

INTENTIONALITY
IN
RELATIONAL DATABASE
DESIGN

By

Alan G. Labouseur

3NF CONSULTING, INC.

PREFACE

In 1957, Canadian philosopher and theologian Bernard Lonergan published his masterpiece, Insight: A Study of Human Understanding. In this work he described his epistemological system called Generalized Empirical Method (GEM). GEM asserts that conscious and intentional operations consist of four parts: experiencing, understanding, judging, and deciding. His algorithm for human knowledge and experience is therefore:

```
while not dead
{
    Experience();
    Understand();
    Judge();
    Decide();
    Act();
}
```

Lonergan gives four corresponding proscriptions for applying GEM to any discipline: be attentive, be intelligent, be reasonable, be responsible.

In 1972 Lonergan applied GEM to the topic of theology in his book, Method in Theology.

In this presentation I will take the first steps in applying GEM to relational database design as an a priori approach to data quality.

INTRODUCTION

Data is dangerous. By itself, data is also meaningless. Worse than that, being just a collection of letters and numbers, data can be interpreted in many ways, most of them probably wrong. What we really want is information.

Information is data given context.

The purpose of a database is to store data in such a way that we can derive information from it. In other words, we need to be able extract data from our database such that it has meaning. Meaning comes from context. (i.e., How does the data fit together?)

Context comes from the relationships among tables.

It is the relationships among tables that form the basis of our queries. If we use the relationships the database designer intended to provide contextual information then our results will be meaningful.

INTRODUCTION

If we query in ways that combine data outside of the intended relationships we are likely to get meaningless results. We'll have data, but not information.

No business can make decisions without information.

That's why data is dangerous. More specifically, data misinterpreted as information is dangerous. It's dangerous because business decisions will be made based on incorrect assumptions. Informationless data is worse than having no data at all. If you have no data, you will know that you don't know, and can make allowances for not knowing. Thinking that you do know something when that something is baseless is far worse.

Paying careful attention to the documented and implemented relationships among the tables in your database, and developing queries based on these relationships, will insure that your results will always be useful information rather than useless, and perhaps misleading, data.

INTRODUCTION

What can we do as data architects to insure, or at least encourage, correct and intended uses of the databases we design?

We need to make our intentions apparent to future users.

This “intentionality” can be expressed in several ways during the logical data modeling phase of database design. We have check constraints, default values, and keys of various types at our disposal. These are all important in forming the intentional basis for our design. But perhaps the most important tool for intentionality is referential integrity. It is through referential integrity that we intentionally imbue the data model with our design requirements and restrictions. That is to say that referential integrity reflects our a priori intentions after the fact.

What is “intentionality”?

WHAT IS “INTENTIONALITY”?

“Intentionality is intrinsic to all cognitive operations involved in the knowing process. It is that aspect of cognitive operations which reaches out to grasp reality and make it present to the subject...”

-William R. Eidle¹, based on writings by Bernard Lonergan²

“Intentionality...[leads us to a] logical method of common sense inference”

-William R. Eidle¹

1. Eidle, W. R. (1990) The Self-Appropriation of Interiority : a foundation for Psychology. New York. Peter Lang Publishing, Inc.

2. Lonergan, B. J. (1972) Method in Theology. New York. Seabury Press.

WHAT DOES INTENTIONALITY HAVE TO DO WITH RELATIONAL DATABASE DESIGN?

“Intentionality is intrinsic to all **cognitive operations** involved in the **knowing process**. It is that aspect of cognitive operations which reaches out to grasp **reality** and make it present to the **subject...**”

database user

queries

intended relationships

meaningful results

Applying this quote to the realm of relational database design, the subject is the database user. The cognitive operations are the queries. The knowing process is the meaningful result derived from the queries. *The reality that needs grasping is the relationship among the tables in the database.*

Making the real purpose and intended use of your database present to its users can be difficult. Whether they are application programmers, systems programmers, or end users, they may not be aware of the nature of the intended relationships. It is through relationships that the database designer *intentionally* made that the data can be given context. There are usually other relationships in the data, but exploiting them may not reveal information, since they are not intentional relationships and therefore provide no a priori context for the data.

WHAT DOES INTENTIONALITY HAVE TO DO WITH RELATIONAL DATABASE DESIGN?

“Intentionality...[leads us to a] logical method of common sense inference”

If database designers document and implement their intentional relationships among the tables, and if database users logically base their queries on those relationships – thus inferring what the designers considered the “common sense” basis for combining their table data – meaningful results will follow.

