IBM Software Group

DB2 Information Management Technical Conference: F04 – *Building a Test Database System*

David R. Schwartz
dschwar@us.ibm.com

DB2 Data Management Software

e business software

IBM DB2 Test Database Generator

- A powerful tool that provides several methods of generating test data for DB2
  - Can generate test data on demand
  - Can create scripts that can be run now, later, and repeatedly
- Supports multiple interfaces
  - ISPF, GUI, Offline
- Supports DB2 across multiple platforms
  - z/OS, Linux, Unix, Windows

Test Database Generator

For when you need to…

- Create test data in new or existing tables
- Debug application failures that are data dependent
- Copy a slice of data instead of all of the data
- Mask and censor sensitive data for testing
- Create a restructured database for testing
- Create test data in a variety of output formats

Agenda

1. An introduction to the Test Database Generator computing model

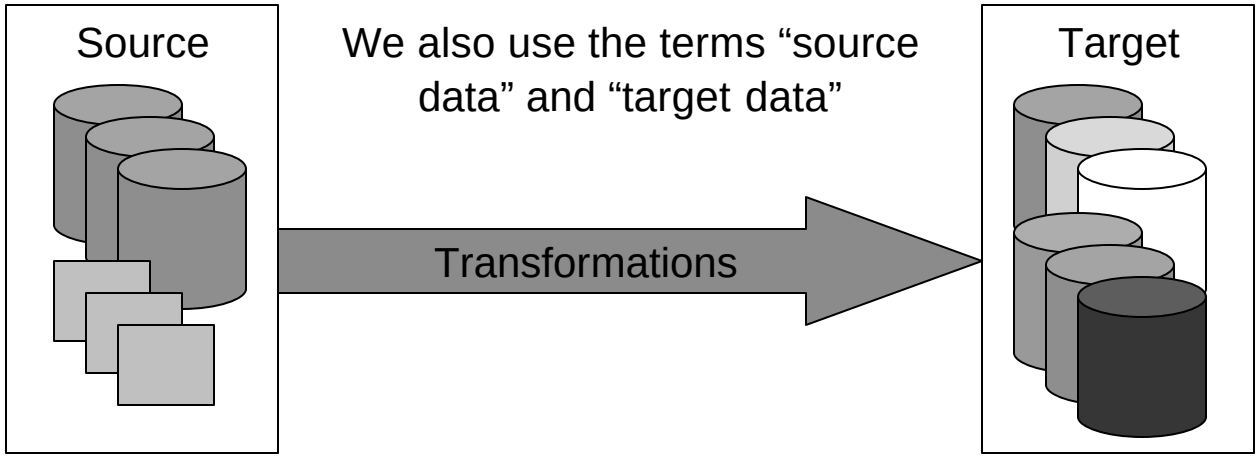
2. An overview of the Test Database Generator architecture

3. Some examples of generating test data using Test Database Generator

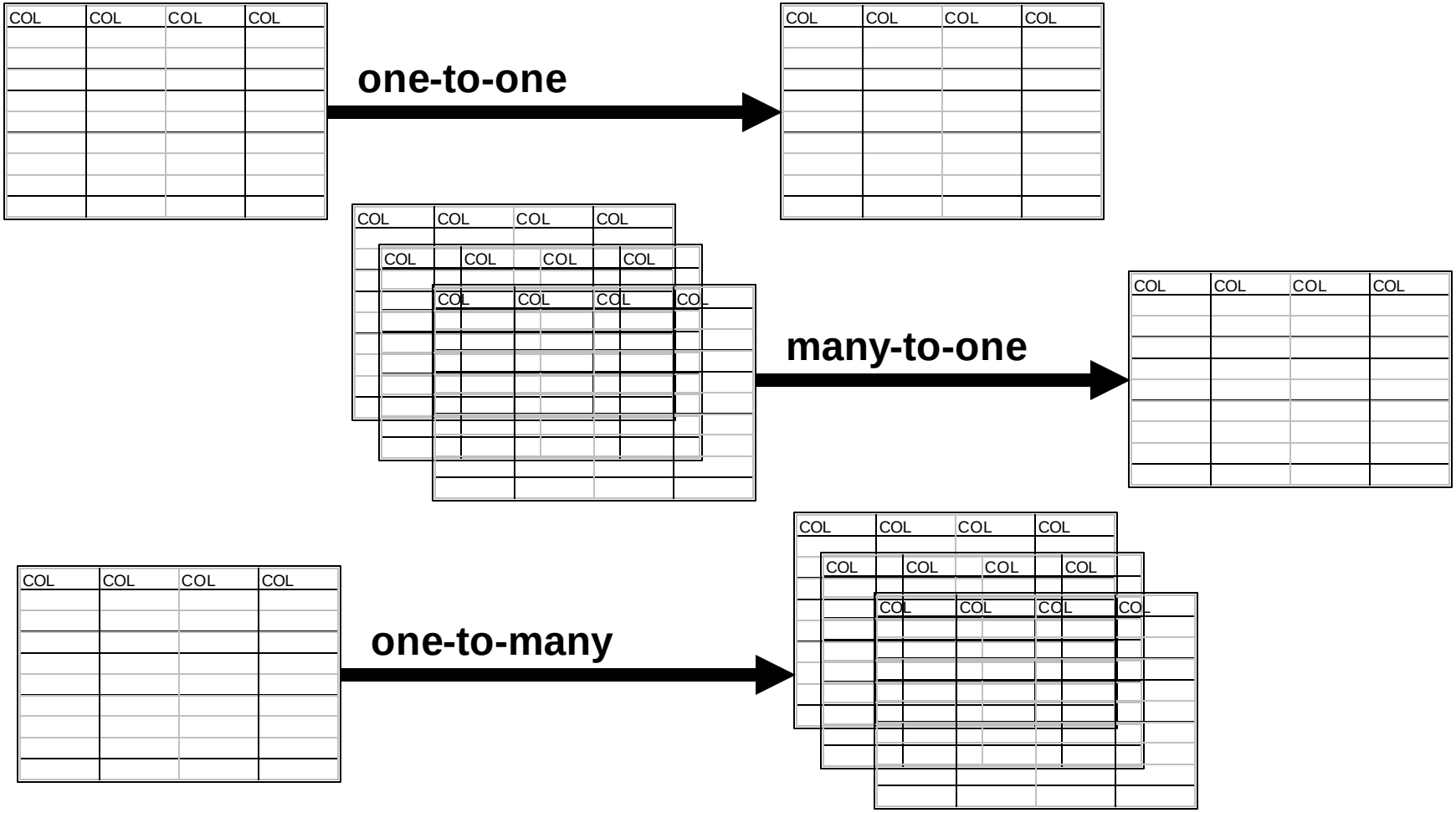
Note: Test Database Generator = TDBG

Introducing the TDBG computing model

1. Start with data that exists somewhere in your enterprise
2. Leverage knowledge of data relationships
3. Apply transformation rules
4. Create test data

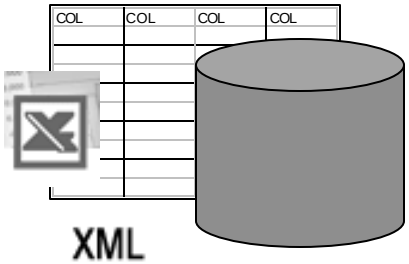


Start with data that exists means more than just make a duplicate copy…



Data sources

- Data sources can be database tables or files
- Database tables supported include
  - DB2
  - Informix (future)



- File types supported include
  - XML file (future)
  - CSV (comma separated value) file
  - Text file with any delimiter
  - Text file of fixed width

How data is accessed

Various protocols are supported, including…

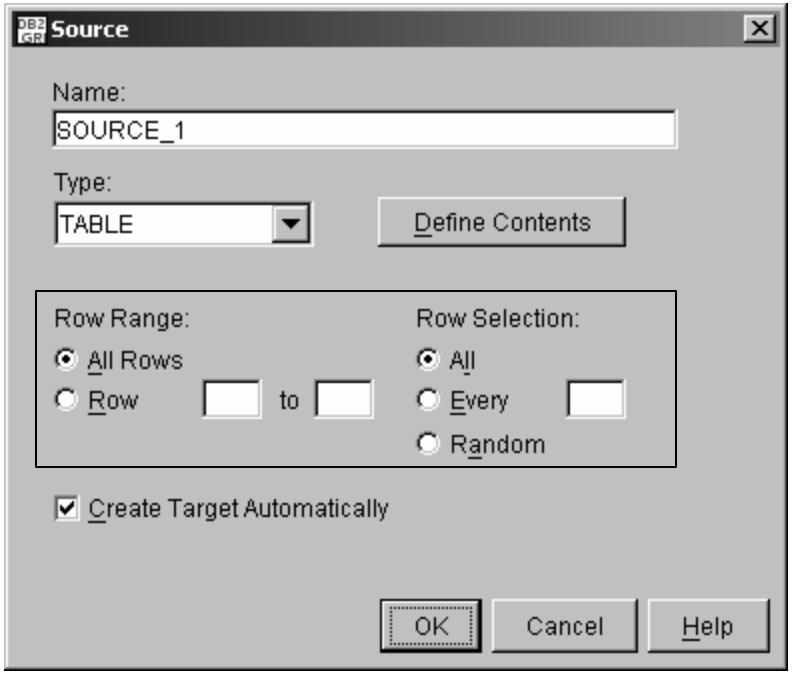
- JDBC
- HTTP
- FTP
- File

Data can be specified directly

- You can specify data values (rows and columns) directly or copy/paste data directly into TDBG using the GUI

Identifying the data you want to start with

- Multiple levels of filtering
- WHERE clause to limit result set
  - When source is a table
- Range
  - All
  - From/to
- Row selection within the range
  - Every row sequentially
  - Every nth row
  - Every row randomly



You don’t have to start with existing data

- Starting with existing data is an optional step
- You can choose to direct TDBG to create test data “from scratch”

Introducing the TDBG computing model

1. Start with data that exists somewhere in your enterprise
2. Leverage knowledge of data relationships
3. Apply transformation rules
4. Create test data

Would you like to include related tables?

