Ein Schlüsselkonzept für das ER-Modell – praktische Konsequenzen für Modellierung und Toolunterstützung

Dipl.-Math. Beate Ritterbach
freie Methodenberaterin Software Engineering
Postfach 50 13 71, 2000 Hamburg 50

November 1992

Gliederung

Abstract

1. Einleitung

2. Das Konzept

3. Methodische Vorteile des Schlüssel-Konzeptes

4. Auswirkungen auf ein CASE-Tool
   4.1. Automatische Konsistenzprüfungen
   4.2. Automatisches Anordnen des ER-Diagramms

Abstract

Dieser Beitrag erläutert ein Schlüsselkonzept für das heute am meisten eingesetzte Datenmodell, das ER-Modell. Obwohl es ansatzweise bereits in etlichen CASE-Tools implementiert ist, findet man in der Literatur oder in Veröffentlichungen fast nichts über das zugrundeliegende Konzept. Das Ziel dieses Beitrags besteht in Verbesserungen bei der ganz alltäglichen Handhabung von Datenmodellen, darum wird vor allem aufgezeigt, welche Vorteile dieses Schlüsselkonzept für die praktische Arbeit mit sich bringt:
- größere Klarheit und Verständlichkeit bei der Modellierung von komplexen Strukturen,
- stark erweiterte Konsistenzprüfungen, die sich in einem CASE-Tool verwirklichen lassen und deren Prüfung deshalb automatisiert werden kann, und
- ein Verfahren für die Anordnung der Entity-Symbole im ER-Diagramm, das bei größeren Modellen eine bessere Übersichtlichkeit bewirkt und den Modellierer von manueller Arbeit entlastet.
"Schon wieder eine Erweiterung des ER-Modells! Das müssen wir uns nicht zum 100sten Male antun. Außerdem: das ER-Modell ist out, Objektorientierung ist angesagt."
So in etwa stelle ich mir Reaktionen auf das obige Abstract vor.

Dem aber möchte ich entgegenhalten:

2. Ein Umstieg auf objekt-orientierte Ansätze würde das Vorhandene nicht überflüssig machen, insbesondere nicht das ER-Modell. Welches Unternehmen wäre bereit, seine aufwendig (und teuer) erarbeiteten Unternehmens- und Projektdatenmodelle ohne weiteres zu begraben?
Das hier vertretene Schlüsselkonzept für das ER-Modell ist bei weitem nicht neu.
Seine Kernidee besteht darin, daß nicht nur Attribute als Bestandteile eines Schlüssels in Frage kommen (wie im relationen Datenmodell), sondern auch Beziehungen zu anderen Entities.

Obwohl Grundzüge dieses Ansatzes in etlichen am Markt etablierten CASE-Tools (ADW, IEF, ...) berücksichtigt sind, wird erstmaligerweise das zugrundeliegende Konzept in der gängigen Literatur zur Datenmodellierung fast nirgends behandelt.

Entweder wird bei der Beschreibung des ER-Modells das Konzept eines Schlüssels völlig übergangen; oder es wird die Existenz eines Schlüssels am Rande erwähnt - dann aber mit dem Schlüsselbegriff des relationen Datenmodells gearbeitet (wobei ein Schlüssel ein oder mehrere Attribute umfaßt). Oft wird dieser dann auch nur mit extrem einfachen Beispielen untermauert (Kunde - Kundennummer, Artikel - Artikelnummer), so daß nicht deutlich wird,
- wie man gerade in nichttrivialen, komplexen Situationen den Schlüssel handhabt, und
- warum ein Schlüssel für die logische Modellierung von Datenstrukturen überhaupt nützlich und notwendig ist.
(Gemeint ist hier keinesfalls ein Zugriffspfad, ein Index, eine Adressierung oder ein sonstiges für die physische Implementierung notwendiges Gebilde.)

Bevor als Basis für alle weiteren Ausführungen eine formale Definition des Schlüssels für das ER-Modell angegeben wird, soll ein Beispiel die grundlegende Idee verdeutlichen.

Vergleichwerten wir uns den folgenden, vereinfachten Ausschnitt aus einem ER-Modell:

```
  Bank
  /-------
  \      /
  \    /
  \  /
  \/
  geht zu

  Bank-Konto
```

Der Schlüssel für das Entity "Bank" ist das Attribut "Bankleitzahl", und ein Schlüssel für das Entity "Bank-Konto" wird gesucht.