GOALS OF INTENTIONALITY

As a database designer

be <i>Attentive</i>	to the relationships you are trying to model
be <i>Intelligent</i>	about understanding the relationships and how to best model them
be <i>Reasonable</i>	when judging your design for accuracy and completeness
be <i>Responsible</i>	about implementing and documenting your intended relationships

As a database user

be <i>Attentive</i>	to the intended relationships among the tables
be <i>Intelligent</i>	about understanding the relationships and how to best use them
be <i>Reasonable</i>	when judging the quality and accuracy of your queries
be <i>Responsible</i>	for using the intentional relationships to write queries that result in information

OUR EXAMPLE: ADDRESSES AND ZIP CODES

Two tables:

Address

- PID, the unique “person ID” for that person – the primary key
- First Name
- Last Name
- Street Address
- Zip – the Zip code

ZipCode

- Zip – the primary key
- City
- State

ADDRESSES AND ZIP CODES, WITH SAMPLE DATA

Address table

PID	FirstName	LastName	StreetAddress	Zip
007	James	Bond	180 Treaty Road	12508
002	Bill	Fairbanks	10 Kirkbride Ave.	07642
006	Alex	Trevelyan	2476 Bolsover St.	27410
008	Lee	Pfeiffer	149 Juniper Way	12508
005	Dave	Worrall	52 Simon Block	94118

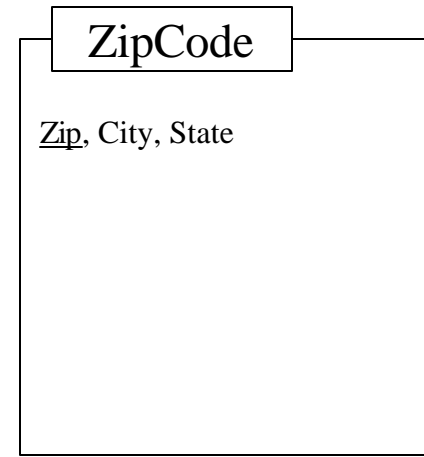
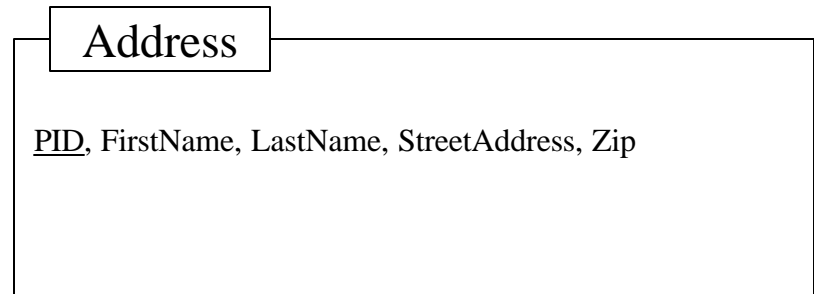
Primary Key is PID

ZipCode table

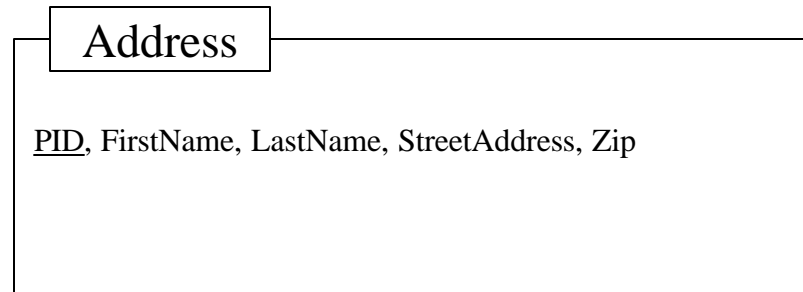
Zip	City	State
12508	Beacon	NY
07642	Hillsdale	NJ
05652	Eden	VY
77473	San Felipe	TX
27410	Greensboro	NC
18848	Towanda	PA
99701	Fairbanks	AK
94118	San Francisco	CA
86402	Kingman	AZ

