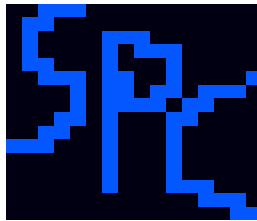


Bedienungsanleitung

SecondoPostGIS Converter

Version 1.0



Autor: Daniel Walther
- München - Herbst 2013 -

Versionen

Version	Status	Datum	Autor
1.0	Final	12.11.13	Daniel Walther

Inhaltsverzeichnis

Versionen.....	I
Abbildungsverzeichnis.....	IV
Tabellenverzeichnis.....	V
Abkürzungsverzeichnis.....	VI
1 Allgemeine Informationen.....	1
1.1 Allgemeines zum SecondoPostGIS Converter.....	1
1.2 Voraussetzungen.....	1
1.3 Externe Programmpakete.....	2
1.4 Automatisch generierte Dateien.....	2
1.4.1 Konfigurationsdatei.....	3
1.4.2 Loggingdateien.....	3
1.4.3 Kopierdateien.....	4
1.5 Starten von SeondoPostGIS Converter.....	5
2 Der SecondoPostGIS Converter.....	6
2.1 Der erste Start.....	7
2.2 Die Hauptansicht der Anwendung.....	7
2.2.1 Menüleiste.....	8
2.2.1.1 File.....	8
2.2.1.1.1 File → Exit.....	8
2.2.1.2 Left.....	8
2.2.1.2.1 Left → Reconnect Server.....	8
2.2.1.2.2 Left → Copy Objects to PostgreSQL.....	8
2.2.1.2.3 Left → Copy Moving Objects to PostgreSQL.....	9
2.2.1.2.4 Left → Generate Moving Objects.....	9
2.2.1.2.5 Left → Secondo Commands.....	9
2.2.1.2.5.1 Left → Secondo Commands → list databases.....	9
2.2.1.2.5.2 Left → Secondo Commands → list types.....	9
2.2.1.2.5.3 Left → Secondo Commands → list type constructors.....	9
2.2.1.2.5.4 Left → Secondo Commands → list objects.....	10
2.2.1.2.5.5 Left → Secondo Commands → list operators.....	10
2.2.1.2.5.6 Left → Secondo Commands → list algebras.....	10
2.2.1.2.6 Left → Supported Types.....	10
2.2.1.3 Right.....	10
2.2.1.3.1 Right → Reconnect Server.....	10
2.2.1.3.2 Right → Copy to SECONDO.....	10
2.2.1.3.3 Right → Show table model.....	11
2.2.1.3.4 Right → Supported Types.....	11
2.2.1.4 Help.....	11
2.2.1.4.1 Help → Set Configuration.....	11
2.2.1.4.2 Help → Parameter Settings.....	12
2.2.1.4.3 Help → About.....	12
2.2.2 Toolbar.....	12
2.2.3 Kontextmenü.....	13
2.2.3.1 Left Side Kontextmenü.....	13

2.2.3.2 Right Side Kontextmenü.....	13
2.2.4 Eingabefelder.....	14
2.3 Unterstützte Datentypen.....	15
2.4 Right Side.....	16
2.4.1 Copy to SECONDO.....	17
2.5 Left Side.....	20
2.5.1 Copy Objects to PostgreSQL.....	21
2.5.2 Copy Moving Objects to PostgreSQL.....	23
2.5.3 Generate Moving Objects.....	25
3 Hinweise zu den Datenbanksystemen.....	28
3.1 Besondere Hinweise.....	28
4 Rechtlicher Hinweis.....	29

Abbildungsverzeichnis

Abbildung 1.1: Konfigurationsdatei mit allen Parametern.....	3
Abbildung 1.2: Auszug aus einer Logdatei, die von SPC erstellt wurde.....	4
Abbildung 2.1: Hauptansicht SecondoPostGIS Converter.....	6
Abbildung 2.2: Beispielergebnis „list databases“.....	9
Abbildung 2.3: Beispielhaftes ERM einer DB mit JGraphX.....	11
Abbildung 2.4: Eingabefenster für die Parameter der Konfigurationsdatei.....	12
Abbildung 2.5: Icons im Programm SPC.....	12
Abbildung 2.6: Ergebnisansicht eines queries an die DB.....	14
Abbildung 2.7: PostgreSQL- to SECONDO-Types.....	15
Abbildung 2.8: SECONDO- to PostgreSQL-Types.....	16
Abbildung 2.9: Auswahlfenster für die Attribute, die von einer Relation kopiert werden sollen.....	18
Abbildung 2.10: Auswahldialog um DB-Name festzulegen.....	19
Abbildung 2.11: Anzeige, dass der DB-Name bereits verwendet wird.....	19
Abbildung 2.12: Eingabefenster für den Objektnamen, falls dieser bereits existiert.....	19
Abbildung 2.13: Fortschrittsanzeige.....	20
Abbildung 2.14: Anzeige, dass der Kopierprozess abgeschlossen ist.....	20
Abbildung 2.15: Auswahlfenster für die Attribute, die von einem Objekt kopiert werden sollen.....	22
Abbildung 2.16: Auswahldialog um DB-Name festzulegen mit Template-Auswahl.....	22
Abbildung 2.17: Eingabefenster für das Kopieren von Moving Objects nach PostgreSQL.....	23
Abbildung 2.18: Zeitauswahl (Monat/Tag/Jahr).....	24
Abbildung 2.19: GUI zum Generieren von Moving Objects.....	25
Abbildung 2.20: Zwei numerische Attribute zusammenführen.....	27
Abbildung 2.21: Moving Objects are generated.....	27

Tabellenverzeichnis

Tabelle 1: Abkürzungsverzeichnis.....	VII
Tabelle 2.1: Kontextmenü Left.....	13
Tabelle 2.2: Kontextmenü Right.....	13
Tabelle 3.1: Produkte und Versionen.....	28

Abkürzungsverzeichnis

Abkürzung	Bedeutung
API	Application Programming Interface
BSD	Berkeley Software Distribution
bzw.	beziehungsweise
CPU	Central Processing Unit
d. h.	das heißt
DB	Datenbank
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
ERM	Entity-Relationship-Modell (ER-Modell)
evtl.	eventuell
GB	Gigabyte
ggf.	gegebenenfalls
GHz	Gigahertz
GIS	Geografisches Informationssystem (Geoinformationssystem)
GNSS	Global Navigation Satellite System
GUI	Graphical User Interface (grafische Benutzeroberfläche)
HTML	Hypertext Markup Language
JDBC	Java Database Connectivity
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MB	Megabyte
MO	Moving Objects
ms	Millisekunde(n)
NoSQL	Not only SQL
o. g.	oben genannte
OQL	Object Query Language
OS	Operating System (Betriebssystem)
PG	PostgreSQL
RAM	Random Access Memory
s	Sekunde(n)
SEC	SECONDO

Abkürzung	Bedeutung
SOS	second-order signature
SPC	SecondoPostGIS Converter
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/ Internet Protocol
usw.	und so weiter
VM	Virtuelle Maschine
WKB	Well-Known Binary
WKT	Well-Known Text
XML	Extensible Markup Language
z. B.	zum Beispiel

Tabelle 1: Abkürzungsverzeichnis

1 Allgemeine Informationen

Aus Gründen der Klarheit und besseren Lesbarkeit wird im Folgenden bei der Nennung von Personen lediglich die maskuline Form verwendet. Alle entsprechenden Begriffe implizieren jedoch sowohl die feminine als auch die maskuline Form.

Im nachfolgenden Verlauf wird immer nur *PostgreSQL* anstatt von *PostgreSQL/PostGIS* geschrieben. Sollten Unterschiede oder Besonderheiten erklärt werden, so wird dies explizit beschrieben. Ansonsten ist davon auszugehen, dass, wenn von *PostgreSQL* geschrieben wird, dies auch für *PostGIS* zutreffend ist.

1.1 Allgemeines zum SecondoPostGIS Converter

Das Programm SecondoPostGIS Converter dient der Überführung von Datensätzen zwischen den beiden Datenbanksystemen *SECONDO* und *PostgreSQL*. Im weiteren Verlauf dieser Dokumentation wird erläutert, wie das Programm funktioniert.

SecondoPostGIS Converter (SPC) wurde für unterschiedliche Betriebssystemplattformen entwickelt und getestet. Dazu zählen Linux (Debian 6, Ubuntu 12), Mac OS X (Snow Leopard 10.6.8) und Microsoft Windows (Windows 7). SPC wurde nicht unter Microsoft Windows 8 getestet.

SPC verfügt zur besseren Anwender- und Fehleranalyse über ein Logverfahren der programmrelevanten Funktionsaufrufe und -abläufe.

Die durchgängige Systemsprache im Programm ist Englisch.

Im Verlauf dieser Dokumentation werden immer wieder Systempfade oder Befehle dargestellt. Sollte keine spezielle Kennzeichnung erfolgen, so sind diese Befehle für ein Linux Betriebssystem (OS).

Das Programm wurde unter dem Projektnamen *Hjort* entwickelt. Dieser Name tritt an einigen Stellen auf.

1.2 Voraussetzungen

Damit SPC ausgeführt werden kann ist eine Java Runtime Umgebung (JRE) notwendig, die mindestens in der Version 5 (Java 1.5) vorliegen sollte. Zudem benötigt SPC Schreibrechte im `$HOME`- und dem `Java-temp-Verzeichnis`. Der Grund dafür wird im Folgenden noch erklärt.

Außerdem ist ein Lese- und Schreibrecht für die beiden Datenbanksysteme erforderlich, wenn der Datentransfer in beide Richtungen funktionieren soll. Es wird angeraten, vor dem Verwenden von SPC ein Backup des Datenbanksystems anzufertigen.

1.3 Externe Programmpakete

Das ausführbare Programm ist in eine *jar*-Datei gepackt. Diese enthält wiederum drei weitere *jar*-Dateien, die mit eingebunden sind. Alle drei sind kostenlos verfügbar und für die Ausführung von SPC notwendig. Namentlich genannt ergibt sich folgendes Bild:

`SecondoInterface.jar` – Bereitstellung von Java-Klassen zur Verbindung und Kommunikation mit einem *SECONDO*-Datenbanksystem.

`postgresql-9.2-1002.jdbc4.jar` – Für die Verbindung und Kommunikation zu einem *PostgreSQL*-Datenbanksystem über JDBC.

`jgraphx.jar` – Graphenbibliothek, die für die grafische Darstellung von *PostgreSQL*-Relationen und deren Beziehungen verwendet wird (<http://www.jgraph.com>).

1.4 Automatisch generierte Dateien

Dieser Punkt ist nichts Kompliziertes oder Spektakuläres, sondern soll nur einen Überblick verschaffen, wie und wo welche Dateien und Ordner von dem Programm SPC angelegt oder verändert werden. In den Pfadangaben der nachfolgenden drei Unterpunkte sollte der User über Lese- und Schreibrechte verfügen. Da nur zwei Standardpfade zur Anwendung kommen, sollte o. g. Bedingung immer gegeben sein, wenn das System nicht irgendwelche Besonderheiten enthält. Als Beispiel dafür sei eine besondere Systemhärtung genannt, die Lese- und Schreibrechte für ausführbare Programme einschränkt.

Für die ersten beiden Unterpunkte ist der Pfad `$HOME` relevant. Als Beispiel: `/home/<username>/`

In diesem Verzeichnis wird immer dann, wenn der Ordner `.Hjort` nicht vorhanden ist, einer mit diesem Namen angelegt. In dem Ordner werden dann weitere Dateioperationen stattfinden.

