

Re: Table Design Question

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-04/4933.html>

From: Alan Howard (Xalan.howardX_at_Xparadise.net.nzX)

Date: 04/27/04

Date: Wed, 28 Apr 2004 08:33:34 +1200

Yep – thanks for providing your perspective Joe – an interesting read.

Alan

"Joe Celko" <jcelko212@earthlink.net> wrote in message
news:%23uGobCBLEHA.1264@TK2MSFTNGP12.phx.gbl...

> >> *I'm interested in what you're implying here – care to elaborate?* <<

>

> *The regulars are cringing now :)*

>

> *A table is a set of things or relationships of the same kind. A key is*

> *a subset of attributes in the entities being modeled which uniquely*

> *identify each element of the set of entities. It is part of*

>

> *An auto-numbering is based on a PHYSICAL location or state inside a*

> *machine. It has absolutely nothing to do with the data model or the*

> *reality of the data.*

>

> *A key should be verifiable within itself. That means that when I see a*

> *particular kind of identifier, I ought to know if it is syntactically*

> *correct. For example, I know that ISBN 0-486-60028-9 has the correct*

> *number of digits and that the check digit is correct for a proper*

> *International Standard Book Number. Later on I can find out that it*

> *identifies the Dover Books edition of AN INVESTIGATION OF THE LAWS OF*

> *THOUGHT by George Boole.*

>

> *An identifier should have repeatable verification against the reality*

> *that you are trying to capture in your data model. If I put the same*

> *data into another database, do I get the same auto-increment number?*

> *Nope!*

>

> *Exactly what verification means can be a bit fuzzy. At one extreme,*

> *prison inmates are moved by taking their fingerprints at control points*

> *and courts want DNA evidence for convictions. At the other end of the*

> *spectrum, retail stores will accept your check on the assumption that*

> *you look like your driver's license photograph.*

>

microsoft.public.sqlserver.programming: Re: Table Design Question

- > *What you are doing is faking a network database using IDENTITY for*
- > *pointers, instead of creating an RDBMS.*
- >
- > *Let me go ahead one more step and play Q&A with the direction I think*
- > *you are going:*
- >
- > *Q: Couldn't a compound key become very long?*
- >
- > *A1: So what? This is the 2000's century and we have much better*
- > *computers than we did in the 1950's when key size was a real physical*
- > *issue. What is funny to me is the number of idiots who replace a*
- > *natural two or three integer compound key with a huge GUID that no human*
- > *being or other system can possibly understand because they think it will*
- > *be faster and easy to program.*
- >
- > *A2: This is an implementation problem that the SQL engine can handle.*
- > *For example, Teradata is an SQL designed for VLDB apps that uses hashing*
- > *instead of B-tree or other indexes. They guarantee that no search*
- > *requires more than two probes, no matter how large the database. A tree*
- > *index requires more and more probes as the size of the database*
- > *increases.*
- >
- > *A3: A long key is not always a bad thing fro performance. For example,*
- > *if I use (city, state) as my key, I get a free index on just (city). I*
- > *can also add extra columns to the key to make it a super-key when such a*
- > *super-key gives me a covering index (i.e. an index which contains all of*
- > *the columns required for a query, so that the base table does not have*
- > *to be accessed at all).*
- >
- > >> *Do you then advocate never using an Identity attribute? Or is it*
- > *acceptable (in the relational model) to have an Identity attribute to*
- > *use as a handle to the row, and for attributes in other tables to use as*
- > *the target for a foreign key? <<*
- >
- > *A handle to the row? Oh, you mean faking a sequential file's positional*
- > *record number, so I can reference the physical storage location? Sure,*
- > *if I want to lose all the advantages of an abstract data model, SQL set*
- > *oriented programming, carry extra data and destroy the portability of*
- > *code!*
- >
- > *More and more programmers who have absolutely no database training are*
- > *being told to design a database. They are using GUIDs, IDENTITY, ROWID*
- > *and other proprietary auto-numbering "features" in SQL products to*
- > *imitate either a record number (sequential file system mindset) or OID*
- > *(OO mindset) since they don't know anything else.*
- >
- > *Experienced database designers tend toward intelligent keys they find in*
- > *industry standard codes, such as UPC, VIN, GTIN, ISBN, etc. They know*
- > *that they need to verify the data against the reality they are modeling.*
- > *A trusted external source is a good thing to have.*
- >

