Eberhard Karls
Universität
Tübingen

DB2 9 for z/OS Stored Procedures

Diplomarbeit

Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften
Universität Tübingen

von
Kolja Treutlein

Betreuer:
Prof. Dr.-Ing. Wilhelm G. Spruth

1. Juli 2009

Zusammenfassung

Im Rahmen der Diplomarbeit wurde ein neues Tutorial für den studentischen Praktikumsbetrieb erarbeitet. Das Thema des Tutorials ist die Erstellung und Verwendung von Stored Procedures unter Verwendung des IBM Data Studio Developers auf den zSeries Großrechnern der Universität Tübingen und Leipzig. Grundlage für diese Arbeit war das IBM Redbook DB2 9 for z/OS Stored Procedures Through the CALL and Beyond [IBM1].

Themen der Übung sind neben einer Installationsanleitung und Einführung in die Funktionsweise vom IBM Data Studio Developer eine Anleitung zur Erstellung und Ausführung der in der Version 9 von DB2 neu eingeführten Native SQL Stored Procedures und von Java Stored Procedures sowie deren Einbindung in eine Java-Beispielanwendung. Konkreter Anwendungsfall ist eine Tabelle mit Informationen zu den Mannschaften in der Fußball-Bundesliga während der Saison 2007/2008. Mittels Stored Procedures werden verschiedene Funktionalitäten wie die Ausgabe der Platzierungstabelle, das Eintragen von Spielergebnissen oder das Berechnen von feststehenden Absteigern implementiert. Diese Funktionalitäten wurden in eine Java-Anwendung eingebunden und können über eine GUI bedient werden.

Alle nötigen Dateien, Programme und das Tutorial sind auf einer DVD vorhanden, die sich im Anhang dieser Arbeit befindet und die den Studierenden bei Praktikumsbeginn zur Verfügung gestellt wird.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ort, Datum	Unterschrift

Danksagungen

Mein herzlicher Dank gilt allen, die mir die Realisierung dieser Diplomarbeit ermöglicht haben. An erster Stelle möchte ich mich bei Herrn Prof. Dr.-Ing. Wilhelm G. Spruth bedanken, der meine Arbeit betreute und mich in vielfältiger Weise während der gesamten Umsetzung unterstützte. Weiterer Dank gilt Herrn Christian Daser (Firma IBM) für seine fachmännische Hilfe und seinen unermüdlichen Einsatz. Und last but not least: Ein großes „Danke“ an María Lucía Spangenberg und Stefan Steinhilber, die mir während der gesamten Umsetzungszeit motivierend zur Seite standen.

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund

An der Universität Tübingen und Leipzig findet im Rahmen des Informatikstudiums jedes Jahr ein Praktikum statt, das die Vorlesung Client/Server-Systeme begleitet und in dem praktische Grundlagen über Großrechnersysteme und deren Betriebssystem z/OS vermittelt werden. Zusätzlich sollen der auf diesen Komponenten aufbauende Transaktionsserver Customer Information Control System (CICS) und das Database Management System DB2 kennen gelernt werden.

Mit der Version 9 von DB2 ist es der Firma IBM gelungen, die Erstellung und Verwaltung von SQL Stored Procedures grundlegend zu vereinfachen. Aus diesem Grund erfreuen sich Stored Procedures neuer Beliebtheit und werden in Zukunft vermutlich eine größere Rolle bei der Realisierung von Businesslogik spielen.

Um diesen Trend im Praktikum nachzuvollziehen, war es die Aufgabe, im Rahmen dieser Diplomarbeit ein Tutorial zu erstellen, das sich gezielt mit dem Thema Stored Procedures auseinandersetzt.

Voraussetzungen für die Durchführung dieses Tutorials sind ein Account auf einem der Großrechner der Universität Tübingen (LPAR Hobbit) oder Leipzig (LPAR Binks), sowie eine eingerichtete Datenbank (Tutorial 4) und Kenntnisse über JDBC (Tutorial 8). Ziel dieser Übung ist es, eine Datenbanktabelle der Mannschaften der Bundesliga in der Saison 2007/2008 zu erstellen und unter Verwendung von Stored Procedures verschiedene Operationen darauf durchzuführen. Außerdem wird gezeigt, wie Stored Procedures in eine Java-Anwendung eingebunden werden können.

Die aus der Arbeit hervorgegangene Übung wird Teil des Client/Server-Praktikums an der Universität Tübingen und Leipzig sein.

1.2 Aufbau der Arbeit

Im nächsten Kapitel dieser Diplomarbeit soll eine kurze Erläuterung grundlegender Begriffe im Bereich Mainframe gegeben werden, die für das Verständnis dieser Diplomarbeit wichtig sind. Im darauffolgenden Kapitel erfolgt eine Einführung in den allgemeinen Aufbau von Stored Procedures. In Kapitel 4, 5 und 6 werden dann die verschiedenen Typen von Stored Procedures dargestellt. Kapitel 4 und 5 beinhalten das in Kapitel 1.1 angesprochene Tutorial. Alle hierfür benötigten Programme und Daten sind auf einer DVD hinterlegt, die der Diplomarbeit beigelegt ist. Der Inhalt der DVD wird im Anhang vorgestellt.

2 Grundlagen

In diesem Kapitel sollen einige Begriffe aus dem Mainframe-Umfeld erklärt werden, die für das Verständnis der Diplomarbeit von großer Bedeutung sind.

2.1 Die Großrechnerarchitektur System z

System z ist die aktuelle Generation an Großrechnern der International Business Machines Corporation (IBM). Zum Einsatz kommen sie vor allem dort, wo ein hohes Maß an Zuverlässigkeit und eine hohe Ein-/Ausgabelast bewältigt werden muss. Möglich wird dies durch eine vollständig redundante Auslegung der Hardware. Für weitere Informationen zum Thema System z sei auf [BEY] verwiesen.

2.2 Das Betriebssystem z/OS

In den 60er Jahren entwickelte die Firma IBM ein Betriebssystem mit dem Namen MVS (Multiple Virtual Storage). Eingesetzt auf Mainframes hatte es geringe Hardwareanforderungen und überzeugte durch hohe Stabilität im Langzeitbetrieb. Primär wurde es für die Bereitstellung von Ressourcen für Terminalverbindungen eingesetzt. Nach Weiterentwicklung änderte die IBM den Produktnamen Mitte der 90er Jahre von MVS in OS/390. Mit Erscheinen der z/Architektur wurde das Produkt erneut umbenannt und heißt nun z/OS. Das Zusammenspiel von z/OS mit der System z-Hardware ist optimiert für sehr hohes I/O-Aufkommen bei sehr vielen Nutzern. Gleichzeitig bietet es größtmögliche Sicherheit und eignet sich für Transaktionen. Im Banken- und Versicherungsumfeld, aber auch im öffentlichen Sektor sind der Großrechner und sein Betriebssystem nicht mehr wegzudenken. In den nächsten beiden Abschnitten werden zwei Komponenten von z/OS kurz vorgestellt: Der Workloadmanager und das Datenbanksystem DB2. Für weitere Informationen zu z/OS sei auf [HER] verwiesen.

2.3 Der Workloadmanager, eine Komponente von z/OS

Der Workloadmanager (WLM) ist Teil des z/OS Betriebssystems und ermöglicht eine dynamische Verteilung der Ressourcen zwischen sogenannten Dienstklassen (Service Classes). Hier können verschiedene Prioritäten und Zielvorgaben definiert werden. Der WLM versucht automatisch die Lastverteilung zu optimieren und berechnet die Verteilung kontinuierlich neu, indem er Daten über vergangene Ereignisse sammelt und auswertet. Für detaillierte Informationen zum Workloadmanager sei auf [HER] verwiesen.

2.4 Das Datenbanksystem DB2

Das Database Management System DB2 ist ein relationales Datenbanksystem der Firma IBM. Der erste Prototyp dieses Systems entstand in den 70er Jahren unter dem Namen System R. Es wurde kontinuierlich weiterentwickelt und war 1983 zum ersten Mal unter dem Namen DB2 (Database 2) für das Betriebssystem MVS verfügbar. Inzwischen gibt es DB2 für alle gängigen Server-Betriebssystemplattformen. Seit 17.März 2007 ist die Version 9 von DB2 for z/OS verfügbar.

2.5 Die Datenbanksprache SQL

SQL ist eine Datenbanksprache, die es erlaubt, Daten in relationalen Datenbanken zu definieren, abzufragen und zu manipulieren. SQL steht für Structured Query Language und ist der Nachfolger von SEQUEL (Structured English Query Language), das von der IBM ca. 1975 entwickelt wurde. 1986

wurde es erstmals unter dem Namen SQL1 von der ANSI und ein Jahr später auch von der ISO als Standard festgelegt. Durch Weiterentwicklung entstand 1992 der Standard SQL2 und 1999 SQL:1999. Außerdem entstand ein weiterer Standard mit Namen SQL/PSM (Persistend Stored Modules), der SQL um prozedurale Programmierelemente wie z.B. Schleifen und Cursor erweitert. Die IBM implementierte diesen Standard in DB2 unter dem Namen SQL PL (SQL Procedural Language). Weitere SQL-Standards folgten in den Jahren 2003, 2006 und zuletzt 2008. Alle aktuellen Datenbanksysteme halten sich im Wesentlichen an diese Standards.

Nachdem die wichtigsten Begriffe definiert sind, soll im nächsten Kapitel eine Einführung in Stored Procedures gegeben werden.

3 Einführung in Stored Procedures

Stored Procedures sind von einem Benutzer geschriebene Programme, die auf einem DB2-Server gespeichert sind, SQL Befehle enthalten und mit einem SQL CALL-Befehl aufgerufen werden können. Es gibt drei Typen von Stored Procedures: Native SQL-Stored Procedures, External SQL Stored Procedures und External high-level language Stored Procedures. Während der Code von Letzteren in einer höheren vom Server unterstützten Programmiersprache geschrieben wird, besteht der Code der ersten beiden Typen aus SQL-Code. Stored Procedures sind DB2-Objekte und müssen deshalb in den DB2-Katalogtabellen angelegt werden. Deshalb besteht das Erstellen einer Stored Procedure aus zwei Teilen: zum einen aus einem in SQL geschrieben Teil, der dem DB2-System Eigenschaften der Stored Procedure bekannt macht (Data Definition Language-Teil oder auch nur DDL) und zum anderen aus einem Teil, der die Logik der Stored Procedure implementiert und sprachspezifisch aufgebaut sein kann.

Um die Vorteile von Stored Procedures aufzuzeigen, betrachten wir zunächst einen Client, der auf ein entferntes Datenbanksystem zugreift und einen Eintrag verändern sowie einen neuen Eintrag hinzufügen soll (s. Abbildung 1). Für die Änderung wird der Client zunächst eine SELECT-Anfrage stellen, um die Werte des vorhandenen Eintrags auszulesen. Anschließend werden die Werte clientseitig verarbeitet und die neuen Werte mittels eines UPDATE-Befehls ins entfernte Datenbanksystem zurückgeschrieben. Zusammen mit dem INSERT-Befehl für das Einfügen eines neuen Tupels werden somit für die gesamte Aktion drei Anfragen an das Datenbanksystem benötigt. Da Client und Server evtl. durch ein Netzwerk getrennt sind, vergeht verhältnismäßig viel Zeit, bis die Operation abgeschlossen ist.

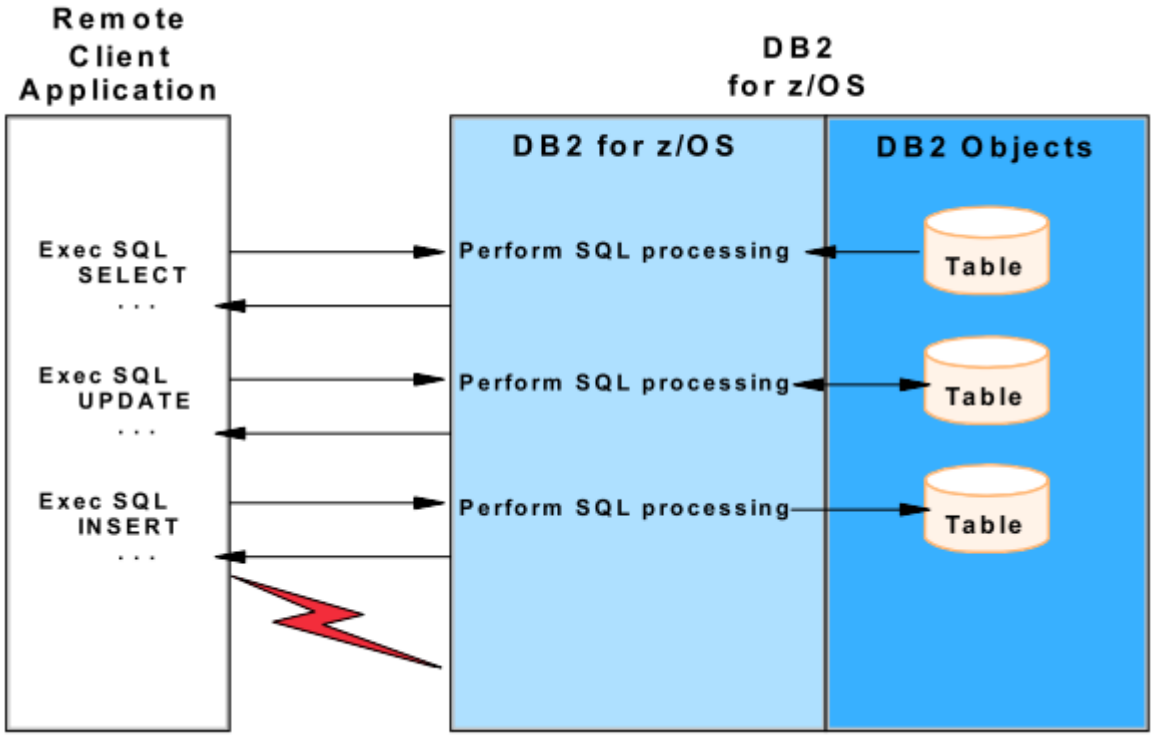


Abbildung 1: Ablauf der Kommunikation ohne Stored Procedures (Quelle: [IBM1])

Ein weiteres Problem ergibt sich, wenn es sich bei der Operation um die Durchführung einer Transaktion handelt. Es muss sichergestellt werden, dass das gelesene Tupel während der Verarbeitung auf dem Client nicht durch einen anderen Client verändert wird und dadurch eine

Änderung verloren geht bzw. sich das System in einem inkonsistenten Zustand befindet. Zudem muss der Administrator sicherstellen, dass der entfernte Client bei der Verarbeitung der Daten gewünschte Businessregeln einhält und nur gültige Werte in die Tabelle zurückschreibt.

Diese Probleme werden durch Stored Procedures gelöst. Durch die Speicherung eines Programms auf dem Datenbankserver genügt eine einzige Anfrage des Clients (nämlich die, das Programm auszuführen), um eine Menge von Operationen durchzuführen (s. Abbildung 2).

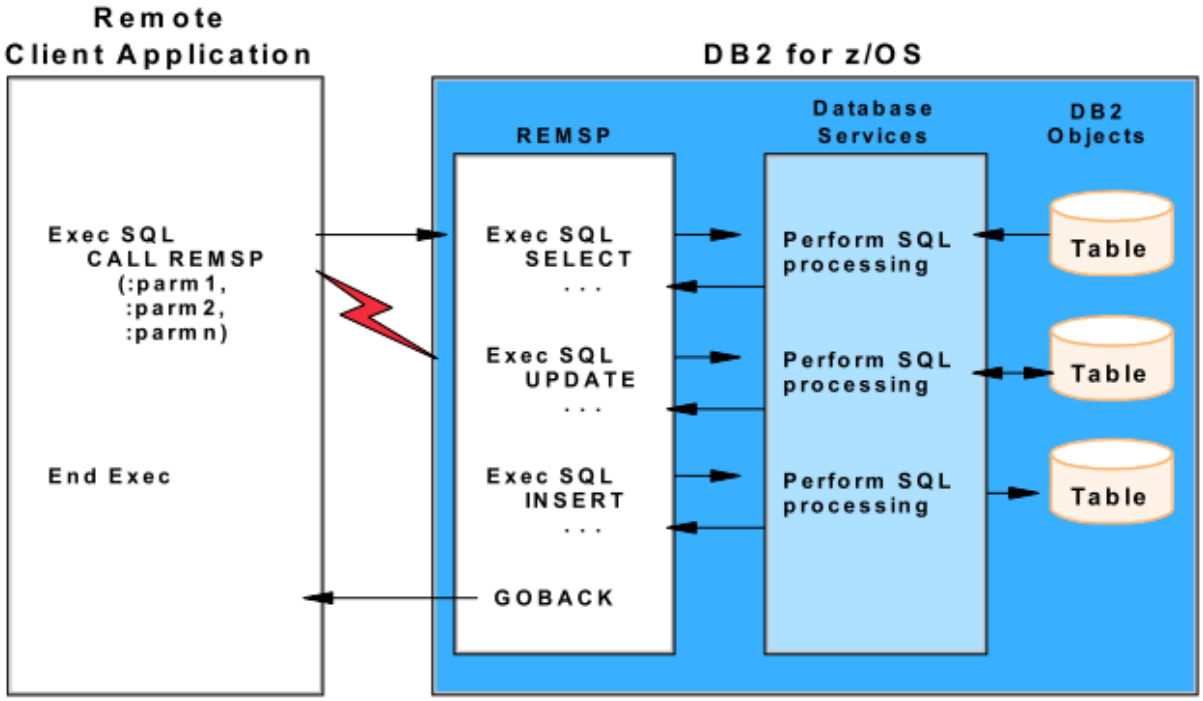


Abbildung 2: Ablauf der Kommunikation mit einer Stored Procedure mit Namen REMSP (Quelle: [IBM1])

Dadurch wird der Netzwerkverkehr in der Regel erheblich reduziert. Hierfür muss Businesslogik auf Serverseite verschoben werden, kann aber dazu verwendet werden, vorgegebene Businessregeln zu erzwingen. Dadurch, dass die ganze Operation auf dem Datenbankserver abläuft, lassen sich auch leicht die ACID-Bedingungen für Transaktionen aufrechterhalten. Stored Procedure sorgen außerdem für eine gute Modularität: Programmierer von Clientanwendungen müssen sich nicht mehr um einzelne Datenbankabfragen kümmern sondern nur noch um den Aufruf einer Stored Procedure. Evtl. kann sogar die Struktur der Datenbanktabellen verborgen bleiben. Die im DB2-System gespeicherte Businesslogik in Form einer Stored Procedure kann außerdem leicht ausgetauscht werden, ohne dass hierfür die Clientanwendungen neu geschrieben werden müssen.

Werden keine Stored Procedures verwendet, so benötigt ein Client, der Änderungen auf einer Datenbanktabelle machen soll, zwingend die Rechte, um Änderungen auf der Tabelle ausführen zu dürfen. Dies ist oft nicht erwünscht, da man dann dafür sorgen muss, dass nicht beliebige Änderungen an der Tabelle vorgenommen werden und damit die Konsistenz der Daten verloren geht. Im Gegensatz dazu benötigt der Benutzer bei der Verwendung von statischem SQL in Stored Procedures nicht mehr die vollen Zugriffsrechte auf der zu ändernden Datenbanktabelle. Es genügt, wenn er das Recht besitzt, das auf dem Server gespeicherte Programm aufzurufen.

3.1 Der Data Definition Language-Teil

Der DDL-Teil besteht bei Stored Procedures aus einem CREATE PROCEDURE- oder ALTER PROCEDURE-Befehl. Er dient dazu, dem DB2-System Informationen zur Stored Procedure wie der Speicherort, die Sprache des Programmcodes, Anzahl und Typ der Parameter und sonstige Umgebungs- und Ausführungsoptionen zu übergeben. Die allgemeine Syntax des CREATE PROCEDURE-Befehls ist in Abbildung 3 zu sehen.

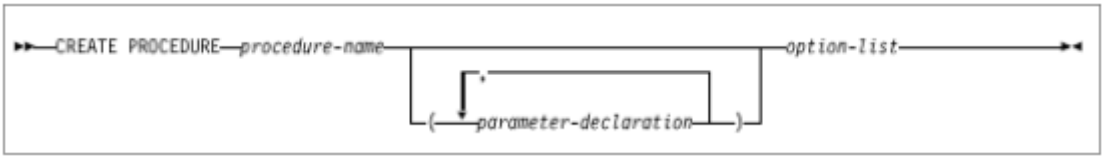


Abbildung 3: Syntax des CREATE PROCEDURE-Befehls (Quelle: [IBM1])

Während bei External high-level language Stored Procedures die Logik in einer externen Programmiersprache geschrieben ist, bestehen Native SQL Stored Procedures und External SQL Stored Procedures nur aus SQL-Code. Im Gegensatz zu Native SQL Stored Procedures, bei denen der SQL-Code in einer internen Repräsentation im Datenbanksystem gespeichert wird, wird der SQL-Code von External SQL Stored Procedures in ein C-Programm übersetzt und dieses zur Laufzeit ausgeführt. External high-level language Stored Procedures und External SQL Stored Procedures müssen in einem separaten Adressraum ausgeführt werden. Je nach Typ der Stored Procedure unterscheiden sich deshalb die Werte für option-list. Eine Übersicht hierfür befindet sich in Anlage A.

Durch das Ausführen der DDL wird dem DB2-System die Stored Procedure bekannt gemacht. Dazu werden Einträge in Katalogtabellen angefertigt. Stored Procedures tauchen in 2 Katalogtabellen auf: SYSIBM.SYSROUTINES und SYSIBM.SYSPARAMS.

In SYSIBM.SYSROUTINES wird pro Stored Procedure ein Eintrag festgehalten. Er enthält den Namen der WLM-Umgebung, verwendete Programmiersprache, Zahl der Parameter, Parametertyp, ob Ergebnismengen zurückgegeben werden können und einige weitere Informationen.

In der SYSIBM.SYSPARAMS werden Informationen zu den Parametern gespeichert wie Name, Datentyp und um was für einen Typ von Parameter es sich handelt (Eingabeparameter, Ausgabeparameter oder sowohl Eingabe- als auch Ausgabeparameter). Für jeden Parameter gibt es einen Eintrag.

Anhand dieser Tabellen wird bei einem Aufruf einer Stored Procedure das entsprechende Programm geladen und ausgeführt.

3.2 Rechte im Bezug auf Stored Procedures

3.2.1 DB2-Rechte, für das Erstellen einer Stored Procedure

Der Name einer Stored Procedure besteht aus drei Teilen: Dem Namen des Datenbanksystems, dem Schemaname und dem Bezeichner der Stored Procedure. Ein Schema ist dabei eine Sammlung von Datenbankobjekten wie Trigger, Userdefined Functions oder Stored Procedures. Datenbanksystem und Schemaname müssen allerdings nicht explizit mit angegeben werden sondern können auch aus

dem Kontext geschlossen werden. Bei keiner Angabe wird das Datenbanksystem verwendet, zu dem die Verbindung besteht und das Schema der aktuell verwendeten Benutzerkennung.

Um Stored Procedures in einem Schema erstellen und ausführen zu können, wird das CREATEIN-Recht auf dieses Datenbankschema benötigt. Hat man dieses nicht, ist die Rückgabe des CREATE-PROCEDURE-Befehls ein SQL-Fehlercode (SQLCode: -552, SQLSTATE 42502). Gelöst werden kann das Problem mit dem Befehl: GRANT CREATEIN ON SCHEMA <schema> TO <user>.

Beinhaltet die Stored Procedure statische SQL-Befehle, wird zusätzlich das BINDADD-Recht benötigt, um das beim Einrichten entstehende Package im DB2-System zu speichern. Besitzt man dieses Recht nicht, gibt der CREATE-PROCEDURE-Befehl folgenden Fehlercode zurück: SQLCODE -567, SQLSTATE 42501. Gelöst werden kann dieses Problem mit dem Befehl GRANT BINDADD TO <user>.

Auf der LPAR Hobbit in Tübingen wurden alle PRAK-Accounts auf die Stufe PACKADM (vgl. Abbildung 4) angehoben. Diese Stufe beinhaltet das CREATEIN- und BINDADD-Recht und ermöglicht es den Benutzern, in ihren Schemas Stored Procedures zu erstellen. Auf der LPAR Binks in Leipzig besitzen Benutzer eines PRAK-Accounts CREATEIN und BINDADD-Rechte auf ihren Schemas. Die Packages der Java Stored Procedures im Tutorial werden in die Collection NULLID gebunden. Auch hierfür wurden entsprechende Rechte verteilt. Das Löschen der Packages eines Benutzers in dieser Collection beim Löschen seines Accounts wird durch eine Stored Procedure durchgeführt.

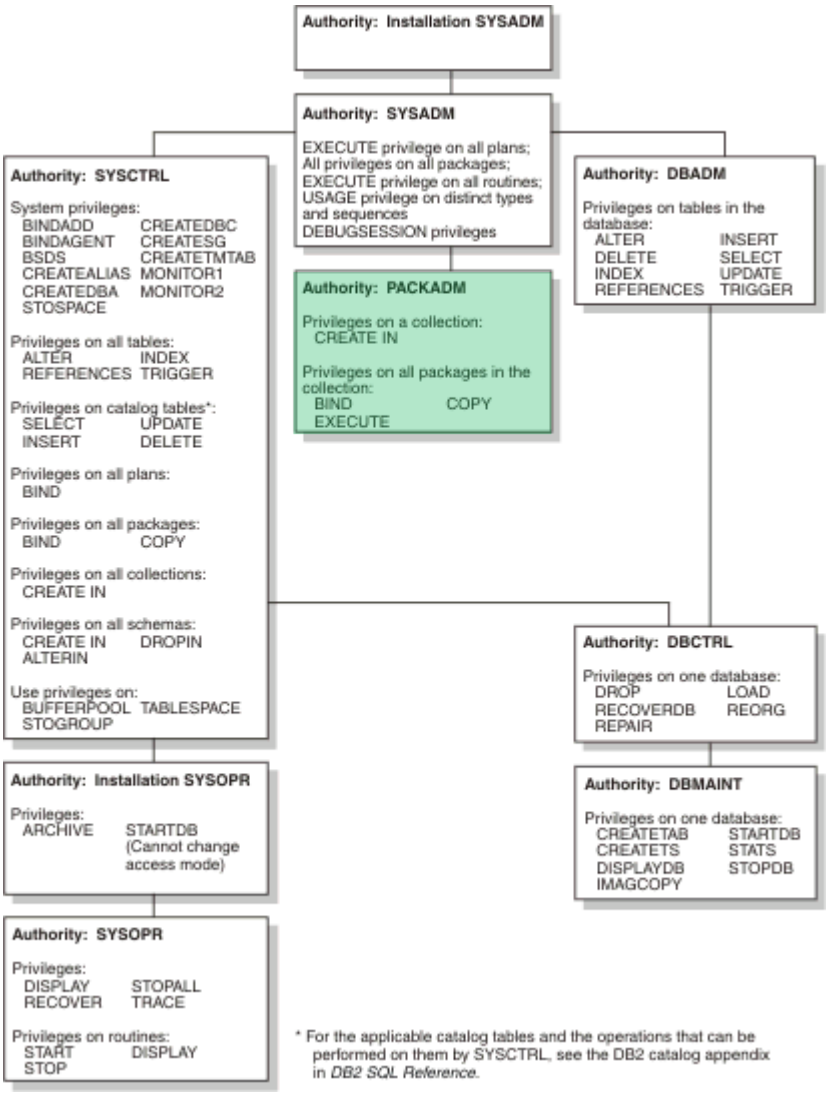


Abbildung 4: Rechte im DB2-System

3.2.2 Rechte für das Ausführen einer Stored Procedure

Eine Stored Procedure wird mit einem CALL-Befehl ausgeführt. Je nachdem, ob die Stored Procedure statische oder dynamische SQL-Befehle enthält, werden unterschiedliche Rechte für die Ausführung benötigt. Statische SQL-Befehle zeichnen sich dadurch aus, dass sie bis auf die Ausprägung von Prädikaten zur Entwicklungszeit feststehen. Dadurch können die SQL-Befehle vom DB2-System optimiert und ein Ausführungsplan in einem Package im Datenbanksystem gespeichert werden. Beim Aufruf der Stored Procedure genügt es dann, das entsprechende Package zu laden und den Ausführungsplan für die Ausführung der SQL-Befehle zu verwenden. Aus diesem Grund benötigt man für die Ausführung der Stored Procedure lediglich eines der drei folgenden Rechte (vgl. Abbildung 4):

- EXECUTE-Recht auf der Stored Procedure
- Besitzer der Stored Procedure
- SYSADM-Rechte

Die Ausführung der Stored Procedure läuft mit den Rechten des Erstellers. Für die Ausführung muss ein Benutzer also nicht zwingend sämtliche Rechte für die Ausführung der einzelnen SQL-Befehle innerhalb der Stored Procedure besitzen. Um eine Rechteüberprüfung bei jedem Aufruf zu erzwingen, kann man in der DDL die DYNAMICRULES-Option verwenden.

Bei dynamischen SQL-Befehlen steht die Logik des SQL-Befehls erst zur Laufzeit fest. Aus diesem Grund kann zur Entwurfszeit kein Ausführungsplan erstellt und im Datenbanksystem gespeichert werden. Somit muss der Ausführende neben den oben genannten Rechten die Rechte besitzen, um jeden dynamischen SQL-Befehl in der Stored Procedure auszuführen. Beinhaltet eine Stored Procedure beispielsweise den dynamischen Befehl SELECT * FROM <eingabeparameter> und möchte ein Benutzer die Stored Procedure mit TABELLE1 als Wert für den Eingabeparamenter ausführen, so benötigt er neben den oben genannten Rechten auch das Recht, auf TABELLE1 ein SELECT ausführen zu dürfen.

4 Native SQL Stored Procedures

4.1 Allgemeines

Wie in der Einleitung erwähnt, lassen sich Stored Procedures in drei Typen aufteilen: Native SQL Stored Procedures, External SQL Stored Procedures und External high-level language Stored Procedures. In diesem Kapitel sollen Native SQL Stored Procedures genauer betrachtet werden.

Bei Native SQL Stored Procedures handelt es sich um Stored Procedures, die in der SQL Procedural Language (SQL PL) geschrieben sind. SQL PL ist eine Erweiterung von SQL, die SQL um prozedurale Elemente erweitert und ähnliche Funktionalität bietet, wie andere Datenbanksprachen (z.B. PL/SQL von Oracle oder T/SQL von Sybase). Bei Native SQL Stored Procedures wird der auszuführende Code in einer internen Repräsentation im Datenbanksystem gespeichert und kann dann im Gegensatz zu External SQL Stored Procedures (Kapitel 6) und External high-level language Stored Procedures (Kapitel 5) direkt in einem der Adressräume von DB2 (DBM1) ausgeführt werden. Diese Art von Stored Procedures wurde neu in DB2 V9 eingeführt. Da DDL und Code der Stored Procedure im CREATE PROCEDURE-Befehl vereint sind, genügt es bei der Erstellung, diesen Befehl auf dem Datenbanksystem auszuführen. Folgendes Tutorial zeigt ihre Erstellung unter Verwendung des IBM Data Studio Developers. Als Beispiel aus der Praxis wird die Fußball-Bundesliga gewählt und ausgehend von einer Tabelle mit den Mannschaften einige Operationen durch Stored Procedures realisiert.

4.2 Erstellung von Native SQL Stored Procedures (Tutorial)

Im Folgenden sollen Stored Procedures unter Verwendung des IBM Data Studio Developers entwickelt werden. Der IBM Data Studio Developer bietet eine integrierte Datenbankentwicklungsumgebung für SQL, XQuery und Java. Er setzt auf die Open Source Platform Eclipse auf. Die Version 2.1 dieses Produkts können Sie von Ihrem Tutor erhalten oder unter http://www-01.ibm.com/software/data/studio/ herunterladen. Als konkretes Anwendungsbeispiel dient eine Datenbanktabelle mit Fakten zu den Mannschaften der Fußball-Bundesliga in der Saison 2007/2008. Nach der Durchführung dieses Tutorials werden Sie in der Lage sein, vier von Ihnen erstellte Stored Procedures, die eine Liste der Teams, eine sortierte Bundesligatabelle und Prognosen für den Ausgang beliebiger Spielpaarungen ausgeben, über eine Java-Anwendung auszuführen. Desweiteren wird es über eine Stored Procedure möglich sein, die Datenbanktabelle durch das Eintragen von Spielergebnissen zu verändern. Abbildung 5 zeigt dabei schematisch den Aufbau. DDF steht hierbei für Distributed Data Facility und ist ein DB2-Adressraum, der Anfragen von entfernten Clients entgegen nimmt.

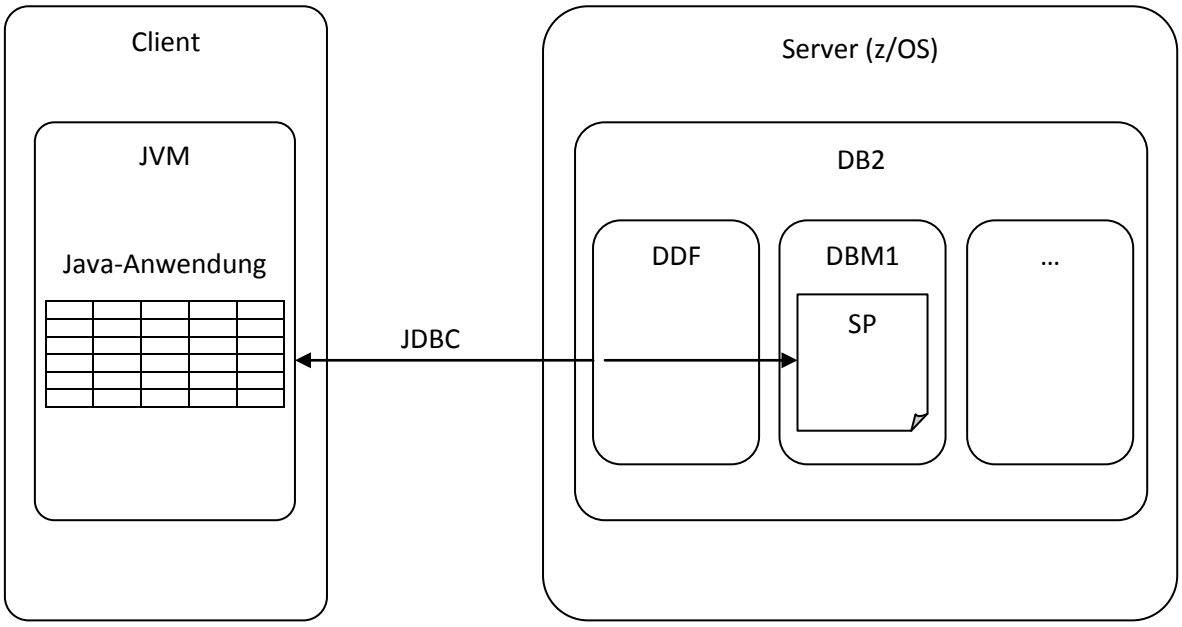
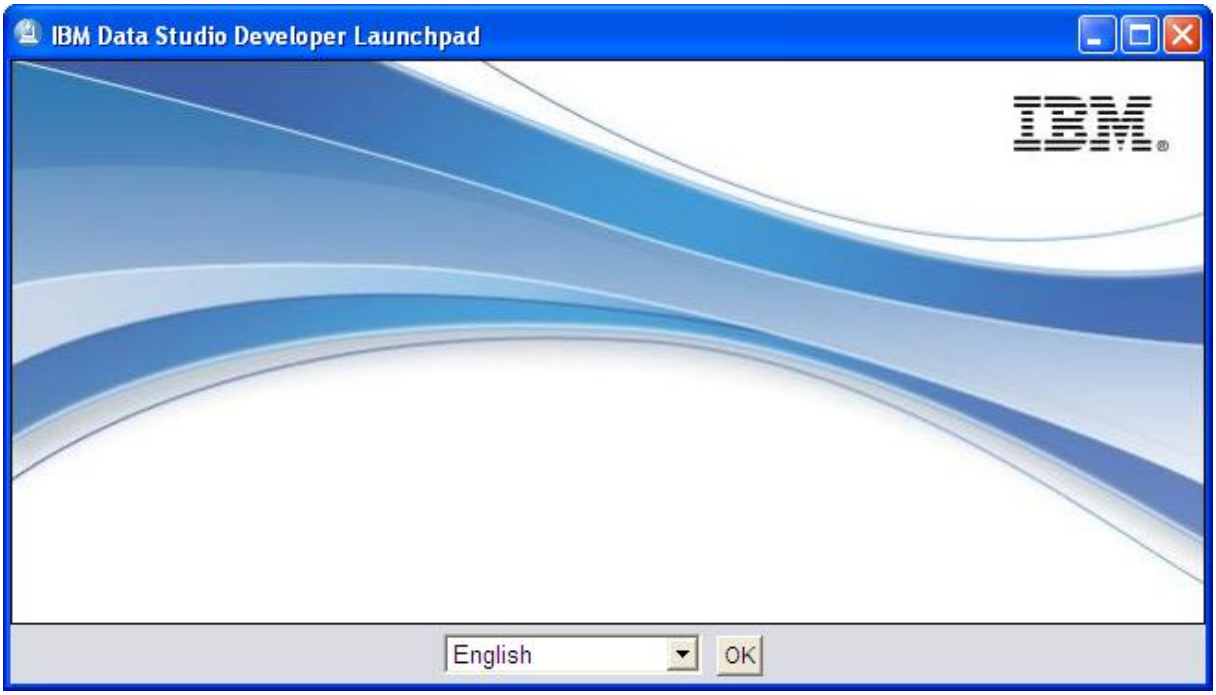


Abbildung 5: Übersicht über den Aufruf von Native SQL Stored Procedures in diesem Tutorial

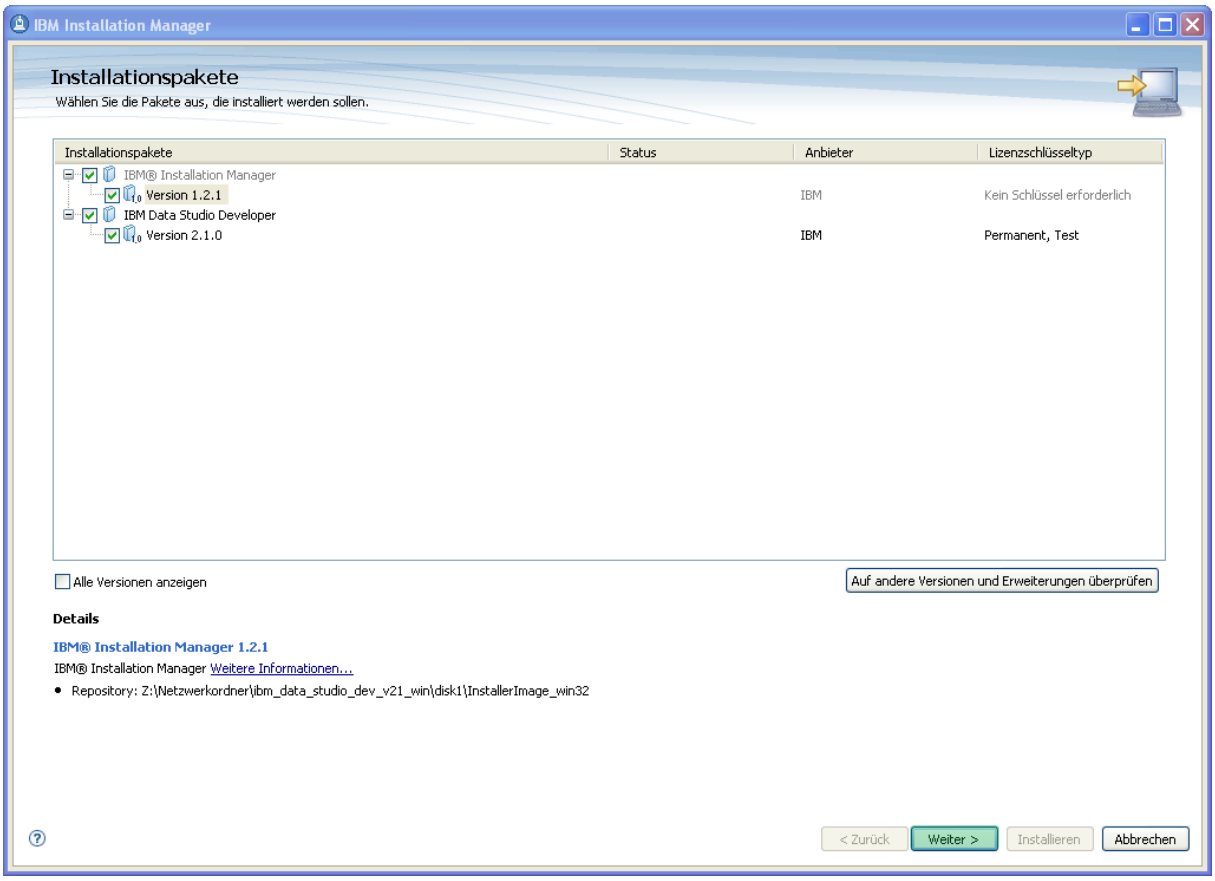
4.2.1 Installation von IBM Data Studio Developer Version 2.1 unter Windows XP

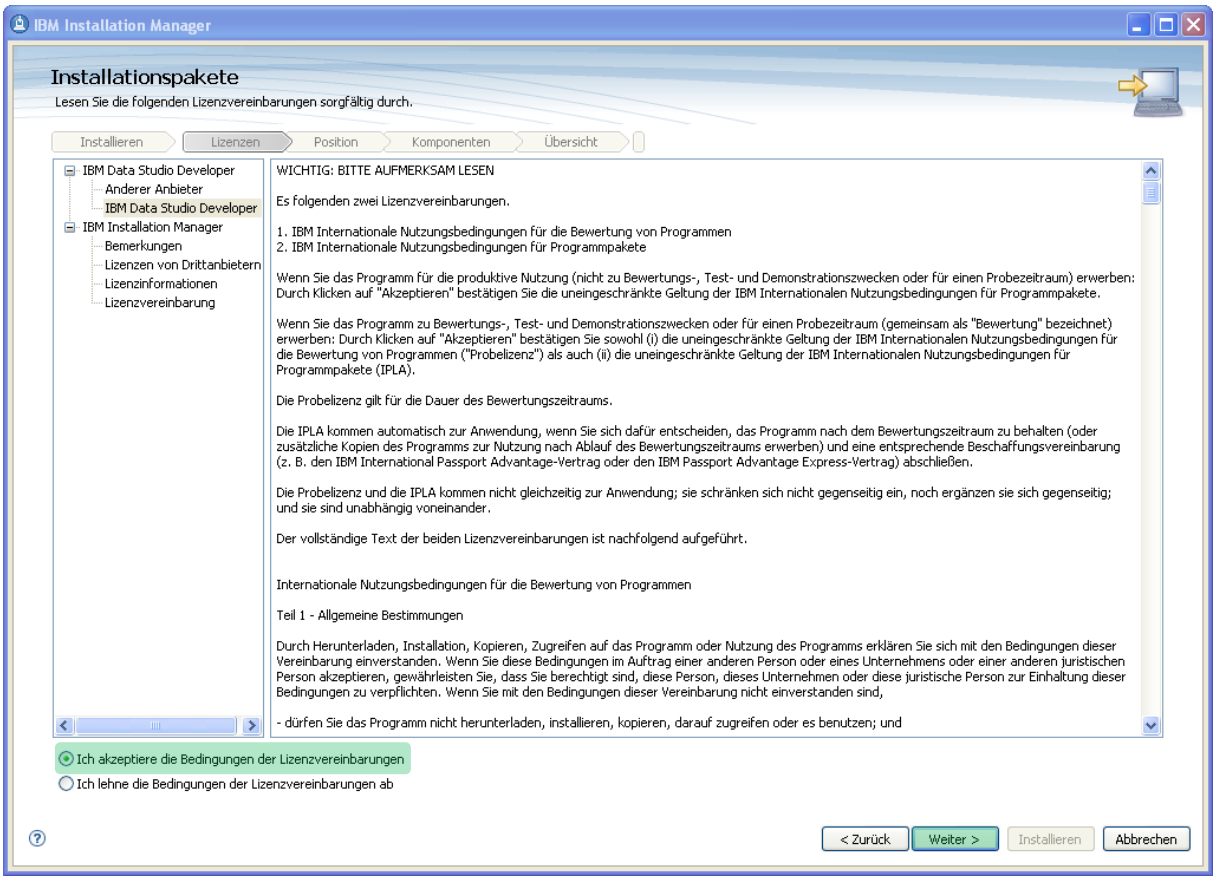
Durch einen Doppelklick auf die Datei setup.exe im Ordner Data Studio startet sich das Installationsprogramm.



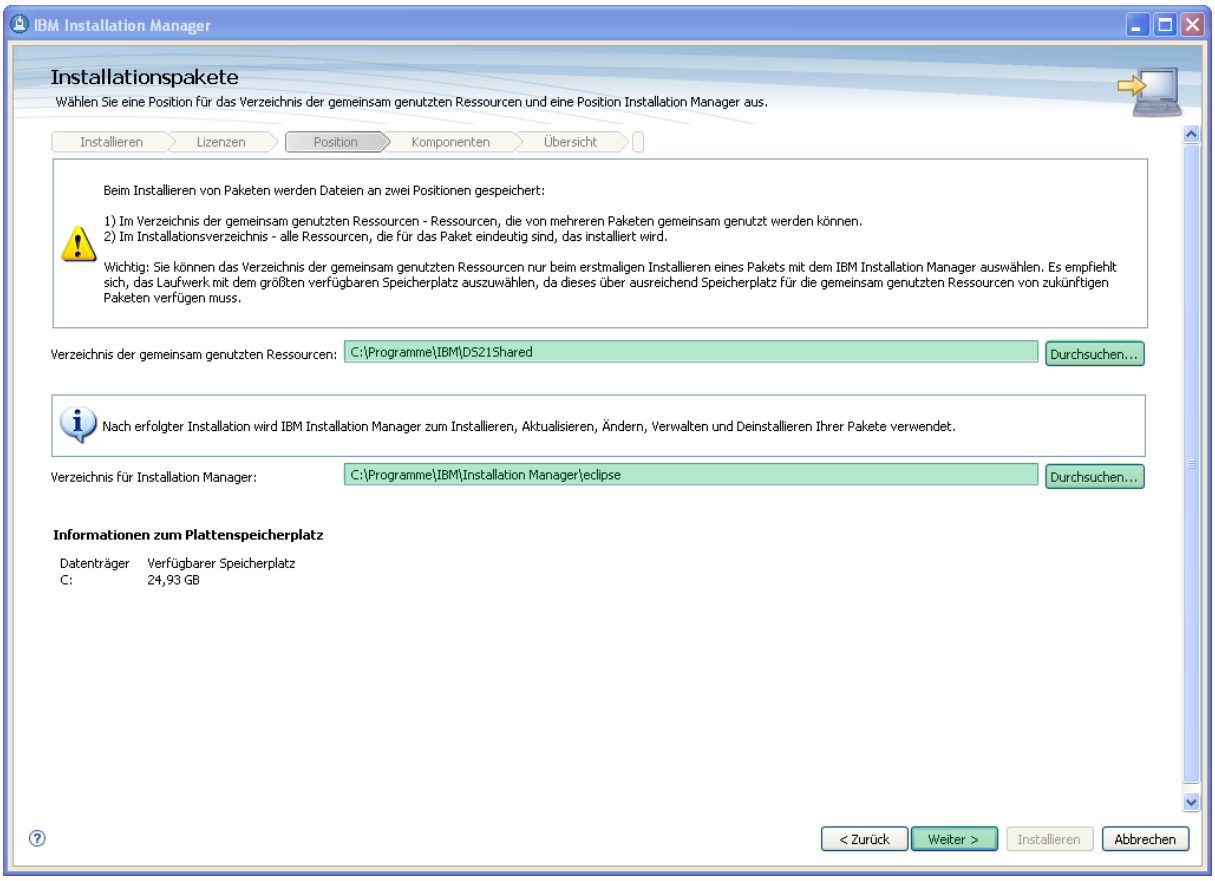
Nachdem Sie die Installationssprache ausgewählt haben und mit „ok“ bestätigt haben, wählen Sie im darauffolgenden Menü auf der linken Seite „Install Product“ aus.

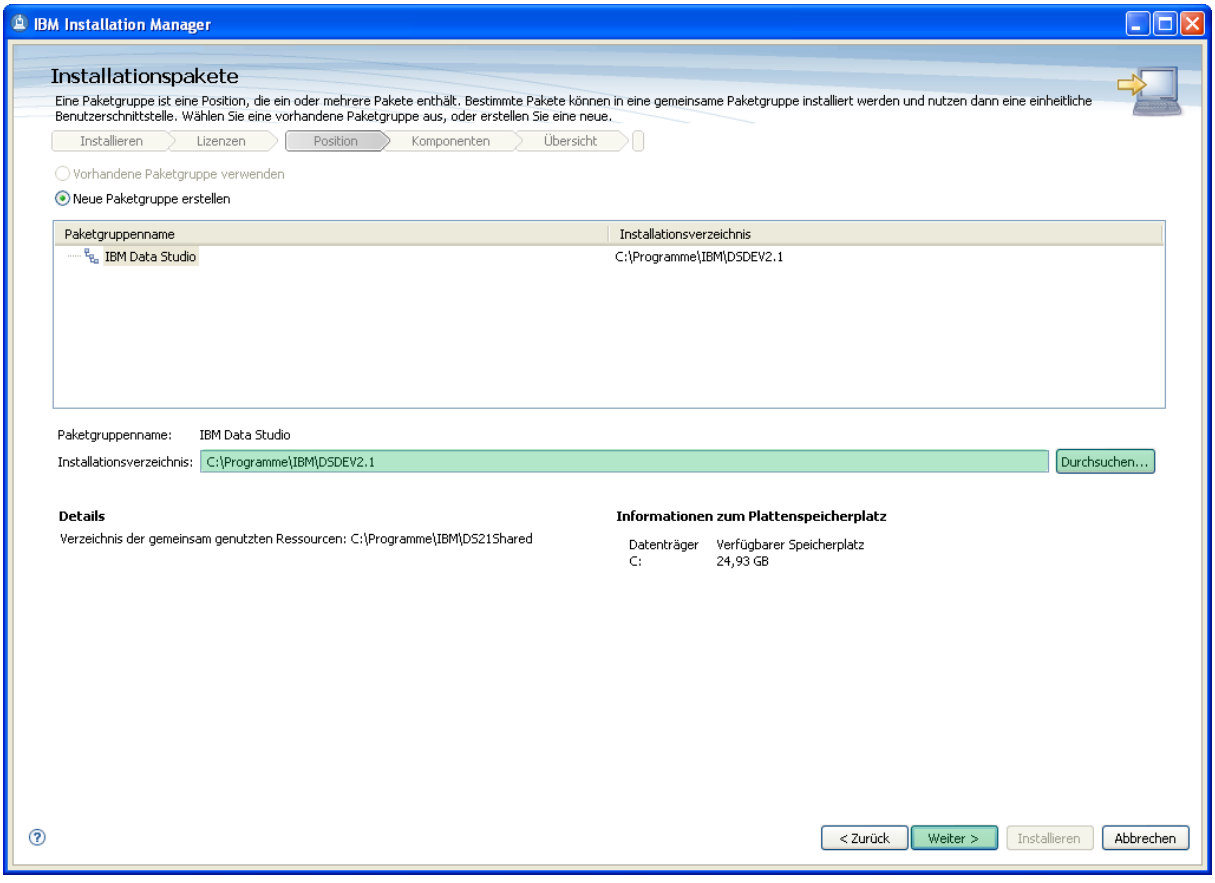
Übernehmen Sie die Standardeinstellungen und klicken Sie auf „Weiter“.





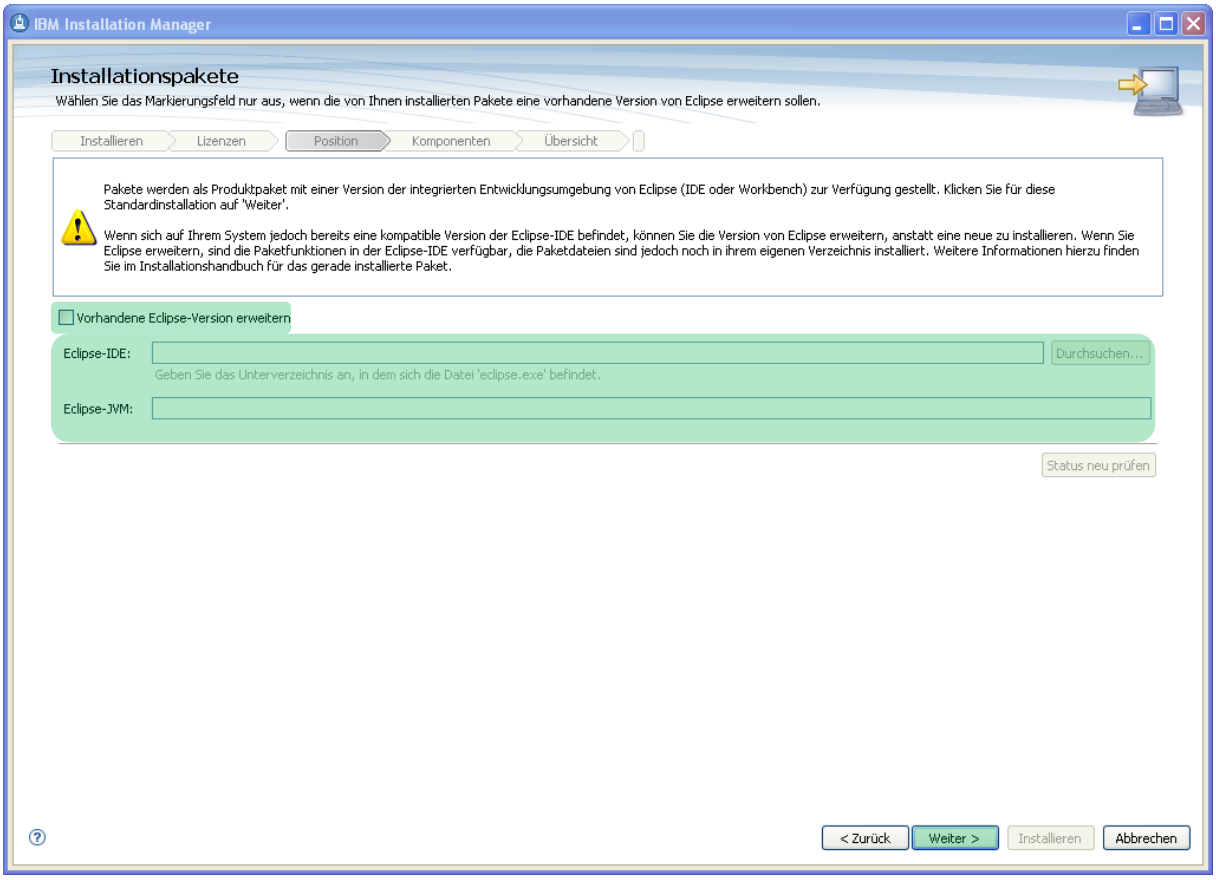
Nachdem Sie den Lizenzvereinbarungen zugestimmt haben, klicken Sie auf „Weiter“. Der Installationsmanager fragt nun nach drei Pfaden (gemeinsame Ressourcen, Installation Manager Verzeichnis und ein Dialogfenster später IBM Data Studio Verzeichnis), in die die entsprechenden Komponenten installiert werden sollen. Auch hier können Sie jeweils den Standardpfad verwenden und auf „Weiter“ klicken.