Primary Key is Zip

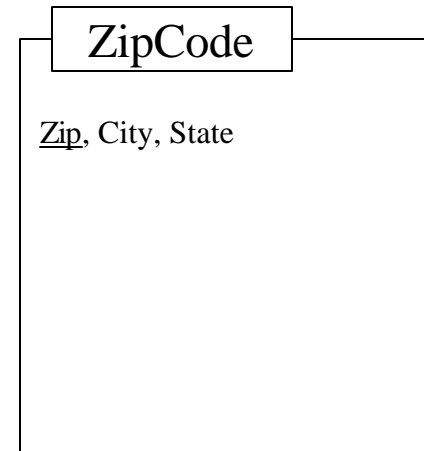
ADDRESSES AND ZIP CODES, AS E-R SYMBOLS



ADDRESSES AND ZIP CODES, CREATE STATEMENTS

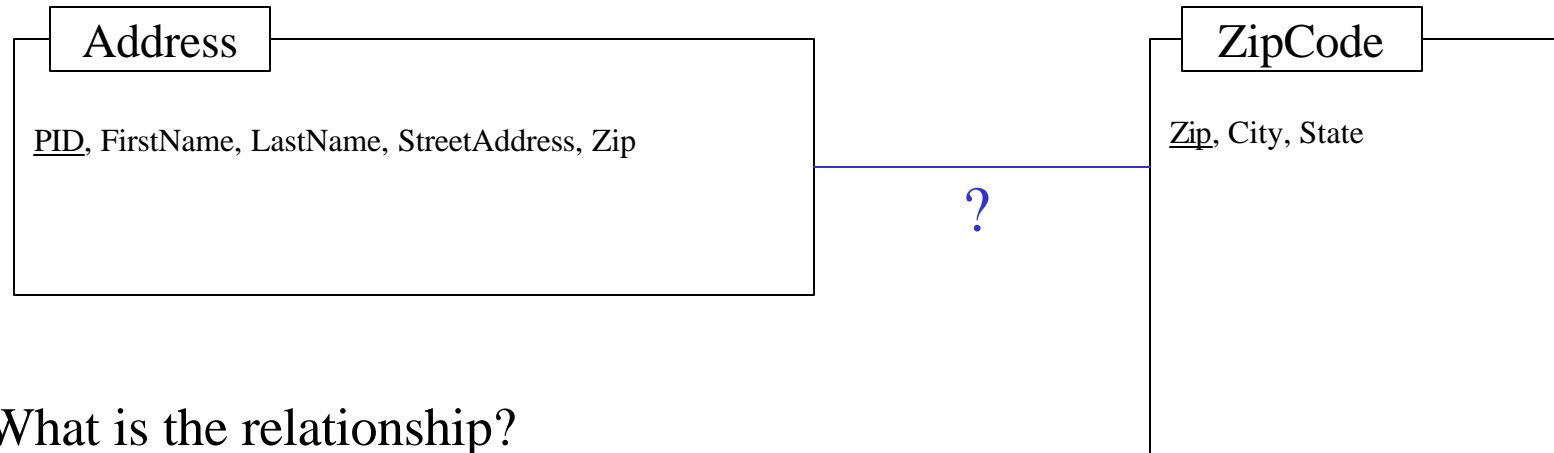


```
Create Table Address (  
  PID          char(3)      not null,  
  FirstName    varchar(30),  
  LastName     varchar(30) not null,  
  StreetAddress varchar(50),  
  Zip          char(5),  
  Primary Key(PID)  
)
```



```
Create Table ZipCode (  
  Zip  char(5)      not null,  
  City varchar(50) not null,  
  State char(2)     not null,  
  Primary Key(Zip)  
)
```

ADDRESSES AND ZIP CODES



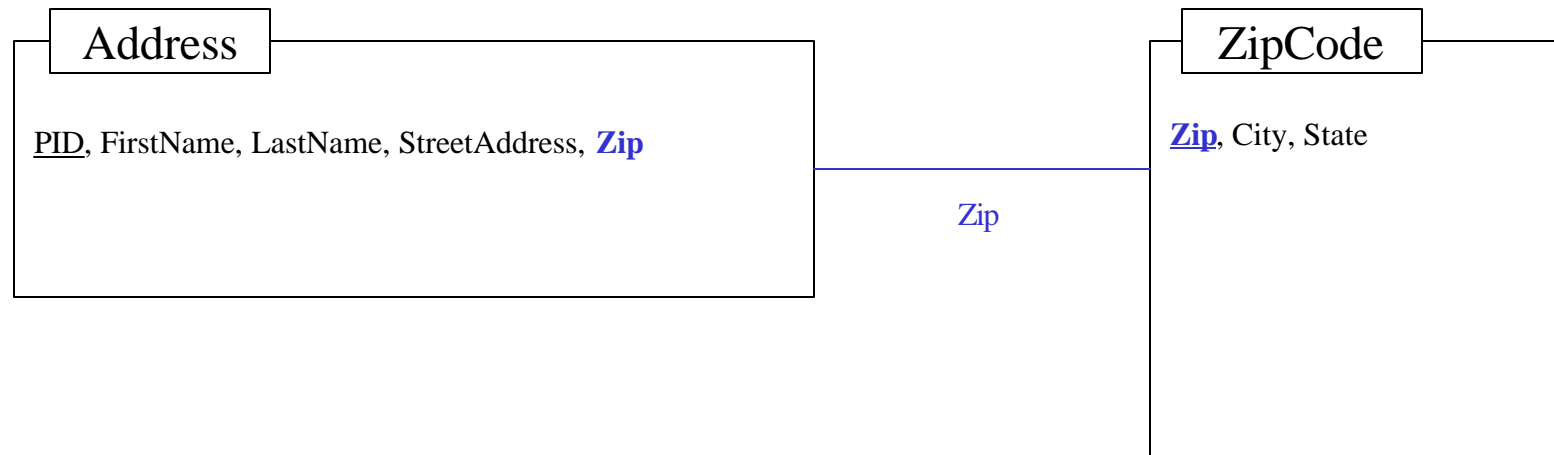
What is the relationship?