Eine naheliegende und dem relationalen Modell entlehnte Lösung besteht darin, einen konkateneierten Schlüssel zu wählen, gebildet aus den beiden Attributen "Bankleitzahl" und "Kontonummer". Im Kontext des ER-Modells bringt dies aber gravierende Probleme mit sich:
Das Attribut "Bankleitzahl" gehört nicht zum Entity "Bank-Konto" und kann infolgedessen nicht so ohne weiteres als Teil des Schlüssels für "Bank-Konto" dienen.

Führt man im Entity "Bank-Konto" das Attribut "Bankleitzahl" ein - eben um den gewünschten Schlüssel bilden zu können - so erhält man lediglich ein Attribut, das mit "Bankleitzahl" im Entity "Konto" den Namen gemeinsam hat. "Geneint" war mit dem hier fehlerplazierten Attribut aber eigentlich eine Verbindung zum Entity "Bank".

Dieses zweite Attribut namens Bankleitzahl im Entity "Bank-Konto" führt zu Redundanzen, denn

- die Bankleitzahl eines jeden Kontos muß dieselbe sein wie die Bankleitzahl der Bank, zu der das Konto gehört (→ Redundanzen in den Daten).

- Bei der Modellierung selbst muß darauf geachtet werden, daß Veränderungen beim einen Attribut (Name, Wertebereich, etc.) auch im anderen Attribut nachvollzogen werden (→ Redundanzen im Modell).

Mögen derartige Redundanzen auch in einer physisch implementierten Datenbank aus speicherungstechnischen Gründen und bei heutiger Technologie unvermeidbar sein - in einem logischen Datenmodell haben sie nichts zu suchen.

Die Liste der entstehenden Probleme kann weiter verlängert werden (z. B. Rollennamen bei Schlüssel-Attributen, Dreifach-Schlüssel mit unklarer Bedeutung, zufällige Namensgleichheiten von Schlüssel-Attributen, ...). Da die grundlegende Problematik aber genügend geschildert ist, statt dessen nun zur Lösung:


Der konventionelle Schlüsselbegriff ist also dahingehend verallgemeinert, daß er zwei Arten von Bestandteilen enthalten kann: Attribute und Beziehungen.
Ausgehend von dieser, wenn auch stark zusammengefaßten, Beispielbetrachtung nun zur versprochenen allgemeingültigen Definition des Schlüssels.

Definition:

Ein Schlüssel eines Entities $E$ ist eine Menge $(A_1, \ldots, A_n, B_1, \ldots, B_n)$ von Attributen, die zu $E$ gehören, und/oder Beziehungen, an denen $E$ beteiligt ist.

$(A_1, \ldots, A_n)$ Attribute von $E$,

$(B_1, \ldots, B_n)$ Beziehungen zwischen $E$ und $E_1, \ldots, E_n$,

wobei für $i \neq j$ $E_i$ und $E_j$ identisch sein dürfen

und die folgende Bedingungen erfüllen:

1. Identifizierung
In Kombination identifizieren die Komponenten jede Ausprägung von $E$ eindeutig, d. h. gibt man ein beliebiges Tupel von Attributwerten von $A_1, \ldots, A_n$ und Ausprägungen von $E_1, \ldots, E_n$ vor, so gibt es höchstens eine Ausprägung von $E$, welche diese Werte annimmt und zu diesen Ausprägungen über $B_1, \ldots, B_n$ in Beziehung steht.

2. Existenz und Eindeutigkeit
Die Komponenten eines Schlüssels müssen vorhanden und eindeutig sein, d. h. die Beziehungen $B_1, \ldots, B_n$ haben Kardinalität $1$ / Optionalität $1$ in $E_1, \ldots, E_n$, die Attribute $A_1, \ldots, A_n$ sind einwertig und obligat.

3. Minimalität
Ein Schlüssel darf nicht unnötig überladen sein, d. h. keine echte Teilmenge des Schlüssels darf bereits die Identifizierungsbedingung erfüllen.

4. Kreisfreiheit
Schlüssel-Beziehungen dürfen nicht kreisförmig geschlossen sein, d. h. kein Entity darf direkt oder indirekt Bestandteil seines eigenen Schlüssels sein.

5. Unveränderbarkeit
Schlüsselkomponenten sind "fixed", d. h. die Werte der Schlüssel-Attribute dürfen sich während der Lebensdauer einer Ausprägung nicht verändern, ebenso darf die Verknüpfung mit den Ausprägungen von $E_1, \ldots, E_n$ nicht geändert werden.

