

A gentle introduction to orthogonal

Maurice Gittens <maurice at gittens dot nl>

1st April 2005

Abstract

This paper introduces the concept of an orthogonal as a generalized application of the concept of an identifying relationship to accommodate both the notion of complex domains and higher order relations. By means of examples, it will be illustrated, that concepts such as complex domains can be accommodated in relational databases without violating the First Normal Form constraint for relational databases. Also, an accessible intuition for higher order relations will be shown to be possible by the application of orthogonal. The concepts presented in this paper might serve as a partial confirmation that the primitives Codd provided for “extending database to capture more meaning” can be made to serve their purpose well. Finally, the concept of an orthogonal also provides additional evidence that Date and Darwen erred, in their identification of the alleged “great blunders” in their book known as “The Third Manifesto”.

The copyright of this document belongs to its author. Making complete and unmodified copies of this document is allowed.

Contents

1	Introduction	2
2	Pertinent concepts from Entity-Relationship modelling	3
2.1	Introduction	3
2.2	Classifying entities	3
2.3	Classifying relationships	3
3	Introducing orthogonal by example	3
3.1	Mandatory one-to-one orthogonal	3
3.2	Mandatory one-to-one orthogonal as complex domains	5
3.3	Optional one-to-one orthogonal	5
3.4	One to many orthogonal	6
3.5	Higher degree Orthogonal	7

4	Higher order relations and Orthogonals	9
4.1	Introduction	9
4.2	Higher order relations by example	9
4.3	So, what is an orthogonal?	11
5	Revisiting the Third Manifesto’s alleged “Great Blunders”	11
5.1	Introduction	11
5.2	Object classes are domains	11
5.3	Rejecting Object Identifiers	11
6	Concluding remarks	11

Status: Preliminary draft...

Revision History

- March 29 2005, Fixup my use of the language even more.
- March 15 2005, generalize the intensional definition of an orthogonal to accommodate higher degree orthogonals.
- March 11 2005, Fix more typos and misuse of the English language. Thanks Elena!
- March 10 2005, More cleanups, remove section on orthogonal data models
- March 6 2005; first draft completed
- July 2004; more changes
- February 24 2004; start with first version of this paper
- March 2003; birth of the idea

1 Introduction

The purpose of this paper is to introduce the concept of an orthogonal by example. The concept of an orthogonal exploits so-called identifying relationships to facilitate both the notion of complex domains and also higher order relations. This document assumes some familiarity with relation databases. The intended audience includes those interested in database design and applications of predicate logic.

2 Pertinent concepts from Entity-Relationship modelling

2.1 Introduction

Entity-Relationship modelling is a popular technique for creating logical models for database systems. This section will highlight a few key Entity-Relationship modeling concepts that will be used as building blocks for introducing Orthogonals in this article.

2.2 Classifying entities

Entity Relationship modelling classifies entities into *independent* and *dependent* entities. An independent entity is an entity that does not rely on another entity for identification. Conversely, dependent entities are entities that depend on other entities for identification.

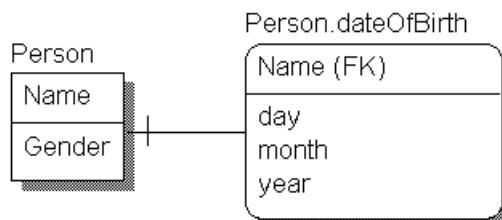
2.3 Classifying relationships

A relationship represents a grouping of two or more entities. Relationships are classified in different ways. Often the classification of relationships includes *degree*, *cardinality*, *existence* and *type*. The degree of relationships is determined by the number of entities participating in the relationship. The cardinality of relationships is usually designated (one-to-one, one-to-many, etc). Existence determines whether a relationship is optional or required. The type of a relationship determines whether the entities in a relationship are independent or not. Relationships between entities that depend on each other are called *identifying relationships*. When entities participating in a relationship are independent we speak of a *non-identifying relationship*. Identifying relationships are of particular interest in this paper because they form the conceptual foundation upon which orthogonal relations and orthogonal entities are built.

