

14 Data dictionary

14.1 What is a data dictionary?

Unfortunately 'data dictionary' (DD) is a very generic term used by many manufacturers to mean many things. e.g. Oracle as a repository of DB object definitions, & CASE tools for systems analysis object definitions. However, they all contain metadata, i.e. data about data.

In this case, it is information about the physical structure of our databases. Such as descriptive information on databases, tables, columns, sps, etc.

14.2 Why have a data dictionary?

- Dynamic on-going documentation of the DBs.
- Central rather than spread over many specs.
- To assist development team:
 - Handovers & bringing staff upto speed.
 - Build better systems
 - Reduce the columns & tables added, due to lack of knowledge of the DB.
 - Provide print outs of DB object information
- To assist administration of our DBs:
 - Help find obsolete objects & data.
 - Hold information on archiving & defragmentation.
 - Table performance tuning information

14.3 Implementation

The DD is held in an Access DB, possibly SQL later, with a VB & or Access front end. You need to tell the DBA if you create new tables. He will add them to the DD & create/edit the Entity Relationship Diagram (ERD).

14.4 Data stored

These are the columns relating to table DD information:

Database
TableName
CreationDate
Creator
MainKnowledgeHolders
Desc
DefragNotes
Archiving
CoreTable
Triggers
Iotype
DefragNeeded
DefragDate
RoughSize
FillFactor
ClusteredIndex

Information is also stored about, databases, columns & later possibly sp's.

15 Entity Relationship Diagrams (ERD)

15.1 Why use ERDs?

An ERD can clearly show the data relationships between tables. It is a very useful diagram when designing or developing a relational DB. It is particularly useful, when foreign key (FK) columns are not named the same as their parent PK.

For very large db's, break it into modules, ie groups of tables that refer to each other frequently, or are part of an application, eg online reporting.

All new developments which involve new tables should have an ERD.
Once the table information has been given to the DBA, he can create an ERD for you.
This will be created in Visio & a jpeg saved to the intranet.

15.2 Terminology

Entity

Some identifiable object relevant to the system being built. Examples of Entities are:

EMPLOYEE
ORGANISATION

Physical representation = Table

Entity instance

An *instance* of an entity is like a specific example:

Bill Gates is an Employee of Microsoft
Greenpeace is an Organization

Physical representation = Row

Attribute

A characteristic of an Entity. Properties used to distinguish one entity instance from another.

Attributes of entity EMPLOYEE might include:

EmployeeID
First Name
Last Name

Physical representation = Column

15.3 Relationships

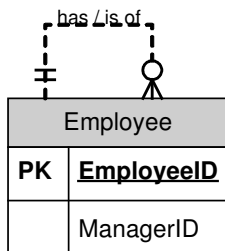
ERD's show the data relationships between tables; relationships are described as follows:

15.3.1 Number of entities in a relationship

A relationship can include one or more entities.
Relationships can be:

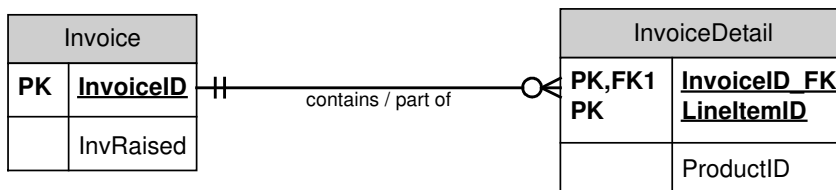
Unary

ie self-join.



Binary

Most relationships in databases are binary



Ternary

Try to resolve these to binary. Only seen in logical diagrams, so can ignore.

15.3.2 Optionality

This is also called: 'minimum degree', 'minimal cardinality' or the 'participation' of a relationship. Participation of instances in a relationship may be mandatory or optional. For example:

one CUSTOMER *may* place many CUSTOMER ORDERS
one EMPLOYEE *must* fill out one or more PAY SHEETS

15.3.3 Cardinality

This is also called: 'maximum degree'.

Relationship Cardinality refers to the number of **entity instances** involved in the relationship. For example:

one CUSTOMER *may* place *many* CUSTOMER ORDERS
many STUDENTS *may* sign up for *many* CLASSES
one EMPLOYEE receives *one* PAYCHECK

1:M One to Many
M:N Many to Many
1:1 One to One

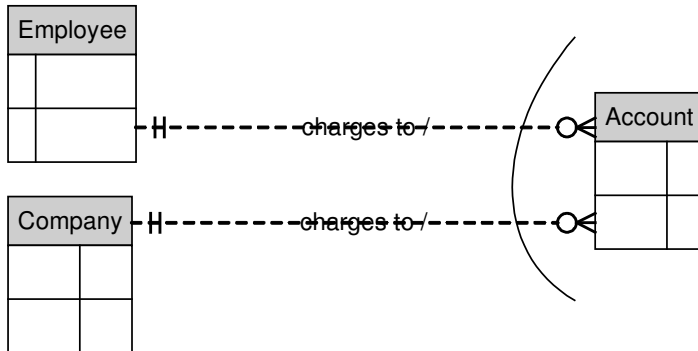
Beware of **1:1** relationships. The two entities involved might be coalesced into one. Also called **HAS-A** relationship. Beware of **M:N** relationships. Typically split these into two **1:M** relationships with an intersection entity.

REFS:

Explains cardinality with examples <http://www.datamodel.org>

15.3.4 Exclusive relationships

These are either-or situations. Visio doesn't support a method for this, so I'm going to use a free form arc.



15.4 Notation

Sometimes referred to as the formalism or methodology. I suggest we adopt the commonly used crow'sfoot notation. Fortunately Visio supports this notation method.

15.4.1 VISIO Notation

Visio has a number of notation methods that it supports, relational, express-G & ORM. We use the Visio relational notation.

15.4.1.1 Relationship lines

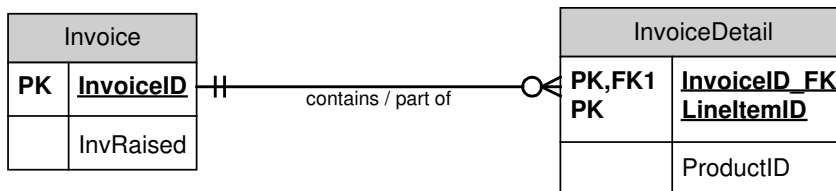
Identifying (solid line)

An identifying relationship, where the foreign key relationship includes a primary key component in the child table. Part of the primary key of the child table is also part of the primary key of the parent table.

Non-identifying (dashed line)

A non-identifying relationship, where the foreign key relationship does not include a primary key component in the child table. The part of the parent table primary key that is shared by the child table is not part of the child table primary key.

15.4.1.2 Setting optionality



To get the parent (ie invoice) to reflect one & only one (ie 2 vertical slashes), you need to set the foreign key as required. Ie an invoice detail can't exist without an associated invoice.

15.4.1.3 Relationship names

Right click on the relationship line > format > text, set to subscript or superscript to be above/below line.