Die Attribute $A_1, \ldots, A_n$ heißen Schlüsselattribute, die Beziehungen $B_1, \ldots, B_n$ Schlüsselbeziehungen von $E$.

$E$ selbst wird als identifiziertes Entity, und $E_1, \ldots, E_n$ als identifizierende Entities bezeichnet.
Diese - ungünstlicherweise sehr umfangreiche und theoretisch klingende - Definition ist der Vollständigkeit halber notwendig.

Da aber das Ziel dieses Beitrages darin besteht, die Konsequenzen dieses Schlüsselbegriffs und seine praktische Nützlichkeit in den Vordergrund zu stellen, wird auf weitere theoretische Erörterungen verzichtet. (Hier wären möglich und wünschenswert:
- detaillierte Erläuterung der einzelnen Bedingungen in obiger Definition
- ein Vergleich mit dem Schlüsselbegriff des relationalen Datenmodells
- Diskussion alternativer Lösungsansätze wie das Zulassen gemeinsamer Attribute oder die Verwendung von surrogate keys, u. v. m.).
Es wird hiermit lediglich darauf hingewiesen, daß derartige theoretische Fundierung vorhanden und an anderem Orte einsehbar ist.

3. Methodische Vorteile des Schlüssel-Konzeptes

Größere Klarheit und Zwang zu gründlicherer Analyse


Der Schlüssel dient im logischen Modell in erster Linie dazu, deutlich zu machen, was mit einem modellierten Entity überhaupt gemeint ist. Der Name des Entities allein - auch wenn er sehr sorgfältig gewählt wurde - kann dies in der überwiegenden Zahl der Fälle nämlich überhaupt nicht leisten. Beispiele: "Abwesenheit", "Belegung", "Kontakt", "Tarif", "Vermeidung", ...
All diese Entities habe ich Datenmodellen aus der praktischen Arbeit entnommen, und sie belegen die Behauptung sehr deutlich.

Eine längere verbale Beschreibung zu jedem Entity würde nur wenig Abhilfe schaffen,
- weil sie nicht Bestandteil des eigentlichen Modells ist, sondern Dokumentation am Rande,
- weil sie selbst rein verbal abgefaßt wird und damit ebenfalls mißverständlich und mehrdeutig sein kann, und
- weil sie umständlicher zu handhaben und nicht auf den ersten Blick im ER-Diagramm sichtbar ist.
Häufig wird erst in Zusammenhang mit anderen Entities des Modells deutlich, was ein Entity wirklich bedeutet - oder besser, laut Konsens des Projekt-Teams bedeuten soll. Und hier kommen Schlüssel, die Beziehungen zu anderen Entities enthalten dürfen, zum Tragen.


Mehrere Beispiele sollen jetzt veranschaulichen, auf welche Art und Weise das Schlüsselkonzept zu mehr Klarheit verhelft.

### Beispiel 1

![ER-Diagramm]

Was ist mit einem Fahrzeug gemeint?
Ein einzelner Gegenstand, der auf der Straße bewegt werden kann, z. B. mein Auto?
Oder vielmehr ein Fahrzeug-Typ - dementsprechend wäre "Daihatsu Cuore" ein Fahrzeug, und "Porsche 911" ein anderes Fahrzeug, unabhängig davon, wie viele einzelne Exemplare davon jeweils gerade existieren.
(Das Beispiel ist nicht fiktiv, sondern eines von etlichen Mißverständnissen aus der Modellierungspraxis ohne Schlüssel. Nach einer halben Stunde intensiver Diskussion bemerkten wir anhand allerhand Ungereimtheiten, daß wir mit "Fahrzeug" unterschiedliches meinen.)

Legt man den Schlüssel des Entities von vornherein fest, so sind die Chancen des Mißverständnisses weitaus geringer:
Meint man das erste, so könnte man beispielsweise das Attribut "Fahrgestellnummer" als Schlüssel verwenden. Im zweiten Fall bietet sich z. B. die Beziehung "...gebaut von - Hersteller" (Daihatsu, Porsche, ...) in Verbindung mit dem Attribut "Typen-Bezeichnung" (Cuore, 911, ...) als Schlüssel an.
Was ist "eine" Reise?
- Kann ein- und dieselbe Reise von mehreren Kunden gebucht werden? Oder wären das dann schon verschiedene Reisen?
- Handelt es sich noch um dieselbe Reise, wenn der Kunde sie unter Reibehalt aller sonstigen Rahmenbedingungen – um zwei Wochen verschiebt? Oder verlängert?
- Ist es dieselbe Reise, wenn ein anderer Veranstalter sie anbietet?
- Wäre es noch dieselbe Reise, wenn nur das Verkehrsmittel verändert würde und der Kunde seinen Zielpunkt z. B. statt mit dem Flugzeug mit der Bahn erreicht?

