

# About the concept of an Orthogonal

## Table of contents

1 The foundation.....	2
2 Visualizing information bricks.....	2
2.1 A first information structure.....	3
2.2 Adding a non-identifying foreign key relations.....	3
2.3 Adding orthogonals.....	4
2.4 Classifying orthogonals.....	5
3 Summarizing.....	7

## 1. The foundation

This page introduces the concept of an orthogonal as a generalisation of the plain vanilla identifying relations as these are known in the realm of Relational Data Models.

### Simple Orthogonal...

For example, in the diagram above there are two entities *Person*, *Person.dateOfBirth* where the entity called *Person.dateOfBirth* has common identifying attributes with the *Person* entities.

The following code fragment illustrates how the above diagrams might be defined in a fictive SQL like Data Definition Language.

```

integer;
        declare orthogonal Date
        {
                day integer; month integer; year
        }

        create table Person
        {
                name char[20];
                gender char[1];
                dateOfBirth Date;
                primary key(name)
        }

```

The above example illustrates that the Date orthogonal may be viewed as a compound domain of the the person table. However, it is important to appreciate that the Date orthogonal represents a table in its own right.

The article entitled, [A Gentle Introduction to Orthogonals](#) introduces a few types of orthogonals. This article needs to be extended to include additional types of orthogonals. Such as *classifying orthogonals* and orthogonals based on cartesian products.

Orthogonals accommodate an intuitive 3D visualisation technique which I would like to share with you on the condition that you forgive my limited diagramming skills.

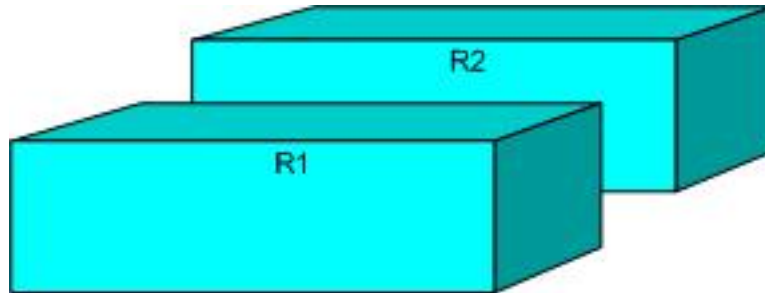
## 2. Visualizing information bricks

The the example driven article [A Gentle Introduction to Orthogonals](#) introduced the concept of an orthogonal in relational terms. The examples in this article where presented in a fictive language similar in syntax and semantics with SQL. It turns out that the syntax of SQL seems to *get in the way* of eloquently expressing information structure.

In this section an intuitive 3D visual representation of information structures, or information models is presented. This visual representation language which I will call *visual bricks* is one source of inspiration in the design of the text based information definition and querying language I have been working on for some time.

## 2.1. A first information structure

Let R1 and R2 in the following diagram represent two relational tables.

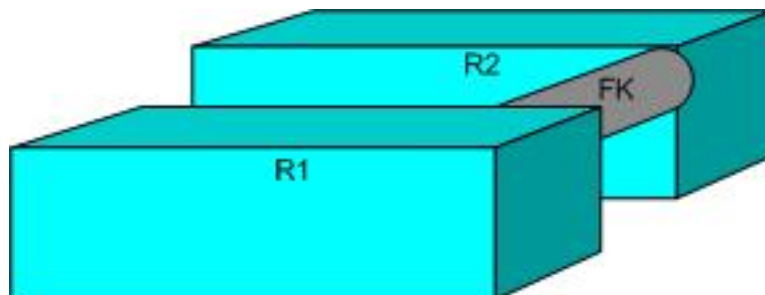


First example structure...

Each relational table, or relation variable if you prefer, is represented by a 3D rectangular block.

