22 Mar 2018

Avoid normalising data

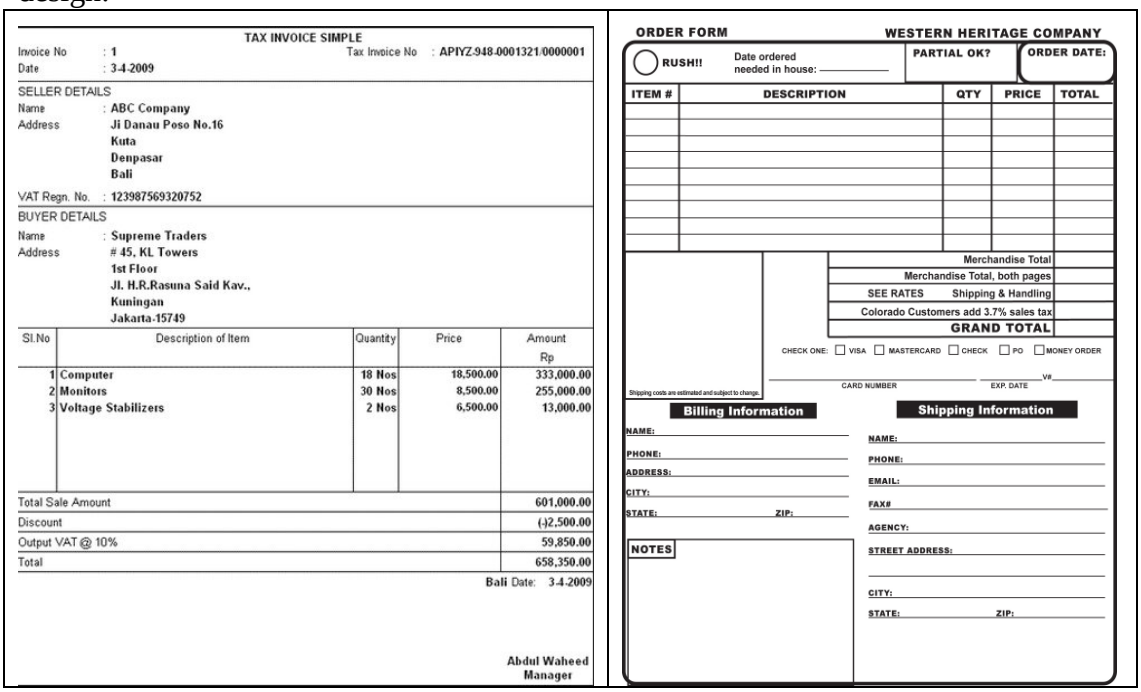
A little knowledge goes a long way to identify your data

Using normalisation as a means to develop a logical database design is simply not a good idea.

Why? If 3 data modellers were given the same set of attributes there is a chance that they would all produce a different database design. This is because:

1) You would need to uncover every conceivable business document that business operative use, in order to find the required attributes
2) Codd's rules and laws are far too implicit

Below is an example of two documents that could be used to design logical database design:



TAX INVOICE SIMPLE

Invoice No : 1
Date : 3-4-2009
Tax Invoice No : APIYZ-948-0001321/0000001

SELLER DETAILS
Name : ABC Company
Address : Ji Danau Poso No.16
Kuta
Denpasar
Bali

VAT Regn. No. : 123987569320752

BUYER DETAILS
Name : Supreme Traders
Address : # 45, KL Towers
1st Floor
Jl. H.R.Rasuna Said Kav.,
Kuningan
Jakarta-15749

Sl.No	Description of Item	Quantity	Price	Amount Rp
1	Computer	18 Nos	18,500.00	333,000.00
2	Monitors	30 Nos	8,500.00	255,000.00
3	Voltage Stabilizers	2 Nos	6,500.00	13,000.00
	Total Sale Amount			601,000.00
	Discount			(-)2,500.00
	Output VAT @ 10%			59,850.00
	Total			658,350.00

Bali Date: 3-4-2009

Abdul Waheed
Manager

ORDER FORM — WESTERN HERITAGE COMPANY

RUSH!! Date ordered needed in house: _____ PARTIAL OK? ORDER DATE:

ITEM #	DESCRIPTION	QTY	PRICE	TOTAL

Merchandise Total
Merchandise Total, both pages
SEE RATES Shipping & Handling
Colorado Customers add 3.7% sales tax
GRAND TOTAL

CHECK ONE: VISA MASTERCARD CHECK PO MONEY ORDER

CARD NUMBER EXP. DATE V#

Shipping costs are estimated and subject to change.

Billing Information

NAME:
PHONE:
ADDRESS:
CITY:
STATE: ZIP:

NOTES

Shipping Information

NAME:
PHONE:
EMAIL:
FAX#
AGENCY:
STREET ADDRESS:
CITY:
STATE: ZIP:

I will now demonstrate how normalising data using attributes from just 2 documents can lead to an inefficient and ineffective logical database schema and hence physical database design.

Firstly I will need to deal with Edgar F. Codd's laws in order to turn the attributes/fields contained in these 2 examples into a physical database design. This is going to be painful but if you refuse to acknowledge the existence of and to develop the 'knowledge model' or do not like the idea of developing a 'conceptual data model' (see my earlier article on this topic), then you are left with having to suffer through the rest of my article. Or you could research this on your own.

