G. Engels, U. Hohenstein, L. Neugebauer, G. Saake *, H.-D. Ehrich
TU Braunschweig, Informatik, Abt. Datenbanken, Braunschweig

Konzeption einer integrierten Datenbank-Entwurfsumgebung

Zusammenfassung

Es werden Konzepte für eine Entwurfsumgebung vorgestellt, in der Werkzeuge zur Unterstützung des konzeptionellen Datenbankentwurfs integriert sind. Grundlage für diese Umgebung sind umfangreiche theoretische Untersuchungen zur Festlegung eines semantisch wohldefinierten Datenmodells, mit dem insbesondere Nicht-Standard-Anwendungen geeignet modelliert werden können. Es werden eine Reihe von Edier-, Analyse- und Ausführungswerkzeugen skizziert, die es ermöglichen, ein konzeptionelles Datenbankschema, also die Beschreibung der statischen Struktur einer Datenbank und der erlaubten Aktionen, zu entwickeln, zu untersuchen und auf einem existierenden Datenbanksystem im Sinne eines "rapid prototyping" auszutesten. Alle diese Werkzeuge werden in einer Umgebung mit einheitlicher Benutzerschnittstelle integriert.

In Proc. DBTA/SI Conf. "Data Dictionaries und Entwicklungswerkzeuge für Datenbankanwendungen", pages 151-157. Verlag der Fachvereine an den Schweiz. Hochschulen und Techniken, Zürich, 1988.

* G. Saake wird von der Deutschen Forschungsgemeinschaft (Az.: Eh 75/6-1) gefördert.

1. Einleitung

Es ist weithin akzeptiert, daß die Erstellung umfangreicher Softwaresysteme durch geeignete Software-Werkzeuge unterstützt werden sollte. Hierzu gehören insbesondere Werkzeuge zur Unterstützung des Entwurfs und der Implementierung einer einem Softwaresystem zugrundeliegenden Datenbank. Wir stellen in dieser Arbeit Konzepte für eine Entwicklungsumgebung vor, in der Werkzeuge zur Unterstützung eines solchen konzeptionellen Datenbankentwurfs sowie zur Umsetzung eines entworfenen konzeptionellen Datenbankschemas, d.h. der Beschreibung der statischen Struktur einer Datenbank und der erlaubten Aktionen, in eine effiziente Implementierung integriert sind. Eine detailliertere Beschreibung dieser Konzepte befindet sich in /SNHE 87/.

Während eine Vielzahl von Software-Entwicklungsumgebungen zur Unterstützung der Erstellung größerer Softwaresysteme ohne Datenbankanwendung existieren (vgl. etwa /He 87/), sind erst wenige Umgebungen zur Unterstützung des Datenbankentwurfs bekannt (z.B. /ADD 85/, /BDRZ 84/, /BH 86/, /BJMSV 87/, /Re 86/, /RP 86/, /SO 87/). Charakteristisch für diese Entwurfsumgebungen ist, daß sie nicht das volle Spektrum der für einen Datenbankentwurf benötigten Werkzeuge bereitstellen. So fehlen in der Regel geeignete Werkzeuge zur Unterstützung der Objekt- und Transaktionsmodellierung für Nicht-Standard-Anwendungen. Weiterhin unterstützen die wenigsten Systeme ein frühzeitiges Austesten ("rapid prototyping") des aktuellen Datenbankschemas. Vor allem derartige Werkzeuge sind für einen interaktiven und iterativen Datenbankentwurf unter Einbeziehung des Endbenutzers besonders wünschenswert.

2. Charakteristika

Die von uns geplante **Datenbank-Entwurfsumgebung** (DBEU) zeichnet sich durch die folgenden Charakteristika aus:

- Die DBEU ist von Anfang an als **integrierte Umgebung** konzipiert. Das betrifft sowohl das externe Verhalten der Umgebung im Dialog mit dem Entwerfer als auch die interne Realisierung der DBEU.

- Hierzu gehört insbesondere eine **einheitliche Benutzerschnittstelle** für alle Werkzeuge. Das heißt, daß der Bildschirmaufbau, die Kommandosprache und das Systemverhalten einheitlich für alle Werkzeuge ist.

- Durch die Werkzeuge der DBEU werden alle **Phasen** des Entwurfs und der Implementierung einer Datenbank unterstützt, angefangen von einer informellen Anforderungsanalyse, der Entwicklung eines konzeptionellen Datenbankschemas, der Umsetzung dieses konzeptionellen Schemas in ein logisches Schema und die anschließende Implementierung auf einem existierenden Datenbanksystem.

