Die ID, die ganze ID und nichts als die ID

Datenmodellierung

Thilo Riegel, München

Häufige Beobachtung: ID falsch verwendet

Es ist immer sehr gefährlich, sich abfällig über Kollegen zu äußern, noch dazu pauschal, aber in diesem Fall geht es nicht anders. Ich muss mich hier mal aus dem Fenster lehnen:

Ich arbeite seit jeher in der IT. Wie in jedem Fach, so gibt es auch in der IT ein gewisses Grund-Basiswissen der meisten, die sich auf dem Feld tummeln; auch bei denen, die nicht mehr „aktiv" sind. Einem Krankenhaus-Leiter müssen sie gewisse Grundlagen der Medizin nicht erklären, auch wenn er selbst schon lange nicht mehr selbst operiert. In der IT gibt es dazu Parallelen: Auch denjenigen, die nicht mehr selbst ganz tief in den Gedärmen der Datenbanken oder des Codes arbeiten (was ich selbst übrigens auch selbst nur noch selten mache), müssen Sie im Allgemeinen nicht mehr Grundlagen wie die z.B. die grundlegende Funktionsweise einer relationalen Datenbank erklären.

Umso mehr wundert es mich immer wieder, wie wenig so genannte IT-Experten davon verstehen, was eine ID ist.

Übliche Fehler, die ich immer wieder sehe, sind:

- IDs mit Semantik
- Nummernkreise
- (Zu frühes) Recycling von IDs
- Timestamps statt IDs
- Teilw. Folge der oberen Punkte: Ineffiziente IDs (ewig lange String-Ketten)
- Keine ordentliche Trennung von fachlichen und technischen IDs

Das sind alles keine echten IDs!

Woran liegt das? Ich habe da meine private Theorie dazu. Mit dazu beitragen tun folgende häufigen Missverständnisse:

- Die ID als Zahl muss „die" / eine logische Reihenfolge wiedergeben (also Vermischung von statischer ID und teilw. Statischer / teilw. dynamischer Folge-Nr.).
- IDs müssen leicht zu merken / „schön" sein. Z.B. sollten sie gewisse fachliche / organisatorische Aspekte widerspiegeln.
- IDs müssen öffentlich bekannt sein.

Diese und ähnliche Missverständnisse führen aus meiner Sicht immer wieder nicht nur zu schlecht designten Systemen, sondern auch zu hohen Folgekosten.

Was ist eine „echte“ ID?

Weil ich in meinem Beruf immer viel mit Datenmodellen und Datenbanken zu tun hatte, habe ich schon ziemlich viel Gutes und leider auch nicht so gutes gesehen. Manches vom nicht so guten, glaube ich, muss im zeitlichen Kontext gesehen werden – früher waren gewisse Vorgehensweisen nötig oder zumindest angebracht, die heute einfach nur obsolet sind. Aber manch andere waren und sind einfach auch nur Unsinn.

Regeln

Komischerweise habe ich bis heute keine einfache Zusammenfassung dazu gefunden, was eine „gute“, echte ID eigentlich ausmacht. Daher hier meine eigenen Thesen dazu, die ich zu Regeln erheben möchte.

Was macht echte IDs aus?

1. Echte IDs brauchen keine Semantik. Keinerlei Semantik. Punkt. Fachliche sowieso nicht, aber auch nicht technisch-organisatorisch.

   Damit fällt auch die Spezial-Variante „Nummernkreis“ aus. Der war früher mal nötig / sinnvoll, hat aber heutzutage größtenteils seine Berechtigung verloren, wo alles jederzeit online ist und man sich somit jederzeit online eine frische ID im Zentralverzeichnis holen kann.[1]

   Fachliche und/oder organisatorische Informationen gehören in eigens dafür vorgesehene, separate Felder. Punkt.

2. Echte IDs ändern sich nicht. Nie.

   „Außer, wenn vielleicht...?“ Nie!

   „Aber was ist, wenn...?“ Nie!

   „Nun gibt es doch aber auch Fälle, wo...“ *Nie!!!*

   „Ja Himmel, aber ich habe doch schon selbst IDs gesehen, die sich ändern!“ Das sind aber keine echten IDs, sondern nur Pseudo-ID-Murks.