Codd's 3rd normal forms used to design fully normalised data models

Codd & Boyce provides us with the following normal forms:

1) Unnormalised which according to the Wikipedia (can anyone really trust this site – this is another problem) is

   An unnormalised data model will suffer the pitfalls of data redundancy, where multiple values and/or complex data structures may be stored within a single field/attribute, or where fields may be replicated within a single table (a way of subverting the first normal form rule of one value per field/attribute)

2) First normal form - according to the Wikipedia is

   "A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain". What exactly does this mean?

   To understand this you will need to define every word in the sentence as follows:

   1) What is a relation? Well according to The Wikipedia

      In relational database theory, a relation, as originally defined by E. F. Codd is a set of tuples (d1, d2, ..., dn), where each element dj is a member of Dj, a data domain. Codd's original definition notwithstanding, and contrary to the usual definition in mathematics, there is no ordering to the elements of the tuples of a relation. Instead, each element is termed an attribute value. An attribute is a name paired with a domain (nowadays more commonly referred to as a type or data type)

   2) What is an attribute value? According to The Wikipedia

      An attribute value is an attribute name paired with an element of that attribute's domain, and a tuple is a set of attribute values in which no two distinct elements have the same name. Thus, in some accounts, a tuple is described as a function, mapping names to values

   3) What is a domain? According to The Wikipedia

      a data domain refers to all the values which a data element may contain

      What exactly does this mean? Well I will use the example provided on the Wikipedia site

      For example, a database table that has information about people, with one record per person, might have a "gender" column. This gender column might be declared as a string data type, and allowed to have one of two known code values: "M" for male, "F" for female, and NULL for records where gender is unknown or not applicable. The data domain for the gender column is: "M", "F

3) Second normal form - according to the Wikipedia is

A relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key

What is a 'candidate' key? According to one source (techopedia.com) it is

a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data

4) Third normal form - according to the Wikipedia an attribute is in third normal form

if and only if both of the following conditions hold:

The relation R (table) is in second normal form (2NF)

Every non-prime attribute of R is non-transitively dependent on every key of R.

What about extended normal forms? See the Wikipedia definitions

If you have read everything and have arrived at this point I will continue on with my example of normalising the 2 documents But first I have to start with the 'Unnormalised' data items from the:

1) Invoice (invoice_number, tax_invoice_number, date, seller_name, seller_address, seller_VAT_number, buyer_name, buyer_address, ((stock _number, description, quantity, unit_price, invoice_line_amount)), total_sale_amount, discount, output_ vat_amount, invoice_total)
2) Order (rush_indicator, date_needed, partial_indicator, order_date, ((item_number, description, qty, price, total)) merchandise_total, merchandise_total_both_pages, shipping_and_handling, state_sales_tax, grand_total, payment_check_box, card_number, expiry_date, vr_number, [billing_information (name, phone_number, address, city, state, zip)], [shipping_information (name, phone, e-mail, fax, agency, street_address, city, state, zip])

I will now take you through the normalisation steps and it should soon become very clear, very quickly the number of assumptions the data modeller will have to make which could compromise the design from the very outset.

Are you still with me or have I totally lost you? Think about this, not only does this not help but suddenly you are asked to examine the possible properties of the data item (in the example you will need to know that the gender column might be declared as a 'string data type'). This means you will have to examine every attribute in the 2 lists I have just provided in order to ascertain which attributes are in 1st normal form. In this particular example there are 50+ attributes. One further word of caution: what about repeating groups? They too need to decomposed into first normal form by removing the repeating group from the rest and to create a separate data grouping.

Are you still willing to continue with this excruciating painful approach?
No? Be prepared to learn how I develop the knowledge model instead.
Yes? Continue to suffer

As far as I can see the only attributes that may need to be examined and changed into 1st normal form are in the:

1) Invoice document namely seller_name, seller_address, buyer_name, buyer_address and
2) Order document namely billing_information_name, billing_information_address, shipping_information_name, shipping_information_address. What about phone numbers?
3) The repeating group in the invoice document, namely ((stock _number, description, quantity, unit_price, invoice_line_amount))
4) The repeating group in the order document, namely ((item_number, description, qty, price, total))

You will now need to ensure that these attributes are all in 1st normal form. So this is where it gets really tricky as not every data modeller would agree with me, nevertheless the task has to be undertaken for every 'unnormalised' attribute in every document. Miss one and you could be in for a 'bumpy ride'.

Seller_name could become organisation_name, organisation_trading_name
Seller_address could become street_number, street_name, po_box_number
Buyer_name could become buyer_first_name, buyer_surname (or in the case where a buyer is another organisation organisation_name, organisation_trading_name)
Shipping_information_name could become the same as the seller_name except with the prefix seller_name
((stock _number, description, quantity, unit_price, invoice_line_amount)) could create a new group which will have to be named (example invoice_stock or invoice_line)
((item_number, description, qty, price, total)) could create a new group which will have to be named (example order_stock or order_line)
For the new repeating groups you will have to create a new 'domain' key, which is normally a combination (or compound) key and remember to remove the repeating groups from the original group.

Let us assume you manage to do this. What is the next step? Well you now need to go through this exercise again and ensure that every group (which can be called an 'entity') is in second and finally third normal form and possibly in one of the extended normal forms, many of which were discovered by people other than Codd and Boyce.

I am not going to bore you with the rest of this exercise, for if you are not convinced at this stage that normalisation is not the right way to proceed with modelling data, then I have no clue as to how to convince you that you will be wasting your and your business operative's time by pursuing this banal activity.

Is it not time to avoid normalising data and use the knowledge model instead?

Charles Meyer Richter
Principal information architect and diagnostician
Ripose Pty Limited
charles.richter@ripose.com

e&oe