Wird Entity "Reise" ohne Schlüssel modelliert, oder mit einem Attribut "Reise-Nr." als Schlüssel, so ist die Gefahr groß, daß mehrere Modellierer in der Analyse völlig aneinander vorbeireiten. Sie verstehen unter "Reise" völlig unterschiedliche Sachverhalte, bemerken es aber nicht im geringsten – das, was jedem individuell vorschwebt und was jeder für sich als Reise bezeichnet, ist ja modelliert worden.

Die Forderung nach einem aussagefähigen Schlüssel deckt solche Unklarheiten auf, hilft aber auch, sie zu beseitigen.

Wählt man etwa "durchgeführt von – Veranstalter" und "führt nach – Ort" als Schlüssel, so können mehrere Kunden dieselbe Reise buchen, die Reise kann mehrfach durchgeführt werden, zu verschiedenen Zeiten, verschieden lange, aber nicht von unterschiedlichen Veranstaltern, nicht zu verschiedenen Orten, etc.

Oder verwendet man als Schlüssel "führt nach – Ort" und "gebucht von – Kunde" sowie "Beginndatum", dann würde die Reise eines anderen Kunden zum selben Ort, am selben Datum, betreut von demselben Veranstalter, per Definition eine andere Reise sein. Ebenfalls erhält man hier eine andere Reise, wenn der Kunde den Start-Term in um eine Woche verschiebt, etc.

Natürlich gibt es diverse weitere Möglichkeiten, was man unter einer Reise verstehen kann und es durch die Wahl des Schlüssels zum Ausdruck zu bringen.
Die beiden angesprochenen Alternativen zeigen bereits sehr deutlich, wie mit der Wahl des Schlüssels die Bedeutung des Entities "Reise" festgelegt wird. Und es soll nochmals betont werden, daß keine der Lösungen "richtig" oder "falsch" ist. Wesentlich ist, festzulegen, was man eigentlich meint, bzw. hierüber bestehende Differenzen frühzeitig aufzudecken.

Beispiel 3

Das hier vorliegende Modellierungsproblem gehört in der Praxis zum Alltag:
Die Information darüber, an welchem Wochentag und um welche Uhrzeit ein Kurs in welchem Raum stattfindet, soll zusätzlich im Modell untergebracht werden. Da es sich dabei zunächst um versteckte Attribute handelt, muß die m:m-Beziehung zwischen "Kurs" und "Raum" aufgelöst werden.

Die Lösung dieser Standardaufgabe sieht folgendermaßen aus:

Die m:m-Beziehung wird entfernt und statt dessen ein neues Entity eingefügt (hier "Kurs-Termin" genannt). Dieses wird mit den ursprünglichen Entities mit neuen Beziehungen, nun vom Typ 1:m, verbunden.
Die Attribute "Wochentag" und "Uhrzeit" sind nun beschreibende Attribute des Entities "Kurs-Termin", und der Schlüssel von "Kurs-Termin" besteht aus den beiden Beziehungen "gehört zu - Kurs" und "findet statt in - Raum".

Nun wird der Schlüssel des neuen Entities überprüft und verglichen mit dem, was "Kurs-Termin" bedeuten soll. Hier stellt sich heraus, daß der in einem ersten, rein mechanischen Schritt vergebene Schlüssel dem vorliegenden Sachverhalt nicht angemessen ist:

Der Zeitpunkt (Wochentag und Uhrzeit) des Kurs-Termins hängt nicht von Kurs und vom Raum ab. Sondern umgekehrt, von Kurs und vom Zeitpunkt hängt ab, in welchem Raum der Kurs-Termin stattfindet.

(Beispiel: Der Kurs "Endliche Automaten" kann im "Hörsaal A" durchaus dienstags um 11 Uhr und freitags um 13 Uhr stattfinden. Möchten Sie dagegen den Kurs "Endliche Automaten" besuchen und es ist Freitag, 13 Uhr, so erfahren Sie eindeutig, daß Sie planmäßig in den "Hörsaal A" gehen müssen.)