We intend to use the ZipCode table as a “lookup” to get city and state values associated with a particular address given that addresses’ zip code.

How do we define the relationship?

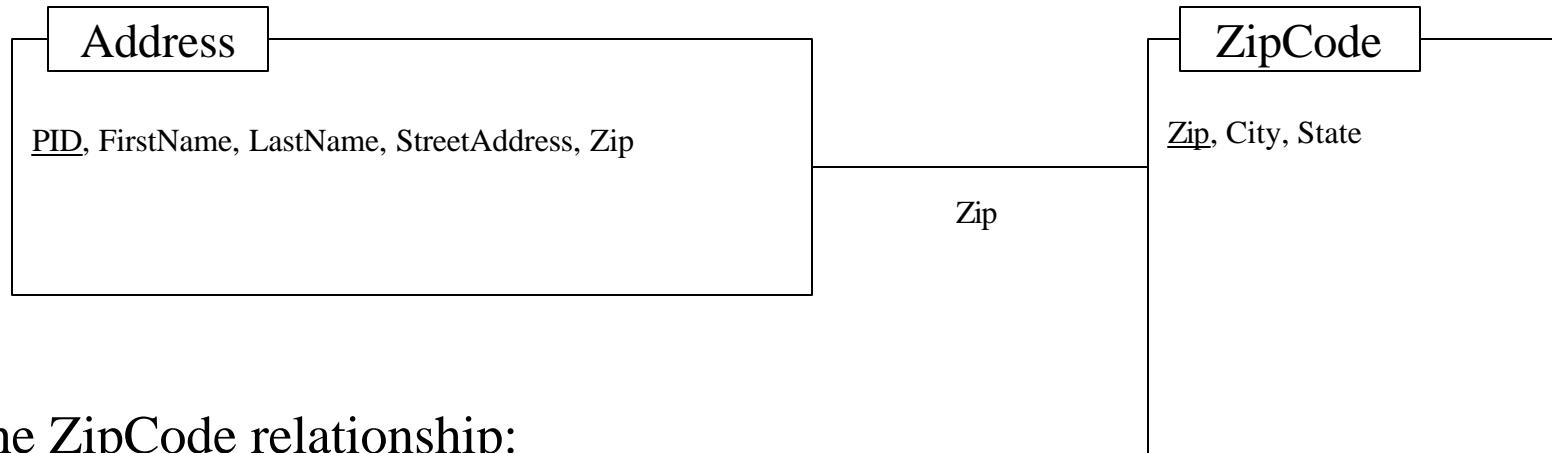
Since this is our intention, we need to implement it in the database. We do so by defining a foreign key relationship. This alerts the database engine to our intention and allows it to enforce this relationship. This also denotes our intention to the database users.

ADDRESSES AND ZIP CODES, REFERENTIAL INTEGRITY



```
Create Table Address (  
  PID          char(3)      not null,  
  FirstName    varchar(30),  
  LastName     varchar(30) not null,  
  StreetAddress varchar(50),  
  Zip          char(5)      references ZipCode(Zip),  
  Primary Key(PID)  
)
```