## 2.2. Adding a non-identifying foreign key relations

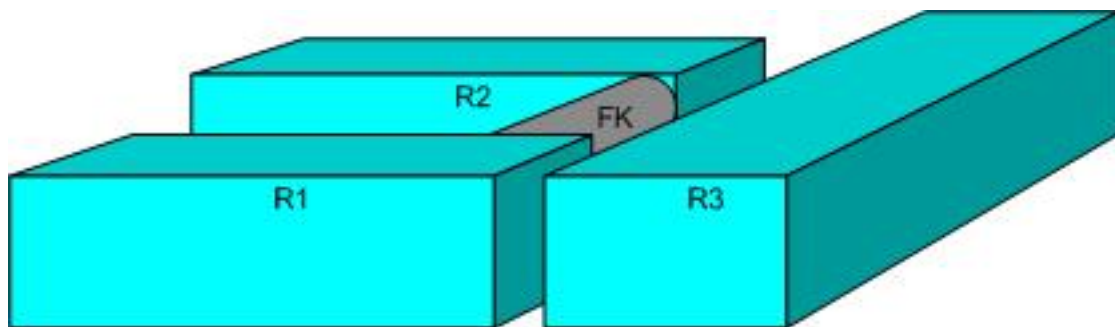
The next step introducing the information brick visualiation language is taken by adding a non identifying foreign key relation, between the information blocks R1 and R2.



Second example structure...

These non identifying foreign key relations are represented as rounded directional tubes connecting no more than two information blocks. Even though this is not evident from the figure above, it is emphasized that the foreign key relations are *directional*. Which is to say, they point in one direction from one information block to another.

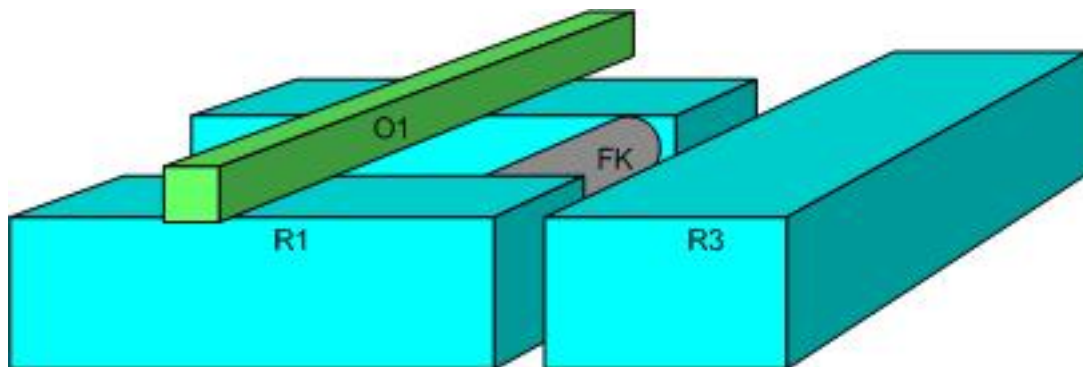
Of course any number of information blocks can be added to the an information structure. As an illustration, the following diagram adds an additional relational table R3.



Third example structure...

### 2.3. Adding orthogonals

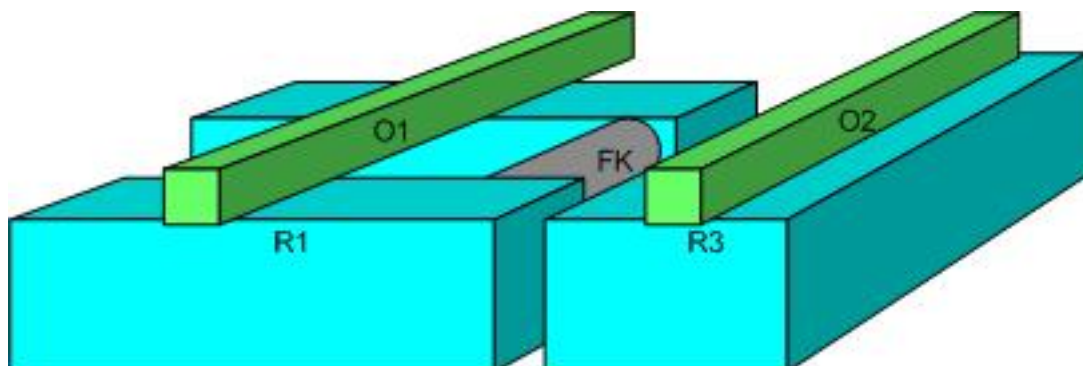
All identifying relations are represented by placing information block on top of each other. In the following figure the orthogonal O1 is placed on the information block R1 and R2.



