CSLIB2000-Projekt SS95

Größere Teilaufgabe:
Erfassung von Kongreßbänden

Gruppe:

Robert Wagner

Dozentin:

Ulrike Reiner

Technische Universität Berlin
Institut für Angewandte Informatik
Einheit Wissensbasierte Systeme
Franklinstraße 28/29
10587 Berlin
Sekr. 5-11

Inhaltsverzeichnis

1 Aufgabenstellung **3**

2 Erfassung der Mediendaten **3**

3 Elektronische Speicherung der erfassten Daten **3**

4 Auslesen der erfassten Mediendaten **5**

4.1 Methode des Auslesens 6
4.2 Modulstruktur 6
4.3 Beschreibung der globalen Datenstrukturen 7
4.3.1 Modul *datei* 7
4.3.2 Modul *entities* 9
4.3.3 Modul *error* 12
4.3.4 Modul *kongress* 12
4.3.5 Modul *relations* 13
4.4 Beschreibung der wichtigsten Funktionen 15
4.4.1 Modul *datei* 15
4.4.2 Modul *kongress* 15
4.5 Erweiterungsmöglichkeiten 16

1 Aufgabenstellung

Nachdem im Rahmen des Projektes CSLIB2000 im SS 1995 ein Informationsmodell für Dokumente aus dem Bestand der Fachbereichsbibliothek des FB 13 (Informatik) der TU Berlin vorgestellt wurde (siehe [1]), galt es nun, exemplarisch aufzuzeigen, wie eine Erfassung konkreter Mediendaten – in diesem Fall derer von Kongreßbänden – anhand dieses Modells stattzufinden hat.

Da die aus dem Informationsmodell erzeugte Datenbank zum Zeitpunkt der Erfassung noch nicht existierte, galt es ferner, eine sinnvolle Repräsentationsform für das elektronische Festhalten der erfassten Mediendaten zu finden, die ein späteres Konvertieren der Daten in INFORMIX-konforme Dateien ermöglicht bzw. erleichtert.

Schließlich hieß es, eine Umgebung zu installieren, die ein Auslesen der erfassten Mediendaten in der jetzigen Form ermöglicht, um aufzuzeigen, wie das Informationsmodell auf die realen Daten angewandt wurde.

2 Erfassung der Mediendaten

Zur Erfassung der entsprechenden Daten der Kongreßbände begab sich der Erfasser mehrmals in die oben genannte Fachbereichsbibliothek. Aus dem dort vorhandenen umfangreichen Sortiment der Kongreßbände (über 3000) wählte er willkürlich 118 aus den unterschiedlichsten Themengebieten der Informatik stammende zur näheren Betrachtung aus. Diese Auswahl konnte notwendigerweise nicht repräsentativ sein, da zu einer repräsentativen Auswahl bereits umfangreiche Kenntnisse über sämtliche Daten der vorhandenen Kongreßbände hätten vorhanden sein müssen, um zu beurteilen, was denn nun repräsentativ ist und was nicht. Trotzdem kann sicherlich davon ausgegangen werden, daß die erfassten Daten einen recht guten Querschnitt durch die vorhandenen Kongreßbände darstellen.

3 Elektronische Speicherung der erfassten Daten

Da – wie bereits oben erwähnt – die Datenbank zum Zeitpunkt der Erfassung noch nicht existierte, mußte eine andere Repräsentationsform für die effiziente Speicherung der erfassten Mediendaten gefunden werden. Es wurde entschieden, für jeden Entity-Typen und jeden Relationship-Typen aus dem Informationsmodell eine Datei zu erzeugen, in der die entsprechen-

Entity-/Relationship-Typ	Dateiname
institution	E_institution
isbn	E_isbn
issn	E_issn
konferenzbericht	E_konferenzbericht
konferenzserie	E_konferenzserie
konferenztagung	E_konferenztagung
person	E_person
rolle	E_rolle
sprache	E_sprache
titel	E_titel
titelart	E_titelart
verbindungsart	E_verbindungsart
erstellt_für	R_erstellt_für
führt_durch	R_führt_durch
hat_isbn	R_hat_isbn
hat_issn	R_hat_issn
hat_sprache	R_hat_sprache
hat_titel	R_hat_titel
produziert	R_produziert
verantwortlich_für	R_verantwortlich_für
verbunden_mit	R_verbunden_mit
zur_serie	R_zur_serie

Tabelle 1: Entity- und Relationship-Typen mit ihren Dateien

den Mediendaten gespeichert werden sollten. Diese Dateien erhielten die Namen des jeweils entsprechenden Äquivalents im Informationsmodell zusammen mit einem Hinweis, ob es sich um einen Entity-Typen oder einen Relationship-Typen handelt. Tabelle 1 enthält die im Informationsmodell enthaltenen Entity- und Relationship-Typen zusammen mit ihren entsprechenden Dateien.

