Kurzeinführung in das DBMS O_2

In einer kurzen Übersicht wird im folgenden das objekt-orientierte Datenbank-Management-System O_2 vorgestellt.

1 Technische Voraussetzungen

O_2 läuft auf verschiedenen Rechnern des IfIS. Um damit arbeiten zu können, muß man sich auf einem dieser Rechner (remote oder im Rechnerraum des IfIS) anmelden. Bei einer Remoteverbindung an das Setzen der DISPLAY-Variable denken!

Mit

`.  O2setup` (für Bourne-Shell-Familie) bzw.

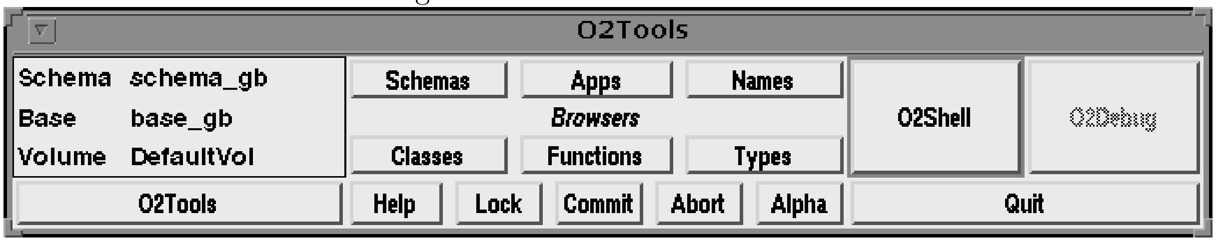
`source /home/ifis/setup/O2setup` (für C-Shell-Familie)

werden die benötigten Umgebungsvariablen gesetzt.

Der Aufruf von O_2 geschieht durch:
`o2 -server <servername> -system <systemname> -toolsgraphic`

Durch die vorhandenen Lizenzen müssen bestimmte Festlegungen gemacht werden, die die einzelnen Server- bzw. Systemnamen betreffen. Diese werden individuell bekanntgegeben.

Nach dem Aufruf erscheint folgendes Fenster:



2 Beschreibung der O_2-Benutzeroberfläche

Die Arbeit mit O_2 geschieht über die graphische Benutzeroberfläche O_2*Tools*, in derem Zentrum sechs Browser für das Datenbankschema stehen. Die einzelnen Komponenten sind nachfolgend stichpunktartig zusammengefaßt. Für eine ausführliche Dokumentation sei auf das Handbuch zu O_2Tools verwiesen.

Quit Hiermit wird O_2 verlassen.

Help Diese Funktion gibt dem Benutzer von O_2Tools Erläuterungen zu den verschiedenen Begriffen, welche schema- -oder systembezogen vorkommen.

Lock Mittels dieser Funktion kann man O_2Tools für andere Benutzer sperren; mit dem eigenen Unix-Password entsperrt man wieder. Auf diese Funktion kann jedoch allgemein verzichtet werden.

Commit Hiermit wird die aktuelle Transaktion freigegeben und mit

Abort wird sie abgebrochen.

Alpha Diese Funktion schaltet auf die alphanumerische Oberfläche um (das kann nicht mehr rückgängig gemacht werden!) - wird hier nicht gebraucht.

O2Tools erlaubt dem Benutzer, bestimmte Optionen für die Arbeit mit O2Tools zu setzen.

O2Shell Alle Kommandos zur Deklaration oder Manipulation des Schemas können auch von der O2Shell aus eingegeben werden. Von hier aus können auch O_2SQL-Anweisungen ausgeführt werden.

Zentral in O_2Tools sind die sechs Browser für den Zugriff und die Manipulation auf Schemainformation, aber auch für die Entwicklung einer Applikation. Jeder Browser kann nach Aufruf über `Window:close` wieder verlassen werden.

Schemas Dieser Browser listet die vorhandenen Schemata auf, zusammen mit den zugehörigen Datenbanken. Der Benutzer kann eines der vorhandenen Schemata zusammen mit einer Datenbank selektieren oder aber ein neues Schema (`Schema: create`) und eine neue Datenbank (`base:create`) erzeugen.

Apps Auf jedes O_2-Schema setzt eine Applikation auf, welche zusammengehörige Funktionen und Operationen auf dem Schema zusammenfaßt und dem Benutzer über ein Menü zur Verfügung stellt. Dieser Browser unterstützt den Benutzer bei der Entwicklung einer Applikation.