3 Introducing orthogonals by example

3.1 Mandatory one-to-one orthogonals

Consider the following ER diagram illustrating a mandatory identifying one-to-one relationship between two dependent entities.



The following fragment in an SQL-like language is intended to represent a means to physically realize this ER diagram in a hypothetical database management system.

```
declare orthogonal Date (day integer, month integer, year integer);

create table Person(name char(20), gender char(1), dateOfBirth Date, primary key(name));
```

The first action performed in the previous fragment is the declaration of an orthogonal called *Date*. The *Date* orthogonal sports three integer attributes representing respectively the day, month and year constituents of dates to be associated with entities to which the *Date* orthogonal will be attached. Subsequently, the orthogonal is employed to attach a *Date* orthogonal called *dateOfBirth* to the *Person* table. In addition to an attachment of the *Date* orthogonal called *dateOfBirth*, the *Person* table sports a *Name* and a *gender* attribute. The *Name* attribute is designated as the primary key of the *Person* table. An additional syntax for the above follows.

```
declare orthogonal Date (day integer, month integer, year integer);

create table Person(name char(20), gender char(1), primary key(name));

attach Date to Person(dateOfBirth);
```

¹ The following fragment shows how the *Person* table might be populated.

```
INSERT INTO Person (name, gender, dateOfBirth.day, dateOfBirth.month, dateOf-
Birty.year)
VALUES ('Joe', 'M', 23, 7, 1987)
INSERT INTO Person (name, dateOfBirth.day, dateOfBirth.month, dateOfBirty.year)
VALUES ('Jane', 'F', 12, 4, 1960)
```

It is important to appreciate that the expression *Person.dateOfBirth* represents the dependent relational *table* with an identifying mandatory one-to-one identifying relationship between it and the the *Person* table.

When the *Person* table is populated using the insert statements above, the following relation will result from the query: *select * from Person.dateOfBirth*.

NAME	day	month	year
Joe	23	7	1987
Jane	12	4	1960

The query *select * from Person* returns the following relation.

NAME	gender	dateOfBirth.day	dateOfBirth.month	dateOfBirth.year
Joe	M	23	7	1987
Jane	F	12	4	1960

¹All attributes are assumed to be NOT NULL by default

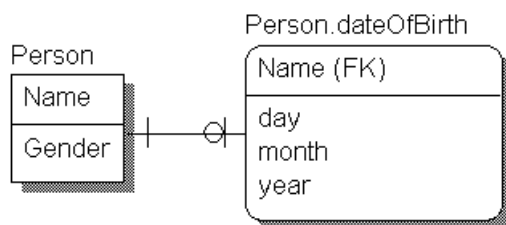
This example illustrates that orthogonal are no more than syntactic dressing on dependent entities and identifying relations, the semantics of updating and deleting for mandatory one-to-one orthogonal are quickly recognized and left for as an exercise for the reader.

3.2 Mandatory one-to-one orthogonal as complex domains

From the previous section it is soon recognized that apart from artificial differences such as details of syntax, mandatory one-to-one orthogonal may be viewed as means to support the notion of complex domains. This is quickly recognized when it is appreciated that orthogonal may employ other orthogonal in their definition. It is interesting to take note of the fact that the complex domains similar to those presented by Date and Darwen ref[1] in the book known as *The Third Manifesto* can be supported by employing only dependent entities and identifying relations. Specifically, these so-called complex domains can be supported without violating the constraint known as the “*First Normal form*”.

3.3 Optional one-to-one orthogonal

As stated earlier, orthogonal are based on dependent entities and identifying relationships. In the case of one-to-one identifying relationships, whenever the identifying relationship employed is not mandatory, a situation is created much resembling the employment of null-able columns. An ER diagram representing this case follows.