ADDRESSES AND ZIP CODES

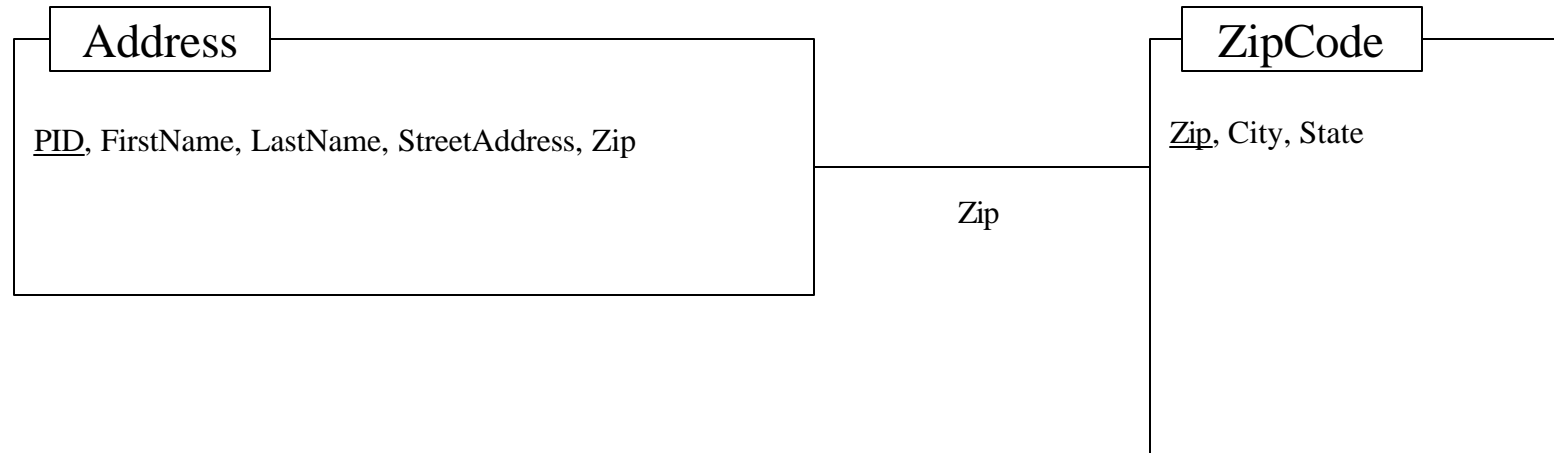


The ZipCode relationship:

Zip in the Address table is a foreign key to the Zip field in the ZipCode table. This means that there can be no Zip in the Address table that does not exist in the ZipCode table. If we try to insert a row into Address that contains a Zip not in the ZipCode table, that insert will fail. This is called referential integrity, and it keeps our data consistent.

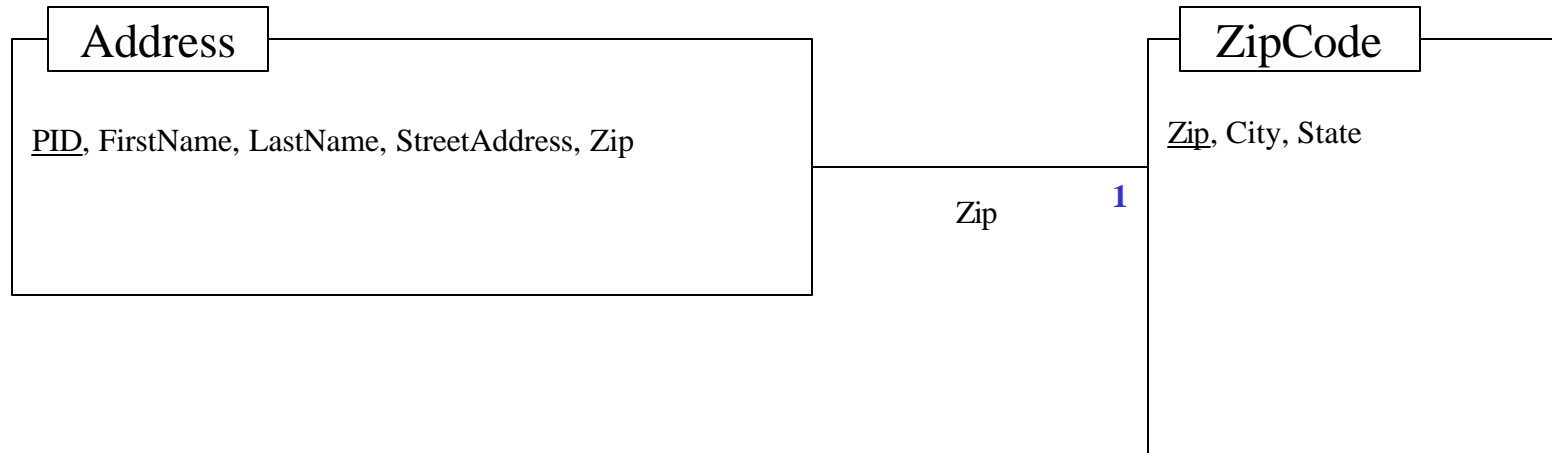
The definition of this foreign key in the create statements as well as our denoting the relationship in the E-R diagram serve to document this intentional relationship.

BUT WAIT... THERE'S MORE!