- > *The IDENTITY column is a holdover from the early programming languages*
- > *which were very close to the hardware. For example, the fields (not*
- > *columns; big difference) in a COBOL or FORTRAN program were assumed to*
- > *be physically located in main storage in the order they were declared in*
- > *the program. The languages have constructs using that model -- logical*
- > *and physical implementations are practically one! The data has meaning*
- > *BECAUSE of the program reading it (i.e. the same bits could be a*
- > *character in one program and be an integer in another).*
- >
- > *The early SQLs were based on existing file systems. The data was kept*
- > *in physically contiguous disk pages, in physically contiguous rows, made*
- > *up of physically contiguous columns. In short, just like a deck of*
- > *punch cards or a magnetic tape. Most programmer still carry that mental*
- > *model, which is why I keep doing that rant about file vs. table, row vs.*
- > *record and column vs. field.*
- >
- > *But physically contiguous storage is only one way of building a*
- > *relational database and it is not the best one. The basic idea of a*
- > *relational database is that user is not supposed to know *how* or*
- > **where* things are stored at all, much less write code that depends on*
- > *the particular physical representation in a particular release of a*
- > *particular product on particular hardware at a particular time.*
- >
- > *One of the biggest errors is the IDENTITY column (actually property, not*
- > *a column at all) in the Sybase/SQL Server family. People actually*
- > *program with this "feature" and even use it as the primary key for the*
- > *table! Now, let's go into painful details as to why this thing is bad.*
- >
- > *The first practical consideration is that IDENTITY is proprietary and*
- > *non-portable, so you know that you will have maintenance problems when*
- > *you change releases or port your system to other products. Newbies*
- > *actually think they will never port code! Perhaps they only work for*
- > *companies that are failing and will be gone. Perhaps their code is such*
- > *crap nobody else want their application.*
- >
- > *But let's look at the logical problems. First try to create a table*
- > *with two columns and try to make them both IDENTITY. If you cannot*
- > *declare more than one column to be of a certain data type, then that*
- > *thing is not a datatype at all, by definition. It is a property which*
- > *belongs to the PHYSICAL table, not the LOGICAL data in the table.*
- >
- > *Next, create a table with one column and make it an IDENTITY. Now try*
- > *to insert, update and delete different numbers from it. If you cannot*
- > *insert, update and delete rows from a table, then it is not a table by*
- > *definition.*
- >
- > *Finally create a simple table with one IDENTITY and a few other columns.*
- > *Use a few statements like*
- >
- > *INSERT INTO Foobar (a, b, c) VALUES ('a1', 'b1', 'c1');*
- > *INSERT INTO Foobar (a, b, c) VALUES ('a2', 'b2', 'c2');*

microsoft.public.sqlserver.programming: Re: Table Design Question

- > *INSERT INTO Foobar (a, b, c) VALUES ('a3', 'b3', 'c3');*
- >
- > *To put a few rows into the table and notice that the IDENTITY*
- > *sequentially numbered them in the order they were presented. If you*
- > *delete a row, the gap in the sequence is not filled in and the sequence*
- > *continues from the highest number that has ever been used in that column*
- > *in that particular table. This is how we did record numbers in*
- > *pre-allocated sequential files in the 1950's, by the way. A utility*
- > *program would then "pack" or "compress" the records that were flagged as*
- > *deleted or unused to move the empty space to the physical end of the*
- > *physical file.*
- >
- > *But now use a statement with a query expression in it, like this:*
- >
- > *INSERT INTO Foobar (a, b, c)*
- > *SELECT x, y, z*
- > *FROM Floob;*
- >
- > *Since a query result is a table, and a table is a set which has no*
- > *ordering, what should the IDENTITY numbers be? The entire, whole,*
- > *completed set is presented to Foobar all at once, not a row at a time.*
- > *There are (n!) ways to number (n) rows, so which one do you pick? The*
- > *answer has been to use whatever the *physical* order of the result set*
- > *happened to be. That non-relational phrase "physical order" again!*
- >
- > *But it is actually worse than that. If the same query is executed*
- > *again, but with new statistics or after an index has been dropped or*
- > *added, the new execution plan could bring the result set back in a*
- > *different physical order.*
- >
- > *Can you explain from a logical model why the same rows in the second*
- > *query get different IDENTITY numbers? In the relational model, they*
- > *should be treated the same if all the values of all the attributes are*
- > *identical.*
- >
- > *Using IDENTITY as a primary key is a sign that there is no data model,*
- > *only an imitation of a sequential file system. Since this "magic,*
- > *all-purpose, one-size-fits-all" pseudo-identifier exists only as a*
- > *result of the physical state of a particular piece of hardware at a*
- > *particular time as read by the current release of a particular database*
- > *product, how do you verify that an entity has such a number in the*
- > *reality you are modeling?*
- >
- > *You will see newbies who design tables like this:*
- >
- > *CREATE Drivers*
- > *(driver_id IDENTITY (1,1) NOT NULL PRIMARY KEY,*
- > *ssn CHAR(9) NOT NULL REFERENCES Personnel(ssn),*
- > *vin CHAR(17) NOT NULL REFERENCES Motorpool(vin));*
- >
- > *Now input data and submit the same row a thousand times, a million*