Für den dritten Menüpunkt ist das `Java-temp-Verzeichnis` relevant, das je nach OS variieren kann. Ein Beispiel für eine Debian Installation ist `/tmp/`.

Die beiden beschriebenen Dateitypen aus den folgenden zwei Unterpunkten sind auch nach Beendigung des Programms existent.

1.4.1 Konfigurationsdatei

Die Konfigurationsdatei ist unter `$HOME/.Hjort/` gespeichert und trägt den Namen `.hjort.cfg`. In der Konfigurationsdatei sind Verbindungsparameter für die Verbindung zu den beiden Datenbanksystemen gespeichert. Diese werden immer dann nacheinander abgefragt, wenn die Datei nicht vorhanden oder nicht lesbar ist. Die `.hjort.cfg` ist eine herkömmliche Textdatei. Daher kann sie mit einem Texteditor aufgerufen und bearbeitet werden. Jedoch ist zu beachten, dass das Passwort kryptiert abgelegt ist. Dies kann also nur über den entsprechenden Menüpunkt im Programm bearbeitet werden.

Folgende Abbildung zeigt beispielhaft eine Konfigurationsdatei für den SPC.

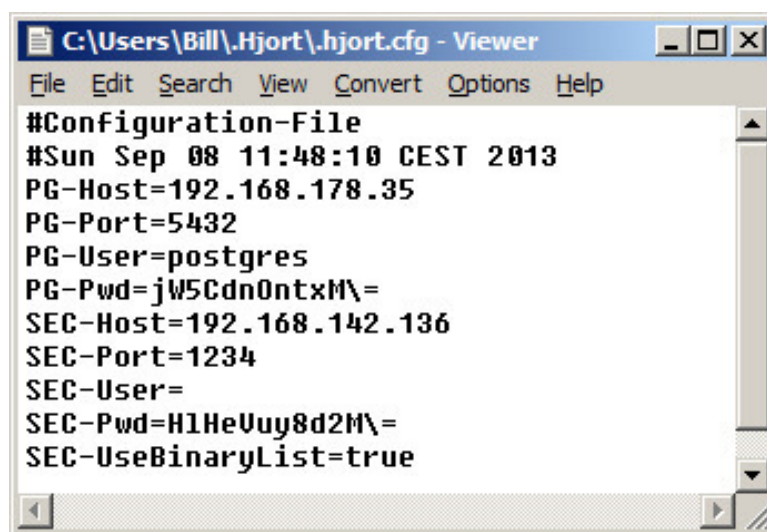


Abbildung 1.1: Konfigurationsdatei mit allen Parametern

1.4.2 Loggingdateien

Wie viele moderne Programme verfügt auch SPC über ein Logverfahren. Aufgrund der Plattformunabhängigkeit, die ein Ziel dieser Anwendung war, wird das Logging in Dateien durchgeführt. Diese Dateien findet der Anwender in dem Verzeichnis `$HOME/.Hjort/`. Aus den Logdateien können Entwickler und ggf. auch Anwender Fehler im Programm oder in Datentypen erkennen.

Damit die Logdateien auch nach einer häufigen Benutzung der Anwendung einen überschaubaren Speicherplatz belegen, wurde das *File-Rotate-Verfahren* aktiviert. Das heißt, eine Datei wird maximal 10 MB groß und es werden maximal zehn Dateien gespeichert. Der Dateiname einer Logdatei sieht so aus: `„.hjort.log.<x>“` ($x=[0-9]\{1\}$)¹. Die Logdatei liegt im XML-Format vor.

¹ Das Zeichen x steht für eine natürliche Zahl zwischen 0 und 9.

Ein kurzer Auszug aus dieser Datei sollte an dieser Stelle genügen.

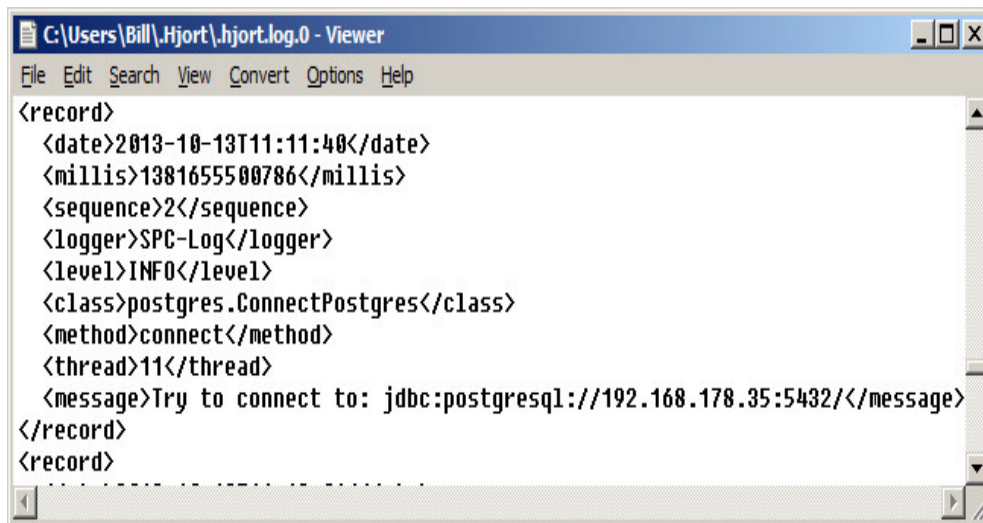


Abbildung 1.2: Auszug aus einer Logdatei, die von SPC erstellt wurde

Des Weiteren erfolgt die Ausgabe der Lognachrichten während der Laufzeit des Programms auf der Konsole.

1.4.3 Kopierdateien

Diese Dateien werden bei jedem Neustart des Programms SPC selbstständig gelöscht. Daher werden diese im `Java-temp-Verzeichnis` des Betriebssystems abgespeichert. Ihr Inhalt wird nur kurzzeitig benötigt. Durch SPC wird hier eine Vielzahl von Dateitypen angelegt. Alle Dateien können mit einem Texteditor geöffnet werden. Folgende Dateien können bei den Kopiervorgängen angelegt werden:

`sec2pg<[0-9]*>.csv` – Das ist die `csv`-Datei, die das Ergebnis einer query-Anfrage an eines der beiden Datenbanksysteme enthält sowie die dazugehörigen Metadaten des Ergebnisses.

`seccommands<[0-9]*>.sec` – Mit dieser Datei kann entweder ein einzelnes Objekt oder eine ganze Datenbank im `SECONDO`-System erzeugt werden.

`pgcommands<[0-9]*>.psql` – Diese Datei enthält SQL-Befehle, die für das Erzeugen von einer Tabelle und deren Inhalten in dem `PostgreSQL`-System benötigt werden.

1.5 Starten von SeondoPostGIS Converter

Für das Starten der Anwendung liegt der ausführbaren *jar*-Datei ein Startup-Skript bei. Für ein Linux OS ist diese die „startup.sh“ und für ein Microsoft Windows OS die „startup.bat“. Es wird dringend empfohlen, diese Dateien für das Starten von SPC zu verwenden, da dadurch notwendige Parameter mit an die Java Virtual Machine (JVM) übergeben werden. Zudem ist zu beachten, dass die Dateien ausführbar sind. Der SPC lässt sich mit folgendem Kommando starten:

Allgemein:

```
java <VM Parameter> -jar <Pfad/> Hjort.jar
```

Für SPC:

```
java -Djava.util.Arrays.useLegacyMergeSort=true ↵  
-XX:+AggressiveHeap -jar Hjort.jar
```

2 Der SecondoPostGIS Converter

Wie bereits erwähnt, handelt es sich bei dieser Anwendung um ein Java-Programm. Die Hauptansicht des Programms stellt bei einer bestehenden Verbindung zu den zwei Datenbanksystemen (*SECONDO* und *PostgreSQL*) diese jeweils in einem *JTree*-Element dar. Zudem enthält die Hauptansicht eine Menüleiste sowie eine Toolbar. Darüber hinaus ist für die Datenbankansicht von *SECONDO* und *PostgreSQL* ein unterschiedliches Kontextmenü verfügbar, das der Nutzer durch einen Rechtsklick auf ein einzelnes *JTree*-Element aufrufen kann. Unterhalb der Datenbankansicht/-schema ist auf jeder Seite ein Eingabefeld vorhanden, aus dem ausschließlich „queries“ an die Datenbanken abgesetzt werden können.

In den Kapiteln 2.3 und 2.4 werden die für den Datentransfer notwendigen Fenster und Abläufe detailliert beschrieben.

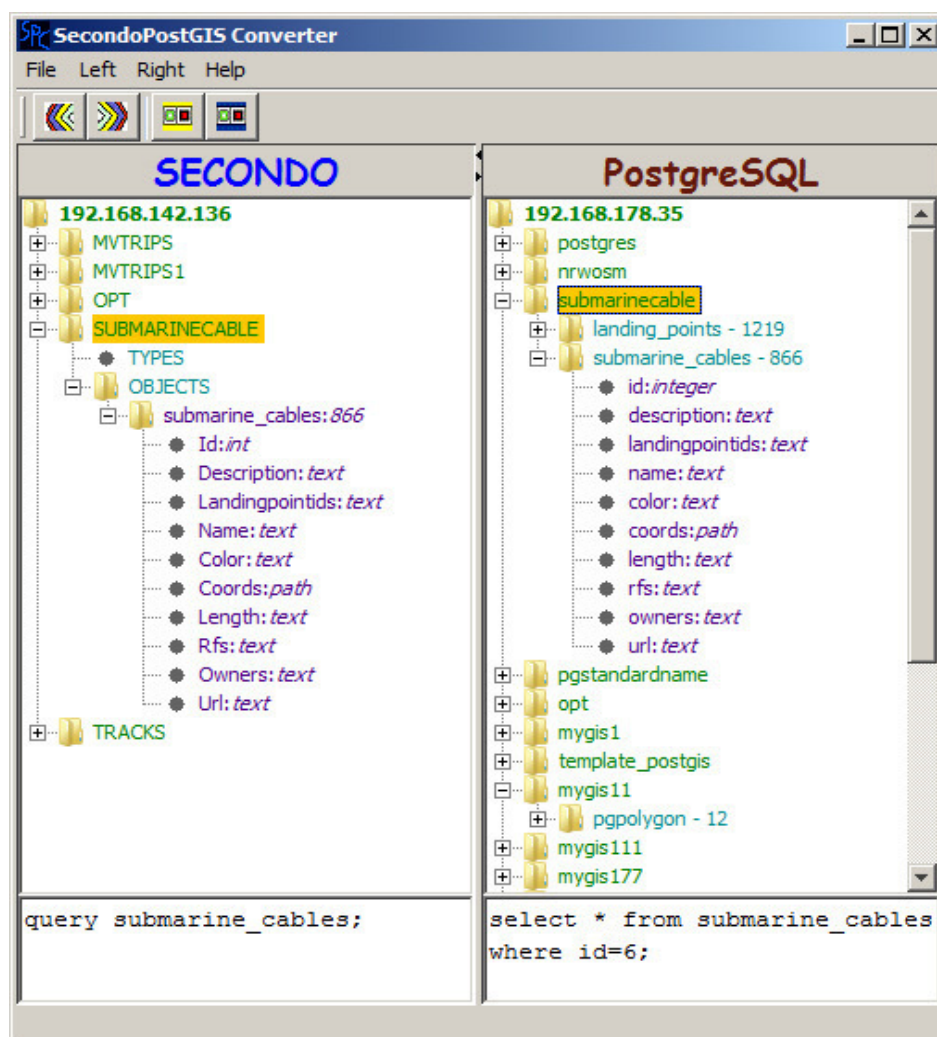


Abbildung 2.1: Hauptansicht SecondoPostGIS Converter

2.1 Der erste Start

Hier sind wir jetzt beim ersten Programm-Start von SPC angekommen. Je nach OS muss hierfür die entsprechende *startup*-Datei ausgeführt werden. Nach erfolgreichem Starten wird der Anwender nacheinander zur Eingabe der Verbindungsparameter aufgefordert. Sobald er diese alle eingegeben hat und sie korrekt sind, versucht SPC die Verbindung herzustellen und das Datenbankschema beider Systeme auszulesen und darzustellen. Dies kann je nach Verbindung, Umfang der Datenbank sowie Rechenleistung des Datenbanksystems eine Weile dauern.