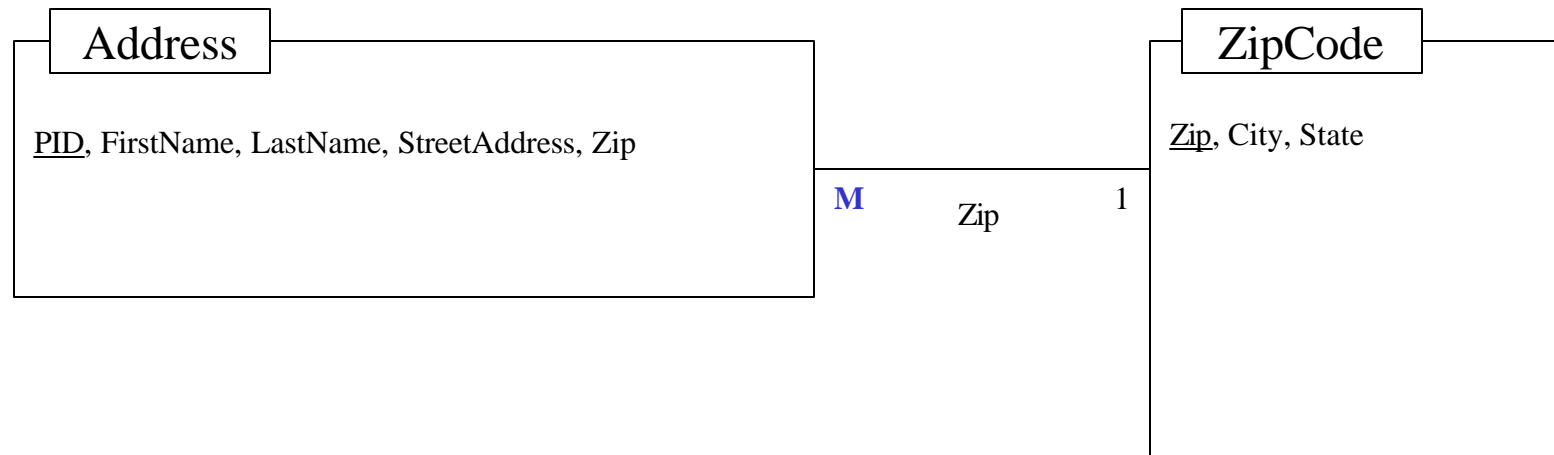
The tables are related on Zip. There are three kinds of relationships: one-to-one (1-1), one-to-many (1-M), and its converse many-to-one (M-1). There is a fourth type of relationship, the many-to-many. M-M relationships are implemented as 1-M, M-1 with a third table in the middle. Which kind of relationship do we have here? Do we need a third table?

ONE-TO-MANY RELATIONSHIPS



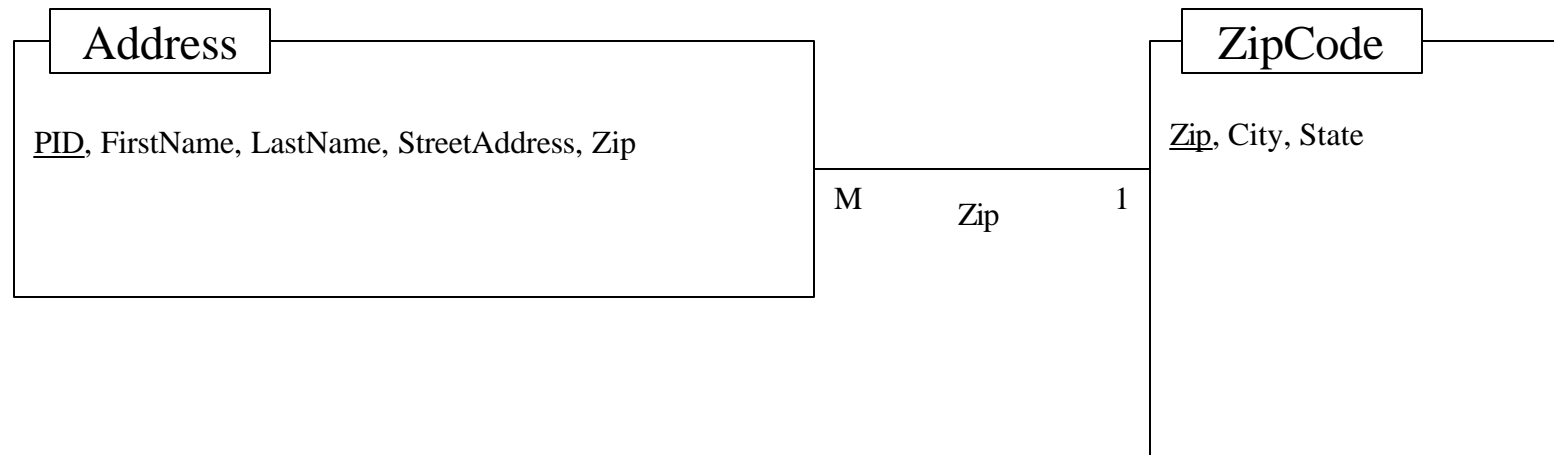
We can only get exactly one set of city and state values in ZipCode for a given Zip in Address. (This is because city and state are “functionally determined” by Zip Code.) We denote this in our E-R diagram by placing a “1” near the ZipCode end of the relationship line.

