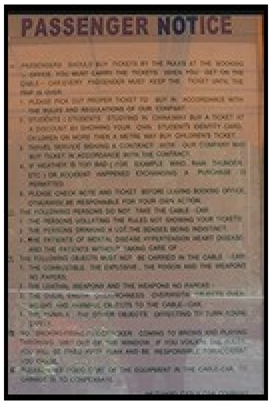
I'm OK, You're Abnormal

7 min. read · View original



PASSENGER NOTICE

[illegible]

Normalization is a way of splitting up data until each table represents propositions about a single type of thing. Normalization is a beloved subject in academic courses on database management systems and in job interviews because (1) it has a patina of formalism, and (2) it is easy to test. If you tell people that you took a DBMS course and then look blank when they ask you whether your data are in Third Normal Form, you'll never get any respect from database nerds. A lot of the ideas will seem like common sense and much of the debate is held over from the 1960s before the RDBMS was conceived and grew to dominate data storage. However, the mid-1990s object-relational database fad has revived some of these issues and it is therefore worth examining normalization.

Abnormal

Suppose that you're determined to marry Winona Ryder, the movie actress who was born "Winona Horowitz" on October 29, 1971. Thanks to the convergence of Hollywood and Silicon Valley, you've managed to get a seat at the Academy Awards ceremony. You'll want to be well-informed about Ms. Ryder's work while sounding casual. You decide to make a little database to coach you before the awards night and to serve the same data on demand to your WAP phone in case you get stumped in conversation with Ms. Ryder.

Movie ID	Title	Year Released	Producer	Director	Writer	Composer	Steadicam Operator
1	Alien: Resurrection	1997	Bill Badalato	Jean-Pierre Jeunet	Joss Whedon	John Frizzell	David Emmerichs
2	The Age of Innocence	1993	Barbara De Fina	Martin Scorsese	Edith Wharton, Jay Cocks, and Martin Scorsese	Elmer Bernstein	Larry McConkey
3	Heathers	1989	Denise Di Novi	Michael Lehmann	Daniel Waters	David Newman	--

If you memorize this material, you can attract Ms. Ryder's attention with such questions as "Didn't you think Larry McConkey's Steadicam work in *The Age of Innocence* was much better than David Emmerichs's in *Alien 4*?" Or if you want to impress with your knowledge of American literature, say "It is amazing how rich a script Jay Cocks and Martin Scorsese produced considering the thinness of the material they had to start with from Edith Wharton's original novella."

Suppose that Winona Ryder is sufficiently impressed by your knowledge to come back to your suite at the Regent Beverly Wilshire. Taking advantage of the in-room Ethernet connection, you've left a remote Emacs window up and running on your laptop. Ms. Ryder sees your SQL queries in the source code for your coaching app and shouts out "Hey, these data aren't even in First Normal Form (1NF)."

My this is a faux pas. If your data aren't in First Normal Form, there isn't even an egghead DBMS nerd term for what you've got. Loosely speaking, your data are *abnormal*. What is wrong with your data model? For *Alien* and *Heathers* you've got a single name in the `Writer` column. For *The Age of Innocence*, you've got a trio of names. This is known as a *repeating group* or a *multivalued column* and it has the following problems:

- you might not have enough space if the number of values in the column grows larger than anticipated
- the combination of table name, column name, and key value no longer specifies a datum
- the basic INSERT, UPDATE, and SELECT operations are not sufficient to manipulate multivalued columns
- programmers' brains will have to adapt simultaneously to unordered data in table rows and ordered data inside a multivalued column
- design opacity. If you use multivalued columns even once, people will never know what to expect when they look under the hood of your design; did you use multiple tables to express a many-to-one relation or multivalued columns?

Do these problems mean that multivalued columns are useless? Probably not. Oracle introduced *two* kinds of support for multivalued columns with the 8.0 release of their server:

```
For modelling one-to-many relationships, Oracle
supports two collection
datatypes: varrays and nested tables. For example,
a purchase order has
an arbitrary number of line items, so you may want
to put the line items
into a collection.
```

```
   Varrays have a maximum number of elements, 
although you can change
   the upper bound. The order of elements is 
defined. Varrays are stored
   as opaque objects (that is, raw or BLOB).

   Nested tables can have any number of elements, 
and you can select,
   insert, delete, and so on the same as with 
regular tables. The order
   of the elements is not defined. Nested tables 
are stored in a storage
   table with every element mapping to a row in 
the storage table.

If you need to loop through the elements in order, 
store only a fixed
number of items, or retrieve and manipulate the 
entire collection as a
value, then use varrays.

If you need to run efficient queries on 
collections, handle arbitrary
numbers of elements, or do mass 
insert/update/delete operations, then
use nested tables. If the collections are very 
large and you want to
retrieve only subsets, you can model the 
collection as a nested table
and retrieve a locator for the result set.

For example, a purchase order object may have a 
nested table of line
items, while a rectangle object may contain a 
varray with 4 coordinates.
```

With the nested tables option, Oracle is simply doing what an RDBMS purist would have told you to do in the first place: use multiple tables to represent many-to-one relations.