- TDBG supports grouping of objects
  - By catalog RI
  - By application
  - And more
- TDBG discovers, through a variety of methods, related tables
- You can also define related sets of tables and edit the discovered relationships

Copy related data

- Copy all rows in a related group
  - Copy all rows that are related across an entire related set of tables
- Copy a slice of data across a related set of tables
  - Start with a specific customer and copy that row and all rows related to that customer across all related tables
- Copy only those rows that are orphaned
  - Help troubleshoot applications that have RI maintained by an application, not DB2
  - TDBG locates children with no parents

Introducing the TDBG computing model

1. Start with data that exists somewhere in your enterprise
2. Leverage knowledge of data relationships
3. Apply transformation rules
4. Create test data

Transformation rules

- Transformation rules define the target test data
  - How to generate test data from source data
  - How to generate test data from scratch
- Examples
  - Create a target column PHONE which is the combination of a country code (derived from COUNTRY file), an area code from TABLE1, and a phone number from TABLE2
  - Create a target column ACCT_BALANCE which is a random number that falls within a specified range
  - Create a target column that is exactly the PIN column with the 3rd and 5th positions replaced (masked) with the letter X

Scopes and sets

- You define your test data one target column at a time
- The scope of a transformation rule set is target column
- Multiple rules can be specified for each target column
- Transformation rules are applied in order
- Each rule can modify, replace, append, or preface the previous value to allow for incremental building of a target column

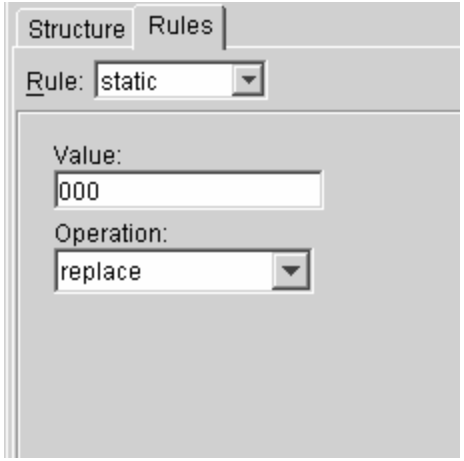
Transformation rules

TDBG currently provides seven (7) transformation rules

1. Static Values
2. Source Column Values
3. Data Lookup
4. Data Masking
5. Expressions
6. Random Values
7. Pattern Generation

Rule 1: Static Values

- Specify static data value
- Source data not used as input for this rule
- Examples
  - Set target column STATE to CA
  - Set target column SPEED to 85
  - Set target column EXP_DATE to 2030-12-31



Rule 2: Source Column Values

- Generate target column value based on source column value
- No transformation (use source data as-is)
- Copies a column value from the currently selected row in a source object
- Used to perform a “copy” function
- Used to supply initial column value from a data source
- TDBG Auto-Target feature makes use of this rule
  - Source Column Values rule is set automatically for each target column

Rule 3: Data Lookup

- Provides a method to replace data based on table lookups
- Uses the current value of the generated column as a key to the lookup
- Specify replacement column
- Example:
  - Source data has a product code which needs to be represented as a product name

    PROD_NAME = LOOKUP(PRODUCT_CODE in PRODUCT_TABLE)

Rule 4: Mask

- Replace or hide sensitive data
- Masks can be set with static text
  - Replace account number with a string of X's
  - Replace last four digits of License ID with 9999
- Mask can be set using a pattern rule (see rule #7)
  - Replace first character with a letter in the range A-F and then replace the next position with a number between 0 and 9
- Mask can be applied to the entire column or to a substring of the column

Rule 5: Expression

Expression provides a method to call database specific functions

- Supports any expression that can be evaluated by the DBMS in which TDBG is installed
  - String manipulation, calculations, etc.

- Examples
  - Calculate the current date / time / timestamp
  - Evaluate a mathematical expression

    Target column = source column * 1.1

Rule 6: Random

- Generate a random value
- Allows for creating random date, time, timestamp, integer, and decimal values
- Specify min / max ranges for the generated values
- Randomly generated values are propagated across related tables
  - Only applies to primary/foreign key relationships that are system or user defined
  - A DEPT_CODE that is randomly generated shows up in both the DEPT table and the EMPLOYEE table

Rule 7: Pattern

- Generate data based on a specified pattern
- Numeric pattern
  - [0-9] evaluates to any single-digit number
- Character pattern
  - [A-Z]*3 evaluates to any three-character uppercase string
  - Character patterns are randomly selected at generation runtime
- String pattern
  - (Mrs|Mr|Ms) evaluates to ‘Mrs’, ‘Mr’, or ‘Ms’.
  - (C[ATO] | A[KLR]) evaluates to ‘CA’, ‘CT’, ‘CO’, ‘AK’, ‘AL’, or ‘AR’

Introducing the TDBG computing model

1. Start with data that exists somewhere in your enterprise
2. Leverage knowledge of data relationships
3. Apply transformation rules
4. Create test data

What can TDBG create?

TDBG can generate test data as a(n)…

- Comma separated value (.csv) file
- Text file with fixed width columns
- Text delimited file
- File of SQL INSERT statements
- DB2 for z/OS internal load format file
- XML file

TDBG can also…

- Generate test data and directly insert it into DB2

You can also choose…

- How many rows of test data to generate
- The exact structure of the target table(s)
  - Inherit from source (auto-target feature)
  - Specify column definitions
- To automatically generate DDL needed to create the target table(s)
  - Includes primary and foreign keys
  - Can be modified

TDBG architecture

- Components
- Data profiles
- Operating environments
- XML
- Unicode

TDBG components

- TDBG Server
  - Installed as a set of DB2 stored procedures
  - Can run on any DB2 running on z/OS, Linux, Unix, Windows
  - Connects to and reads from data sources

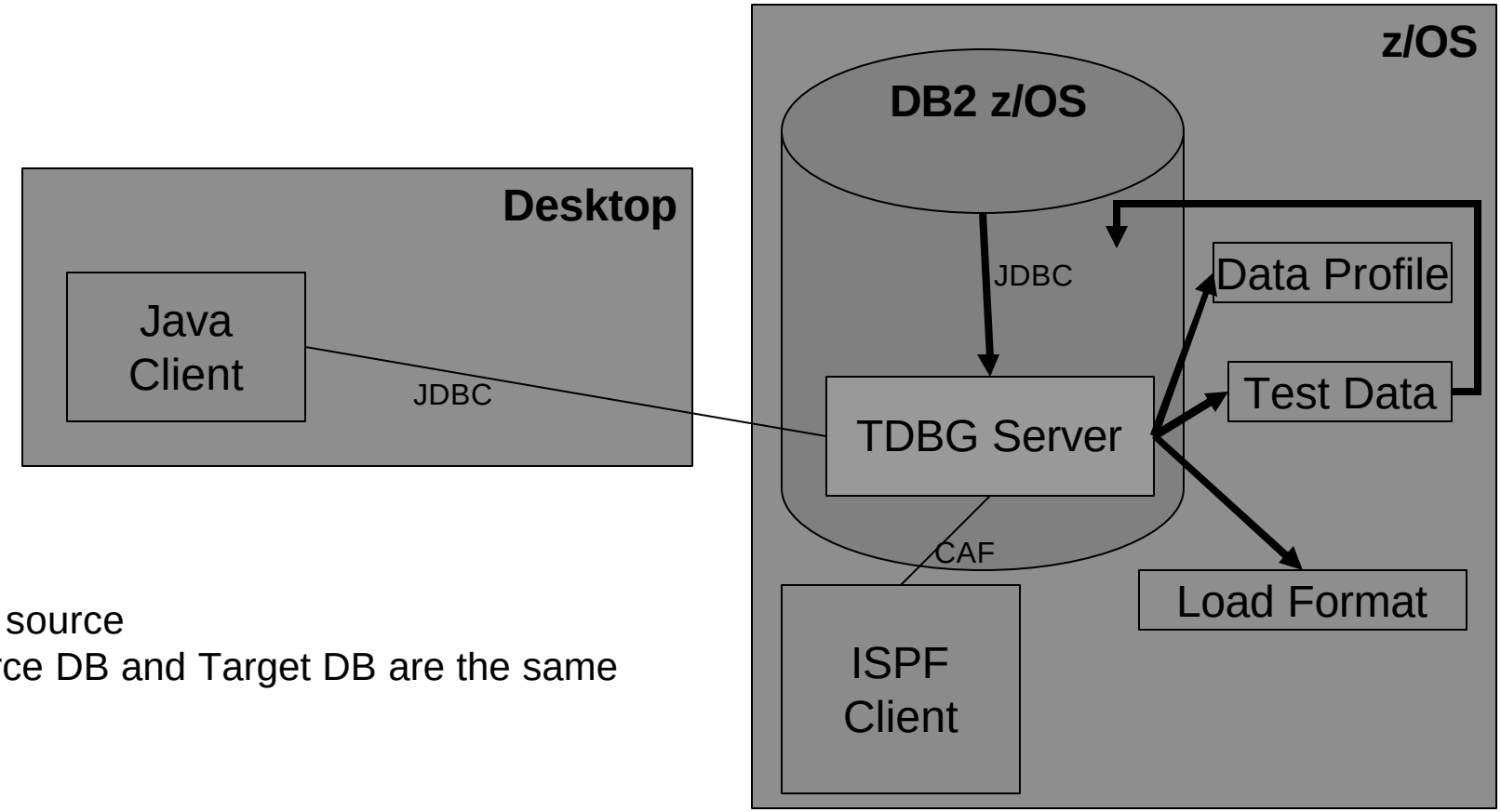
    What databases can be accessed is governed by the product license
  - Is the component that reads Data Profiles and generates target test data

- TDBG Client
  - ISPF client running on z/OS
  - GUI (Java) client running on Linux, Unix, Windows
  - Connects to TDBG Server
  - Provides a user interface wizard and panels to guide you through the creation of a Data Profile

Data profiles

- Data Profiles define the generation process
  - Source data objects used to seed generation.
  - Definition and relational rules of the targets.
  - Describes how data will be copied, filtered, masked, and/or transformed.
- Data Profiles are stored in the HFS by the server
- Data Profiles are written in Test Database Generator Markup Language (a.k.a. GRIML)
  - GRI: Internal product code
  - ML: Markup Language
  - GRIML is an XML-based markup language