ONE-TO-MANY RELATIONSHIPS



We could get many addresses in Address for a given Zip in ZipCode. Denote this by placing an “M” near the Address end of the relationship line.

ONE-TO-MANY RELATIONSHIPS



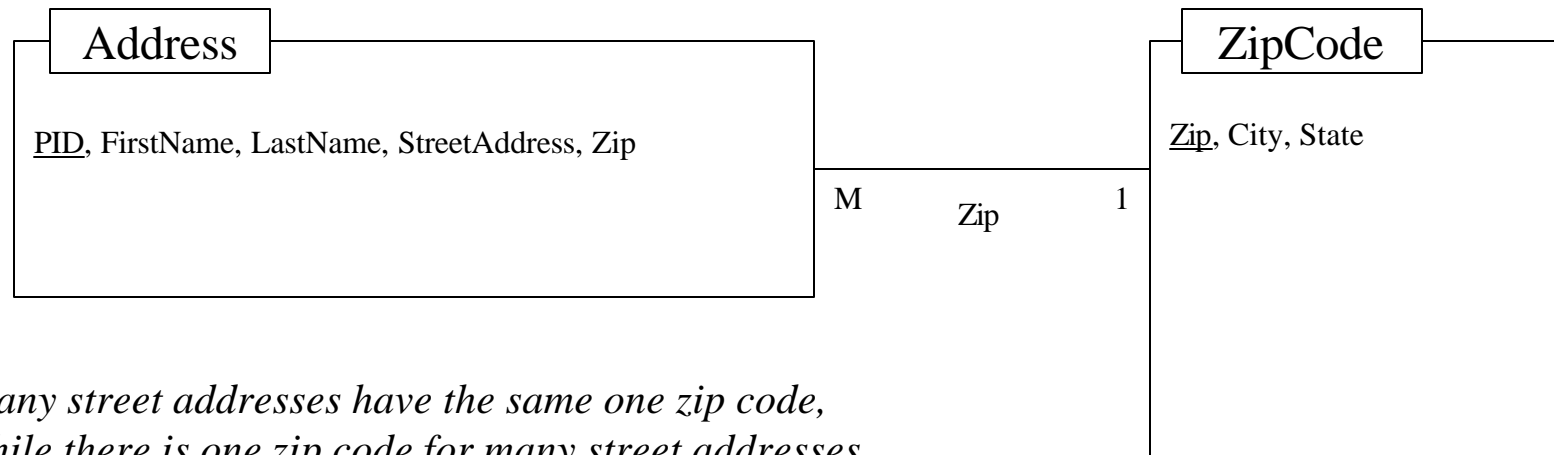
Test the validity of the relationship by applying the common sense test:

“Many street addresses have the same one zip code, while there is one zip code for many street addresses.”

If it sounds right, and rings true, then you probably have the relationship correct.

Remember: be attentive to the relationships, and be reasonable by judging their correctness.

ONE-TO-MANY RELATIONSHIPS



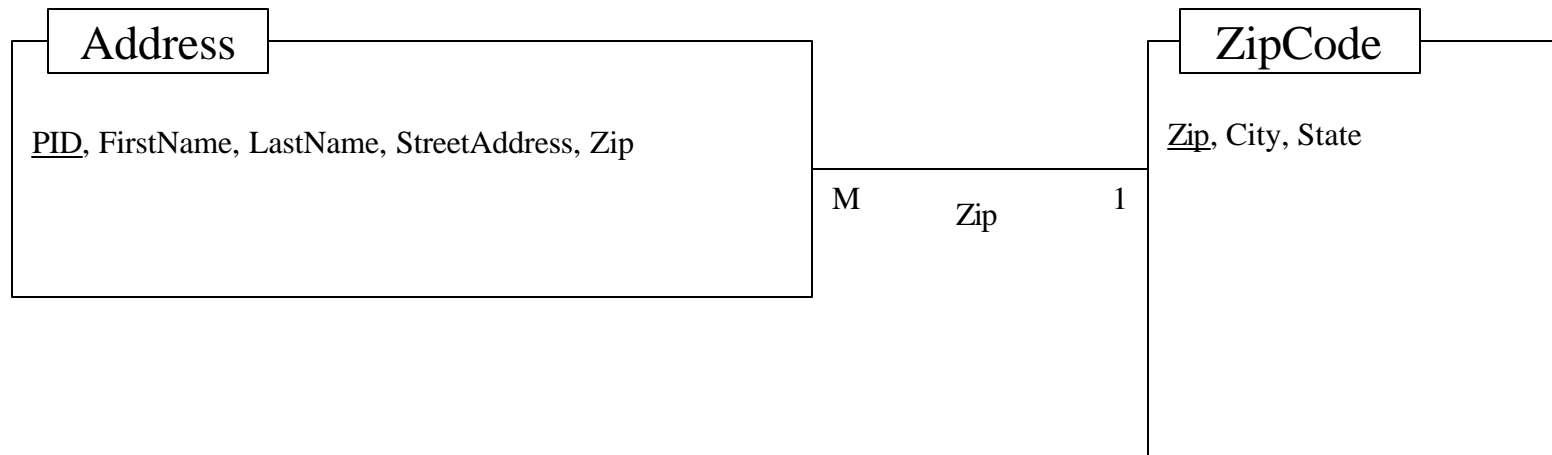
Many street addresses have the same one zip code, while there is one zip code for many street addresses.