Will man dies abbilden, so lautet ein korrekter Schlüssel für "Kurs-Termin":

"...gehört zu - Kurs", "Wochentag" und "Uhrzeit",
er enthält also eine Beziehung und zwei Attribute.
Die Beziehung "Kurs-Termin - findet statt in - Raum" wird zur Nicht-Schlüssel-Beziehung "degradiert".

Die Forderung nach einem angemessenen Schlüssel zwingt auch hier dazu, jeden Sachverhalt so zu modellieren, daß das Gemeinte von allen Projektbeteiligten gleichermaßen nachvollzogen werden kann, und zwar in dem Präzisionsgrad, der dafür benötigt wird. Vage, "unausgegrenzte" Entities werden so vermieden.

Anmerkung:
Drei Aspekte sind an diesem Fallbeispiel so interessant, daß ich der Versuchung nicht widerstehen kann, sie zu erläutern, auch wenn sie mit den Vorteilen des hier vertretenen Schlüsselkonzeptes nur am Rande zu tun haben.

1. Der Name "Kurs-Termin" für das eingefügte Entity klingt ziemlich gestelzt und nichtssagend; es ist schwierig, für den modellierten Sachverhalt überhaupt einen Namen zu finden. Dies ist kein Charakteristikum des gewählten Beispiels, sondern bei der Auflösung von mm-Beziehungen sehr häufig anzutreffen. Gerade dann aber zeigt der Schlüssel um so deutlicher, was das Entity bedeuten soll. Dadurch, daß er aus "...gehört zu - Kurs", "Wochentag" und "Uhrzeit" besteht, ist im Grunde alles gesagt - und zwar präziser, als ein sorgfältig gewählter Name oder eine verbale Beschreibung das hier könnten.

3. Wird eine n:m-Beziehung aufgelöst, so erhält das neu eingefügende Entity als Schlüssel immer die beiden 1:m-Beziehungen zu den beiden "alten" Entities:

Das Entity C ist dann assoziativ.

Das obige Beispiel, in dem nach der Auflösung der n:m-Beziehung der Schlüssel verändert werden mußte, ist nur eine scheinbare Ausnahme dieser (mathematisch beweisbaren!) Umformungsregel.
Vielmehr war die Modellierung einer einfachen n:m-Beziehung zwischen "Kurs" und "Raum" dem komplexen Zusammenhang zwischen Kursen, Räumen, Zeitpunkten und Kurs-Terminen nicht angemessen.

Fazit:
Zum einen ermöglicht das Schlüssel-Konzept, klar und mit formalen Mitteln festzulegen, was unter einem Entity verstanden werden soll.
Zum anderen erzwung es auch derartig klare Aussagen innerhalb des Modells und trägt somit zur konstruktiven Qualitätssicherung und zu frühzeitiger Beseitigung von Mehrdeutigkeiten bei.
Der vorangegangene Abschnitt hat die konzeptionellen Vorteile des hier vertretenen Schlüsselkonzeptes aufgezeigt: größere Ausdrucksmöglichkeit und Klarheit.

Nun soll gezeigt werden, daß das Schlüssel-Konzept auch Ansatzpunkte für Toolunterstützung bietet und so den Modellierer von aufwendiger manueller Arbeit entlasten kann. Hier sind zwei wesentliche Aspekte zu nennen:

1. Automatische Konsistenzprüfungen bzw. konstruktive Elemente in der Toolfunktionalität (→ Abschnitt 4.1)
2. Automatisches Anordnen des ER-Diagramms auf Basis der Informationen in den Schlüsseln (→ Abschnitt 4.2)

### 4.1. Automatische Konsistenzprüfungen

Am Markt befindlichen CASE-Tools haben sich häufig den Vorwurf gefallen lassen müssen, reine Zeichen-Werkzeuge zu sein. Welche Funktionalität eines CASE-Tools aber kann über die eines Zeichen-Werkzeugs hinausgehen? Hier sind zu nennen

- Konsistenzprüfungen und qualitätssichernde Maßnahmen, d. h. Abweisen oder zumindest Aufspüren von inkonsistenten Informationen,
- komfortable Such-, Auswertungs- und Dokumentationsunterstützung,
- Generierung von Ergebnistypen für Folgephasen.

Auf die beiden letztgenannten Punkte soll hier nicht näher eingegangen werden, denn die Umsetzung des ER-Modells in ein relationales Modell oder eine DDL ist in etlichen CASE-Tools mehr oder minder zufriedenstellend gelöst worden. und komfortable Dokumentationsunterstützung hat mit dem zugrundeliegenden methodischen Konzept wenig zu tun.