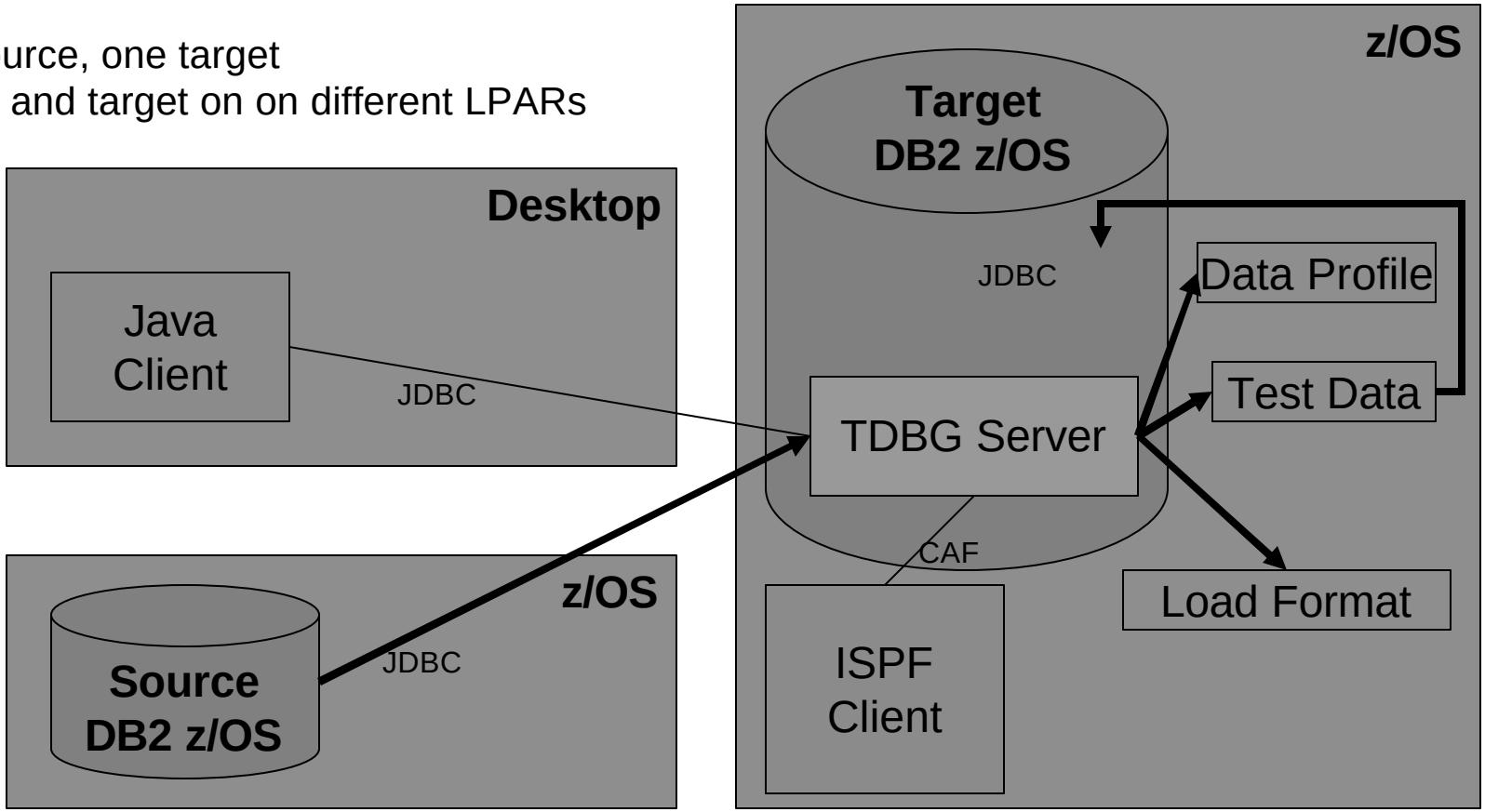
Operating environments: A z/OS example



-One source
-Source DB and Target DB are the same

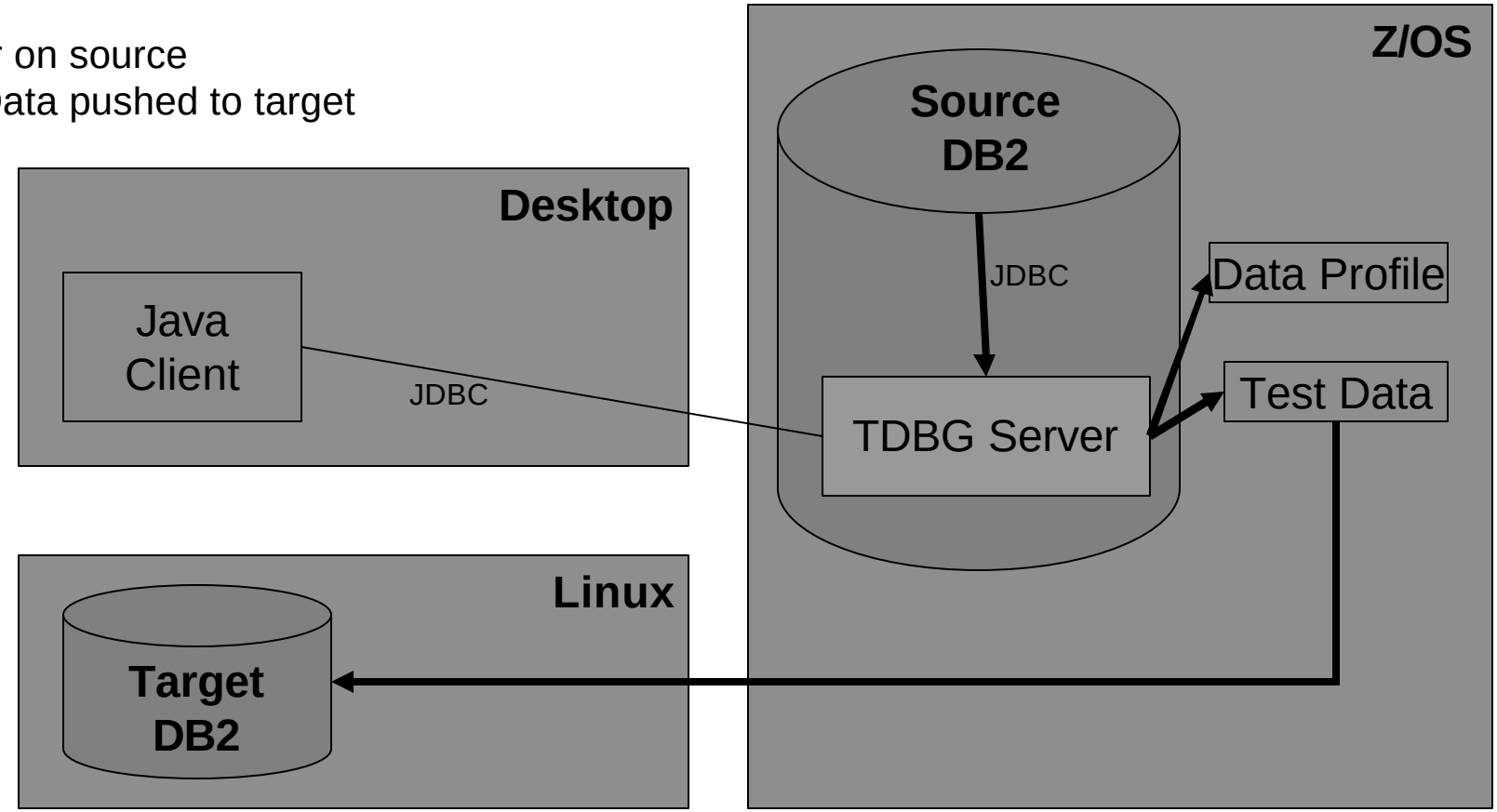
Operating environments: A z/OS example

-One source, one target
-Source and target on on different LPARs



Operating environments: A Multiplatforms example

-Server on source
-Test Data pushed to target

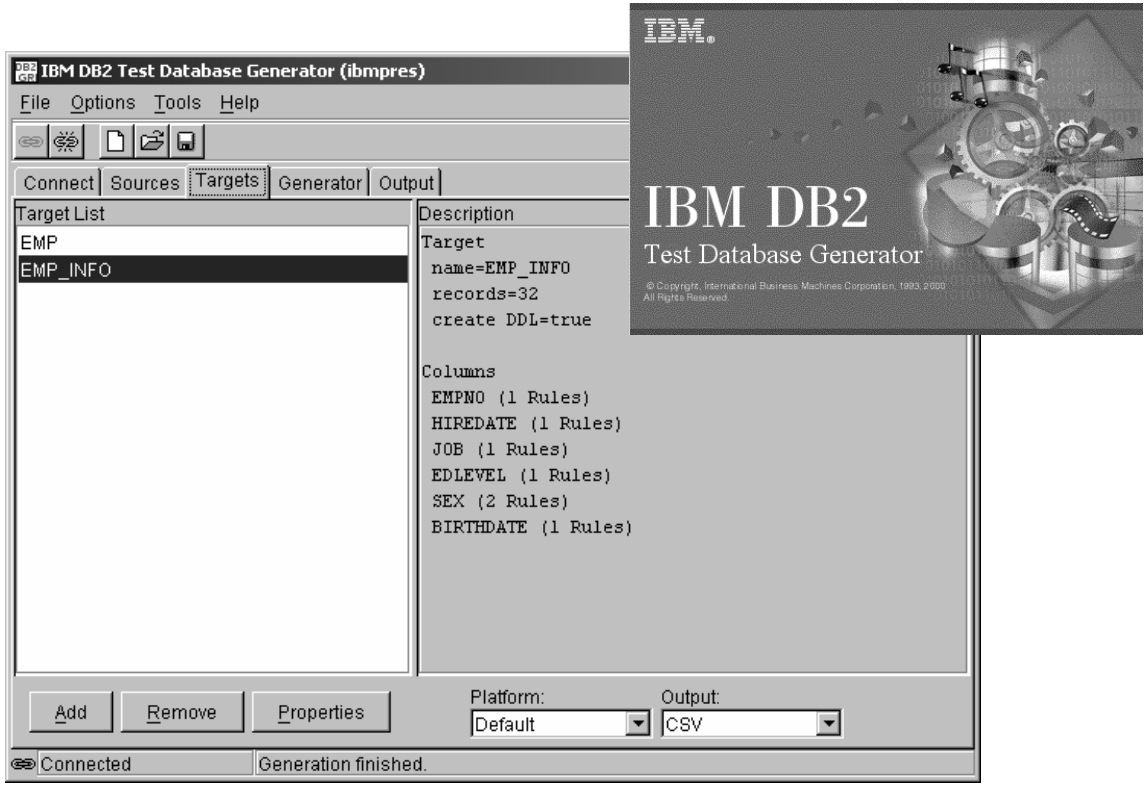


Operating environment: ISPF

```
GRI$MAIN  V2R1 --------  Test Database Generator  ------- 2003/05/06
Option ===>


  Current Server: W32B     Current SQLID XYZHAW       User:  XYZHAW
 ---------------------------------------------------------------------------

  Data Profile:

   1   Sources                                                  Sources: 0
   2   Targets                                                  Targets: 0
   3   Generator
   4   Load Data Profile
   5   Save Data Profile
   6   Reset Data Profile

   S   Setup
   A   About
   X   Exit
```

Operating environment: Java GUI

- Java application running on your desktop
- Can be launched from DB2 Control Center



TDBG and XML

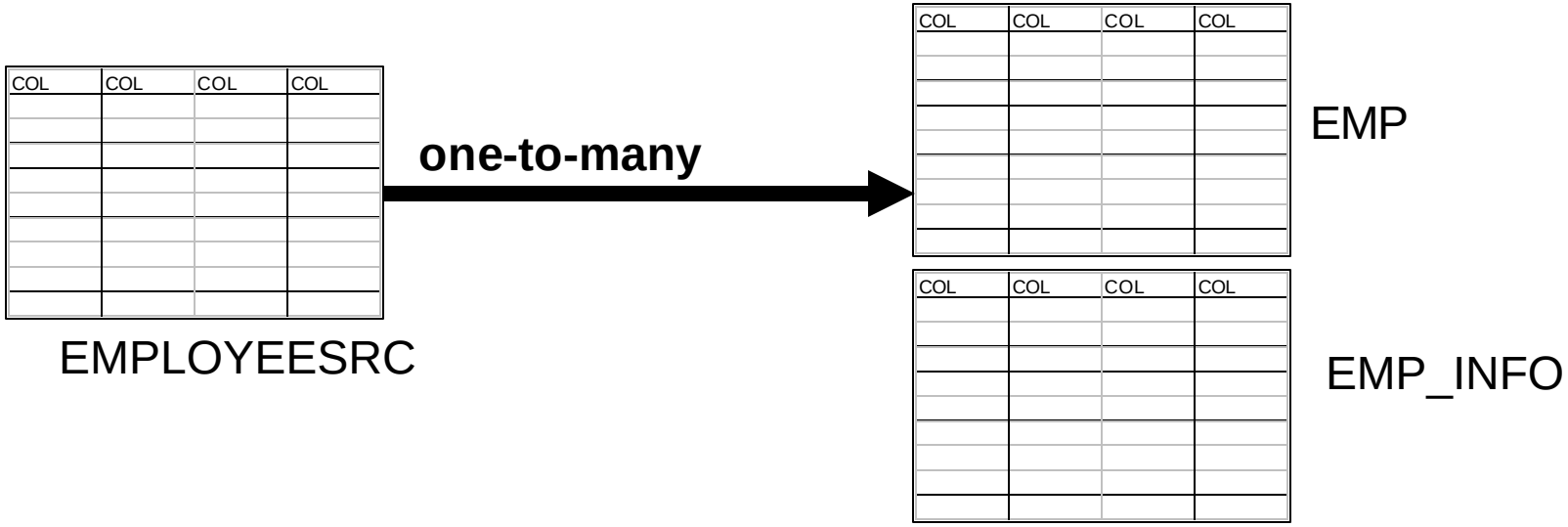
- Data Profiles
  - Data Profiles are stored in an XML document
  - You can create a Data Profile manually using an XML editor
- XML data sources (future)
  - You can specify any row-and-column oriented XML document as a source of data
  - For example, a query result from DB2 Web Query Tool
- XML target
  - Test data can be generated into an XML document