- Neben dieser Unterstützung der inhaltlichen Entwicklung werden auch Werkzeuge zur Unterstützung des **Entwurfsvorgangs** (z.B. Zuständigkeiten im Entwerferteam), zur Verwaltung der erstellten Entwurfsdokumente (Datenhaltung inkl. **Varianten-/ Versionskontrolle**), sowie zur Erstellung der **technischen Dokumentation** zur Verfügung gestellt.

- Die DBEU ist als **offene** und **anpaßbare** Umgebung konzipiert. Das bedeutet, daß eine eventuelle Erweiterung des Kommando- oder Werkzeugsatzes oder gar die Unterstützung weiterer Aufgabenbereiche von Anfang an berücksichtigt wird. (Hierbei ist insbesondere an eine spätere Einbettung dieser DBEU in eine alle Aufgaben der Software-Entwicklung unterstützende Software-Entwicklungsumgebung gedacht.)

- Um unabhängig von der jeweils zur Verfügung stehenden Hardware zu sein, wird die **Portabilität** der DBEU insbesondere durch die Benutzung von verbreiteten, weitgehend normierten Schnittstellen (UNIX, GKS, X–Windows, C) gewährleistet.

3. Theoretischer Hintergrund

Alle in dieser Auflistung der Charakteristika erwähnten Aufgabenbereiche werden durch geeignete Softwarewerkzeuge unterstützt. In der ersten Phase unseres Projektes konzentrieren wir uns auf die Entwicklung von Werkzeugen zur **Unterstützung des konzeptionellen Entwurfs**. Hierzu existieren bereits umfangreiche theoretische Arbeiten zur Entwicklung eines semantisch wohldefinierten Datenmodells (/HNSE 87/). Dabei wurden insbesondere Anforderungen berücksichtigt, die bei der Modellierung von Nicht-Standard-Anwendungen auftreten. Ein zugehöriger konzeptioneller Entwurf besteht nun aus einer 4-Schichten-Spezifikation, die - nacheinander und aufeinander aufbauend - zu erstellen ist:

1. In der **Datenschicht** können beliebige Datentypen spezifiziert werden. So können neben den üblichen Standarddatentypen (z.B. INTEGER oder STRING) auch eigene, anwendungsspezifische Datentypen (z.B. POINTS, DATE) spezifiziert werden.

2. Die **Objektschicht** enthält die Spezifikation der Objekttypen, d.h. der Objektsorten mit ihren Beziehungen und Attributen. Die Grundlage dazu bildet ein erweitertes Entity-Relationship-Modell (kurz: EER-Modell) (ähnlich /EWH 85/), welches das ursprüngliche ER-Modell (/Ch 76/) um die Konzepte Typkonstruktion (Generalisierung/ Spezialisierung nach /SS 77/), komplexe Objekte (/RNLE 85/), abgeleitete Informationen, erweitertes Schlüsselkonzept und der Möglichkeit zur Spezifikation statischer Integritätsbedingungen ergänzt. Die zuvor spezifizierten Datentypen werden dabei als Wertebereiche der Attribute verwendet.

Beide Schichten beschreiben zusammen die Struktur der zulässigen Zustände der Datenbank. Die Beschreibung des Verhaltens der Datenbank erfolgt in den nächsten beiden Schichten:

3. In der **Entwicklungsschicht** werden die zulässigen Entwicklungen, d.h. die zeitlichen Abfolgen von Zuständen, durch dynamische Integritätsbedingungen spezifiziert.

4. Zur Spezifikation von Aktionen (zustandsabhängigen Operationen) wird in der **Aktionsschicht** eine Datenmanipulationssprache auf der Grundlage des EER-Modells angeboten, bestehend aus einer EER-Anfragesprache und Änderungsoperationen.

Jede dieser vier Schichten erfordert ihre eigene Spezifikationssprache, hinter der jeweils eine fundierte theoretische Grundlage in Form von aufeinanderaufbauenden Logik-Kalkülen steht. So erfolgt die Spezifikation
1. der (abstrakten) Datentypen algebraisch (z.B. durch Gleichungsspezifikationen),
2. der Objektschicht mittels eines auf der Prädikatenlogik 1. Stufe aufbauenden EER-Objektmodells (/SSE 87/),
3. der zulässigen Entwicklungen mittels einer Temporalen Logik (/LS 87/) und
4. der Wirkung von Aktionen durch eine Dynamische Logik sowie einen EER-Objektkalkül zur Formulierung von Anfragen, der als semantische Grundlage einer deskriptiven EER-Anfragesprache verwendet wird.