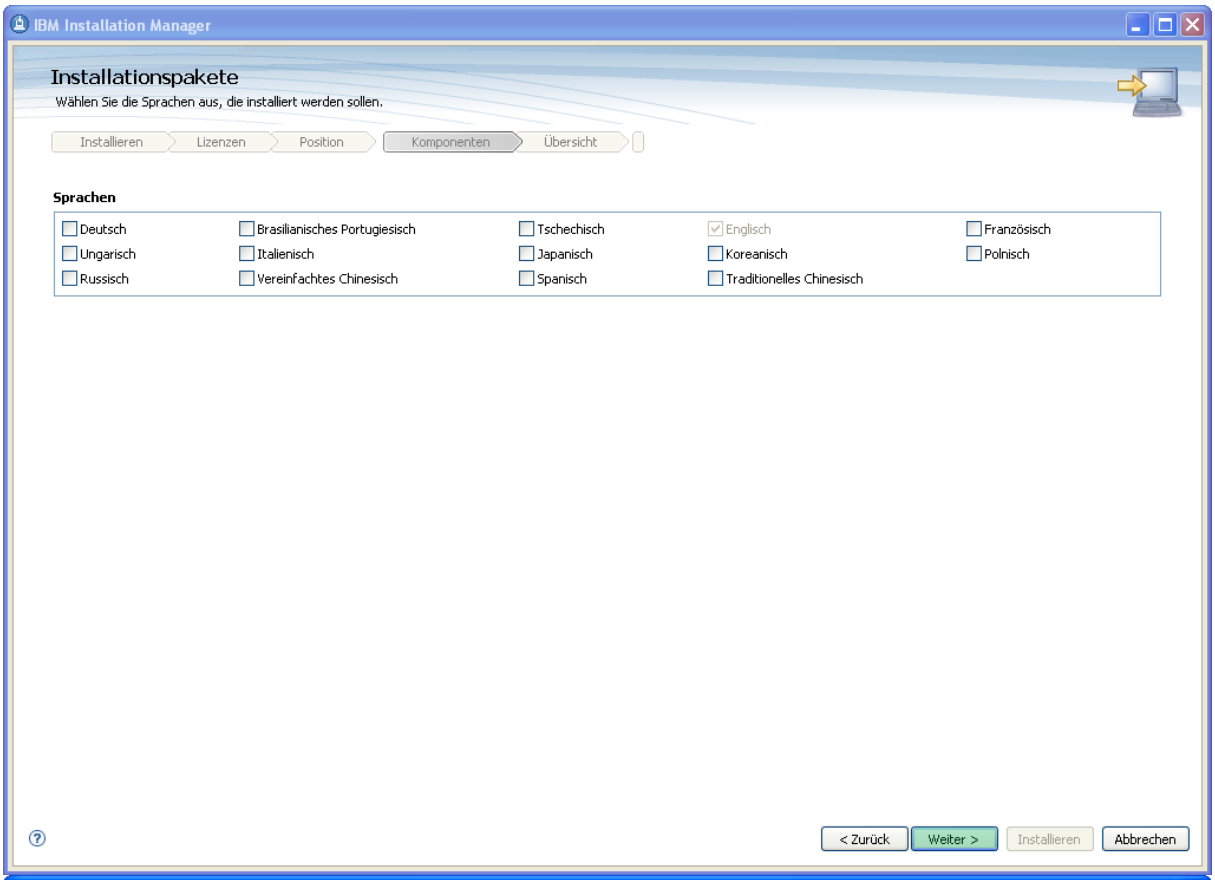


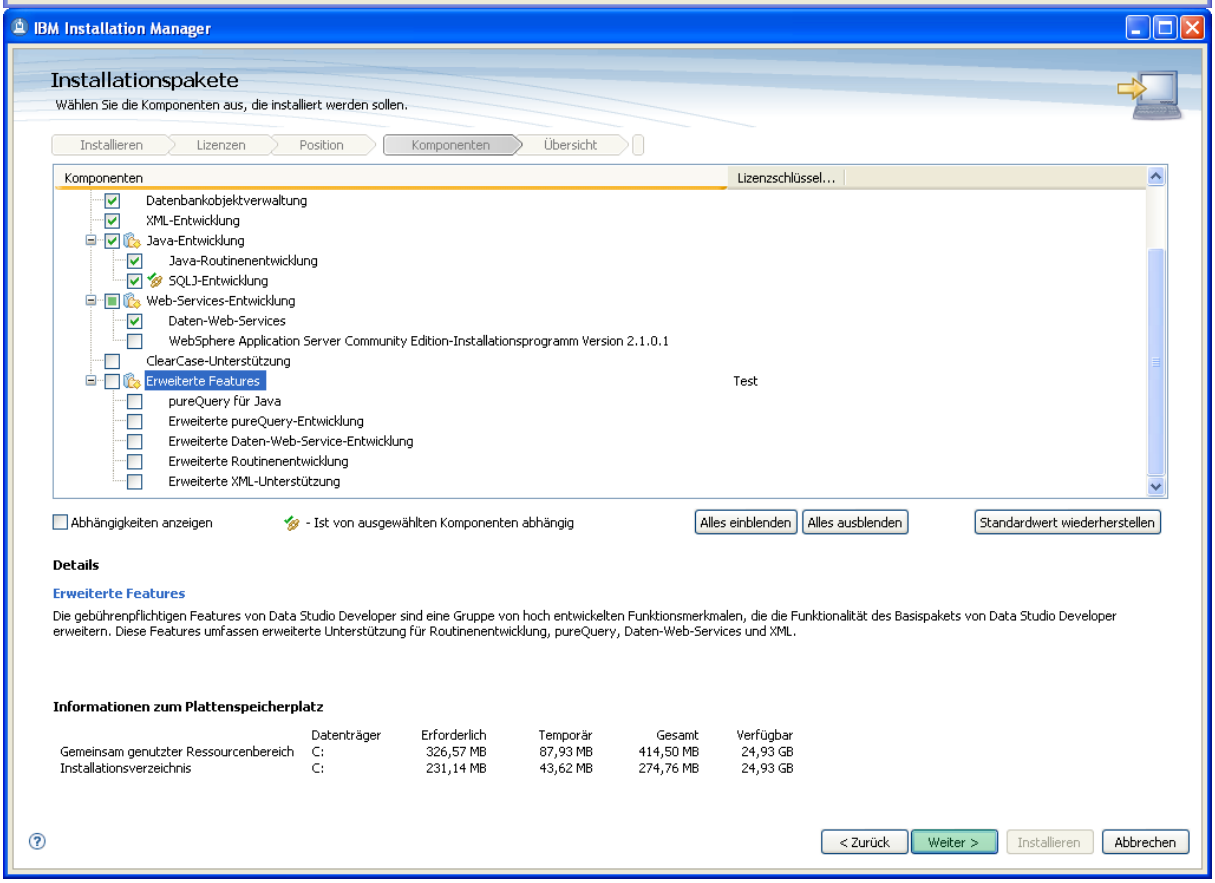
IBM Data Studio ist eine Erweiterung von Eclipse. Deswegen werden Sie an dieser Stelle gefragt, ob eine Eclipseversion/Javaversion bereits installiert ist, auf die das Data Studio aufbauen darf. Wichtig:

Das Data Studio benötigt mindestens eine JVM der Version 1.5 und Eclipse der Version 3.2. Wenn eine solche Version bereits installiert ist, markieren Sie „Vorhandene Eclipse-Version erweitern“ und geben den Pfad zur bereits installierten Eclipse- und Javaversion an. Wenn keine solche Javaversion installiert oder kein Eclipse vorhanden ist, lassen Sie die Felder frei und klicken Sie auf „Weiter“.

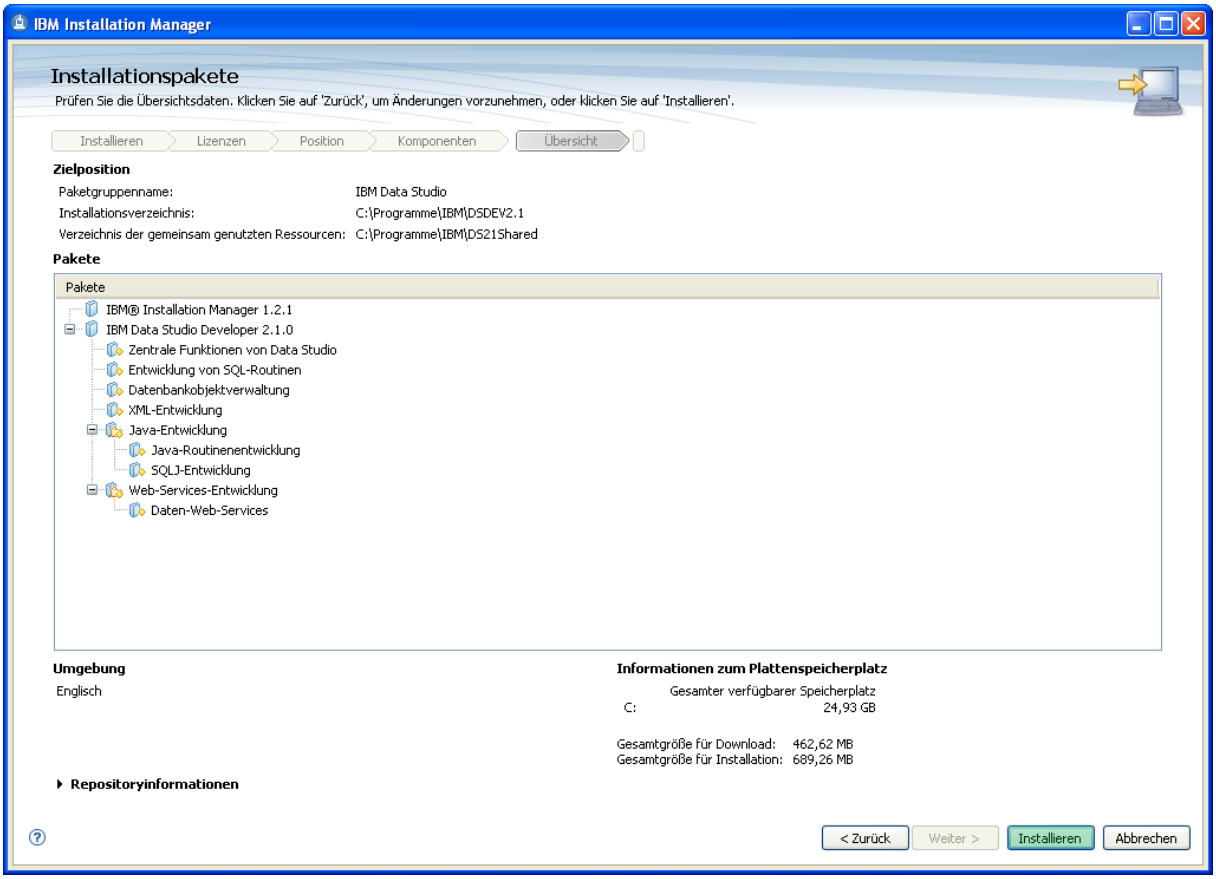


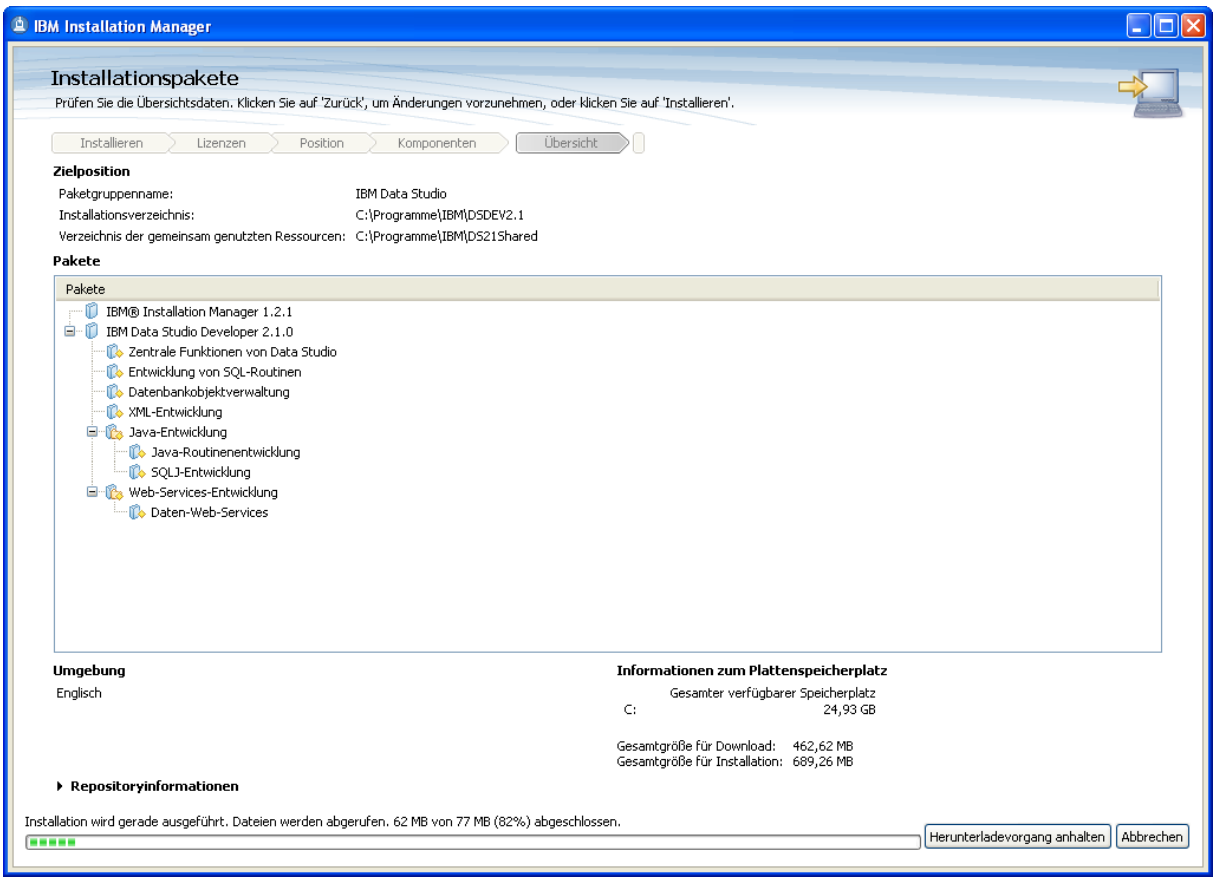
Nun haben Sie die Möglichkeit, verschiedene Sprachunterstützungen auszuwählen. Da die englische Version sprachlich etwas genauer ist, entfernen Sie das Häckchen bei „Deutsch“ und klicken Sie auf „Weiter“. Außerdem lässt sich auswählen, mit welchen Komponenten das Data Studio installiert werden soll: Entfernen Sie das Häckchen bei „Erweiterte Features“. Übernehmen Sie ansonsten die Standardeinstellungen und klicken Sie auf „Weiter“.

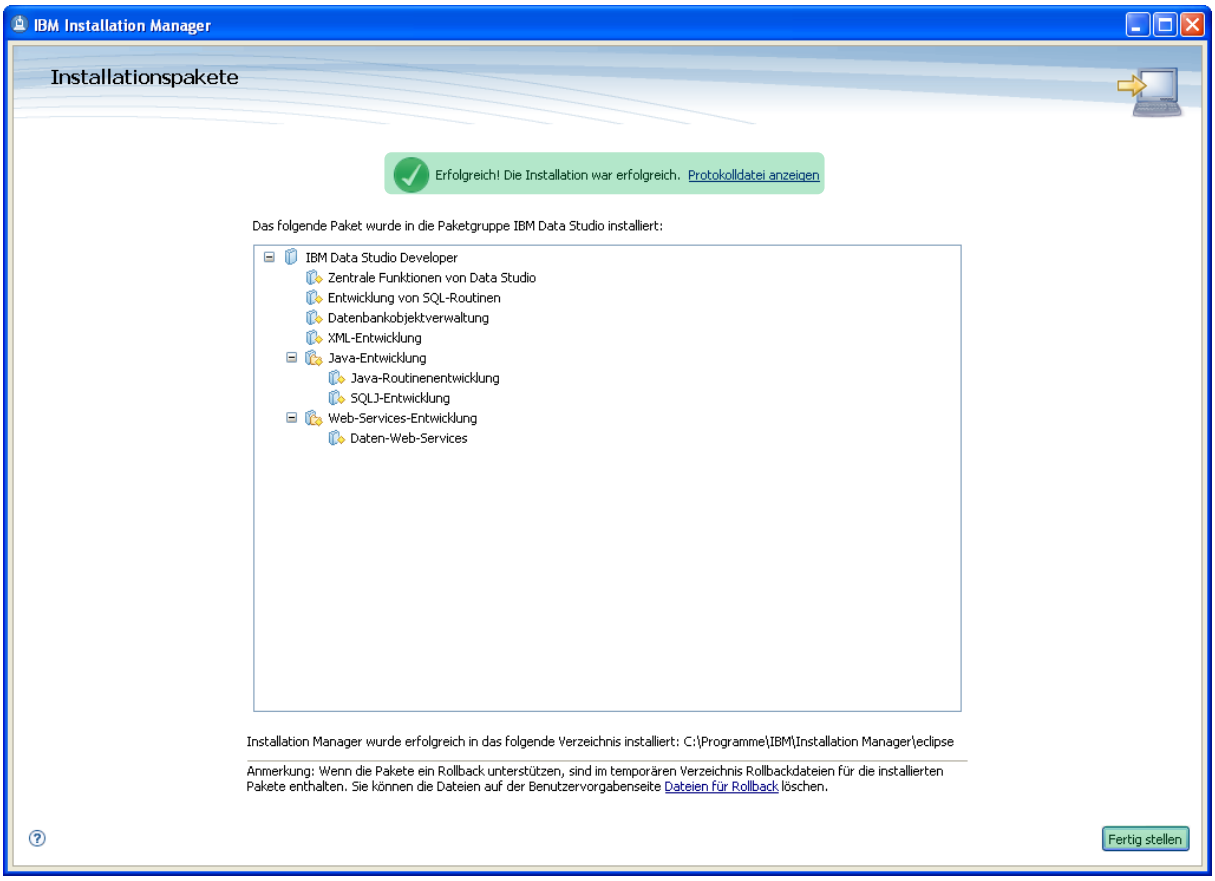

IBM Installation Manager
Installationspakete
Wählen Sie die Sprachen aus, die installiert werden sollen.
Installieren
Lizenzen
Position
Komponenten
Übersicht
Sprachen
Deutsch
Brasilianisches Portugiesisch
Tschechisch
Englisch
Französisch
Ungarisch
Italienisch
Japanisch
Koreanisch
Polnisch
Russisch
Vereinfachtes Chinesisch
Spanisch
Traditionelles Chinesisch
< Zurück
Weiter >
Installieren
Abbrechen


IBM Installation Manager
Installationspakete
Wählen Sie die Komponenten aus, die installiert werden sollen.
Installieren
Lizenzen
Position
Komponenten
Übersicht
Komponenten
Lizenzschlüssel...
Datenbankobjektverwaltung
XML-Entwicklung
Java-Entwicklung
Java-Routinenentwicklung
SQLJ-Entwicklung
Web-Services-Entwicklung
Daten-Web-Services
WebSphere Application Server Community Edition-Installationsprogramm Version 2.1.0.1
ClearCase-Unterstützung
Erweiterte Features
Test
pureQuery für Java
Erweiterte pureQuery-Entwicklung
Erweiterte Daten-Web-Service-Entwicklung
Erweiterte Routinenentwicklung
Erweiterte XML-Unterstützung
Abhängigkeiten anzeigen
- Ist von ausgewählten Komponenten abhängig
Alles einblenden
Alles ausblenden
Standardwert wiederherstellen
Details
Erweiterte Features
Die gebührenpflichtigen Features von Data Studio Developer sind eine Gruppe von hoch entwickelten Funktionsmerkmalen, die die Funktionalität des Basispakets von Data Studio Developer erweitern. Diese Features umfassen erweiterte Unterstützung für Routinenentwicklung, pureQuery, Daten-Web-Services und XML.
Informationen zum Plattenspeicherplatz
Datenträger
Erforderlich
Temporär
Gesamt
Verfügbar
Gemeinsam genutzter Ressourcenbereich
C:
326,57 MB
87,93 MB
414,50 MB
24,93 GB
Installationsverzeichnis
C:
231,14 MB
43,62 MB
274,76 MB
24,93 GB
< Zurück
Weiter >
Installieren
Abbrechen

Der Installations Manager zeigt jetzt noch einmal eine Übersicht über die bevorstehende Installation. Durch einen Klick auf „Installieren“ wird die Installation gestartet.

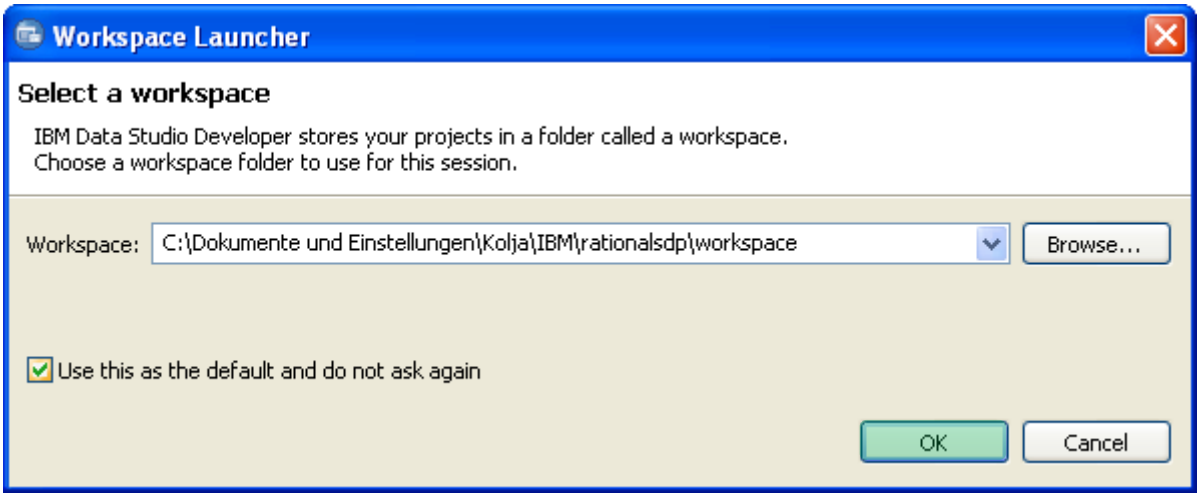




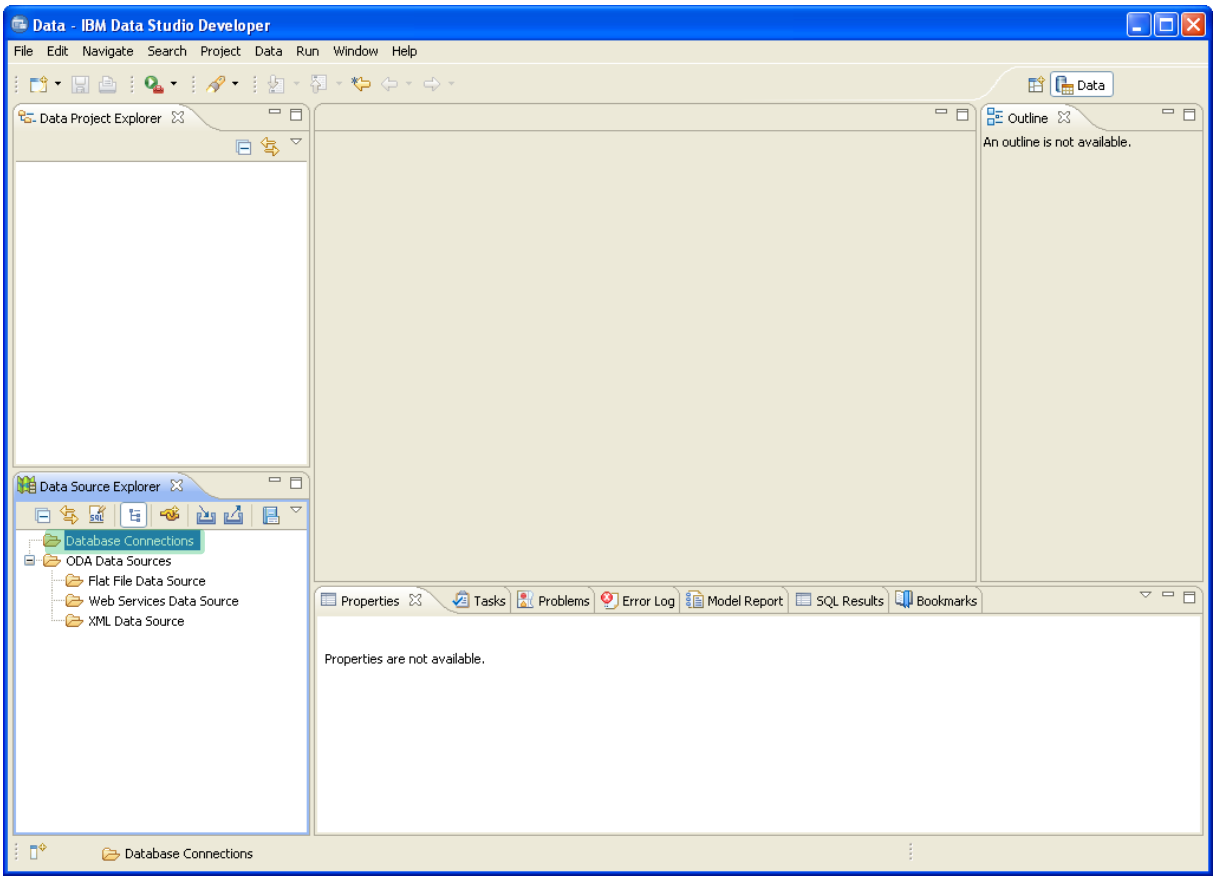


Ist der Installationsvorgang fertig, wird Ihnen angezeigt, ob die Installation erfolgreich verlaufen ist. Klicken Sie auf „Fertigstellen“ und schließen Sie alle zum Data Studio zugehörigen Fenster. Danach starten wir den IBM Data Studio Developer. Ein Link hierfür wurde automatisch dem Windows-Startmenü hinzugefügt.

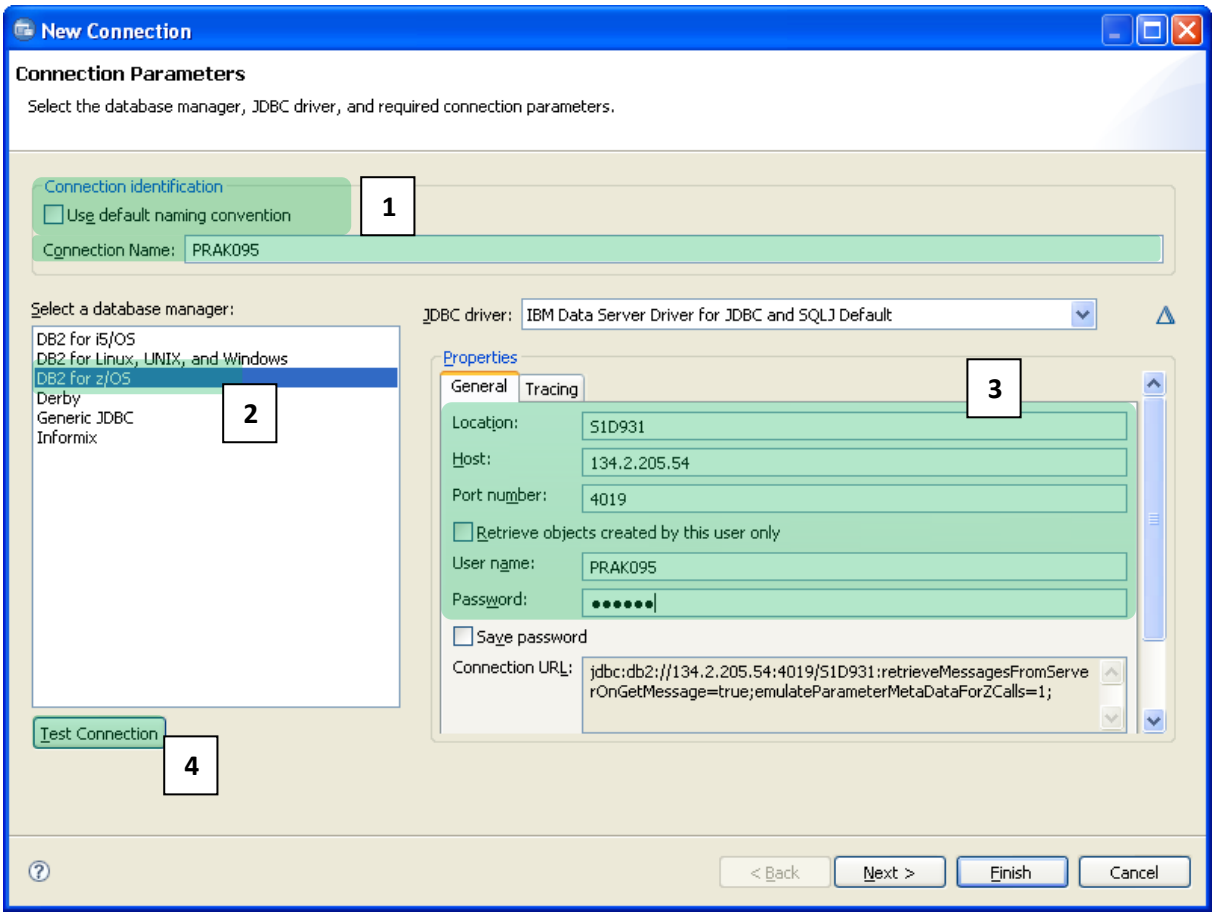
Zunächst werden Sie nach einem Speicherort für den Workspace gefragt. Wählen Sie einen Pfad aus und betätigen Sie den ok-Knopf.



Nachdem Sie den Willkommensbildschirm geschlossen haben, zeigt sich Ihnen eine übliche Eclipseoberfläche.



Um eine Verbindung mit dem DB2-System auf dem Großrechner herzustellen, wird im Data Source Explorer mit der rechten Maustaste auf „Database Connections“ geklickt und „New …“ ausgewählt.

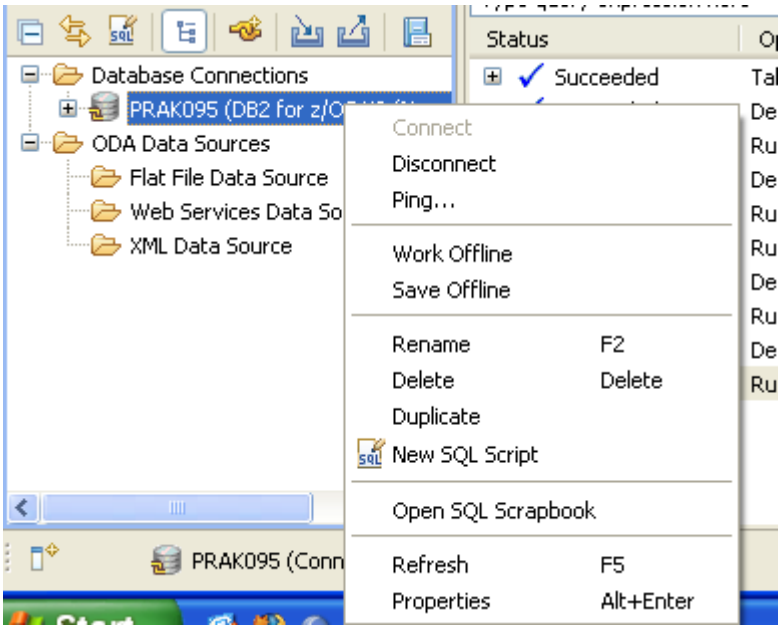


Im ersten Schritt wird ein Name für die Verbindung eingegeben. Um eine Verbindung mit dem Tübinger/Leipziger Großrechner herzustellen, müssen Sie auf der linken Seite „DB2 for z/OS“ auswählen und folgende Informationen eintragen:

Für die Tübinger LPAR Hobbit:		Für die Leipziger LPAR Binks:	
Location:	S1D931	Location:	S1D931
Host:	134.2.205.54	Host:	139.18.4.34
Port:	4019	Port:	4019

Entfernen Sie das Häkchen bei „Retrieve objects created by this user only“ und geben Sie im Feld „User name“ Ihre Benutzerkennung und darunter Ihr Passwort ein. Danach können Sie mit dem Knopf „Test Connection“ überprüfen, ob die Verbindung aufgebaut werden kann. IBM Data Studio Developer baut hierzu eine JDBC-Typ 4-Verbindung auf (Beim Typ-4-Treiber werden die JDBC-API-Befehle direkt in DBMS-Befehle des jeweiligen Datenbankservers übersetzt und an diesen übertragen. Ein Middleware-Treiber wird dabei nicht verwendet).

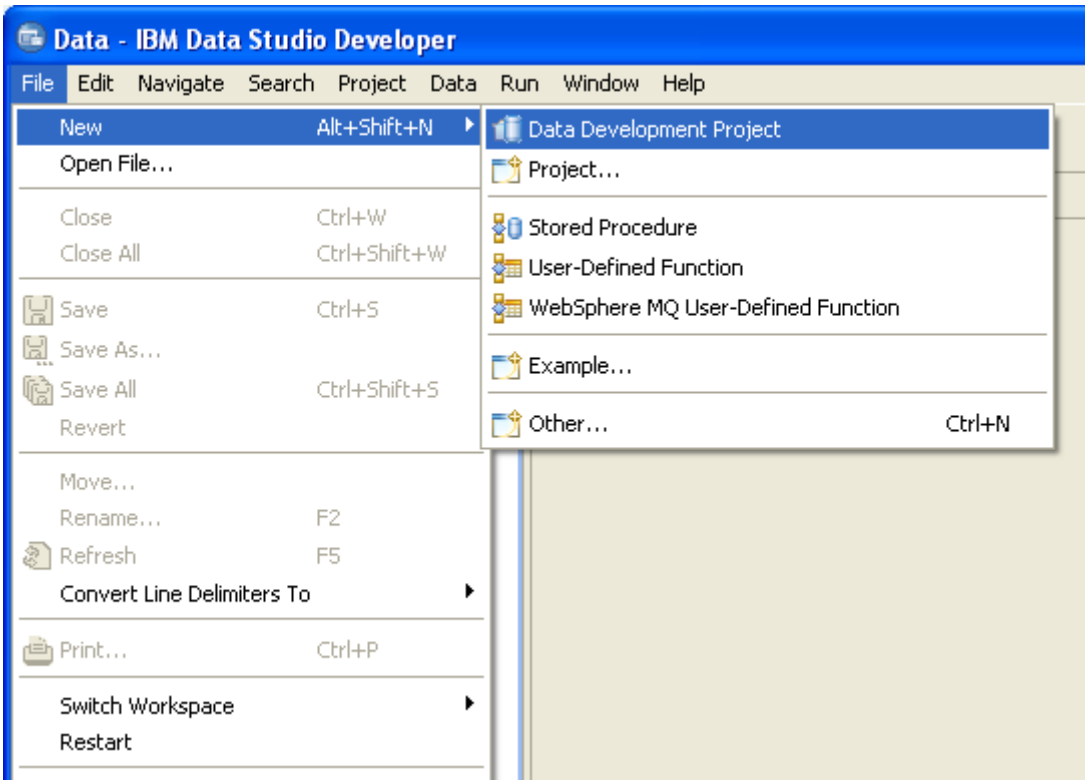
Mit einem Klick auf „Finish“ schließt sich das Fenster und die Verbindung wird im Datasource-Explorer aufgelistet. Durch einen Klick mit der rechten Maustaste auf die Verbindung kann sie jederzeit gesteuert werden (Eigenschaften bearbeiten, trennen oder verbinden).


Database Connections
PRAK095 (DB2 for z/O
ODA Data Sources
Flat File Data Source
Web Services Data So
XML Data Source
Status
Succeeded
Connect
Disconnect
Ping...
Work Offline
Save Offline
Rename F2
Delete Delete
Duplicate
New SQL Script
Open SQL Scrapbook
Refresh F5
Properties Alt+Enter
PRAK095 (Conn

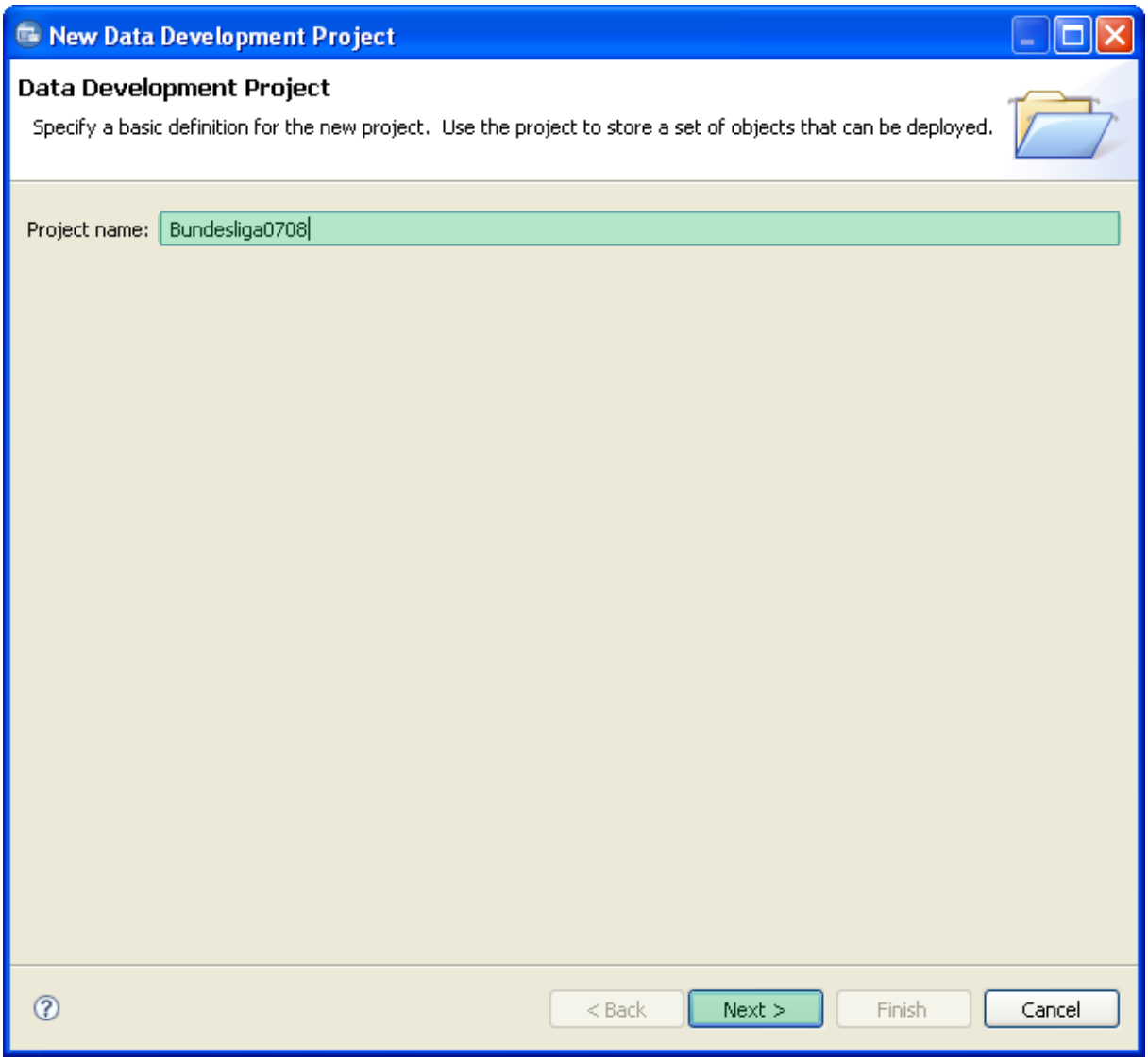
Um den nachfolgenden Teil durchführen zu können, wird vorausgesetzt, dass eine Datenbank eingerichtet ist. Wenn Sie Tutorial 4 durchgeführt haben, sind keine weiteren Schritte nötig.

Um mit dem IBM Data Studio Developer vertraut zu werden, werden zunächst ein paar Grundfunktionalitäten gezeigt: Eine Datenbanktabelle wird erstellt, mit Daten gefüllt und eine SELECT-Anfrage darauf über den IBM Data Studio Developer gestellt.

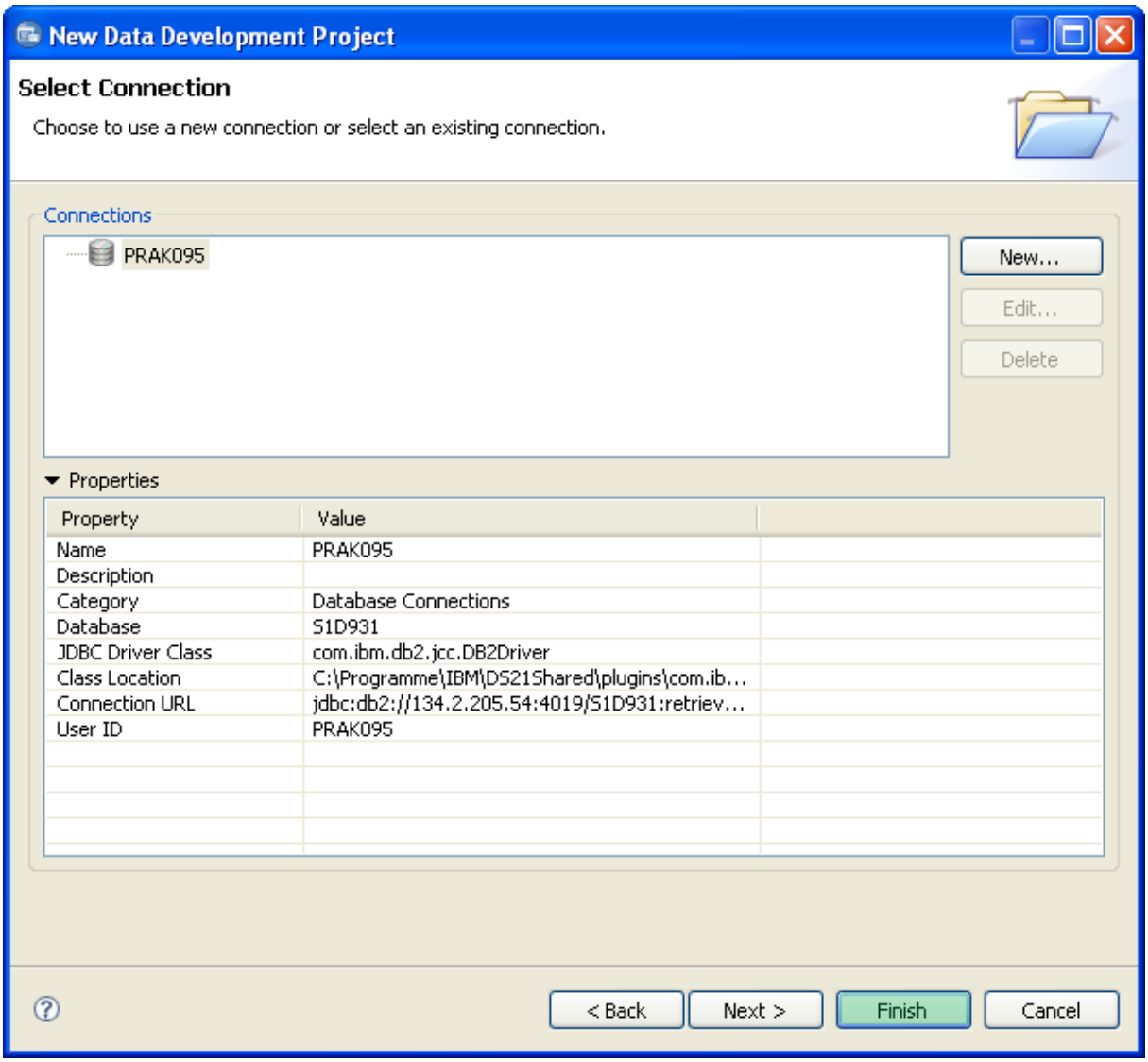
Zunächst legen Sie ein neues Datenentwicklungsprojekt an, indem Sie im Menü „File“ unter „New“, „Data Development Project“ auswählen.



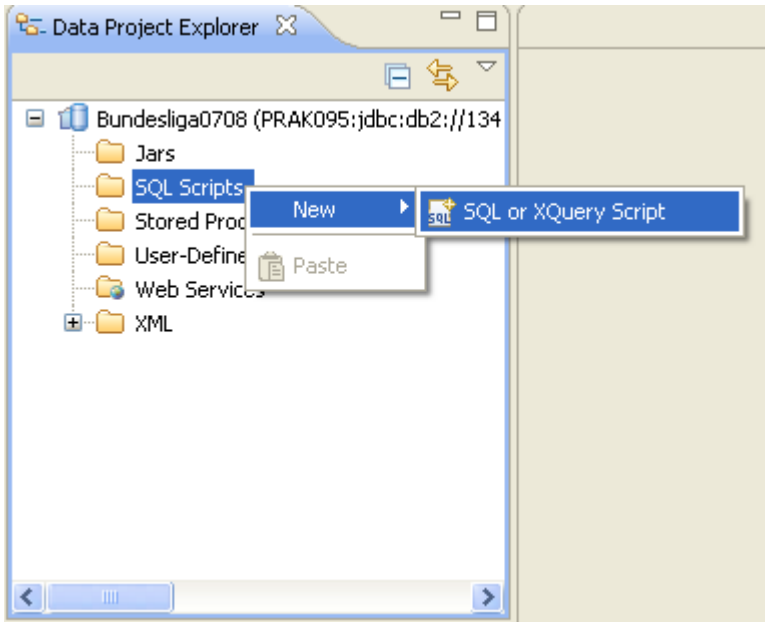
Geben Sie einen Namen für Ihr Projekt ein und klicken Sie auf „Next“.



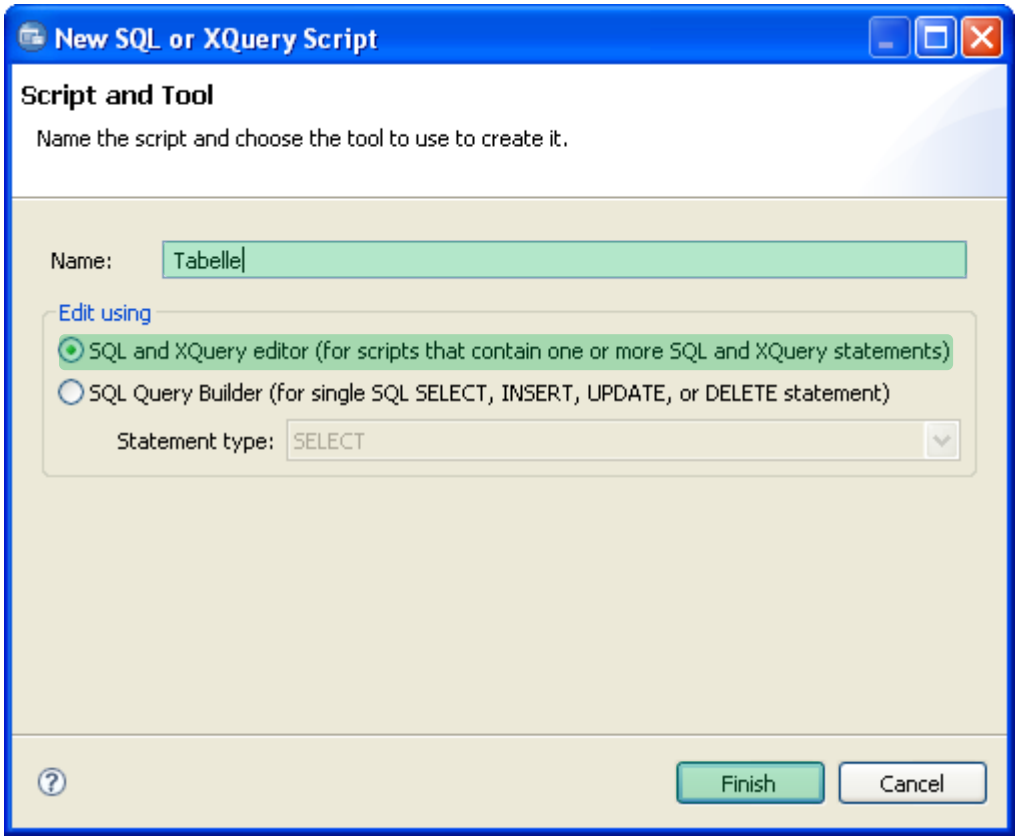
Nun können Sie auswählen, welche Verbindung verwendet werden soll, um die späteren SQL-Befehle auszuführen. Standardmäßig wird die vorher erstellte Verbindung ausgewählt. Durch einen Klick auf „Finish“ schließt sich das Fenster und Sie bekommen das Projekt im Data Project Explorer auf der linken Seite angezeigt.



Durch einen Klick auf das Plussymbol vor dem Projekt, werden die einzelnen Komponenten des Projekts angezeigt.



Um eine neue Datenbanktabelle zu erstellen, klicken Sie mit der rechten Maustaste auf den Ordner „SQL-Scripts“ und wählen unter dem Menüeintrag „New“ ein „SQL- or XQuery-Script“ aus. Es öffnet sich ein Fenster, in das Sie einen Namen für das Skript eintragen und auswählen können, ob die SQL-Befehle per Drag&Drop mit dem SQL-Builder erstellt werden sollen oder ob Sie sie selbst schreiben möchten. Da hier die SQL-Befehle selbst geschrieben werden sollen, wählen Sie den SQL-Editor aus und klicken auf „Finish“. Den SQL-Builder werden Sie später noch kennenlernen.

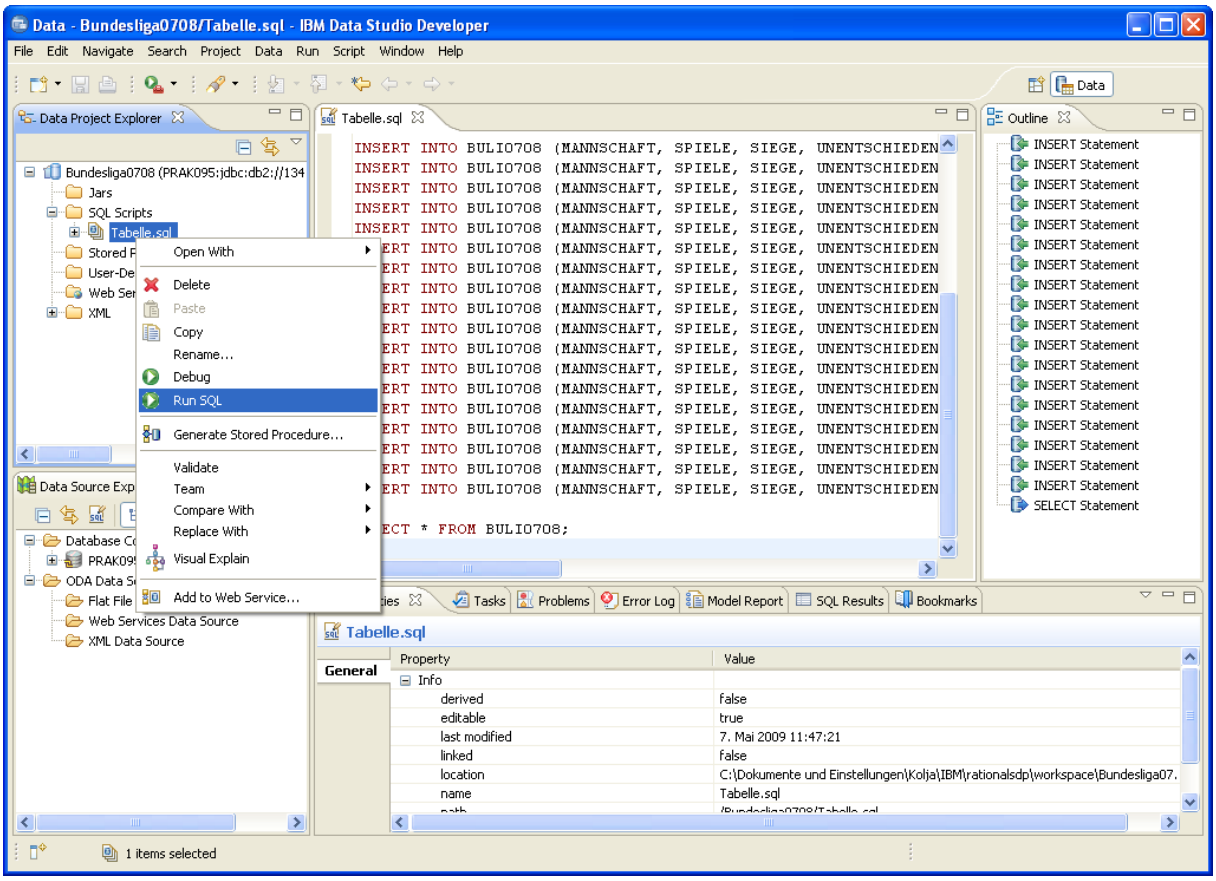


Nun soll eine Tabelle der Fußball-Bundesligaspiele der Saison 2007/2008 nach dem 31.Spieltag erstellt werden. Deshalb werden in das Skript folgende SQL-Befehle eingefügt:

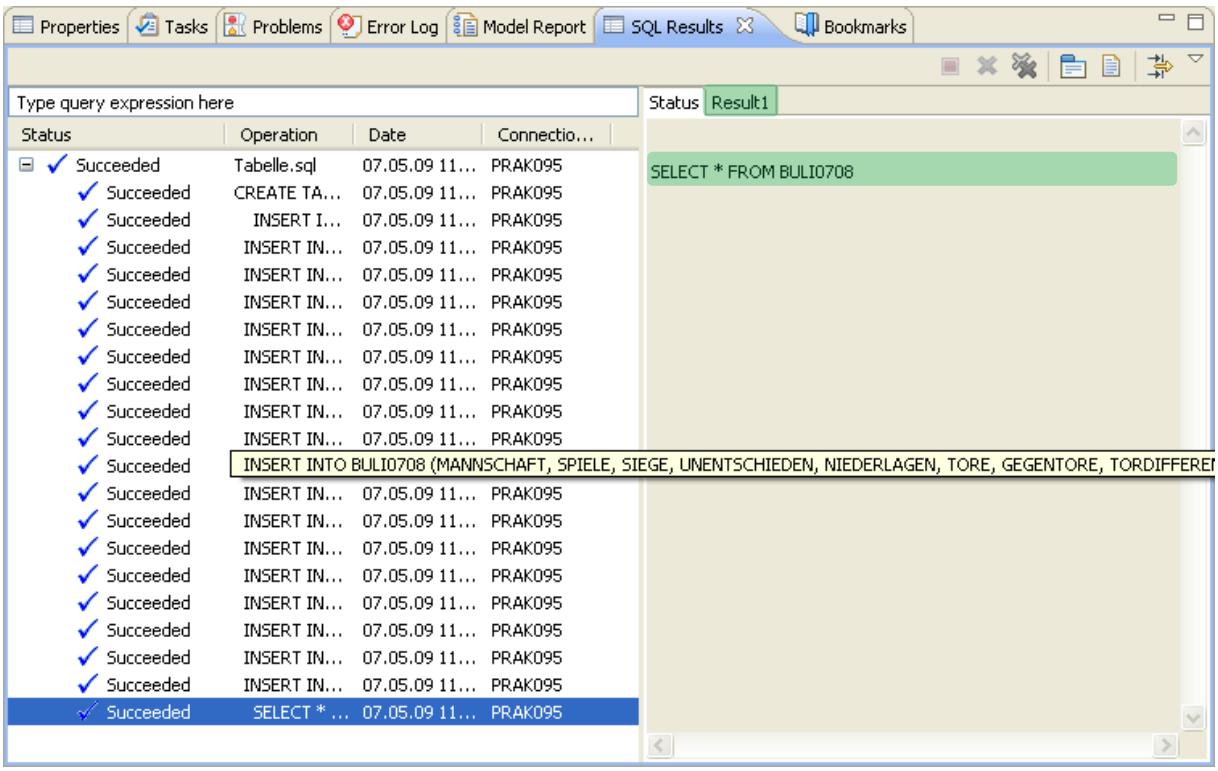
```
CREATE TABLE BULI0708(
MANNSCHAFT VARCHAR(20) NOT NULL,
SPIELE SMALLINT NOT NULL,
SIEGE SMALLINT NOT NULL,
NIEDERLAGEN SMALLINT NOT NULL,
UNENTSCHIEDEN SMALLINT NOT NULL,
TORE SMALLINT NOT NULL,
GEGENTORE SMALLINT NOT NULL,
TORDIFFERENZ SMALLINT NOT NULL,
PUNKTE SMALLINT NOT NULL
);

INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Arminia
Bielefeld', 31, 8, 8, 15, 31, 54, -23, 32);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Bayer
Leverkusen', 31, 14, 6, 11, 54, 36, 18, 48);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Bayern
München', 31, 19, 10, 2, 59, 18,    41, 67);
```

```
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Borussia
Dortmund', 31, 9, 9, 13, 43, 54, -11, 36);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Eintracht
Frankfurt', 31, 11, 10, 10, 37, 44, -7, 43);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Energie
Cottbus', 31, 8, 8, 15, 32, 51, -19, 32);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('FC Schalke 04',
31,   15, 10, 6, 49, 32, 17, 55);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Hamburger SV',
31, 13, 12, 6, 40, 23, 17, 51);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Hannover 96',
31, 11, 10, 10, 46, 50, -4, 43);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Hansa Rostock',
31, 7, 6, 18, 27, 46, -19, 27);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Hertha BSC',
31, 10, 8, 13, 35, 39, -4, 38);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Karlsruher SC',
31, 11, 9, 11, 36, 42, -6, 42);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('MSV Duisburg',
31, 8, 5, 18, 32, 46, -14, 29);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('VfB Stuttgart',
31, 16, 3, 12, 53, 48, 5, 51);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('VfL Bochum',
31, 9, 11, 11, 44, 48, -4, 38);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('VfL Wolfsburg',
31, 12, 9, 10, 47, 42, 5, 45);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('Werder Bremen',
31, 17, 6, 8, 67, 44, 23, 57);
INSERT INTO BULI0708 (MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN,
NIEDERLAGEN, TORE, GEGENTORE, TORDIFFERENZ, PUNKTE) VALUES('1. FC
Nürnberg', 31, 6, 10, 15, 33, 48, -15, 28);

SELECT * FROM BULI0708;
```

Der CREATE TABLE-Befehl zu Beginn legt eine Tabelle mit dem Namen BULI0708 an. Die 18 folgenden INSERT-Befehle fügen jeweils eine Mannschaft mit den zugehörigen Informationen wie Anzahl der Spiele, Anzahl der Siege, Anzahl der Unentschieden, Anzahl der Niederlagen, Anzahl der Tore, Anzahl der Gegentore, Tordifferenz und der Anzahl der Punkte ein. Der letzte Befehl ist ein SELECT-Befehl, der sämtliche Einträge der Tabelle wieder ausgibt. Speichern Sie das Skript ab und klicken Sie im Data Project Explorer mit der rechten Maustaste darauf. Im aufklappenden Menü wählen Sie „Run SQL“ aus.



Nun wird das SQL-Script im DB2-System auf dem Großrechner ausgeführt. Den Erfolg sehen Sie im Feld SQL Results im unteren Bereich des Data Studios:



Der oberste Befehl ist der SELECT-Befehl. Er liefert 18 Datensätze zurück, die im Feld Result1 betrachtet werden können.