UNICODE

- Data profiles are written in UNICODE in UTF-8
- Targets are written in UNICODE in UTF-8
- Provides platform independence and portability for your test data
- All UTF-8 characters are supported

Use case

- Creating two new tables of test data
- Source data will be read from the EMPLOYEESRC table
- Two target tables (EMP and EMP_INFO) will be created



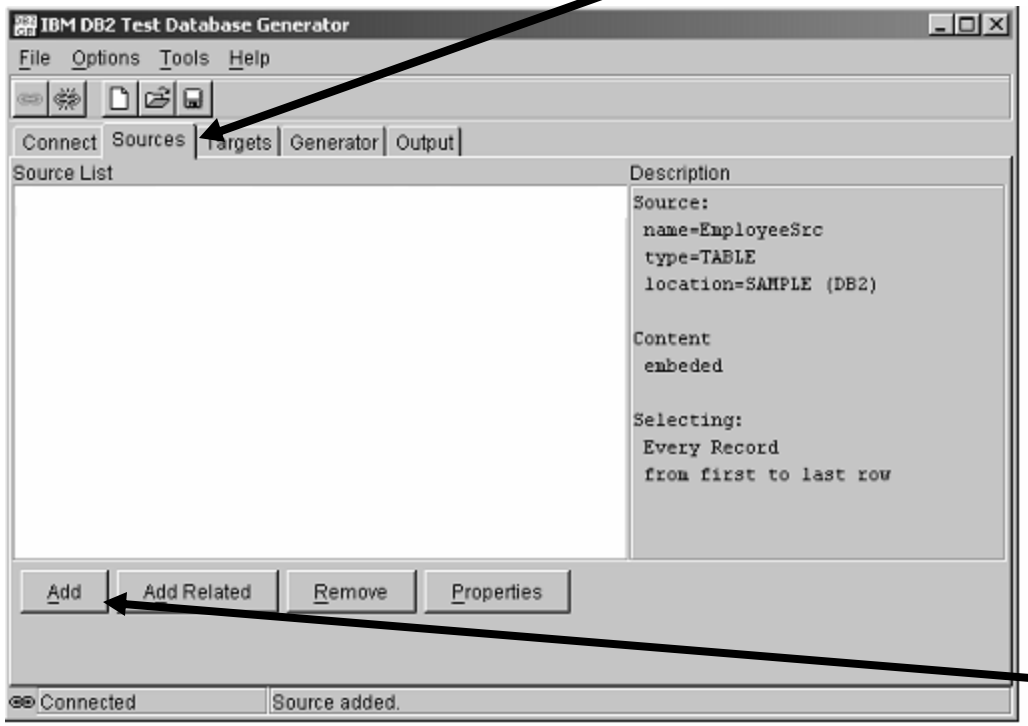
Use case

The steps we'll follow are

- Specify source
  - The EMPLOYEESRC table consisting of 14 columns
- Specify targets
  - Create two targets (EMP and EMP_INFO)
  - EMP will have the first 6 columns from EMPLOYEESRC
  - EMP_INFO will have the next 5 columns from EMPLOYEESRC (plus the column that relates the rows – EMPNO). The remaining columns in EMPLOYEESRC are not needed in the test environment.
- Specify rules
  - To copy specific columns and to protect sensitive information
- Generate test data
  - Watch a generation session

Use case : Specify source

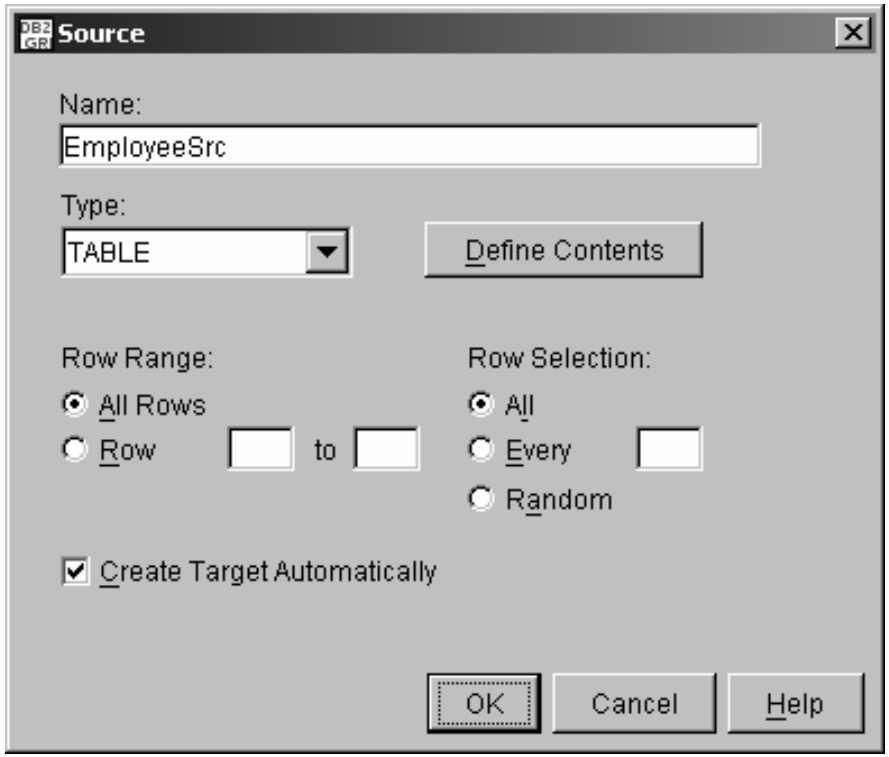
- Start from an empty Source List on the ***Sources*** tab



Add source using the Add button

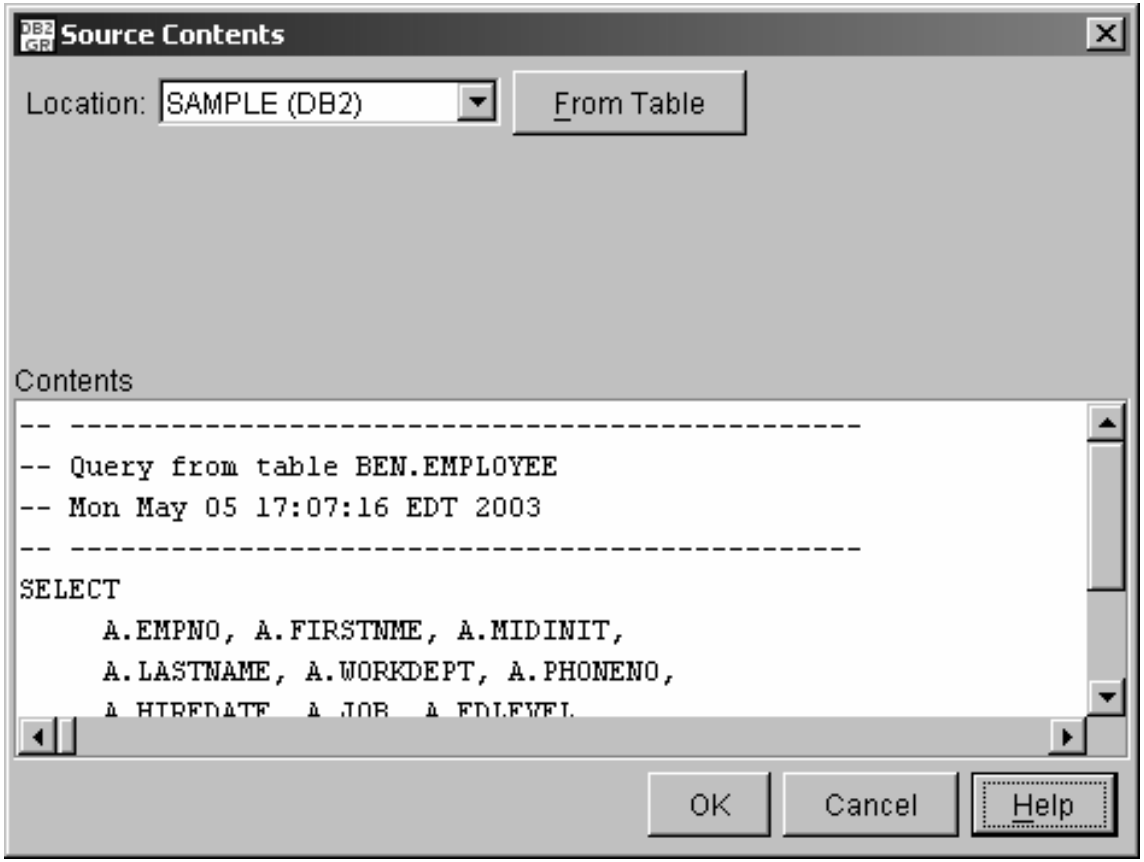
Use case : Specify source

- Define a source EmployeeSrc as “all rows from a TABLE”



