ed Fox (Virginia Tech); my brothers, Steve Wake (MSL, Blacksburg, VA) and Doug Wake (University of Hawaii); and the conference reviewers and participants. This work took place while the author was partially supported by National Science Foundation Grant #IRI-9116991.