Sollte ein Fehler in der Verbindung auftreten, wird dies für jedes Datenbanksystem (DBS) signalisiert.

Die Verbindungsparameter können unter dem Menüpunkt „*Help* → *Parameter Settings*“ erneut eingegeben werden. Dazu später mehr.

2.2 Die Hauptansicht der Anwendung

In der Hauptansicht sind alle grundlegenden Menüpunkte, die für einen Kopierprozess notwendig sind, erreichbar. Zudem gibt es auf den mit *Left* (*SECONDO*) und *Right* (*PostgreSQL*) *Side* benannten Baumansichten verschiedene Kontextmenüpunkte. Als Beispiel sei der Punkt „*Generate Moving Objects*“ auf der *Left Side* genannt, den es auf der *Right Side* nicht gibt. Bevor die Erklärung des Programms ins Detail geht, soll kurz die Darstellung des Datenbankschemas erklärt werden.

Da sich beide Datenbanksysteme unterscheiden, ist natürlich nur eine ähnliche Darstellung der beiden Seiten möglich. Aus dem dargestellten Schema lässt sich die Struktur der beiden Datenbanksysteme erkennen. Auf der *SECONDO*-Seite sind dies die Datenbanken mit den Typen, Objekten, Attributen der Objekte sowie die Datentypen der Attribute, die von dem Datenbanksystem gelesen werden konnten. Soweit möglich wird für jedes Objekt die Anzahl der Tupel ermittelt. Analog dazu bildet sich folgendes Datenbankschema für *PostgreSQL*-/ *PostGIS*-Darstellung. Auch hier werden für die Darstellung die Datenbanknamen mit Tabellen, deren Spaltennamen und die Datentypen der Spaltennamen, die von dem Datenbanksystem gelesen werden konnten, dargestellt. Soweit es möglich ist, wird für jede Tabelle die Anzahl der Tupel bestimmt sowie eine Kennzeichnung von Spalten, die als primary- oder foreign key fungieren.

2.2.1 Menüleiste

In der Menüleiste sind alle wichtigen Punkte vorhanden, damit der Anwender Daten von *PostgreSQL* nach *SECONDO* oder umgekehrt kopieren kann. Die Zeitdauer der Kopiervorgänge ist von mehreren Faktoren abhängig:

- Rechenleistung des Clients, Quell- und Zielservers
- Umfang der Datenmenge (Anzahl Tupel, Datentypen sowie deren Attribute)

Die nachfolgenden Menüpunkte im Kapitel 2.2.1 stellen die Menüleiste im Detail dar.

2.2.1.1 File

Unter diesem Menüpunkt findet der Anwender nicht viel. Es ist lediglich ein Menüpunkt, der für das Schließen der Anwendung genutzt werden kann.

2.2.1.1.1 File → Exit

Schließt die Anwendung SPC und löscht ggf. die Dateien, die durch Kopiervorgänge automatisch erzeugt wurden.

2.2.1.2 Left

Der Menüpunkt „*Left*“ bezieht sich nur auf die linke Seite der Anwendung. Alle aufgeführten Menüpunkte sind auf das linke Datenbankschema anwendbar.

2.2.1.2.1 Left → Reconnect Server

Hiermit lässt sich die Datenbankverbindung erneut herstellen und es wird das linke Datenbankschema aktualisiert. Dies führt ein Update der Ansicht der linken Seite durch. Dieser Befehl sollte ausgeführt werden, wenn die Verbindungsparameter zum *SECONDO*-DBS geändert wurden.

2.2.1.2.2 Left → Copy Objects to PostgreSQL

Hiermit lässt sich das Fenster aufrufen, das für das Kopieren nach *PostgreSQL* zuständig ist. Anzumerken sei, dass durch dieses Fenster *Moving Objects* in *PostgreSQL* nur als `text`-Datentyp (`String`) abgelegt werden. Ist dies jedoch nicht erwünscht, so muss der Anwender den Menüpunkt wählen, der speziell für das Kopieren von *Moving Objects* zuständig ist.

2.2.1.2.3 Left → Copy Moving Objects to PostgreSQL

Mit diesem Aufruf wird das Fenster geöffnet, womit *Moving Object*-Datentypen in ihre ursprünglichen Datentypen überführt werden können. Dies bedeutet, dass aus einem *Moving Object* (MO), das mindestens zwei Zeitpunkte enthält, evtl. mehrere Tupel in einer *PostgreSQL*-Datenbank (DB) generiert werden. Weitere Erläuterungen dazu folgen später in diesem Dokument.

2.2.1.2.4 Left → Generate Moving Objects

Das Fenster, das nach diesem Aufruf folgt, dient dazu, um *Moving Objects* aus bereits existierenden Datensätzen eines *SECONDO*-Objektes zu erstellen.

2.2.1.2.5 Left → Secondo Commands

Hier findet der Anwender verschiedene nützliche Menüpunkte wieder, die es ihm ermöglichen, ein besseres Bild über das Datenbankschema einer *SECONDO*-DB zu erhalten.

2.2.1.2.5.1 Left → Secondo Commands → list databases

Sendet den Befehl `list databases` an das *SECONDO*-DBS und stellt das Ergebnis wie folgt dar:

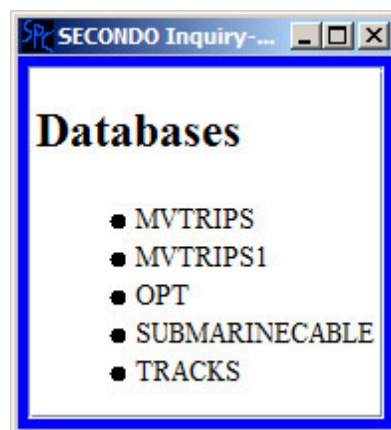


Abbildung 2.2: Beispielergebnis „list databases“

2.2.1.2.5.2 Left → Secondo Commands → list types

Durch Betätigen dieses Menüpunktes werden alle Typen zu der ausgewählten Datenbank angezeigt. Dies entspricht dem *SECONDO*-Befehl `list types`.

2.2.1.2.5.3 Left → Secondo Commands → list type constructors

Sobald dieser Menüpunkt ausgelöst wird, sendet SPC den Befehl `list type constructors` an das *SECONDO*-DBS und präsentiert das Ergebnis in einem eigenen Fenster.

2.2.1.2.5.4 Left → Secondo Commands → list objects

Das Drücken des Punktes „*list objects*“ in der Menüleiste leitet das Kommando `list objects` an das *SECONDO*-DBS. Dazu muss der Anwender eine Datenbank im Datenbankschema selektiert haben.

2.2.1.2.5.5 Left → Secondo Commands → list operators

Der Anwender erhält nach Aufruf dieses Menüpunktes das Ergebnis des Befehls `list operators`, der an das *SECONDO*-DBS geschickt wird, in einem Fenster dargestellt.

2.2.1.2.5.6 Left → Secondo Commands → list algebras

Der Aufruf von „*list algebras*“ führt wie zu erwarten den entsprechenden Befehl aus, um dem Nutzer die Algebra-Module des *SECONDO*-Datenbanksystems anzuzeigen.

2.2.1.2.6 Left → Supported Types

Ruft das Fenster auf, das die Datentypen anzeigt, die von dem Programm SPC unterstützt werden. Aus dieser Tabellenansicht ist ersichtlich, welcher Datentyp einem anderen bei der Überführung von *SECONDO* nach *PostgreSQL* zugeordnet wird. Die Tabelle stellt also dar, welchen Datentypen der Ausgangsdattentyp nach dem Kopiervorgang im anderen Datenbanksystem hat. Mehr dazu im Kapitel „Unterstützte Datentypen“.

2.2.1.3 Right

Der Menüpunkt „*Right*“ bezieht sich nur auf die rechte Seite der Anwendung; anders gesagt, in allen Unterpunkten unter „*Right*“ findet der Anwender Menüpunkte, die sich nur auf das rechte Datenbankschema anwenden lassen.

2.2.1.3.1 Right → Reconnect Server

Hiermit lässt sich die Datenbankverbindung erneut herstellen. Dieser Befehl sollte ausgeführt werden, wenn die Verbindungsparameter zum *PostgreSQL*-DBS geändert wurden. Es wird zudem das Datenbankschema in der rechten Seite der Ansicht aktualisiert.

2.2.1.3.2 Right → Copy to SECONDO

Dieser Menüpunkt ist wahrscheinlich einer der wichtigsten Menüpunkte auf der rechten Seite, denn mit diesem lässt sich das Fenster zum Kopieren einer Relation aufrufen. Damit dieser Menüpunkt ausgeführt werden kann, muss der Anwender zuvor eine Relation einer Datenbank ausgewählt haben.

2.2.1.3.3 Right → Show table model

Hiermit kann ein grafisches Bild einer DB erzeugt und angezeigt werden. Dazu wird mit Hilfe der JGraphX-Bibliothek ein Entity-Relationship-Modell (ERM, ER-Modell) der ausgewählten Datenbank erstellt.

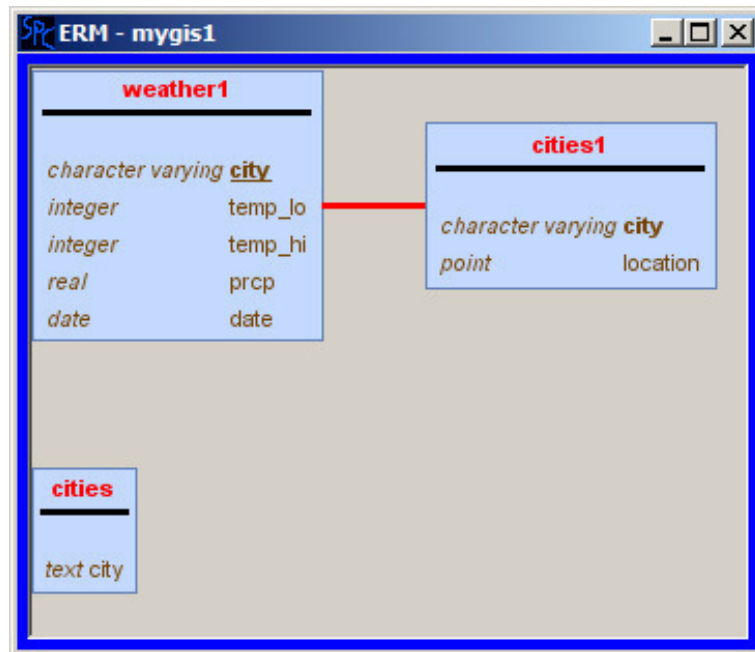


Abbildung 2.3: Beispielhaftes ERM einer DB mit JGraphX

2.2.1.3.4 Right → Supported Types

Analog zu dem Menüpunkt „*Left → Supported Types*“ wird durch Aufruf dieses Menüpunktes ein Fenster geöffnet. Der Inhalt ist die Zuordnung der Datentypen, die bei einem Kopiervorgang von PostgreSQL nach SECONDO angewendet werden.