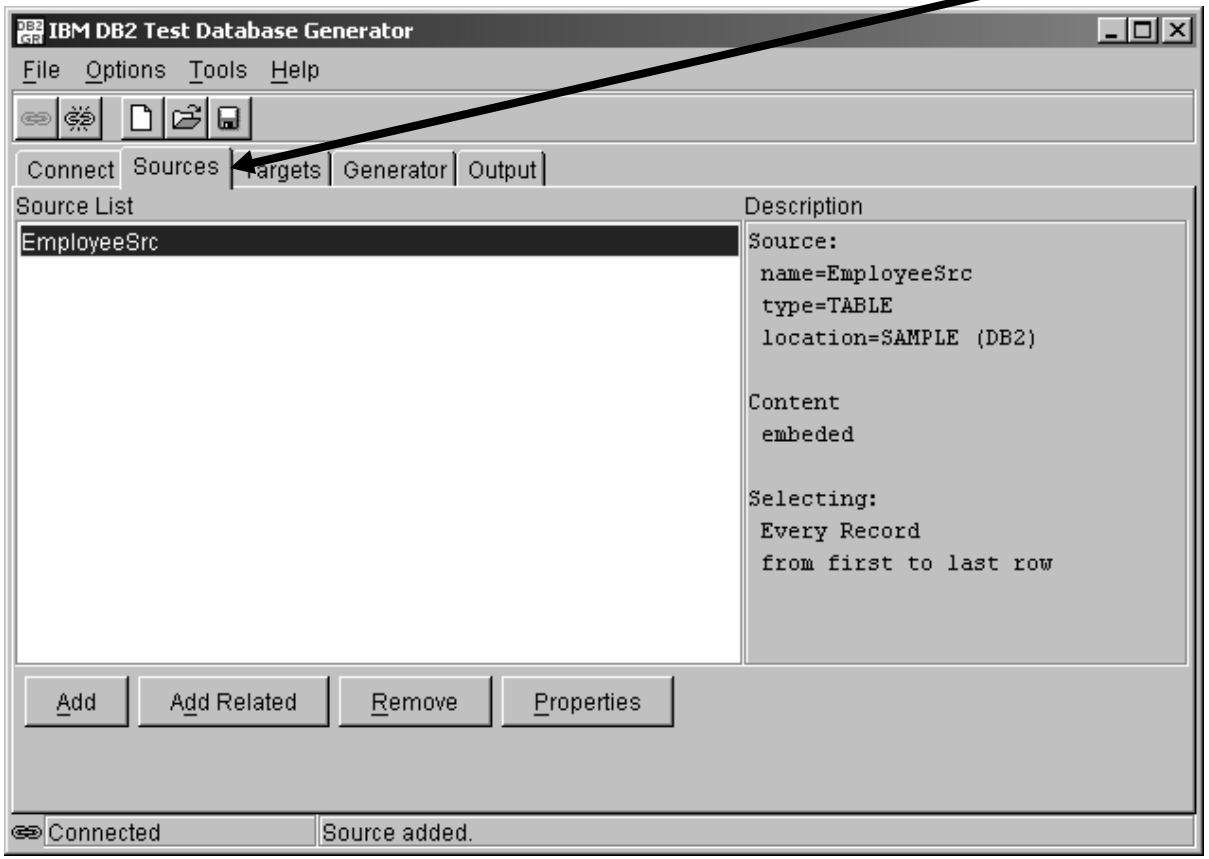
Use case : Specify source

- Here is the query that will be used to obtain the source data



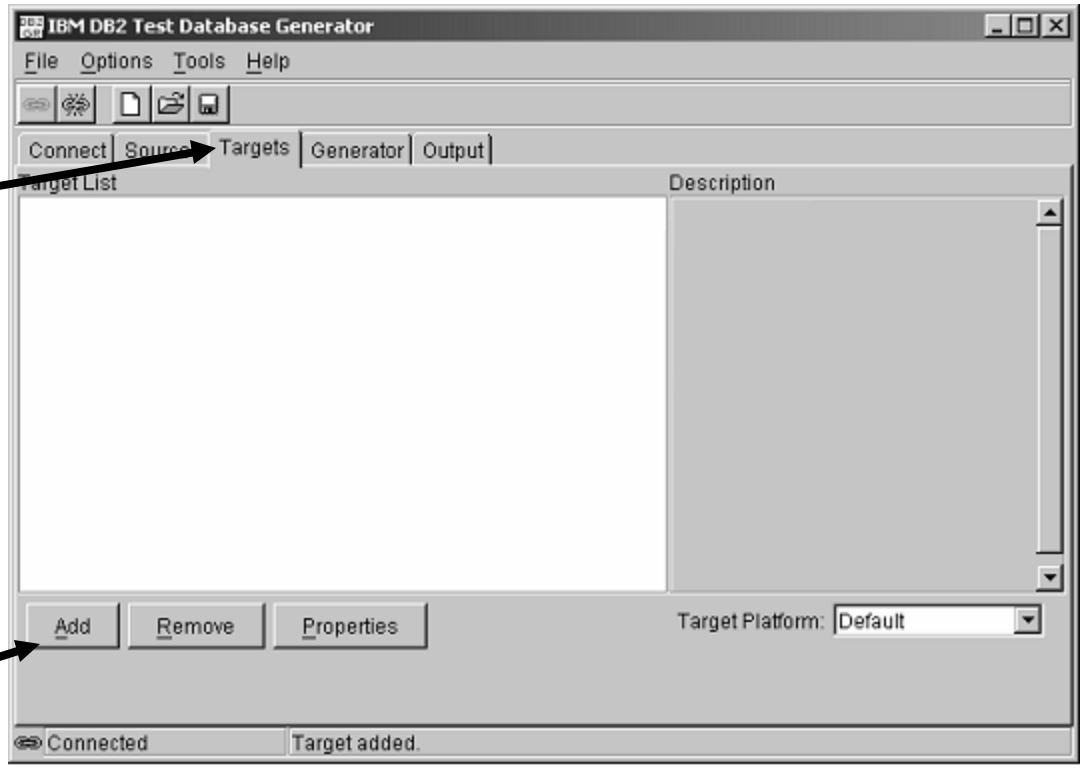
Use case : Specify source

- The source appears in the main window on the ***Sources*** tab



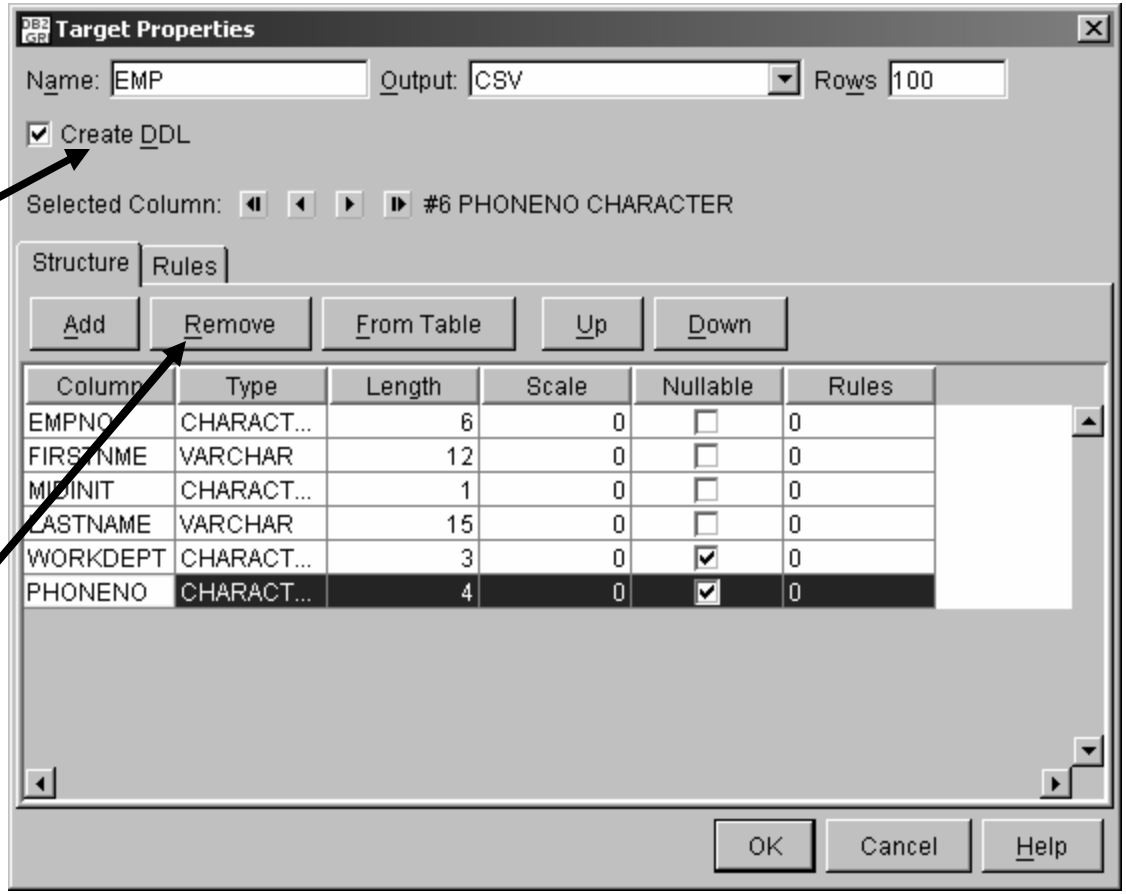
Use case: Specify targets

- Start with an empty Target List in the ***Targets*** tab
- Add first target using the Add button



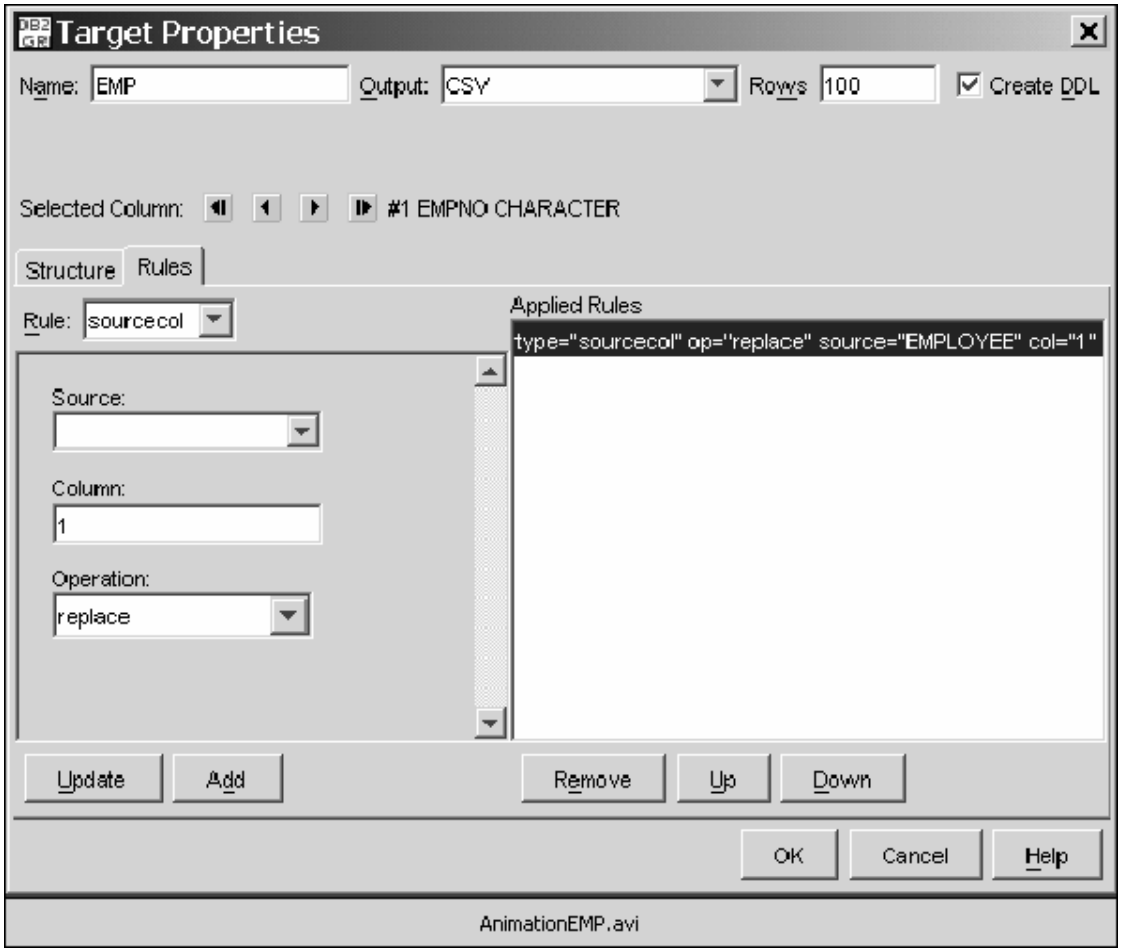
Use case: Specify target 1

- Add a target for the EMP table
- Select the Create DDL check-box so that DDL will be created for this target
- Remove unwanted columns from the source definition