3. Code, der IDs nach irgendeiner Logik zusammensetzt, also IDs quasi „versteht“, ist schlechter, gefährlicher Code! Guter Code nimmt IDs einfach nur als Ergebnisse einer Suche und reicht sie weiter für diverse Lese- und Schreiboperationen.

   Eine ID wird im Grunde nie „betrachtet“, nie analysiert. Denn: sie sagt nichts aus (s. Punkt 1)! Das einzige, was sie aussagt, ist: „Ich stehe für einen bestimmten Datensatz, und ich bin eindeutig! Versuche nicht, mich zu verstehen, ich bedeute nichts, außer dass ich Referenz bin – Daher bedeutet die konkrete Ausprägung der Zahlen und Buchstaben, aus denen ich mich zusammensetze, nicht. *Read my lips*: NICHTS! Du, Programm(ierer), sollst mich nicht betrachten, sondern einfach nur weiterreichen. Funktion A liefert mich, Funktion B braucht mich als Input. Das war's. Dazwischen hat mich keiner anzuschauen und auch nicht zu 'verstehen' oder gar zu ändern!“

4. Alle Operationen an den Daten gehen ausschließlich über die ID.

5. Todsünde: (zu frühes) Recycling von IDs, das macht ständig Probleme. Am besten recycelt man gar nicht.

1 Sofern man nicht ohnehin gleich GUIDs verwendet.

Und last not least:

6. Diese Regeln sind konsequent einzuhalten. Und mit konsequent meine ich *wirklich* konsequent, total konsequent, ganz furchtbar konsequent, einfach *absolut* konsequent. Einfach ohne Ausnahme.

Wie man's nicht machen sollte

- Timestamps als IDs: Was, bitte, soll das? Dafür gibt's Timestamp-Felder, die haben ihre Berechtigung, haben aber eine andere Funktion. Ich setze ja auch nicht einen Timestamp ins Namens- oder ins Bemerkungs-Feld. ID-Generatoren sind für viele Fälle die einfachste Sache der Welt, und in komplexeren Fällen gibt's die zum Herunterladen (z.B: GUID).

  [Davon abgesehen: angebliche Super-DB-Experten begründen solch ein Vorgehen manchmal mit Performance bei Suche mit Foreign-Key-Beziehungen. Kann ich nicht nachvollziehen: Erstens ist so etwas nicht (mehr) nötig, und zweitens sowieso ein klarer Denkfehler.]

- Die groooße Lösung: Nummernkreise, Semantik, Timestamp-Logik, usw.: Wer sich für ganz super-clever hält, kombiniert das alles. Dann haben wir eine gefühlt 50-stellige ID, und die ist dann ganz klasse zu handhaben! Und bei der nächsten Umstellung / Migration /Fusion usw. kommen dann gleich wieder 10 Stellen hinzu.

  Folgendes habe ich tatsächlich mal erlebt: Eine 21-stellige ID in einem DWH-Projekt für ein paar Milliarden Datensätze einer Banken-Union. 21 Stellen mit Zahlen und Buchstaben! Damit hätte man locker das ganze Universum durch nummerieren können, hätte man es nur richtig angepackt! Die Semantik, die darin verpackt war, war natürlich erstens nicht ohne und zweitens sehr problematisch, auch wg. Historienführung. Ineffizient war's sowieso, weil die ID als String gespeichert wurde.

Und fürs Gruselkabinett die ganz heftigen Fehler (auch alles schon gesehen):

- Durchnummerieren eines Datenabzugs auf CSV-Datei, ohne dass das im Original-Bestand so geführt würde – in jeder Lieferung neu! So einen Mist zu sehen, tut körperlich weh!
- Ändern der IDs von Version zu Version (besonders beliebt bei Pseudo-IDs in Excel-Dateien, die dann in verschiedenen Versionen kursieren). Wenn man sich darauf bezieht, weiß man nie, was gemeint ist, weil so ein Held mal wieder meinte, aus ästhetischen Gründen die ID ändern zu müssen.

Wie man's stattdessen machen sollte

- Der beste, leichtest-gewichtige und effizienteste Ansatz für eine ID ist der einfachste: Ein Zähler. Fertig.