> times. Your data integrity is trashed. The natural key was this:
>
> CREATE Drivers
> (ssn CHAR(9) NOT NULL REFERENCES Personnel(ssn),
> vin CHAR(17) NOT NULL REFERENCES Motorpool(vin),
> PRIMARY KEY (ssn, vin));
>
> To demonstrate, here is a typical idiot newbie schema -- you will them
> all over the news groups. I call them "idiots" because they always name
> the IDENTITY property column "id" in EVERY table. They don't understand
> basic data modeling -- one and only name for an attribute. About half
> the time they don't use any DRI, but let's show it.
>
> CREATE TABLE MotorPool
> (id IDENTITY (1,1) NOT NULL PRIMARY KEY,
> ssn CHAR(9) NOT NULL REFERENCES Personnel(id),
> vin CHAR(17) NOT NULL REFERENCES Vehicle(id));
>
> CREATE TABLE Personnel
> (id IDENTITY (1,1) NOT NULL PRIMARY KEY,
> ssn CHAR(9) NOT NULL UNIQUE,
> ..);
>
> CREATE TABLE Vehicles
> (id IDENTITY (1,1) NOT NULL PRIMARY KEY,
> vin CHAR(17) NOT NULL UNIQUE,
> ..);
>
> Now change the natural key in Personnel:
>
> UPDATE Personnel
> SET ssn = '6666666666'
> WHERE ssn = '000000000';
>
> Nothing happened in Motorpool, did it? You can do the same thing with a
> VIN.
>
> Now you are REALLY thinking about relations and keys instead of 1950's
> sequential record numbering. Adding an IDENTITY column to either of
> these tables as a candidate key would be dangerously redundant; one
> query uses the IDENTITY and another uses the real key, and like a man
> with two watches, you are never sure what time it is.
>
> Finally, an appeal to authority, with a quote from Dr. Codd: "..Database
> users may cause the system to generate or delete a surrogate, but they
> have no control over its value, nor is its value ever displayed to them
> .."(Dr. Codd in ACM TODS, pp 409-410) and Codd, E. (1979), Extending
> the database relational model to capture more meaning. ACM Transactions
> on Database Systems, 4(4). pp. 397-434.
>
> This means that a surrogate ought to act like an index; created by the

microsoft.public.sqlserver.programming: Re: Table Design Question

> user, managed by the system and NEVER seen by a user. That means never
> used in queries, DRI or anything else that a user does.

>

> Codd also wrote the following:

>

> "There are three difficulties in employing user-controlled keys as
> permanent surrogates for entities.

>

> (1) The actual values of user-controlled keys are determined by users
> and must therefore be subject to change by them (e.g. if two companies
> merge, the two employee databases might be combined with the result that
> some or all of the serial numbers might be changed.).

>

> (2) Two relations may have user-controlled keys defined on distinct
> domains (e.g. one uses social security, while the other uses employee
> serial numbers) and yet the entities denoted are the same.

>

> (3) It may be necessary to carry information about an entity either
> before it has been assigned a user-controlled key value or after it has
> ceased to have one (e.g. and applicant for a job and a retiree).

>

> These difficulties have the important consequence that an equi-join on
> common key values may not yield the same result as a join on common
> entities. A solution – proposed in part [4] and more fully in [14] – is
> to introduce entity domains which contain system-assigned surrogates.
> Database users may cause the system to generate or delete a surrogate,
> but they have no control over its value, nor is its value ever displayed
> to them....." (Codd in ACM TODS, pp 409–410).

>

> References

>

> Codd, E. (1979), Extending the database relational model to capture more
> meaning. ACM Transactions on Database Systems, 4(4). pp. 397–434

>

> --CELKO--

> =====

> Please post DDL, so that people do not have to guess what the keys,
> constraints, Declarative Referential Integrity, datatypes, etc. in your
> schema are.

>

> *** Sent via Developersdex <http://www.developersdex.com> ***

> Don't just participate in USENET...get rewarded for it!