In den Entity-Dateien befindet sich nun pro Zeile genau ein Entity, dessen Attribute jeweils durch ein Tabulatorzeichen getrennt sind. In den Relationship-Dateien befindet sich entsprechend pro Zeile eine Relationship, wobei die Entities, zwischen denen diese Relationship besteht, durch ihre Zeilennummern innerhalb ihrer Entity-Dateien gekennzeichnet sind und jeweils durch ein Tabulatorzeichen getrennt sind. Hier einige Beispiele, die den Aufbau der Dateien verdeutlichen sollen:

Der Entity-Typ *person* besteht aus den Attributen *nachname*, *vorname* und *titel*. Ein Eintrag in `E_person` könnte nun folgendermaßen aussehen:

```
Derigs  Ulrich  Prof. Dr. Dr.
```

Hierbei ist „Derigs" der Nachname der Person, „Ulrich" der Vorname und „Prof. Dr. Dr. " der Titel dieser Person. Hat eine Person keinen Titel, so befindet sich an dieser Stelle ein „-".

Der Relationship-Typ *produziert* verbindet die Entity-Typen *dokumente* – in diesem speziellen Fall *konferenzbericht* –, *rolle* und *produzent*. Daher könnte ein Eintrag in `R_produziert` wie folgt aussehen:

```
2       1       p4
```

Der Konferenzbericht befindet sich hier also in der 2. Zeile von `E_konferenzbericht`, die Rolle des Produzenten befindet sich in der 1. Zeile von `E_rolle`. Der Entity-Typ *produzent* ist eine Generalisierung der beiden Entity-Typen *institution* und *person*. Folglich muß bei dem Verweis auf den Produzenten zusätzlich angegeben werden, ob es sich um eine Institution oder eine Person handelt. Dies geschieht durch das vorangestellte „p", welches hier für Person steht. Handelte es sich bei dem Produzenten um eine Institution, so stünde an dieser Stelle ein „i". Da *produzent* der einzige generalisierte Entity-Typ in dem Datenmodell ist, gibt es keine weiteren Relationship-Dateien, in denen solch eine Besonderheit vorkommt. Nach dem „p" steht die Zahl „4", was bedeutet, daß die den Konferenzbericht produzierende Person in Zeile 4 von `E_person` angesiedelt ist.

Die gewählte Strukturierung hat den entscheidenen Vorteil, daß keinerlei redundante Daten gespeichert werden. Die Daten einer Institution (bspw. eines Verlages) werden nur genau einmal abgelegt, selbst wenn sich dieser Verlag für mehrere Konferenzbände verantwortlich zeichnet.

4 Auslesen der erfassten Mediendaten

Um die Ergebnisse der Informationserfassung nachvollziehen zu können, wurde eine Umgebung benötigt, die die gespeicherten Daten ausliest und in geeigneter Form auf dem Bildschirm darstellt. Es sei ausdrücklich darauf hingewiesen, daß sich dieses Programm nicht als Datenbankmanagementsystem versteht. Es stellt keine Anfragesprache zur Verfügung, sondern es können lediglich alle oder einer der 118 erfassten Kongreßbände auf dem

Bildschirm angezeigt werden. Ferner werden bei der Ausgabe der Daten zusätzlich die Entity- und Relationship-Typ-Namen mitausgegeben, um zu verdeutlichen, wie die erfassten Daten mit Hilfe des Informationsmodells repräsentiert wurden. Ferner dürfte es keinen großen Aufwand bereiten, das entstandene Programm um eine zusätzliche Funktionalität zu erweitern, mit Hilfe derer die erfaßten Daten in eine von einer INFORMIX-Datenbank verstandene Form konvertiert werden.

Das Programm wurde in der Programmiersprache *C* implementiert.