2.2.1.4 Help

Unter diesem Menüpunkt können unter anderem die Verbindungsparameter zu den Datenbanksystemen neu gesetzt werden.

2.2.1.4.1 Help → Set Configuration

Hierbei werden die einzelnen Verbindungsparameter nacheinander abgefragt. Genau so, wie beim ersten Start der Anwendung. Zudem werden die aktuellen Parameter als Vorauswahl in den Eingabefeldern angezeigt.

2.2.1.4.2 Help → Parameter Settings

Hierbei öffnet sich ein Dialog, in dem alle Parameter, die in der Konfigurationsdatei gespeichert sind, neu gesetzt bzw. angezeigt werden können. Die Passwortfelder haben einen Tooltip, sodass deren Inhalt sichtbar wird.

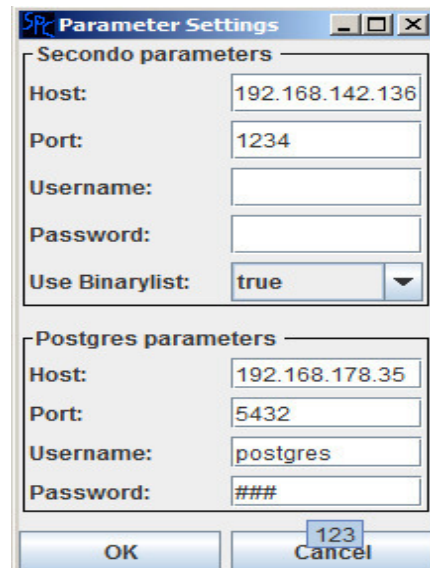


Abbildung 2.4: Eingabefenster für die Parameter der Konfigurationsdatei

2.2.1.4.3 Help → About

Nach Betätigen dieses Menüpunktes bekommt der Anwender ein Fenster, in dem allgemeine Informationen zu SPC dargestellt sind.

2.2.2 Toolbar

Die Toolbar im Programm SPC ist recht überschaubar und verfügt über vier Icons. Die Funktion der Icons wird hier mit den Menüpunkten aus Kapitel 2.2.1 verglichen. Die Icons der folgenden Abbildung werden von links nach rechts erklärt.



Abbildung 2.5: Icons im Programm SPC

Das erste Icon links entspricht dem Menüpunkt „*Right → Copy to SECONDO*“, das nächste dem Menüpunkt „*Left → Copy Objects to PostgreSQL*“. Die anderen beiden Icons sind für den Reconnect zu dem jeweiligen Datenbankserver zuständig.

2.2.3 Kontextmenü

Das Programm verfügt nicht nur in der Hauptansicht über ein Kontextmenü, sondern auch in manchen anderen Fenstern, die zu dem Programm gehören. Diese sollen aber hier nicht extra erwähnt werden. Vielmehr wird das Kontextmenü der *Left*- und *Right Side* vorgestellt und mit den dazugehörigen Menüpunkten verknüpft, soweit diese vorhanden sind.

2.2.3.1 Left Side Kontextmenü

Kontextmenüpunkt	Entspricht Menüpunkt in der Menüleiste
Secondo Commands → list databases	Left → Secondo Commands → list databases
Secondo Commands → list types	Left → Secondo Commands → list types
Secondo Commands → list type constructor	Left → Secondo Commands → list type constructor
Secondo Commands → list objects	Left → Secondo Commands → list objects
Secondo Commands → list operators	Left → Secondo Commands → list operators
Secondo Commands → list algebras	Left → Secondo Commands → list algebras
Secondo Commands → delete database	SIEHE WEITER UNTEN
Copy Objects to PostgreSQL	Left → Copy Objects to PostgreSQL
Copy Moving Objects to PostgreSQL	Left → Copy Moving Objects to PostgreSQL
Generate Moving Objects	Left → Generate Moving Objects

Tabelle 2.1: Kontextmenü Left

Der Punkt „*delete database*“ ist nur im Kontextmenü zu finden. Die Funktion muss nicht weiter erläutert werden. Nach Auslösen dieses Punktes wird der Nutzer gefragt, ob er wirklich eine Datenbank löschen möchte. Wird diese Frage mit „yes“ beantwortet, so wird die selektierte Datenbank auf dem Server gelöscht.

2.2.3.2 Right Side Kontextmenü

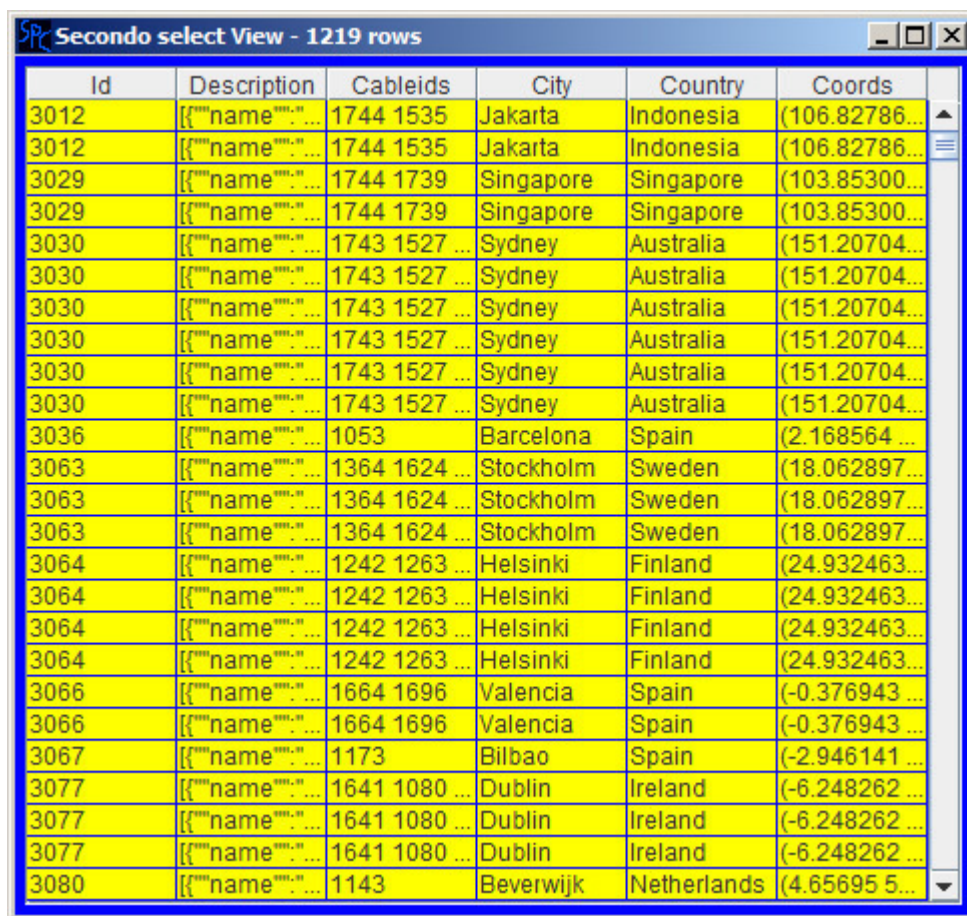
Kontextmenüpunkt	Entspricht Menüpunkt in der Menüleiste
Copy to SECONDO	Right → Copy to SECONDO
Show table model	Right → Show table model
drop database	SIEHE WEITER UNTEN

Tabelle 2.2: Kontextmenü Right

Der Kontextmenüpunkt „*drop database*“ hat das gleiche Verhalten wie „*delete database*“ auf der linken Seite, nur dass dieser eine *PostgreSQL*-DB vom Server löscht.

2.2.4 Eingabefelder

Aus den beiden Eingabefeldern, die in der Hauptansicht von SPC vorhanden sind, können jeweils nur query-Befehle abgesetzt werden. Der Anwender sollte dabei jedoch beachten, dass große „queries“ viel Hauptspeicher benötigen und unter Umständen einige Zeit dauern können. Das Senden erfolgt bewusst nicht in einem eigenen Thread². Es wird vom Programm versucht, den query-Befehl zu analysieren und diesen, wenn nicht vom Nutzer getan, auf eine maximale Anzahl von 50 Datensätzen zu begrenzen. Dies funktioniert nicht immer. Das Ergebnis wird in einer Tabellenansicht dargestellt. Jedes Statement muss mit einem Semikolon abgeschlossen sein.



Id	Description	Cableids	City	Country	Coords
3012	{{name}} ...	1744 1535	Jakarta	Indonesia	(106.82786...
3012	{{name}} ...	1744 1535	Jakarta	Indonesia	(106.82786...
3029	{{name}} ...	1744 1739	Singapore	Singapore	(103.85300...
3029	{{name}} ...	1744 1739	Singapore	Singapore	(103.85300...
3030	{{name}} ...	1743 1527 ...	Sydney	Australia	(151.20704...
3030	{{name}} ...	1743 1527 ...	Sydney	Australia	(151.20704...
3030	{{name}} ...	1743 1527 ...	Sydney	Australia	(151.20704...
3030	{{name}} ...	1743 1527 ...	Sydney	Australia	(151.20704...
3030	{{name}} ...	1743 1527 ...	Sydney	Australia	(151.20704...
3036	{{name}} ...	1053	Barcelona	Spain	(2.168564 ...
3063	{{name}} ...	1364 1624 ...	Stockholm	Sweden	(18.062897...
3063	{{name}} ...	1364 1624 ...	Stockholm	Sweden	(18.062897...
3063	{{name}} ...	1364 1624 ...	Stockholm	Sweden	(18.062897...
3064	{{name}} ...	1242 1263 ...	Helsinki	Finland	(24.932463...
3064	{{name}} ...	1242 1263 ...	Helsinki	Finland	(24.932463...
3064	{{name}} ...	1242 1263 ...	Helsinki	Finland	(24.932463...
3064	{{name}} ...	1242 1263 ...	Helsinki	Finland	(24.932463...
3066	{{name}} ...	1664 1696	Valencia	Spain	(-0.376943 ...
3066	{{name}} ...	1664 1696	Valencia	Spain	(-0.376943 ...
3067	{{name}} ...	1173	Bilbao	Spain	(-2.946141 ...
3077	{{name}} ...	1641 1080 ...	Dublin	Ireland	(-6.248262 ...
3077	{{name}} ...	1641 1080 ...	Dublin	Ireland	(-6.248262 ...
3077	{{name}} ...	1641 1080 ...	Dublin	Ireland	(-6.248262 ...
3080	{{name}} ...	1143	Beverwijk	Netherlands	(4.65695 5...

Abbildung 2.6: Ergebnisansicht eines queries an die DB

Angedacht sind diese Eingabefelder dafür, dass der Anwender sich einen kleinen Überblick über die entsprechenden Inhalte der Attribute einer Relation bzw. eines Objektes verschaffen kann.