An dieser Stelle müssen einige Überlegungen eingeschoben werden, die auf den ersten Blick rein theoretischer Natur sind:
Jedes Modell (z. B. das ER-Modell, das relationale Datenmodell, das SA-Modell, etc.) beinhaltet Objekttypen und Regeln. Ein Objekttyp (Entity, ..., Relation, ..., Funktion, ...) ist ein Grundbaustein, sozusagen ein elementarer Bestandteil
des jeweiligen Modells. Eine Regel dagegen ist kein zusätzlicher Objekt-Typ, sondern eine Aussage darüber, wie die Grundbausteine untereinander zusammenhängen.

(Das gleiche gilt übrigens auch für Kardinalitäten und Optionalitäten: es sind Regeln, keine Objekte.)
Und einige der Regeln, die einen Schlüssel ausmachen, sind innerhalb des Modell, und damit auch algorithmisch, durch ein Werkzeug, prüfbar.

Im folgenden sollen die Vorteile herausgearbeitet werden, die das ER-Modell mit dem hier vertretenen Schlüsselkonzept gegenüber dem "konventionellen" ER-Modell (ohne Schlüsselkonzept bzw. nur mit Attributen im Schlüssel) aufweist.
Dazu werden die auf Werkzeug-Ebene (=Modell-Ebene) prüfbaren Regeln, die aus den beiden Ansätzen resultieren, einander gegenübergestellt.

Die prüfbaren Regeln des "konventionellen" ER-Modells sind in fast allen heute gängigen CASE-Tools implementiert. Sie gewährleisten, ...

- daß keine zwei Entities mit gleichem Namen eingegeben werden können (Homonym-Vermeidung, gilt abgewandelt auch für Attribute und Beziehungen)
- daß jedes Attribut zu genau einem Entity gehört
- daß jede Beziehung genau zwei Entities verbindet (die auch identisch sein dürfen, also rekursive Beziehungen)
- und manche überprüfen, ob die Kardinalitäten und Optionalitäten der Beziehungen und Attribute vollständig vorliegen, ob zu jedem Entity eine Beschreibung vorhanden ist, etc.


Legt man das ER-Modell mit dem hier vertretenen Schlüssel-Konzept zugrunde, so erhält man zusätzlich zur obigen Liste die folgenden modell-intern prüfbaren Konsistenz-Regeln:
- Schlüssel-Attribute müssen Kardinalität 1/ Optionalität 1 aufweisen (= sie sind einwertig und obligat)

- Schlüssel-Beziehungen müssen im identifizierenden Entity Kardinalität 1/ Optionalität 1 haben

- Schlüssel-Beziehungen müssen im identifizierten Entity Kardinalität m aufweisen, falls kein Sub-Entity, also eine IS-A-Beziehung vorliegt

- Ist eine Schlüssel-Beziehung eine IS-A-Beziehung, so muß sie im identifizierten Entity (= dem Sub-Entity) Kardinalität 1/ Optionalität 0 aufweisen

- Schlüssel-Beziehungen dürfen nicht zirkulär verlaufen.

- Besteht ein Schlüssel eines Entities aus der Menge $S = \{A_1, \ldots, A_n, B_1, \ldots, B_n\}$ von Attributen und/oder Beziehungen, dann kann keine echte Teilmenge oder Obermenge von $S$ ebenfalls Schlüssel sein.

Einige der aufgeführten Regeln sind Bestandteil der Schlüssel-Definition, andere (z. B. die dritte und die vierte) erfordern eine längere Herleitung, auf die ich hier aus Platzgründen verzichte.

(Die erste und die letzte Regel könnte übrigens schon bei Zugrundelegen des Schlüssel-Konzeptes aus dem relationalen Modell gefordert werden. Meines Wissens nach werden die hier genannten sechs Regeln von keinem der marktgängigen CASE-Tools abgeprüft.)

Das Schlüssel-Konzept läßt sich aufgrund der oben genannten Regeln sogar nutzen, um Schlüssel-Beziehungen und Schlüssel-Attribute mit ihren Kardinalitäten und Optionalitäten interaktiv zu generieren, anstatt sie vom Modellierer mühsam in allen Einzelheiten eingeben zu lassen.

Ein bereits bekanntes Beispiel soll veranschaulichen, wie die Arbeit mit einem derartigen CASE-Tool ablaufen könnte.