4.1 Methode des Auslesens

Die Extrahierung der Informationen eines Kongreßbandes läuft nun wie folgt ab:

Die angegebene Nummer des Kongreßbandes (also 1-118) ist der Index in die Entity-Datei `E_konferenzbericht`. Gleichzeitig können mit ihrer Hilfe die Relationships *hat_isbn*, *hat_issn*, *hat_sprache*, *hat_titel*, *produziert* sowie *erstellt_für* gefüllt werden, da diese mit dem Entity *konferenzbericht* verbunden sind. Zu diesem Zweck muß in der linkesten Spalte der entsprechenden Relationshipdatei nach dem Index des Kongreßbandes (genauer: nach dem i-ten Index für den i-ten Kongreßband) gesucht werden und in einer der darauffolgenden Spalten die Zeilennumer des Entities in der entsprechenden Entity-Datei ermittelt werden, um daraufhin dieses Entity mit seinen Attributen auszulesen.

Wird dabei eine *konferenztagung* extrahiert, so werden mit Hilfe ihrer Zeilennummer die Relationships *zur_serie* und *führt_durch* nach demselben Schema gefüllt, da diese Relationships mit dem Entity *konferenztagung* verbunden sind.

Wird dabei wiederum auf eine *konferenzserie* getroffen , so wird mit ihrer Zeilennummer die Relationship *verantwortlich_für* gefüllt.

Jeweils wenn während dieser gesamten Vorgänge auf eine *institution* getroffen wird (z. B. als Verlag oder Kongreßverantwortlicher), werden die Verbindungen dieser Institution mit anderen Institutionen überprüft und gegebenenfalls in der Relationship *verbunden_mit* festgehalten.

Schließlich werden die ermittelten Relationships mit ihren Entities und deren Attributen nacheinander auf dem Bildschirm angezeigt.

4.2 Modulstruktur

Die Implementierung besteht aus sieben Modulen:

- *datei*: Dieses Modul stellt Datenstrukturen und Funktionen zur Verfügung, mit Hilfe derer die erfassten Mediendaten in den Hauptspeicher gelesen werden können.
- *entities*: Hier befinden sich die Datenstrukturen für die Entities.
- *error*: Sollten die Dateien, in denen sich die erfassten Daten befinden, aus irgendeinem Grund gelöscht oder unbrauchbar werden, so können die Daten nicht mehr korrekt extrahiert werden. *error* stellt eine Funktion zur Verfügung, die eine entsprechende Meldung ausgibt und das Programm abbricht.
- *kongress*: Hierin befinden sich die Funktionen, um aus den in den Speicher eingelesenen Daten diejenigen zu extrahieren, die für den angegebenen Kongressband relevant sind.
- *main*: Mit Hilfe der Funktion dieses Moduls, wird das Programm gestartet. Zusätzlich finden hier Variableninitialisierungen und ein Teil des Speichermanagements statt.
- *print*: *print* gibt die Daten eines konkreten Kongreßbandes auf dem Bildschirm aus.
- *relations*: Hier befinden sich die Datenstrukturen für die Relationships.

4.3 Beschreibung der globalen Datenstrukturen

4.3.1 Modul *datei*

In `datei.h` befinden sich die beiden Datentypen `t_entities` und `t_relations`:

```
typedef struct entities
{
    char *konferenzbericht;
    int len_konferenzbericht;
    char *institution;
    int len_institution;
    char *isbn;
    int len_isbn;
    char *issn;
    int len_issn;
```

```
    char *konferenzband;
    int len_konferenzband;
    char *konferenzserie;
    int len_konferenzserie;
    char *konferenztagung;
    int len_konferenztagung;
    char *person;
    int len_person;
    char *rolle;
    int len_rolle;
    char *sprache;
    int len_sprache;
    char *titel;
    int len_titel;
    char *titelart;
    int len_titelart;
    char *verbindungsart;
    int len_verbindungsart;
} t_entities;

typedef struct relations
{
    char *erstellt_fuer;
    int len_erstellt_fuer;
    char *fuehrt_durch;
    int len_fuehrt_durch;
    char *hat_isbn;
    int len_hat_isbn;
    char *hat_issn;
    int len_hat_issn;
    char *hat_sprache;
    int len_hat_sprache;
    char *hat_titel;
    int len_hat_titel;
    char *produziert;
    int len_produziert;
    char *verantwortlich_fuer;
    int len_verantwortlich_fuer;
    char *verbunden_mit;
```

```
    int len_verbunden_mit;
    char *zur_serie;
    int len_zur_serie;
} t_relations;
```

In Variablen dieser Typen werden innerhalb des Programms die Inhalte der entsprechenden Entity- und Relationship-Dateien zusammen mit ihren Längen gespeichert.