A possible syntax for optional one-to-one orthogonal is shown in the following.

```
create table Person(Name char(20), dateOfBirth Date NULLABLE, primary key(name));
```

In the context of the previous example it becomes possible to populate the person table as in the following example:

```
INSERT INTO Person (Name, dateOfBirth.day, dateOfBirth.month, dateOfBirty.year)
```

```
VALUES ('Joe', 'M', 23, 7, 1987)
```

```
INSERT INTO Person (name,gender) VALUES ('Jane', 'F')
```

In the case the *Person* table is populated using the insert statements above, the query *select * from Person.dateOfBirth* will return the following relation.

Person.Name	day	month	year
Joe	23	7	1987

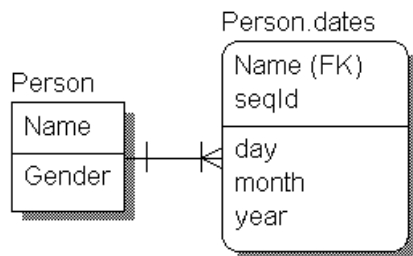
The query *select * from Person* will return the following relation.

Person.Name	gender	dateOfBirth.day	dateOfBirth.month	dateOfBirth.year
Joe	M	23	7	1987
Jane	F			

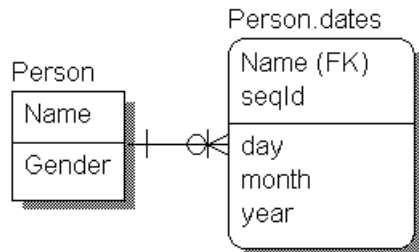
In this case the *dateOfBirth* of *Jane* is unknown and does not appear in the *Person* relation.

3.4 One to many orthogonal

Based on the different ways in which identifying relationships may be classified orthogonal relationships are also classified. Therefore, mandatory one-to-many orthogonal relationships as in the following ER diagram can be identified.



Similarly, optional one to many orthogonal relationships can be recognized.



Obviously, one to many orthogonal relationships add an additional set of attributes to the identifying set of tuples for each tuple in the orthogonal table called *Person.memorableDate* in our

example. An example using an SQL like syntax for creating a mandatory one to many orthogonal is provided in the following.

```
declare orthogonal DateSequence(seqId integer, day integer, month integer,
year integer, primary key(seqId));

create table Person( Name char(20), gender char(1), dates DateSequence, primary key(Name));
```

It is noted that in this example an attribute *seqId* representing a sequence number is added to the declaration of the *DateSequence* orthogonal as a primary key attribute. The *Person* table might be populated using the following insert statements.

```
INSERT INTO Person (Name, gender, dates.seqId, dates.day, dates.month, dates.year)
VALUES ('Joe', 'M', 1, 23, 7, 1987)

INSERT INTO Person (Name, gender, dates.seqId, dates.day, dates.month, dates.year)
VALUES ('Joe', 'M', 2, 12, 8, 1992)

INSERT INTO Person (Name, gender, dates.seqId, dates.day, dates.month, dates.year)
VALUES ('Jane', 'F', 1, 8, 8, 1970)
```

In this case the result of the query: *select * from Person* would be:

Name	gender	dates.seqId	dates.day	dates.month	dates.year
Joe	M	1	23	7	1987
Joe	M	2	12	8	1992
Jane	F	1	8	8	1970

The query: *select * from Person.dates* would be:

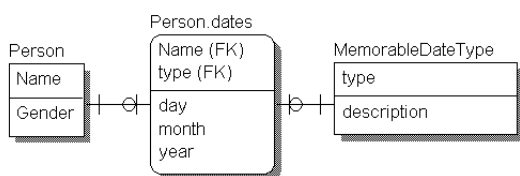
Name	seqId	day	month	year
Joe	1	23	7	1987
Joe	2	12	8	1992
Jane	1	8	8	1970

An example illustrating optional one to many orthogonals is left as an exercise for the reader.