Names Zu jedem Schema kann eine Menge von *Namen* spezifiziert werden, welche persistent sind und als Einstieg in die Datenbank zur Laufzeit dienen. Daten oder Objekte, welche nicht über einen Namen direkt oder indirekt zugreifbar sind, besitzen keine Persistenz.

Classes Dieser wohl am häufigsten verwendete Browser unterstützt den Datenbankentwickler beim Anlegen und Implementieren von Klassen.

Types In O_2 wird zwischen Klassen und Typen unterschieden. Dieser Browser dient der Definition von Typen. Typen beschreiben Werte und diese besitzen im Gegensatz zu Klassen kein eigenes Verhalten.

Functions Funktionen können sowohl auf Objekten als auch auf Werten definiert sein. Funktionen unterscheiden sich von Methoden dadurch, daß sie nicht dem Verhalten eines Objektes zugeordnet werden, bzw. sie ermöglichen Operationen auch auf Werten. Dieser Browser unterstützt den DB-Entwickler bei der Definition und Implementation von Funktionen.

2.1 "Mini"-Tutorial

Nachfolgend wollen wir anhand einer kleinen Beispielsitzung den prinzipiellen Ablauf einer Applikations-Entwicklung mit O_2 skizzieren. Dazu betrachten wir das aus der ersten Übung bekannte Universitätsbeispiel. Es ist (allerdings nicht ganz vollständig) als Schema in O_2 bekannt und wird im Laufe dieses kleinen Tutorials an einigen Stellen erweitert.

Es ist sinnvoll, regelmäßig die bereits gemachten (Schema-) Änderungen durch Betätigen des COMMIT-Schalters (beim nächsten Fenster muß auch „commit“ angeclickt werden) in die Datenbank zu schreiben. (Dabei verschwinden kurzzeitig alle Fenster.)

Schema und Datenbank auswählen Zu Beginn einer O_2-Sitzung muß ein Schema selektiert werden. Dieses geschieht über den Browser `Schemas`, welcher die verfügbaren Schemata anzeigt. Wir selektieren (mit der Maus) das Schema `universitaet` (bzw. `universitaet2`) und machen dieses Schema zum aktuellen über den Menüpunkt `Schema:set`.

Danach wird in O_2Tools als aktuelles Schema `universitaet` angezeigt.

Klassenhierarchie Die Klassen des aktuellen Schemas kann man sich über den Browser `Classes` ansehen. Die Klassen werden hier aufgelistet, bzw. in einer Klassenhierarchie graphisch angezeigt. Durch Selektion einer Klasse und anschließenden Aufruf von `classes:edit` kann man sich den Typ der Klasse ansehen. In der *description* oberhalb der graphischen Anzeige ist jeweils der Typ der selektierten Klasse in alphanumerischer Form zu sehen.

Wir wollen eine neue Klasse als Subklasse von `Person` erzeugen. Dazu selektieren wir zunächst die Klasse `Person` und aktivieren im Menü den Punkt `classes:create`. Die neue Klasse wird dann als Subklasse von `Person` angelegt. Wir werden aufgefordert, einen Klassennamen einzugeben, in diesem Fall ist es der `Dozent`, der in dem Schema noch fehlt, und bestätigen den Namen mit <`return`> (oder Clicken des OK-Buttons). Sodann finden wir uns im Klasseneditor wieder, wo wir als *Type* von `Dozent` eingeben:

```
tuple (Titel:  string,
Gehaltsgruppe:  string,
Veranstaltung:  set(Veranstaltung))
```

Wir deklarieren diese Klasse als `public` über die `Visibility`-Leiste und übersetzen die Klasse mit Hilfe des Menüpunktes `class:compile`. Tritt kein Fehler auf, so erhalten wir im unteren Teil des Klassen-Editors die `Message`

```
class compilation
no error
```

Mit `Window:close` kann der Klassen-Editor verlassen werden.