4.3.2 Modul *entities*

In `entities.h` wird für jeden Entity-Typen ein Variablentyp definiert:

```
typedef enum entity_typ
{
    KONFERENZBERICHT, INSTITUTION, ISBN, ISSN, KONFERENZBAND,
    KONFERENZSERIE, KONFERENZTAGUNG, PERSON, PRODUZENT, ROLLE, SPRACHE,
    TITEL, TITELART, VERBINDUNGSART
} t_entity_typ;

typedef struct konferenzbericht
{
    char *druck;
    char *erscheinungsdatum;
    char *format;
    char *bandnummer;
} t_konferenzbericht;

typedef struct isbn
{
    char *isbn;
} t_isbn;

typedef struct issn
{
    char *issn;
} t_issn;

typedef struct titel
{
```

```
    char *titel;
} t_titel;

typedef struct sprache
{
    char *sprache;
} t_sprache;

typedef struct titelart
{
    char *titelart;
} t_titelart;

typedef struct rolle
{
    char *rolle;
} t_rolle;

typedef struct person
{
    char *name;
    char *vorname;
    char *titel;
} t_person;

typedef struct institution
{
    char *name;
    char *untertitel;
    char *adresse;
    char *reichweite;
} t_institution;

typedef union produzent
{
    t_person person;
    t_institution institution;
} t_produzent;
```

```
typedef struct verbindungsart
{
    char *verbindungsart;
} t_verbindungsart;

typedef struct konferenzserie
{
    char *konferenztitel;
    char *konferenzintervall;
} t_konferenzserie;

typedef struct konferenztagung
{
    char *motto;
    char *konferenzort;
    char *konferenzdatum;
} t_konferenztagung;

typedef union entity
{
    t_institution institution;
    t_isbn isbn;
    t_issn issn;
    t_konferenzbericht konferenzbericht;
    t_konferenzserie konferenzserie;
    t_konferenztagung konferenztagung;
    t_person person;
    t_rolle rolle;
    t_sprache sprache;
    t_titel titel;
    t_titelart titelart;
    t_verbindungsart verbindungsart;
} t_entity;
```

`t_produzent` entspricht dabei dem im Informationsmodell generalisierten Entity-Typen *produzent*. Ob nun eine Person oder eine Institution in einer Variable dieses Typs gespeichert ist, wird mit Hilfe eines Index im Datentyp `t_produziert` vermerkt (siehe unten).

4.3.3 Modul *error*

Der einzige Fehler, der theoretisch auftreten könnte, ist eine fehlerhafte Datei, in der erfasste Daten abgelegt sind. Dies kann entweder eine Entity- oder Relationship-Datei sein:

```
typedef enum error
{
    ENTITY, RELATION
} t_error;
```

4.3.4 Modul *kongress*

Ein konkreter Kongreßband besteht aus den im Informationsmodell dargestellten Relationships, wobei nicht alle vorhanden sein müssen, und den durch diese Relationships verbundenen Entities. Wieviele Relationships eines bestimmten Relationship-Typs existieren wird ebenfalls festgehalten:

```
typedef struct kongressband
{
    int anz_fuehrt_durch;
    t_fuehrt_durch *fuehrt_durch;
    int anz_erstellt_fuer;
    t_erstellt_fuer *erstellt_fuer;
    int anz_hat_isbn;
    t_hat_isbn *hat_isbn;
    int anz_hat_issn;
    t_hat_issn *hat_issn;
    int anz_hat_sprache;
    t_hat_sprache *hat_sprache;
    int anz_hat_titel;
    t_hat_titel *hat_titel;
    int anz_produziert;
    t_produziert *produziert;
    int anz_verantwortlich_fuer;
    t_verantwortlich_fuer *verantwortlich_fuer;
    int anz_verbunden_mit;
    t_verbunden_mit *verbunden_mit;
    int anz_zur_serie;
    t_zur_serie *zur_serie;
} t_kongressband;
```