3.5 Higher degree Orthogonals

Higher degree orthogonals are formed by including the identifying attributes of three or more dependent entities into the set of identifying attributes of tuples in the higher

degree orthogonal. The following ER diagram presents the case of an orthogonal based on a ternary identifying relationship. In this case we speak of an orthogonal of the second degree. All orthogonals introduced up till now in this paper have been first degree orthogonals.



```

declare orthogonal memorableDate (day integer, month integer, year integer);

create table Person(name char(20), gender char(1), primary key (name));

create table MemorableDateType(type char(1), desc char(256), primary key (type));

attach memorableDate to Person(dates nullable), MemorableDateType(instances nullable)
    
```

In the above data definition is important to note that *MemorableDate* orthogonal is attached to both the *Person* and the *MemorableDateType* tables. Thus creating one dependent table identified by both *Person.dates* and *MemorableDateType.instances*. The following insert statements add a *Person* tuple to our example database.

```

insert into Person(Name, gender, dates.type, dates.day, dates.month,
dates.year, MemorableDate.desc) value('Joe', 'M', 'B', 20,10, 1970).
    
```

To emphasize that the *Person.dates* and *MemorableDateType.instances* are basically synonyms for the same table, the following fragment illustrates an alternative statement to the previous one which adds the same information to the database.

```

insert into MemorableDate(dates.Name, Person.gender, dates.type, dates.day,
dates.month, dates.year, MemorableDate.desc) value('Joe', 'M', 'B', 20,10, 1970).
    
```

With suitable population statements the query: *select * from Person* might result in the following table.

Name	gender	dates.type	dates.day	dates.month	dates.year	MemorableDates.desc
Joe	M	B	23	7	1987	Birth day
Joe	M	M	12	8	1992	Marriage date
Jane	F	M	8	8	1970	Marriage date

In this case the queries *select * from Person.memorableDate* and *select * from MemorableDateType.instances* would both result in the following table.

Name	type	day	month	year
Joe	B	23	7	1987
Joe	M	12	8	1992
Jane	M	8	8	1970

Since only dependent entities and identifying relationships have been employed in the introduction of higher degree orthogonal, it is clear that the semantics of data manipulation (deletes, updates, inserts) has already been defined in the relational algebra.

4 Higher order relations and Orthogonals

4.1 Introduction

Up until this point, orthogonal have been introduced as a particular generalized employment of identifying relationships. This section will venture out of the realm of the current relational systems. However, the ventures beyond these frontiers will be guided by Codd's "Extending the Database Relational Model to Capture More Meaning". Specifically, higher order relations will employ concepts which are at least similar in spirit to Codd "E-domains" which represent the set of surrogates (probably better known as Object Identifiers), and his "RN-domains" which is the set of identifiers for relations.

4.2 Higher order relations by example

Consider a simple database consisting of three tables all with a single orthogonal attached as in the following.

```
declare orthogonal Date (day integer, month integer, year integer);
create table Person(name char(20), primary key(name));
create table Organisation(name char(20), primary key(name));
create table Vehicle(id char(20), primary key(id));
attach Date to Person(dateOfBirth);
attach Date to Organisation(dateOfIncorp)
```

attach Date to Vehicle(dateOfMan)

As previously illustrated, the above arranges that dependent entities identified by *Person.dateOfBirth*, *Organization.dateOfIncorporation* and *Vehicle.dateOfManufacture* are created with identifying relations to the parent tables, respectively *Person*, *Organization* and *Vehicle*.