First Normal Form

If you'd seen *The Crucible*, where Winona Ryder plays Abigail Wiliams, you'd have remembered the following scene:

> **Abigail:** I am but God's finger, John. If he would condemn Elizabeth, she will be condemned. Arguments against normalization continue to sway practitioners. Fabian Pascal says that this costs dearly and reveals the poor understanding of sound database principles by even those who profess to be experts in the field. It is both a major reason for and a consequence of deficiencies in SQL implementations and for technology regressions, such as ODBMS and OLAP, that have come to haunt SQL DBMSs in the town of Salem.

Before leaving your hotel room, Ms. Ryder recasts your movie database in First Normal Form:

Movie Facts

Movie ID	Title	Year Released	Producer	Director	Composer	Steadicam Operator
1	Alien: Resurrection	1997	Bill Badalato	Jean-Pierre Jeunet	John Frizzell	David Emmerichs
2	The Age of Innocence	1993	Barbara De Fina	Martin Scorsese	Elmer Bernstein	Larry McConkey
3	Heathers	1989	Denise Di Novi	Michael Lehmann	David Newman	--

Movie Writers

Movie ID	Writer
1	Joss Whedon
2	Edith Wharton
2	Jay Cocks
2	Martin Scorsese
3	Daniel Waters

Event Registration

Having struck out with Winona Ryder due to the abnormal nature of your data model, you decide to go on a nationwide speaking tour. You're going to give three different talks:

- How to pick up Jewish babes in Hollywood
- Using your SQL programming expertise to get a seat at the Academy Awards

- What I learned about normalization from Winona Ryder

Each of these three talks will be given about 20 times in different cities. We'll call a particular occurrence of a talk an *event*. Rather than hire a staff to process registration for each lecture, you build a database-backed Web application:

```
-- one row for every different talk that we give

create table talks (
        talk_id         integer primary key,
        talk_title      varchar(100) not null,
        description     varchar(4000),
        speaker_name    varchar(100),
        speaker_bio     varchar(4000)
);

-- one row for every time that we give a talk

create table events (
       event_id         integer primary key,
       -- which of the three talks will we give
       talk_id          references talks,
       -- Location
       venue_name       varchar(200) not null,
       street_address   varchar(200) not null,
       city             varchar(100) not null,
       -- state if this is in the US
       usps_abbrev      char(2),
       -- country code is this is a foreign city
       iso              char(2) default 'us',
       -- Date and time
       start_time       date not null,
       end_time         date not null,
       ticket_price     number
);
```

This data model is in First Normal Form. There are no multivalued columns. However, it has some deficiencies. Suppose that you fly into New York City and give each of your three talks over three days. Each time you're speaking at the same venue: Radio City Music Hall. Because of the way that the `events` table is designed, you'll be recording the street address of Radio City Music Hall three times. If the street address were to change, you'd have to change it in three places. If you're contemplating using a new venue and want to enter the street address, city, and country code or state abbrevation, you've nowhere to store the information unless you've already got an event scheduled for a specific time. If there is only one event in the database scheduled for a particular venue, deleting that event also deletes all information for the venue.

Second Normal Form

If all columns are functionally dependent on the *whole key*, the data model is in second normal form. Less formally, a second normal form table is one that is in first normal form with a key that determines all non-key column values.

```
-- one row for every different talk that we give

create table talks (
        talk_id         integer primary key,
        talk_title      varchar(100) not null,
        description     varchar(4000),
        speaker_name    varchar(100),
        speaker_bio     varchar(4000)
);

-- one row for every place where we give a talk

create table venues (
       venue_id         integer primary key,
       venue_name       varchar(200) not null,
       street_address   varchar(200) not null,
       city             varchar(100) not null,
       -- state if this is in the US
       usps_abbrev      char(2),
       -- country code
       iso              char(2) default 'us'
);

-- one row for every time that we give a talk

create table events (
       event_id         integer primary key,
       -- which of the three talks will we give
```

```
        talk_id           references talks,
        -- Location
        venue_id          references venues,
        -- Date and time
        start_time        date not null,
        end_time          date not null,
        ticket_price      number
);
```

Note that any data model in second normal form is also in first normal form.

Third Normal Form

If all columns are *directly* dependent on the whole key, the data model is in third normal form. How is this different from second normal form? For example, suppose the price of the talk is a function of which talk is being given and the talk length. Thus the pair of `start_time` and `end_time` will determine the value of the `ticket_price` column. If we add one more table, our data model will be in third normal form:

```
create table ticket_price (
        up_to_n_minutes          integer,
        price                    number
);
```

Note that any data model in third normal form is also in second normal form.

Reference

Next: Afterword

---

mbooth@arsdigita.com
philg@mit.edu

Add a comment | Add a link