4.3.5 Modul *relations*

Hier wird analog zum Modul *entities* für jeden Relationship-Typ ein entsprechender Variablentyp definiert:

```
typedef struct hat_isbn
{
    t_konferenzbericht konferenzbericht;
    t_isbn isbn;
} t_hat_isbn;

typedef struct hat_issn
{
    t_konferenzbericht konferenzbericht;
    t_issn issn;
} t_hat_issn;

typedef struct hat_titel
{
    t_konferenzbericht konferenzbericht;
    t_sprache sprache;
    t_titelart titelart;
    t_titel titel;
} t_hat_titel;

typedef enum index
{
    PERS, INST
} t_index;

typedef struct produziert
{
    t_konferenzbericht konferenzbericht;
    t_rolle rolle;
    t_index index;
    t_produzent produzent;
} t_produziert;

typedef struct verbunden_mit
{
```

```
    t_institution institution;
    t_institution institution2;
    t_verbindungsart verbindungsart;
} t_verbunden_mit;

typedef struct hat_sprache
{
    t_konferenzbericht konferenzbericht;
    t_sprache sprache;
} t_hat_sprache;

typedef struct verantwortlich_fuer
{
    t_institution institution;
    t_konferenzserie konferenzserie;
} t_verantwortlich_fuer;

typedef struct fuehrt_durch
{
    t_institution institution;
    t_konferenztagung konferenztagung;
} t_fuehrt_durch;

typedef struct zur_serie
{
    t_konferenztagung konferenztagung;
    t_konferenzserie konferenzserie;
} t_zur_serie;

typedef struct erstellt_fuer
{
    t_konferenzbericht konferenzbericht;
    t_konferenztagung konferenztagung;
} t_erstellt_fuer;
```

Der `index` im Datentyp `t_produzent` gibt die Art des Produzenten an: entweder eine Person oder eine Institution.

4.4 Beschreibung der wichtigsten Funktionen

4.4.1 Modul *datei*

- `read_entities()`:
  - Einlesen der einzelnen Entity-Dateien in entsprechende char-Arrays

4.4.2 Modul *kongress*

- `get_kongressband()`:
  - füllt alle Relationships, die zwischen Entities des Typs *konferenzbericht* und entsprechenden anderen Entities bestehen, also die Relationships *hat_isbn*, *hat_issn*, *hat_sprache*, *hat_titel*, *produziert* sowie *erstellt_für*.
- `get_konferenztagung()`:
  - füllt alle Relationships, die zwischen Entities des Typs *konferenztagung* und entsprechenden anderen Entities bestehen, also die Relationships *zur_serie* und *führt_durch*.
- `get_konferenzserie()`:
  - füllt die Relationships des Typs *verantwortlich_für*, die zwischen Entities der Typen *konferenzserie* und *institution* bestehen.
- `get_institution()`:
  - füllt die Relationships des Typs *verbunden_mit*, die zwischen jeweils zwei Entities der Typen *institution* bestehen.
- `get_hat_isbn()`:
  - Ermitteln der Anzahl der Relationships zwischen *konferenzbericht* und *isbn* mit `get_anzahl()` (*kongress*)
  - Holen der Entities *konferenzbericht* und *ISBN* mit `get_entity()` (*kongress*)
- die übrigen Prozeduren aus der `get`-Familie arbeiten analog
- `get_entity()`:

  - in den Relationships nach der übergebenen Zeilennummer suchen
  - nach der i-ten dieser Zeilennummern suchen
  - in die angegebene Spalte springen und dortige Zeilennummer auslesen
  - Zeilennummer in entsprechendes Entity umwandeln mit `zeile2entity()` (*kongress*)

- `zeile2entity()`
  - in der entsprechenden in den Hauptspeicher eingelesenen Entity-Datei in die angegebene Zeile springen
  - Rückgabe der dortigen Werte mit `set_entity()` (*kongress*)

4.5 Erweiterungsmöglichkeiten

Die simple Ausgabe der erfassten Kongressbanddaten auf dem Bildschirm ist zwar als exemplarische Darstellung recht nett anzusehen, in bezug auf die Integration dieser Daten in eine bestehende Datenbank jedoch offensichtlich unzureichend. Wie bereits oben erwähnt, war die Datenbank zum Zeitpunkt dieser Datenerfassung und -speicherung noch nicht vorhanden, jedoch sollte es problemlos möglich sein, das Programm um eine Konvertierungsfunktion in eine von einer INFORMIX-Datenbank verstandene Form zu erweitern. Der Autor stellt sich dafür gerne – selbstverständlich unentgeldlich – zur Verfügung.

Literatur

[1] Arbeitsgruppen 2, 10, 12, 16: Datenspezifikation und Datenmodellierung (1995)