![Diagramm]

Abbildung: Beispiel für ein CASE-Tool.
Das vorliegende Modell enthält bereits einige Entities und Beziehungen, es soll das Entity "Kurs-Termin" zusätzlich eingefügt werden.

Der Benutzer weist das Tool an, ein neues Entity zu erstellen: "Add Entity Kurs-Termin"

-> Das Tool "erfragt" den Schlüssel von "Kurs-Termin". Dazu bietet es eine Liste der bereits vorhandenen Entities an, so daß man sie als Schlüsselkomponenten auswählen kann, außerdem bietet es ein Fenster, welches die Eingabe von mehreren Schlüssel-Attributen ermöglicht.

Zu erstellendes Entity: Kurs-Termin

Bitte bestimmen Sie die Schlüssel-Komponenten:

Beziehung(en) zu...

Attribut(e)

[Diagramm mit den Beziehungen Dozent, Fach, Kurs, Raum]

Der Benutzer wählt Entity "Kurs" als eine Komponente des Schlüssels aus. (Im Bedarfsfalle können mehrere Entities, sogar dasselbe Entity mehrfach, als Schlüsselkomponente gewählt werden.)

Der Benutzer gibt "Wochentag" als Name für ein neues Attribut ein, welches zum Schlüssel gehören soll.

Der Benutzer gibt "Uhrzeit" als Name für ein neues Attribut ein, welches ebenfalls zum Schlüssel gehören soll.

Der Benutzer bestätigt, daß hiermit die Zusammensetzung des Schlüssels für "Kurs-Termin" beendet ist ("Fertig").

-> Das Tool erstellt aufgrund dieser Angaben

- das Entity "Kurs-Termin"
- die dazu gehörenden Schlüssel-Attribute "Wochentag" und "Uhrzeit", jeweils mit Kardinalität 1 und Optionalität 1
- eine Schlüssel-Beziehung zwischen "Kurs" und "Kurs-Termin" mit Kardinalität 1/ Optionalität 1 in "Kurs" und Kardinalität m in "Kurs-Termin".

33
Der Benutzer wird nun aufgefordert, für jede generierte Beziehung (hier: eine) die Namen einzugeben sowie die noch fehlende Optionalität.

Erst damit ist das Erstellen eines neuen Entities abgeschlossen. Schlüssel-Beziehungen werden im Bedarfsfall generiert, ebenso die festgelegten Kardinalitäten und Optionalitäten aller Schlüsselkomponenten.

Der Benutzer kann das Modell im Nachhinein ändern, dabei kann er allerdings nicht

- die vom Tool vergebenen Kardinalitäten und Optionalitäten von Schlüssel-Attributen und -Beziehungen ändern

- den Schlüssel von "Kurs-Termi"n löschen (ohne einen anderen Schlüssel anzugeben)


Das Tool erzwinge somit, daß jedes Entity von vornherein (mindestens) einen Schlüssel besitzt. Daß diese Forderung sinnvoll ist, wurde in Abschnitt 3 unter Beweis gestellt. Das Tool erlaubt dem Benutzer aber auch, die Angabe des Schlüssels auf sehr komfortable Weise abzuwickeln, unter Rückgriff auf die im Modell bereits vorhandenen Entities und mit Generierung aller erschließbaren Informationen.

Zusätzlich wird ein schrittweises Vorgehen vom Einfachen zum Komplexen gefördert. Im obigen Beispiel etwa ist es nicht möglich, "Kurs-Termi"n einzugeben, bevor "Kurs" modelliert ist.
4.2. Automatisches Anordnen des ER-Diagramms

Dieser Abschnitt spricht einen besonders interessanten und nach den bisherigen Ausführungen eher unerwarteten Aspekt des Schlüsselkonzeptes an.

Wie von etlichen Seiten betont wird, dient das ER-Modell der Kommunikation der Projektbeteiligten. Immer wieder wird seine grafische Komponente hervorgehoben, welche eine Visualisierung der Ergebnisse und einen raschen Überblick über komplexe Zusammenhänge ermöglicht.

In der Praxis relativiert sich dieser Vorteil: Reale Datenmodelle, die oft 30 bis 60, manchmal über 100 Entities umfassen, sind nicht mehr handhabbar. Das ER-Diagramm wird hoffnungslos "verknüpft" und unübersichtlich. Das zwingt den Modellierer dazu, einen wesentlichen Teil seiner Zeit für die Anordnung des ER-Diagramms aufzuwenden, um wenigstens ein Mindestmaß an Übersichtlichkeit zu erhalten. Diese Arbeit aber dient keinerlei substantieller Weiterentwicklung des Modells.