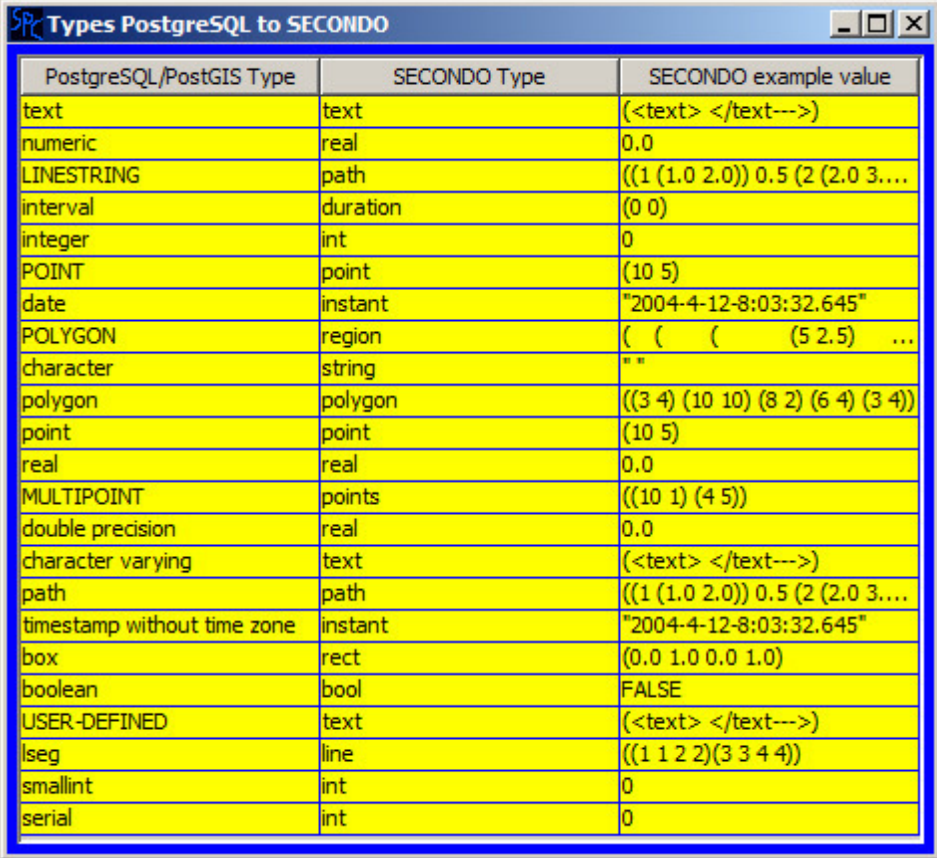
Sollte keine Begrenzung der Datenmenge vom Programm her möglich sein, ist der Nutzer dafür selbst verantwortlich.

² Ähnlich einem Prozess, die fast parallel ablaufenden Aktivitäten ergeben den Eindruck von Gleichzeitigkeit während der Ausführung.

Damit die Befehle aus den Eingabefeldern ausgeführt werden, muss sich der Cursor im entsprechenden Eingabefeld befinden und der Anwender „*CTRL + EINGABE*“ Taste oder die Taste „*F5*“ drücken. Zudem ist zu beachten, dass in der Ansicht des Datenbankschemas eine Datenbank ausgewählt oder geöffnet ist. Die Taste „*F12*“ löscht das Eingabefeld wieder.

2.3 Unterstützte Datentypen

Jedes der beiden Datenbanksysteme besitzt eine Vielzahl von Datentypen. Bei dem Projekt *Hjort* stand die Überführung von Bewegungsdaten im Vordergrund. Daher ist aktuell für die Überführung nur eine Auswahl an Datentypen von einem in das andere DBS und umgekehrt möglich. Die beiden nachstehenden Grafiken zeigen, welcher Typ in welchen überführt wird. Zudem ist jeweils ein beispielhafter Wert eines jeden Datentyps in einer Spalte dargestellt. Datentypen, die nicht in diesem Fenster aufgeführt sind, werden als Ausgangsdattentyp „*USER-DEFINED*“ behandelt. So werden Datentypen, die keine Konvertierungs-/ Überführungsregel haben, als `text`-Datentyp kopiert.



PostgreSQL/PostGIS Type	SECONDO Type	SECONDO example value
text	text	(<text> </text--->)
numeric	real	0.0
LINESTRING	path	((1 (1.0 2.0)) 0.5 (2 (2.0 3....
interval	duration	(0 0)
integer	int	0
POINT	point	(10 5)
date	instant	"2004-4-12-8:03:32.645"
POLYGON	region	((((5 2.5) ...
character	string	" "
polygon	polygon	((3 4) (10 10) (8 2) (6 4) (3 4))
point	point	(10 5)
real	real	0.0
MULTIPOINT	points	((10 1) (4 5))
double precision	real	0.0
character varying	text	(<text> </text--->)
path	path	((1 (1.0 2.0)) 0.5 (2 (2.0 3....
timestamp without time zone	instant	"2004-4-12-8:03:32.645"
box	rect	(0.0 1.0 0.0 1.0)
boolean	bool	FALSE
USER-DEFINED	text	(<text> </text--->)
lseg	line	((1 1 2 2)(3 3 4 4))
smallint	int	0
serial	int	0

Abbildung 2.7: PostgreSQL- to SECONDO-Types



SECONDO Type	PostgreSQL/PostGIS Type	PostgreSQL example value
text	text	' '
int	integer	0
line	lseg	'[(2 , 2),(3 , 3)]'
label	character varying(48)	' '
rect	box	'((1.0,1.0),(0.0,0.0))'
date	timestamp without time zone	'2004-4-12 8:03:32.645'
polygon	polygon	'((1.0,2.0),(2.0,3.0))'
real	numeric	0.0
point	point	POINT(0.0 , 0.0)
duration	interval	'0ms'
string	character varying(48)	' '
path	path	'[(1.0,2.0),(2.0,3.0)]'
USER-DEFINED	text	' '
bool	boolean	false
instant	timestamp without time zone	'2004-4-12 8:03:32.645'
POSTGIS DATA TYPES	POSTGIS DATA TYPES	
polygon	POLYGON	'POLYGON((0 0,0 10,10 10,1...
region	POLYGON	'POLYGON((0 0,0 10,10 10,1...
point	POINT	'POINT(0.0 0.0)'
path	LINESTRING	'LINESTRING(10 10,20 20,30...
points	MULTIPOINT	'MULTIPOINT(10 10,20 20)'
MOVING OBJECTS DATA TYPES	MOVING OBJECTS DATA TYPES	
mint	integer	0
mline	lseg	'[(2 , 2),(3 , 3)]'
mlabel	character varying(48)	' '
mreal	numeric	0.0
mpoint	point	POINT(0.0 , 0.0)
mstring	character varying(48)	' '
mbool	boolean	false
mregion	POLYGON	'POLYGON((0 0,0 10,10 10,1...
mpoint	POINT	'POINT(0.0 0.0)'

Abbildung 2.8: SECONDO- to PostgreSQL-Typen

2.4 Right Side

In diesem Kapitel werden die wichtigsten Funktionen der rechten Seite im Detail vorgestellt. Zudem sei erwähnt, dass für das Erzeugen von *Moving Objects* aus *PostgreSQL*-Tupeln ein zweistufiger Prozess vom Anwender durchgeführt werden muss. Der erste Schritt ist das Transferieren der benötigten Datensätze in ein *SECONDO*-DBS. Dieser Schritt ist im nächsten Unterpunkt beschrieben. Danach muss der Anwender das Generieren der *Moving Objects* aus einem Objekt in einer *SECONDO*-DB anstoßen und die nötigen Eingaben tätigen. Dazu muss er die Menüpunkte der *Left Side* verwenden.

2.4.1 Copy to SECONDO

Damit sich nachfolgendes Fenster öffnet, ist, wie bereits zuvor beschrieben, die Anwahl über den Menüpunkt oder das Kontextmenü möglich. Dieser Auswahldialog ist für jede Relation einer Datenbank, die nach *SECONDO* überführt werden soll, aufzurufen.

Der Aufbau dieses Fensters ist relativ simpel gehalten. Im linken Teil der grafischen Benutzeroberfläche (GUI) ist die Relation mit den dazugehörigen Attributen und Datentypen aufgelistet. Zudem steht am oberen Fensterrand der Name der ausgewählten Tabelle. Rechts daneben ist ein Eingabefeld für den neuen Namen. Dieser Name muss den Zeichenkonventionen des *SECONDO*-Datenbanksystems entsprechen. Unter diesem Namen wird das neu zu erzeugende Objekt in einer *SECONDO*-DB abgelegt. Die Liste unter dem Punkt „TO DB“ entspricht den Attributen eines *SECONDO*-Objektes. Hier kann der Anwender auswählen, aus welchen Attributen der Tabelle das *SECONDO*-Objekt zusammengesetzt werden soll. Für die Festlegung dieser Auswahl stehen dem Benutzer die vier Buttons „>“, „>>“, „*remove*“ und „*remove all*“ zur Verfügung.

In dem Eingabefeld, das mit der Überschrift „where condition“ versehen ist, kann der Benutzer eine SQL-*where*-Klausel einfügen. Damit ist die Selektion von Tupeln, die kopiert werden sollen, gewährleistet. Es wird kein Syntaxcheck der *where*-Clausel durchgeführt.

Der wahrscheinlich wichtigste Button auf dieser GUI ist der Button „*Copy...*“. Die zeitliche Dauer eines Kopiervorgangs hängt wie bereits erwähnt von mehreren Faktoren ab.

Bei der Angabe der zu kopierenden Spalten soll der oberste Eintrag einem Datentypen entsprechen, nach dem sortiert werden kann. Gemeint ist ein Datentyp, der mit dem „order by“ einer SQL-Anweisung aus dem *PostgreSQL*-DBS sortiert werden kann. Es gibt nur wenige Datentypen, die davon ausgeschlossen sind. Jedoch wird diese Prüfung nicht automatisch durchgeführt. Eine Umsortierung der ausgewählten Spalten kann im Falle eines auftretenden Fehlers Abhilfe schaffen.

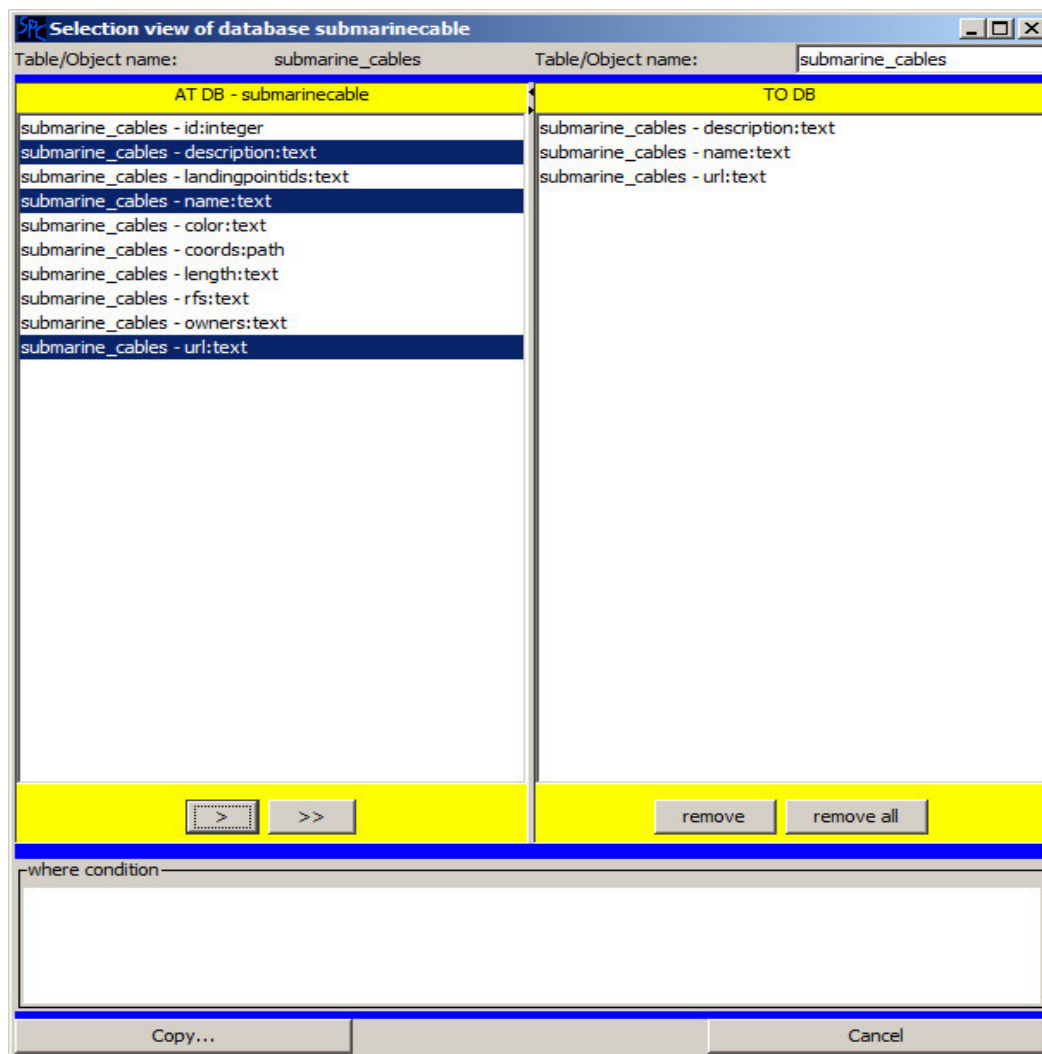


Abbildung 2.9: Auswahl Fenster für die Attribute, die von einer Relation kopiert werden sollen