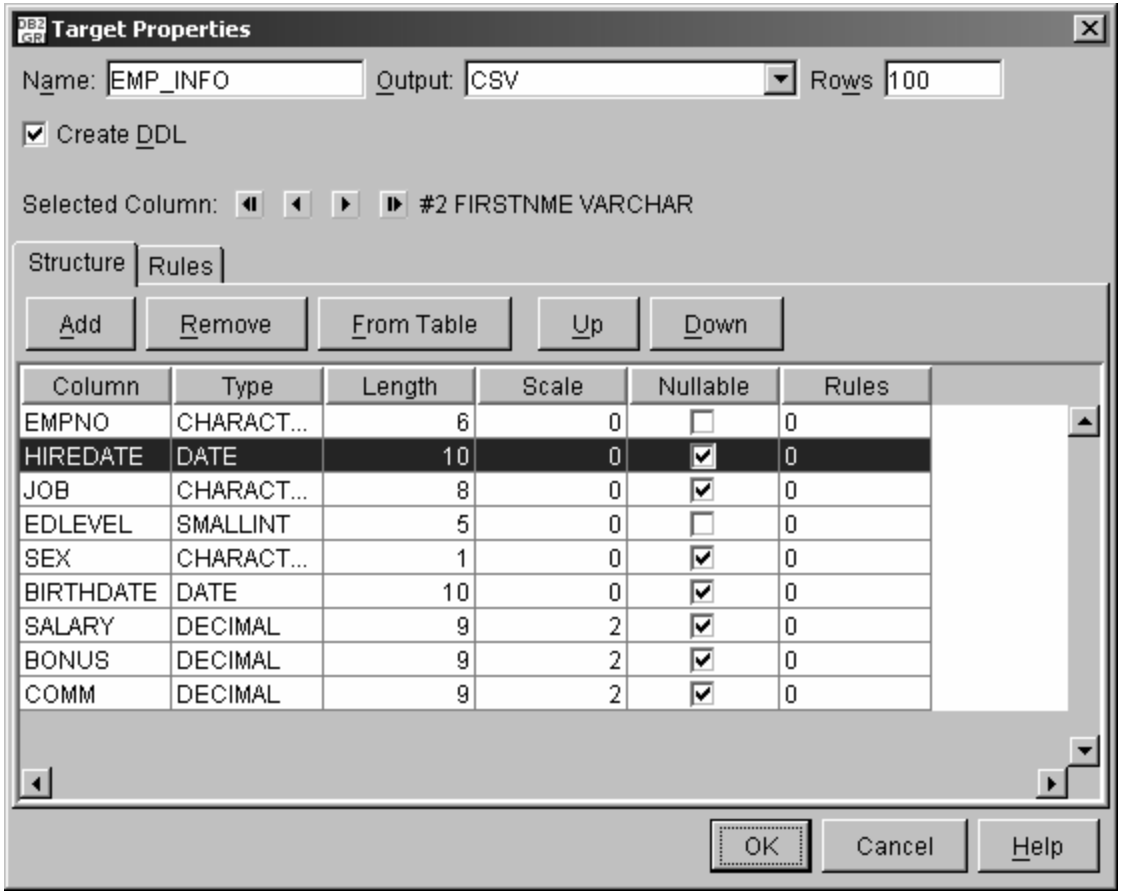
Use case: Specify rules for columns target 1

- SourceCol rule applied to each column
- PHONE is not obtained from source
- Instead, PHONE is generated using a combination of pattern, static, and random rules



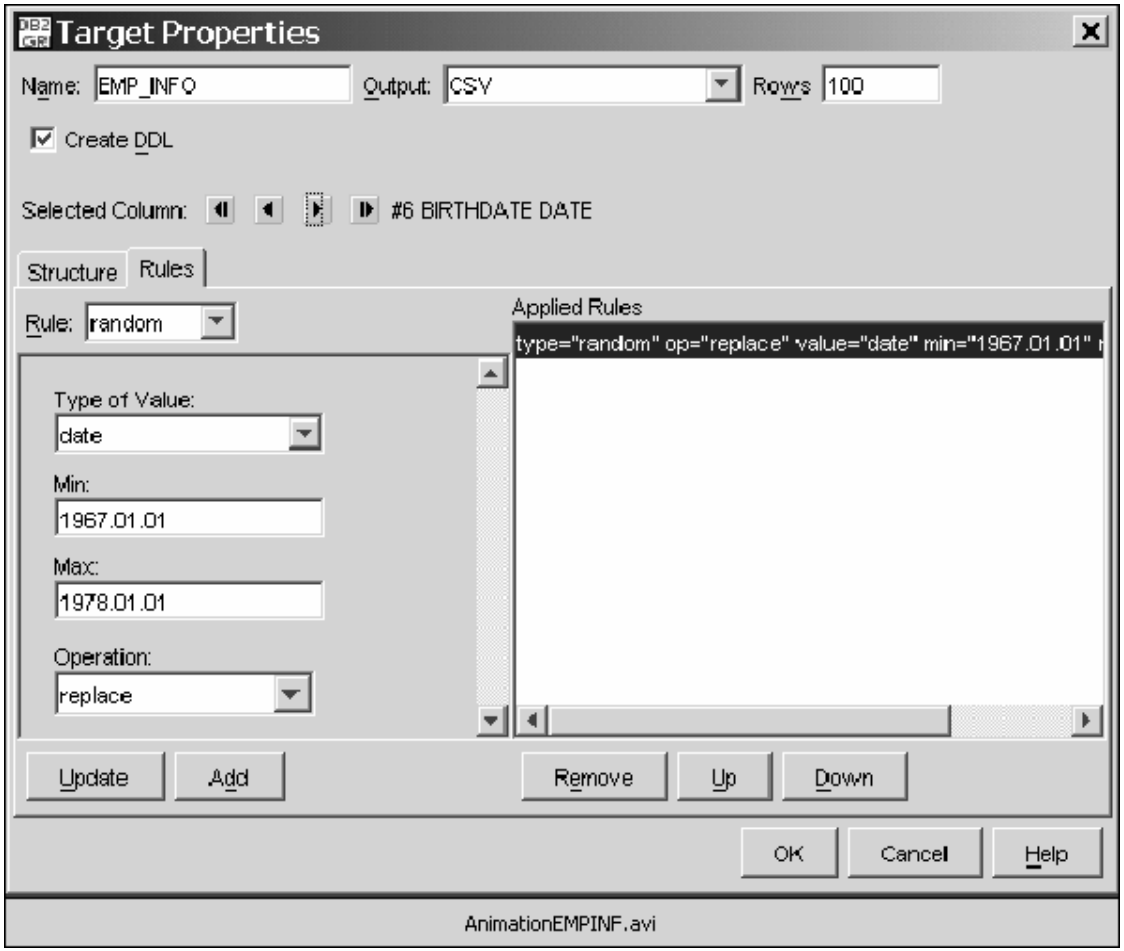
Use case: Specify target 2

- Add a target for the EMP_INFO table
- Select the “Create DDL” check-box so that DDL will be created for this target
- Remove unwanted columns from the source definition



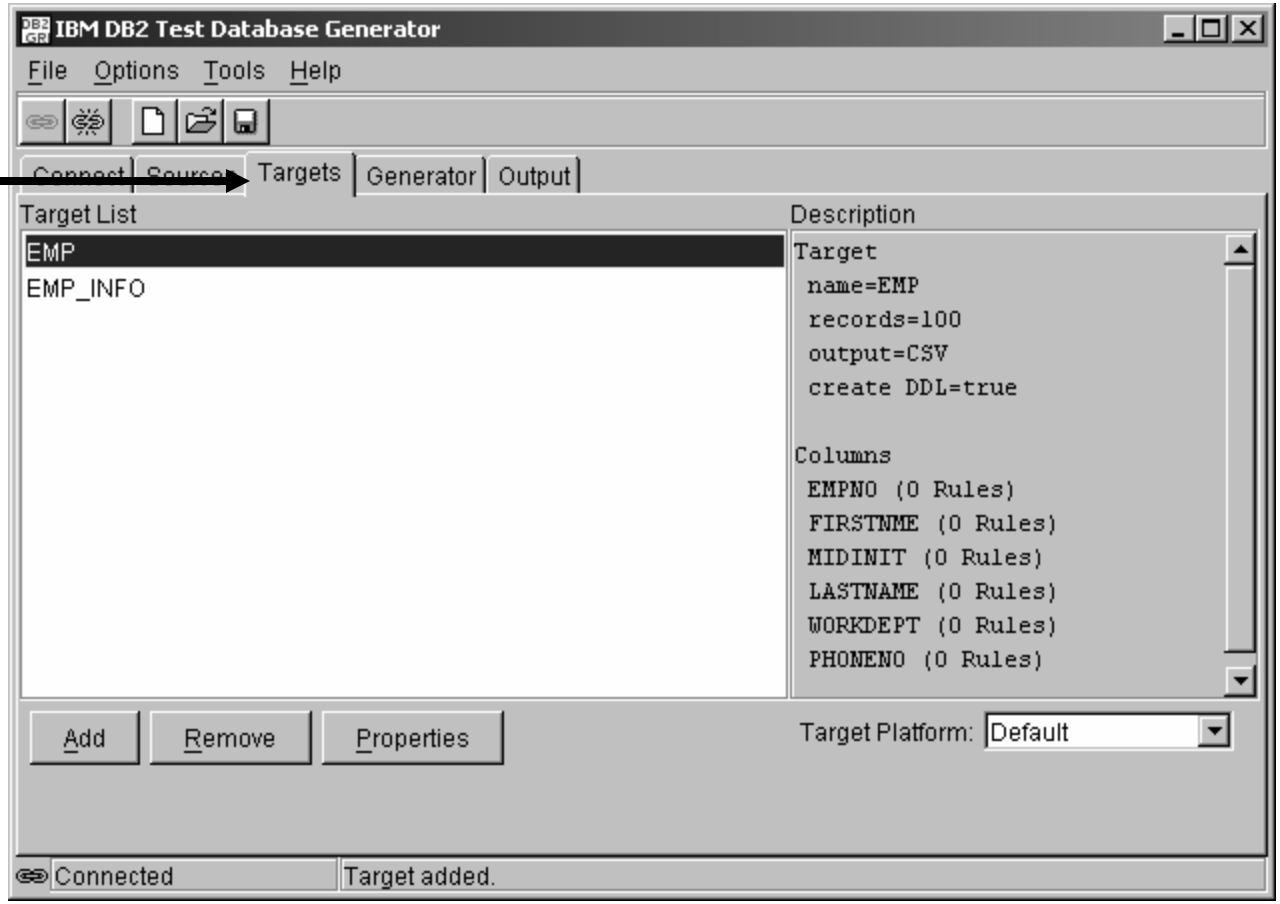
Use case: Specify rules for columns in target 2

- SourceCol rule applied to each column
- BIRTHDATE is not obtained from source
- Instead a random BIRTHDATE is generated in a range starting from January 1, 1967



