30 Oct 2016

Knowledge crafting and security

A little knowledge goes a long way to secure your data

In my previous article I introduced the idea of using the container approach to build the knowledge base and using it to design the databases

I would now like to expand on that idea and demonstrate how anyone with a bit of nous can use a single select statement to breach the security of an entire database if the navigation pathway to the database is known by all and sundry (programmers as well as operatives trained in using the query language).

A database designed using Codd's 3rd or BC (Bryce-Codd) normal form opens up the access to the database structure to a great number of people. The threat of a breach is reduced dramatically by having firstly developed the corporate 'knowledge' model (which contains all the navigation pathways) secondly using the 'knowledge' model to develop the 'data' models and thirdly generate the databases.

I will now use the example from my previous article to illustrate this point.

Below is an example of the document that was used to design both the knowledge model and the 3rd normal form 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

I will now demonstrate the 2 design approaches and how the corporate knowledge model provides the corporation with the best form of security.

Codd's 3rd normal form design

As the data items on the invoice ara as follows (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). The database design could look like this:

Entity list

Customer – customer#, buyer name, buyer address

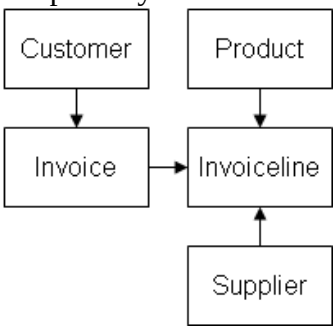
Supplier – seller#. seller name, seller address, seller VAT number

Product - stock number, description, unit price, vat percentage

Invoice - invoice number, tax invoice number, date, total sale amount, discount, output vat amount, invoice total, customer#

Invoice_line – invoice#, stock number, quantity, invoice line amount, supplier#

Graphically



Hence to get all the details for all customers all that is required is the following SQL select statement -select * from customer;

The knowledge model approach

This is not the complete knowledge model and I will not try to produce the graphical illustration of it as I would not have enough space on the A4 page to fit it, however, this list should serve to illustrate my point.

Identity – id#
- Legal entity – LE name
  - Person – person first name
  - Staff(F) – staffnumber password
  - Organisation - VAT number
  - Debtor(F)
  - Creditor(F)

Offering – offering#, offering name, offering description, vat percentage
- Product - stock number

Document – document#, date,
- Financial document - total amount, discount, output vat amount, fin doc total
  - Invoice - tax invoice number

Demand – demand#, quantity, document#(fk), offering#(fk)
- Financial demand line,- fin line amount
  - Invoice line

Location – location#
- Address

Residency – residency#, location#(fk), id#(fk)
- Home address
- Business address
- Billing address(F)

Transaction – transaction# - data, time
- Access transaction – id# (staff), id# (customer)

Hence even if the physical database design was the same as in the first design, with the exception of having a transaction table (Xaction), no programmer would ever have to know the structure and to get access to the customer details every program would have to use the GetCustomerDetails(customer#, staffnumber, password) function call to access the data about a customer's details (note the staff member would also be stored in the customer file as they too could buy something. The inclusive nature (F) would indicate that the customer was a staff member) as follows:

```
GetCustomerDetails(customer#, staffnumber, password) {
// A called function that has to pass the password of the person requesting the data
// the pseudo code would look like this
Write Xaction record – record date, time, and id# (staff); (record the access)
Read customer using staffnumber and password; (To verify staff member exits)
If valid{
Read customer file using customer number;
Return (details);
}
else{
Report security breach;}
}
```

All that has to be enforced is to disallow all external sql statements from being executed on the network. A better method would be not to use sql, but as this is now an industry standard, perhaps this will be a bit more difficult.

I trust that I have now demonstrated the power of the knowledge model and the weakness of the 3rd normal form database design without it.

As any organisation will have over 150 of these knowledge classes, you had better have an intelligent repository to store them. Caspar is my AI engine which does this very efficiently and effectively.

e&oe

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