Oft wird als Konsequenz das ER-Diagramm gar nicht mehr gepflegt, und damit auf die Visualisierung der Ergebnisse, einen der Hauptvorteile des ER-Modells, ganz verzichtet.


Das hier vorgeschlagene Anordnungsverfahren für das ER-Diagramm basiert direkt auf den in den Schlüssel-Beziehungen enthaltenen Informationen. Voraussetzung ist, den Schlüssel eines jeden Entities auch optisch, also im ER-Diagramm, erkennbar zu machen.

Das wird durch die folgende Erweiterung der für ER-Diagramme üblichen grafischen Notation erreicht (jeweils illustriert anhand des bereits bekannten Fallbeispiels):

35
1. Alle bisherigen Elemente der Grafik (die Symbole für Entities, Beziehungen, Kardinalitäten und Optionalitäten) werden beibehalten.


4. Das identifizierte Entity wird immer unterhalb (bzw. schräg unterhalb) seines/ seiner Identifizierenden Entities plaziert.

Der vierte Punkt schreibt lokal die Anordnung eines Entities im Verhältnis zu seinem/ seinen Identifizierenden vor (nämlich unterhalb). Dieses Element ist ein Teil des Verfahrens, das dann global die Anordnung des gesamten Diagramms steuert.
Ein etwas größeres Beispiel soll die Wirkungsweise des Anordnungs-Verfahrens verdeutlichen.

ER-Diagramm, von Hand angeordnet und mit konventioneller grafischer Notation:

ER-Diagramm des gleichen Modells, mit der oben vorgeschlagenen erweiterten Notation und durch das Verfahren angeordnet (der Einfachheit halber wurden die Namen der Beziehungen weggelassen):
Hierin liegt die Wirkungsweise dieses Anordnungs-Verfahrens:

- Das Diagramm ist aufgeteilt in mehrere Ebenen, wobei die Schichtung dieser Ebenen einen Anstieg von Einfachen zum Komplexen beinhaltet.
- Die Ebene eines Entities (seine y-Koordinate) lässt den Grad seiner Komplexität erkennen.
- Assoziative Entities (deren Schlüssel nur Beziehungen enthalten), befinden sich immer in der Nähe der Entities, die sie verknüpfen, nämlich dazwischen und schräg darunter. Z. B. befindet sich "Vorwurf" zwischen und schräg unter "Buch" und "Leser".
- "Teil-assoziative" Entities (deren Schlüssel sowohl Beziehungen als auch Attribute enthalten) befinden sich immer in der Nähe desjenigen Entities, das sie näher beschreiben, nämlich direkt darunter bei einer Schlüssel-Beziehung, oder dazwischen und schräg darunter bei mehreren Schlüssel-Beziehungen. Z. B. befindet sich "Buchexemplar" direkt unter "Buch".

- Das ER-Diagramm ist strukturiert, denn es besteht nicht mehr aus einem ungerichteten Graphen, sondern (verwandte der Schlüssel-Beziehungen) aus einer Quasihierarchie (gerichteter, kreisfreier Graph).

- Die Bedeutung der Entities spiegelt sich besser in der Struktur des Diagramms wieder, denn inhaltlich einander "nahestehende" Entities werden auch räumlich eng beieinander positioniert.

(Naturlich ist die Anordnung der Ebenen von oben nach unten hier rein willkürlich gewählt, ebensogut ist eine Orientierung von Links nach rechts denkbar, oder von unten nach oben.)

Durch all diese Eigenschaften ist das ER-Diagramm leicht lesbar, und Außenstehende oder neue Projektmitarbeiter können sich leichter darin orientieren.

Zusätzlich sollten die Vorteile nicht unterschätzt werden, welche die Möglichkeit der Automatisierung dieses Verfahrens in einem CASE-Tool mit sich bringt.

- Der Modellierer kann die optische Anordnung per Knopfdruck abrufen und braucht selbst keine Zeit auf die manuelle Umordnung eines ER-Diagramms aufzuwenden.
- Die Verwendung eines solchen Anordnungsverfahrens führt zu größerer Einheitlichkeit. Bei manueller Anordnung können selbst identische Modelle völlig unterschiedlich aussehen, wenn die räumliche Position der Entities und Beziehungen jeweils anders gewählt wurde.