  Ein Zähler hat keinerlei Bedeutung, keine Präferenz, sagt noch nicht einmal etwas über eine Reihenfolge aus (die man theoretisch auswerten könnte)! Er ist einfach nur eine eindeutige Nummer, die den Datensatz identifiziert. Ein- für allemal. Für alle Zeiten. So eine eine ID wird einmal vergeben, und das war's!

- Ein Zähler hat außerdem wunderschöne Eigenschaften:
  - Ein ID-Vergabe-Service, der auf Zählern basiert, ist wunderbar einfach zu implementieren.

- Solch ein ID-Vergabe-Service kann auch äußerst effizient mit großen Datenmengen umgehen (i.Ggs. z.B. zu Services, die kompliziertere Konstrukte erzeugen wie z.B. zufällige Zahlen-Buchstaben-Kombinationen).

  Wenn man, etwa für einen bevorstehenden Batch-Import, eine Million IDs benötigt, dann muss man nicht eine Million IDs einzeln erzeugen, sondern setzt den Zähler einfach um eine Million hoch. Fertig, das war's. Und diese 1 Million Datensätze müssen dann nicht etwa sofort, noch vor dem nächsten regulären neuen Datensatz, importiert werden, sondern sie können *irgendwann* in der Zukunft importiert werden, wenn's gerade passt. Wenn nötig, auch erst in einem Jahr; denn die IDs sind ja für diese Daten reserviert, und zwar für immer!

Sonderaspekte

Jaja, ich kann die Einwände schon hören... Klar, auf ein paar Sachen muss man natürlich noch Acht geben. Aber ich behaupte, dass es für jeden Fall (einige zähle ich hier auf) eine saubere Lösung gibt, die die obigen Prinzipien respektiert – und Ihnen und Ihren Nachfolgern viel Ärger erspart.

- Fachliche vs. technische ID:

  Ich meinen mit meinen „schönen“ Zählern oben natürlich immer nur technische IDs.

  Es stimmt schon: Fachliche IDs (die bei u.U. auch nicht immer auf Anhieb von technischen zu unterscheiden sind) können sich natürlich schon mal – berechtigterweise – ändern. Dann muss man eben zwei Felder führen: Eine technische ID (wie beschrieben) und eine fachliche. Und bitte nicht kombinieren in einem Feld!

  Fachliche und technische IDs sind immer konsequent zu trennen.

  These 4 von oben wird dann sinngemäß präzisiert: „Alle Operationen an den Daten gehen *letztlich* ausschließlich über die *technische* ID. Operationen über die fachliche ID dürfen lediglich die Suche nach dem Datensatz (also nach seiner technischen ID) kapseln, mit der dann die eigentliche Operation durchgeführt wird.“

- Historienführung:

  Zugegeben, das Thema ist nicht ganz einfach. Aber dafür gibt’s bewährte Konzepte, die am hier gezeigten Prinzip und den obigen Regeln nichts Grundlegendes ändern. Diese hier darzulegen, würde den Rahmen des Artikels sprengen.

  Mein Punkt ist: Das Problem ist konzeptionell gelöst, und zwar sauber! Und ich habe auch schon live Systeme gesehen, wo's sauber implementiert ist.

- Migration / Fusion von Datenbeständen:

  Ebenfalls ein nicht ganz einfaches Thema. Wenn man aber genauer hinschaut, merkt man schnell:

  - Das Thema ist nur dann kompliziert, wenn die obigen Grundregeln früher nicht beachtet wurden. Ansonsten ist es nämlich relativ einfach
  - Falls die Regeln früher nicht beachtet wurden und man jetzt nun einmal vor dem Problem steht, sich eine Lösung ausdenken und implementieren zu müssen, dann ist der Ansatz wie folgt:

Die bisherige technische (ggf. auch teil-fachliche) ID wird zu einer neue fachlichen ID umdefiniert und eine Zeit lang mit geführt, und es wird per Zähler eine neuer technische ID eingeführt (bei Migration per einfachem „Durchzählen“ mit Zähler-ID).

Dieses Verfahren mag in der Praxis nicht ganz ohne sein, aber das Konzept dahinter ist sauber.