The common sense test illustrates two important points:

- relationships are between tables as a whole, not their individual rows
- relationships are reflective; they always work both ways

The second point is worth exploring. A one-to-many relationship must, by definition, also be a many-to-one relationship. This works in exactly the same way as familial relationships: if Karen is my mother then I am by definition her son. I cannot be her son without her being my mother.

THE RIGHT QUERY



Now that we are aware of the relationship between these tables, we can write an appropriate query when asked to derive information from them. An appropriate query will use the foreign key relationships.

To add city and state information to the addresses:

```
Select *  
From Address, ZipCode  
Where Address.Zip = ZipCode.Zip
```

THE RIGHT QUERY

```
Select *  
From Address, ZipCode  
Where Address.Zip = ZipCode.Zip
```

PID	FirstName	LastName	StreetAddress	Zip_A	Zip_B	City	State
007	James	Bond	180 Treaty Road	12508	12508	Beacon	NY
002	Bill	Fairbanks	10 Kirkbride Ave.	07642	07642	Hillsdale	NJ
006	Alex	Trevelyan	2476 Bolsover St.	27410	27410	Greensboro	NC
008	Lee	Pfeiffer	149 Juniper Way	12508	12508	Beacon	NY
005	Dave	Worral	52 Simon Block	94118	94118	San Francisco	CA

This query correctly correlates city and state sets to the people and addresses based on the zip code. The foreign key relationship, intentionally put forth by the database designer, and intentionally used by the database user, ends up generating meaningful results: **data given context**.

A WRONG QUERY

```
Select *  
From Address, ZipCode  
Where Address.LastName = ZipCode.City
```

PID	FirstName	LastName	StreetAddress	Zip_A	Zip_B	City	State
002	Bill	Fairbanks	10 Kirkbride Ave.	07642	99701	Fairbanks	AK

This query incorrectly correlates a city and state to a person based on an unintentional relationship imposed on the database by the query writer. This happens when the query writer is not attentive to the intentional relationships among the tables in the database.

It's easy to see that the results here are not very informative. The data is not in the correct context.
There is no information!

CONCLUSIONS

Data is dangerous. What we really want is information. Information is data given context. Context comes from relationships between tables. Foreign key constraints implement relationships between tables. E-R diagrams further document these intentional relationships.

Relationships are 1-1, 1-M, or M-1. They are always reflective: 1-M is also M-1 in the other direction. Use the common sense test to evaluate the validity of your relationships.

As a database designer

be <i>Attentive</i>	to the relationships you are trying to model.
be <i>Intelligent</i>	about understanding the relationships and how to best model them.
be <i>Reasonable</i>	when judging your design for accuracy and completeness
be <i>Responsible</i>	about implementing and documenting your intended relationships

As a database user

be <i>Attentive</i>	to the intended relationships among the tables
be <i>Intelligent</i>	about understanding the relationships and how to best use them
be <i>Reasonable</i>	when judging the quality and accuracy of your queries
be <i>Responsible</i>	for using the intentional relationships to write queries that result in information

ABOUT THE AUTHOR

Alan G. Labouseur

With more than a decade of IT consulting experience, Mr. Labouseur has worked in more than 10 industries on over 35 projects in broad range of capacities. From hands-on database and software development to training and support, from networking to strategy and project management, he has done it all.

Mr. Labouseur has Bachelor's and Master's degrees in Computer Science and teaches database systems analysis and design for several universities. In addition to his lengthy experience with a wide variety of database products, Mr. Labouseur's specialty is data modeling and database design. He continues to consult on both large-scale and small-scale projects in the areas of data architecture, object-oriented programming, messaging, n-tier architecture, Web-server integration, workflow and data warehousing.

Contact Alan by e-mail at alan@3NFconsulting.com .



Alan teaching in 2001