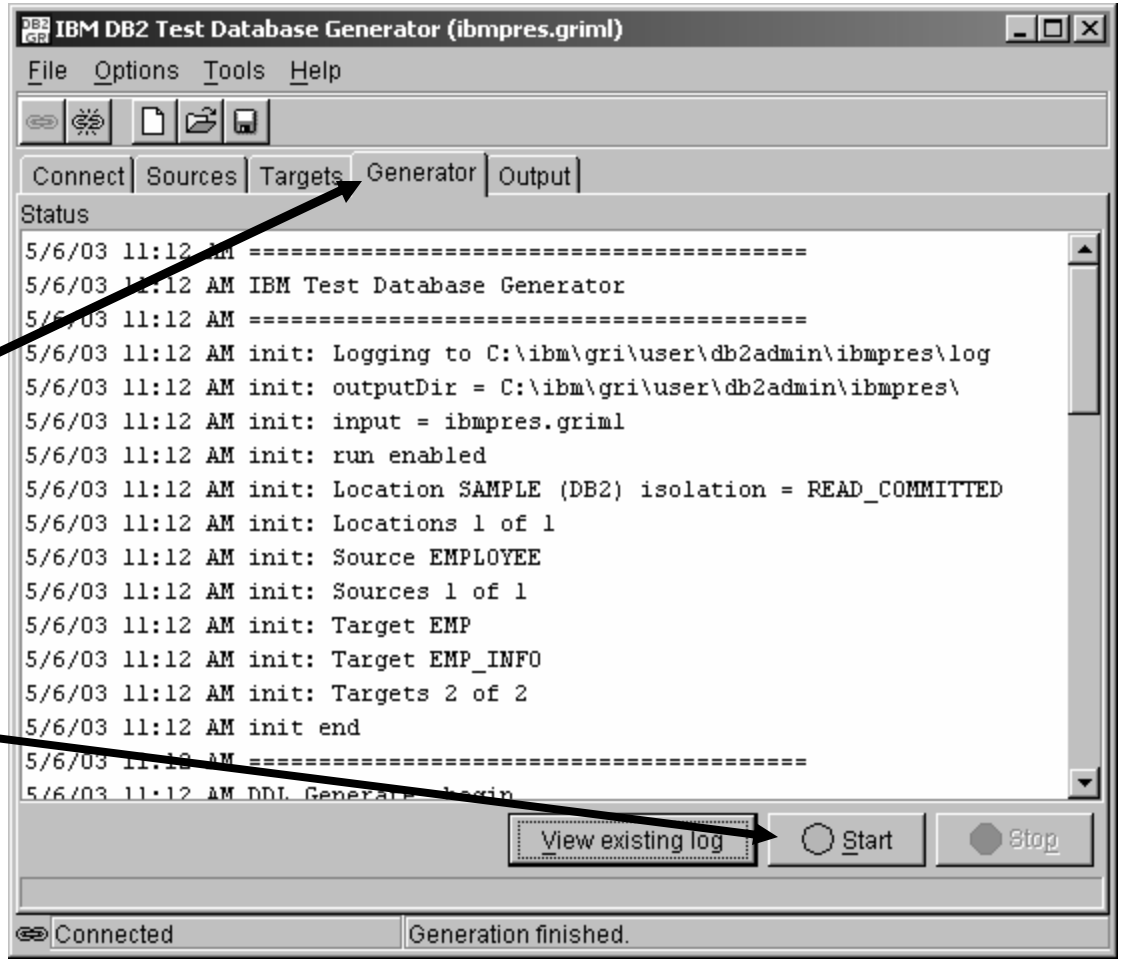
Use case: Specify targets

- The targets appear in the main window of the ***Targets*** tab
- An at-a-glance description appears in the right-hand frame
- You can save this data profile
- You can generate test data



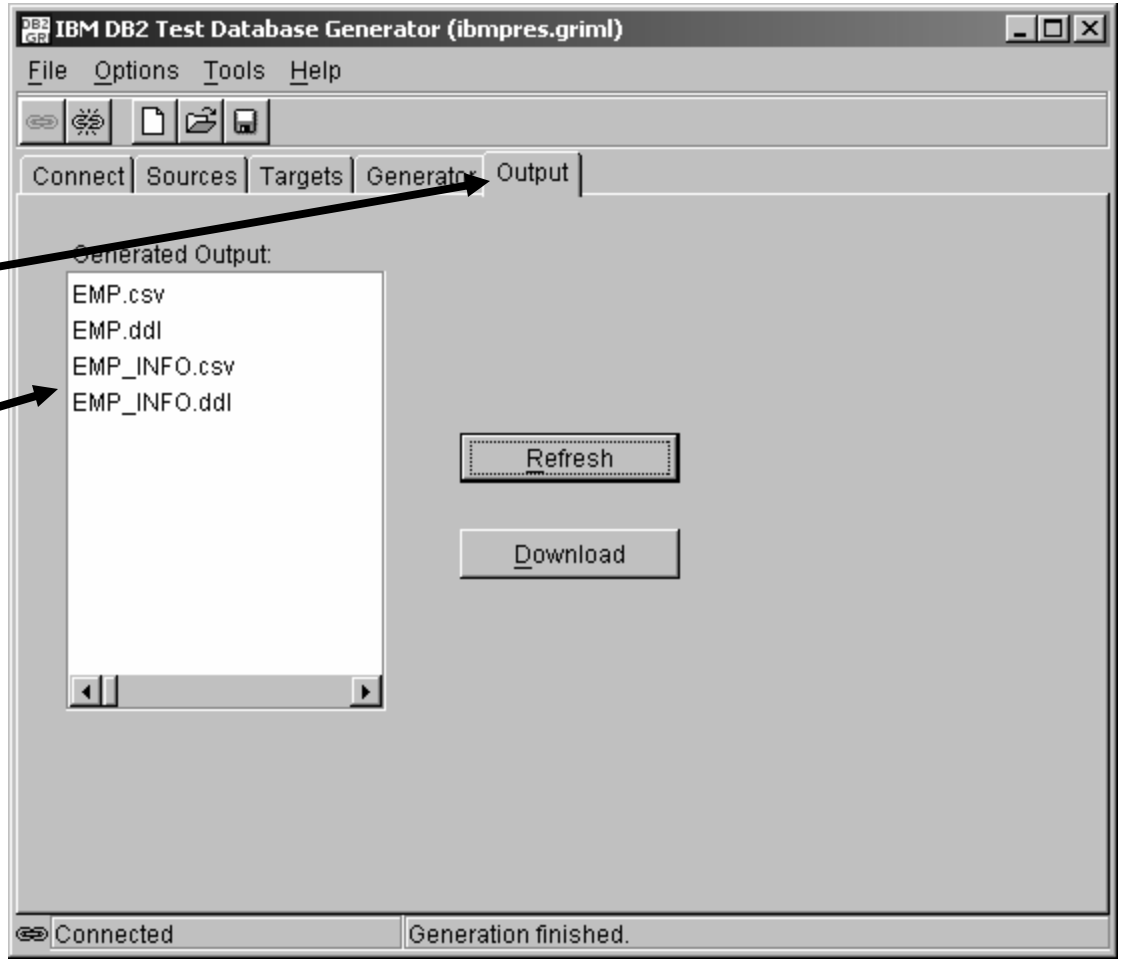
Use case: Generate test data

- Watch the log as your targets are created
- The log appears in the main window of the ***Generator*** tab
- Sessions can be started and stopped using the buttons at the bottom