Bestehende Klassen können später noch modifiziert werden; bspw. können wir den Typ der Klasse *Gebäude*, welcher gegenwärtig ein Tupel bestehend aus den Attributen `Sockwerke: integer, Fahrradstaender: boolean` und `Parkplaetze: boolean` ist, um das Attribut `Adresse: Adresse` erweitern (Achtung: Dieser Typ weicht von dem in der Übung 1 ab!). Eine erfolgreiche Übersetzung der so modifizierten Klasse ist jedoch nur dann möglich, wenn zuvor die Klasse oder der Typ `Adresse` angelegt wurde. Wie man einen Typ anlegt, werden wir am Beispiel des Typs `Adresse` als nächstes erläutern.

Typen Wir aktivieren den Browser `Types` und wählen `types:create` im Menü aus. Als Typname geben wir sodann `Adresse` an und spezifizieren im Typ-Editor

```
tuple(Strasse: string, Ort: string)
```

als Struktur von `Adresse`.

Der so angelegte Typ wird mit `type:compile` übersetzt und wir verlassen den Typ-Editor mit `Window:close`.

Da der Typ `Adresse` nun existiert, kann die Klasse `Gebäude` mit dem zusätzlichen Attribut `Adresse: Adresse` übersetzt werden. Dazu gehen wir noch einmal in den Klassen-Browser, selektieren dort die Klasse `Gebäude` und aktivieren im Menü `classes:edit`. Im Klassen-Editor rufen wir nach Einfügen des genannten Attributs `class:compile` auf. Nach erfolgreicher Übersetzung (es erscheint dabei die Meldung, daß ein Default-Value für Adresse gesetzt wurde) haben wir die Klasse *Gebäude* somit um ein Attribut erweitert.

Mit `classes:Confirm All` werden die Klassendefinitionen endgültig in das Schema übernommen.

Persistenz von Objekten Bisher haben wir lediglich Schemamodifikation erläutert. Jetzt geht es um die Instanzen dieses Schemas, das heißt, um die eigentliche Datenbank. Zu jedem Schema können eine oder mehrere Instanzen gehören. Diese werden als *base* bezeichnet und analog zum Schema ausgewählt. Hier wird im Schemabrowser die Datenbank `uebung1` (analog bei Schema `universitaet2` ist es `uebung2`) ausgewählt und über `bases:set` gesetzt.

Nun kann mit persistenten Objekten gearbeitet werden, wozu der letzte in diesem Tutorial vorgestellte Browser dient. Es ist der Browser für *Namen*. Namen sind persistent und dienen als Einstieg in die Datenbank. Dies bedeutet insbesondere, daß Instanzen nur dann persistent sind, wenn sie über einen Namen direkt oder indirekt zugreifbar sind.

Wir haben hier für die Objekte einer jeden Klasse einen Namen angelegt, bspw. ist `Personen` der Name für die Menge aller Objekte in der Klasse `Person`. Wird eine Instanz der Klasse `Person` erzeugt, und soll diese persistent sein, so kann dies z. B. durch das Einfügen in die Mengen `Personen` geschehen.

Den Wert eines Namens kann man sich über `Names:Display Value` ansehen; bspw. falls der Name `Personen` selektiert wurde, bekommt man mit `Names:Display Value` ein Anwendungsfenster, welches (Beschreibung s.u.) das Anzeigen und Editieren von Werten erlaubt.

Ein häufiger Fehler, der hier auftritt, ist der, daß versucht wird, auf ein `Nil`-Objekt zuzugreifen. Dies hat einen Transaktions-Abbruch zur Folge. Dabei verschwinden alle O_2 Fenster vom Bildschirm, kommen jedoch kurz darauf wieder zurück. Die Änderungen seit dem letzten Commit sind jedoch verloren.

Für den Dozenten muß jetzt ein solcher Name angelegt werden. Dies erfolgt durch Eingabe von `Names:create`. Nach Angabe des Namens wird das Editierfenster geöffnet, wo der Typ (analog zu den Klassendefinitionen) eingegeben wird. Er lautet hier `set(Dozent)`. Damit wird dem Namen `Dozenten` eine – bisher leere – Menge von Dozenten zugeordnet. Mit `Name:Compile` wird der Name übersetzt.

Applikationen [1] Ein Applikationsfenster – was z.B. bei der Wahl von `Names:Display Value` (s.o.) geöffnet wird – besitzt als Default immer zwei *Buttons*, welche das Wegschreiben ("Stift") in die Datenbank, bzw. das Löschen ("Radiergummi") der soeben gemachten Änderungen symbolisieren.