4. Entwurfswerkzeuge

Der Entwurf eines konzeptionellen Datenbankschemas in Form einer derartigen 4-Schichten-Spezifikation soll in der geplanten Datenbank-Entwurfsumgebung durch die folgenden vier Werkzeuggruppen unterstützt werden:

i) Zur Eingabe, Änderung und Dokumentation des konzeptionellen Datenbankschemas steht eine Familie **syntaxgestützter Editoren** zur Verfügung:

- ein syntaxgestützter Editor zur textuellen Eingabe und Änderung von algebraischen Spezifikationen zur Beschreibung von benutzerdefinierten Datentypen,

- ein strukturorientierter, graphischer Editor zur Eingabe, Änderung und Inspektion der Objektschicht in der Form erweiterter Entity-Relationship-Diagramme,

- ein syntaxgestützter Editor zur textuellen Eingabe und Änderung von dynamischen Integritätsbedingungen, d.h. von temporalen Formeln,

- ein syntaxgestützter Editor zur textuellen Eingabe und Änderung der Beschreibung von erlaubten Aktionen, insbesondere Änderungsoperationen, auf den im Schema definierten Objekten und Beziehungen.

ii) Die oben genannten Editoren ermöglichen verschiedene Konsistenzüberprüfungen und garantieren danach weitgehende kontextfreie und kontextsensitive Korrektheit des Schemas. Weitergehende Zusammenhänge innerhalb des Schemas können durch das Werkzeug der **statischen Analyse** vom Entwickler ermittelt werden. Hierzu gehört z.B. eine Analyse der Datentypen-Spezifikationen durch ein Termersetzungssystem. Ein Schwerpunkt der statischen Analyse der konzeptionellen Schemata ist die Untersuchung der Erfüllbarkeit der spezifizierten Integritätsbedingungen. Modellinhärente und durch den Entwerfer definierte explizite statische Integritätsbedingungen werden mit Methoden der logischen Programmierung auf Erfüllbarkeit untersucht.

Dynamische Integritätsbedingungen, d.h. Formeln einer temporal erweiterten Prädikatenlogik, werden mit den in /LS 87/ vorgestellten Methoden untersucht. Kern dieser Analyse ist die Konstruktion von Transitionsgraphen aus temporalen Formeln, die zusätzlich zur Konsistenzprüfung eine graphische Darstellung der spezifizierten Objektlebensläufe ermöglichen.

iii) Für einen Datenbankentwerfer ist es hilfreich und informativ, bereits in einer sehr frühen Phase spezifizierte Aktionen der entworfenen Datenbank im Sinne eines 'rapid prototyping' ausführen zu lassen. Hierzu wird ein entsprechendes **Ausführungswerkzeug** zur Verfügung gestellt, das das entworfene konzeptionelle Schema zunächst in ein logisches Schema eines vorhandenen Datenbanksystems transformiert. Eine dementsprechend definierte Testdatenbank kann der Entwerfer mit Hilfe eines Testdatengenerators (/NN 85/) mit Daten füllen lassen und anschließend Anfragen und Aktionen auf der Datenbank austesten. Dieser Transformationsschritt wurde bisher für die Umsetzung in ein relationales Datenbankschema untersucht (/HNS 86/). Die Transformation geschieht größtenteils automatisch. An einigen Stellen ist der Entwerfer dafür verantwortlich, deskriptive Beschreibungen in prozedurale zu transformieren, so z.B. auf der Ebene der Datentypen und Änderungsaktionen. Derzeit wird untersucht, inwieweit andere Datenbanksysteme mit komplexeren Datenmodellen (z.B. NF^2-Datenmodell (/SS 86/)) für ein derartiges Werkzeug geeigneter sind.

iv) Während der Ausführung von Aktionen auf einer Testdatenbank steht ein **dynamisches Analysewerkzeug** zur Verfügung. Hierzu gehört ein aufgrund der dynamischen Integritätsbedingungen erstellter Integritätsmonitor (/Hü 88/, /LS 87/). Dieser konstruiert vor Beginn der Ausführung für jede der zu überwachenden dynamischen Integritätsbedingungen ein optimiertes Überprüfungsschema. Das transformierte Schema der Datenbank wird erweitert, um die für die Überwachung benötigte minimale historische Information in der Datenbank zu halten. Zur Ausführungszeit wird die Einhaltung der statischen und dynamischen Integrität nach jeder Aktion anhand dieser Überprüfungsschemata kontrolliert und die historische Information über die betroffenen Objekte aktualisiert. Der Integritätsmonitor erkennt alle Aktionsausführungen, die die Integritätsbedingungen verletzen.

Im Rahmen von Diplom- und Promotionsarbeiten werden diese Werkzeuge zur Entwicklung eines konzeptionellen Schemas innerhalb einer Prototypimplementierung realisiert, wobei Erfahrungen aus der Implementierung einer allgemeinen Software-Entwicklungsumgebung einfließen (/ENS 87/). Die Implementierung erfolgt in der Programmiersprache C auf 68020-Workstations unter dem Betriebssystem UNIX.