Status Result1

	MANNSCHAFT	SPIELE	SIEGE	NIEDERLAGEN	UNENTSCHIEDEN	TORE	GEGENTORE	TORDIFFERENZ	PUNKTE
1	Arminia Biel...	31	8	15	8	31	54	-23	32
2	Bayer Lever...	31	14	11	6	54	36	18	48
3	Bayern Mün...	31	19	2	10	59	18	41	67
4	Borussia Do...	31	9	13	9	43	54	-11	36
5	Eintracht Fr...	31	11	10	10	37	44	-7	43
6	Energie Cot...	31	8	15	8	32	51	-19	32
7	FC Schalke 04	31	15	6	10	49	32	17	55
8	Hamburger SV	31	13	6	12	40	23	17	51
9	Hannover 96	31	11	10	10	46	50	-4	43
10	Hansa Rostock	31	7	18	6	27	46	-19	27
11	Hertha BSC	31	10	13	8	35	39	-4	38
12	Karlsruher SC	31	11	11	9	36	42	-6	42
13	MSV Duisburg	31	8	18	5	32	46	-14	29
14	VfB Stuttgart	31	16	12	3	53	48	5	51
15	VfL Bochum	31	9	11	11	44	48	-4	38
16	VfL Wolfsburg	31	12	10	9	47	42	5	45
17	Werder Bre...	31	17	8	6	67	44	23	57
18	1. FC Nürnb...	31	6	15	10	33	48	-15	28

Total 18 records shown

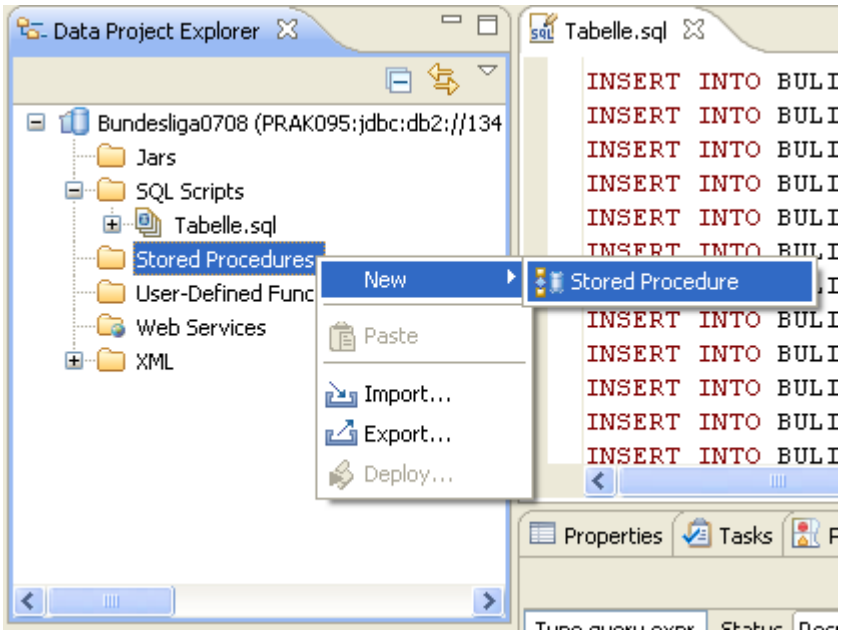
Wenn das Script erneut ausgeführt werden soll, muss davor die Tabelle BULI0708 wieder gelöscht werden. Dies lässt sich durch das Einfügen von

```
DROP TABLE BULI0708;
```

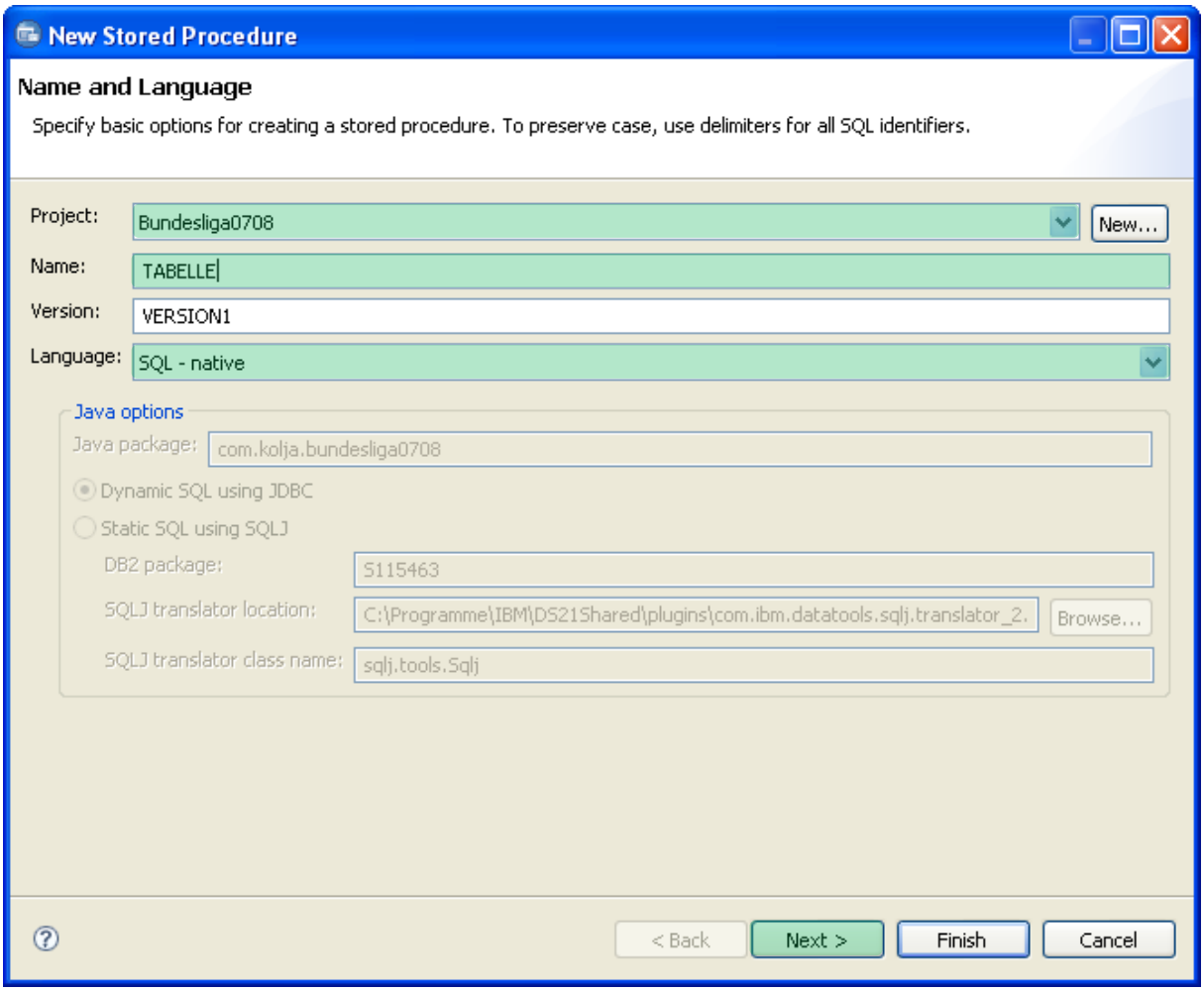
zu Beginn des Skripts erreichen.

4.2.2 Stored Procedures anlegen

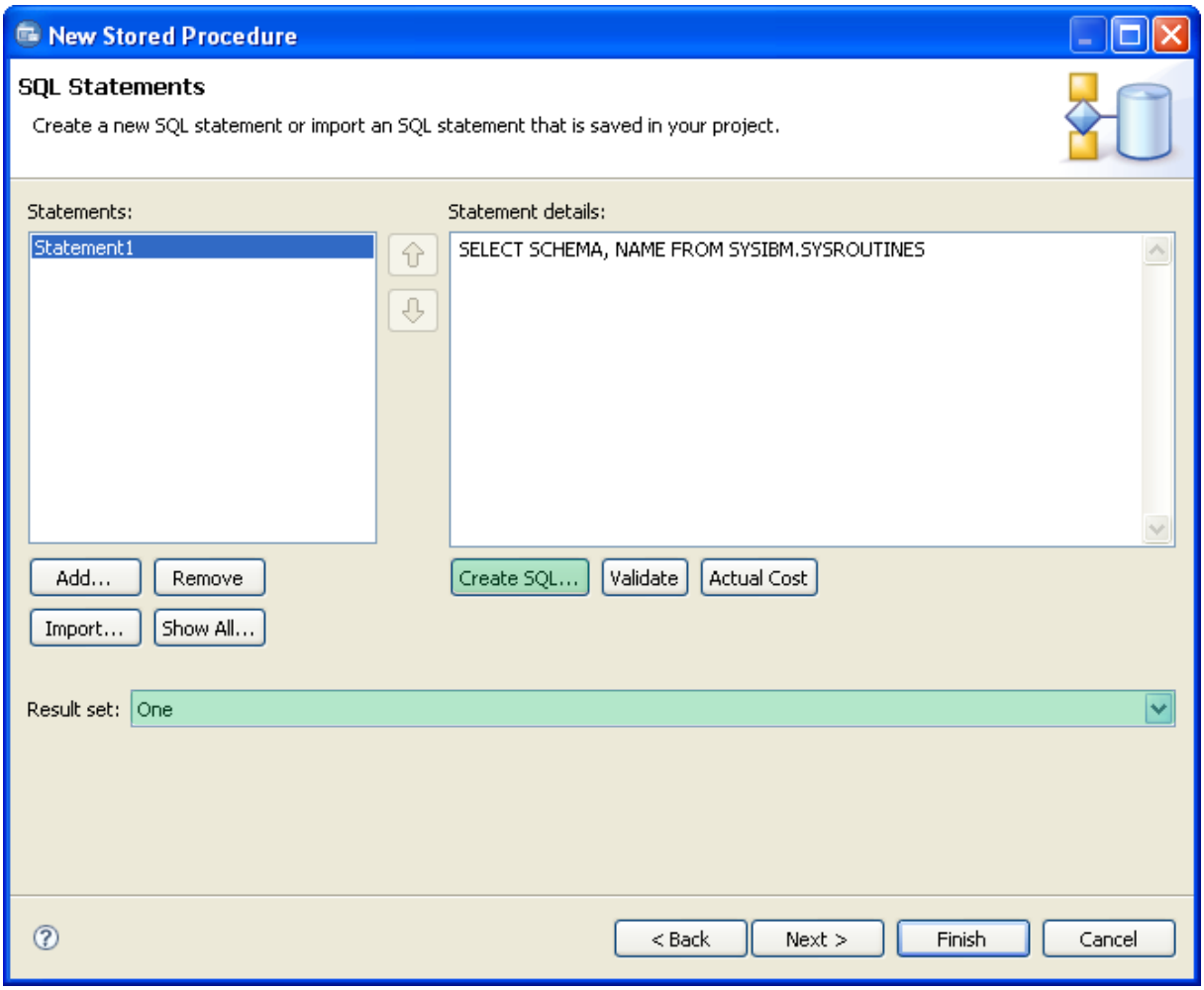
Im folgenden Teil soll eine Stored Procedure angelegt werden, die fast dieselbe Rückgabe hat, wie das Select aus dem vorigen Teil. Die Stored Procedure soll die sortierte Bundesligatabelle zurückgeben. Hierzu klicken Sie mit der rechten Maustaste auf den Ordner Stored Procedures des Projekts im Data Project Explorer. Im erscheinenden Menü wählen Sie „New“ und danach „Stored Procedure“.

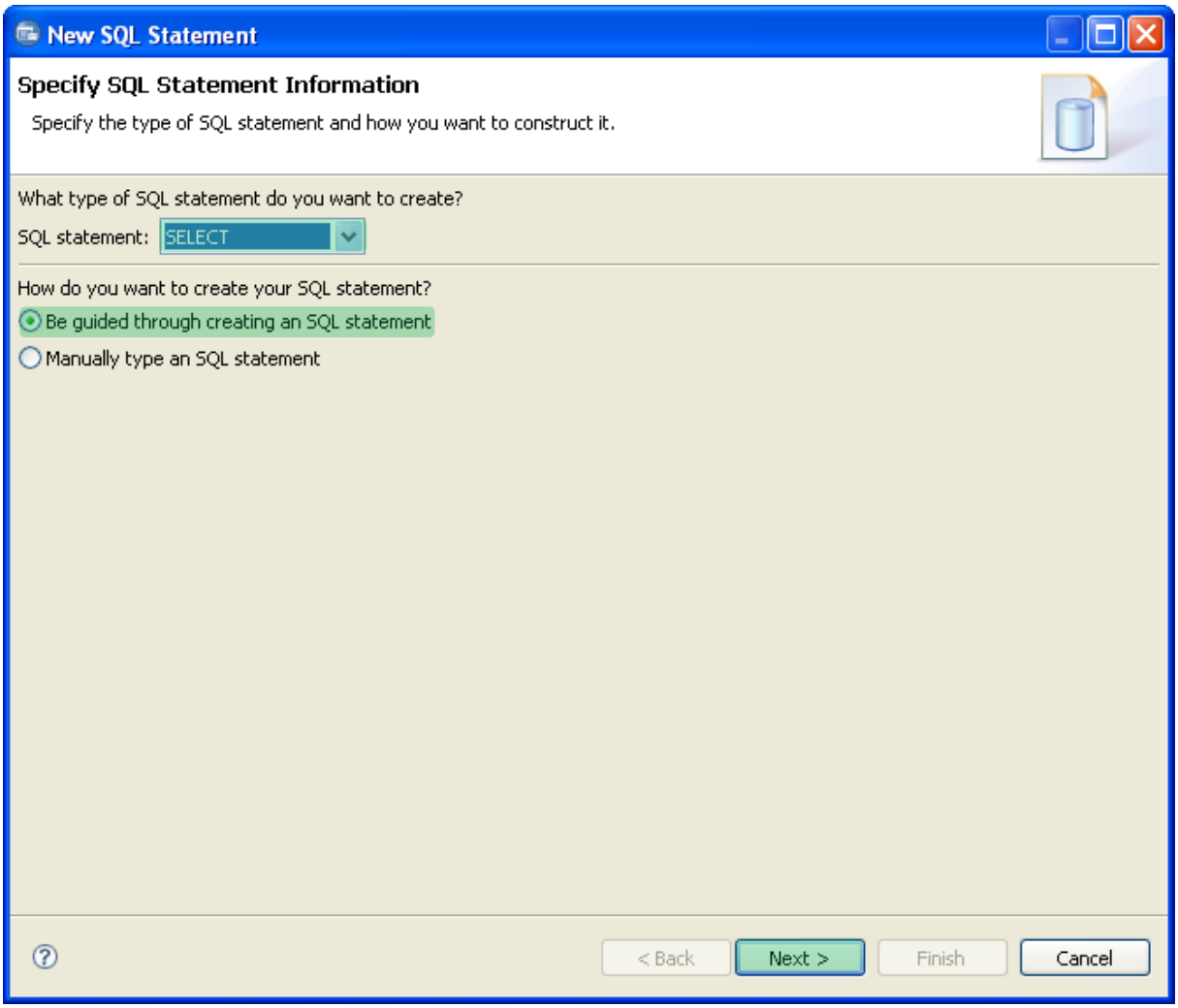


Der IBM Data Studio Developer ermöglicht es, Native SQL Stored Procedures, External SQL Stored Procedures, sowie Java Stored Procedures über JDBC oder SQLJ zu erstellen. Wählen Sie „SQL – native“ aus und geben Sie als Namen für die Stored Procedure „TABELLE“ an. Anschließend klicken Sie auf „Next“.



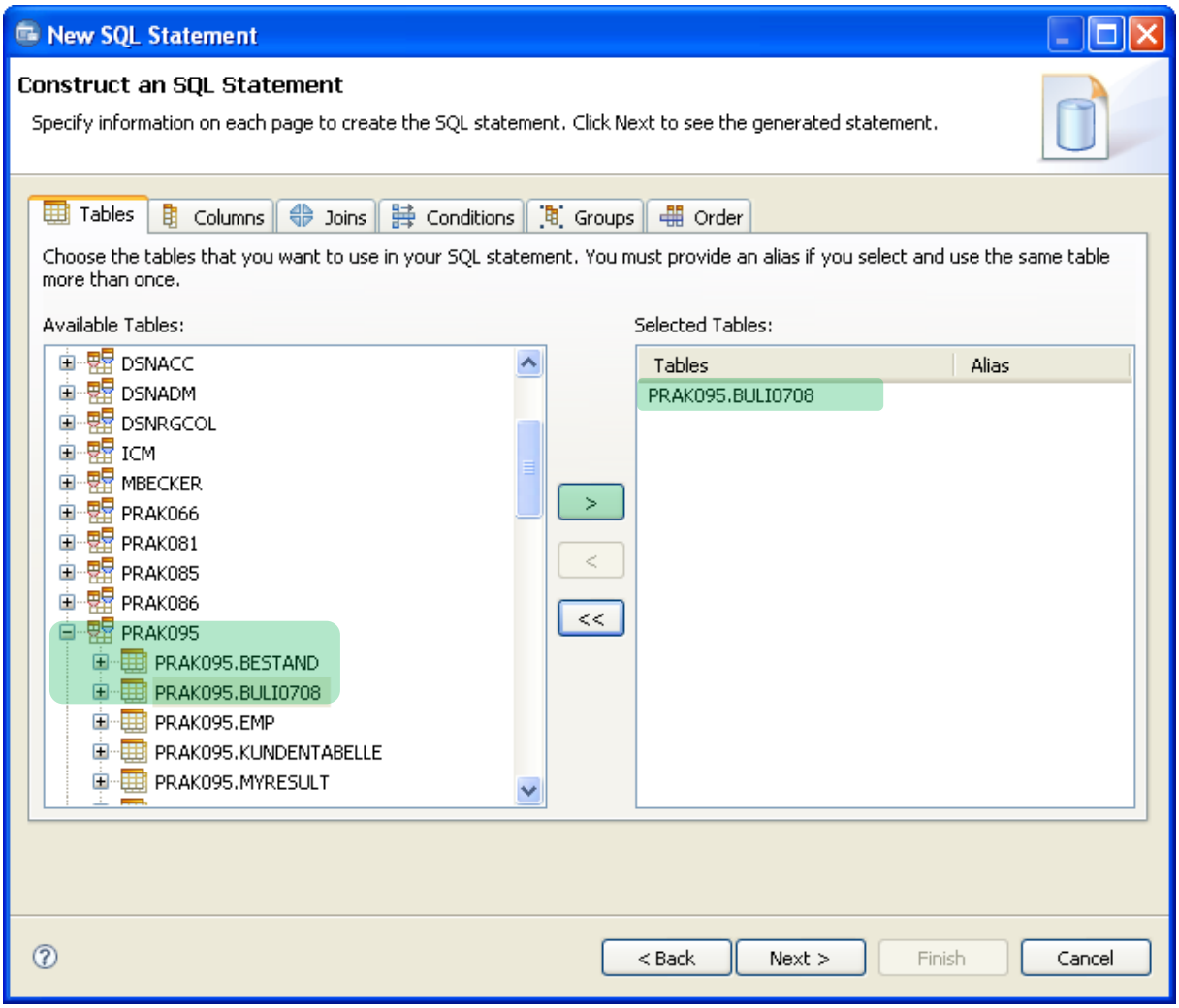
Nun haben Sie die Möglichkeit, über den Knopf „Create SQL…“ den Quelltext der Stored Procedure per Maus zusammenzuklicken. Dieses Tool ist für SQL-Neulinge eine große Hilfe, deckt allerdings nicht alle SQL-Befehle ab und ist bei komplizierten Anweisungen sehr zeitintensiv. Für die erste Stored Procedure soll es verwendet werden. Stellen Sie ein, dass die Stored Procedure eine Ergebnismenge zurückliefern soll, indem Sie im Feld „Result set“ „One“ auswählen und klicken danach auf „Create SQL“.


New Stored Procedure
SQL Statements
Create a new SQL statement or import an SQL statement that is saved in your project.
Statements:
Statement1
Statement details:
SELECT SCHEMA, NAME FROM SYSIBM.SYSROUTINES
Add...
Remove
Import...
Show All...
Create SQL...
Validate
Actual Cost
Result set:
One
< Back
Next >
Finish
Cancel



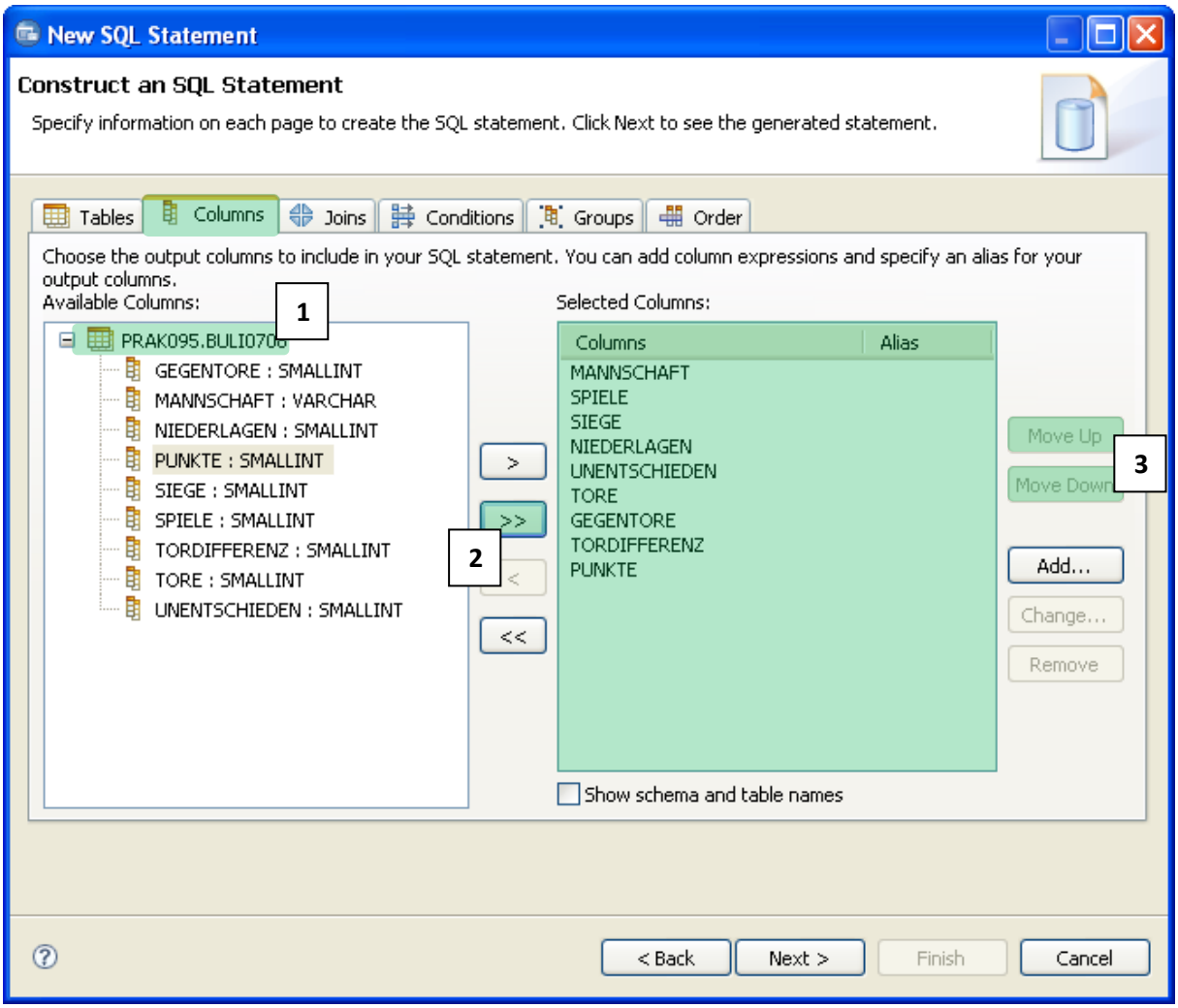
Nun haben Sie die Möglichkeit, einen Befehlstyp auszuwählen. Da eine SELECT-Anfrage an die Bundesligatabelle gestellt werden soll, wird „SELECT“ ausgewählt. Stellen Sie außerdem ein, dass der Befehl nicht per Texteingabe erstellt wird, indem Sie auf „Be guided through creating an SQL statement“ auswählen und klicken Sie auf „Next“.

Im folgenden Fenster können Sie die Tabellen auswählen, auf die sich der SELECT-Befehl bezieht. Da der Befehl nur die Tabelle BULI0708 verwendet, wählen Sie nur diese aus und bringen Sie sie durch einen Klick auf den Pfeilknopf in die Liste der ausgewählten Tabellen.

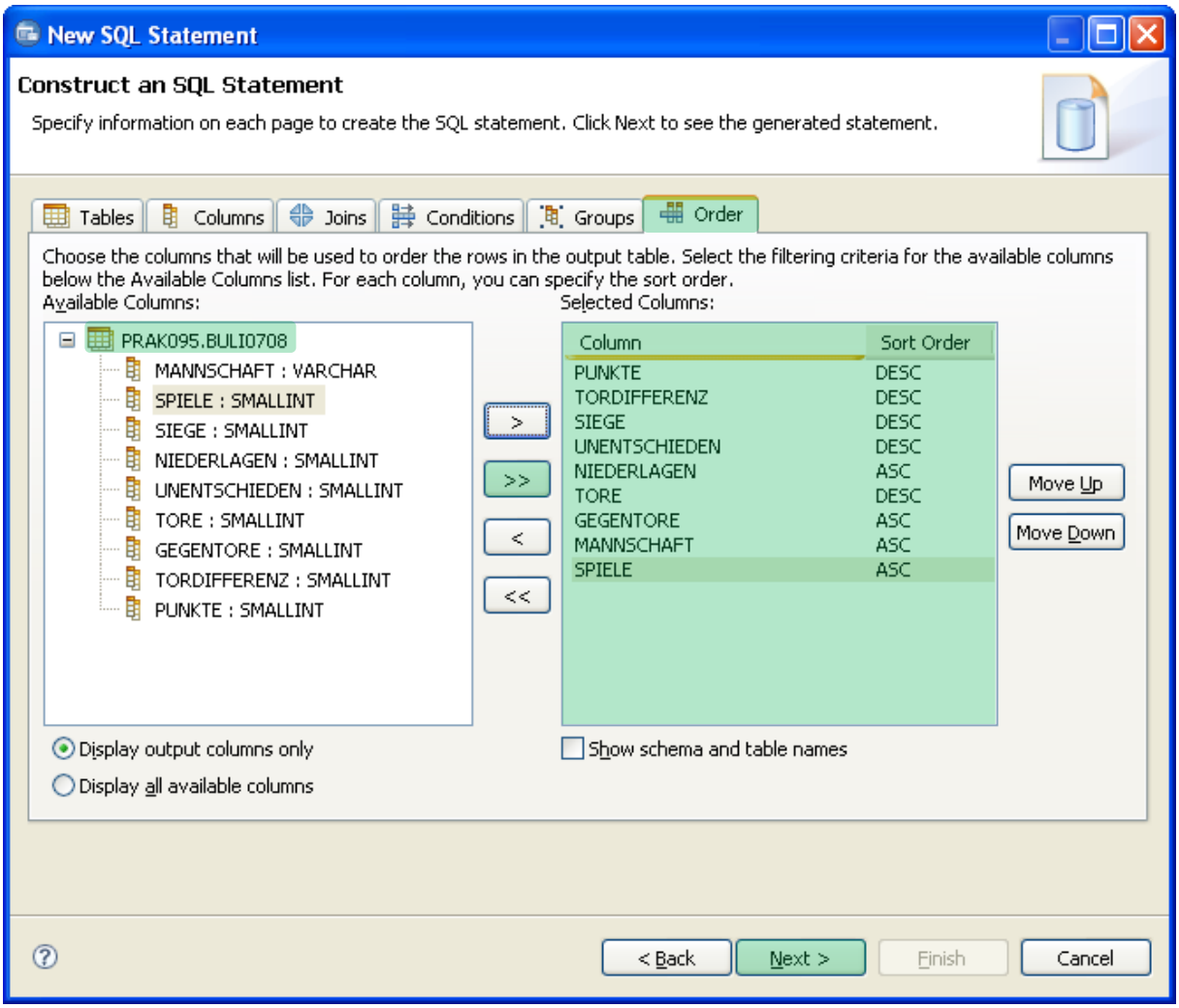


Im Register über den Tabellen können Sie nun im Menüpunkt „Columns“ die Spalten auswählen, die der SELECT-Befehl zurückgeben soll. Durch einen Klick auf den Doppelpfeilknopf fügen Sie alle Spalten in die Liste der ausgewählten Spalten. Mit den Knöpfen „Move Up“ und „Move Down“ können Sie die Reihenfolge der Spalten in der Ausgabe festlegen. Wählen Sie folgende Reihenfolge:

Mannschaft
Spiele
Siege
Niederlagen
Unentschieden
Tore
Gegentore
Tordifferenz
Punkte

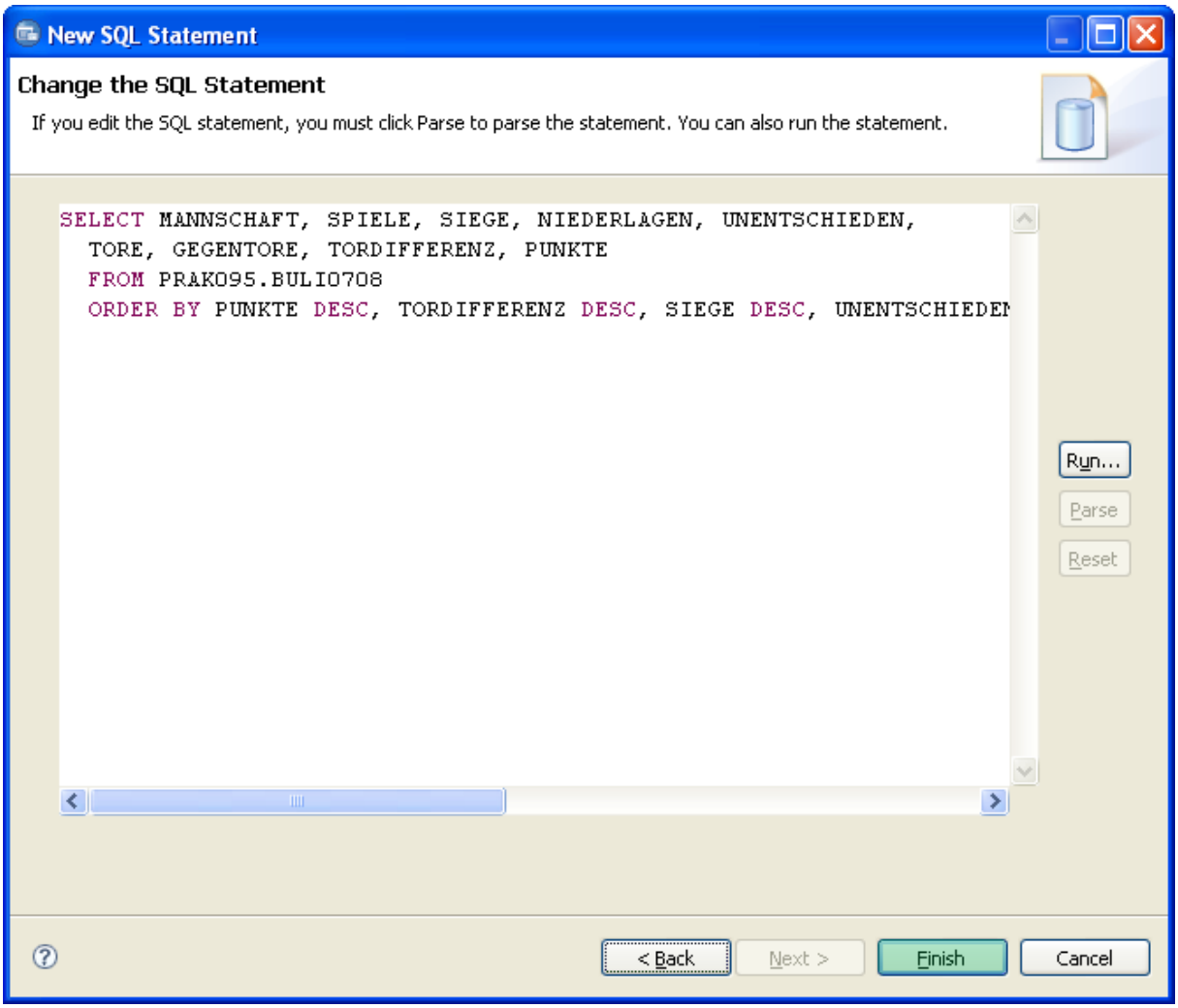


Da ein sehr einfacher Befehl verwendet wird, in dem kein Join, keine Bedingung und auch keine Gruppierung auftaucht, können Sie nun auf den Menüpunkt „Order“ klicken um die Sortierung der Ausgabe festzulegen. Nachdem Sie die Spalten erneut durch den Doppelpfeil in die Liste der ausgewählten Spalten kopiert haben, können nun durch einen Klick auf „Move Up“/“Move Down“ die Reihenfolge der Attribute für die Sortierung und durch einen Klick auf den Eintrag in der Spalte „Sort Order“ die Sortierung (absteigend oder aufsteigend) festgelegt werden. Lassen Sie die Spalten so aufsteigend/absteigend sortieren, dass Sie eine üblich sortierte Fußballtabelle erhalten.

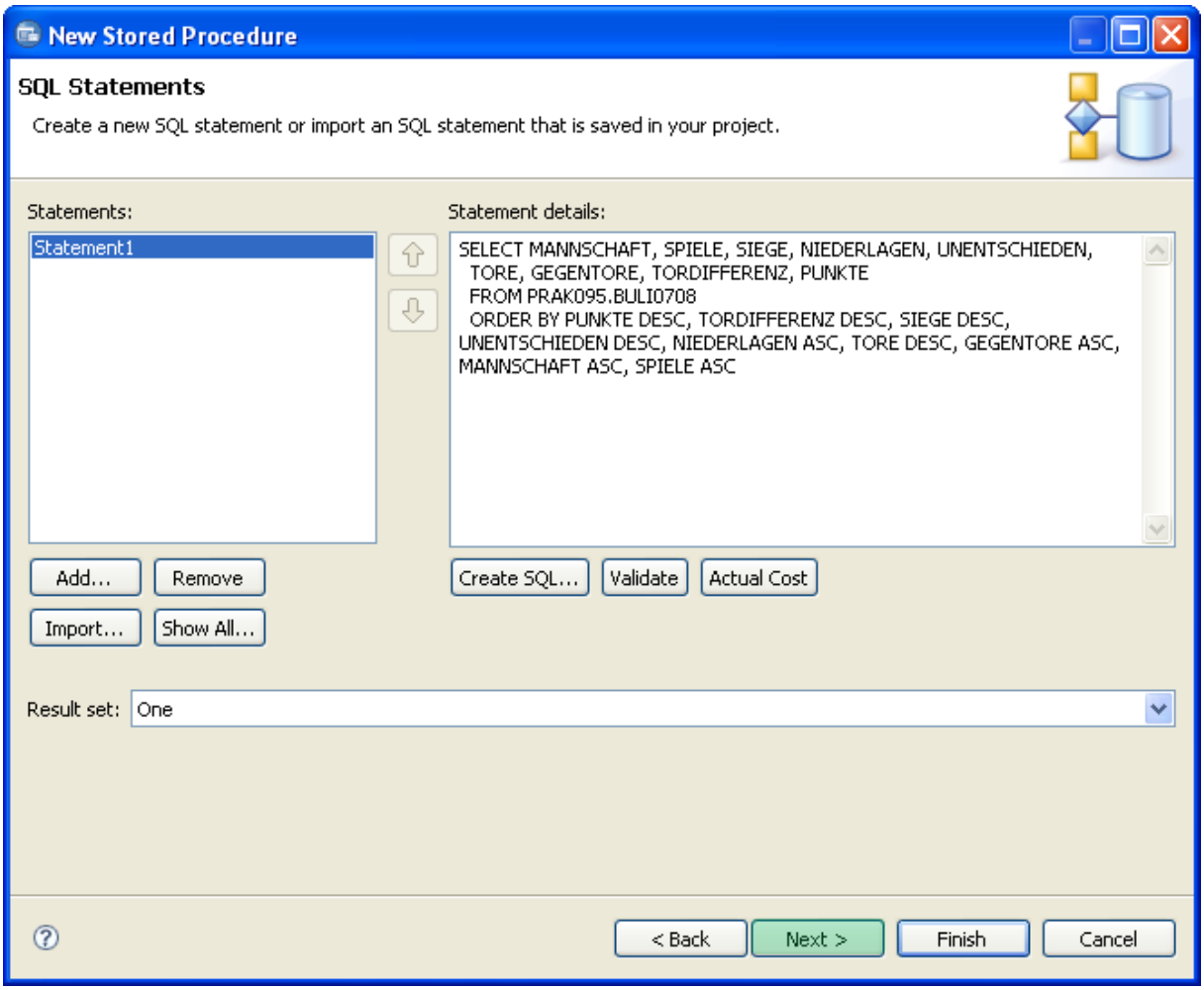


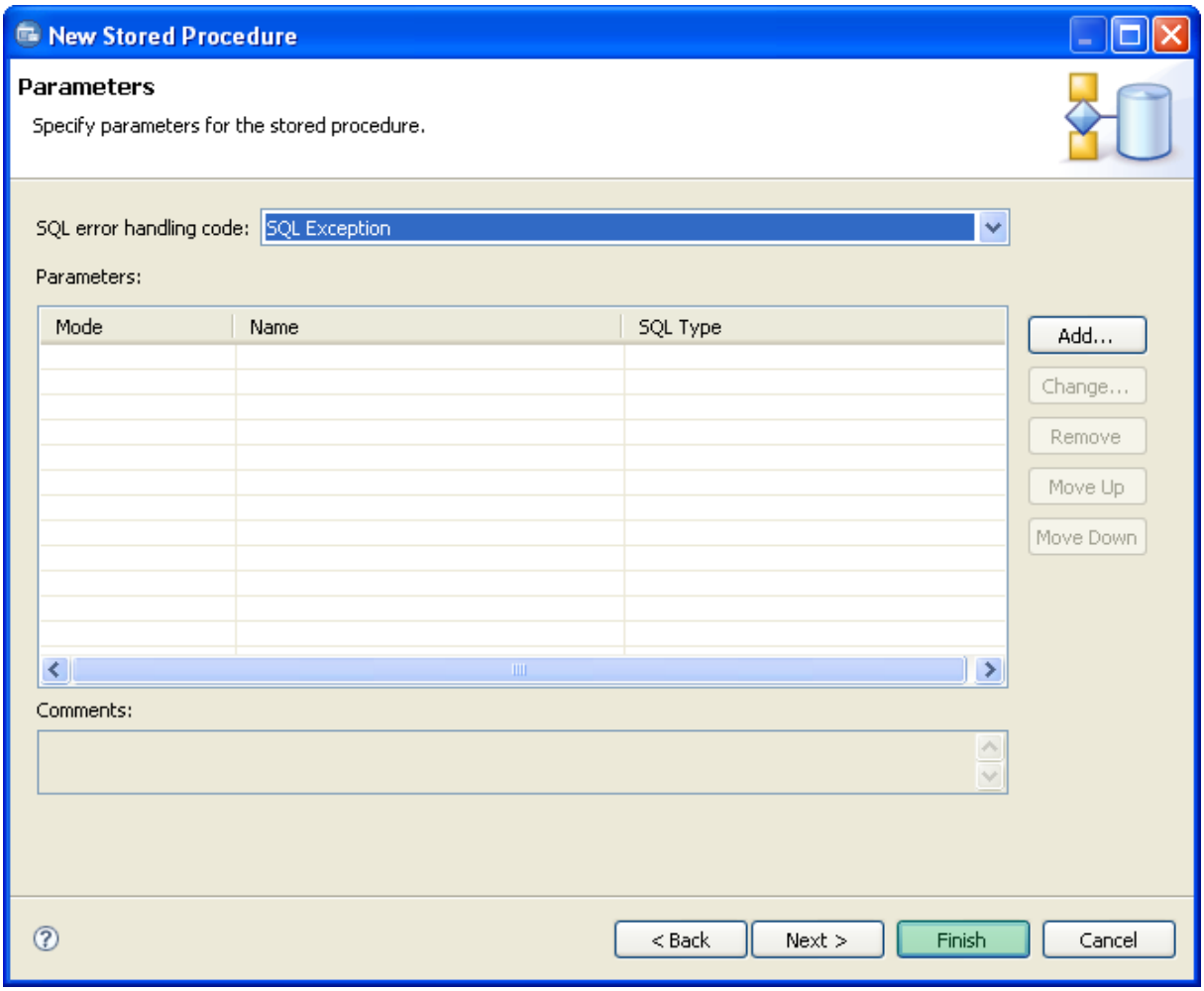
Mit einem Klick auf „Next“ erhalten Sie eine Vorschau des SQL-Befehls angezeigt, so wie Sie ihn zusammengestellt haben.

Sollten hier noch Änderungen per Texteingabe nötig sein, ist es empfehlenswert, die Richtigkeit des Befehls durch einen Klick auf „Run“ zu überprüfen. Da momentan keine Änderung am Befehl benötigt wird, klicken Sie auf „Finish“.



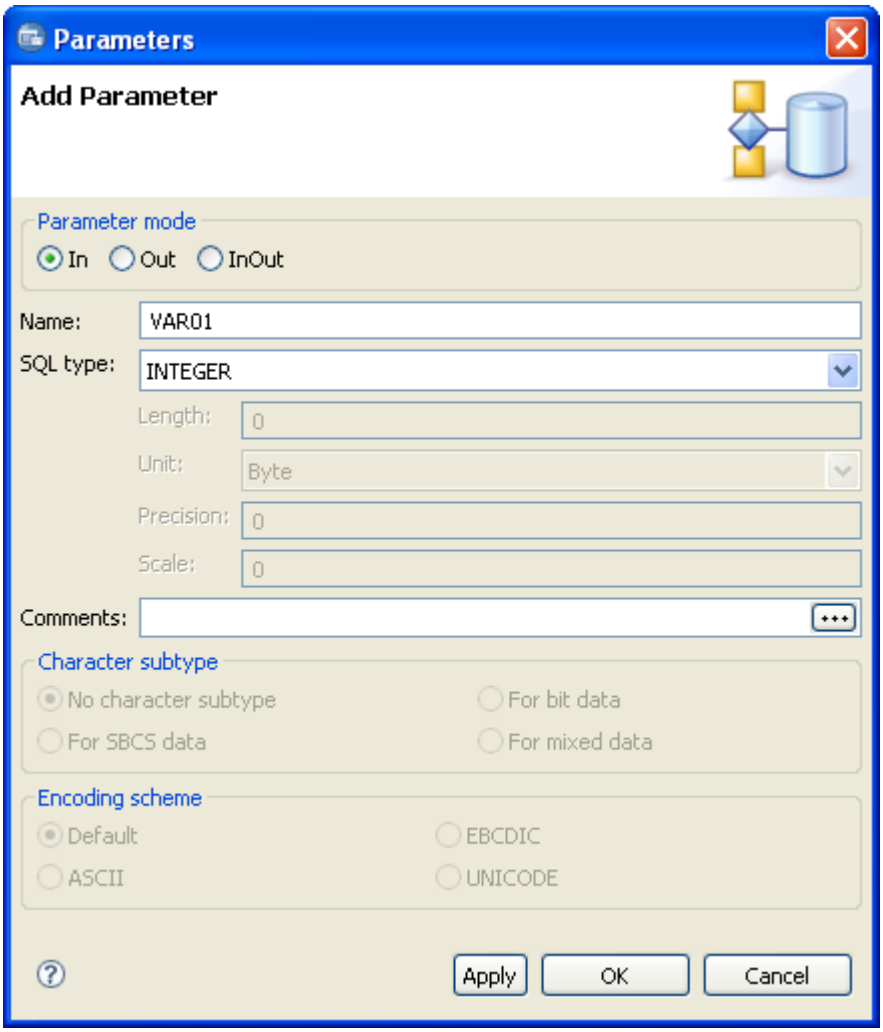
Sie bekommen nun wieder das ursprüngliche Fenster angezeigt, diesmal mit dem von Ihnen erstellten SELECT-Befehl im Feld „Statement Details“. Wenn Sie weitere Befehle anlegen wollen, können Sie dies hier durch einen Klick auf „Add“ machen. Da momentan aber kein weiterer Befehl benötigt wird, klicken Sie auf „Next“.

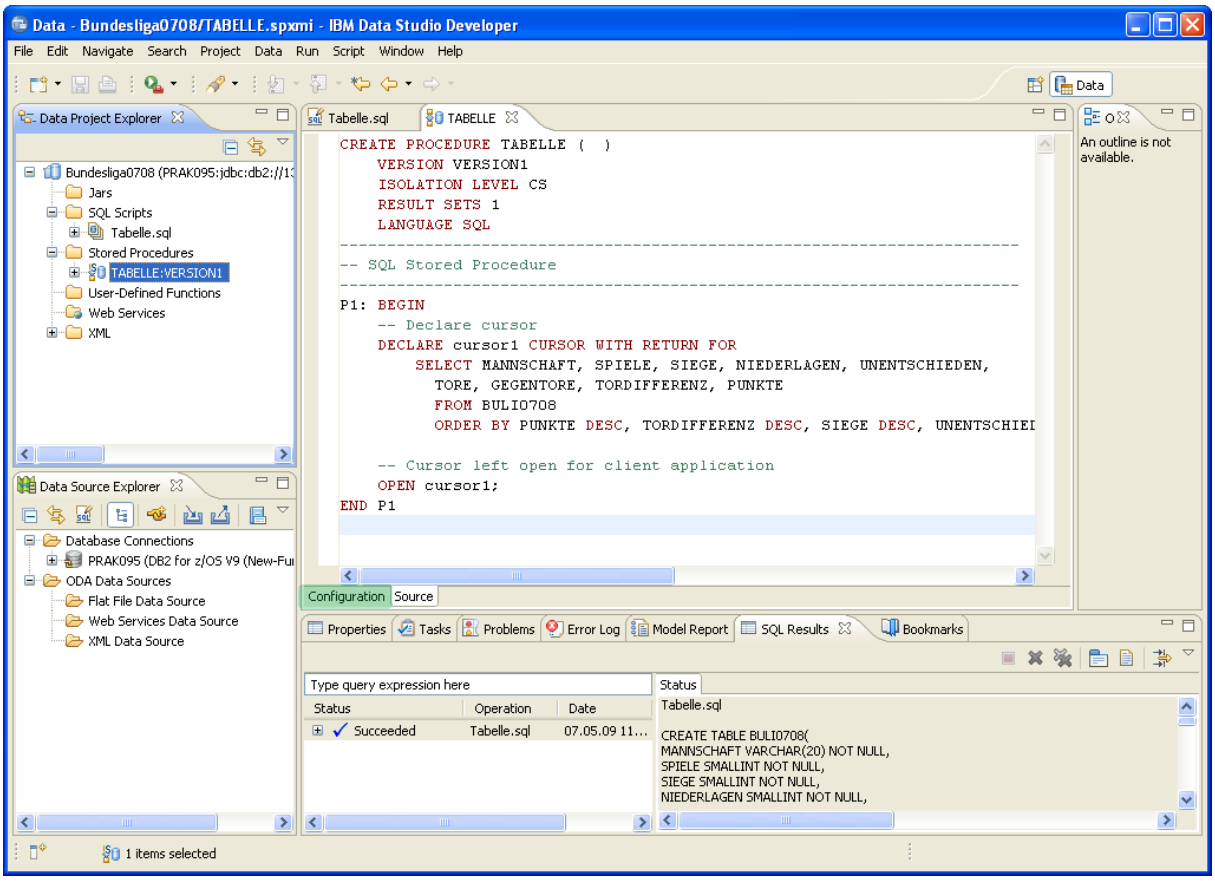

New Stored Procedure
SQL Statements
Create a new SQL statement or import an SQL statement that is saved in your project.
Statements:
Statement1
Statement details:
SELECT MANNSCHAFT, SPIELE, SIEGE, NIEDERLAGEN, UNENTSCHIEDEN,
TORE, GEGENTORE, TORDIFFERENZ, PUNKTE
FROM PRAK095.BULI0708
ORDER BY PUNKTE DESC, TORDIFFERENZ DESC, SIEGE DESC,
UNENTSCHIEDEN DESC, NIEDERLAGEN ASC, TORE DESC, GEGENTORE ASC,
MANNSCHAFT ASC, SPIELE ASC
Add...
Remove
Import...
Show All...
Create SQL...
Validate
Actual Cost
Result set:
One
< Back
Next >
Finish
Cancel



Im folgenden Dialog können Sie Eingabe- und Ausgabeparameter der Stored Procedure definieren. Durch einen Klick auf „Add“ öffnet sich ein neues Fenster, in dem Sie den Parametertyp, Namen und Datentyp angeben können. Für die aktuelle Stored Procedure benötigen Sie keine Ein- und Ausgabeparameter, weshalb der Dialog wieder geschlossen werden kann. Für spätere Stored Procedures wird dieser Schritt allerdings benötigt.

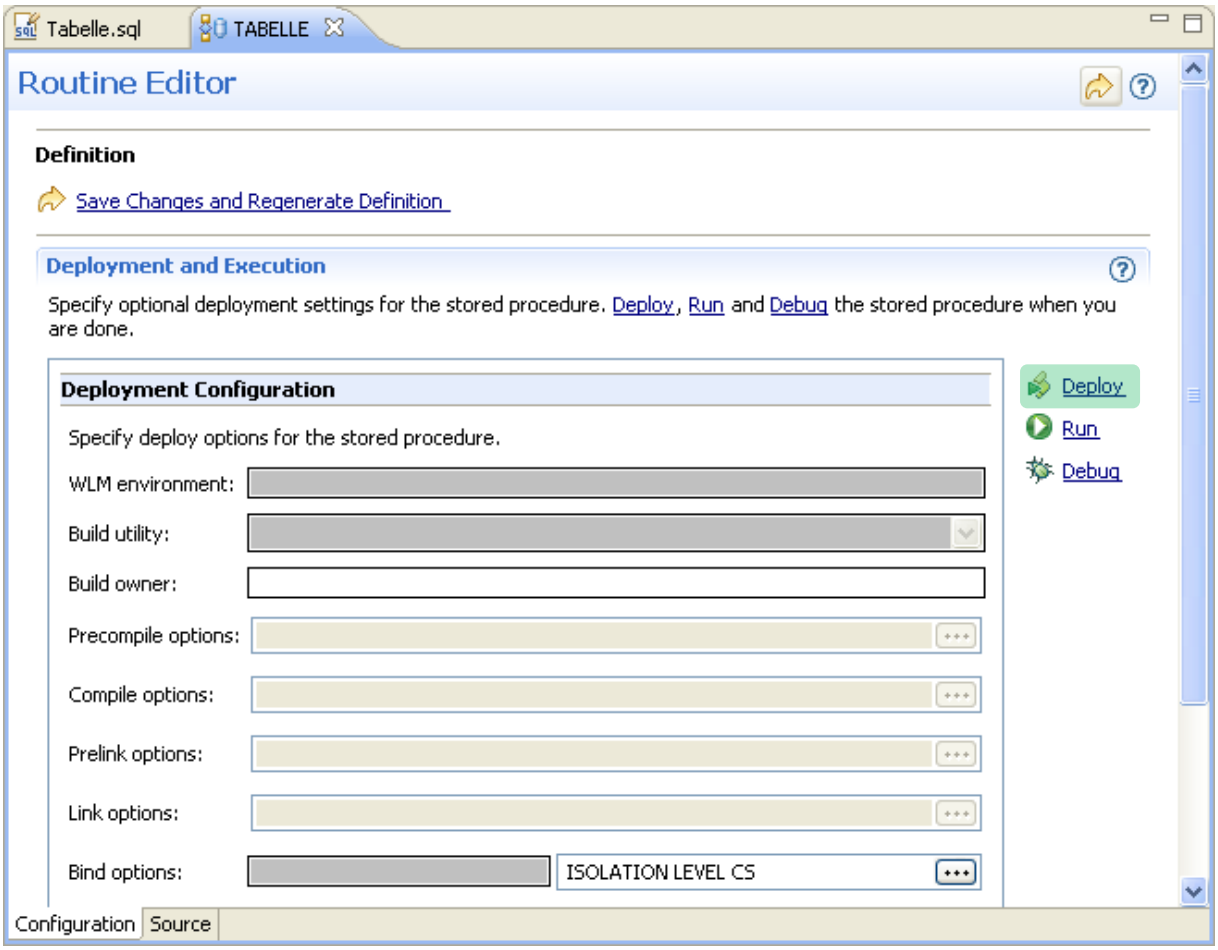
Klicken Sie auf „Finish“ um das Erstellen der Stored Procedure abzuschließen.


Parameters
Add Parameter
Parameter mode
In
Out
InOut
Name:
VAR01
SQL type:
INTEGER
Length:
0
Unit:
Byte
Precision:
0
Scale:
0
Comments:
Character subtype
No character subtype
For bit data
For SBCS data
For mixed data
Encoding scheme
Default
EBCDIC
ASCII
UNICODE
Apply
OK
Cancel



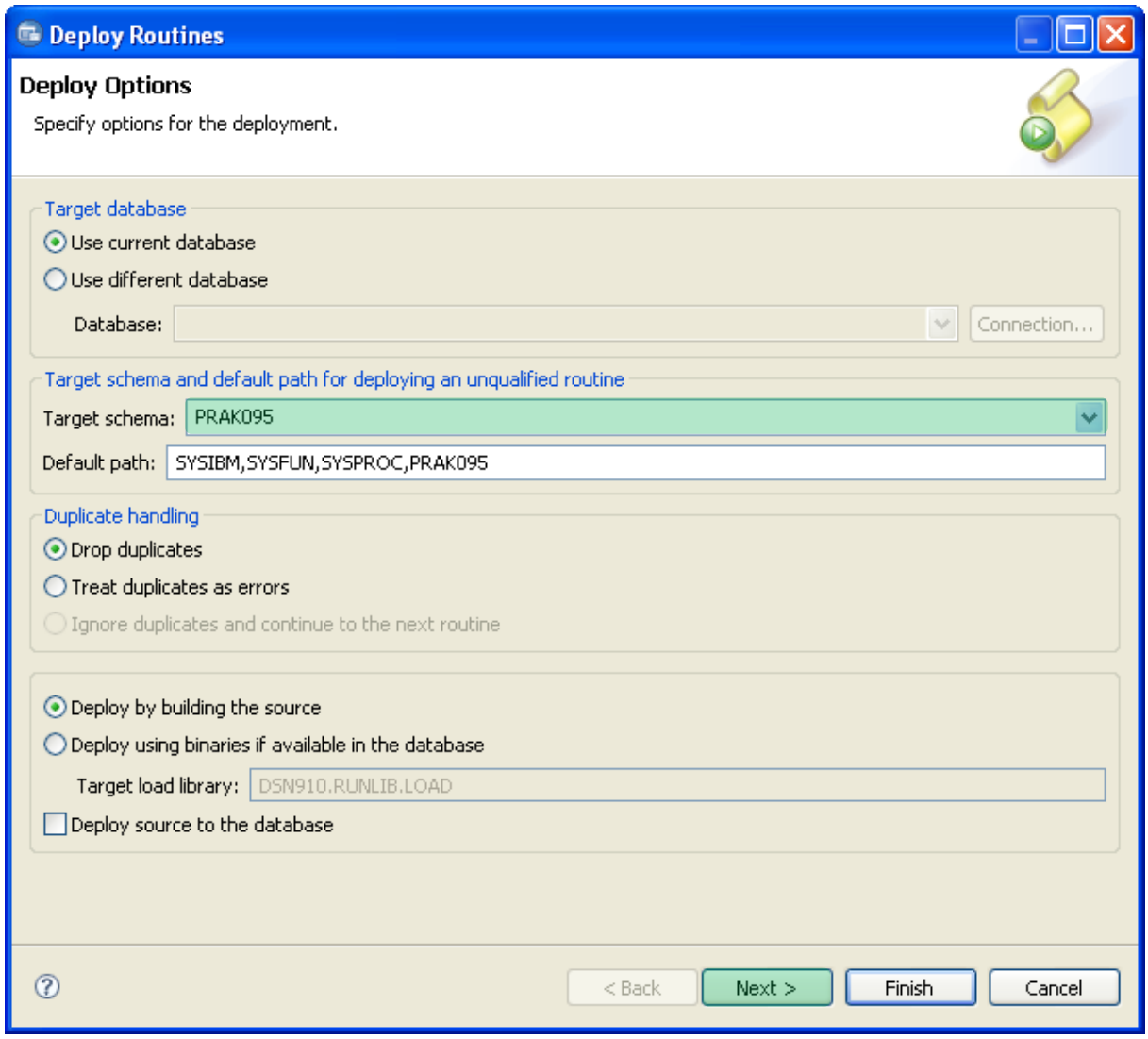
IBM Data Studio Developer erstellt nun automatisch den kompletten Code der Stored Procedure. Bei Native SQL Stored Procedures sind DDL und Logik der Stored Procedure im CREATE PROCEDURE-Befehl vereint. Im oberen Teil des Codes erkennt man den DDL-Teil, der mit CREATE PROCEDURE beginnt. Außer dem Namen und der Parameter beinhaltet die DDL zusätzliche Informationen für die Ausführung der Stored Procedure. In vorliegenden Fall beinhaltet sie eine Versionsnummer, die Anzahl der Rückgabemengen und einer Angabe zu einem Isolation Level. Diese Angabe ist für Stored Procedures wichtig, die transaktionale Eigenschaften besitzen. Durch die Wahl des Isolation Levels ist festgelegt, welche Befehle in Stored Procedures parallel zu anderen SQL-Befehlen (z.B. parallel ablaufender Stored Procedures) ausgeführt werden dürfen. DB2 unterscheidet hierfür vier verschiedene Isolation Level. Weitere Informationen zu Isolation Levels finden Sie unter http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.perf/db2z_chooseisolationoption.htm. Der Teil ab P1 ist der Code, der beim Aufruf der Stored Procedure ausgeführt werden wird.

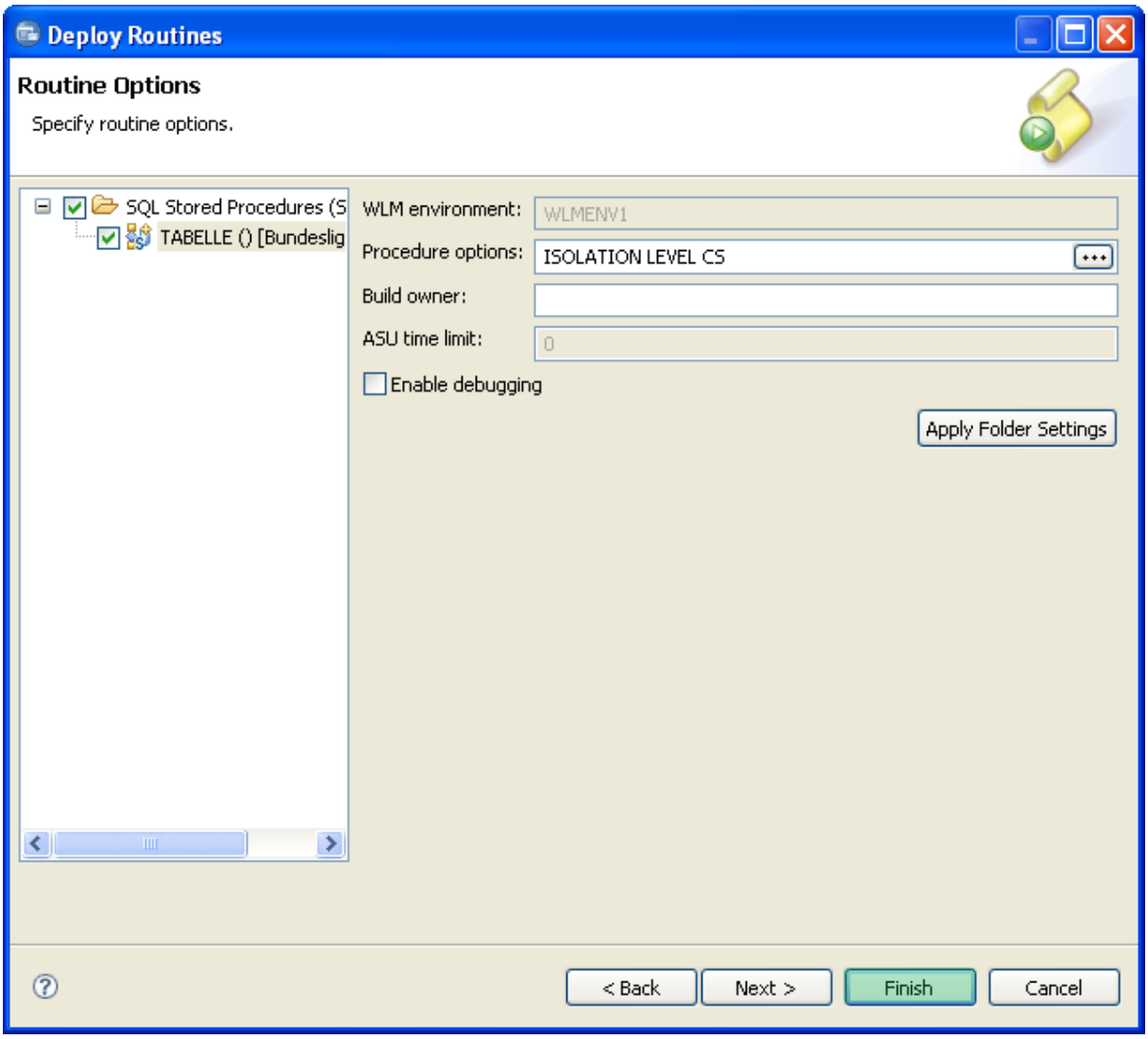
Klicken Sie auf „Configuration“ im unteren Bereich der Stored Procedure.