In einem Applikationsfenster arbeitet man durch Klicken mit der rechten Maustaste, wodurch ein Menü sichtbar wird. Ein einmal erzeugtes Element kann angezeigt oder auch editiert werden, indem man wieder die rechte Maustaste aktiviert. Bspw. erlaubt das Editieren die Eingabe von Name und Alter der Person.

Die Anwendungsfenster werden durch Aktivieren des "Schreib"- oder des "Lösch"-Buttons wieder verlassen.

Damit wollen wir an dieser Stelle die Ausführungen zu den Browsern in O_2 beenden.

O_2-Shell Abschließend wollen wir über die O_2Shell noch auf Manipulations- und Anfragemöglichkeiten an die vorhandene O_2-Datenbank eingehen.

[1] Für die Übung noch nicht so von Interesse, das würde den Rahmen sprengen, wir arbeiten überwiegend mit der O_2-Shell

Zuerst sollen Instanzen von *Personen* (das ist ein existierender Name) eingegeben werden. Dazu muß ein sogenannter *body* in O_2C geschrieben werden. Im `Input`-Fenster wird dafür

```
run body {
o2 Person p = new Person;
p->Name = “Oberwichtig“;
p->Alter= 65;
Personen += set(p);
p->edit; }
```

In der ersten Zeile des Programmtextes des Body wird eine Variable `p` vom o2-Typ `Person` definiert und mit `new` angelegt. Wie in C üblich, kann nun (in den nächsten beiden Zeilen) auf die einzelnen Komonenten zugegriffen werden. Mit `Personen += set(p);` wird die vorhandene Menge `Personen` um die neu angelegte erweitert. Interessant ist die letzte Zeile: mit `edit` wird ein Applikationsfenster geöffnet (s.o. voriger Abschnitt), in dem die gerade gesetzten Werte verändert werden können. Anmerkung: Die einzelnen Komponenten müssen nicht im Body gesetzt werden. Der Aufruf von `p->edit` öffnet das Applikationsfenster auch zum Eingeben neuer Werte.

Um den Body nun ausführen zu können, wird `Execution: Run` selektiert. Nach dem Wegschreiben im Applikationsfenster ist nun eine neue Instanz in der Personenmenge vorhanden. Mit

```
run body{
display (Personen);}
```

wird ein Applikationsfenster mit allen Personen (jeweils mit *Person* gekennzeichnet) geöffnet. Durch Anclicken einer Person, anschließenden Öffnen des Menüs mit der rechten Maustaste und Auswahl von `display` wird der Wert des betreffenden Objekts angezeigt. Dies nur der Vollständigkeit halber - eine ausführliche Beschreibung der Applikationen führt hier zu weit. Wenn nacheinander alle Fenster der Applikation wieder geschlossen sind, kann in der O_2-Shell weitergearbeitet werden.

Analog soll nun ein Dozent angelegt werden:

```
run body {
o2 Dozent d = new Dozent;
d->Name = “Oberwichtig“;
d->Alter= 65;
Personen += set(d);
Dozenten += set(d); p->edit; }
```

Hinweis: Es existieren jetzt zwei Personen (plus eine, die bereits vorher vorhanden war!) mit dem Namen „Oberwichtig“, jedoch ist nur eine Dozent. Bei `run body{ display (Personen);}` wird dies auch kenntlich gemacht.

Nach dem Anlegen von Objekten kann jetzt auch danach gesucht werden: Eine Anfrage der Form "Selektieren der Namen aller Personen" können wir (im `Input`-Fenster) wie folgt formulieren:

```
select p.Name
from p in Personen
```

Durch Auswahl des Menüpunktes `Query:Query` wird diese Anfrage ausgeführt, bzw. wenn wir die Anfrage vorher mit der Maus selektiert haben, wird die Anfrage durch `Query:Query On Selection` ausgeführt. Ratsam ist es, immer zunächst zu selektieren und dann den zuletzt genannten Menüpunkt zu aktivieren.

Mit diesem kurzen Durchgang durch das System wollen wir die Beschreibung von O_2 vorerst abschließen.

3 Weitere Dokumentation

Die wichtigste Literatur zum $\mathrm{O_2}$ DBMS besteht aus dem $\mathrm{O_2}$C-Beginner's Guide, dem $\mathrm{O_2}$Tools User Manual und dem OQL User Manual – sie sind im IfIS vorhanden und werden auf Anfrage ausgeliehen!