Fourth example structure...

Semantically, this entails that there is an identifying relationship between O1 and R1 and also between O2 and R2. More specifically, this entails that the primary key of the table O1 identifies tuples in both the R1 and the R2 tables.

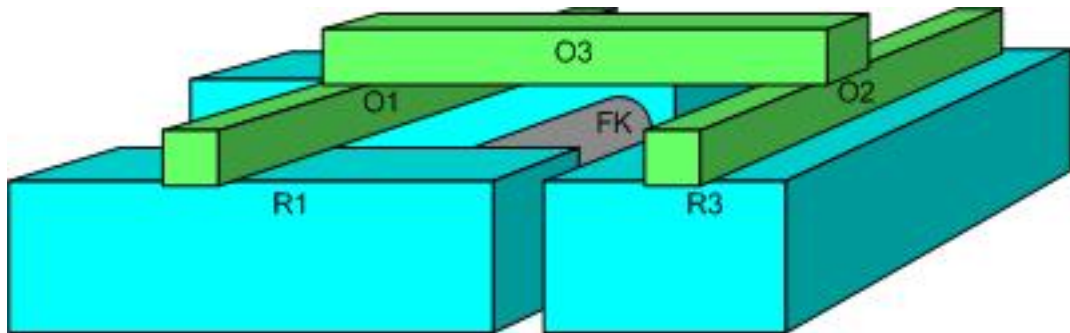
Similarly, we can add an orthogonal O2 to the relation R3.



### Fifth example structure...

Again O3 is placed on top of R3 to express the fact that there is an existence dependency between tuples in R3 and tuples in O2.

Finally, we add yet another relation O3 on to of O2 and O2.



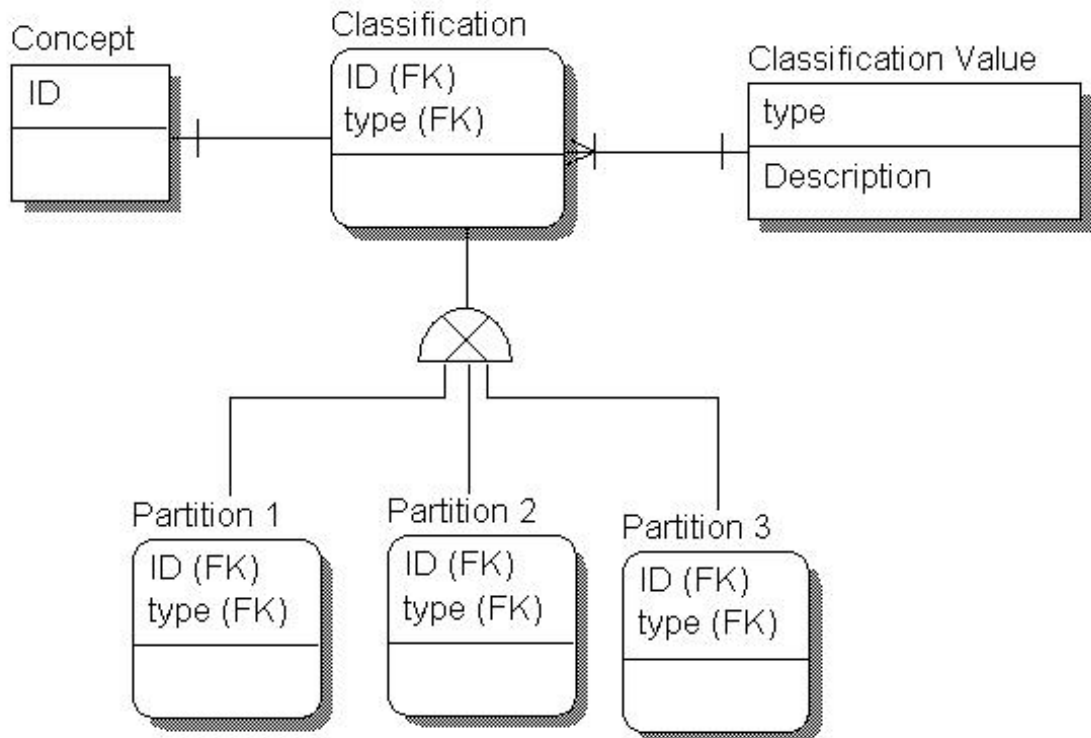
### Sixth example structure...