- Nummernkreise für verschiedene Organisatorische Einheiten / Mandanten usw.:

  Das ist der einzige Fall, der mir einfällt, bei dem in der Vergangenheit berechtigterweise semantische IDs eingeführt wurden. Schließlich gab es mal Zeiten, in denen nicht alles miteinander vernetzt war.

  Heute allerdings ist dieser Ansatz obsolet, und wir können das Verfahren begraben (denn es verletzt These Nr. 1). Einen zentralen ID-Generator (mit Zähler) kann man problemlos als online-Service implementieren, und damit kann sich jede Organisationseinheit / jeder Mandant usw. jederzeit beliebig viele IDs reservieren. Es wird dabei noch nicht einmal vorausgesetzt, dass die Datenbestände physisch gemeinsam gespeichert werden (aber wenn man das möchte, kann man es dank dieses Verfahrens jederzeit problemlos nachholen):

  Und wenn's doch aus bestimmten Gründen Offline-Bestände sein müssen: Dann besser über den Gebrauch von GUIDs nachdenken.

Höre ich weitere Einwände?

- Das ist nicht praktikabel / zu teuer? Quatsch! Ich habe schon Dutzende von Datenbanken nach diesem Prinzip designt und realisiert, und die größten Probleme lagen immer darin, ein paar Betonköpfe zu überzeugen, dass das der beste Weg ist.
- Ihre Migration / Fusion ist aber anders? Sorry, glaube ich Ihnen nicht. Ich habe selbst auch schon öfter Migrationen und Fusionen mit solchen Ansätzen begleitet. Das funktioniert wunderbar, man kann das auch in größeren und komplexeren Beständen sehr konsequent durchhalten.
- Diese künstliche technische ID, diese Nummer kann sich keiner merken? Na und? Muss man das? Wer hat gesagt, dass man sie veröffentlichen muss? Schon mal was von Suchfunktionalitäten gehört? Außerdem, wenn's aus bestimmten Gründen doch öffentlich sein muss: So komplex ist es ja nun auch wieder nicht. Wir reden immerhin nur von einer einzigen ganzen Zahl!
- Der heutige Code in Ihrem XYZ-Murks-Framework kann das so nicht? Ganz tolles Argument! Dann muss er eben umgeschrieben werden! (Ernsthaft, sonst kommt das Problem immer wieder, und jedes Mal wird’s noch komplizierter.)

Fazit

Die (echte) ID führt ein undankbares Dasein. Dabei ist sie so unglaublich wichtig an so vielen Stellen. Ohne sie geht nichts, und doch wird sie so oft misshandelt und vergewaltigt (= mit Gewalt unecht gemacht) von angeblichen Experten.

Und warum ist es so wichtig, dass man die Regeln – insbesondere die Konsequenz-Regel Nr. 6 – beachtet? So wichtig, dass der Autor hier einen ganzen Artikel dafür spendiert?

- Weil es, wenn man sich nicht daran hält, auf lange Sicht immer und ausnahmslos furchtbaren Aufwand und damit große Kosten verursacht.

  Das ist so ähnlich wie mit den Y2K-Jahreszahlen: Man kann eine Weile lang mit den unsauberen Lösungen leben, teilw. sogar ziemlich lang. Aber irgendwann holt einen das Thema ein. Immer, und dann aber umso heftiger! „Pragmatismus“ ist hier fehl am Platz!

- Externe Referenzen – sofern es sie gibt – müssen stabil bleiben (Zusatzlisten, Verweise aus anderen Systemen, Korrektursätze, nachgelagerte / dispositive Systeme usw.)

  Wenn es eine fachliche ID gibt, dann ist diese allerdings für externe Systeme zu bevorzugen. Wir sind uns einig: Eigentlich gehört eine technische ID nirgendwo extern gespeichert, sondern sie sollte intern bleiben. Aber manchmal ist das Veröffentlichen / externe Referenzieren eben von technischen IDs eben doch nötig. Und und in diesen Fällen ist es umso wichtiger, dass man die obigen Regeln befolgt, weil sich sonst das Problem, das alles wieder in ein paar Jahren aufräumen zu müssen, sogar noch potenziert – dann hat man nicht nur großen Aufräum-Aufwand, sondern sogar *sehr* großen!