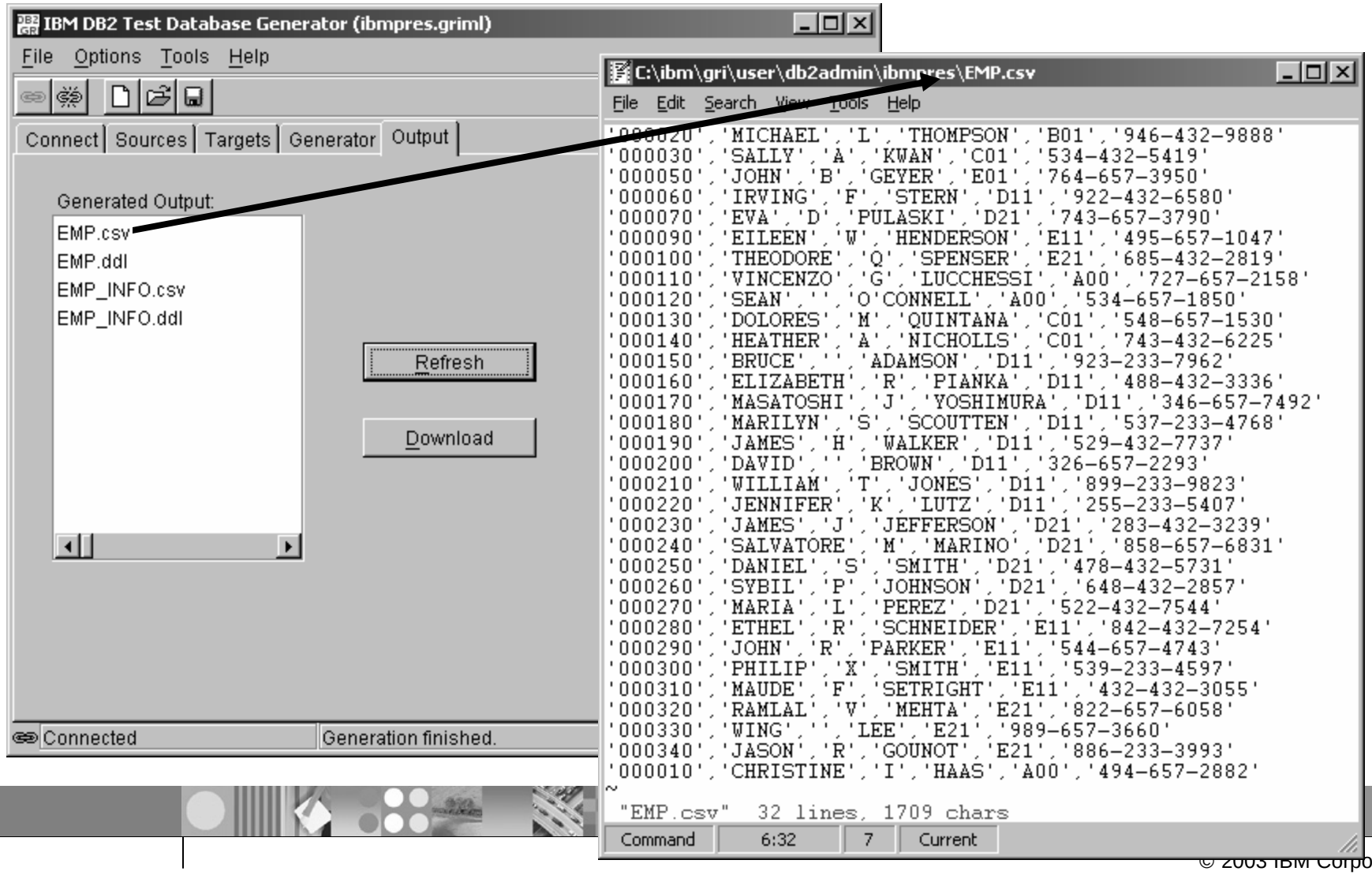


Use case: View list of targets

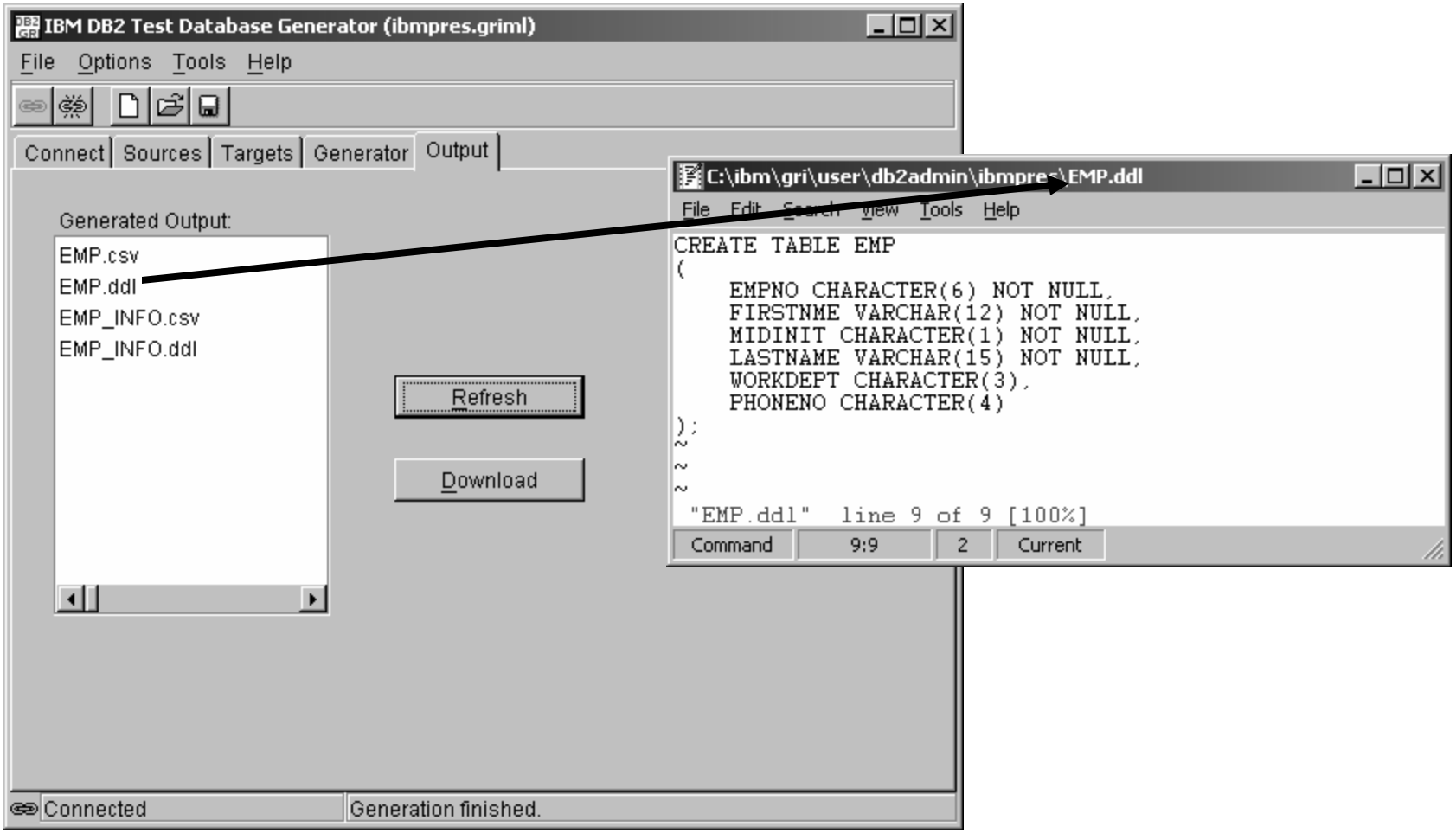
- The generated target files appear in the main window of the ***Output*** tab
- DDL files and target data files



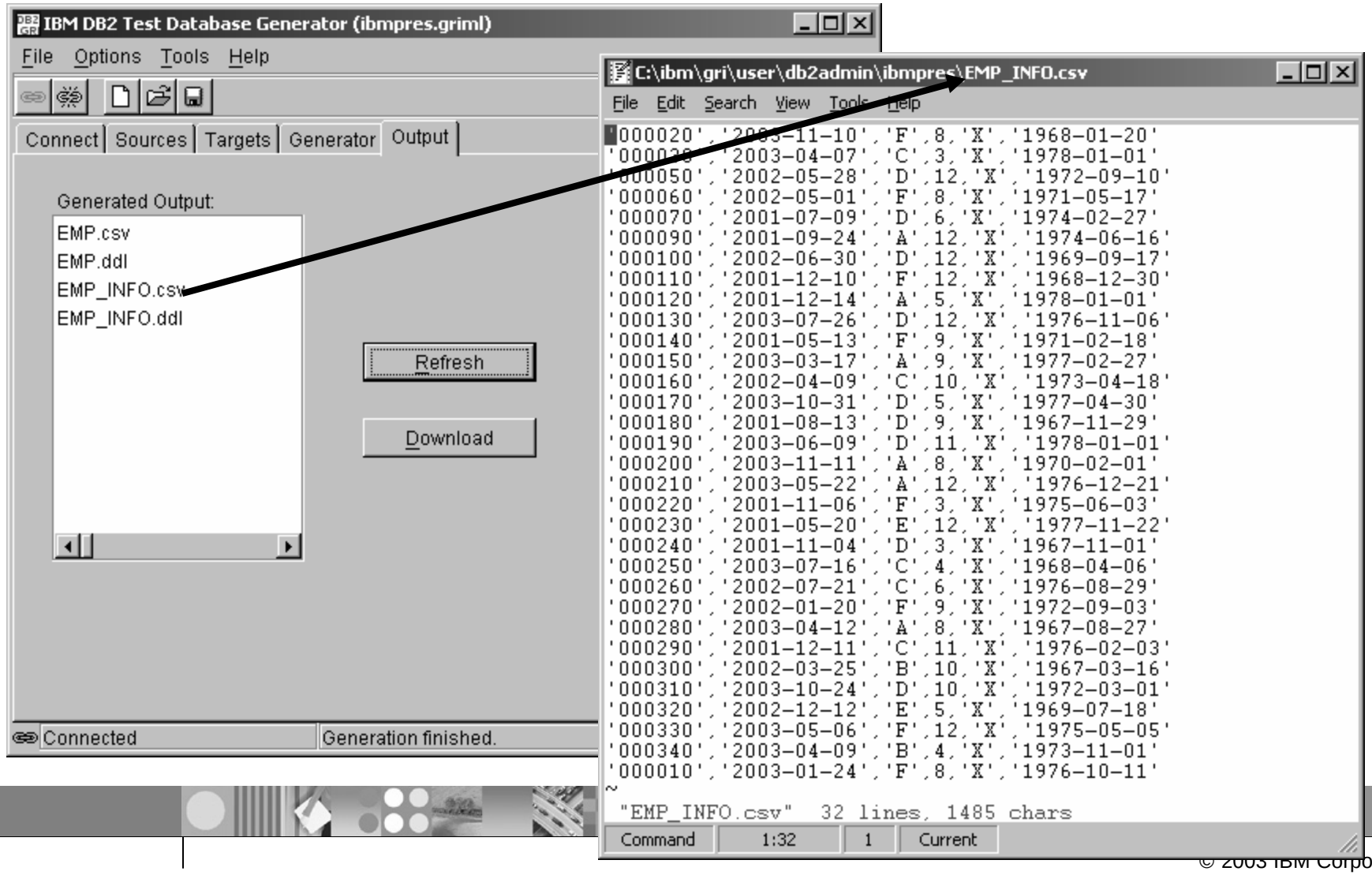
Use case: View generated test data



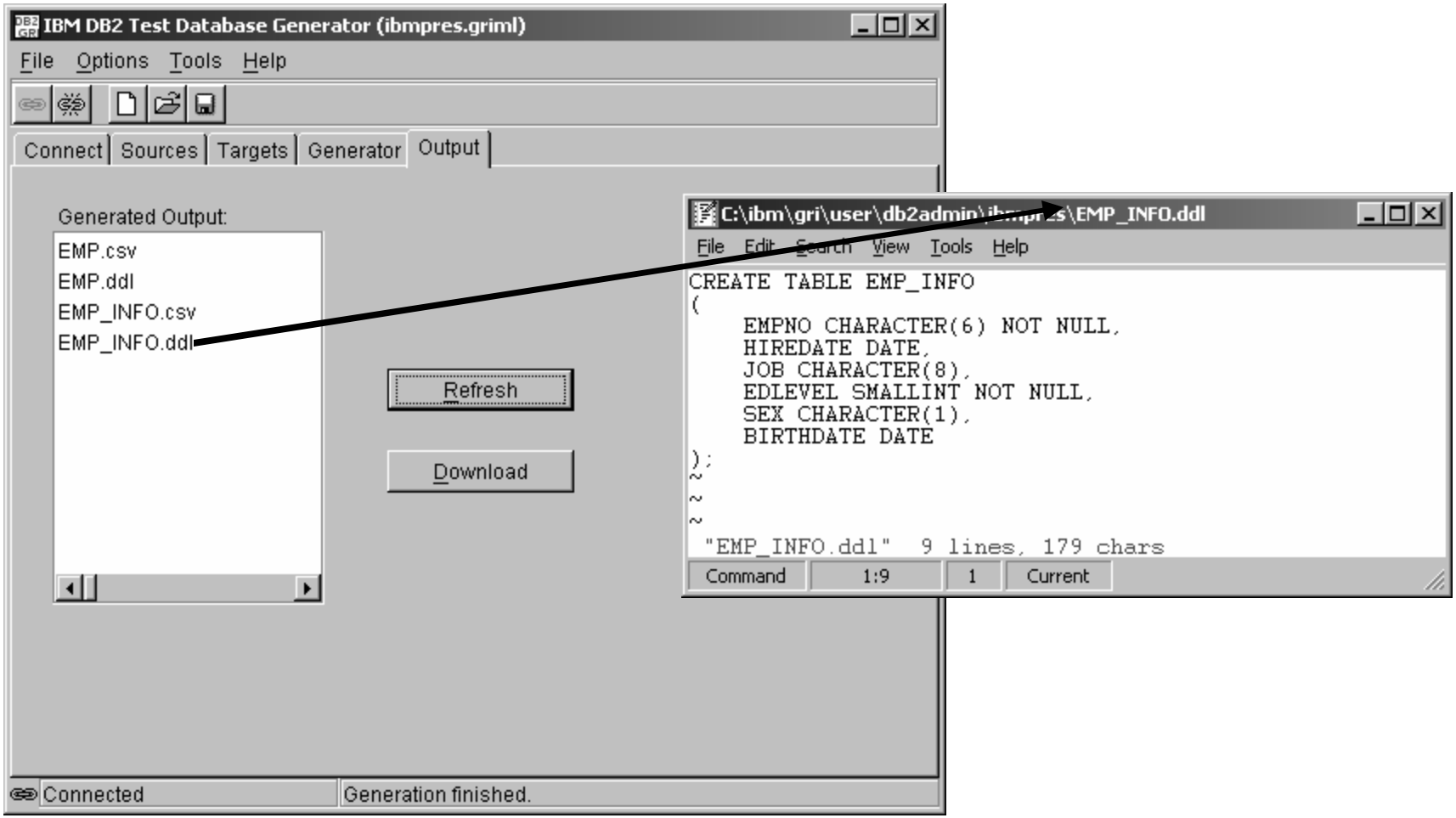
Use case: View generated DDL



Use case: View generated test data



Use case: View generated DDL



END