In the following it is agreed that the example database presented above is populated with the following insert statements.

```
INSERT INTO Person (Name, dateOfBirth.day, dateOfBirth.month, dateOfBirth.year)
```

```
VALUES ('Joe', 23, 7, 1997)
```

```
INSERT INTO Person (Name, dateOfBirth.day, dateOfBirth.month, dateOfBirth.year)
```

```
VALUES ('Jane', 2, 12, 1987)
```

```
INSERT INTO Organisation (Name, dateOfIncorp.day, dateOfIncorp.month, dateOfIncorp.year)
```

```
VALUES ('Joe', 12, 3, 1980)
```

```
INTO Vehicle (Name, dateOfMan.day, dateOfMan.month, dateOfMan.year)
```

```
VALUES ('23-RD', 3, 4, 1970)
```

Imagine now for a moment that one wants to have an overview of all *Date* instances in this example database. This can be achieved by the query: *select * from Date*.

Table Identity	Primary Key	day	month	year
Person.dateOfBirth	Oid({Joe})	23	7	1997
Person.dateOfBirth	Oid({Jane})	2	12	1987
Organization.dateOfIncorp	Oid({My Corp})	12	3	1980
Vehicle.dateOfMan	Oid({23-RD})	3	4	1970

This query results in the relation above which has columns for all attributes of the *Date* orthogonal. Additionally, the result includes a RN-Domain column identifying the table in which the particular *Date* instance occurs and also an E-domain column with the object identifier² of the particular *Date* instances. The example above illustrates that orthogonal relations may be viewed as relations including RN-Domains columns and E-Domain domains. This is another way to say that higher order orthogonals are analogous to higher order predicates in predicate logic.

²Thus expressions such as Oid({Joe}) in the example above represent the identity of a tuple in some relation.

4.3 So, what is an orthogonal?

An intensional definition of an orthogonal might be a relation which has tuples identified by a pair consisting of a RN-Domain and a E-domain for each degree of the orthogonal. In any particular database the extensional definition of an orthogonal would be: the total set of tuples in all tables of the database based on the particular orthogonal.

5 Revisiting the Third Manifesto's alleged "*Great Blunders*"

5.1 Introduction

This section goes off on a tangent related to the position of Date and Darwen [ref 1] voiced in their book known as "The Third Manifesto" with regards to concepts to be supported in future databases. For the reader interested only in the introduction to orthogonals provided in this article, this section may be safely skipped.

5.2 Object classes are domains

In The third manifesto Date and Darwen argue, contrary to Codd, that the constraint known as First Normal Form is to be violated in pursuit of their notion that an object class should be equated to a relational domain. Date & Darwen even claim: "*The question as to what data types are supported is orthogonal to the question of support of the relational model.*".

The orthogonals introduced in this paper illustrate that plain vanilla dependent entities and identifying relations, suitably syntactically and semantically coated can allow the concepts of complex domains to be supported without violation of the First Normal Form constraint.

5.3 Rejecting Object Identifiers

Again, contrary to Codd, Date and Darwen would have Object Identifiers (Codd calls them surrogates) be dismissed from future databases. The section on higher order orthogonals presented in this paper illustrates by example that Object Identifiers, however one chooses to call them, form the basis of higher order relations. While an elaboration on the possibilities of higher order relations has not been pursued in this paper, it seems evident that dismissing identifying the form of object identifiers, (or surrogates if you prefer), from future databases is not a dictate of prudence at this moment in time.

6 Concluding remarks

The purpose of this paper is to introduce the concept of an orthogonal by example. Orthogonals were shown to employ dependent entities and identifying relationships.

Orthogonals allow the intuition of complex domains to be incorporated in relational databases without violation of the First Normal Form constraint. Additionally, orthogonals provide an intuitive way to introduce higher order relations in relational databases as envisaged by Codd.

References

- [1] C.J. Date, Hugh Darwen [2000] Foundation for Future Database Systems, Addison-Wesley Publishing Company.
- [2] E.F. Codd, [1979] “Extending the Database Relational Model to Capture more Meaning”
- [3] Maurice Gittens, [2003] “A critical reading of the third manifesto”, <http://www.gittens.nl>
- [4] Information Technology Services, University of Texas at Austin, [2005], <http://www.utexas.edu/its/windows/database/datamodeling/dm/erintro.html>,