Nach dem Auslösen des Buttons wird das *SECONDO*-DBS ausgelesen, also das Datenbankschema geladen. Das ist der Grund dafür, weshalb es einen Moment dauern kann, bis das nächste Eingabefenster angezeigt wird.

Zu Beginn des Kopierprozesses fordert SPC den Anwender auf, einen Datenbanknamen festzulegen oder eine bestehende DB auszuwählen, die um ein Objekt erweitert werden soll. Der Button „*Cancel*“ bricht wie erwartet den Vorgang ab. Wählt der Anwender die Option „*new database*“ aus, wird auf dem *SECONDO*-DBS eine Datenbank mit dem entsprechenden Namen angelegt. Diese DB enthält dann ein Objekt mit dem zuvor festgelegten Namen, Attributen (Spalten) und Inhalten (Tupeln). SPC prüft vor dem Anlegen der neuen DB, ob bereits eine DB mit diesem Namen existiert.

Hat der Anwender jedoch die Option „*existing database*“ ausgewählt, so wird vom SPC geprüft, ob ein Objekt mit dem zuvor festgelegten Namen in der angewählten Datenbank existiert.

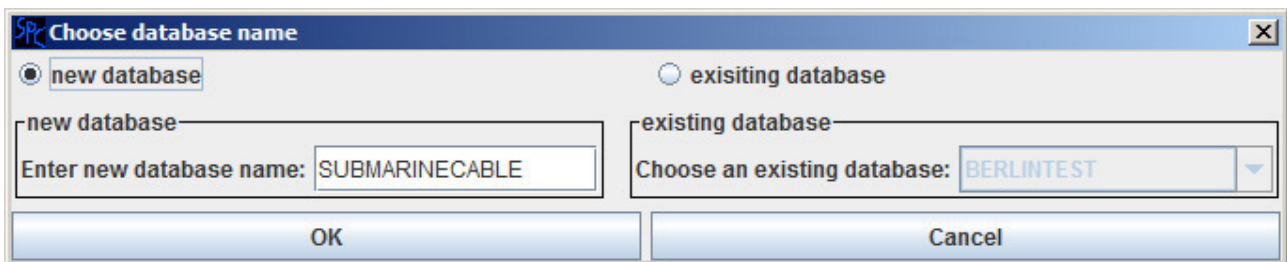


Abbildung 2.10: Auswahldialog um DB-Name festzulegen

Ist dies der Fall, so wird der User darüber informiert. Der Benutzer kann dann, wenn er dies möchte, einen neuen Namen für das neue Objekt festlegen oder das existierende Objekt überschreiben lassen.



Abbildung 2.11: Anzeige, dass der DB-Name bereits verwendet wird

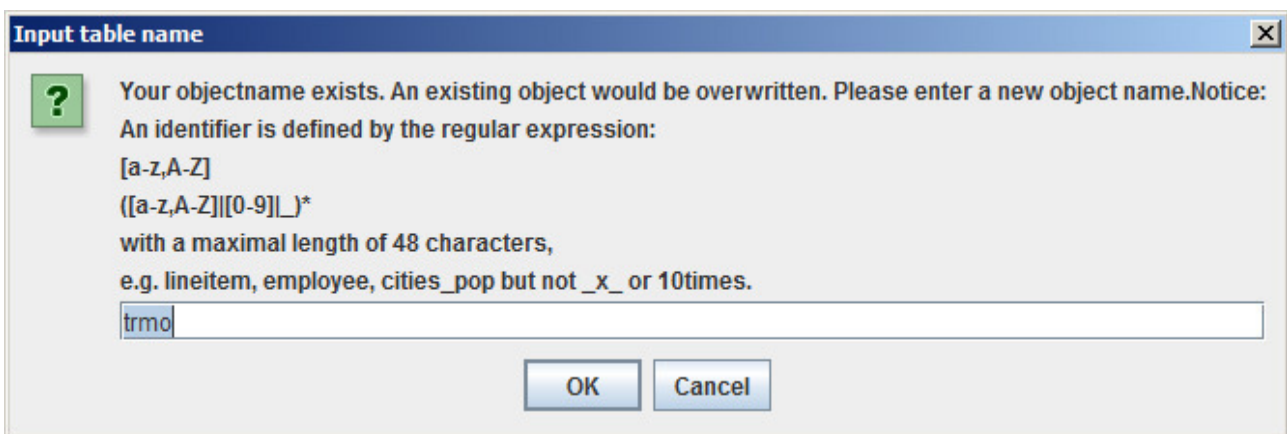


Abbildung 2.12: Eingabefenster für den Objektnamen, falls dieser bereits existiert

Nachdem alle notwendigen Eingaben für einen Kopiervorgang abgefragt und bestätigt wurden, startet der Prozess, der die selektierten Daten von *PostgreSQL* nach *SECONDO* überführt. Da dieser Prozess technisch in mehrere Prozessstufen aufgeteilt ist, kann dieser trotzdem vorzeitig abgebrochen werden. Allerdings kann das nur mit einer gewissen Einschränkung erfolgen. Damit das Programm SPC keine inkonsistente, fehlerhafte oder sogar beschädigte Tabelle oder Datenbank

erzeugt, reagiert der „*Cancel*“ Button der Fortschrittsanzeige nur, wenn lesende Datei- oder Datenbankzugriffe erfolgen. Der Schreibvorgang in einer DB wird damit nicht unterbrochen. Die Beendigung erfolgt in diesem Fall erst nach Abschluss des Schreibvorganges. Im Allgemeinen kann eine kleine Verzögerung bei der Beendigung erfolgen. Grund dafür ist, dass offene Verbindungen sorgfältig geschlossen und getrennt werden müssen.

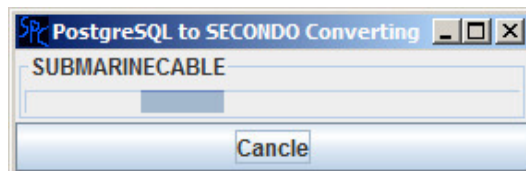


Abbildung 2.13: Fortschrittsanzeige

Sobald der Kopiervorgang erfolgreich abgeschlossen und die linke Seite des Datenbankschemas aktualisiert wurde, bekommt der Anwender einen Dialog angezeigt, der ihm das signalisiert.

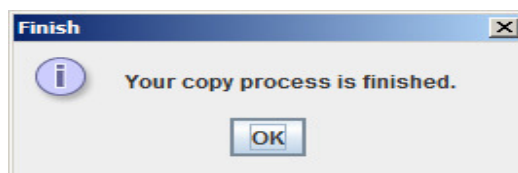


Abbildung 2.14: Anzeige, dass der Kopierprozess abgeschlossen ist

Bei jedem Kopiervorgang können Fehler auftreten. Damit der Anwender nicht ahnungslos bleibt, wird er über auftretende Fehler im Programmablauf informiert. Dies ist z. B. dann der Fall, wenn eine Verbindung zur Datenbank fehlschlägt oder ein Syntaxfehler in der *where*-Klausel ist.

2.5 Left Side

In den nachfolgenden Unterkapiteln dieses Abschnittes wird erläutert, wie Tupel von *SECONDO* nach *PostgreSQL* kopiert werden. Dazu gibt es die Möglichkeit aus *Moving Objects* relationale Tabellen zu erstellen. Ein weiterer wichtiger Punkt ist das komfortable Erstellen von *Moving Objects*, das hier vorgestellt und beschrieben wird.

2.5.1 Copy Objects to PostgreSQL

Dieser Ablauf ist analog zu dem zuvor beschriebenen Ablauf und den Eingaben von dem Kapitel „Copy to SECONDO“. Der Unterschied ist, dass die Daten von *SECONDO* nach *PostgreSQL* überführt werden. Daher werden bei diesem Ablauf nur die Punkte erklärt, die sich vom vorherigen Kapitel unterscheiden.

Zum einem besteht der Unterschied darin, dass das Eingabefeld auf der GUI nicht mehr mit „where condition“ sondern mit „filter condition“ überschrieben ist. Das bedeutet, dass hier für die Selektion der Daten von einem Objekt eine *filter*-Bedingung eingetragen werden kann. Für die Definition einer solchen *filter*-Bedingung muss der Anwender die *SECONDO* eigene Syntax verwenden.

Auf der GUI für das Kopieren von Objekten von *SECONDO* nach *PostgreSQL* ist eine zusätzliche Checkbox oberhalb des „Copy..“ Buttons platziert. Diese Checkbox ist dafür vorgesehen, um dem Anwender die Möglichkeit der Auswahl zu geben, ob für eine gewisse Art von Datentypen im *SECONDO*-System *PostgreSQL*- oder *PostGIS*-Datentypen beim Kopieren der Daten erzeugt werden sollen. Für welche Ausgangsdattentypen (*SECONDO*-Datentypen) dies möglich ist, ist aus dem Kapitel „Left → Supported Types“ zu ermitteln.

Sollte sich der Anwender dafür entscheiden, *PostGIS*-Datentypen erzeugen zu wollen, so muss er noch eine weitere Besonderheit beachten. Damit *PostGIS*-Datentypen erzeugt werden können, muss die *PostGIS*-Extension vorhanden sein. Das heißt der Anwender muss eine DB auswählen, die bereits über die *PostGIS*-Extension verfügt und dort die neue Tabelle hinzufügen. Ist der Anwender dazu gezwungen oder möchte er eine neue DB anlegen, muss diese natürlich auch eine *PostGIS*-Extension enthalten. Dafür benötigt der Anwender ein Datenbanktemplate, das eine *PostGIS*-Extension besitzt und muss dieses beim Erzeugen angeben. Das Template kann eine herkömmliche DB sein, die die entsprechende Extension installiert hat. Sollte der Anwender versuchen *PostGIS*-Datentypen zu erzeugen, ohne dass diese Extension in der entsprechenden DB installiert ist, führt dies zu einem Fehler. Die Spalten und Inhalte können nicht in die entsprechende Tabelle geschrieben werden. Daher ist das zuvor Beschriebene stets zu beachten.