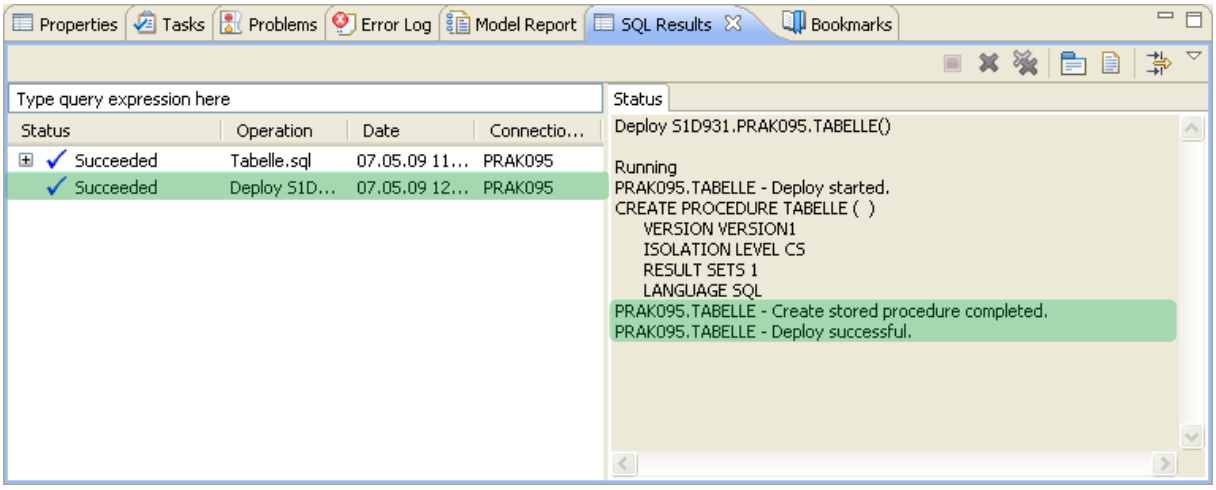


Um die Stored Procedure auf dem entfernten DB2-System einzurichten, muss der SQL-Code nun in eine interne Repräsentation übersetzt, zu einem Paket (Package) gebunden und im DB2-System gespeichert werden. Außerdem müssen Informationen zur Stored Procedure aus der DDL (Name, Parameteranzahl und Typ, Sprache, Name des Packages, Ersteller, …) in Katalogtabellen gespeichert werden. Beim Aufruf wird dann anhand dieser Tabellen überprüft, ob die Zahl und der Typ der Parameter korrekt sind und die richtige Stored Procedure ausgeführt. Um diesen Vorgang einzuleiten, klicken Sie auf „Deploy“ im Menü auf der rechten Seite

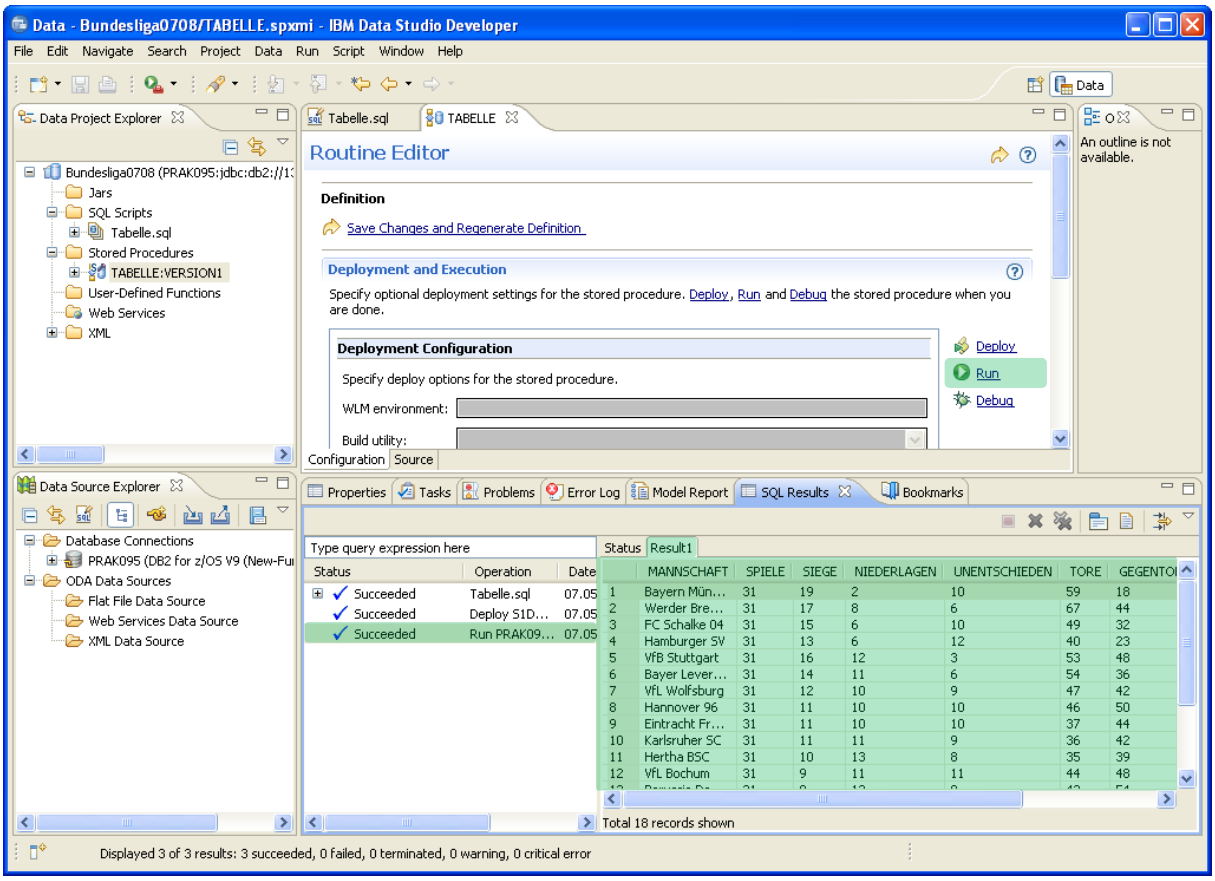
Sie haben nun die Möglichkeit, Details für das Portieren auf den Mainframe anzugeben. Stellen Sie sicher, dass als Schema Ihre Benutzerkennung eingetragen ist, klicken Sie auf „Next“ und im folgenden Fenster auf „Finish“.


Deploy Routines
Deploy Options
Specify options for the deployment.
Target database
Use current database
Use different database
Database:
Connection...
Target schema and default path for deploying an unqualified routine
Target schema: PRAK095
Default path: SYSIBM,SYSFUN,SYSPROC,PRAK095
Duplicate handling
Drop duplicates
Treat duplicates as errors
Ignore duplicates and continue to the next routine
Deploy by building the source
Deploy using binaries if available in the database
Target load library: DSN910.RUNLIB.LOAD
Deploy source to the database
< Back
Next >
Finish
Cancel





Ein erfolgreiches Einrichten der Stored Procedure im DB2-System auf den Mainframe wird im SQL Results-Fenster angezeigt. Sie können nun die Stored Procedure durch ein SQL-Skript aufrufen, das den CALL-Befehl „CALL TABELLE()“ enthält. Der IBM Data Studio Developer stellt hierfür eine schnellere Methode zur Verfügung: Sie müssen lediglich auf „Run“ im Configuration-Panel klicken, um die Stored Procedure auszuführen. Das Ergebnis des Aufrufs befindet sich danach im Register SQL Results im Unterregister Result1.



Aufgabe: Schreiben Sie eine Stored Procedure mit Namen TEAMS, die alle Teams der Tabelle BULI0708 zurückgibt, sortiert nach dem Namen. Verwenden Sie dazu den SQL-Builder.

Die bisherigen Stored Procedures hatten keine Eingabe-/Ausgabeparameter und haben sich auf einen einfachen SELECT-Befehl beschränkt. Durch den Einsatz der SQL Procedural Language kann man aber eine Stored Procedure beliebig kompliziert machen.

Wichtige SQL PL-Sprachelemente

1. **Kommentare**

   Kommentare können mit ‚--' eingeleitet oder zwischen /* und */ geschrieben werden.

2. **BEGIN … END (compound statement)**

   Enthält einen oder mehrere Befehle aus dieser Liste. Die Reihenfolge, in der die Befehle stehen müssen ist:
   1. SQL Variablen, Deklarationen von Bedingungen und Returncodes
   2. Deklarationen von Cursor
   3. Deklarationen von Handlern
   4. SQL-Procedure-Befehle

3. **DECLARE**

   Mit einem DECLARE-Befehl können Variablen, Cursor und Handler deklariert werden. Die verwendeten DECLARE-Befehle müssen dabei immer zu Beginn eines compound statements stehen. Für Variablennamen gelten folgenden Einschränkungen:
   Die Länge der Variablennamen darf 128Byte nicht überschreiten
   Es wird nicht zwischen Groß- und Kleinbuchstaben unterschieden
   Variablennamen dürfen nicht identisch mit Parameternamen sein
   Variablennamen dürfen nicht mit einem Doppelpunkt beginnen
   Beispiel:
   DECLARE COUNTER INT DEFAULT 0;
   -- Definiert eine Variable COUNTER mit Initialwert 0.

   Cursor dienen dazu, eine Ergebnismenge zu durchschreiten. Mit folgendem Befehl wird ein Cursor mit Namen <cursornamen> für den SELECT-Befehl <selectstatement> erstellt:
   DECLARE <cursorname> CURSOR FOR <selectstatement>.
   Um auf einzelne Tupel der Ergebnismenge zugreifen zu können, muss der Cursor mittels OPEN geöffnet werden. Mit einem FETCH-Befehl können dann Tupel ausgelesen werden. Im Bezug auf Stored Procedures ist dabei darauf zu achten, dass für jeden offenen Cursor eine Ergebnisrelation aus den noch nicht gelesenen Tupeln erstellt wird.
   Cursor lassen sich mit CLOSE <cursorname> schließen.
   Beispiel:
   DECLARE C1 CURSOR FOR SELECT MANNSCHAFT FROM BULI0708;
   FETCH C1 INTO myMannschaft;
   CLOSE C1;

   Tritt bei der Ausführung einer Stored Procedure ein Fehler auf, so bricht die Stored Procedure ab, es sei denn, ein Handler wurde definiert. Dieses Konzept ist vergleichbar mit Exceptions in der Programmiersprache Java. Handler lassen sich für folgende Ereignisse definieren:
   - SQL error
   - SQL warning
   - Leere Mengen bei Anfragen
   - Spezielle SQLSTATEs

Der Befehl um einen Handler zu erstellen, sieht typischerweise so aus:
DECLARE <handler-type> HANDLER FOR <condition> <SQL-procedure-statement>;
Beispiel:

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION
      L1: LOOP
        stmt1;
        ...
        LEAVE L1;
      END LOOP L1;
```

4. SET

Der SET-Befehl weist einer Variablen einen bestimmten Wert zu. Er wird mit SET eingeleitet.
Beispiel:

```
SET A = 1;
SET ERROR_MSG = ‚Ein Fehler ist aufgetreten‘;
```

5. CALL

Mit dem CALL-Befehl ruft man eine Stored Procedure auf. Hierbei spielt es keine Rolle, in welcher Programmiersprache diese geschrieben wurde.
Beispiel:

```
CALL TABELLE();
```

6. CASE

Der CASE-Befehl steuert einen Ausführungspfad in Abhängigkeit einer oder mehrerer Bedingungen.
Beispiel:

```
CASE RANK
WHEN ‚MANAGER‘ THEN
        SET SALARY = 50000;
WHEN ‚SPECIALIST‘ THEN
        SET SALARY = 30000;
ELSE
        SET SALARY = 15000;
END CASE;
```

7. GOTO

Führt einen Sprung zu einem vom Benutzer erstellten Label durch
Beispiel:

```
START:
…
CASE TMPVAR
WHEN 1
        GOTO START;
END CASE;
```

8. IF

Verzweigt den Ausführungspfad in Abhängigkeit von der Auswertung einer Bedingung
Beispiel:

```
IF TMPVAR = 1 THEN GOTO START;
        ELSE GOTO END;
END IF;
```

9. LEAVE

Springt aus einer Schleife.
Beispiel s. LOOP.

10. LOOP

Führt einen Befehl/mehrere Befehle mehrfach aus.
Beispiel:

```
OPEN C2;
GETEACH: LOOP
        FETCH C2 INTO MYEMPNO, MYSALARY;
         IF  SQLCODE = 100 THEN LEAVE GETEACH;
        END IF;
END LOOP;
```

11. REPEAT

Führt einen Befehl/mehrere Befehle so lange aus, bis eine Bedingung erfüllt ist.
Beispiel:

```
SET I=1;
DOIT: REPEAT
SET I = I+1;
UNTIL I>5
END REPEAT DOIT;
```

12. WHILE

Führt einen Befehl/mehrere Befehle so lange aus, bis eine Bedingung erfüllt ist.
Beispiel:

```
SET I=1;
WHILE I<6
DO
SET I = I+1;
END WHILE;
```

13. ITERATE

Führt einen Sprung zum Anfang einer Schleife aus. Ist deshalb nur erlaubt in einem LOOP, REPEAT oder WHILE-Befehl.
Beispiel:

```
SET COUNTER = 0;
GETIT: LOOP
        COUNTER = COUNTER +1;
        IF COUNTER = 5 THEN LEAVE GETIT;
        END IF;
        ITERATE GETIT
END LOOP;
```

14. SIGNAL

Erzeugt eine Fehler- oder Warnmeldung.
Beispiel:

```
DECLARE EXIT HANDLER FOR SQLSTATE VALUE ‘57011’
SIGNAL SQLSTATE ‘57001’
SET MESSAGE_TEXT = ‘EIN FEHLER IST AUFGETRETEN’;
```

Wenn nun ein Fehler mit SQLSTATE 57001 auftritt, wird der EXIT-Handler aufgerufen und dieser gibt folgende Meldung an die aufrufende Anwendung zurück:
SQL0438N Application raised error with diagnostic text: "EIN FEHLER IST AUFGETRETEN". SQLSTATE=75001

15. RESIGNAL

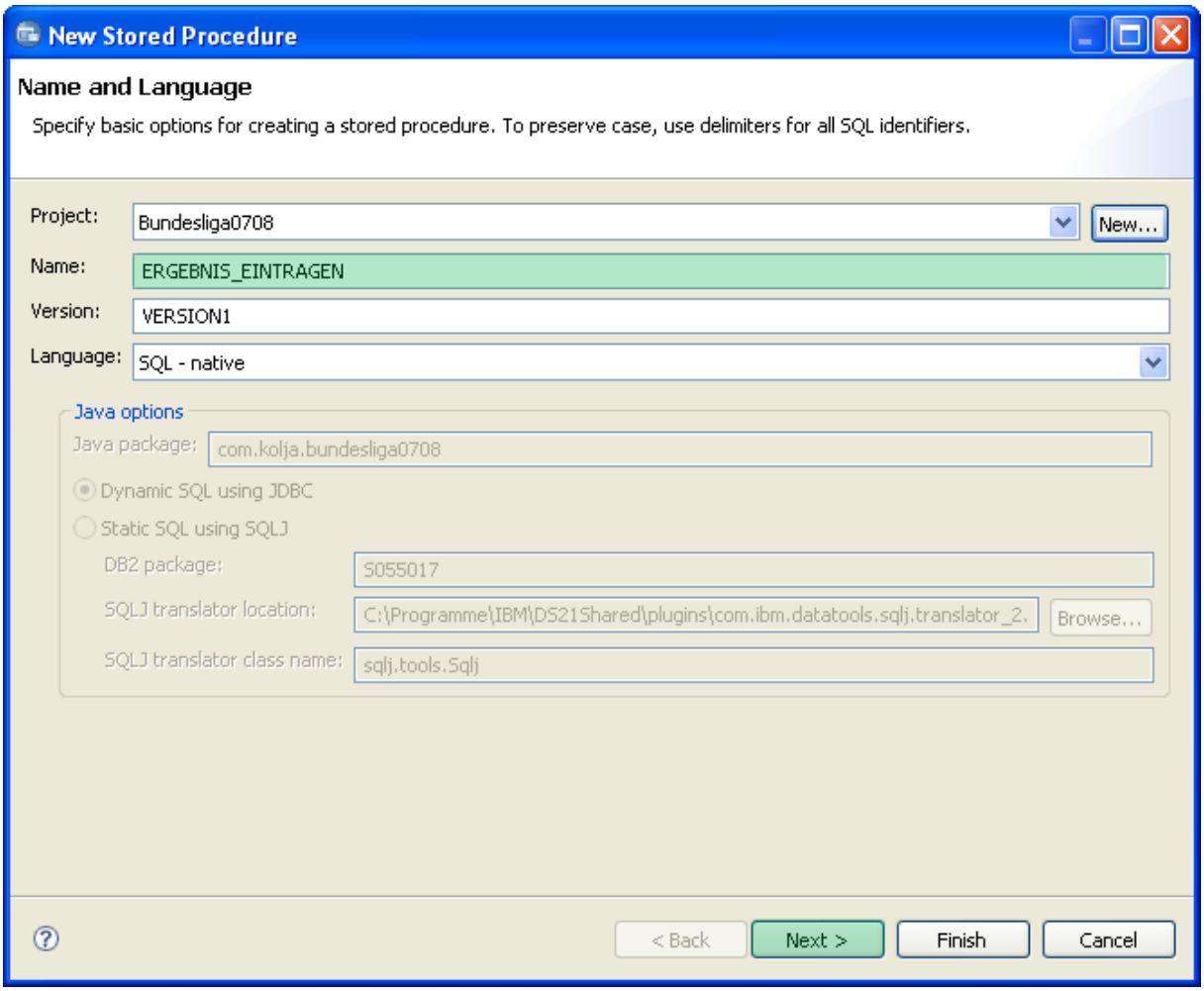
Gleiche Funktionalität wie SIGNAL. Kann im Gegensatz zu SIGNAL nur in der Definition eines Handlers verwendet werden und ermöglicht es dann, einen anderen SQLCODE als den Ursprünglichen zurückzugeben.

16. RETURN

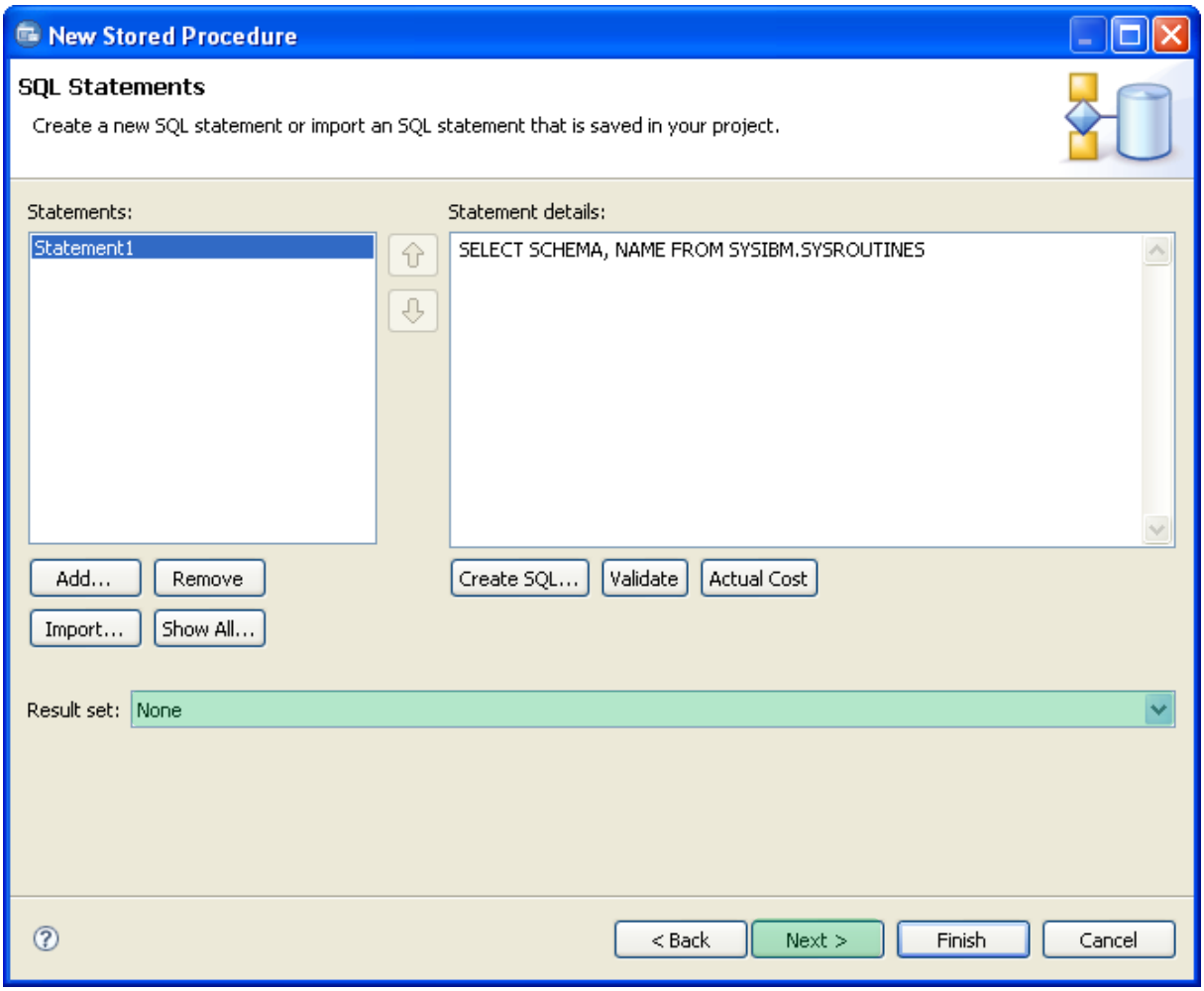
Wird als Rücksprung aus einer Routine verwendet. Ein Integerwert kann optional angegeben werden. Er wird dann als SQLCODE an die aufrufende Anwendung zurückgegeben.
Beispiel:

```
DECLARE ANY_ERRORS CHAR(1);
...
IF  ANY_ERRORS = 'Y' THEN RETURN 16;
                ELSE RETURN 0;
END IF;
```

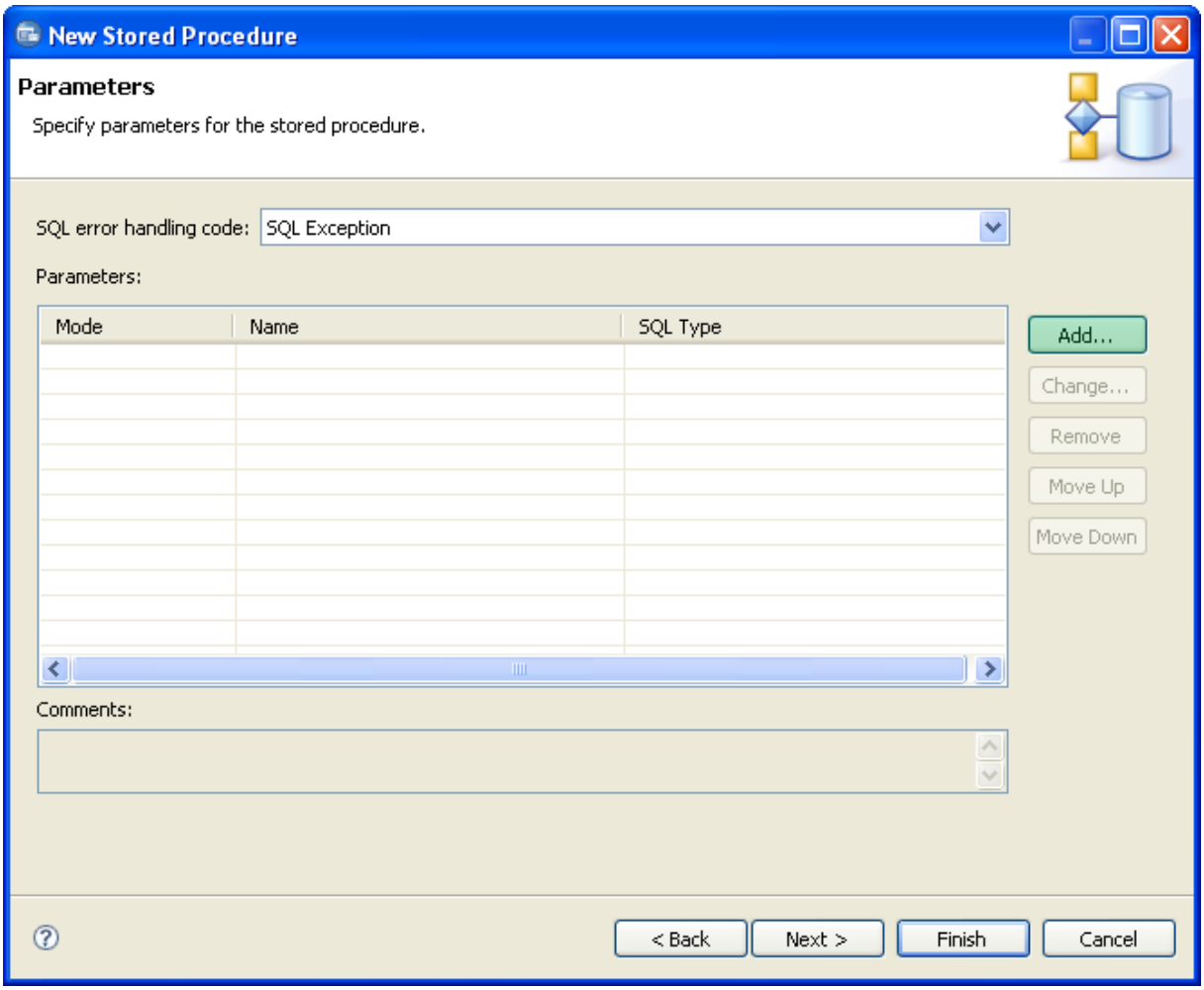
Im Folgenden soll nun eine Stored Procedure erstellt werden, die es ermöglicht, ein Spielergebnis in die Bundesligatabelle einzutragen. Die Stored Procedure benötigt 4 Eingabeparameter (Heimmannschaft, Gastmannschaft, Heimtore, Gasttore) und einen Ausgabeparameter (Status) und soll die Anzahl der Spiele, Siege, Niederlagen, Unentschieden, Tore, Gegentore und die Tordifferenz anpassen. Erstellen Sie also wieder eine Stored Procedure indem Sie mit der rechten Maustaste auf den Ordner Stored Procedures des Data Developement Projects klicken und über „New“ „Stored Procedure“ auswählen. Tragen Sie als Name „ERGEBNIS_EINTRAGEN“ ein und klicken Sie auf „Next“.



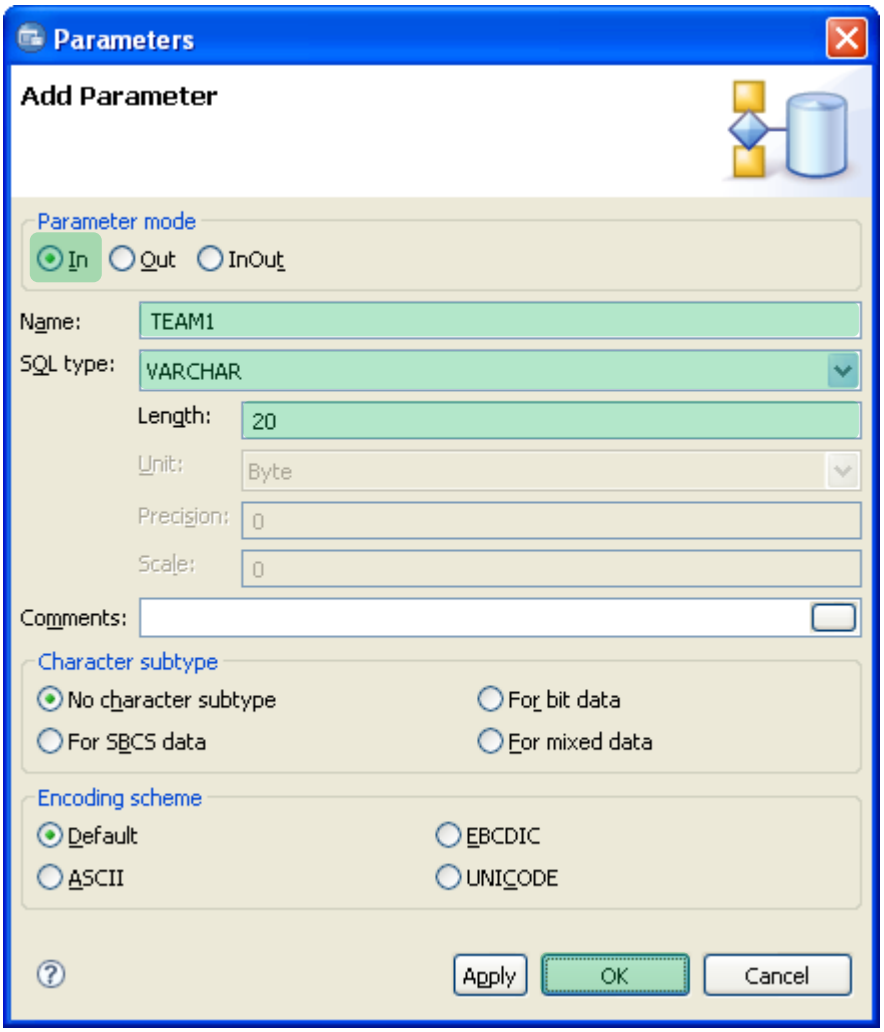
Da die Stored Procedure keine Tabelle zurückgeben wird, wählen Sie im Feld „Result set“ „None“ aus und klicken auf „Weiter“.



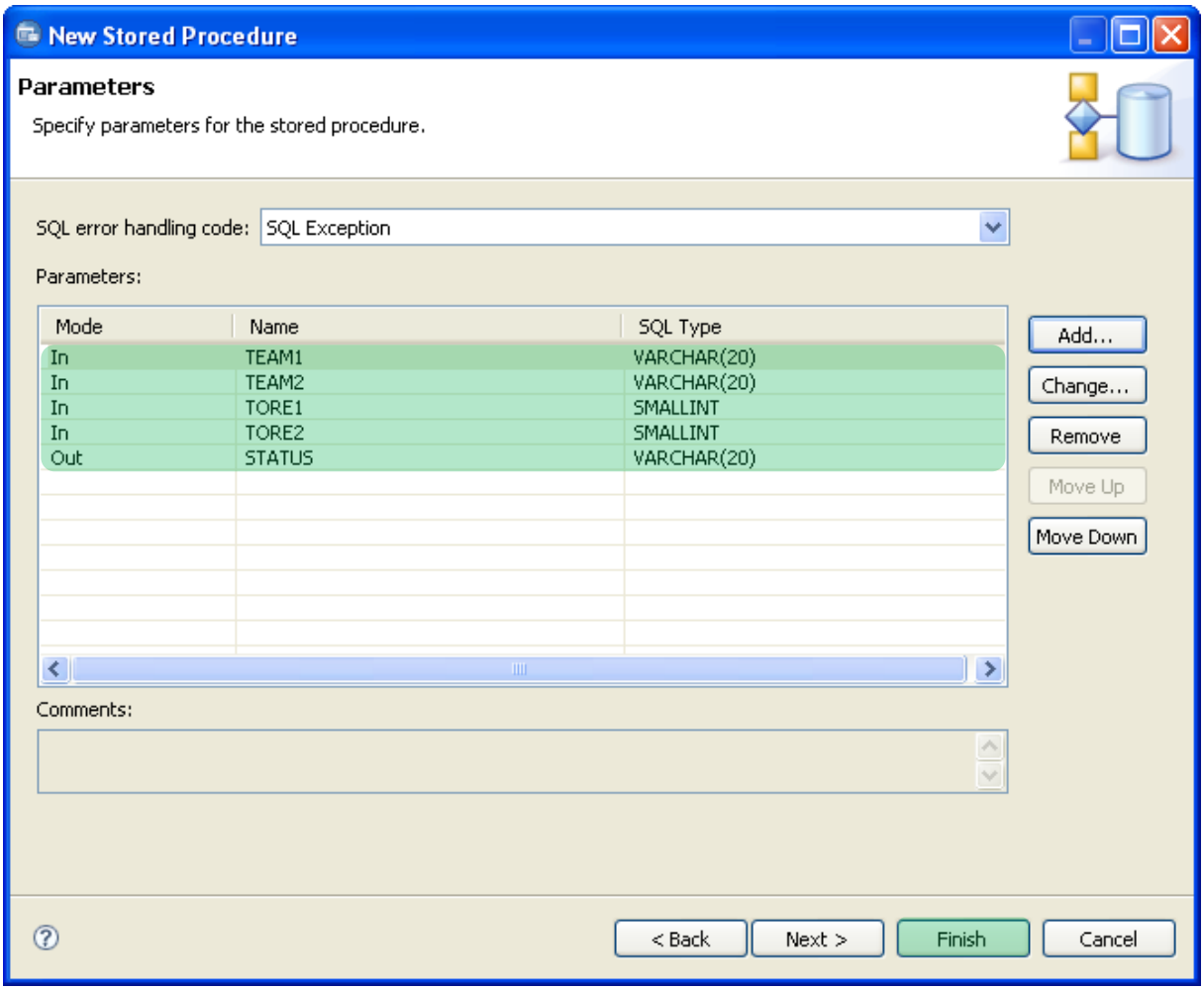
Die Befehle der Stored Procedure werden nachher selbst geschrieben (ohne den SQL-Builder). Klicken Sie deshalb nur auf „Next“.



Nun müssen der Stored Procedure Parameter hinzugefügt werden. Nach einem Klick auf „Add“, können Sie einen Eingabeparameter mit Namen TEAM1 und Typ VARCHAR(20) angeben. Durch einen Klick auf „OK“ bzw. „Apply“ wird der Parameter in die Liste der Parameter übernommen.



Fügen Sie außerdem einen Eingabeparameter TEAM2 vom Typ VARCHAR(20), zwei Eingabeparameter (TORE1 und TORE2) vom Typ SMALLINT, sowie einen Ausgabeparameter (STATUS) vom Typ VARCHAR(20) hinzu. Das Ergebnis der Parameterliste sieht dann folgendermaßen aus:



Anschließend verlassen Sie den Dialog durch einen Klick auf „Finish“.

Der IBM Data Studio Developer erstellt nun die Stored Procedure mit folgendem Text:

```
CREATE PROCEDURE ERGEBNIS_EINTRAGEN ( IN TEAM1 VARCHAR(20),
                                      IN TEAM2 VARCHAR(20),
                                      IN TORE1 SMALLINT,
                                      IN TORE2 SMALLINT,
                                      OUT STATUS VARCHAR(20) )
      VERSION VERSION1
      ISOLATION LEVEL CS
      LANGUAGE SQL
------------------------------------------------------------------------
-- SQL Stored Procedure
      -- TEAM1
      -- TEAM2
      -- TORE1
      -- TORE2
      -- STATUS
------------------------------------------------------------------------
P1: BEGIN
      -- Declare variables
      --DECLARE ENDTABLE INT DEFAULT 0;
      DECLARE STATUS_TMP VARCHAR(20) DEFAULT ' ';

      -- Declare cursor
      DECLARE cursor1 CURSOR FOR
            SELECT SCHEMA, NAME FROM SYSIBM.SYSROUTINES;
```

```
      -- Declare handler
      --DECLARE CONTINUE HANDLER FOR NOT FOUND
            --SET ENDTABLE = 1;

      -- Sample code to access results.  Goes with the lines
      -- "DECLARE ENDTABLE..." and "DECLARE CONTINUE..." shown above.
      --OPEN cursor1;
      --SET ENDTABLE = 0;
      --WHILE ENDTABLE = 0 DO
            --FETCH cursor1 INTO <Declared variables> ;
      --END WHILE;
      --CLOSE cursor1;
      SET STATUS = STATUS_TMP;
END P1
```

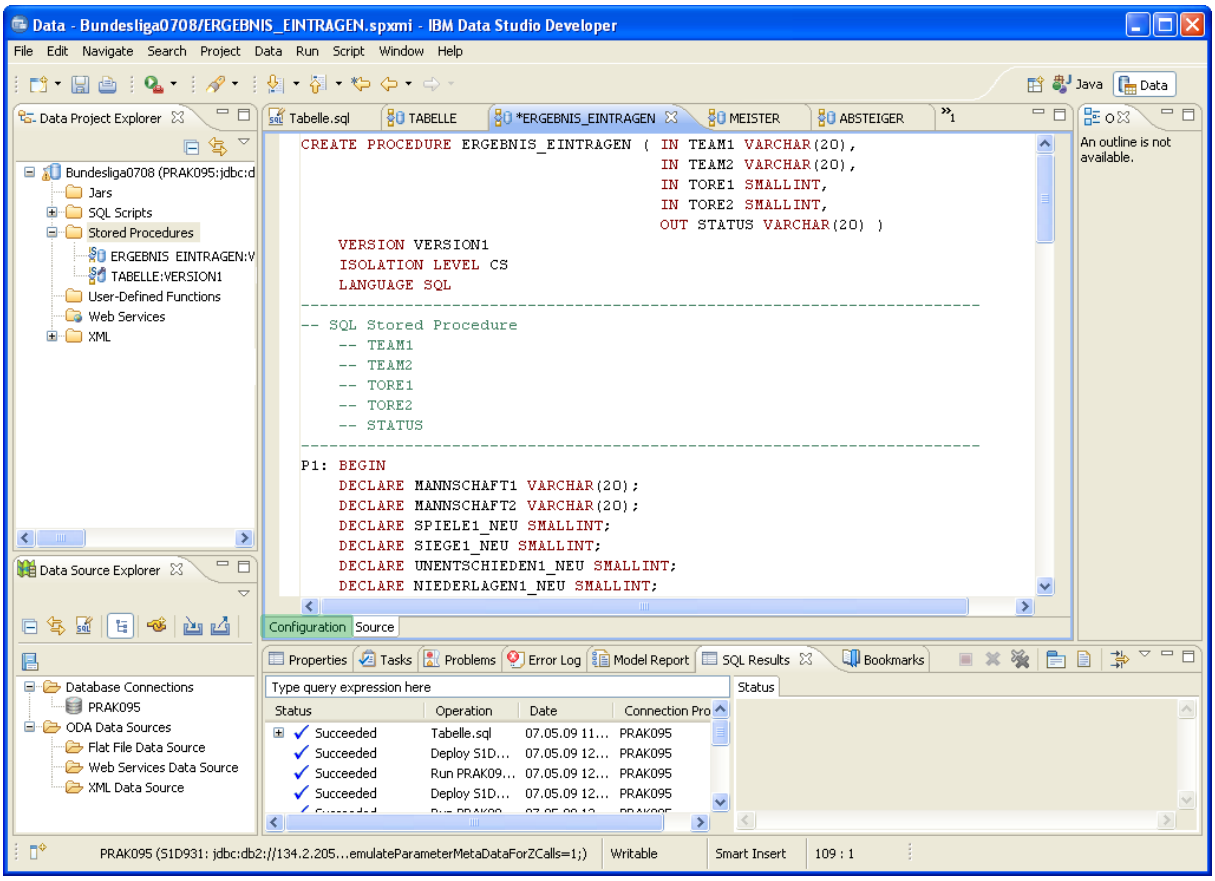
Der DDL-Teil der Stored Procedure (bis Zeile 14) ist schon korrekt, allerdings muss die Logik der Stored Procedure noch ersetzt werden. Hierzu soll folgender Code verwendet werden:

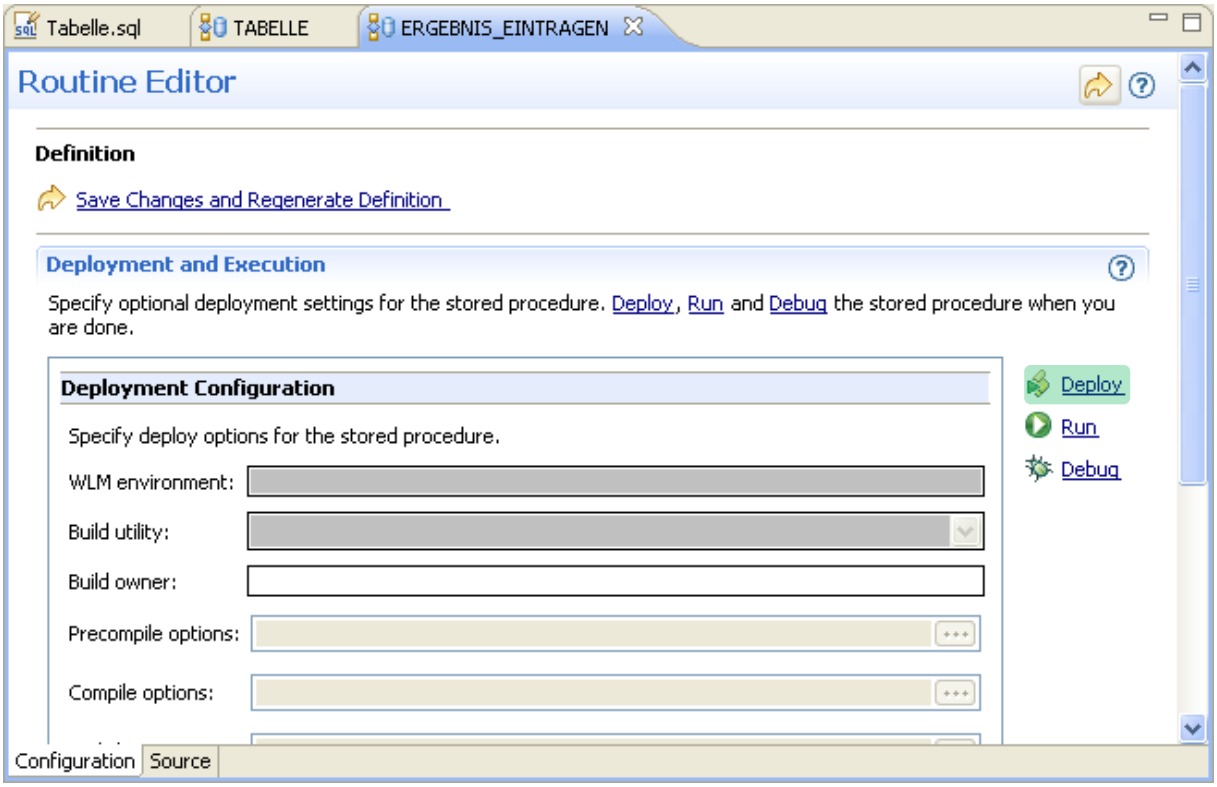
```
P1: BEGIN
      DECLARE MANNSCHAFT1 VARCHAR(20);
      DECLARE MANNSCHAFT2 VARCHAR(20);
      DECLARE SPIELE1_NEU SMALLINT;
      DECLARE SIEGE1_NEU SMALLINT;
      DECLARE UNENTSCHIEDEN1_NEU SMALLINT;
      DECLARE NIEDERLAGEN1_NEU SMALLINT;
      DECLARE TORE1_NEU SMALLINT;
      DECLARE GEGENTORE1_NEU SMALLINT;
      DECLARE TORDIFFERENZ1_NEU SMALLINT;
      DECLARE PUNKTE1_NEU SMALLINT;
      DECLARE SPIELE2_NEU SMALLINT;
      DECLARE SIEGE2_NEU SMALLINT;
      DECLARE UNENTSCHIEDEN2_NEU SMALLINT;
      DECLARE NIEDERLAGEN2_NEU SMALLINT;
      DECLARE TORE2_NEU SMALLINT;
      DECLARE GEGENTORE2_NEU SMALLINT;
      DECLARE TORDIFFERENZ2_NEU SMALLINT;
      DECLARE PUNKTE2_NEU SMALLINT;


      SELECT MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN, NIEDERLAGEN, TORE,
               GEGENTORE, TORDIFFERENZ, PUNKTE
            INTO MANNSCHAFT1, SPIELE1_NEU, SIEGE1_NEU, UNENTSCHIEDEN1_NEU,
               NIEDERLAGEN1_NEU, TORE1_NEU, GEGENTORE1_NEU,
               TORDIFFERENZ1_NEU, PUNKTE1_NEU
            FROM BULI0708
            WHERE MANNSCHAFT = TEAM1;

      SELECT MANNSCHAFT, SPIELE, SIEGE, UNENTSCHIEDEN, NIEDERLAGEN, TORE,
               GEGENTORE, TORDIFFERENZ, PUNKTE
            INTO MANNSCHAFT2, SPIELE2_NEU, SIEGE2_NEU, UNENTSCHIEDEN2_NEU,
               NIEDERLAGEN2_NEU, TORE2_NEU, GEGENTORE2_NEU,
               TORDIFFERENZ2_NEU, PUNKTE2_NEU
            FROM BULI0708
            WHERE MANNSCHAFT = TEAM2;

            IF(MANNSCHAFT1 IS NULL) THEN
                  SET STATUS = 'TEAM1 existiert nicht';
            ELSEIF (MANNSCHAFT2 IS NULL) THEN
                  SET STATUS = 'TEAM2 existiert nicht';
            ELSE
```

```
                IF (TORE1 > TORE2) THEN
                      SET SIEGE1_NEU = SIEGE1_NEU +1;
                      SET NIEDERLAGEN2_NEU = NIEDERLAGEN2_NEU + 1;
                      SET PUNKTE1_NEU = PUNKTE1_NEU + 3;
                ELSEIF (TORE1 < TORE2) THEN
                      SET NIEDERLAGEN1_NEU = NIEDERLAGEN1_NEU + 1;
                      SET SIEGE2_NEU = SIEGE2_NEU + 1;
                      SET PUNKTE2_NEU = PUNKTE2_NEU + 3;
                ELSE
                      SET UNENTSCHIEDEN1_NEU = UNENTSCHIEDEN1_NEU + 1;
                      SET UNENTSCHIEDEN2_NEU = UNENTSCHIEDEN2_NEU + 1;
                      SET PUNKTE1_NEU = PUNKTE1_NEU + 1;
                      SET PUNKTE2_NEU = PUNKTE2_NEU + 1;
                END IF;


                SET SPIELE1_NEU = SPIELE1_NEU+1;
                SET SPIELE2_NEU = SPIELE2_NEU+1;
                SET TORE1_NEU = TORE1_NEU + TORE1;
                SET TORE2_NEU = TORE2_NEU + TORE2;
                SET GEGENTORE1_NEU = GEGENTORE1_NEU + TORE2;
                SET GEGENTORE2_NEU = GEGENTORE2_NEU + TORE1;
                SET TORDIFFERENZ1_NEU = TORE1_NEU - GEGENTORE1_NEU;
                SET TORDIFFERENZ2_NEU = TORE2_NEU - GEGENTORE2_NEU;

                UPDATE BULI0708 SET
                      SPIELE=SPIELE1_NEU,
                      SIEGE = SIEGE1_NEU,
                      UNENTSCHIEDEN = UNENTSCHIEDEN1_NEU,
                      NIEDERLAGEN = NIEDERLAGEN1_NEU,
                      TORE = TORE1_NEU,
                      GEGENTORE = GEGENTORE1_NEU,
                      TORDIFFERENZ = TORDIFFERENZ1_NEU,
                      PUNKTE = PUNKTE1_NEU
                WHERE MANNSCHAFT = TEAM1;

                UPDATE BULI0708 SET
                      SPIELE=SPIELE2_NEU,
                      SIEGE = SIEGE2_NEU,
                      UNENTSCHIEDEN = UNENTSCHIEDEN2_NEU,
                      NIEDERLAGEN = NIEDERLAGEN2_NEU,
                      TORE = TORE2_NEU,
                      GEGENTORE = GEGENTORE2_NEU,
                      TORDIFFERENZ = TORDIFFERENZ2_NEU,
                      PUNKTE = PUNKTE2_NEU
                WHERE MANNSCHAFT = TEAM2;