The semantics of this operation has already been explained in the previous section.

In the following section one additional building block will be added to our information structure construction tool box. These building blocks will be called *classifying orthogonals*.

## 2.4. Classifying orthogonals

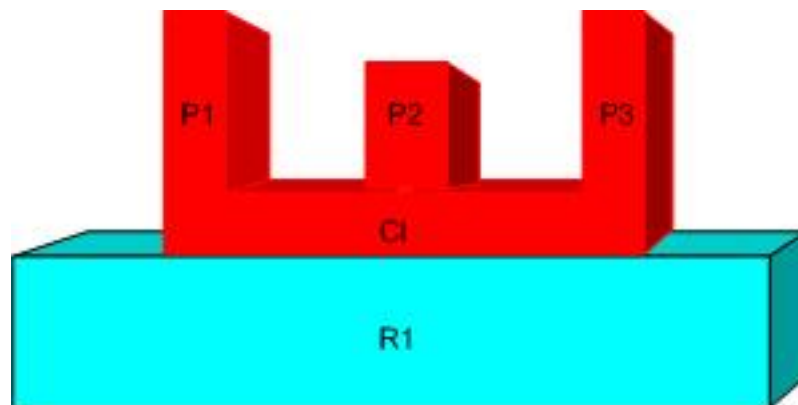
A useful generalisation of an orthogonal is known as a classification. Classifications are common in relational data models. For example datamodellers often classify the concepts such for example a *Party* into a number of subset such as Individual, Organisations and Organisation Units. This classification is often called *PartyType*. Classifications are so common that certain modelling languages have notations corresponding to them. An example is provided in the following diagram.



### A classification...

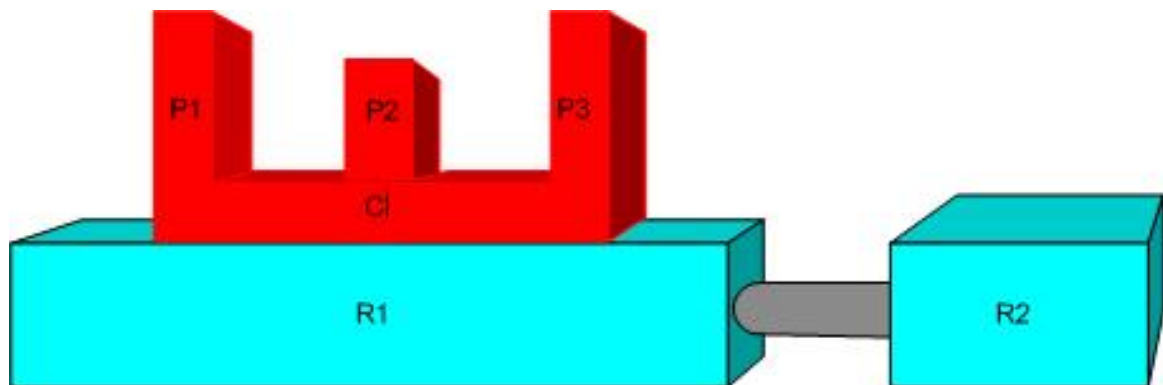
The diagram presents a concept classified in to three disjoint partitions which, in general, will have their own specific characteristics.

A visual brick a corresponding with a classification such as the above follows.



### Seventh example structure...

The above is a figure in which a brick R1 is classified into three partitions P1, P2 and P3 by a classification C1. In general a specific concept may be classified by any number of classifying orthogonals.



Eight example structure...

The diagram illustrates the obvious point that information structures employing classifying orthogonals can be extended with non identifying relationships.

### 3. Summarizing

This page provided a quick introduction to topic of Orthogonal structures. Hopefully this introduction has not been hampered too much by feeble attempts at visualizing the information bricks.

There is much more to be said on the topic of orthogonals, the page on [Semantic Grammars](#) will elaborate on this topic from a different perspective.

There is also a page about a [Data Definition and Manipulation](#) language based on the ideas presented here.