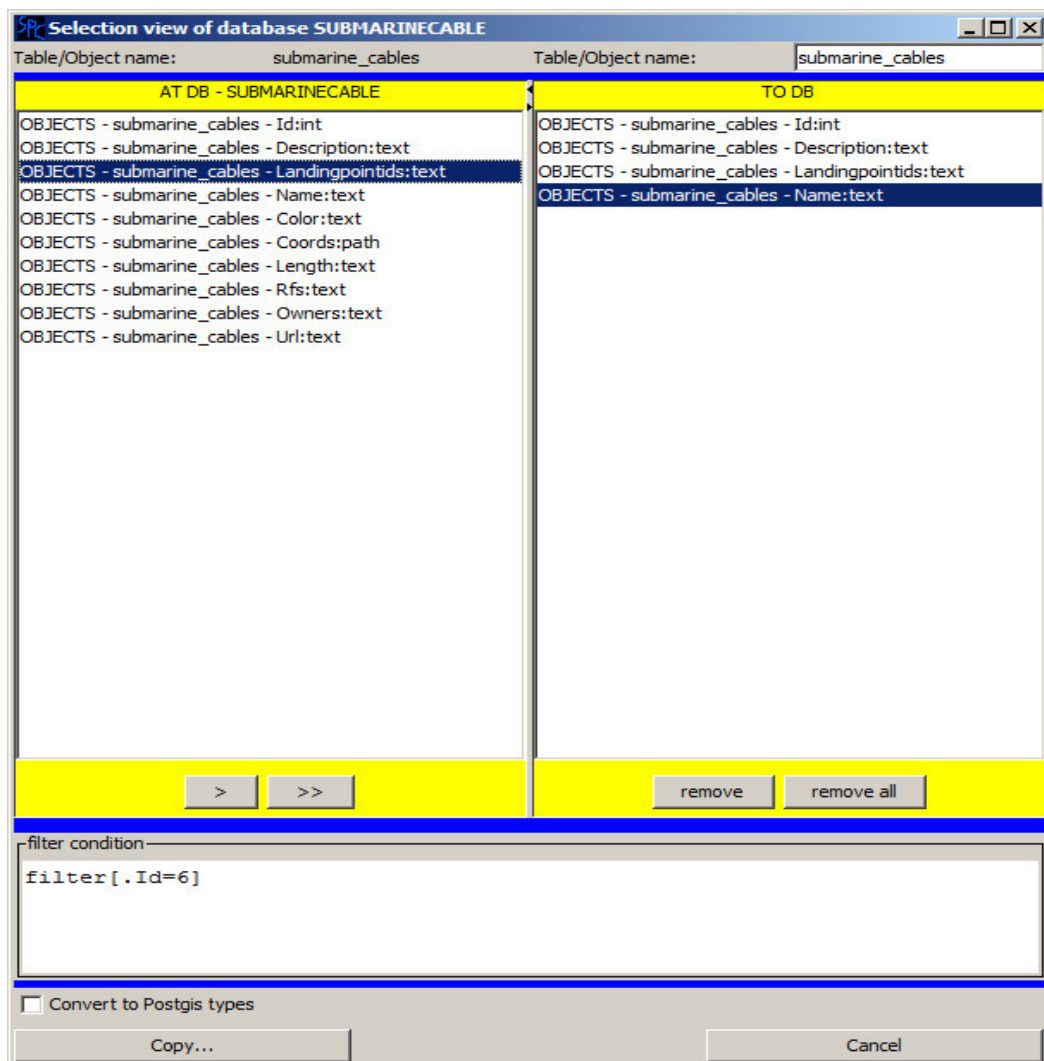


Abbildung 2.15: Auswahl Fenster für die Attribute, die von einem Objekt kopiert werden sollen

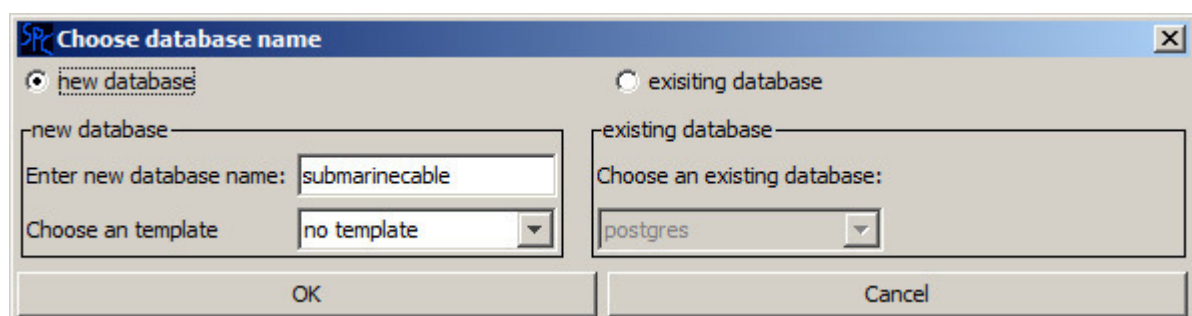


Abbildung 2.16: Auswahldialog um DB-Name festzulegen mit Template-Auswahl

2.5.2 Copy Moving Objects to PostgreSQL

Damit das Kopieren von *Moving Objects* nach *PostgreSQL* funktioniert sei hier kurz genannt, was die Besonderheit eines solchen Datentypen ist. Diese Datentypen geben dem Anwender und Entwickler die Möglichkeit, für einen bestimmten Zeitpunkt einen bestimmten Wert festzulegen. Das ist auch in relationalen Systemen möglich, jedoch kann dies in *SECONDO* in einem einzigen Attribut erfolgen. Zudem ist für die Beschreibung eines Wertes über einen Zeitraum lediglich eine Datenzeile notwendig. Solche Datentypen können z. B. für das Beschreiben von Unwetterregionen oder für die Beschreibung einer Flugroute dienen. Es ist also möglich, jedem Zeitpunkt einen Wert oder einen Wert einem Zeitpunkt zuzuordnen. Diese Datentypen beschreiben also einen zeitlichen Verlauf und die Veränderung eines Wertes in dem beschriebenen Zeitraum.

Damit *Moving Objects* nach *PostgreSQL* überführt werden können, sind einige Eingabefelder in der GUI „Copy MO to PostgreSQL“ auszufüllen.

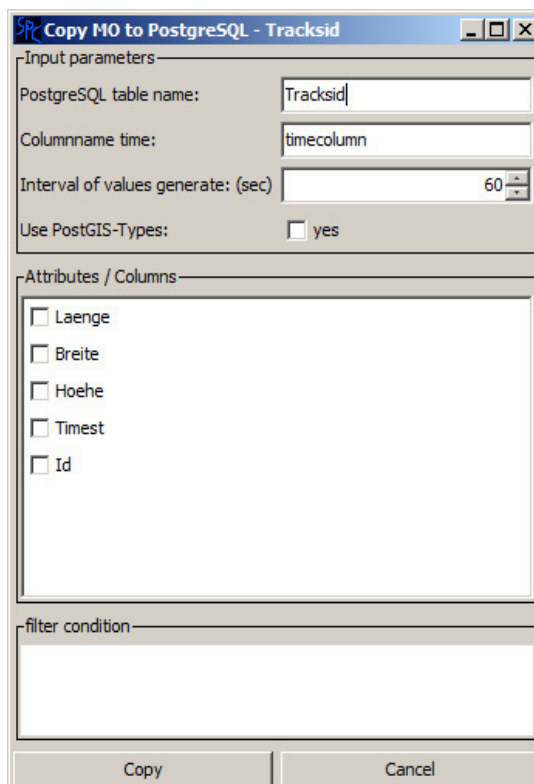


Abbildung 2.17: Eingabefenster für das Kopieren von *Moving Objects* nach *PostgreSQL*

Der Anwender kann wie gewohnt den Namen für die zu erzeugende Relation in einem Eingabefeld bestimmen. Dazu muss er die Konventionen von *PostgreSQL* beachten. Das zweite Eingabefeld stellt die erste Besonderheit dar. Hier muss der Benutzer den Namen des Attributes bestimmen, in dem die Zeitpunkte gespeichert werden. Diese Spalte enthält die Zeitpunkte (Datum und Uhrzeit). Des

Weiteren kann der User bestimmen, in welchem Intervall aus den vorliegenden *Moving Objects* Tupel erzeugt werden. Die Angabe erfolgt in Sekunden und bedeutet bei dem Eintrag von 60 Sekunden, dass über einen noch zu bestimmenden Zeitraum aus den *Moving Objects* alle 60 Sekunden ein Wert ermittelt wird. Diesen Wert speichert SPC dann mit dem Zeitpunkt in der Tabelle ab. Mit der Checkbox „*Use PostGIS-Types*“ kann der Anwender bestimmen, ob er, wenn es möglich ist, aus den vorliegenden Attributen *PostGIS*- oder *PostgreSQL*-Datentypen erstellen möchte.

Welche Attribute in der neuen Tabelle vorhanden sein sollen, kann der Benutzer mit Hilfe der Checkboxes festlegen. Diese Checkboxes, die die Attribute des Objektes repräsentieren, enthalten jeweils einen Tooltip. Der Tooltip ist der Datentyp des Attributes.

Attribute, die nicht einem *MO*-Datentypen angehören, können natürlich auch mit kopiert werden. Die Werte aus diesen Attributen werden dann in den entsprechenden Spalten je Zeitpunkt eingetragen.

Das letzte Eingabefeld, das wieder mit der Überschrift „filter condition“ versehen ist, kann für die Eingabe einer *filter*-Bedingung in der Syntax des *SECONDO*-Systems genutzt werden.

Hat der User alle notwendigen Datenfelder ausgefüllt, kann der Button „*Copy*“ betätigt werden. Dieser liest das Datenbankschema des *PostgreSQL*-Datenbanksystems aus und fordert den Benutzer auf, einen Datenbanknamen festzulegen.

Nach dieser Abfrage muss der Anwender erneut eine Anfrage beantworten. Die nachfolgende GUI zeigt den Zeitraum an, der durch die *Moving Objects* ermittelt wurde. Die Ermittlung des Zeitraumes erfolgt anhand der ausgewählten und selektierten *MO*-Datentypen. Der Anwender hat also mit diesem Auswahldialog die Fähigkeit den Zeitraum einzugrenzen. Für diesen begrenzten Zeitraum wird dann aus dem *SECONDO*-Objekt eine Tabelle in einer *PostgreSQL*-DB generiert.

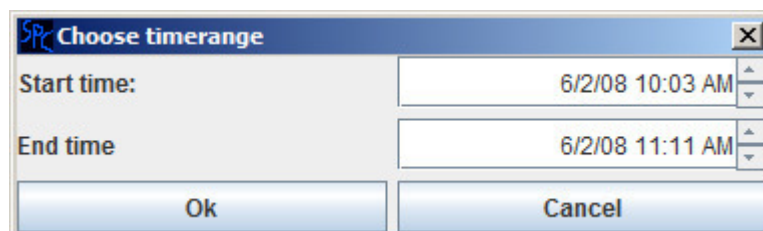


Abbildung 2.18: Zeitauswahl (Monat/Tag/Jahr)

„Es kann vorkommen, dass bei dem Kopierprozess von *Moving Objects* nach *PostgreSQL* ein

Secondo-SMI Error ([Berkeley-DB Error] Cannot allocate memory ->

[bdbEnvironment.cpp:597]) ausgelöst wird. Dann muss der Vorgang erneut gestartet und *filter*-Bedingungen angegeben oder der Zeitraum für die Abfrage verkleinert oder das Intervall vergrößert werden, da der Fehler auftritt, wenn zu viele Abfragen gleicher Art in kurzen Abständen erfolgen. Es handelt sich dabei um einen Fehler des *SECONDO*-Datenbanksystems.“ [Masterarbeit Walther]

2.5.3 Generate Moving Objects

Für das Generieren von *Moving Objects* ist eine GUI zu verwenden, die etwas komplexer ist. Jedoch soll diese nicht abschrecken, denn alle Einstellungen, die vorgenommen werden müssen, sind die gleichen, die ein Nutzer in einem *SECONDO*-Befehl wiederfindet. Nachfolgend werden die Abschnitte der GUI vorgestellt.