                SET STATUS = 'Spiel eingetragen';
          END IF;
END P1
```

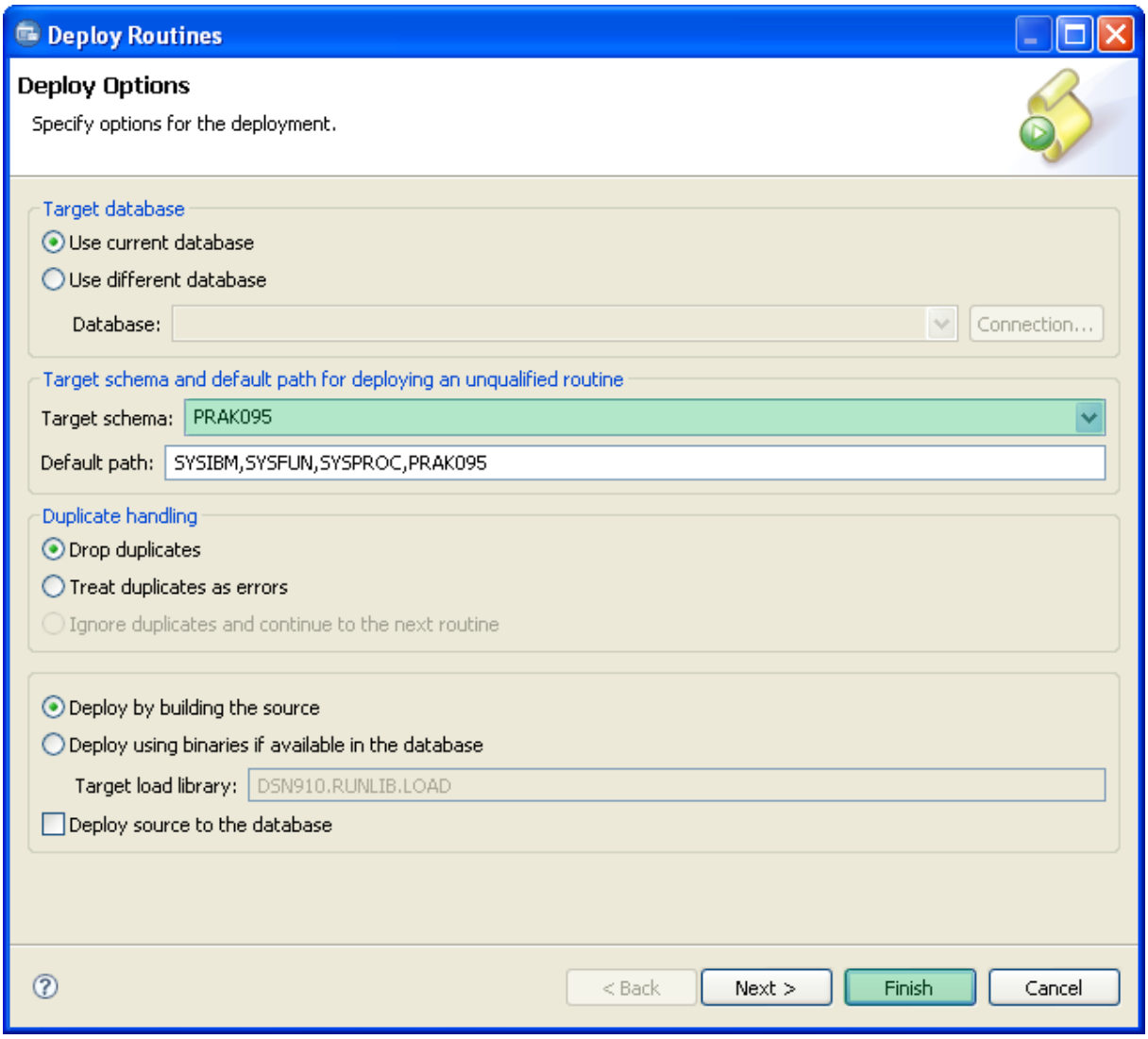
Zunächst werden in den DECLARE-Befehlen einige temporäre Variablen deklariert. In sie werden zunächst die alten Tabellenwerte ausgelesen (durch die ersten zwei SELECT-Befehle). Wenn eine Mannschaft eingegeben wurde, die nicht existiert, wird dies im Ausgabeparameter STATUS vermerkt und keine Änderung an der Tabelle vorgenommen. Andernfalls werden durch die SET-Befehle die ausgelesenen Werte modifiziert und über die UPDATE-Befehle zurück in die Datenbank geschrieben.

Nun soll die Stored Procedure im entfernten DB2-System eingerichtet werden. Hierzu wechseln Sie wieder vom Quellcode in das Configuration-Fenster.

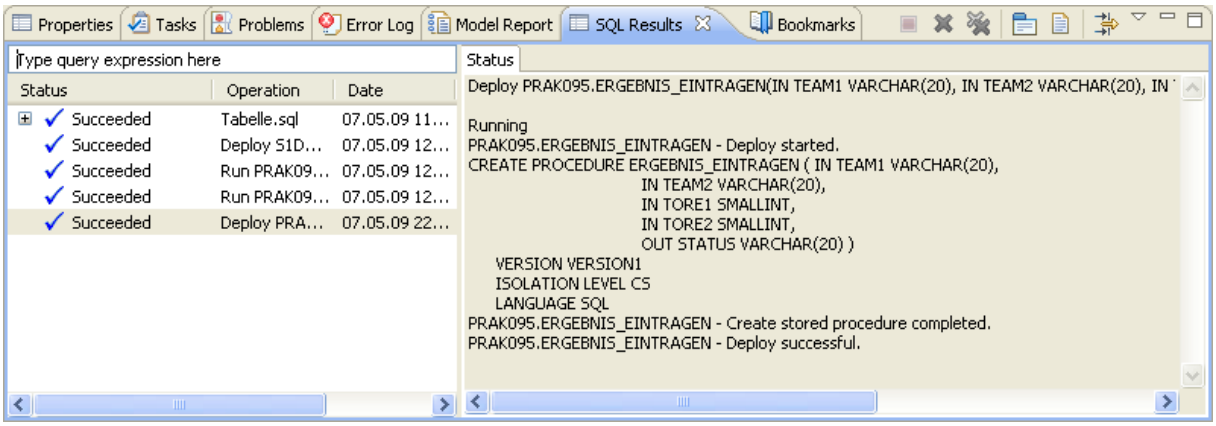




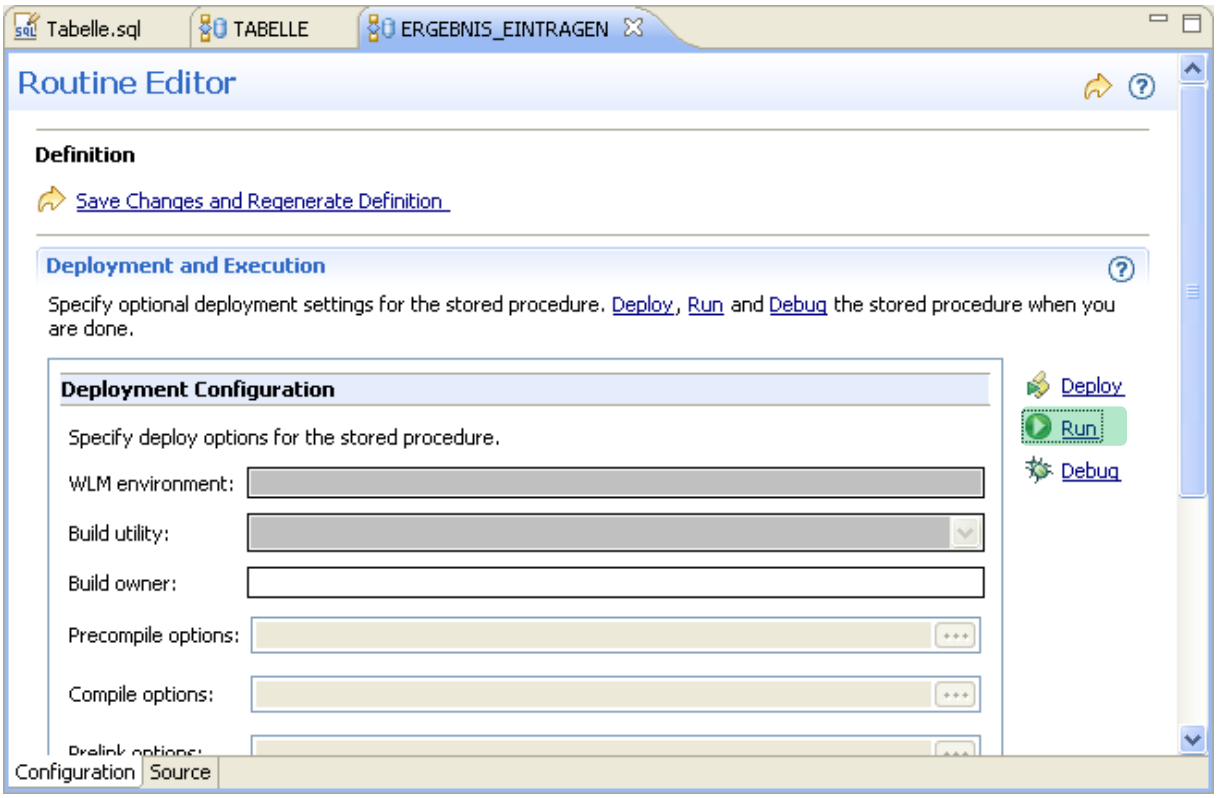
Durch einen Klick auf „Deploy“ öffnet sich ein neues Fenster. Stellen Sie sicher, dass Ihr Datenbankschema (=Benutzerkennung) ausgewählt ist und klicken Sie auf „Finish“.



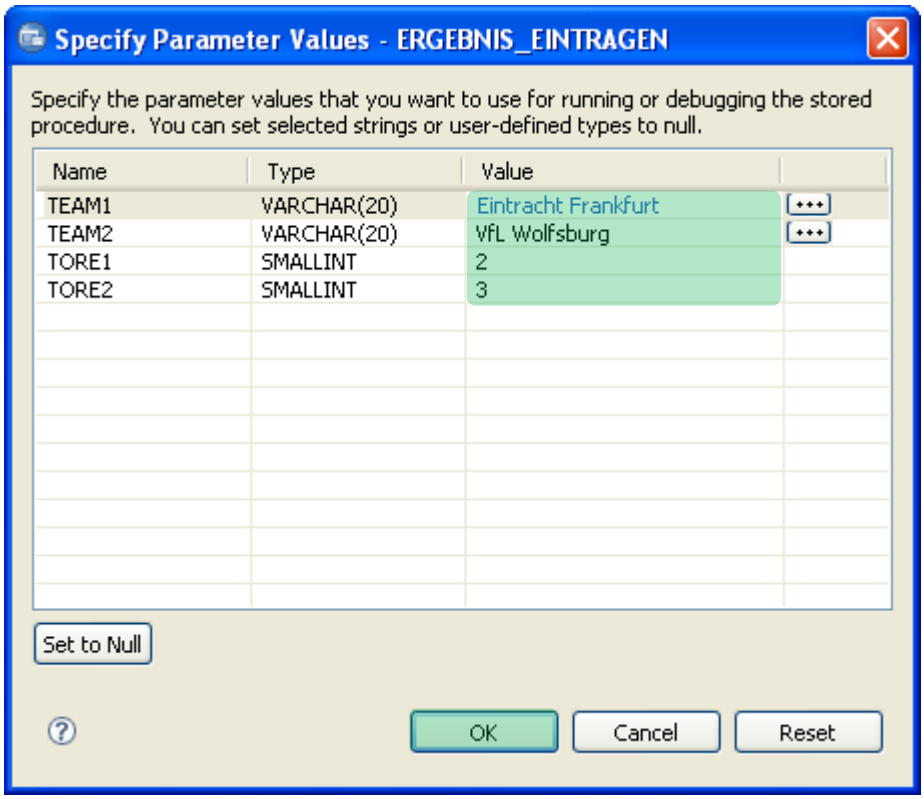
Das Ergebnis bekommen Sie wieder im SQL Results-Feld ausgegeben:



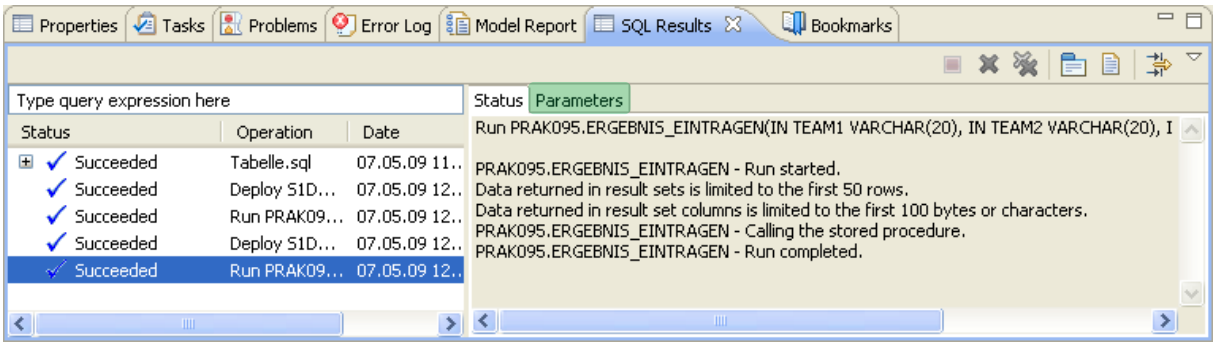
Nun können Sie die Stored Procedure durch einen Klick auf „Run“ ausführen.



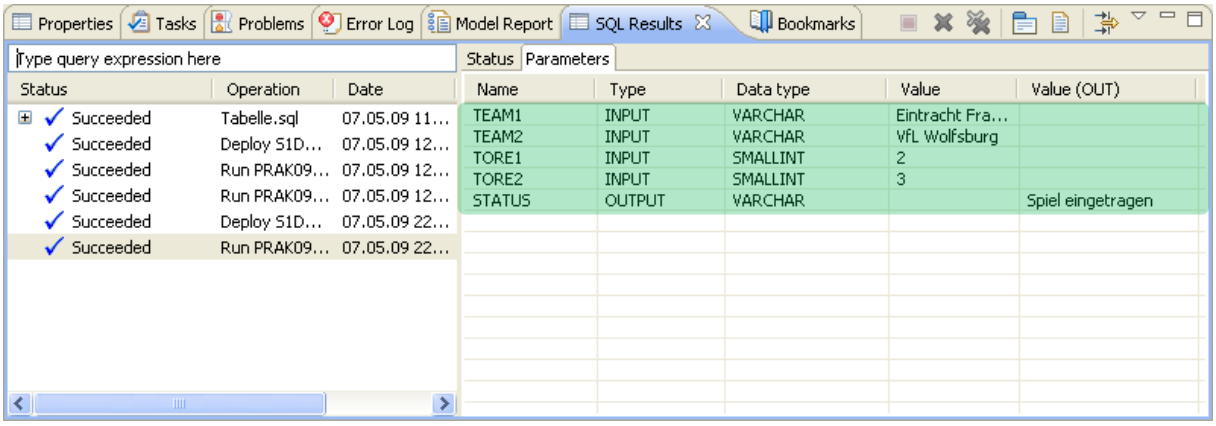
Da unsere Stored Procedure Parameter erwartet, öffnet sich ein Dialog, in den Sie die nötigen Werte eintragen können. Tragen Sie z.B. das Spiel zwischen Eintracht Frankfurt und VfL Wolfsburg (richtige Groß- und Kleinschreibung beachten!) ein, das 2:3 ausgegangen ist.



Nach einem Klick auf „OK“ bekommen Sie das Ergebnis der Ausführung wieder im SQL Results-Register angezeigt.

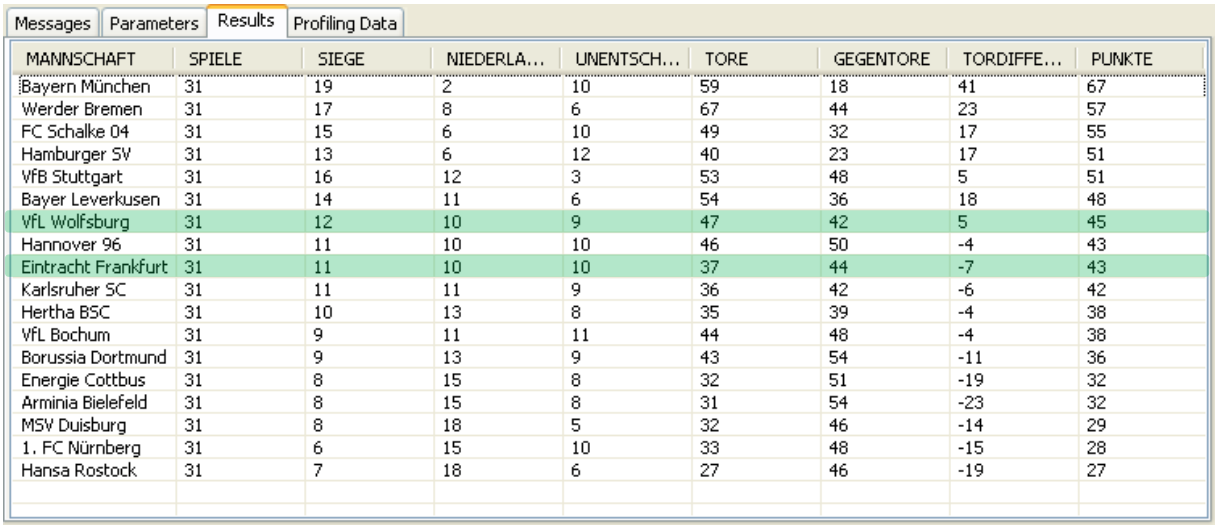


Um den Parameter STATUS auszulesen, wird auf „Parameters“ geklickt.



Wenn Sie jetzt die vorher angelegte Stored Procedure TABELLE ein zweites Mal ausführen, sehen Sie, dass das Eintragen des Spiels erfolgreich war.

Tabelle vor dem Aufruf von ERGEBNIS_EINTRAGEN:



MANNSCHAFT	SPIELE	SIEGE	NIEDERLA...	UNENTSCH...	TORE	GEGENTORE	TORDIFFE...	PUNKTE
Bayern München	31	19	2	10	59	18	41	67
Werder Bremen	31	17	8	6	67	44	23	57
FC Schalke 04	31	15	6	10	49	32	17	55
Hamburger SV	31	13	6	12	40	23	17	51
VfB Stuttgart	31	16	12	3	53	48	5	51
Bayer Leverkusen	31	14	11	6	54	36	18	48
VfL Wolfsburg	31	12	10	9	47	42	5	45
Hannover 96	31	11	10	10	46	50	-4	43
Eintracht Frankfurt	31	11	10	10	37	44	-7	43
Karlsruher SC	31	11	11	9	36	42	-6	42
Hertha BSC	31	10	13	8	35	39	-4	38
VfL Bochum	31	9	11	11	44	48	-4	38
Borussia Dortmund	31	9	13	9	43	54	-11	36
Energie Cottbus	31	8	15	8	32	51	-19	32
Arminia Bielefeld	31	8	15	8	31	54	-23	32
MSV Duisburg	31	8	18	5	32	46	-14	29
1. FC Nürnberg	31	6	15	10	33	48	-15	28
Hansa Rostock	31	7	18	6	27	46	-19	27

Tabelle nach dem Aufruf von ERGEBNIS_EINTRAGEN:

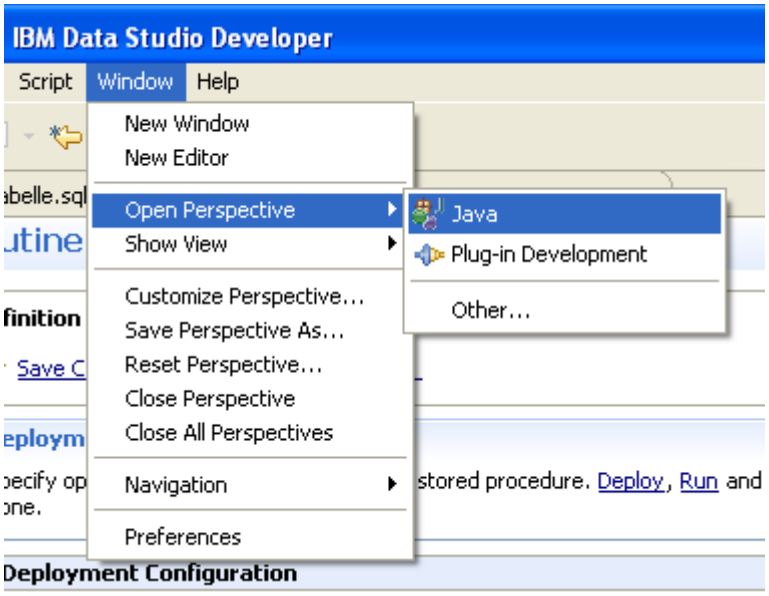
Messages | Parameters | Results | Profiling Data

MANNSCHAFT	SPIELE	SIEGE	NIEDERLA...	UNENTSCH...	TORE	GEGENTORE	TORDIFFE...	PUNKTE
Bayern München	31	19	2	10	59	18	41	67
Werder Bremen	31	17	8	6	67	44	23	57
FC Schalke 04	31	15	6	10	49	32	17	55
Hamburger SV	31	13	6	12	40	23	17	51
VfB Stuttgart	31	16	12	3	53	48	5	51
Bayer Leverkusen	31	14	11	6	54	36	18	48
VfL Wolfsburg	32	13	10	9	50	44	6	48
Hannover 96	31	11	10	10	46	50	-4	43
Eintracht Frankfurt	32	11	11	10	39	47	-8	43
Karlsruher SC	31	11	11	9	36	42	-6	42
Hertha BSC	31	10	13	8	35	39	-4	38
VfL Bochum	31	9	11	11	44	48	-4	38
Borussia Dortmund	31	9	13	9	43	54	-11	36
Energie Cottbus	31	8	15	8	32	51	-19	32
Arminia Bielefeld	31	8	15	8	31	54	-23	32
MSV Duisburg	31	8	18	5	32	46	-14	29
1. FC Nürnberg	31	6	15	10	33	48	-15	28
Hansa Rostock	31	7	18	6	27	46	-19	27

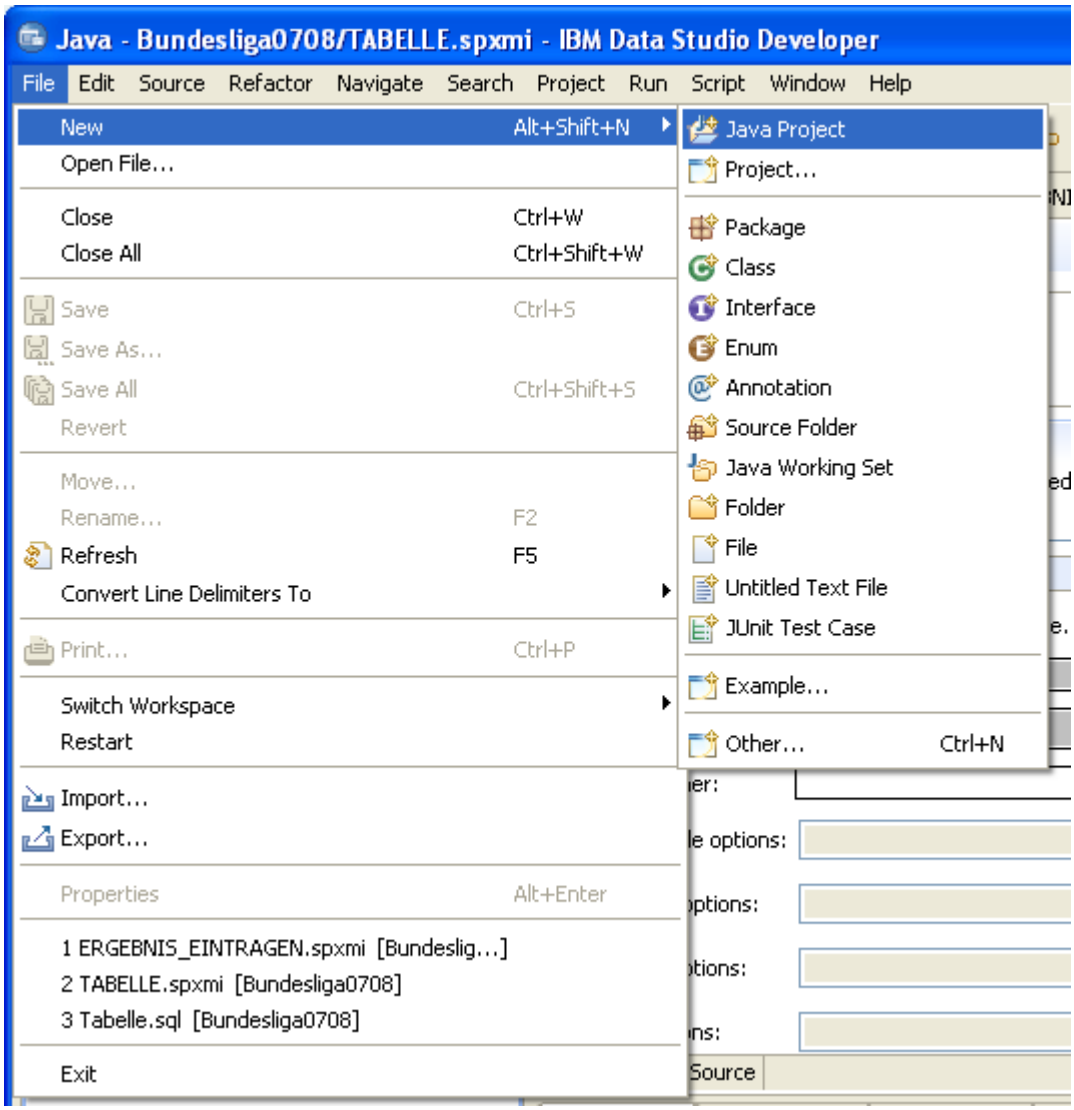
Aufgabe: Schreiben Sie eine Stored Procedure mit Namen PROGNOSE, die zwei Eingabeparameter (TEAM1 VARCHAR(20), TEAM2 VARCHAR(20)) und zwei Ausgabeparameter (TORE1 SMALLINT, TORE2 SMALLINT) besitzt. Die Stored Procedure soll nach einem Algorithmus ihrer Wahl eine Prognose für das Ergebnis des Spiels TEAM1 gegen TEAM2 liefern. Ein möglicher Algorithmus könnte z.B. die durchschnittliche Anzahl der Tore pro Spiel einer Mannschaft zurückliefern.

4.2.3 Einbindung von Stored Procedures in eine Java-Anwendung

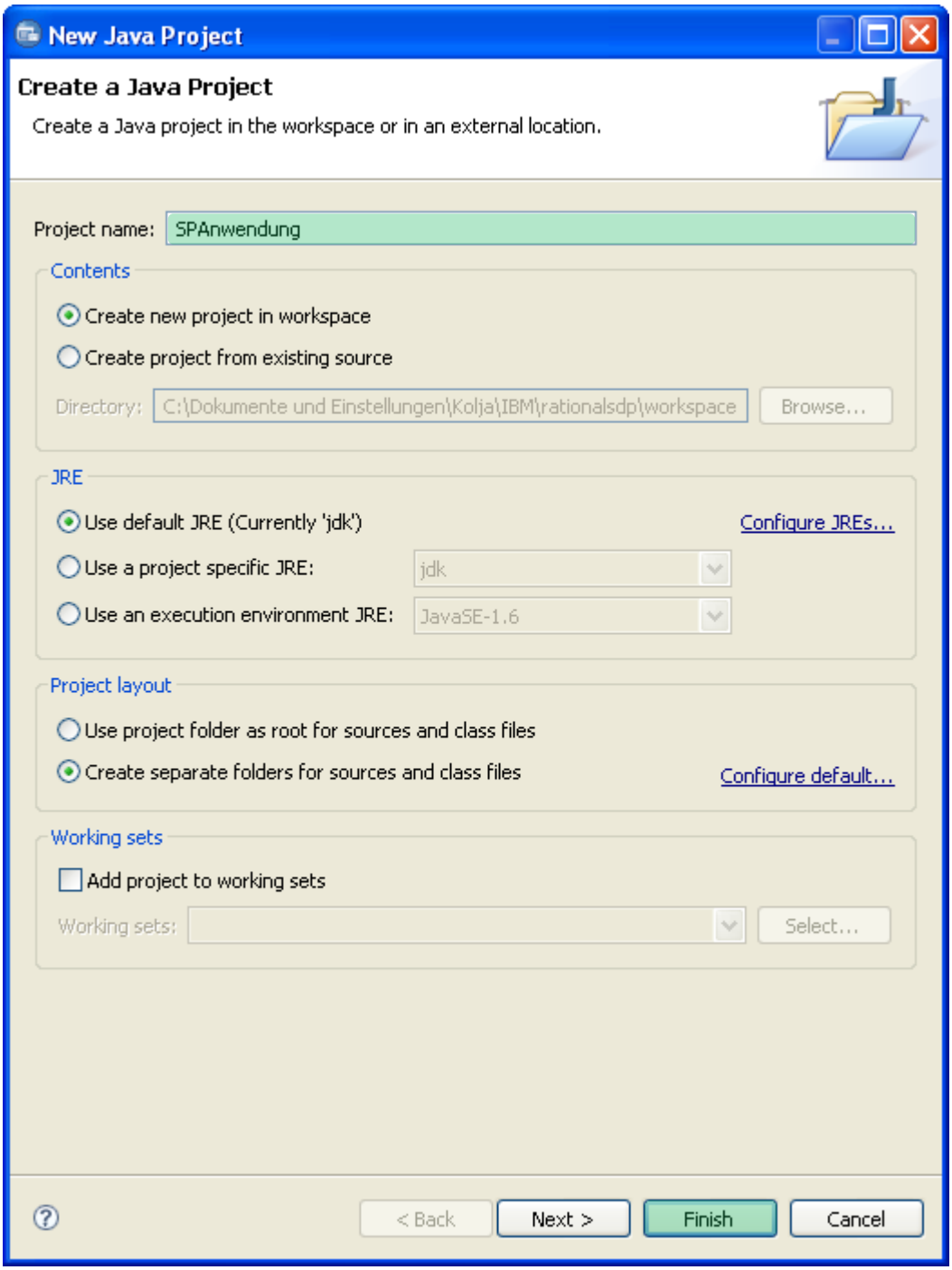
Im Folgenden soll betrachtet werden, wie eine Stored Procedure aus einer Java-Anwendung über die JDBC-Schnittstelle aufgerufen werden kann. Hierzu wechseln Sie zunächst in die Java-Perspektive über „Window“, „Open Perspective“ und danach „Java“.



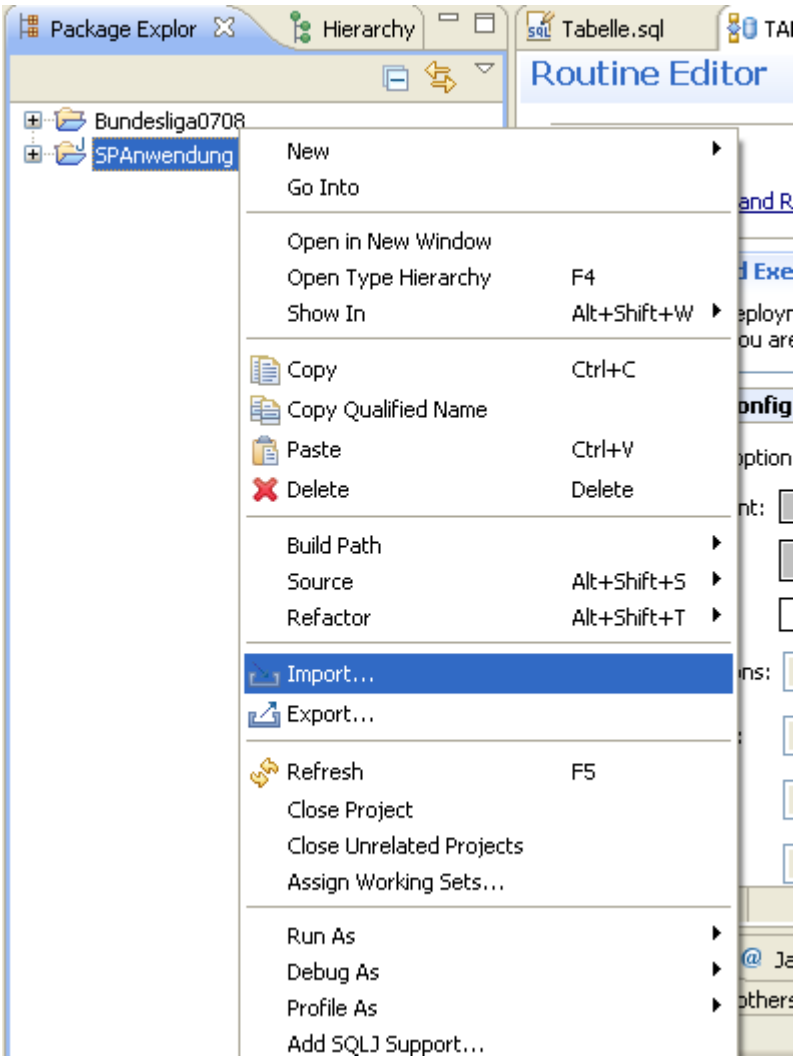
Legen Sie ein normales Java-Projekt an und importieren Sie das Archiv Anwendung_NAT_SQL in ein neu erstelltes Java-Projekt. Klicken Sie hierzu auf „File“, „New“ und danach „Java Project“.



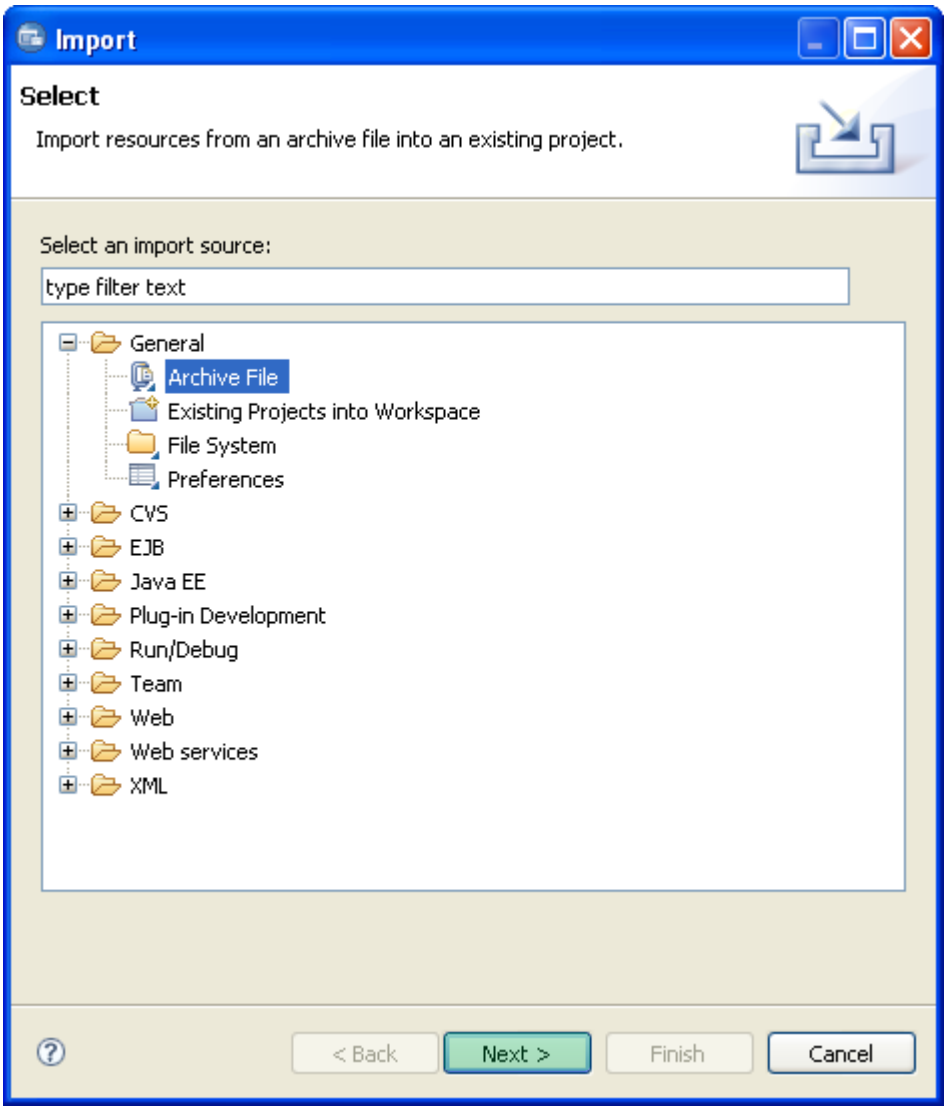
Geben Sie als Namen für das Projekt SPAnwendung im sich öffnenden Fenster ein und klicken Sie auf „Finish“.



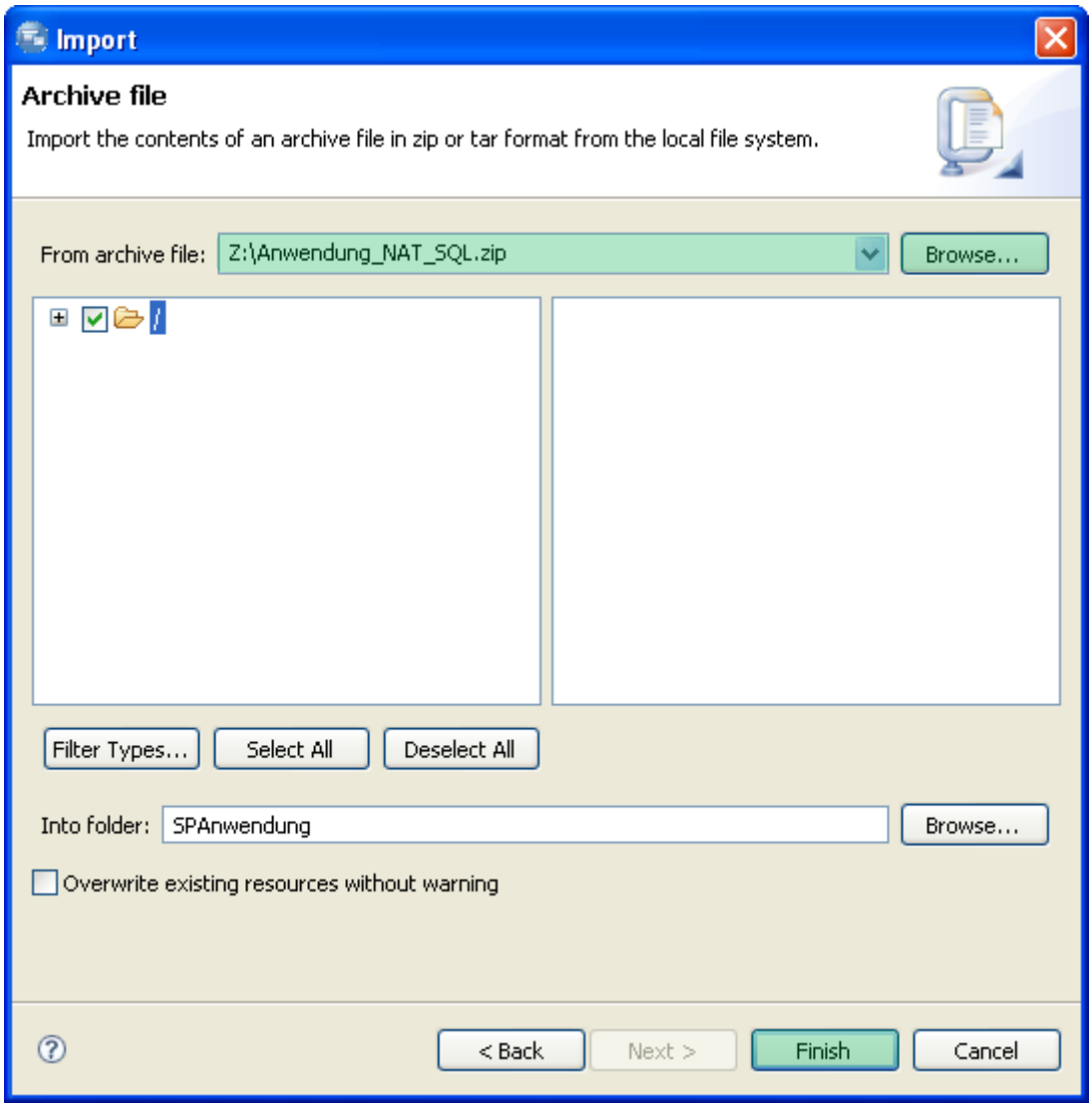
Die Beispielanwendung liegt als Zip-Archiv vor und kann über Ihren Tutor bezogen werden. Um das Zip-Archiv zu importieren, müssen Sie mit der rechten Maustaste auf das neue Projekt klicken und dort „Import“ auswählen.

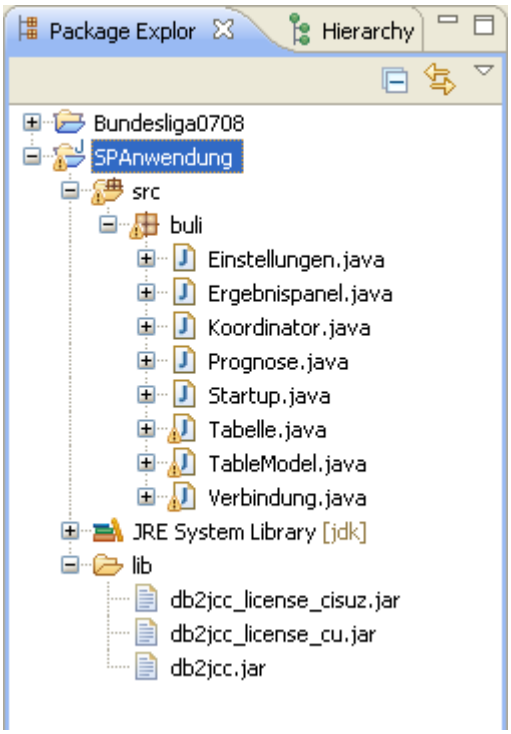


Unter dem Menüpunkt „General“ befindet sich ein Eintrag für archivierte Dateien.



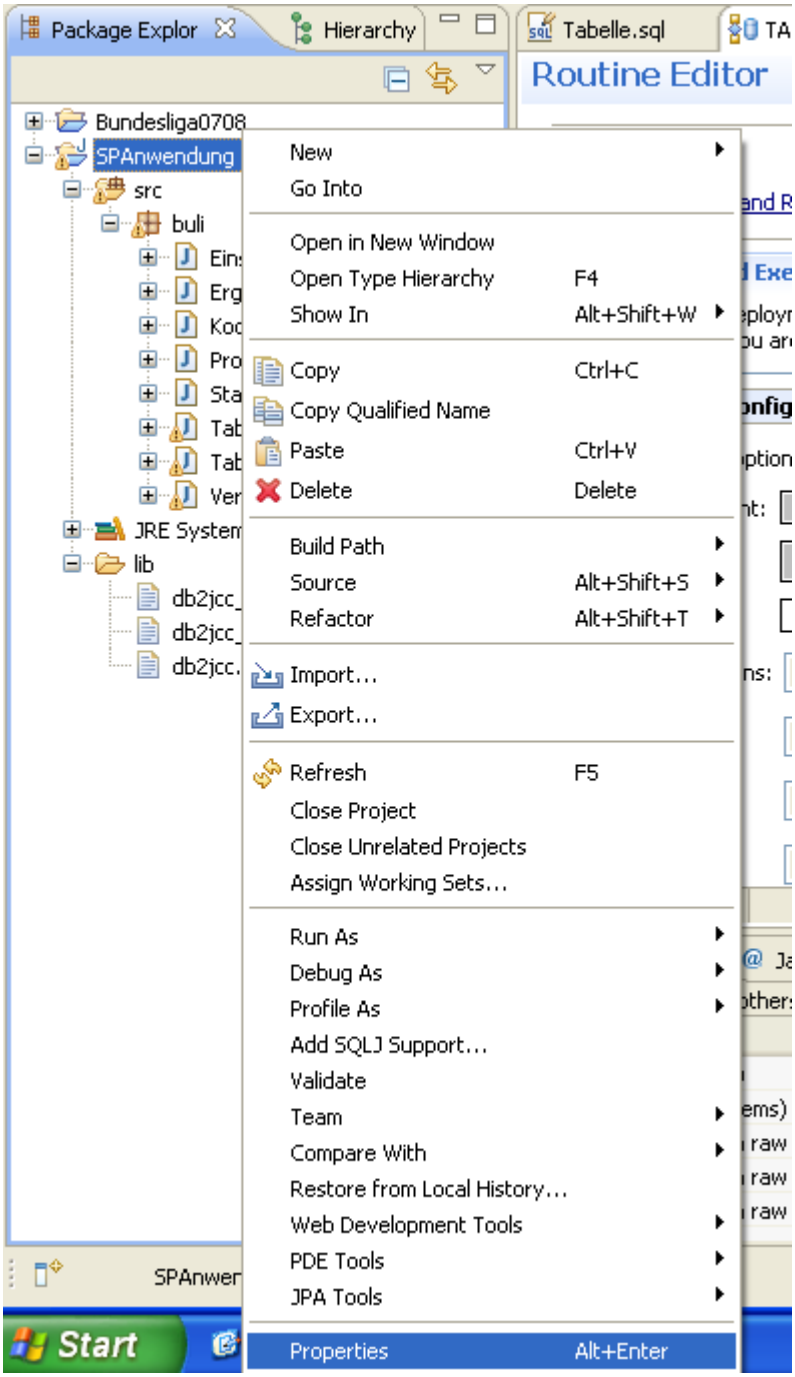
Mit dem Knopf „Browse“ können Sie den Speicherort des Archivs auswählen. Klicken Sie danach auf „Finish“.


Import
Archive file
Import the contents of an archive file in zip or tar format from the local file system.
From archive file:
Z:\Anwendung_NAT_SQL.zip
Browse...
/
Filter Types...
Select All
Deselect All
Into folder:
SPAnwendung
Browse...
Overwrite existing resources without warning
< Back
Next >
Finish
Cancel

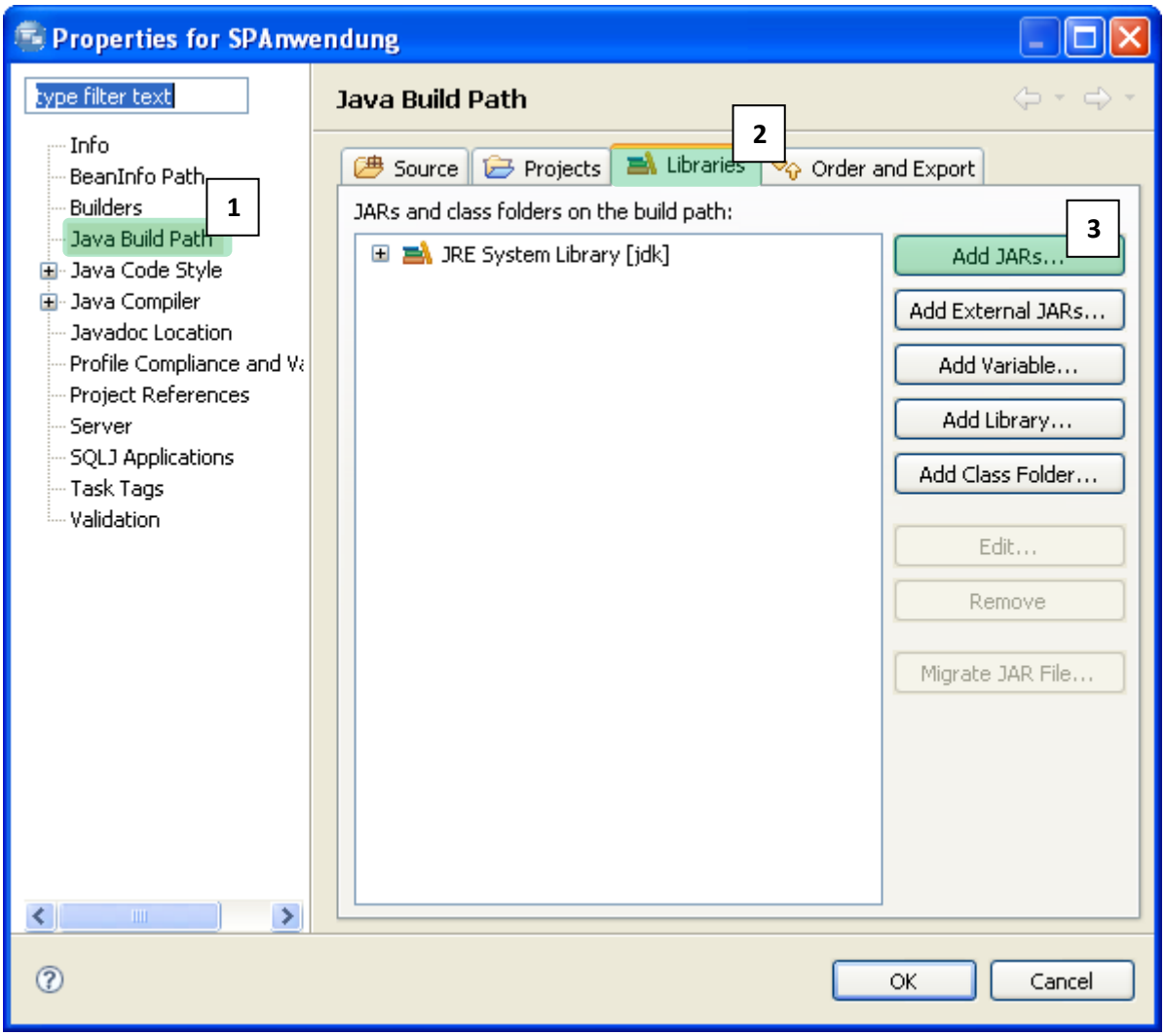

Package Explor
Hierarchy
Bundesliga0708
SPAnwendung
src
buli
Einstellungen.java
Ergebnispanel.java
Koordinator.java
Prognose.java
Startup.java
Tabelle.java
TableModel.java
Verbindung.java
JRE System Library [jdk]
lib
db2jcc_license_cisuz.jar
db2jcc_license_cu.jar
db2jcc.jar

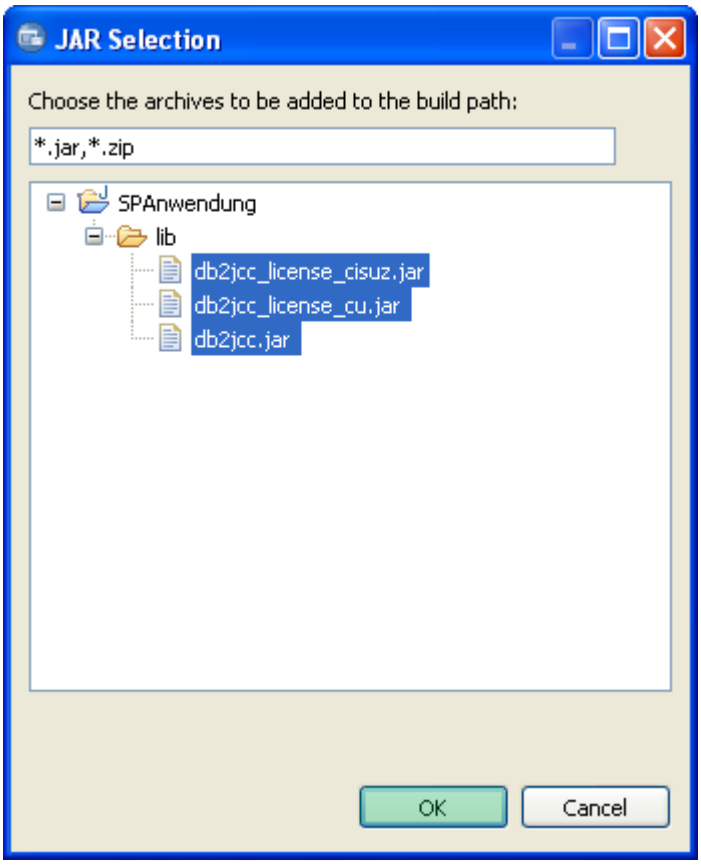
Wenn Sie das Projekt nun betrachten, können Sie feststellen, dass es aus einem Package buli besteht, das einige Java-Klassen enthält und einem Ordner lib. Die drei enthaltenen Jar-Archive kennen Sie bereits aus Tutorial 8. Sie enthalten die DB2Connect-Treiber und sorgen für die Verbindung zur entfernten Datenbank. Näheres über den Zugriff auf die entfernte Datenbank mittels DB2Connect befindet sich in Tutorial 8.

Die Archive müssen als Bibliotheken für das Projekt eingebunden werden. Klicken Sie dazu mit der rechten Maustaste auf das Projekt SPAnwendung und wählen Sie im aufklappenden Menü „Properties“ aus.



Im Java Build Path werden die DB2Connect-Treiber im Register Libraries über den Knopf „Add Jars“ dem Projekt hinzugefügt. Hierzu müssen alle drei Archive markiert werden.


Properties for SPAnwendung
type filter text
Info
BeanInfo Path
Builders
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Profile Compliance and V
Project References
Server
SQLJ Applications
Task Tags
Validation
1
Java Build Path
2
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
JRE System Library [jdk]
3
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Edit...
Remove
Migrate JAR File...
OK
Cancel



Durch zweimaligen Klick auf „OK“ schließen sich die Fenster wieder.

Das Programm ist folgendermaßen aufgebaut:

Die Klassen Einstellungen, Ergebnispanel, Prognose, Saisonpanel, Tabelle und TableModel dienen nur der grafischen Ausgabe und sollen hier nicht weiter betrachtet werden. Der Koordinator dient als Vermittler zwischen GUI und der Verbindungsklasse. Wenn ein Benutzer eine Aktion auf der GUI tätigt, sorgt der Koordinator dafür, dass auf einem Objekt der Klasse Verbindung die richtige Stored Procedure aufgerufen und das Ergebnis an die GUI zurückgeliefert wird. Die Klasse Startup dient dazu, Objekte der vorhandenen Klassen zu instanziieren. Auch der Code dieser beiden Klassen soll hier nicht weiter betrachtet werden.

Hier soll nur noch der Teil betrachtet werden, der zur Ausführung der Stored Procedures führt, der sich in der Klasse Verbindung befindet:

Mit dem Befehl

```
Connection db = DriverManager.getConnection("jdbc:db2://" + host + ":" +
port + "/" + location , userid , passwort);
```

wird in der Methode setConnectionParams eine JDBC-Verbindung zum entfernten Datenbanksystem aufgebaut. Dabei ist db vom Typ `java.sql.Connection`. Über dieses Interface kann über die Methode `db.createStatement()` ein `java.sql.Statement` erstellt werden. Auf einem Objekt dieser Klasse können SQL-Befehle mittels `executeQuery(`*`Anfragestring`*`)` ausgeführt werden. Der Rückgabewert dieses Aufrufs ist ein Objekt vom Typ `java.sql.ResultSet`. Mit Hilfe dieses Objekts kann iterativ durch die Ergebnismenge geschritten werden (`while (RS.next()){…}`).

Die SQL-Anweisungen, die mittels `executeQuery()` an die Datenbank gesendet werden, müssen zunächst auf syntaktische Korrektheit überprüft werden. Dann werden sie in einen internen Ausführungsplan der Datenbank übersetzt und mit anderen Transaktionen optimal verzahnt. Dadurch nimmt die Ausführung solcher Befehle relativ viel Zeit in Anspruch. Sinnvoller wäre es, eine Art Vorübersetzung für SQL-Anweisungen zu nutzen. Diese Vorübersetzung ist eine Eigenschaft, die JDBC unterstützt und die sich `PreparedStatement` nennt. Dabei werden die Anweisungen in einem ersten Schritt zur Datenbank geschickt und dort in ein internes Format umgesetzt. Später verweist ein Programm auf diese vorübersetzten Anweisungen, und die Datenbank kann sie schnell ausführen, da sie in einem optimalen Format vorliegen.

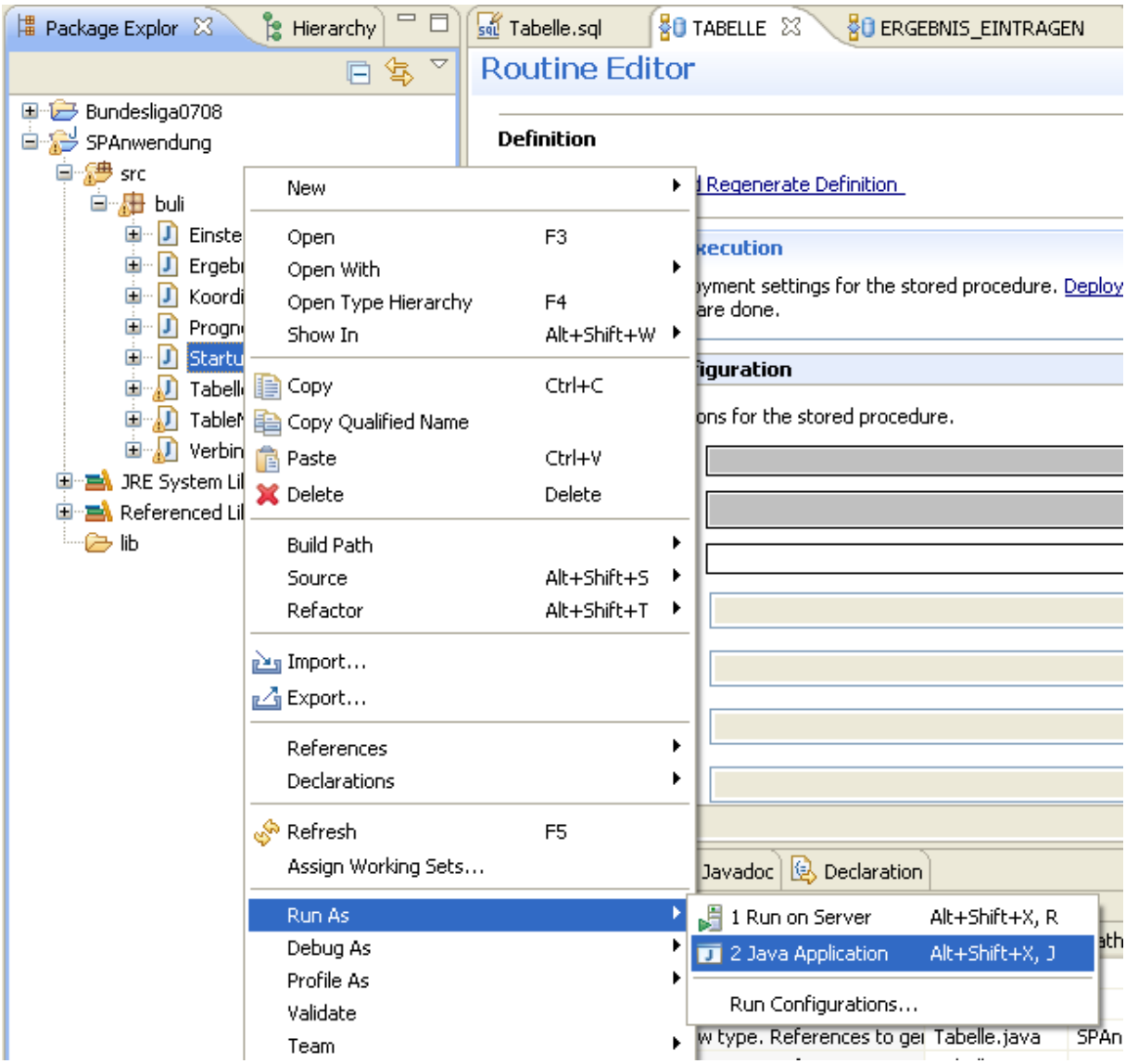
Für CALL-Befehle zum Aufruf von Stored Procedures gibt es ein weiteres Anfrage-Interface mit Namen `CallableStatement`. Es ermöglicht einen leichten Umgang mit Input- und Outputparametern. Inputparameter werden zunächst mittels eines „?“ im Anfragestring angegeben und später über die jeweiligen Setter-Methoden durch den Wert einer Variablen ersetzt. Die Position des jeweiligen Parameters muss über einen Index angegeben werden.

```
String query = "CALL ERGEBNIS_EINTRAGEN(?,?,?,?,?)";
CallableStatement Stmt = db.prepareCall(query);
Stmt.setString(1, team1);
Stmt.setString(2, team2);
Stmt.setInt(3, Integer.parseInt(tore1));
Stmt.setInt(4, Integer.parseInt(tore2));
```

Mittels der Funktion `registerOutParameter` kann ein Outputparameter an den Befehl gebunden werden. Es enthält nach dem Ausführen des Befehls den entsprechenden Rückgabewert.