Literatur

/ADD 85/ Albano, A./ DeAntonellis, V./ DiLeva, A. (eds.): Computer-Aided Database Design: The DATAID Project. Amsterdam: North-Holland 1985

/BDRZ 84/ Brägger, R./ Dudler, A./ Rebsamen, J./ Zehnder, C.A.: Gambit - An Interactive Database Design Tool for Structures, Integrity Constraints, and Transactions. Proc. IEEE Int. Conf. on Software Engineering, Los Angeles, April 1984, 399-407

/BH 86/ Bryce, D./ Hull, R.: SNAP: A Graphics-based Schema Manager, Proc. Int. Conf. on Data Engineering, Los Angeles, 1986, 151-163

/BJMSV 87/ Borgida, A./ Jarke, M./ Mylopoulos, J./ Schmidt, J.W./ Vassiliou, Y.: The Software Development Environment as a Knowledge Base Management System, Technischer Bericht MIP-8710, Universität Passau, 1987

/Ch 76/ Chen, P.P.: The Entitiy-Relationship-Model - Towards a Unified View of Data, ACM ToDS Vol. 1, No.1, 1976, 9-36

/EWH 85/ Elmasri, R./ Weeldreyer, J./ Hevner, A.: The Category Concept: An Extension to the Entity-Relationship Model. Data & Knowledge Engineering Vol. 1, 1985, 75-116

/ENS 87/ Engels, G./ Nagl, M./ Schäfer, W.: On the Structure of Structure-Oriented Editors for Different Applications, in /He 87/, 190-198

/He 87/ Henderson, P. (ed.): Proc. of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, ACM SIGPLAN Notices, Vol. 22, No. 1, January 1987

/HNS 86/ Hohenstein, U./ Neugebauer, L./ Saake, G.: An Extended Entity-Relationship Model for Non-Standard Databases. Proc. "Workshop über Relationale Datenbanken" (ed. A. Heuer), Lessach (Österreich), Juni 1986, TU Clausthal-Zellerfeld, 185-211

/HNSE 87/ Hohenstein, U./ Neugebauer, L./ Saake, G./ Ehrich, H.-D.: Three Level Specification of Databases Using an Extended Entity Relationship Model. Proc. GI-Fachtagung "Informationsbedarfsermittlung und -analyse für den Entwurf von Informationssystemen" (R.R. Wagner; R. Traunmüller; H.C. Mayr (eds.)), Informatik-Fachberichte Bd. 143, Berlin: Springer 1987, 58-88

/Hü 88/ Hülsmann, K.: Entwurf eines Systems zur Überwachung dynamischer Integritätsbedingungen, Diplomarbeit, TU Braunschweig, Abt. Datenbanken 1988

/LS 87/ Lipeck. U.W./ Saake, G.: Monitoring Dynamic Integrity Constraints Based on Temporal Logic. Information Systems, Vol. 12, No. 3, 1987

/NN 85/ Neugebauer, L./ Neumann, K.: Schemagesteuerte Testdatenerzeugung für relationale Datenbanken, TU Braunschweig, Informatik-Berichte, Bericht Nr. 85-02, 1985

/Re 86/ Reiner, D. et al.: A Database Designer's Workbench, Proc. 5th Int. Conf. on ER-Approach, Dijon, France, 1986, 127-140

/RNLE 85/ Ramm, I./ Neumann, K./ Lipeck, U.W./ Ehrich, H.-D.: Eine Benutzerschnittstelle für geowissenschaftliche Datenbanken, Informatik-Bericht Nr. 85-08, TU Braunschweig, 1985

/RP 86/ Rolland, C./ Proix, C.: An Expert System Approach to Information System Design, Information Processing 1986, 241-249

/SO 87/ Schönthaler, F./ Oberweis, A.: Prototyping zur Unterstützung des konzeptionellen Entwurfs interaktiver Informationssysteme, Proc. Fachtagung Informationsbedarfsermittlung und Analyse, Linz 1987

/SNHE 87/ Saake, G./ Neugebauer, L./ Hohenstein, U./ Ehrich, H.-D.: Konzepte und Werkzeuge für eine Datenbank-Entwurfsumgebung, TU Braunschweig, Informatik-Berichte, Bericht Nr. 87-05, 1987

/SS 77/ Smith, J.M./ Smith, D.C.P.: Database Abstractions: Aggregation and Generalization. ACM ToDS Vol. 2, No. 2, 1977, 105-173

/SS 86/ Scheck, H.-J./ Scholl, M.H.: The Relational Model with Relation-valued Attributes, in Information Systems, Vol. 11, No. 2, 1986, 137-147

/SSE 87/ Sernadas, A./ Sernadas, C./ Ehrich, H.-D.: Object-Oriented Specification of Databases: An Algebraic Approach. Proc. 13th Int. Conf. on Very Large Databases. Brighton 1987