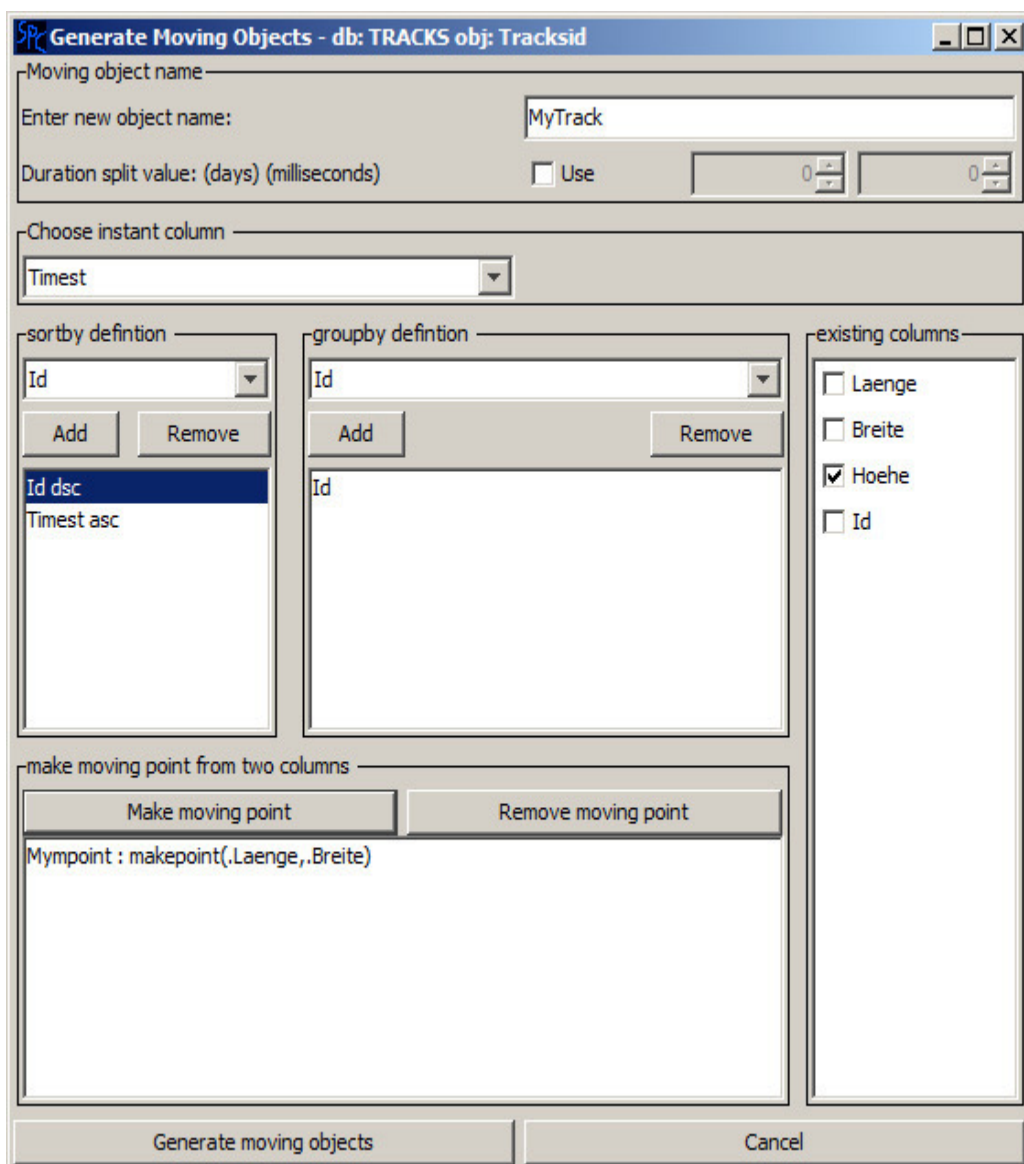


Abbildung 2.19: GUI zum Generieren von *Moving Objects*

Abschnitt „Moving object name“: Der Anwender hat wieder die Möglichkeit, den Namen für das zu generierende Objekt festzulegen. Durch das Anwählen der Checkbox „*Use*“ werden zwei weitere Elemente auf der GUI aktiv. Mit diesen beiden Feldern kann eine Zeitdauer definiert werden. Diese Dauer ist beim Erzeugen wichtig, denn sie legt fest, wie groß der Abstand zwischen zwei aufeinander folgenden Zeitpunkten sein darf. Diese Zeitdauer gibt also an, welche maximale Dauer zwischen zwei Zeitpunkten liegen darf, um zu einem *MO* vereint zu werden. Sollte dieser Wert überschritten werden, so wird an dieser Stelle ein neuer Track (Tupel) erzeugt und gespeichert.

Abschnitt „Choose instant column“: In dieser Combobox werden nur Attribute mit dem Datentypen *instant* aufgeführt. Hier muss der User das Attribut auswählen, das die Zeitpunkte enthält, die zum Erzeugen des *Moving Objects* notwendig sind.

Abschnitt „sortby definition“: In diesem Bereich der GUI kann der User festlegen, welche Attribute für die Sortierung benutzt werden. Dafür muss er die Buttons „*Add*“ und „*Remove*“ benutzen. Ein Doppelklick auf ein selektiertes Element in der Liste verändert das Kürzel „*asc*“ nach „*dsc*“. Das legt also fest, ob aufsteigend oder absteigend sortiert wird.

Abschnitt „groupby definition“: Der Abschnitt der GUI ist für die Gruppierungseinstellung der Attribute zuständig. Damit ein Attribut aus der Combobox in die darunter befindliche Liste eingetragen werden kann, muss der Anwender den Button „*Add*“ verwenden. Zum Löschen eines angewählten Elementes aus der Liste ist der Button „*Remove*“ vorhanden.

Abschnitt „existing columns“: In dieser Liste kann der Anwender festlegen, welche Attribute im neuen Objekt vorhanden sein sollen. Zu beachten ist, dass die in dieser Liste ausgewählten Attribute nicht als Attribut in der Gruppierungseinstellung ausgewählt werden. Es wird aber später auch noch vom SPC geprüft, ob diese Bedingung eingehalten wurde.

Abschnitt „make moving point from two columns“: In diesem Abschnitt gibt es zwei weitere Buttons. Der Button „*Remove moving point*“ löscht einen angewählten Eintrag aus der Liste darunter. Das Betätigen des Buttons „*Make moving point*“ ruft ein weiteres Fenster auf. Mit diesem kann der Anwender ein neues Attribut aus zwei Attributen erstellen. Dafür muss der Anwender einen Namen für das Attribut vergeben und die beiden Attribute festlegen, aus denen das neue Attribut bestehen soll. Bestätigt der Anwender seine Eingabe mit „*OK*“, werden die Einstellungen auf der GUI in die dafür vorgesehene Liste eingetragen. Somit kann aus zwei einzelnen numerischen Attributen (*int*, *real*) temporär ein Attribut vom Typ *point* erzeugt

werden. Aus diesem würde dann beim Generieren ein `mpoint`-Datentyp. Anwendung findet dies z. B. wenn der Anwender ein Ausgangsobjekt hat, das Koordinateninformationen in zwei einzelnen Attributen (`Laenge`, `Breite`) gespeichert hat und diese aber gerne als `mpoint`-Datentyp speichern möchte.

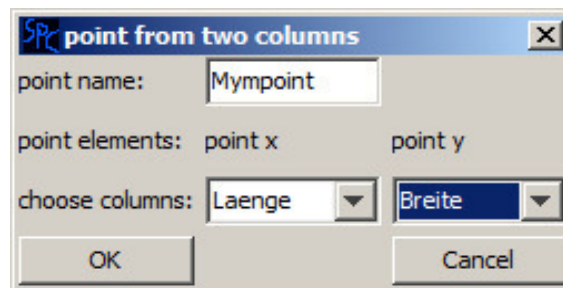


Abbildung 2.20: Zwei numerische Attribute zusammenführen

Sobald der Anwender alle Informationen auf der GUI eingetragen hat, wird durch das Betätigen des Buttons „*Generate moving objects*“ der Prozess auf dem *SECONDO*-DBS gestartet, der für das Erzeugen des neuen Objektes zuständig ist. Zuvor werden noch die vom Nutzer getätigten Eingaben überprüft. Sollten hierbei Unstimmigkeiten festgestellt werden, so wird der User mit einer Meldung darüber informiert. SPC überprüft natürlich auch, ob das Objekt bereits existiert und informiert den Anwender darüber mit der Möglichkeit, einen anderen Namen einzugeben.

Ist der Prozess für das Erzeugen einmal angestoßen, so wird der Prozess komplett vom *SECONDO*-DBS durchgeführt und kann auch nicht abgebrochen werden. Sollte alles fehlerfrei funktioniert haben, wird die folgende Meldung ausgegeben.



Abbildung 2.21: Moving Objects are generated

3 Hinweise zu den Datenbanksystemen

In diesem Kapitel soll nur kurz erwähnt werden, mit welchen Datenbanksystemen SPC getestet wurde. Zudem gibt es ein Kapitel, das einen allgemeinen Hinweis zu der Verwendung von SPC und den beiden Datenbanksystemen gibt.

3.1 Besondere Hinweise

Für die reibungsfreie Nutzung der Anwendung müssen dem Datenbankbenutzernamen des Anwenders Lese- und Schreibrechte zugewiesen sein. Grund dafür ist, dass SPC nicht nur Lese-, sondern auch Schreiboperationen durchführt, wenn diese vom Nutzer gewünscht sind. Schreiboperationen treten z. B. beim Kopieren einer DB, Relation oder eines Objektes auf.

Es ist zu empfehlen, ein Backup der DB anzulegen, bevor Sie das erste Mal mit dem SPC arbeiten.

Außerdem benötigt der SPC-Anwender die Berechtigung im *PostgreSQL*-DBS Systemtabellen abfragen zu dürfen.

Produkt	Version
SECONDO	3.3.0
PostgreSQL	8.4, 9.0, 9.1 und 9.2
PostGIS	1.5 und 2.0

Tabelle 3.1: Produkte und Versionen

4 Rechtlicher Hinweis

Alle in dieser Dokumentation beschriebenen Abläufe und Darstellungen wurden nach bestem Wissen und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Daher sind vorliegende Informationen mit keiner Verpflichtung oder irgendeiner Garantie verbunden. Ich übernehme infolgedessen keine juristische Verantwortung und werde keine daraus folgende Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen - oder Teilen davon - entsteht, auch nicht für die Verletzung von Patentrechten, die daraus resultieren können.

Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. werden ohne Gewährleistung der freien Verwendbarkeit benutzt und können auch ohne besondere Kennzeichnung eingetragene Marken oder Warenzeichen sein und als solche den gesetzlichen Bestimmungen unterliegen.

© 2013

Daniel Walther

Brantstraße 2

80687 München