```
String query = "CALL PROGNOSE(?,?,?,?)";
CallableStatement Stmt = db.prepareCall(query);
Stmt.setString(1, team1);
Stmt.setString(2, team2);
Stmt.registerOutParameter(3, java.sql.Types.SMALLINT);
Stmt.registerOutParameter(4, java.sql.Types.SMALLINT);
Stmt.execute();

int tore1 = Stmt.getInt(3);
int tore2 = Stmt.getInt(4);
```

Durch einen Klick mit der rechten Maustaste auf die Startup-Klasse, können Sie auswählen: „Run as“ „Java Anwendung“.



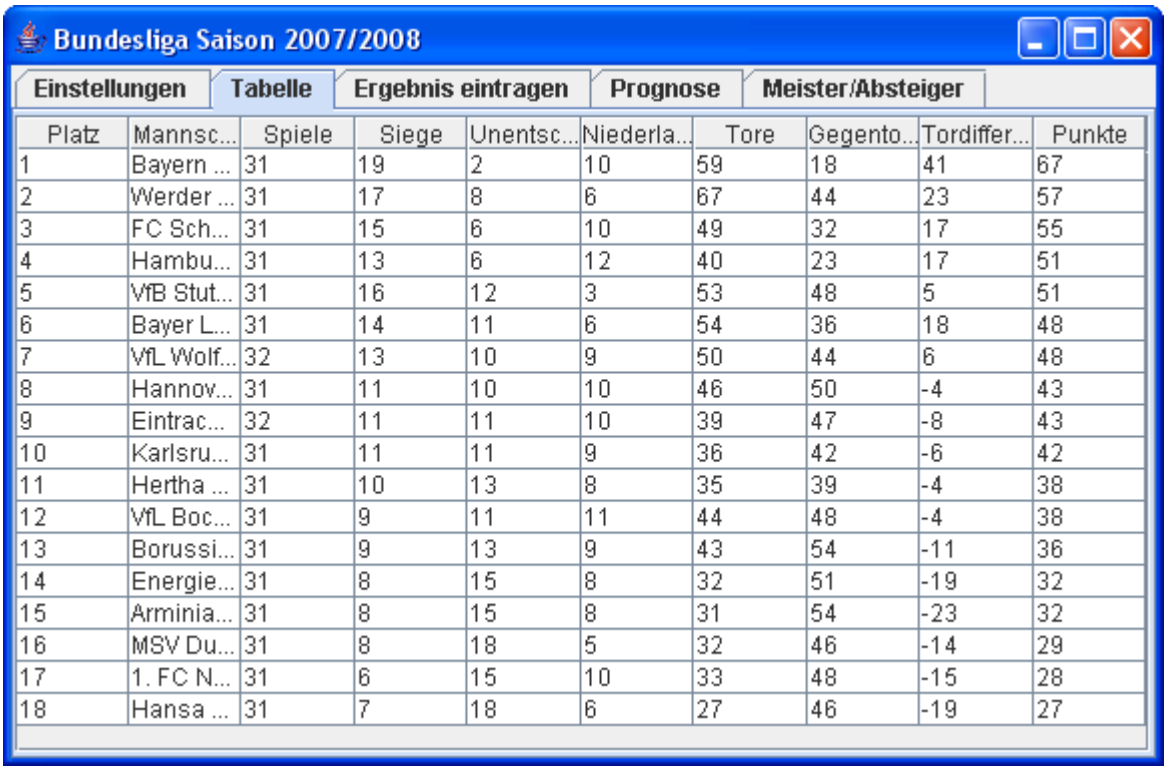
Nun müssen die entsprechenden Datenbankdaten angegeben werden:

Für die Tübinger LPAR Hobbit gilt wieder:
Host: 134.2.205.54
Port: 4019
Location: S1D931

Und für Binks in Leipzig:
Host: 139.18.4.34
Port: 4019
Location: S1D931

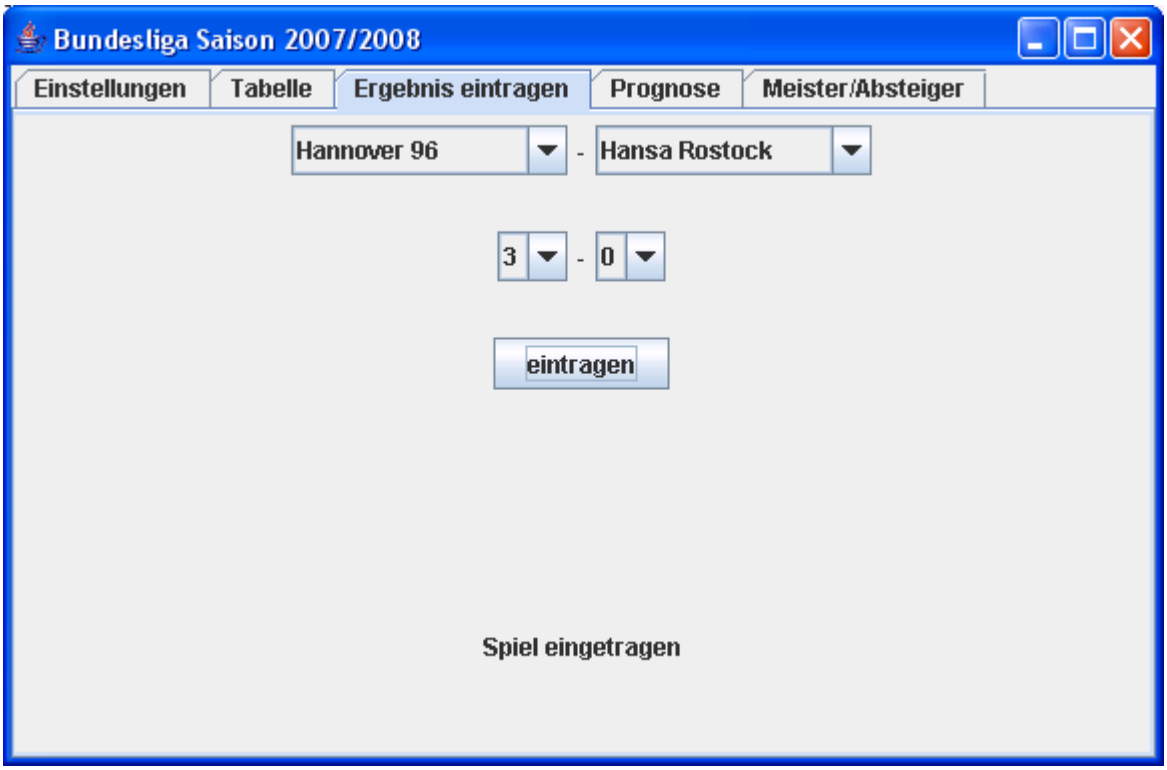
User-ID und Passwort entsprechen Ihren Login-Daten.
Durch einen Klick auf „Speichern“ wird versucht die Verbindung herzustellen. Sollte dies nicht funktionieren, überprüfen Sie die Daten.

Beim Verbindungsaufbau wird außerdem die Stored Procedure TABELLE und TEAMS aufgerufen. Das Ergebnis kann im Register „Tabelle“ bzw. „Ergebnis eintragen“/„Prognose“ angeschaut werden.



Platz	Mannsc...	Spiele	Siege	Unentsc...	Niederla...	Tore	Gegento...	Tordiffer...	Punkte
1	Bayern ...	31	19	2	10	59	18	41	67
2	Werder ...	31	17	8	6	67	44	23	57
3	FC Sch...	31	15	6	10	49	32	17	55
4	Hambu...	31	13	6	12	40	23	17	51
5	VfB Stut...	31	16	12	3	53	48	5	51
6	Bayer L...	31	14	11	6	54	36	18	48
7	VfL Wolf...	32	13	10	9	50	44	6	48
8	Hannov...	31	11	10	10	46	50	-4	43
9	Eintrac...	32	11	11	10	39	47	-8	43
10	Karlsru...	31	11	11	9	36	42	-6	42
11	Hertha ...	31	10	13	8	35	39	-4	38
12	VfL Boc...	31	9	11	11	44	48	-4	38
13	Borussi...	31	9	13	9	43	54	-11	36
14	Energie...	31	8	15	8	32	51	-19	32
15	Arminia...	31	8	15	8	31	54	-23	32
16	MSV Du...	31	8	18	5	32	46	-14	29
17	1. FC N...	31	6	15	10	33	48	-15	28
18	Hansa ...	31	7	18	6	27	46	-19	27

Um die Stored Procedure ERGEBNIS_EINTRAGEN oder PROGNOSE zu testen, können Sie die entsprechenden Formulare auf den gleichnamigen Registern ausfüllen – die Tabelle wird nach dem Eintragen eines neuen Ergebnisses neu geladen.



Aufgabe: Fertigen Sie einen Screenshot an von der Bundesligatabelle bevor Sie ein Ergebnis eingetragen haben und einen danach. Erstellen Sie einen weiteren Screenshot, der die korrekte Funktionalität der Stored Procedures PROGNOSE zeigt. Geben Sie außerdem Ihren Algorithmus für die Berechnung der Prognose an und erklären Sie ihn.

5 External high-level language Stored Procedures

5.1 Allgemeines

Nachdem im letzten Kapitel die Erstellung von Native SQL Stored Procedures erläutert wurde, sollen in diesem Kapitel External high-level language Stored Procedures genauer betrachtet werden.

Während sowohl Native SQL Stored Procedures und External SQL Stored Procedures aus SQL-Code bestehen, ist der Code von Stored Procedures des dritten Typs, External high-level language Stored Procedures, in einer vom Server unterstützten Programmiersprache geschrieben. Auf z/OS sind das COBOL, PL/1, C, C++, Assembler, REXX und Java. Mit Ausnahme von REXX müssen die Programme vorkompiliert, kompiliert, gelinkt und an das Datenbanksystem gebunden werden. Je nachdem ob die Programme statisches oder dynamisches SQL beinhalten, finden diese Vorgänge zur Laufzeit oder zur Compilezeit statt. Bei statischem SQL steht der im Programm ausgeführte SQL-Code bis auf Parameter fest. Deshalb können für die SQL-Befehle zur Compilezeit optimierte Zugriffspläne erstellt werden und in sogenannte Packages gespeichert werden. Um solche eine Stored Procedure nutzen zu können, benötigt man lediglich die EXECUTE-Rechte auf der Stored Procedure. Im Gegensatz dazu ist bei Programmen, die dynamisches SQL verwenden zur Programmierzeit nicht klar, welche SQL-Befehle beim Aufruf des Programms ausgeführt werden. Aus diesem Grund kann hier im Voraus kein Package und der Zugriffsplan damit erst zur Laufzeit erstellt werden. Deshalb benötigen Benutzer der Stored Procedure neben dem EXECUTE-Recht die Rechte für jeden auszuführenden Befehl in der Stored Procedure. Unter Verwendung von statischem SQL hat man demnach einen Geschwindigkeitsvorteil zur Ausführungszeit, da ein Zugriffsplan schon vorhanden ist. Allerdings ist zu beachten, dass der optimale Zugriffsplan, der im Package gespeichert ist, nach Datenänderungen evtl. nicht mehr optimal ist. Deshalb sollten die Zugriffspläne regelmäßig erneuert werden.

Bei External high-level language Stored Procedures steht der Quellcode der Stored Procedure nicht beim DDL-Teil der Stored Procedure, da er sprachspezifisch verarbeitet werden muss. Im DDL-Teil wird auf das auszuführende Programm verwiesen. Die Ausführung einer Stored Procedure läuft hier nicht im Adressraum von DB2. Ein eigener Adressraum muss für sie durch einen Workloadmanager aufgebaut werden. Die genaue Umsetzung soll in folgendem Tutorial gezeigt werden, in dem Stored Procedures unter der Verwendung der Programmiersprache Java erstellt werden.

5.2 Erstellung von Java Stored Procedures (Tutorial)

5.2.1 Java Stored Procedures

Im Folgenden sollen Stored Procedures unter Verwendung der Programmiersprache Java erstellt werden, die es ermöglichen, einen schon feststehenden Meister sowie bereits feststehende Absteiger der Bundesliga auszugeben. Hierzu soll sowohl statisches SQL unter Verwendung von SQLJ als auch dynamisches SQL über die JDBC-Schnittstelle zum Einsatz kommen. Über JDBC können SQL-Befehle dynamisch zur Laufzeit ausgeführt werden. SQL-Befehle unter Verwendung von SQLJ müssen zum Zeitpunkt der Programmierung bis auf die Ausprägung der Prädikate feststehen und können sich zur Laufzeit nicht mehr ändern.

Während bei der Einrichtung von Stored Procedures, die JDBC verwenden, lediglich ein Kompilierschritt nötig ist, muss die Stored Procedure bei der Verwendung von SQLJ vorkompiliert

und kompiliert, außerdem ein Profil angepasst und die Stored Procedure gebunden werden (vgl. Abbildung 6). Ziel ist es, einen optimierten Zugriffsplan zu erstellen und im Datenbanksystem zu speichern. Bei der Verwendung von JDBC ist dies zur Übersetzungszeit nicht möglich, da dynamisches SQL ausgeführt wird, d.h. es steht erst zur Laufzeit fest, was für SQL-Befehle ausgeführt werden sollen. Hier behilft man sich dadurch, dass man den zur Laufzeit erstellten Zugriffsplan in einem Cache speichert. Spätestens nach einem Systemneustart sind diese Zugriffspläne allerdings wieder gelöscht und müssen beim Aufruf der Stored Procedure neu erstellt werden.

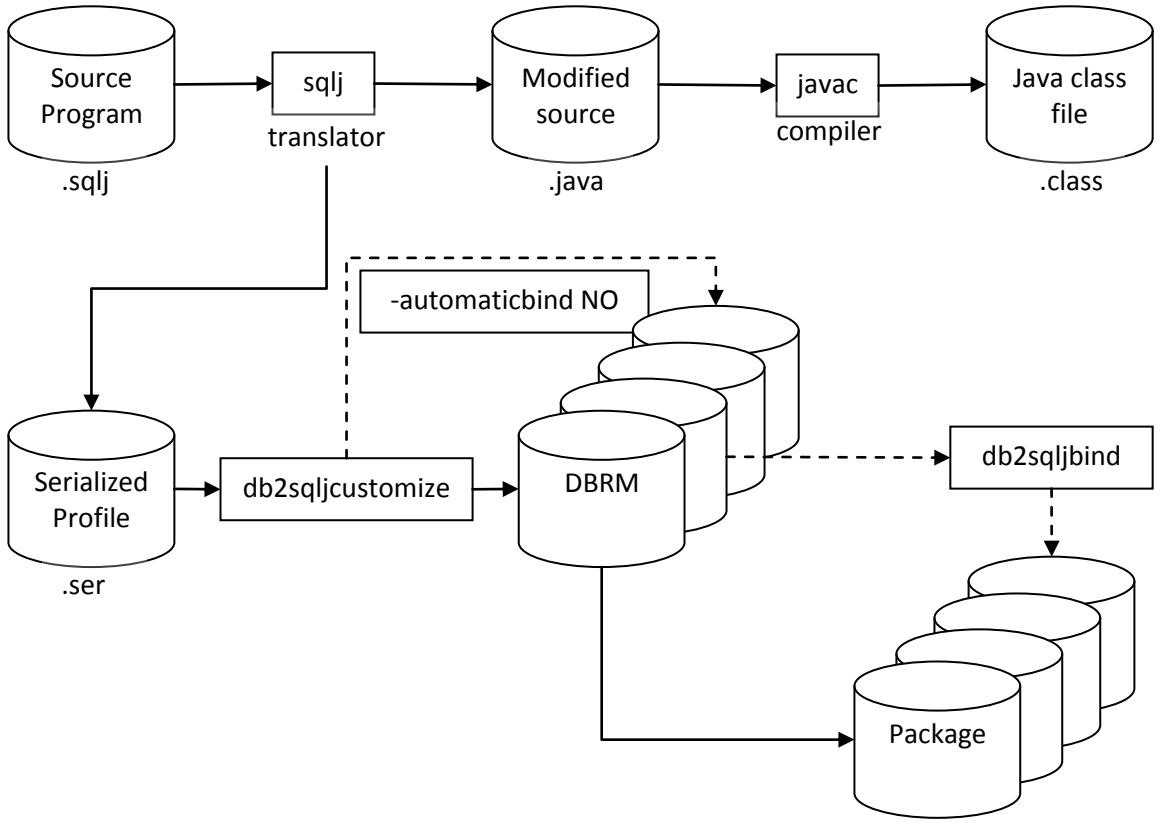


Abbildung 6: Schritte für die Vorbereitung von Stored Procedures unter Verwendung von SQLJ

Da der javac-Compiler die SQLJ-Syntax nicht versteht, muss der Programmcode mit den eingebetteten SQL-Befehlen zuvor vorkompiliert werden. Der für Java verwendete Precompiler heißt SQLJ-Translator und übernimmt folgende Aufgaben:

- Er sucht alle SQL-Anweisungen und setzt sie in entsprechende Java-Sprachelemente um. Dabei wird der SQL-Code auskommentiert und durch Java-Programmcode ersetzt.
- Die Informationen der gefundenen SQL-Anweisungen werden in einer separaten Datei gespeichert. Diese Dateien werden üblicherweise Bind-Files genannt bzw. im Javaumfeld SQLJ-Profiles.

Der neue Java-Programmcode kann durch den Standard-Java-Compiler (javac) zu Java Bytecode umgewandelt werden.

Der DB2 SQLJ Profile Customizer generiert aus den dem vom SQLJ-Precompiler erstellten Profil ein DB2-spezifisches Ergebnis. Ihm werden deshalb beim Aufruf unter anderem Informationen über das Datenbanksystem (URL zum Datenbanksystem, Benutzerkennung und Passwort) mitgegeben. Das Ergebnis sind 4 Database Request Module (DBRM) – eines für jeden Isolation Level. Aus diesen DBRMs wird ein Package erstellt, das dann den Zugriffsplan für die enthaltenen SQL-Befehle enthält. Wird beim Aufruf von db2sqljcustomize der Parameter automaticbind NO angegeben, muss dieser Schritt über den Befehl db2sqljbind manuell durchgeführt werden.

Im Folgenden soll eine Java Stored Procedure MEISTER unter der Verwendung von dynamischen SQL über JDBC erstellet werden, die berechnet, ob die Meisterschaft in der laufenden Saison schon entschieden ist, so wie eine Stored Procedure ABSTEIGER, die statisches SQL enthält und als Ergebnis die bereits feststehenden Absteiger ausgibt.

External high-level language Stored Procedures werden in einem externen Adressraum ausgeführt und nicht in einem der Adressräume von DB2. Der hierfür benötigte Adressraum wird von einem Workloadmanager gestartet und zur Verfügung gestellt. Der Workloadmanager für Java Stored Procedures heißt auf dem Tübinger und Leipziger Mainframe WLMD931J. Abbildung 7 zeigt den Aufruf der Java Stored Procedures in diesem Tutorial.

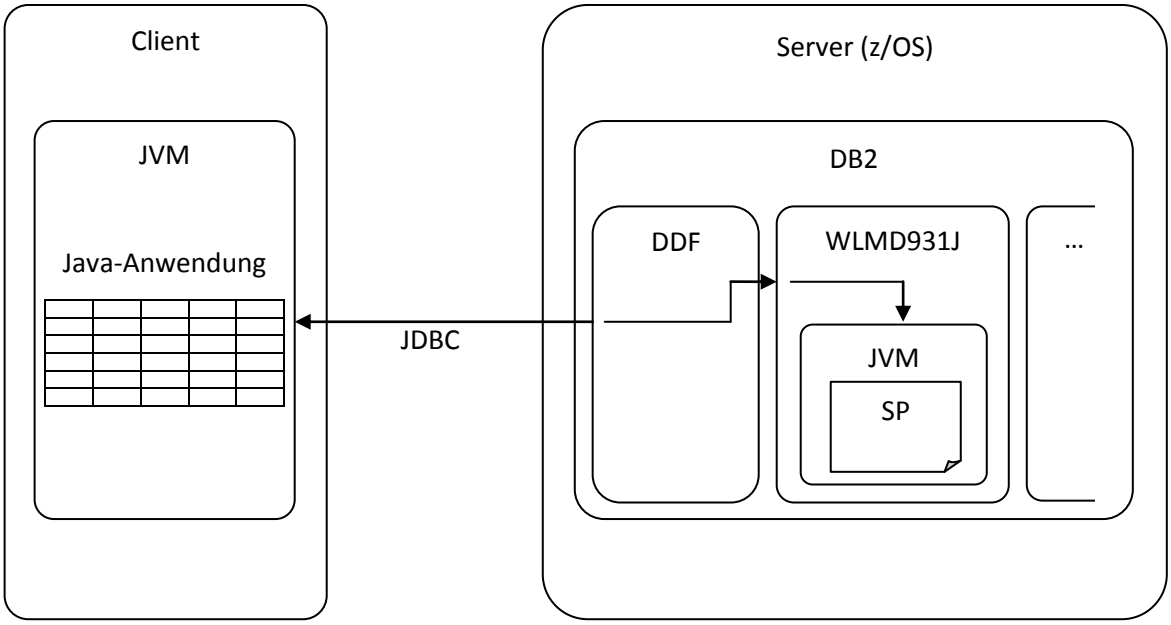
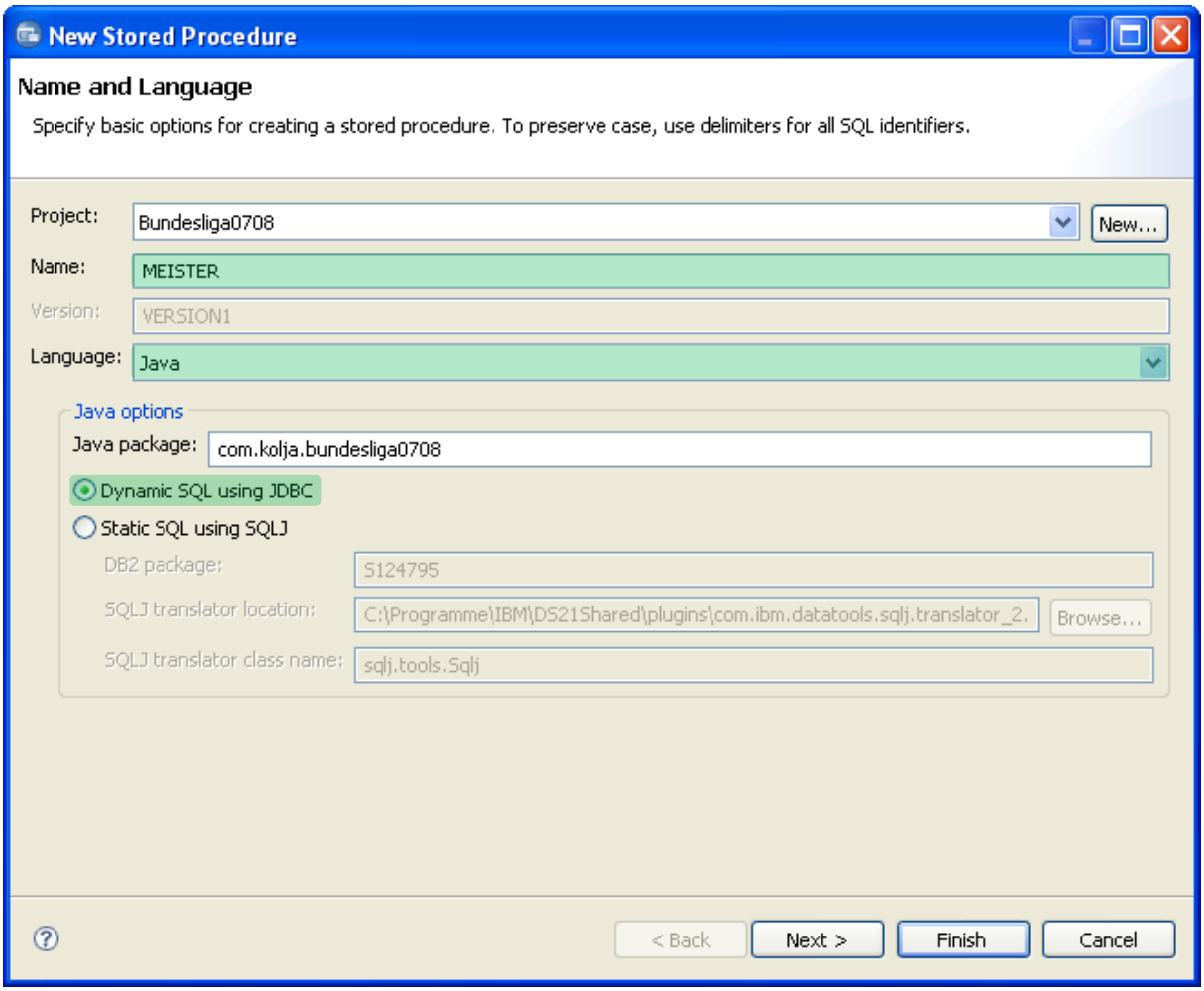


Abbildung 7: Überblick über den Aufruf der Java Stored Procedures in diesem Tutorial

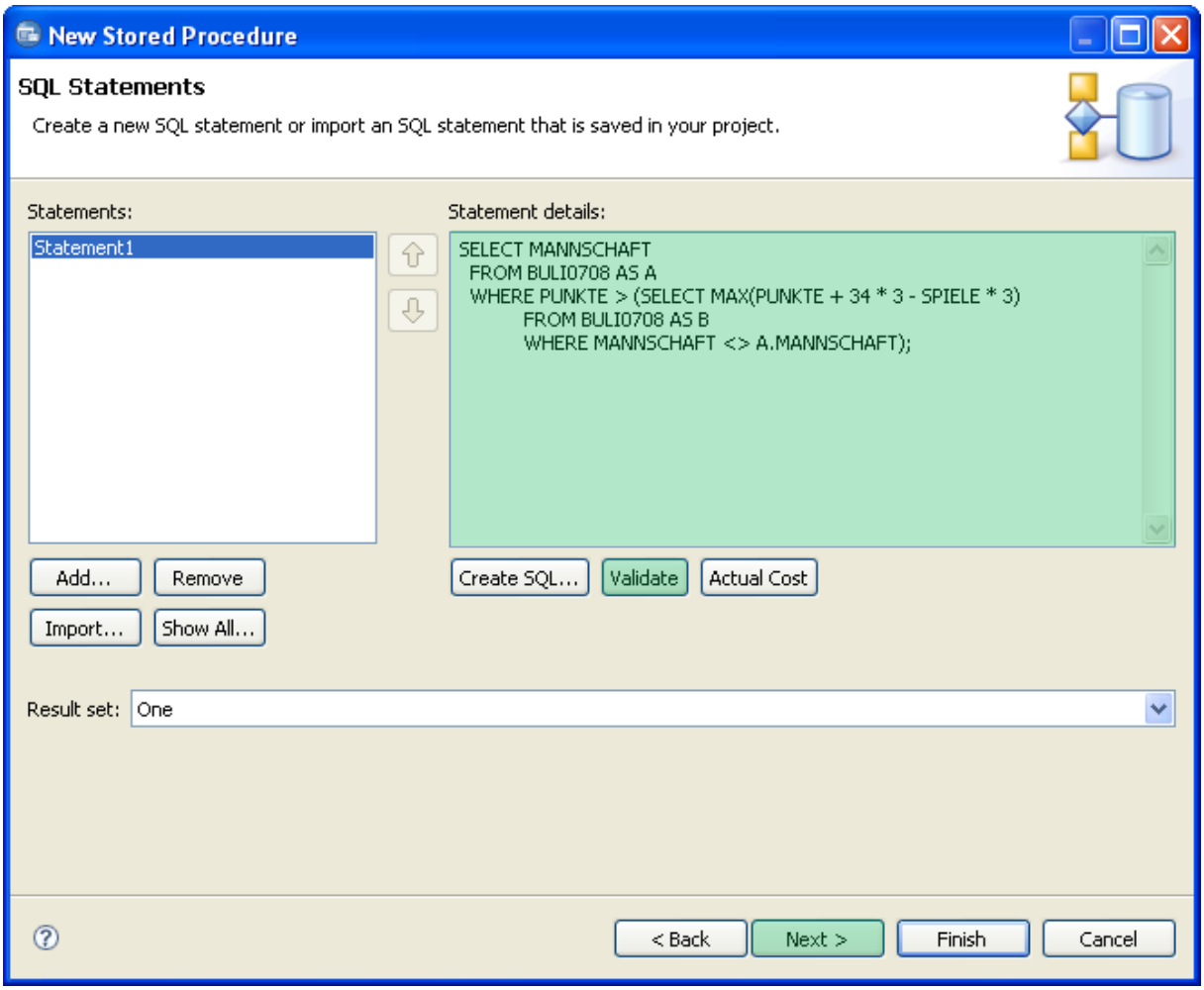
Durch einen Klick mit der rechten Maustaste auf den Ordner Stored Procedures des Projekts im Data Project Explorer erscheint das schon bekannte Kontextmenü. Wählen Sie „New“ und dann „Stored Procedure“. Es startet sich der Dialog zum Erstellen einer neuen Stored Procedure.



Die Stored Procedure soll in Java geschrieben sein und dynamisches SQL enthalten. Deshalb wird unter Language „Java“ eingetragen und in den Java Options „Dynamic SQL using JDBC“ aktiviert. Durch einen Klick auf „Next“ erreichen Sie das Fenster zur Befehlseingabe. Folgender SQL-Befehl ermittelt den feststehenden Meister der aktuellen Fußball-Bundesligatabelle:

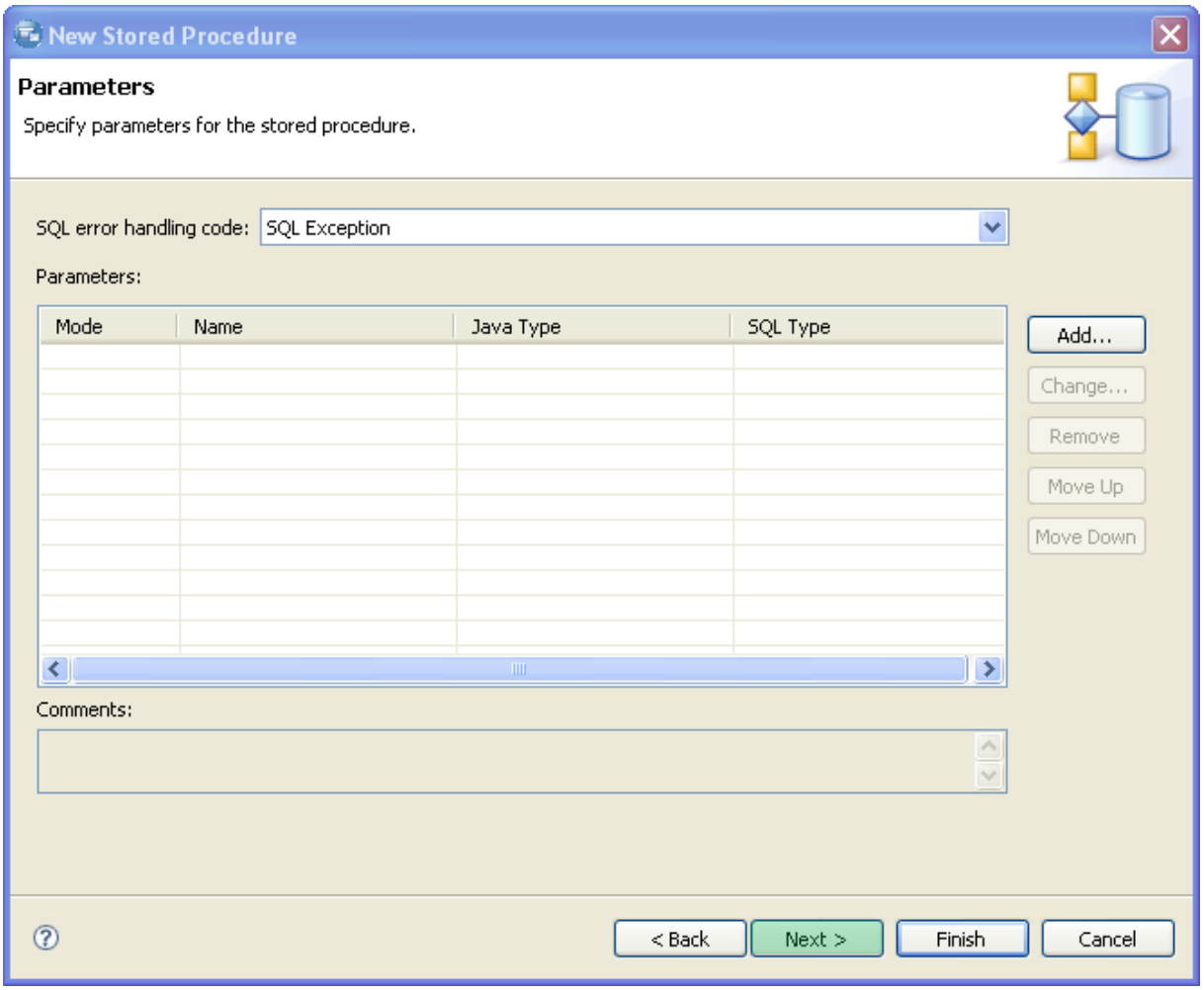
```
SELECT MANNSCHAFT
  FROM BULI0708 AS A
  WHERE PUNKTE > (SELECT MAX(PUNKTE + 34 * 3 - SPIELE * 3)
            FROM BULI0708 AS B
            WHERE MANNSCHAFT <> A.MANNSCHAFT);
```

Damit dieser Befehl ausgeführt wird, muss der bisherige Standardbefehl im Feld „Statement details“ durch dieses ersetzt werden. Durch einen Klick auf „Validate“ lässt sich die Syntax des Befehls überprüfen.

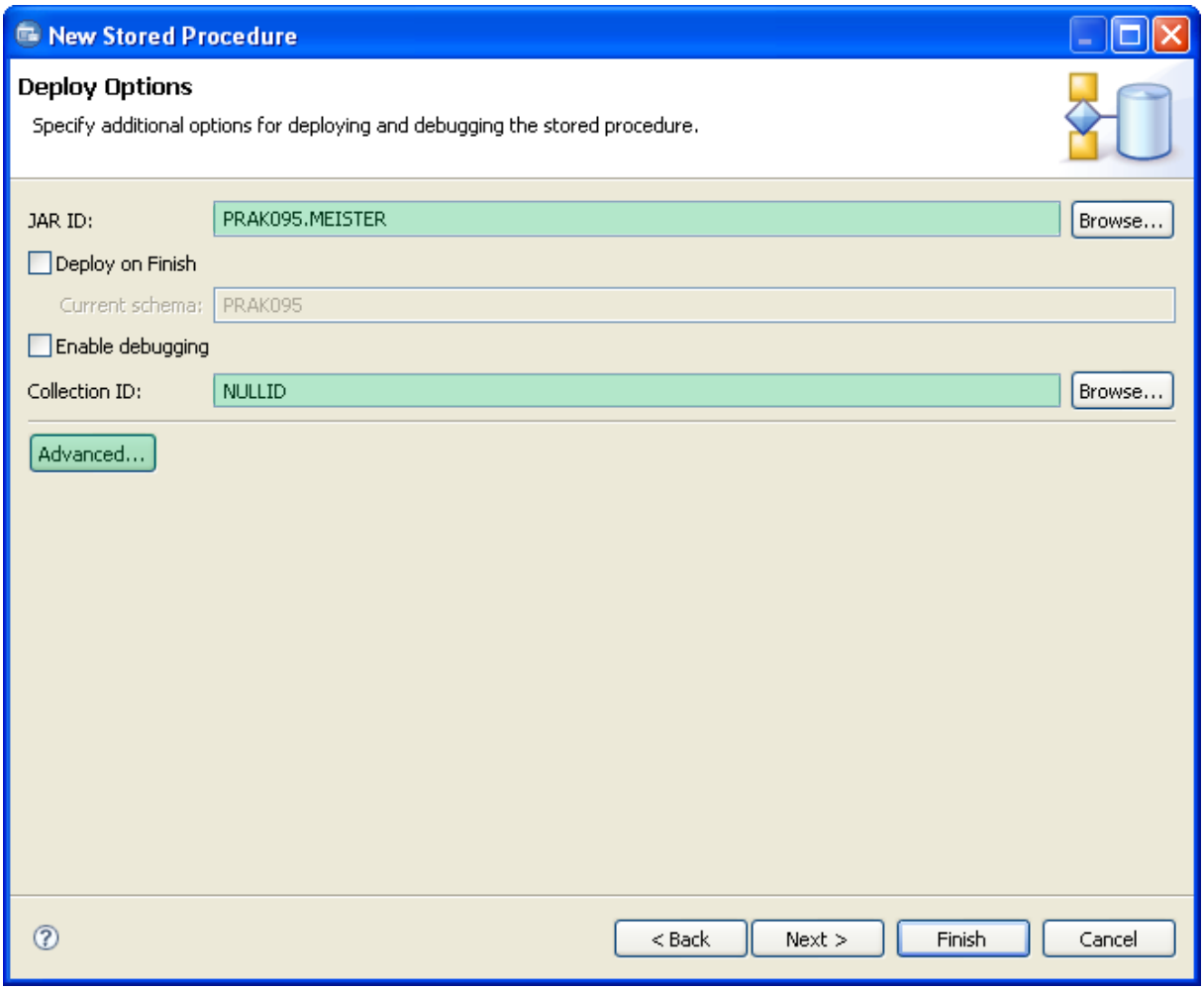


Das Ergebnis der Überprüfung wird in einem Popup-Fenster angezeigt.

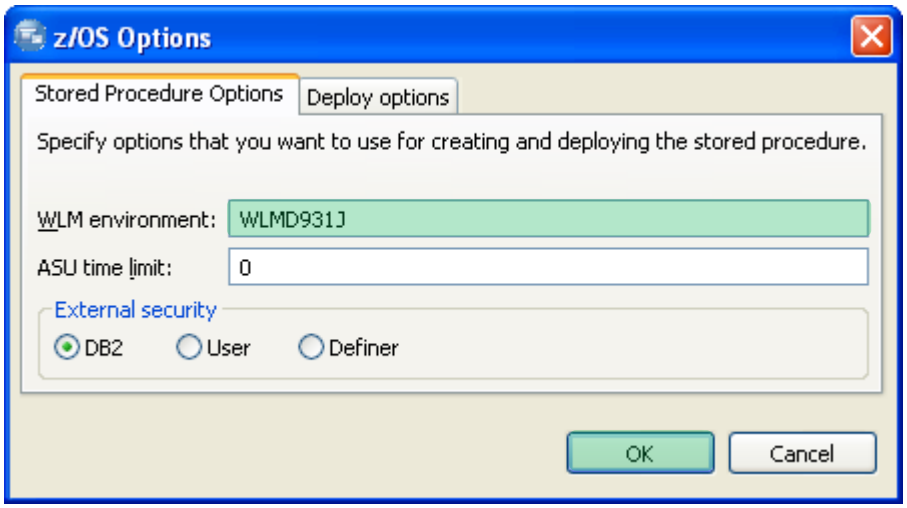
Durch einen Klick auf „Next“ haben Sie nun die Möglichkeit, Ein-/Ausgabeparameter anzugeben. Da die Stored Procedure dies nicht benötigt, klicken Sie auf „Next“.



Im folgenden Fenster müssen noch einige mainframespezifische Anpassungen vorgenommen werden. Der Zugriff über JDBC erfordert serverseitig einige gebundene Packages. Diese Packages sind schon eingerichtet, liegen allerdings in der Collection mit der ID NULLID. Deshalb muss im Feld Collection ID hier NULLID angeben werden.

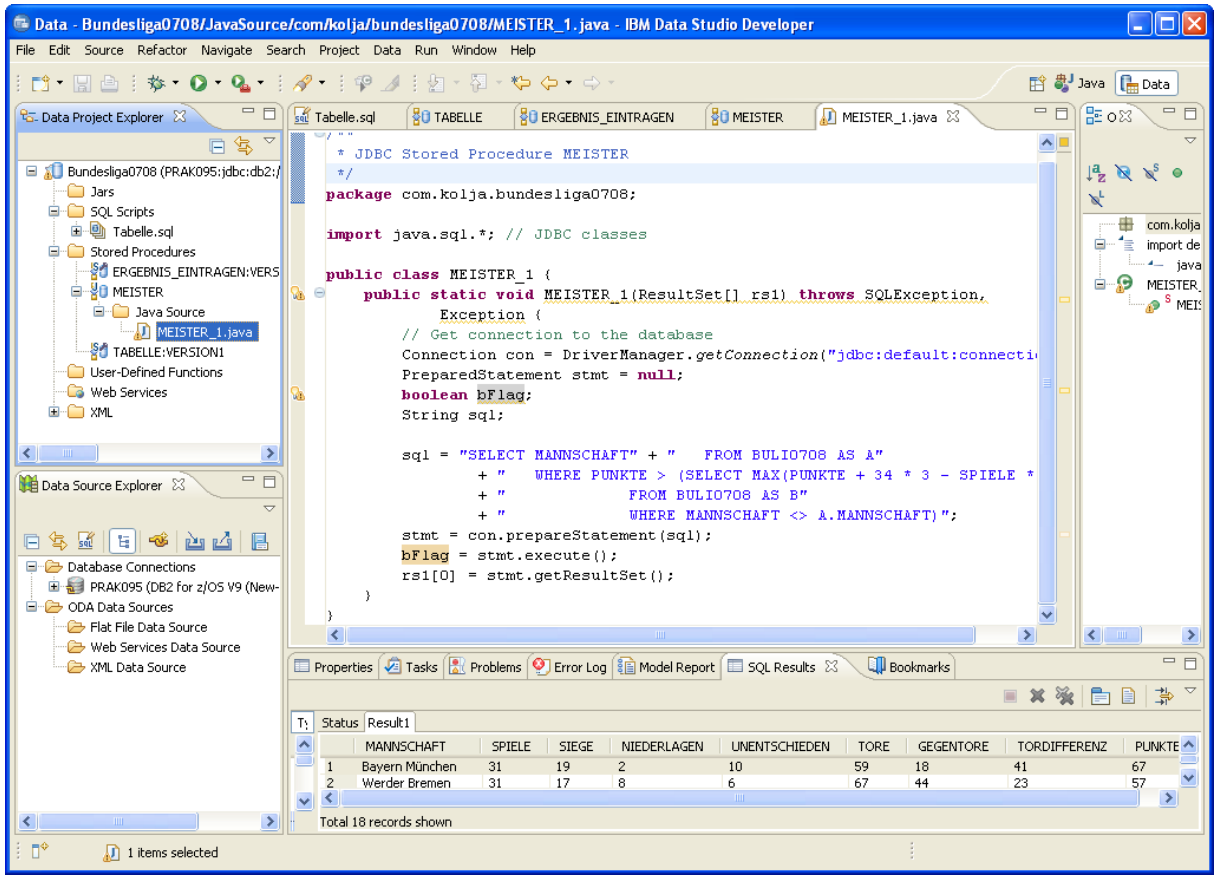


External high-level language Stored Procedures werden in einem externen Adressraum ausgeführt und nicht in einem der Adressräume von DB2. Der hierfür benötigte Adressraum wird von einem Workloadmanager gestartet und zur Verfügung gestellt. Der Workloadmanager für Java Stored Procedures heißt auf dem Tübinger und Leipziger Mainframe WLMD931J. Um ihn zu verwenden, muss er unter Advanced im Feld WLM environment eingetragen werden.

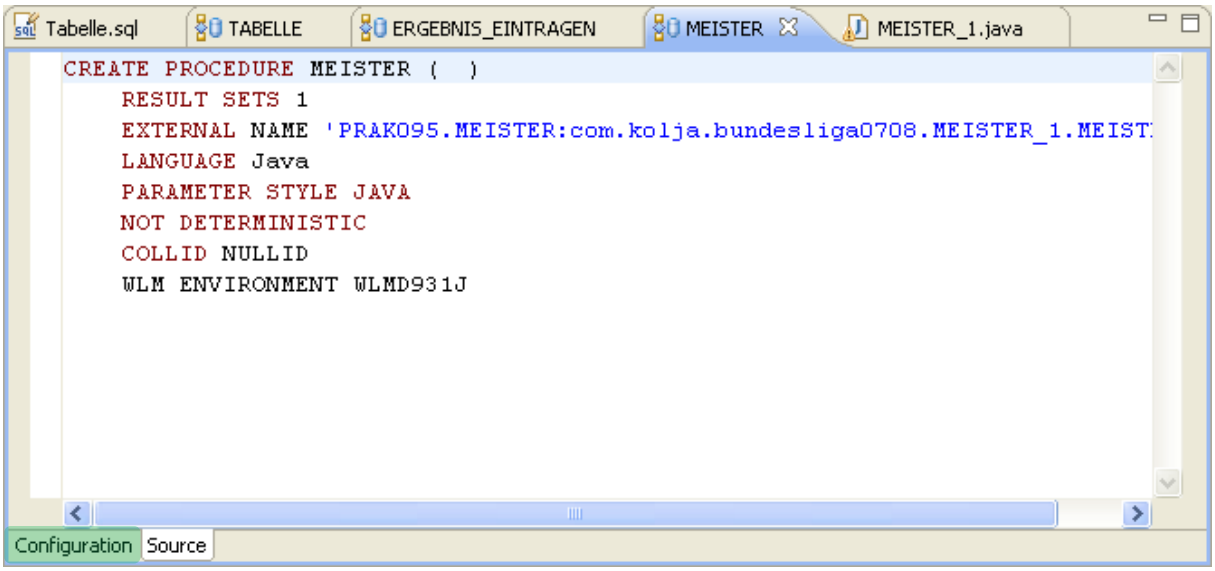


Durch einen Klick auf „Finish“ beendet sich der Dialog zur Erstellung von Stored Procedures und der DDL- und Programmcode wird erstellt. Indem man den Ordner Java Source aufklappt, erkennt man,

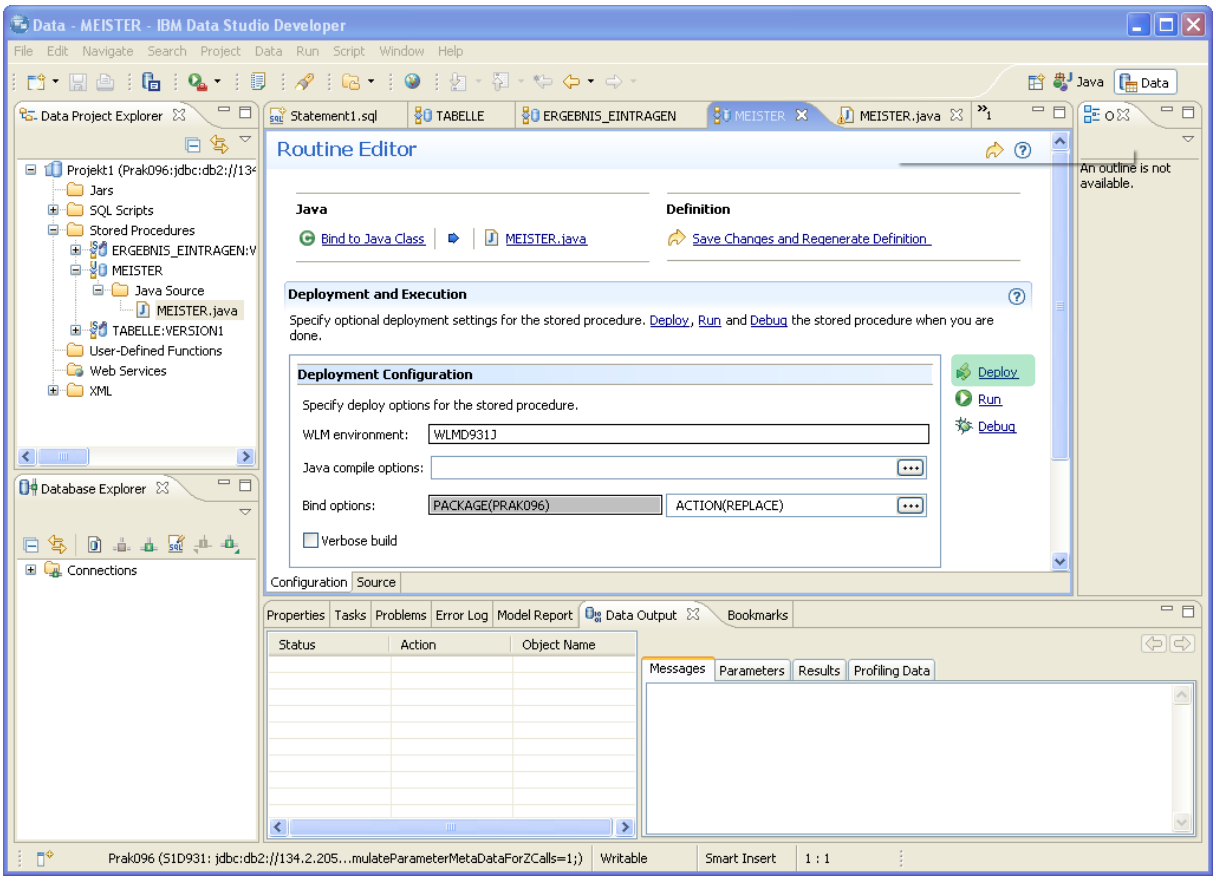
dass eine normale .java-Datei erstellt wurde, die den SQL-Befehl in Form eines Prepared Statements ausführen wird.

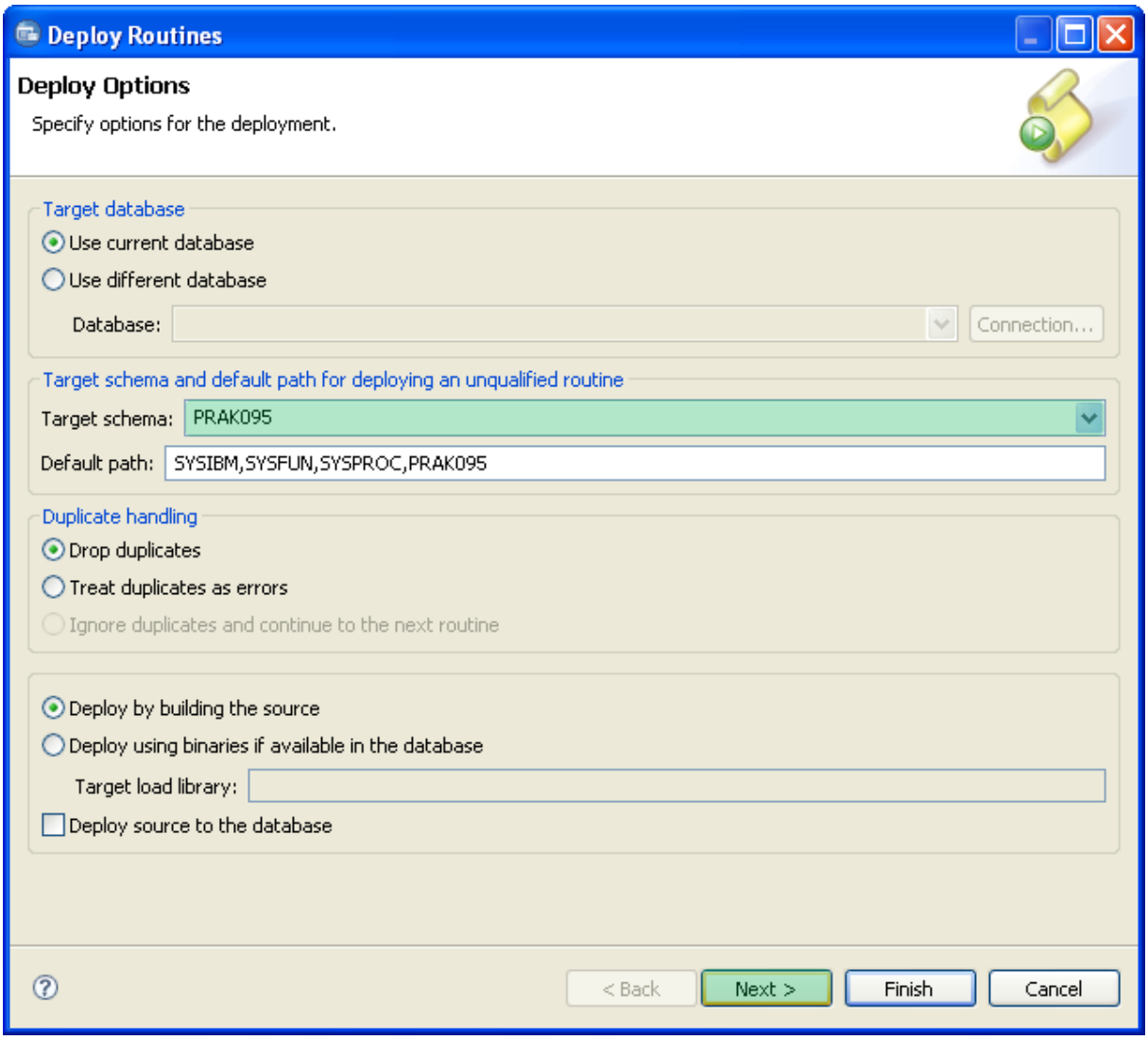


Im DDL-Teil der Stored Procedure finden Sie die eingestellten Parameter der Stored Procedure wieder (Sprache (LANGUAGE), Collection ID (COLLID), Workloadmanager (WLM EINVIRONMENT)) sowie den Namen der Javaklasse und die aufzurufende Methode (mEISTER).

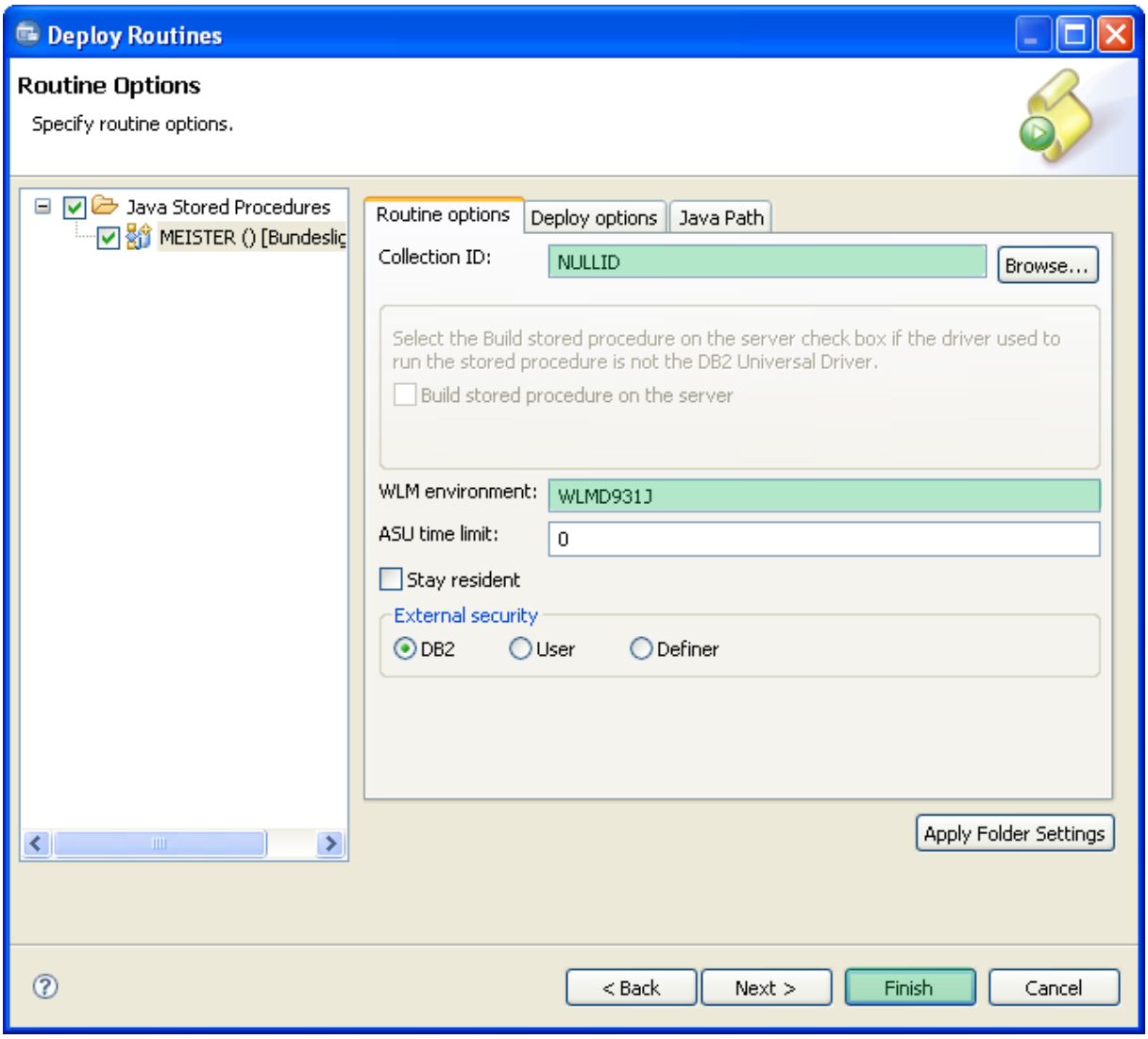


Um die Stored Procedure dem entfernten DB2-System bekannt zu machen, klicken Sie wieder auf „Deploy“ in der Konfigurationsansicht.

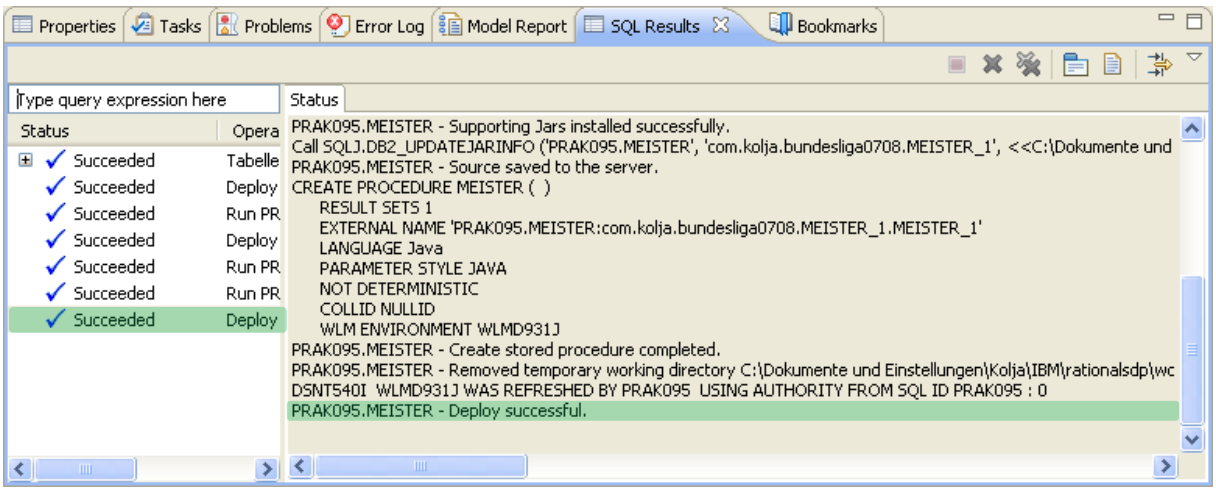

Data - MEISTER - IBM Data Studio Developer
File Edit Navigate Search Project Data Run Script Window Help
Java
Data
Data Project Explorer
Projekt1 (Prak096:jdbc:db2://13
Jars
SQL Scripts
Stored Procedures
ERGEBNIS_EINTRAGEN:V
MEISTER
Java Source
MEISTER.java
TABELLE:VERSION1
User-Defined Functions
Web Services
XML
Database Explorer
Connections
Statement1.sql
TABELLE
ERGEBNIS_EINTRAGEN
MEISTER
MEISTER.java
Routine Editor
Java
Bind to Java Class
MEISTER.java
Definition
Save Changes and Regenerate Definition
Deployment and Execution
Specify optional deployment settings for the stored procedure. Deploy, Run and Debug the stored procedure when you are done.
Deployment Configuration
Specify deploy options for the stored procedure.
WLM environment:
WLMD931J
Java compile options:
Bind options:
PACKAGE(PRAK096)
ACTION(REPLACE)
Verbose build
Deploy
Run
Debug
Configuration
Source
An outline is not available.
Properties
Tasks
Problems
Error Log
Model Report
Data Output
Bookmarks
Status
Action
Object Name
Messages
Parameters
Results
Profiling Data
Prak096 (S1D931: jdbc:db2://134.2.205...mulateParameterMetaDataForZCalls=1;)
Writable
Smart Insert
1 : 1



Überprüfen Sie, ob das richtige Schema angegeben ist und klicken Sie dann auf „Next“.



Überprüfen Sie, ob die Angaben zu Collection ID und Workloadmanager richtig übernommen wurden und klicken Sie auf Finish.

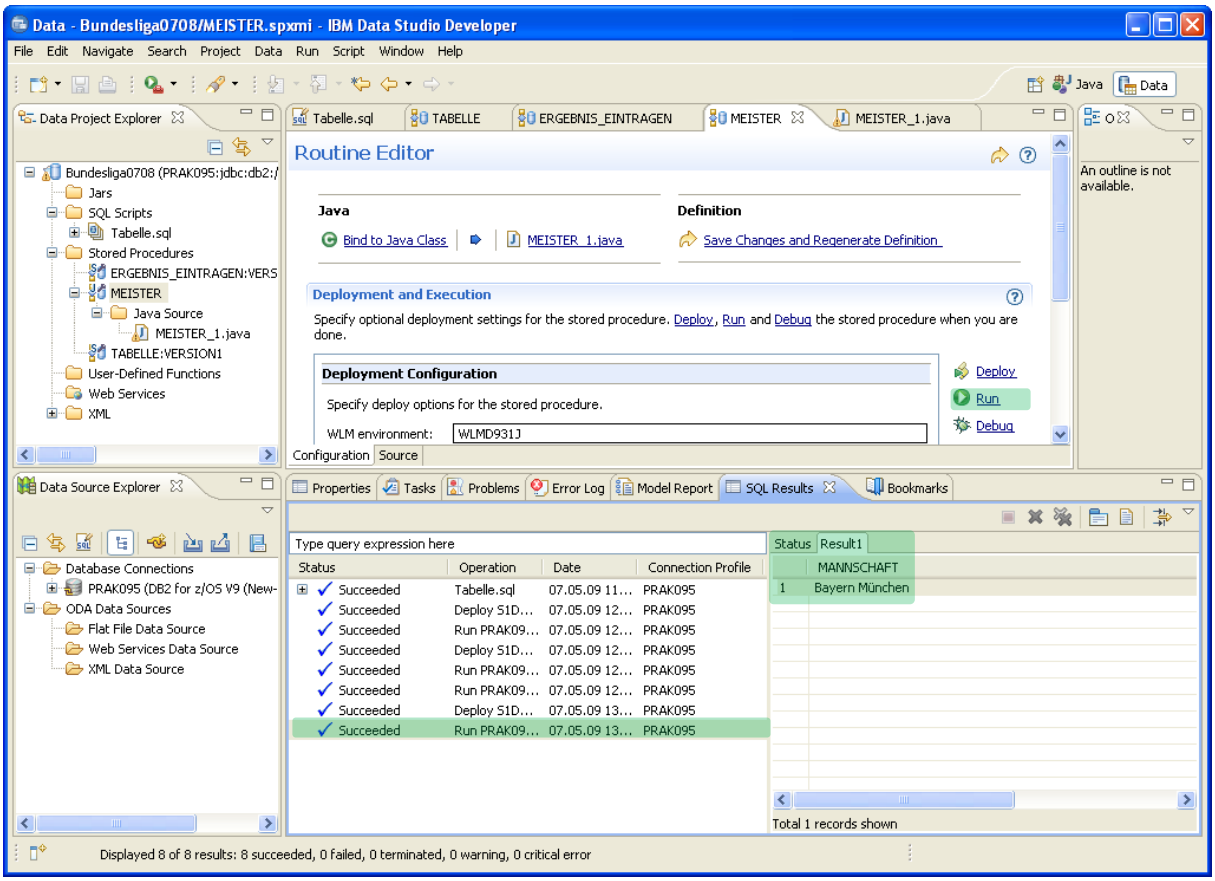


Im SQL Results-Fenster sehen Sie das Ergebnis.

Man erkennt grob die folgenden vorgenommenen Schritte:

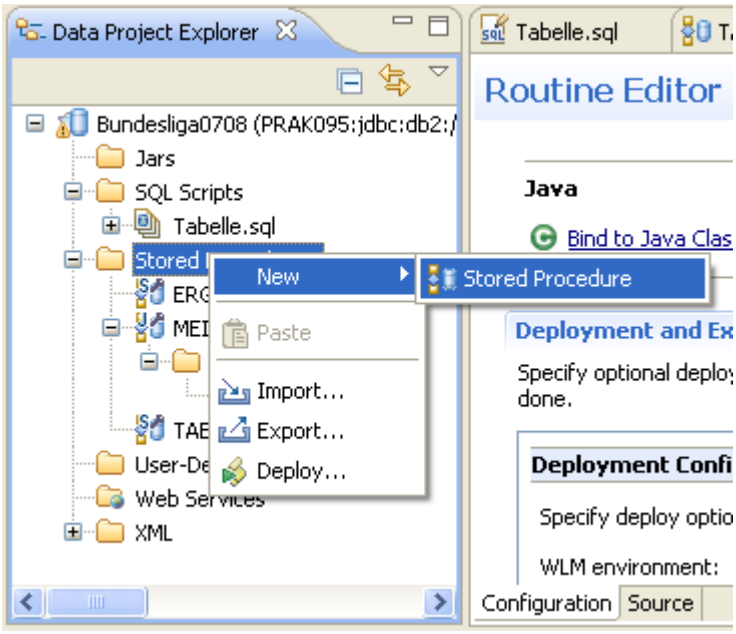
- Kompilierung der Java-Datei
- Erstellung eines Jar-Files aus den kompilierten Dateien
- Löschen möglicherweise schon vorhandener Jar-Files, die zur gleichen Stored Procedure gehören
- Installation des Jar-Archivs im DB2-System
- Ausführen der DDL

Nun können Sie durch einen Klick auf „Run“ die Stored Procedure ausführen. Da in der Bundesligasaison 2007/2008 schon nach dem 31. Spieltag Bayern München als Meister feststand, erhalten Sie diese Mannschaft als Ergebnis des Aufrufs.

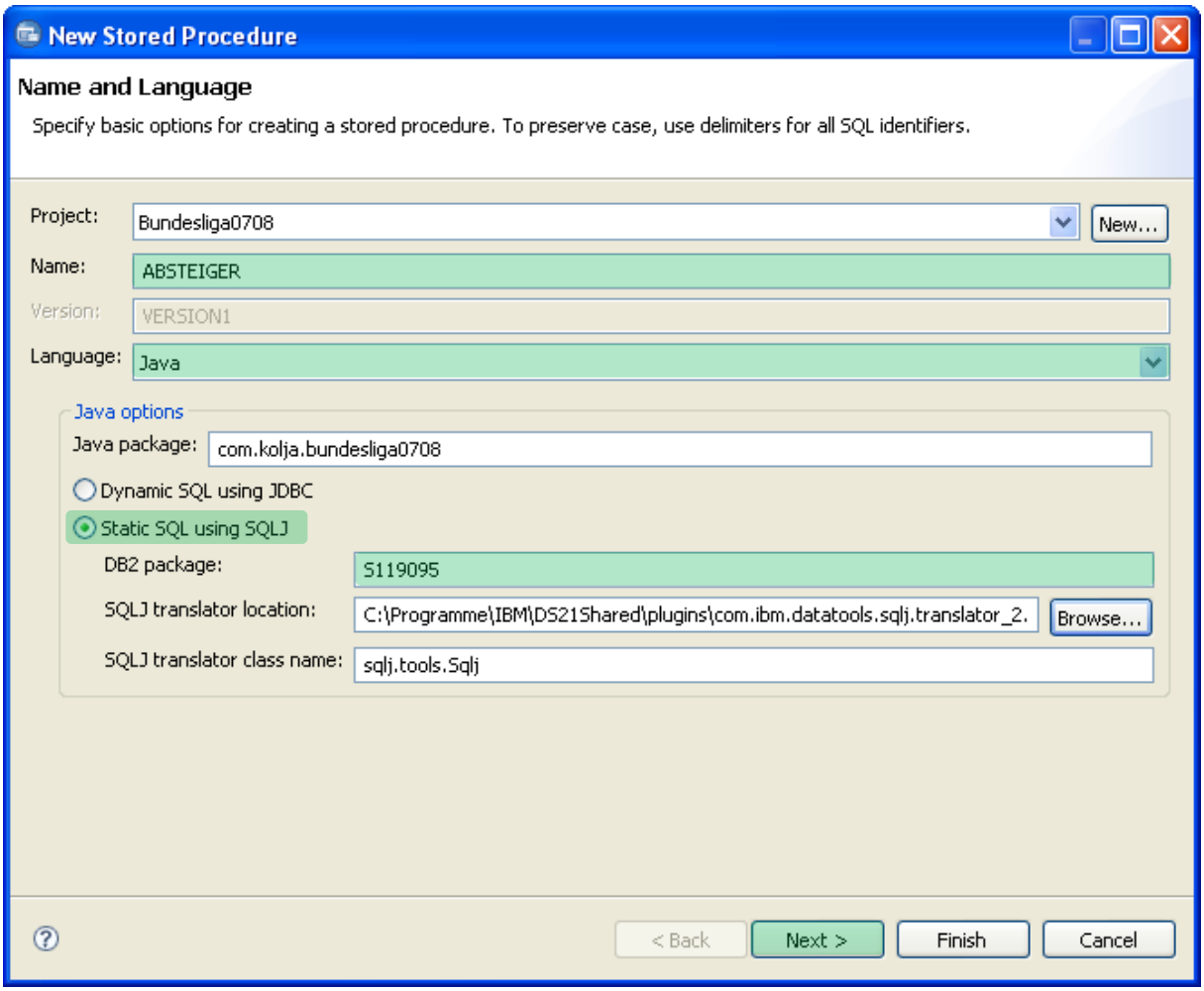


Nun soll noch eine Stored Procedure erstellt werden, die statischen SQL-Code unter Verwendung von SQLJ enthält. Sie soll ABSTEIGER heißen und feststehende Absteiger der Bundesligasaison 2007/2008 zurückgeben.

Der Dialog zum Erstellen einer neuen Stored Procedure wird wieder durch einen Klick mit der rechten Maustaste auf das Verzeichnis „Stored Procedures“ des Datenentwicklungsprojekts ausgelöst.



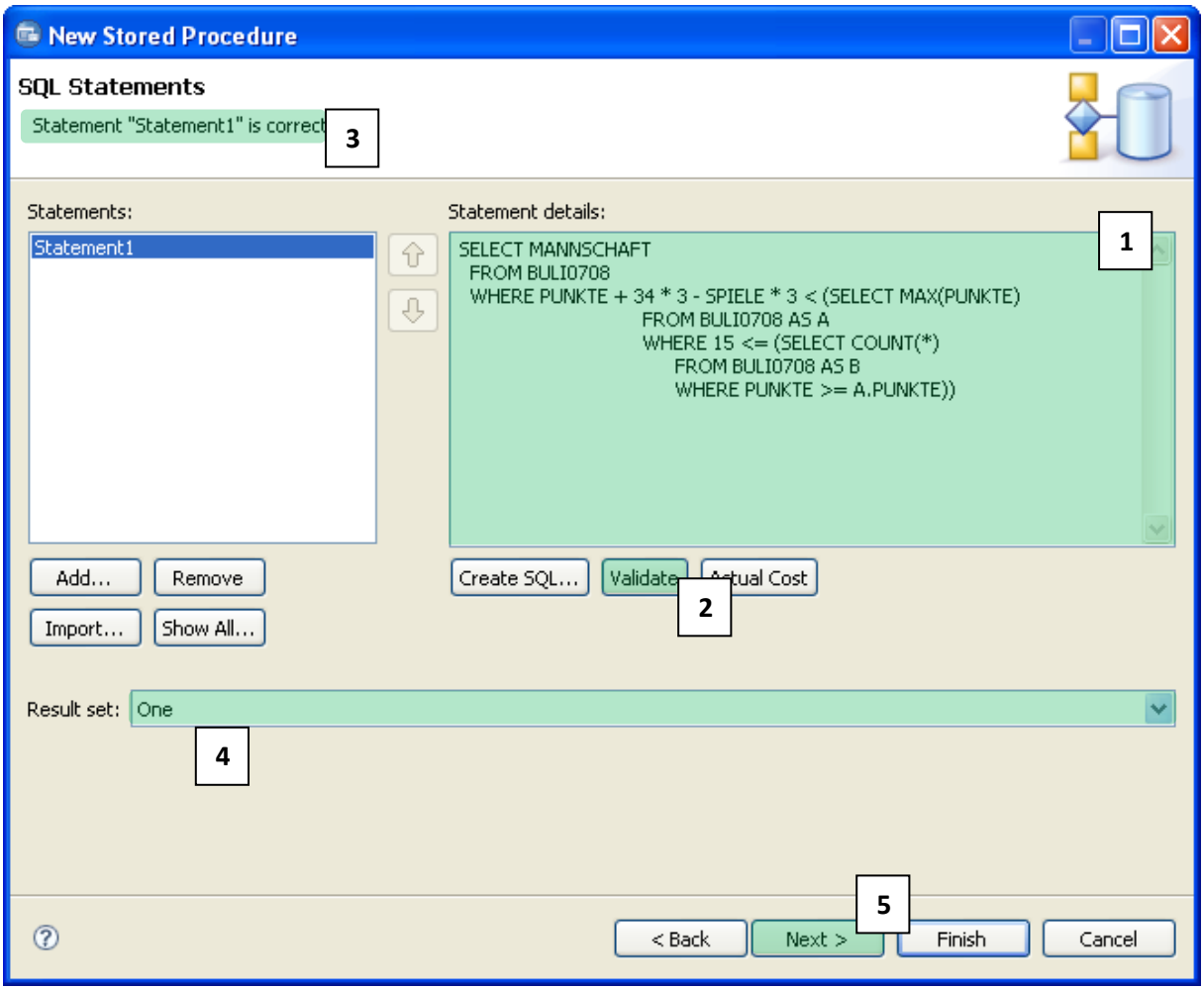
Als Namen der Stored Procedure wird ABSTEIGER eingetragen und als Programmiersprache Java ausgewählt. Dieses Mal soll SQLJ zum Einsatz kommen, weshalb wir „Static SQL using SQLJ“ aktivieren. Um Konflikte mit anderen Packages zu vermeiden werden die letzten drei Zeichen im Namen des DB2-Packages durch die letzten drei Zeichen Ihrer Benutzerkennung ersetzt (hier 096). Bitte achten Sie darauf, dass der Name des Packages nicht mehr als 7 Zeichen enthält.



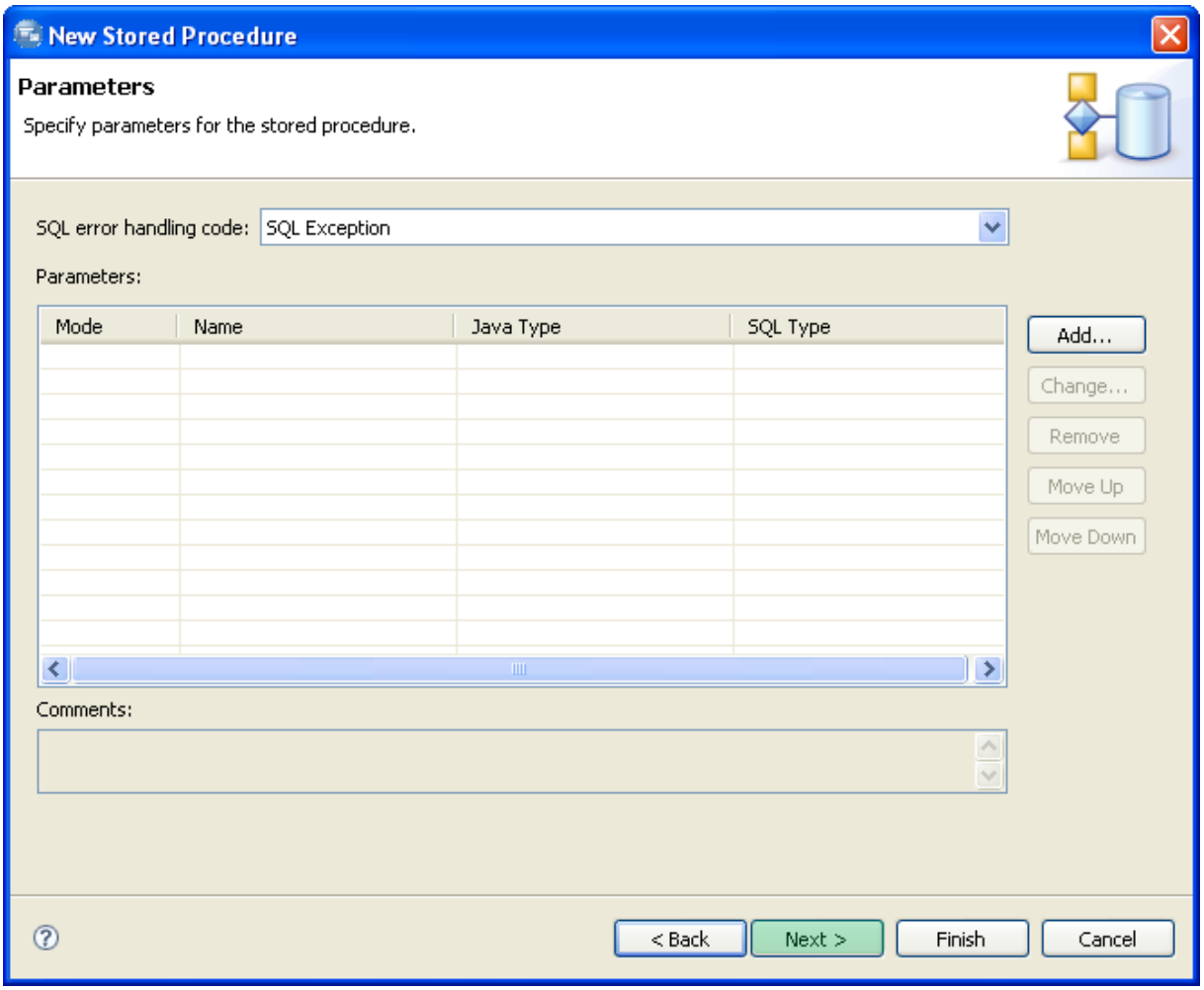
Im nächsten Dialogfenster fügen Sie folgenden SQL-Befehl in das Textfeld „Statement details“ ein, das die feststehenden Absteiger zurückgibt:

```
SELECT MANNSCHAFT
  FROM BULI0708
  WHERE PUNKTE + 34 * 3 - SPIELE * 3 < (SELECT MAX(PUNKTE)
                                        FROM BULI0708 AS A
                                        WHERE 15 <= (SELECT COUNT(*)
                                              FROM BULI0708 AS B
                                              WHERE PUNKTE >= A.PUNKTE))
```

Durch einen Klick auf „Validate“ wird die Syntax des Befehls überprüft und die Korrektheit in der Statusleiste angezeigt. Bevor Sie auf „Next“ klicken, geben Sie an, dass der Rückgabewert der Stored Procedure aus einer Ergebnismenge besteht, indem Sie im Feld „Result set“ „One“ auswählen.

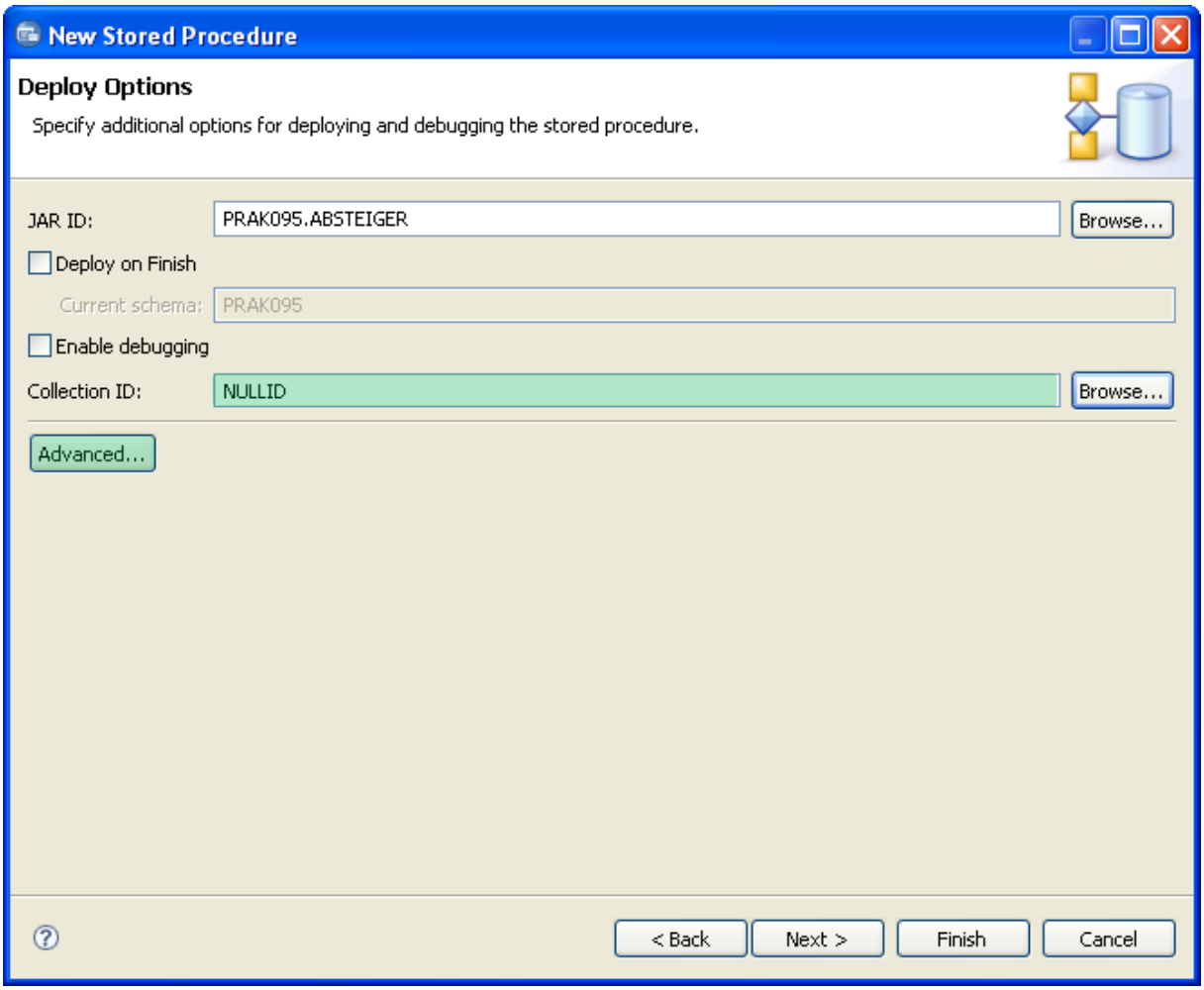


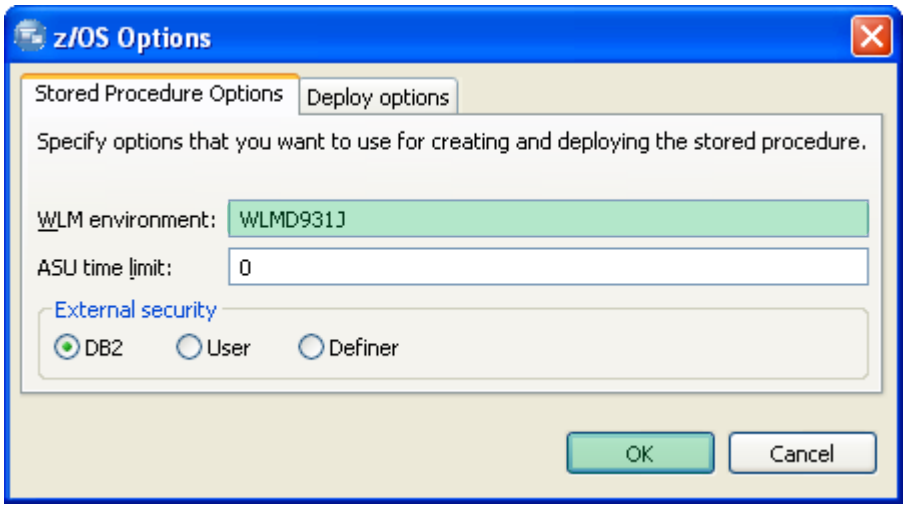
Da die Stored Procedure keine Eingabe-/Ausgabeparameter besitzen soll, klicken Sie auf „Next“.



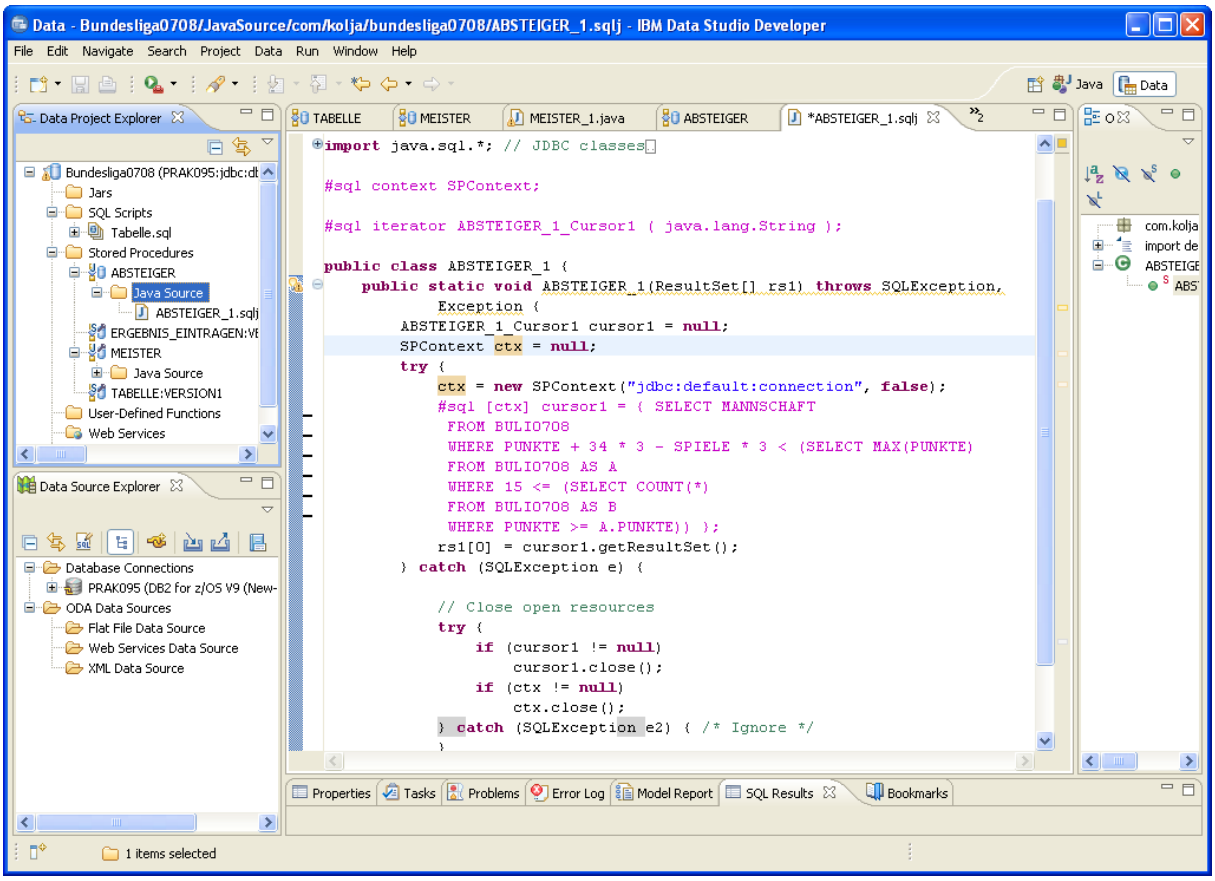
Im nächsten Dialogfenster wird als Collection ID wieder NULLID ausgewählt. Unter „Advanced“ wird außerdem WLMD931J als zu verwendeten Workloadmanager eingetragen.

Durch einen Klick auf „OK“ und anschließend „Finish“ beendet sich der Dialog und die Stored Procedure wird erstellt.

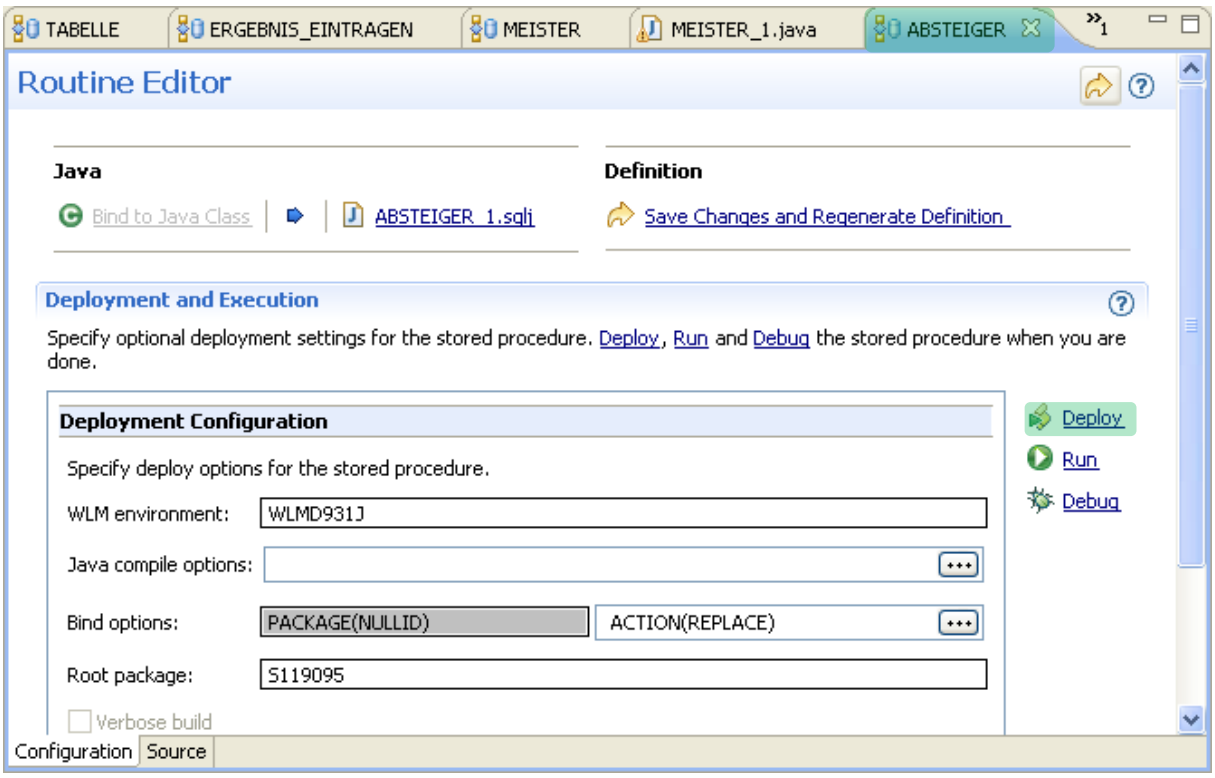




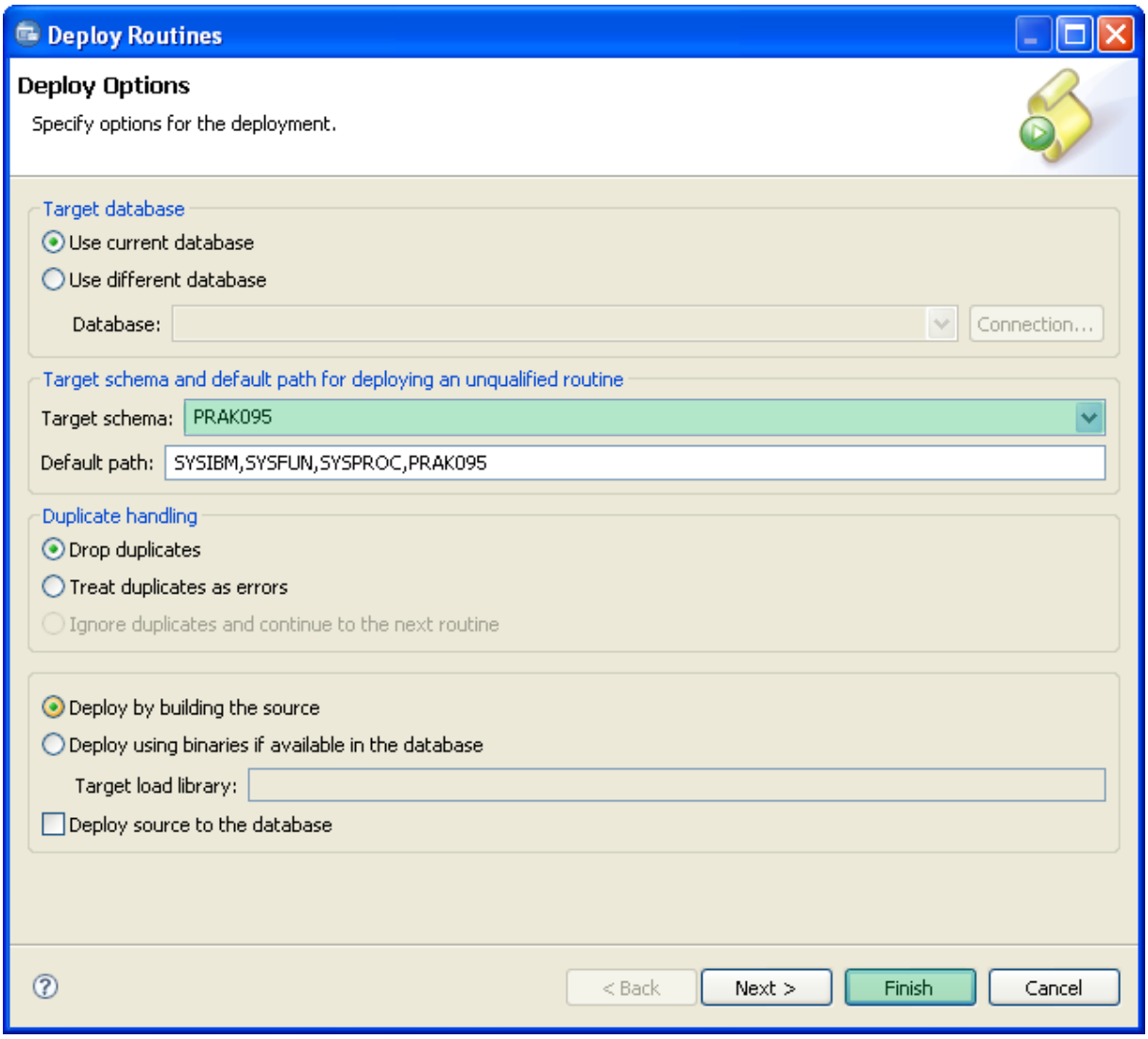
Indem Sie die Stored Procedure im Data Project Explorer aufklappen, können Sie den generierten Source Code durch einen Doppelklick auf ABSTEIGER.sqlj betrachten. Dieses Mal wurde eine .sqlj-Datei erstellt, in der der SQL-Befehl als Anweisungen für den Precompiler steht (gekennzeichnet durch die vorausgehende Raute).



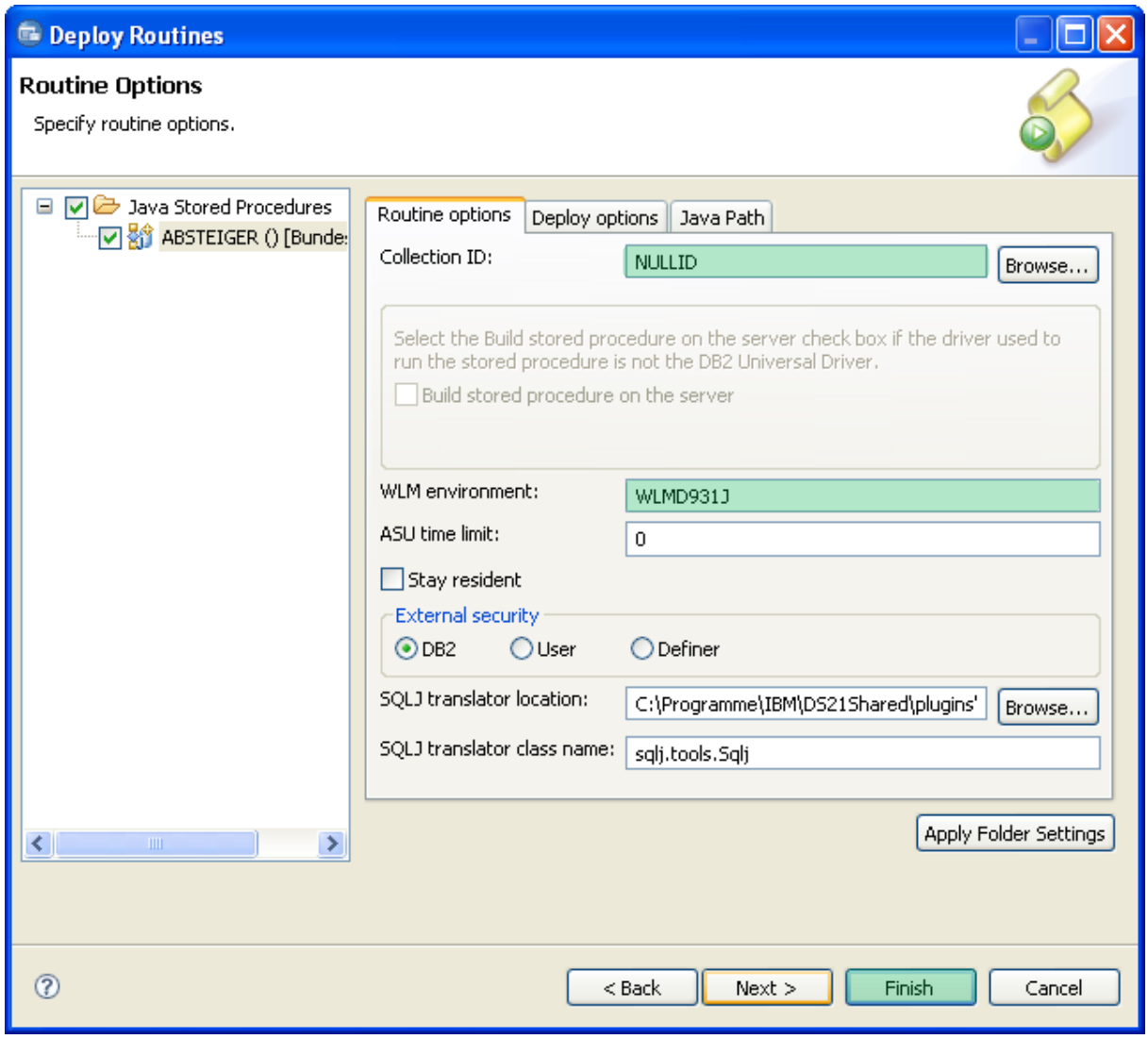
Wie üblich wird im Configuration-Fenster bei der DDL auf „Deploy“ geklickt, um die Stored Procedure auf dem Mainframe einzurichten.



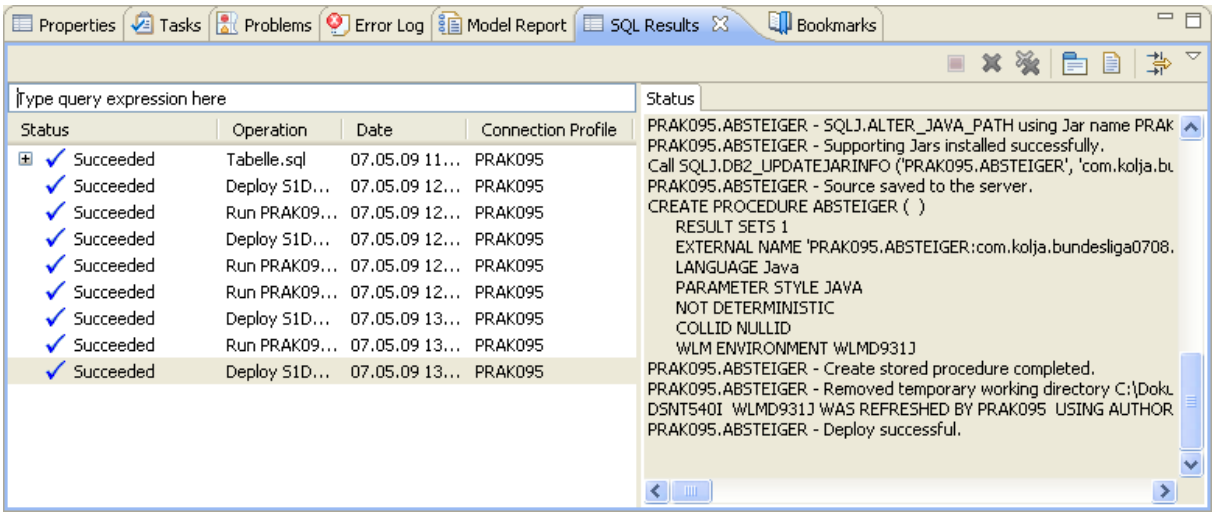
Kontrollieren Sie im sich öffnenden Dialog die Schemaangabe und klicken Sie auf „Next“.



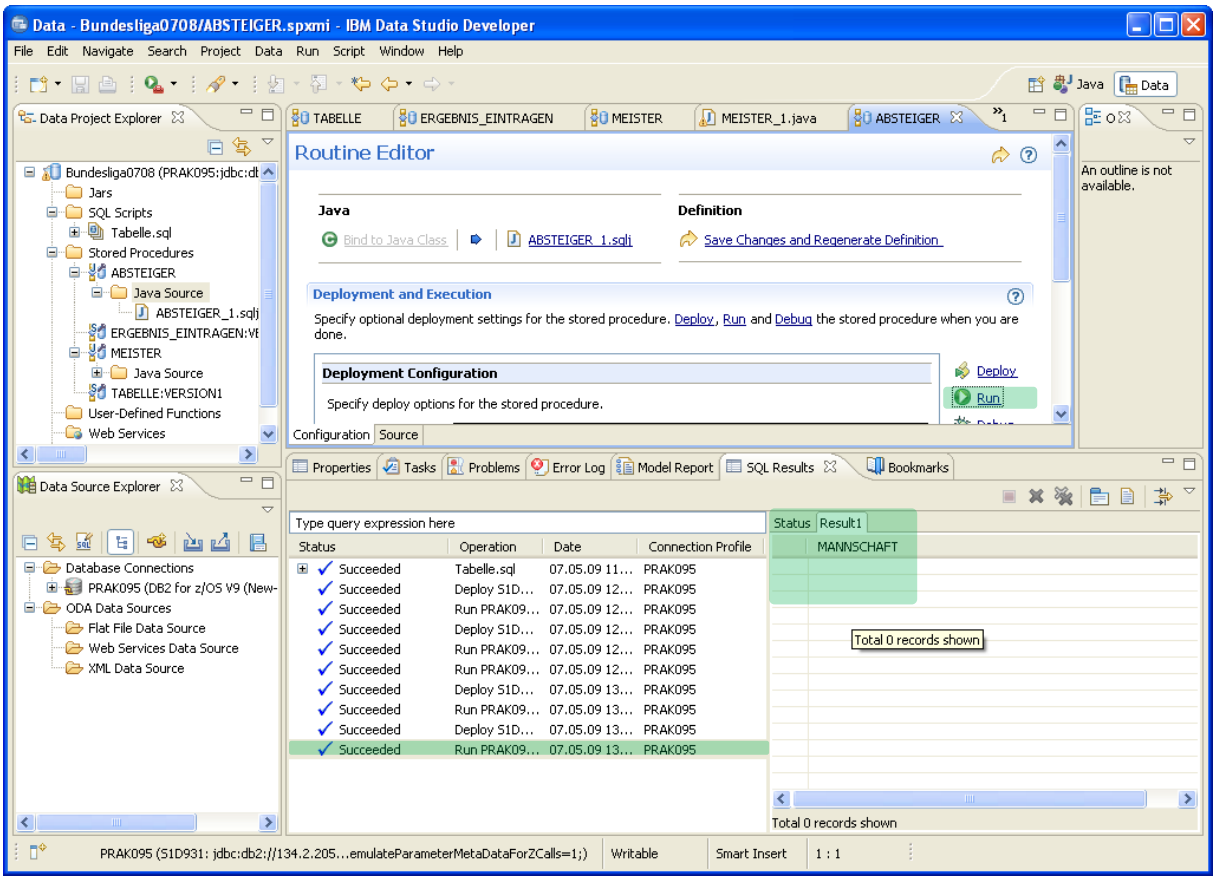
Überprüfen Sie im nächsten Fenster die Angaben zu Collection ID und WLM environment und klicken Sie dann auf „Finish“.



Nun läuft der zu Beginn beschriebene Vorgang ab: Durch Precompiler und Compiler wird ein ausführbarer Code und ein Profile erstellt. Außerdem erstellt der SQLJ Profile Customizer die nötigen Packages und bindet sie an das DB2-System. Ein erfolgreicher Ablauf wird im SQL Results-Fenster angezeigt.



Die Stored Procedure kann nun durch einen Klick auf „Run“ ausgeführt werden.



Da nach dem 31.Spieltag noch kein Absteiger feststand, ist das Ergebnis eine leere Tabelle. Um einen Absteiger festzulegen, reicht es, für Hansa Rostock zwei Niederlagen gegen Arminia Bielefeld und Energie Cottbus einzutragen (abweichend von den wirklichen Spielpaarungen der Saison 2007/2008).

Die beiden Stored Procedures können auch über die Beispielanwendung ausgeführt werden. Hierzu werden die jeweiligen Knöpfe im Register Meister/Absteiger angeklickt.

Aufgabe: Erstellen Sie Screenshots, die die korrekte Funktionalität der Stored Procedures MEISTER und ABSTEIGER zeigen.

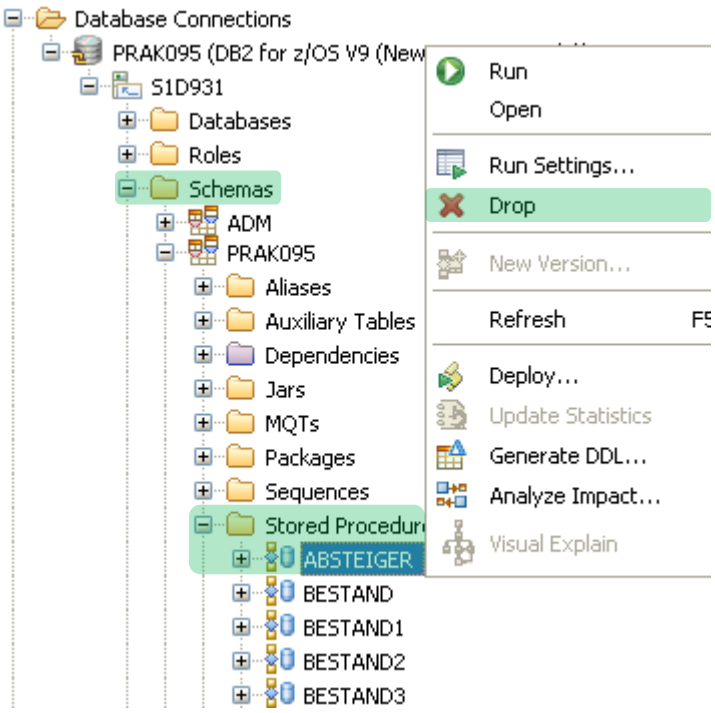
5.2.2 Löschen von Stored Procedures

Um eine Stored Procedure zu löschen, sind drei Schritte nötig:

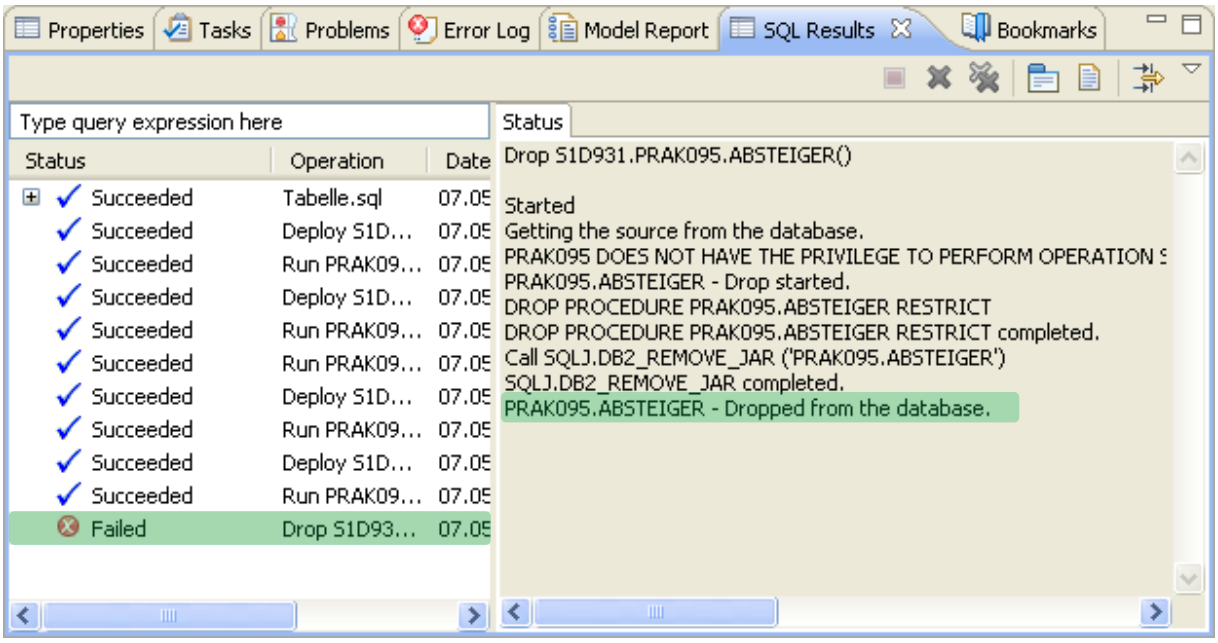
- Entfernen der Einträge in den Katalogtabellen
- Entfernen des Programmcodes
- Entfernen des Packages

Um Einträge zu der Stored Procedure aus den Katalogtabellen zu entfernt, genügt ein DROP-Befehl, d.h. das Ausführen des SQL-Befehls DROP PROCEDURE <stored-procedure-name>. Für das Entfernen des Java-Programmcodes stellt das DB2-System eine Stored Procedure zur Verfügung mit Namen SQLJ.DB2_REMOVE_JAR.

Das Data Studio bietet eine einfache Möglichkeit, diese beiden Schritte durchzuführen. Im Data Source Explorer der Eclipse-Oberfläche lassen sich sämtliche Stored Procedures auflisten und über das Kontextmenü löschen.



Die Nachfrage, ob wirklich gelöscht werden soll, wird mit „OK“ bestätigt.



Im SQL Results-Fenster sieht man, dass der IBM Data Studio Developer ebenfalls die oben erklärten Befehle ausführt. Es wird zwar ein Fehler angezeigt, der allerdings keinen Einfluss auf den Löschvorgang hat.

Für Stored Procedures, für die ein Package erstellt wurde, muss dieses noch entfernt werden. Da dies über den IBM Data Studio Developer etwas komplizierter ist, werden wir uns hierzu mit einem 3270-Terminalemulator mit dem entsprechenden Mainframe verbinden. Nachdem man sich im TSO angemeldet hat, gelangt man durch die Auswahl von „M“ im ISPF ins IBM Products Panel.

```
  Menu  Help
 ------------------------------------------------------------------------------
                            IBM Products Panel
                                                                        More:     +
This panel contains 20 options, you may need to F8 for more.
 1  SMP/E       System Modification Program/Extended
 2  ISMF        Integrated Storage Management Facility
 3  RACF        Resource Access Control Facility
 4  HCD         Hardware Configuration Dialogs
 5  SDSF        Spool Search and Display Facility
 6  IPCS        Interactive Problem Control System
 7  DITTO       DITTO/ESA for MVS Version 1
 8  RMF         Resource Measurement Facility
 9  DFSORT      Data Facility Sort
 10 OMVS        MVS OpenEdition
 11 DB2 V8      DB2 8.1.0 Must use volume S7DB81 & TSOPROC DBSPROC
 12 DB2ADM      Data Base Admin Tool
 13 QMF V8      QMF 8.1.0 Must use volume S7DB81 & TSOPROC DBSPROC
 14 PE          Performance Expert
 15 MQ          WMQ Series Operations and Control
 16 WLM         Workload Manager
 Option ===> 12
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions  F12=Cancel
```

Hier wählt man das DB2 Admin Tool durch die Option „12“ aus.

```
DB2 Admin ------------------- Active DB2 Systems ---------------- Row 1 from 2


 This is a list of the active DB2 systems on this MVS system.

 Enter:
 DB2 system name ===> D931          Retain DB2 system name ===> YES  (Yes/No)

 Or select the one you wish to use, or press END to exit.

 Sel DB2 System Description                                              Group
 --- ---------- ------------------------------------------------------- -----
     DB8G       DB2 V8
     D931
 ******************************** Bottom of data ********************************




 Command ===>                                                 Scroll ===> PAGE
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Durch Eingabe von „D931“ wird das DB2-System der Version 9 ausgewählt.

```
DB2 Admin -------------- DB2 Administration Menu 7.1.0 ------------------ 16:19
 Option ===> 1

    1 - DB2 system catalog                                  DB2 System: D931
    2 - Execute SQL statements                              DB2 SQL ID: PRAK095
    3 - DB2 performance queries                             Userid    : PRAK095
    4 - Change current SQL ID                               DB2 Rel   : 915
    5 - Utility generation using LISTDEFs and TEMPLATEs
    P - Change DB2 Admin parameters
   DD - Distributed DB2 systems
    E - Explain
    Z - DB2 system administration
   SM - Space management functions
    W - Manage work statement lists
    X - Exit DB2 Admin

.-------------------------------------------------------------------------.
| Database 2 Administration Tool.                                         |    +
| 5697-L90 (C) Copyright IBM Corporation 1995, 2005.                      |
| All rights reserved.  Licensed materials - property of IBM.             |
| US Government Users Restricted Rights - Use, duplication or disclosure  |
| restricted by GSA ADP schedule contract with IBM Corp.                  |
'-------------------------------------------------------------------------' GE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Wählen Sie „1“ für den DB2 Systemkatalog und dort „K“ um die vorhandenen Packages auflisten zu lassen.

```
DB2 Admin ---------------------- D931 System Catalog --------------------- 16:19
 Option ===> K

                                                                  More:     +
 Object options:                                         DB2 System: D931
   AO - Authorization options                            DB2 SQL ID: PRAK095
    G - Storage groups                P - Plans
    D - Databases                     L - Collections
    S - Table spaces                  K - Packages
    T - Tables, views, and aliases    M - DBRMs
    V - Views                         H - Schemas
    A - Aliases                       E - User defined data types
    Y - Synonyms                      F - Functions
    X - Indexes                       O - Stored procedures
    C - Columns                       J - Triggers
    N - Constraints                   Q - Sequences
   DS - Database structures         DSP - DS with plans and packages

 Enter standard selection criteria (Using a LIKE operator, criteria not saved):
 Name     ===>                    >  Grantor  ===>          >
 Owner    ===>          >            Grantee  ===>          >
 In D/L/H ===>                    >
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

```
DB2 Admin ------------------- D931 Packages ---------------- Row 1 to 9 of 383

 Commands:  BIND  REBIND  FREE  VERSIONS  GRANT
 Line commands:
  DP - Depend  A - Auth  T - Tables  V - Views  X - Indexes  S - Table spaces
  Y - Synonyms  RB - Rebind  F - Free  B - Bind  BC - Bind copy   GR - Grant
  EN -Enab/disab con  PL - Package lists  P - Local plans  LP - List PLAN_TABLE
  I - Interpretation  SQ - SQL in package  VE - Versions  D - Databases
                                                        V I V O Quali-   R E D
 S  Collection          Name     Owner    Bind Timestamp   D S A P fier     L X R
    *                   *        PRAK095  *                * * * * *        * * *
 -- ------------------ -------- -------- ---------------- - - - - -------- - - -
    DSNTWR91            DSNTWR   BUWD     2007-03-27-07.10 R S Y Y BUWD       N
    DSNTWR              DSNTWR   BUWD     2007-03-27-07.10 R S Y Y BUWD       N
    DSNREXX             DSNREXX  BUWD     2007-03-27-07.10 B S Y Y BUWD       N
    DSNREXUR            DSNREXX  BUWD     2007-03-27-07.10 B U Y Y BUWD       N
    DSNREXCS            DSNREXX  BUWD     2007-03-27-07.10 B S Y Y BUWD       N
    DSNREXRS            DSNREXX  BUWD     2007-03-27-07.10 B T Y Y BUWD       N
    DSNREXRR            DSNREXX  BUWD     2007-03-27-07.10 B R Y Y BUWD       N
    DSNTIAP             DSNTIAP  BUWD     2007-03-27-08.50 R   Y Y BUWD       N
    DSNESPCS            DSNESM68 BUWD     2007-03-27-08.50 R S Y Y BUWD       N
 Command ===>                                                 Scroll ===> PAGE
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Um die Liste filtern und alle Packages anzuzeigen, die Sie erstellt haben, tragen Sie unter Owner Ihre Benutzerkennung ein und drücken Sie die Entertaste.

```
DB2 Admin ------------------- D931 Packages ----------------- Row 223 from 383

 Commands:  BIND  REBIND  FREE  VERSIONS  GRANT
 Line commands:
  DP - Depend  A - Auth  T - Tables  V - Views  X - Indexes  S - Table spaces
  Y - Synonyms  RB - Rebind  F - Free  B - Bind  BC - Bind copy   GR - Grant
  EN -Enab/disab con  PL - Package lists  P - Local plans  LP - List PLAN_TABLE
  I - Interpretation  SQ - SQL in package  VE - Versions  D - Databases
                                                        V I V O Quali-   R E D
 S  Collection          Name     Owner    Bind Timestamp   D S A P fier     L X R
    *                   *        PRAK095* *                * * * * *        * * *
 -- ------------------ -------> -------- ---------------- - - - - -------- - - -
    PRAK095             PROGNOSE PRAK095  2009-03-24-18.31 R S N Y PRAK095  C N R
    PRAK095             TEAMS    PRAK095  2009-04-08-14.03 R S Y Y PRAK095  C N R
    PRAK095             PROCEDUR PRAK095  2009-01-26-18.36 R S N Y PRAK095  C N R
    NULLID              S6374761 PRAK095  2009-04-20-13.21 B U Y Y PRAK095  C N B
 F  NULLID              S1190951 PRAK095  2009-03-10-19.13 B U Y Y PRAK095  C N B
 F  NULLID              S1190952 PRAK095  2009-03-10-19.13 B S Y Y PRAK095  C N B
 F  NULLID              S1190953 PRAK095  2009-03-10-19.13 B T Y Y PRAK095  C N B
 F  NULLID              S1190954 PRAK095  2009-03-10-19.13 B R Y Y PRAK095  C N B
    PRAK095             S552391  PRAK095  2009-03-10-19.31 B U Y Y PRAK095  C N B
 Command ===>                                                 Scroll ===> PAGE
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Es befinden sich die 4 Packages in der Liste, die vorher durch die Java Stored Procedure unter Verwendung von SQLJ angelegt wurden und die in der Collection NULLID gespeichert sind. Um diese zu löschen wird in die entsprechenden Zeilen in die erste Spalte ein „F“ (Free) eingetragen. Durch

Drücken der Eingabetaste werden nun die einzelnen Packages mit einem FREE PACKAGE-Befehl gelöscht.

```
DB2 Admin ---------------------- D931 Free Package ---------------------- 16:26
 Command ===>


  FREE PACKAGE (

  Location   ===>                  >  (Blank for local)

  Collection ===> NULLID             >

  Name       ===> S1190951 >

  (
  Version    ===>

  ))









  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Durch erneutes Drücken der Entertaste bekommt man die Statusmeldung über den Löschvorgang.

```
DB2 Admin ------------------- D931 Command Output Display ---------------------

 FREE PACKAGE(NULLID.S1190951)


******************************** Top of Data ********************************
 DSNT232I  -D931 SUCCESSFUL FREE FOR
            PACKAGE = S1D931.NULLID.S1190951.()
******************************** Bottom of Data *****************************









 Command ===>                                                 Scroll ===> PAGE
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Durch Betätigen der Taste F4 wird man nun aufgefordert, die Informationen für den nächsten FREE-PACKAGE-Befehl zu bestätigen. Nachdem alle vier Packages gelöscht wurden, gelangt man zurück in

die Liste der Packages. Es sollten sich nun keine von Ihnen erstellten Packages mehr in der Collection NULLID befinden.

Aufgabe: Um die Collection NULLID nicht mit Packages zu belasten, sollen ihre sämtlichen Packages in dieser Collection gelöscht werden. Erstellen Sie einen Screenshot der Packageliste, die nur ihre Packages anzeigt.

6 External SQL Stored Procedures

6.1 Allgemeines

Neben Native SQL Stored Procedures und External high-level language Stored Procedures existiert noch ein dritter Typ von Stored Procedures: External SQL Stored Procedures.

Hierbei handelt es sich um Stored Procedures, die wie Native SQL Stored Procedures in der SQL Procedural Language (SQL PL) geschrieben sind. Der SQL PL-Code wird hier jedoch in ein C-Programm übersetzt, dass dann wie eine External high-level language Stored Procedure an das Datenbanksystem gebunden wird. Dies bedeutet, dass nachdem ein Precompiler aus den SQL-Befehlen Database Request Module (DBRM) erstellt hat und das C-Programm entsprechend modifiziert hat, die DBRMs in ein Package gebunden werden. Anschließend wird die Stored Procedure dem DB2-System als External high-level language Stored Procedure bekannt gemacht. Im Unterschied zu diesen kann hier der SQL-Programmcode bei der DDL dabei stehen. Die Ausführung der Stored Procedure findet in einem separaten Adressraum statt. Voraussetzung für die Entwicklung von External SQL Stored Procedures ist deshalb das Vorhandensein eines Workloadmanagers, der es erlaubt, einen Adressraum mit der nötigen Umgebung bereitzustellen. Die komplette Umsetzung erledigt auf dem Tübinger Großrechner eine Stored Procedure mit Namen DSNTPSMP. Diese muss in einem separaten dafür eingerichteten Adressraum ablaufen. Die Procedure des hierfür notwendigen Workloadmanagers befindet sich im Member D931PSMP des Datasets ADCD.Z18.PROCLIB.

Um External SQL Stored Procedures zu erstellen, benötigt man als Benutzer eines Praktikumaccounts in Tübingen zusätzlich folgende Rechte:

UPDATE auf DSN931.DSN.V910.DBRMLIB.DATA
UPDATE auf DSN931.DSN.V910.RUNLIB.LOAD
UPDATE auf USER.PSMLIB.DATA

Da Praktikumaccounts diese Rechte aus administrativen Gründen nicht besitzen, wird im Folgenden nur grob auf die Erstellung von External SQL Stored Procedures auf dem Tübinger Mainframe eingegangen.

6.2 Erstellung von External SQL Stored Procedures (Anleitung)

Im Folgenden soll eine External SQL Stored Procedure erstellt werden, die den Inhalt der Tabelle SYSIBM.SYSROUTINES ausgibt.

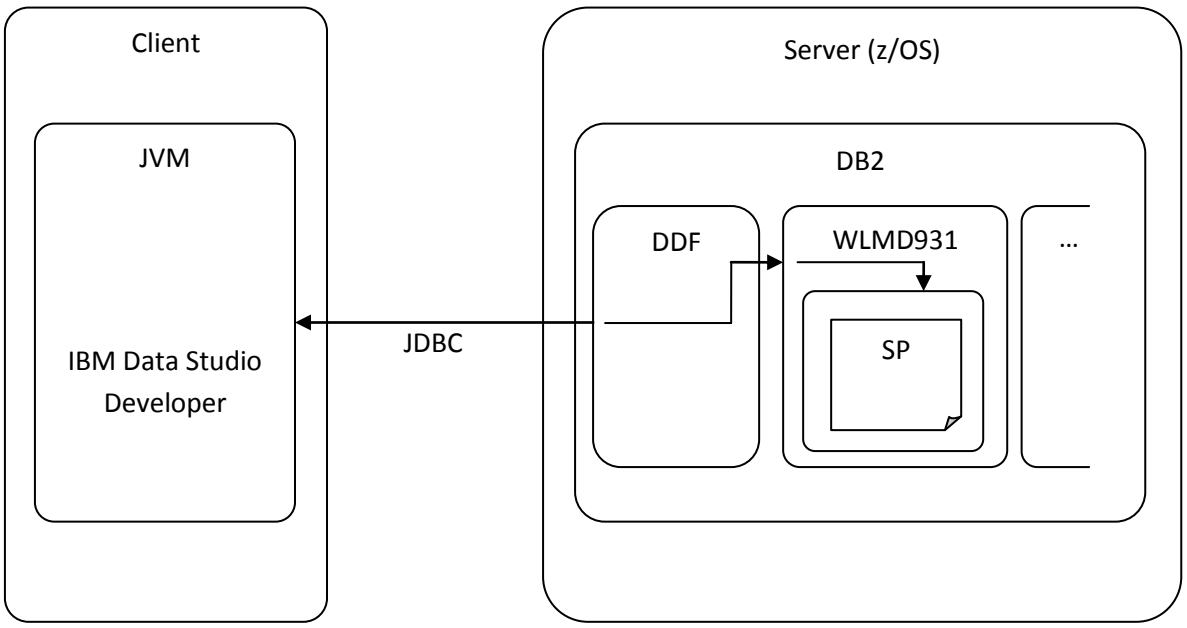
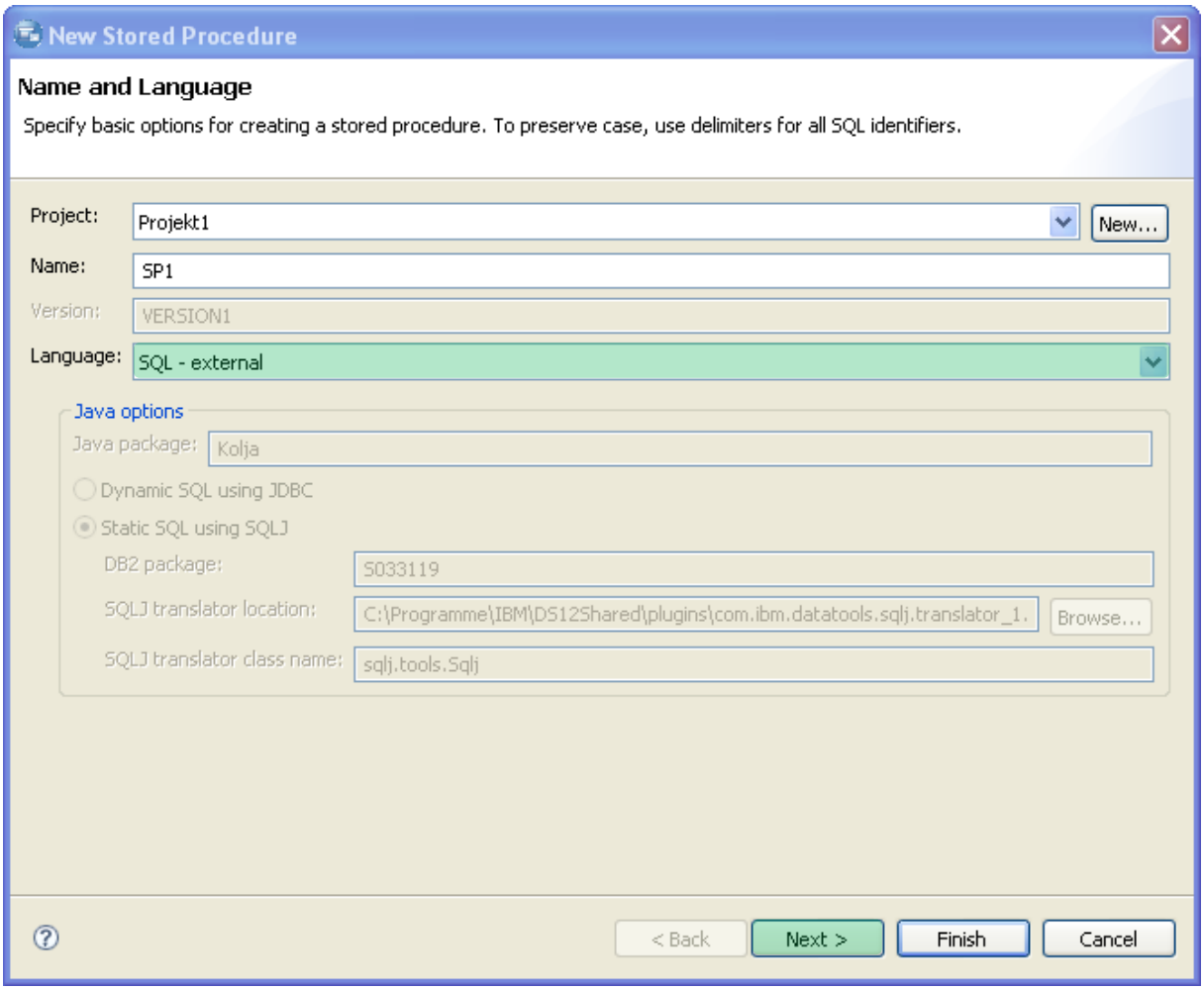
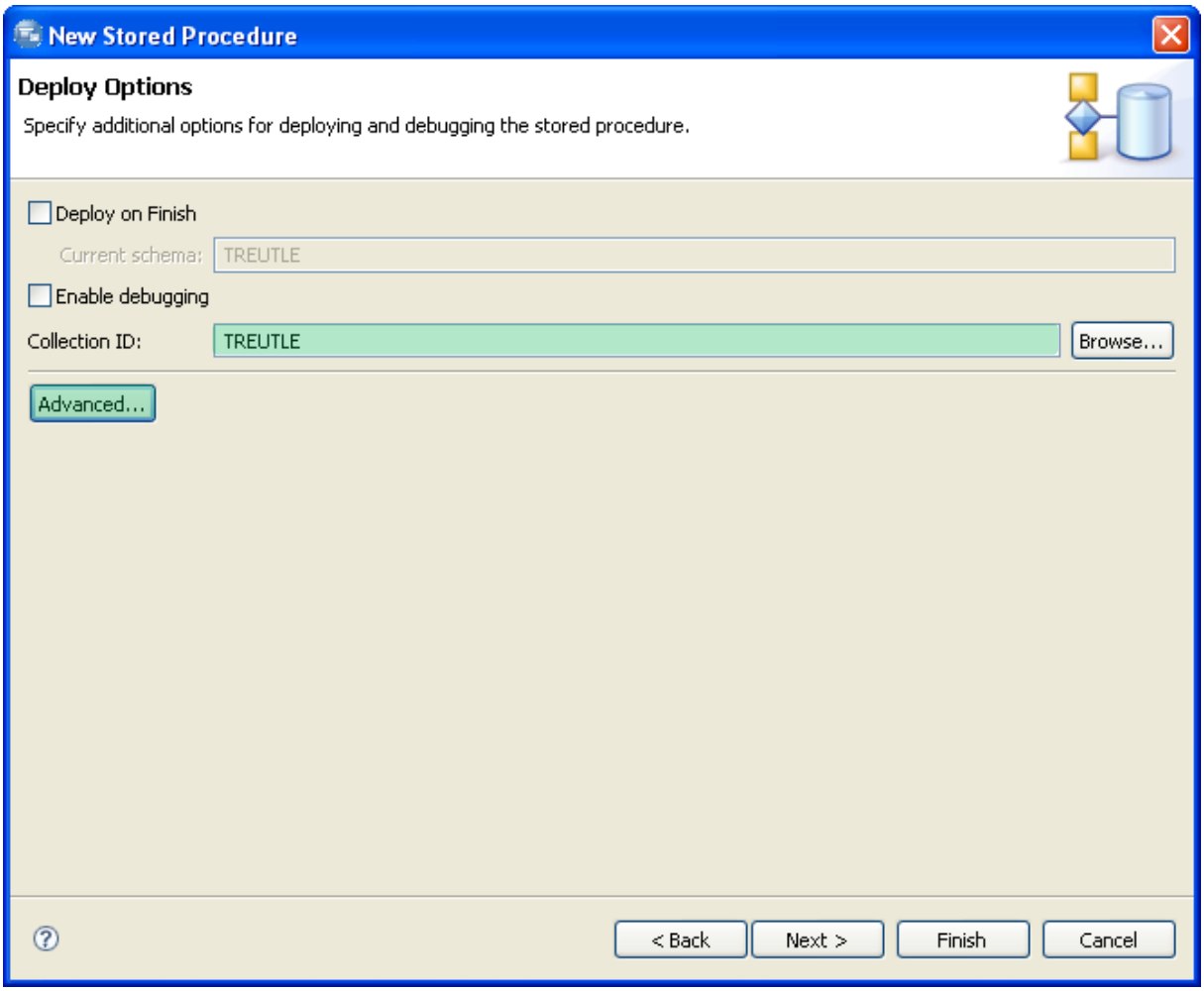


Abbildung 8: Übersicht über den Aufruf einer External SQL Stored Procedure über den IBM Data Studio Developer

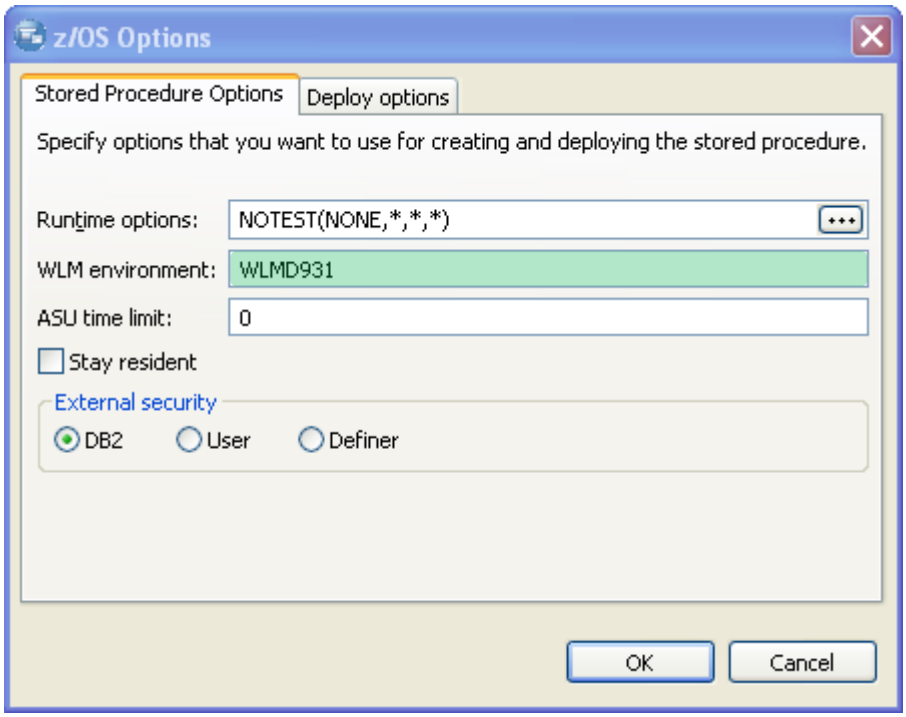
Um eine External SQL Stored Procedure mit dem IBM Data Studio Developer zu erstellen, wählt man im IBM Data Studio Developer im Dialog zum Erstellen einer neuen Stored Procedure als Sprache „SQL-External“ aus.



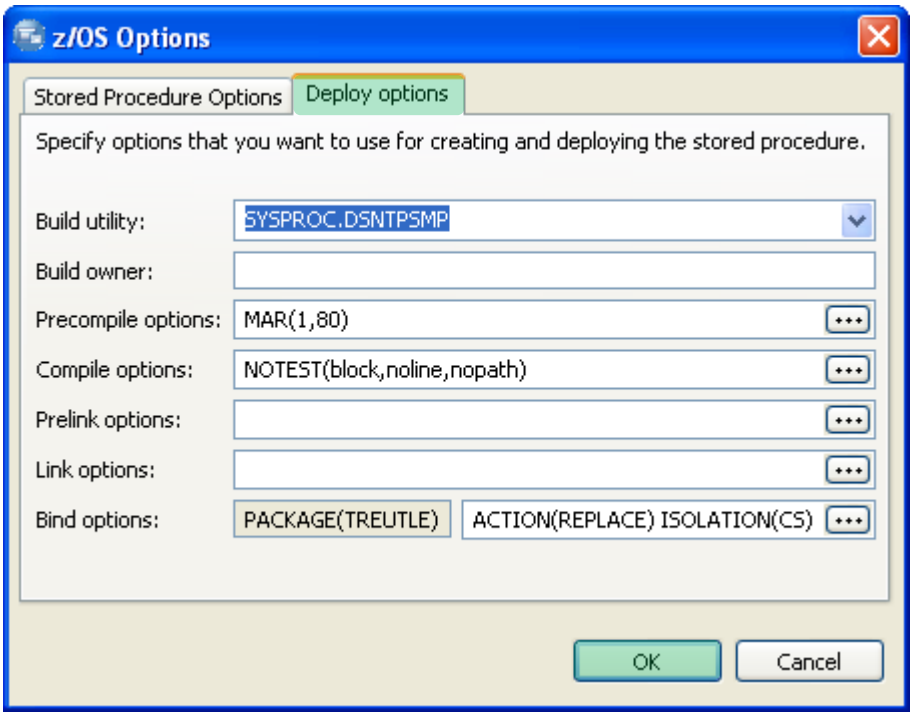
Im Feld Collection-ID wird Ihr Schema (= Benutzerkennung) ausgewählt (hier TREUTLE).



Da die Stored Procedure in einem externen Adressraum ausgeführt werden muss, muss ein Workloadmanager angegeben werden, der diesen Adressraum zur Verfügung stellt. Unter „Advanced“ muss als Workloadmanager deshalb WLMD931 eingetragen werden.



Im Register „Deploy options“ erkennt man, dass beim Einrichten auf dem entfernten DB2-System standardmäßig die Stored Procedure SYSPROC.DSNTPSMP aufgerufen wird. Diese Stored Procedure wurde auch in Tübingen eingerichtet, übersetzt den Programmcode der Stored Procedure in ein C-Programm und bindet ihn an das Datenbanksystem.

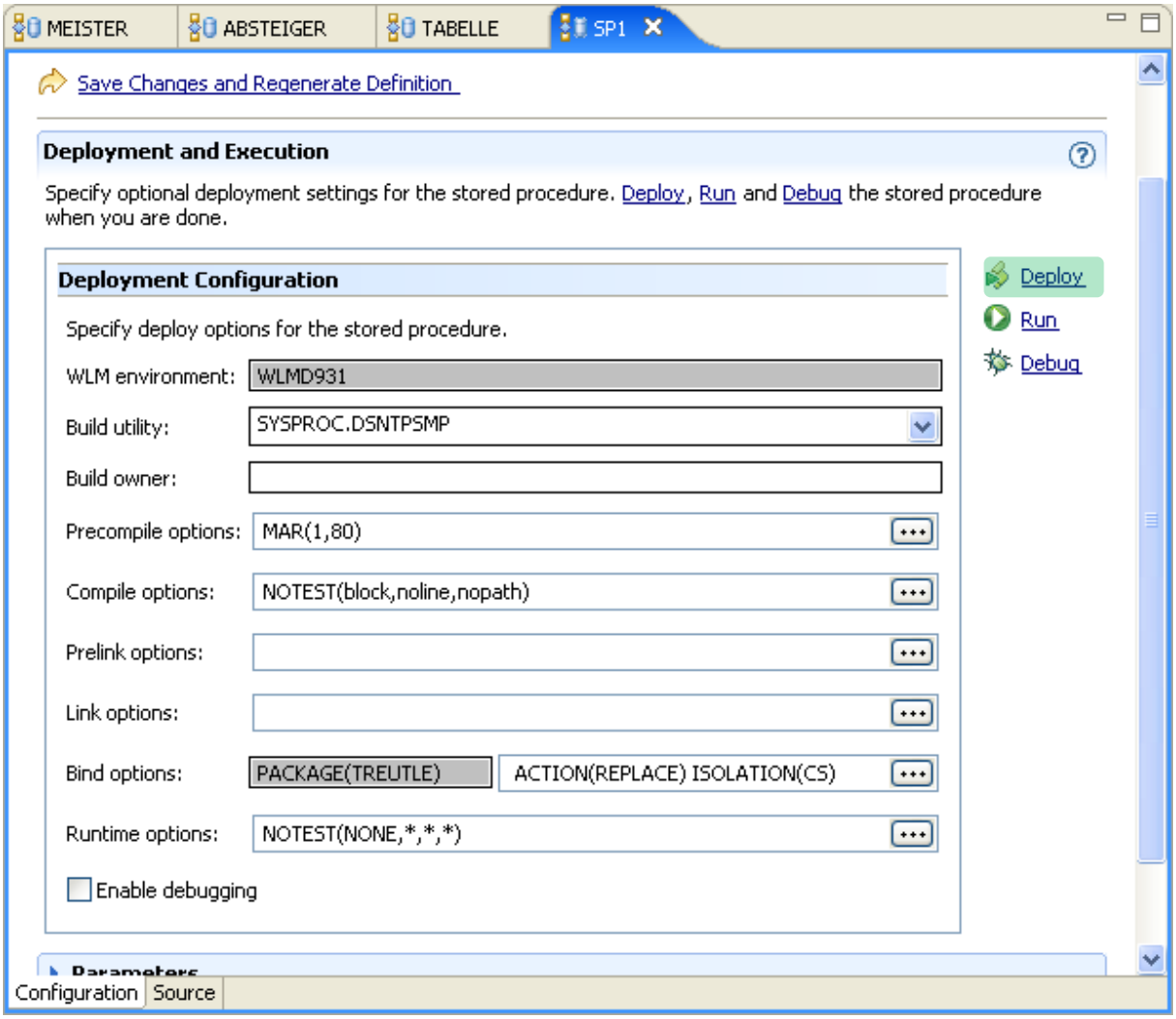


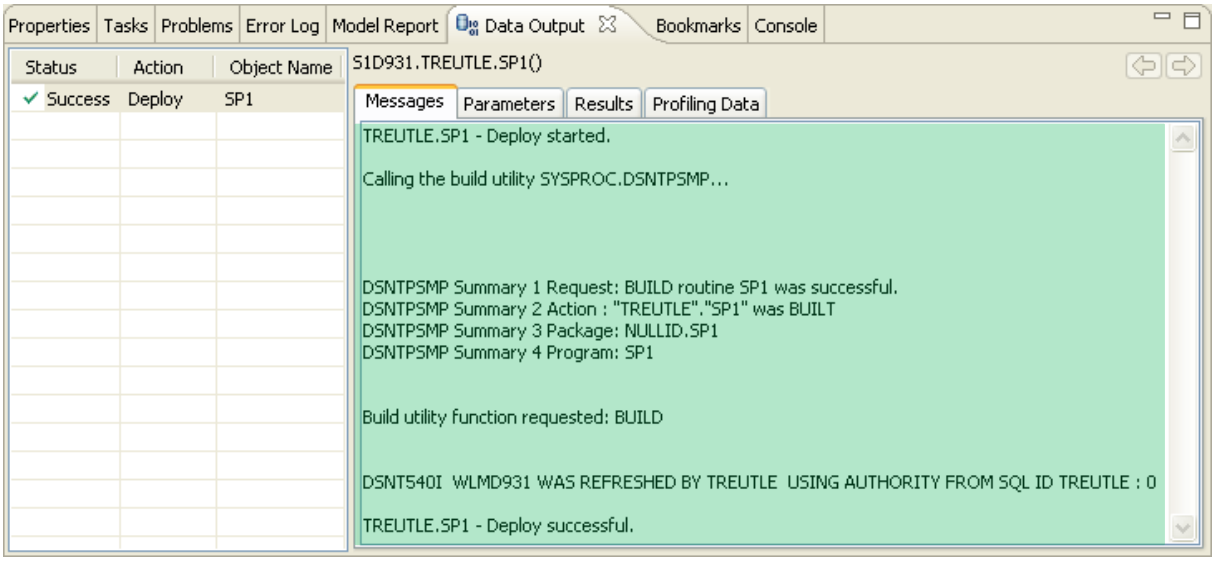
Nach dem Abschluss des Erstellassistenten wird folgende Stored Procedure erstellt:

```
CREATE PROCEDURE SP1 (  )
      RESULT SETS 1
      LANGUAGE SQL
      FENCED
      COLLID TREUTLE
      WLM ENVIRONMENT WLMD931
      RUN OPTIONS 'NOTEST(NONE,*,*,*)'
-----------------------------------------------------------------------
-- SQL Stored Procedure
-----------------------------------------------------------------------
P1: BEGIN
      -- Declare cursor
      DECLARE cursor1 CURSOR WITH RETURN FOR
            SELECT SCHEMA, NAME FROM SYSIBM.SYSROUTINES;

      -- Cursor left open for client application
      OPEN cursor1;
END P1
```

Von einer Native SQL Stored Procedure unterscheidet sie sich nur durch unterschiedliche Angaben in den Optionen des CREATE PROCEDURE-Befehls. Die Angabe FENCED signalisiert, dass die Ausführung der Stored Procedure in einem externen Adressraum stattfinden soll. Über die Angabe WLM ENVIRONMENT wird angegeben, dass der Workloadmanager WLMD931 einen Adressraum hierfür zur Verfügung stellen wird. Durch die Option RUN OPTIONS können der Stored Procedure noch sprachspezifische Laufzeitoptionen mitgegeben werden. Für Details hierzu sei auf [IBM5] verwiesen.

Durch einen Klick auf „Deploy“ im Configuration-Fenster der Stored Procedure wird diese im entfernten DB2-System eingerichtet.





Im SQL Results-Fenster erkennt man eine erfolgreiche Einrichtung der Stored Procedure. Nun lässt sie sich durch einen Klick auf „Run“ ausführen.

7 Zusammenfassung und Ausblick

In der vorliegenden Arbeit sollte beschrieben werden, wie Stored Procedures auf einem Mainframe eingerichtet werden können und welche Voraussetzungen dafür nötig sind. Die konkrete Umsetzung wurde im Rahmen eines Tutorials dargestellt, das für das Praktikum der Informatikstudierenden eingesetzt werden kann.

Die verschiedenen Typen von Stored Procedures haben jeweils verschiedene Vor- und Nachteile und müssen daher nach Bedarf gewählt werden. Folgende Tabelle soll einen zusammenfassenden Überblick über die verschiedenen Typen von Stored Procedures sowie deren spezifische Merkmale geben:

	Native SQL Stored Procedures	**External SQL Stored Procedures**	**External high-level language Stored Procedures**
Programmiersprache der Logik der Stored Procedure	SQL	SQL	Höhere Programmiersprache
Trennung DDL und Logik der Stored Procedure	Nein	Nein	Ja
Adressraum für die Ausführung	Adressraum von DB2 (DBM1)	Externer Adressraum	Externer Adressraum
Komplexität der Logik der Stored Procedure	Beschränkt durch SQL, Zugriff nur auf DB2-Ressourcen	Beschränkt durch SQL, Zugriff nur auf DB2-Ressourcen	Je nach Programmiersprache, i.d.R. groß durch Programmbibliotheken, Zugriff auch auf externe Ressourcen
Performance *	+++	++	COBOL und C/C++ Stored Procedures: ++++ Java und REXX Stored Procedures: +

* Mehr Pluszeichen bedeuten i.d.R. bessere Performance

Die drei Stored Procedure-Typen schneiden in einem direkten Vergleich ihrer Laufzeiten alle recht ähnlich ab. Dennoch lassen sich nach [IBM1] leichte Tendenzen erkennen: COBOL und C/C++ Stored Procedures haben durchschnittlich leichte Performancevorteile. Im Gegensatz dazu skalieren Stored Procedures, deren Code zur Laufzeit erst interpretiert werden muss (z.B. Java oder REXX), am schlechtesten. External SQL Stored Procedures sind im Vergleich zu Native SQL Stored Procedures

langsamer, da sie intern zu einem C-Programm umgewandelt werden müssen und dann in einem externen Adressraum ausgeführt werden.

Die Wahl eines Stored Procedure-Typs sollte allerdings nicht nur von der Performance abhängig gemacht werden. Vielmehr sollte auch auf die verfügbare Sprachumgebung sowie auf Programmiersprachenkenntnisse der Entwickler geachtet werden. Des Weiteren spielt die Komplexität des benötigten Programms eine entscheidende Rolle. Während man mit Native SQL und External SQL Stored Procedures nur Zugriff auf DB2-Ressourcen hat, kann unter der Verwendung von External high-level Stored Procedures auch auf externe Ressourcen zugegriffen werden.

Zusammenfassend lässt sich festhalten, dass Stored Procedures eine praktische Möglichkeit darstellen, Geschäftslogik auf einen Server zu verlagern und dadurch Performance zu gewinnen. Anzumerken ist allerdings, dass durch Stored Procedures die Programmierzuständigkeit von Entwickler und Datenbankadministrator nicht mehr scharf getrennt sind. Vielmehr ist eine interdisziplinäre Zusammenarbeit gefragt. Außerdem wird für die Erstellung von Anwendungen mehr Planungsaufwand benötigt. Der Anwendungsprogrammierer muss nicht mehr die zugrunde liegende Datenbankstruktur, dafür aber die Schnittstellen zum Datenbanksystem kennen und in der Lage sein, diese entsprechend zu nutzen.

Mit dem IBM Data Studio Developer ist ein praktisches Werkzeug verfügbar, das das Erstellen von Native SQL und External SQL sowie External high-level language Stored Procedures sehr gut unterstützt. Auch der Einsatz der weit verbreiteten Eclipse-Plattform erleichtert Anfängern den Einstieg, da häufig Eclipse schon von der Programmierung mit Java eine bekannte Entwicklungsumgebung darstellt.

Auf dieser Arbeit aufbauend sind im Rahmen einer Studien- oder Diplomarbeit weitere Tutorials für das Client/Server-Praktikum denkbar. Ein Beispiel hierfür wäre eine Abhandlung über External high-level language Stored Procedures unter Verwendung der Programmiersprache COBOL. COBOL-Programme sind seit der Einführung der Mainframes im Einsatz. Teilweise sind heute noch Programme im Einsatz, die vor über 30 Jahren erstellt wurden. Auch 50 Jahre nach der Entwicklung dieser Programmiersprache werden mehr als 83% aller Transaktionen durch COBOL-Programme auf Mainframes durchgeführt. Desweiteren bietet das Data Studio eine einfache Möglichkeit, Stored Procedures als Web Services zur Verfügung zu stellen. Eine detaillierte Anleitung hierzu findet sich in [IBM8]. Ein Web Service ist eine im Netz bereitgestellte Komponente, die eine Abstraktionsebene einer Anwendungslogik darstellt. Auf diesen Dienst kann über Internetstandardprotokolle zugegriffen werden. Durch den Einsatz von XML soll eine einfache Bereitstellung und hohe Verfügbarkeit von Web Services gewährleisten werden. Das Ziel ist die Interoperabilität von Softwaresystemen, um unabhängig von Plattform und Programmiersprachen miteinander kommunizieren und arbeiten zu können.

Abkürzungsverzeichnis

ACID	Atomicity, consistency, isolation, durability
BLOB	Binary Large Object
CICS	Customer Information Control System
CLOB	Character Large Object
COBOL	Common Business Oriented Language
CPU	Central Processing Unit
DBMS	Database Management System
DBRM	Database Request Module
DDL	Data Definition Language
GUI	Graphical User Interface
ISPF	Interactive System Productivity Facility
JCL	Job Control Language
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
LOB	Large Object
LPAR	Logical Partition
MVS	Multiple Virtual Storage
OS/390	Operating System 390
RACF	Resource Access Control Facility
REXX	Restructured Extended Executor
SEQUEL	Structured English Query Language
SQL	Structured Query Language
SQLJ	Structured Query Language for Java
SQL PL	Structure Query Language Procedural Language
SQL/PSM	Structured Query Language /Persistent Stored Modules
TSO	Time Sharing Option
WLM	Workload Manager

Literaturverzeichnis

[BEY] Beyerle M(2007). *Erstellung und Portierung mehrerer Tutorials mit dem Thema WebSphere Developer for zSeries für den studentischen Praktikumsbetrieb.* Diplomarbeit, Wilhelm-Schickard Institut für Informatik, Universität Tübingen. http://www-ti.informatik.uni-tuebingen.de/~spruth/DiplArb/MattBey.pdf (Stand April 2009).

[HER] Herrmann P, Kebschull U, Spruth WG(2004). *Einführung in z/OS und OS/390*. Oldenbourg, München. Zweite Auflage.

[IBM1] Bruni P, Kaschta S, Kutsch M, McGeoch G, Scanlon M, Vandensande J(2008). *DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond.* IBM Redbook SG247604. http://www.redbooks.ibm.com/abstracts/sg247604.html (Stand April 2009).

[IBM2] IBM. *IBM Information Management Software for z/OS Solutions Information Center*. http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc/db2prodhome.htm (Stand April 2009).

[IBM3] IBM. *IBM Data Studio Developer*. http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=de_DE&synkey=R491071Z63067U48 (Stand April 2009).

[IBM4] IBM. *IBM Data Studio*. http://www-01.ibm.com/software/data/studio/ (Stand November 2008).

[IBM5] IBM. *Language Environment Programming Reference*. http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zos.r9.ceea300/ceea3180.htm (Stand April 2009)

[IBM6] IBM. *Release Notes - IBM Data Studio Developer Version 1.1.2*. http://download.boulder.ibm.com/ibmdl/pub/software/data/studio/developer/11/112/docs/readme/readme.html (Stand April 2009).

[IBM7] IBM. *IBM System Z.* http://www-03.ibm.com/systems/de/z/ (Stand April 2009)

[IBM8] Klitsch J. *Data Web Services on DB2 for z/OS, Part 1: Unlock business functions using DB2 for z/OS stored procedures and Data Web Services.* IBM developer works. http://www.ibm.com/developerworks/data/library/techarticle/dm-0905db2zosstoredprocedures/index.html?S_TACT=105AGX11&S_CMP=FP (Stand Mai 2009)

[KIC] Olympia-Verlag GmbH. *kicker online*. http://www.kicker.de/news/fussball/bundesliga/spieltag/tabelle/liga/1/tabelle/1/saison/2007-08/spieltag/31 (Stand April 2009)

[LUD] Ludwig A(2007). *Praktikum Objektrelationale Datenbanksysteme*. Skriptum, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen.

[ORH] Orhandovic J, Grodtke I, Tiefenbacher M (2007). *DB2 Administration*. Addison-Wesley, München. Erste Auflage.

[SCHE] Scheibe MK. *JCL. Job Control Language im z/OS-Umfeld praktisch anwenden*. http://jedi.informatik.uni-leipzig.de/OLD/pubs/Lehre/JCLBuch.pdf (Stand April 2009).

[SUN] Sun Microsystems, Inc. *Java™ 2 Platform Standard Edition 5.0 API Specification*. http://www.cs.ubc.ca/local/computing/software/jdk-1.5.0/docs/api/ (Stand April 2009).

[SPO] Spoden M. *SQL/PL-Guide.* http://www.sqlpl-guide.com/ (Stand April 2009).

[SPRU] Spruth WG(2009). *Client/Server Systeme – CICS Transaktionsmonitor*. Skriptum, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen.

[TEU] Teuffel M(2001). *TSO. Time Sharing Option im Betriebssystem OS/390 MVS*. Oldenbourg, München. Sechste Auflage.

[ULL] Ullenboom C (2009). *Java ist auch eine Insel*. Galileo Press, Bonn. Achte Auflage. http://openbook.galileocomputing.de/javainsel8/ (Stand April 2009).

[W3C] W3C Working Group. *Web Services Architecture.* http://www.w3.org/TR/ws-arch/ (Stand Mai 2009)

Anhang

A. Syntax des CREATE PROCEDURE-Befehls

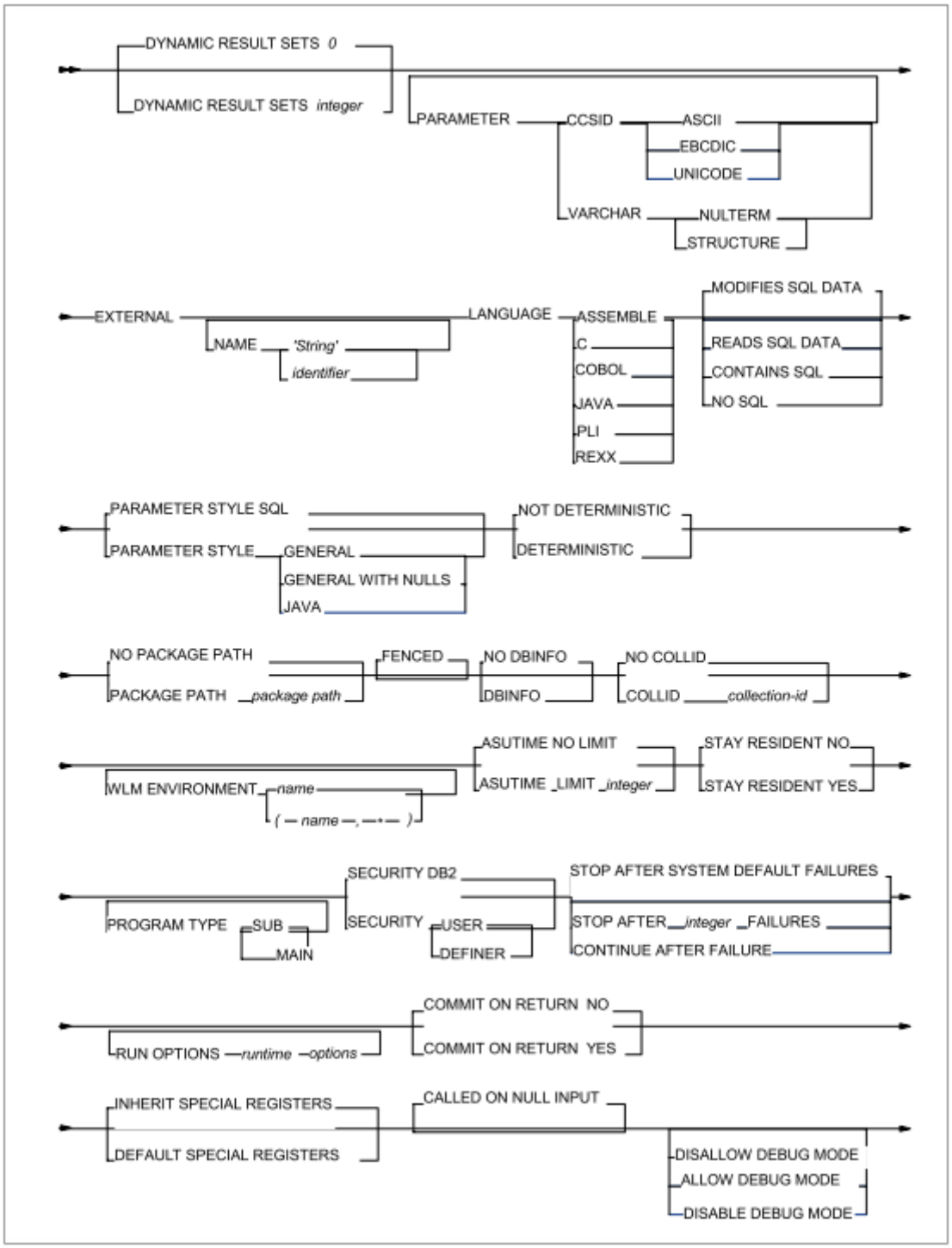


Abbildung 9: Option-list für External high-level language Stored Procedures (Quelle: [IBM1])

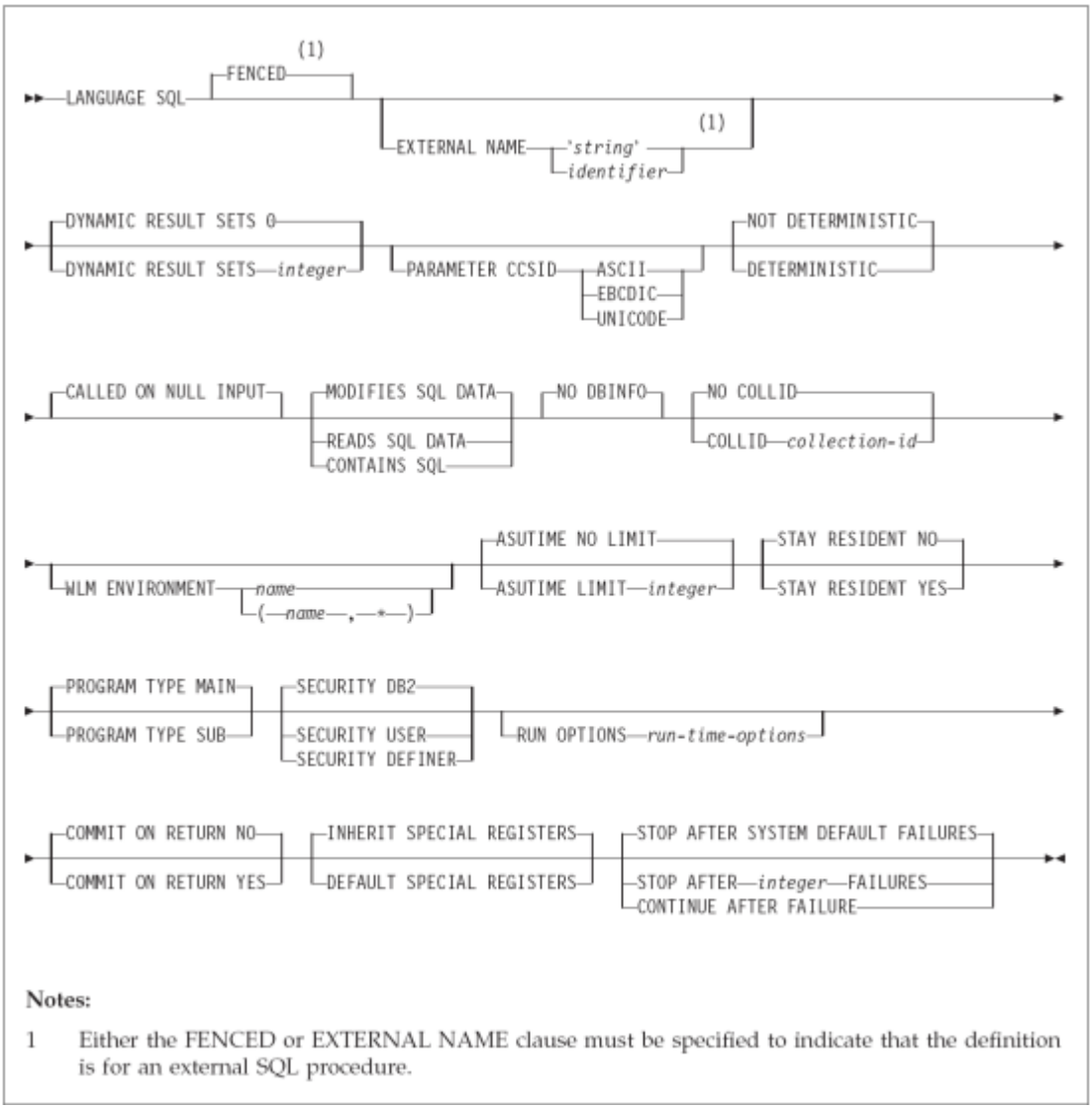


Abbildung 10: Option-list für External SQL Stored Procedures (Quelle: [IBM1])

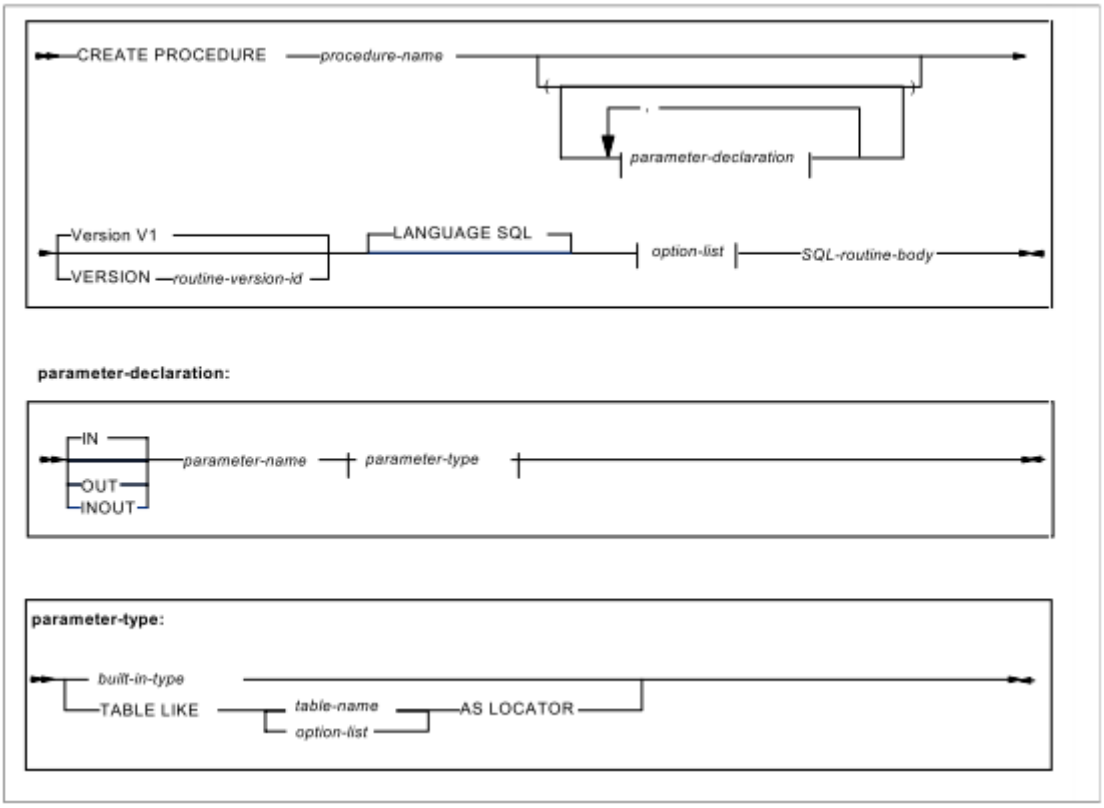


Abbildung 11: CREATE PROCEDURE-Befehl für Native SQL Stored Procedures (Quelle: [IBM1])

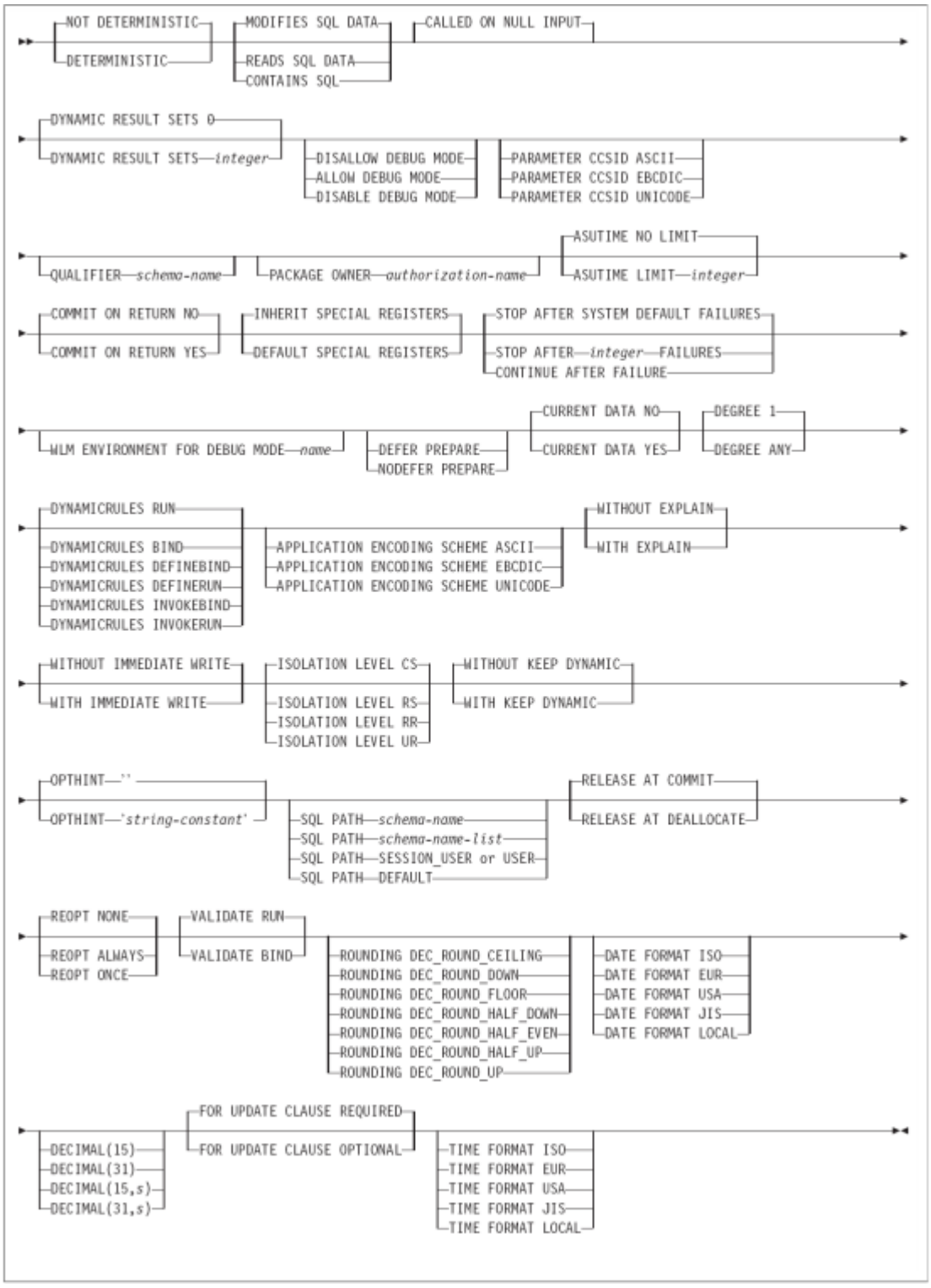


Abbildung 12: Option-list für Native SQL Stored Procedures (Quelle: [IBM1])

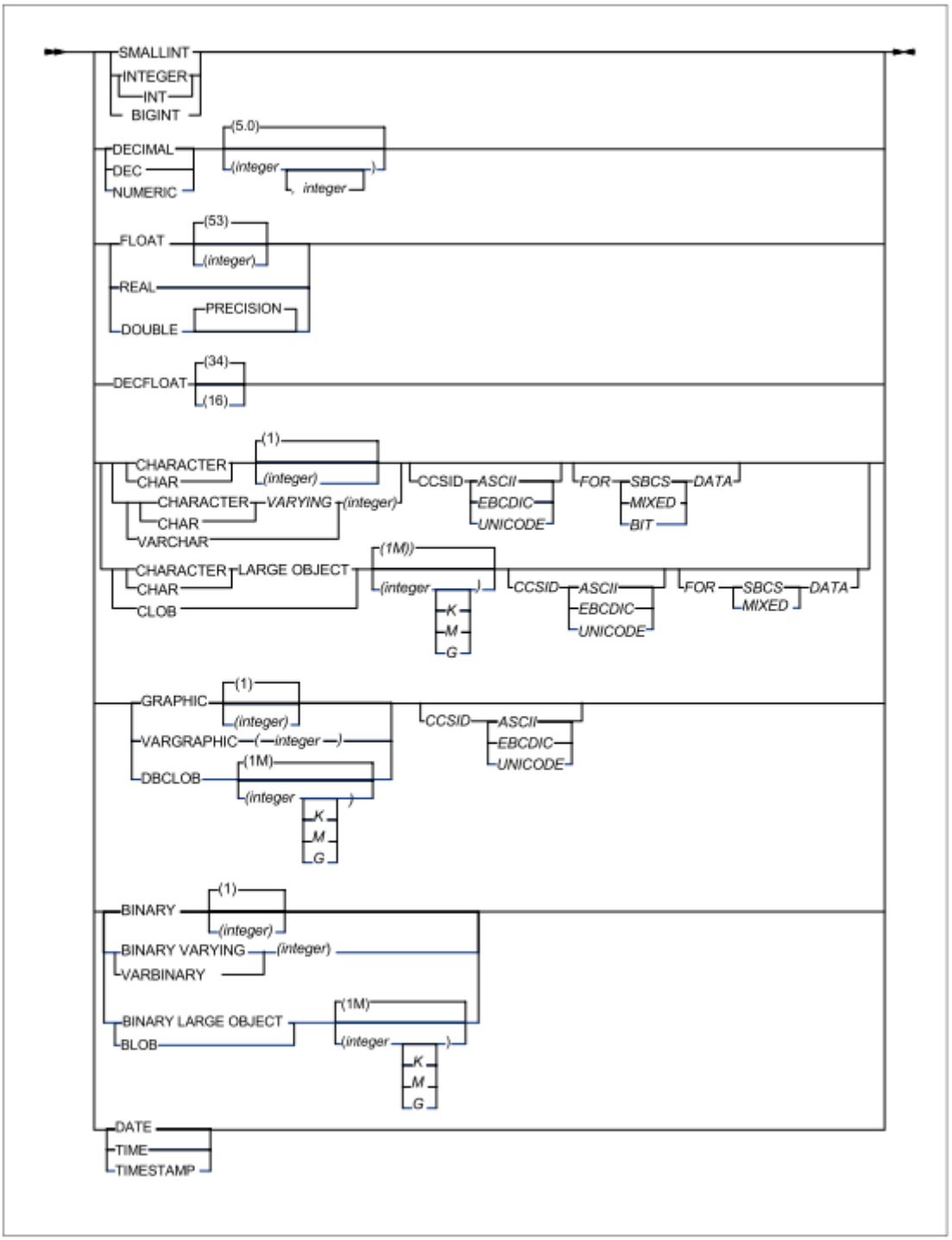


Abbildung 13: built-in-type (Quelle: [IBM1])

Erklärung der wichtigsten Optionen:

Name (procedure-name)

Legt den Namen der Stored Procedure fest.

Parameter (parameter-declaration)

Durch IN wird ein Eingabeparameter definiert, durch OUT ein Ausgabeparameter und durch INOUT ein Parameter, der sowohl Ein- als auch Ausgabeparameter ist. Zusätzlich müssen Parameter mit einem Namen und einen Datentyp angegeben werden. Sie können eine Tabelle darstellen oder ein SQL-Datentyp (vgl. Abbildung 13).

DYNAMIC RESULT SETS:

Eine Stored Procedure kann neben den Ausgabeparametern auch ganze Tabellen zurückgeben. Die maximale Anzahl der Tabellen wird mit dem DYNAMIC RESULT SETS-Parameter festgehalten. Defaultwert bei keiner Angabe ist 0.

LANGUAGE:

Der LANGUAGE-Parameter gibt an, in welcher Sprache der Stored Procedure-Quellcode vorliegt. Defaultwert bei keiner Angabe ist SQL. Je nach Sprache bestehen Abhängigkeiten zu den Parametern PARAMETER STYLE und PROGRAM TYPE. Wenn z.B. LANGUAGE JAVA verwendet wird, muss PARAMETER STYLE JAVA verwendet werden.

Typ des verwendeten SQL-Codes:

NO SQL: Die Stored Procedure kann keine SQL-Befehle ausführen.

MODIFIES SQL DATA: Die Stored Procedure kann INSERT, UPDATE und DELETE-Befehle enthalten. Dies ist der Defaultwert.

READS SQL DATA: Die Stored Procedure kann nur lesend auf Tabellen zugreifen.

CONTAINS SQL: Die Stored Procedure beinhaltet SQL-Befehle, kann aber auf keine Tabellen zugreifen.

PARAMETER STYLE

Der Parameter style spezifiziert eine Format, wie und welche Parameter mit der Stored Procedure beim Aufruf übergeben werden.

SQL: Wird dieser Parameter gewählt, so werden der Stored Procedure zusätzliche Parameter wie Indikatorvariablen (um zu signalisieren, dass ein Parameterwert NULL ist und der Parameter deshalb aus Effizienzgründen nicht übergeben wird), SQLState oder Diagnostic-String mit übergeben.

GENERAL: Nur die Parameter aus dem CALL-Befehl der Stored Procedure werden übergeben. NULL-Werte sind nicht für IN oder INOUT-Parameter erlaubt.

GENERAL WITH NULLS: Wie GENERAL nur dass auch NULL-Werte erlaubt sind. Dafür wird der Stored Procedure ein Array aus Indikatorvariablen übergeben.

JAVA: Gibt an, dass die Stored Procedure eine Parameterübergabekonvention verwendet, wie Java und SQLJ sie spezifiziert. Kann deshalb nur für Java verwendet werden und ist für Java die einzige Möglichkeit.

(NOT) DETERMINISTIC

Gibt an, dass die Stored Procedure immer das gleiche Ergebnis liefert, wenn sie mit gleichen Eingabeparametern aufgerufen wird. Wenn die Stored Procedure z.B. eine Zufallszahl zurückgibt, muss sie als NOT DETERMINISTIC deklariert werden. Das DB2-System kann mit Hilfe dieser Angabe die Stored Procedure optimieren.

COLLECTION ID

Bei Verwendung von SQL-Befehlen in Stored Procedures erzeugt der Precompiler ein Package, das den Zugriffsplan für den SQL-Code enthält. Dieses Package wird in einer Sammlung von Packages (Collection) mit einer Collection ID gespeichert. Bei der Ausführung der Stored Procedure wird dann die Collection ID der Stored Procedure ermittelt und der Zugriffsplan aus dem entsprechenden Package geladen.

PACKAGE PATH

Oftmals befindet sich der Code für die Ausführung einer Stored Procedure in mehreren Packages, die in unterschiedlichen Collections liegen. Durch die Option PACKAGE PATH können die Collection IDs durch Komma getrennt angegeben werden. Z.B. benötigen Java Stored Procedures die JDBC verwenden serverseitig einige Packages, die auf dem Tübinger und Leizpiger Großrechner in der Collection mit der ID NULLID liegen. Soll eine Stored Procedure nun aber in die Collection des Useraccounts (i.d.R. PRAKxxx) gebunden werden, so muss für die Ausführung sowohl auf die Collection mit der ID NULLID, als auch auf die Collection mit ID PRAKxxx zugegriffen werden[1].

[1] Interessanterweise scheint dies in der Version 9 von DB2 noch nicht zu funktionieren. Im Tutorial werden deshalb die Java Stored Procedures in die Collection mit der ID NULLID gebunden.

ASUTIME

Beschränkt die Rechenzeit der CPU (gemessen in CPU service units) für die Ausführung einer Stored Procedure. Wird das Limit überschritten, wird die Ausführung der Stored Procedure abgebrochen und SQLCODE -905, SQLSTATE 57014 zurückgegeben. Mit ASUTIME NO LIMIT (Defaultwert) wird kein Zeitlimit angegeben.

STAY RESIDENT

Durch STAY RESIDENT YES wird festgelegt, dass der binäre Programmcode von External high-level language Stored Procedures (außer Java), auch load module genannt, nach der Ausführung der Stored Procedure im Hauptspeicher gehalten wird. Diese Option eignet sich v.a. für Stored Procedures, die häufig aufgerufen werden. Durch STAY RESIDENT NO (Defaultwert) wird der Hauptspeicher, den das load modul belegt, nach der Ausführung der Stored Procedure freigegeben.

COMMIT ON RETURN YES/NO

Durch die Angabe von COMMIT ON RETURN YES wird erreicht, dass die Änderungen, die die Stored Procedure an Tabellen vorgenommen hat, sofort nach der Ausführung durchgeschrieben werden. Defaultwert ist COMMIT ON RETURN NO.

CALLED ON NULL INPUT

Gibt an, dass es erlaubt sein soll, die Stored Procedure aufzurufen, obwohl Parameter auf NULL gesetzt sind. Diese Einstellung ist standardmäßig (auch bei keiner Angabe) gesetzt und lässt sich auch nicht ausschalten. Es wird aber empfohlen, die Angabe wegen möglichen zukünftigen Änderungen zu machen.

WLM ENVIRONMENT

Gibt das WLM Application Environment an, das für die Ausführung der Stored Procedure einen Adressraum zur Verfügung stellt.

EXTERNAL NAME (nur für External SQL oder External high-level language Stored Procedures)

Name des externen Programms, das beim Aufruf der Stored Procedure ausgeführt wird.

B. Inhalt der beigelegten DVD

Auf der beigelegten DVD befinden sich im Verzeichnis *IBM Data Studio Developer* die Installationsdateien für das gleichnamige Produkt für Windows. Die Datei *Setup.exe* startet die Installation.

Im Verzeichnis *Anwendungsprogramm* befindet sich ein Archiv mit den Daten des Java-Anwendungsprogramms.

Die Dateien *Tutorial.pdf* und *Tutorial.doc* im Verzeichnis *Tutorial* beinhalten das Tutorial, das im Client/Server-Praktikum zum Einsatz kommt.

Die meisten der verwendeten Onlinequellen können im Verzeichnis *Quellen* gefunden werden.

Eine elektronische Version dieser Diplomarbeit ist außerdem im Verzeichnis *Diplomarbeit* enthalten.