ConditionedReinforcement

A Whisker client

---

by Rudolf Cardinal

www.whiskercontrol.com

Distributed by Campden Instruments Ltd (www.campden-inst.com)

ConditionedReinforcement

Printed: May 2017 in Cambridge, UK

Creator (Whisker)

Rudolf N. Cardinal

Design and Programming (Whisker)

Rudolf N. Cardinal

Michael R. F. Aitken

Legal Advisor (CUTS)

Adjoa D. Tamakloe

Sales (Campden)

Julie Gill

Contacting the authors:

For information about Whisker, visit http://www.whiskercontrol.com/.

If you have sales enquiries about Whisker, contact Campden Instruments Ltd at http://www.campden-inst.com/.

If you have comments or technical enquiries that cannot be answered by the sales team, contact the authors:

Rudolf Cardinal (rudolf@pobox.com)
Mike Aitken (m.aitken@psychol.cam.ac.uk)

Table of Contents

Foreword 1

Part I ConditionedReinforcement 2

1 About ConditionedReinforcement 2
2 Required devices 2
3 Using the task 4
4 Parameters 6
5 Creating a new ODBC source 8
6 Database structure 12

Index 13

Foreword

WARNING

Whisker is a system designed for research purposes only, and should never be used to control medical apparatus or other devices that could endanger human life.

DISCLAIMER

The authors, copyright holders, and distributors disclaim all responsibility for any adverse effects that may occur as a result of a user disregarding the above warning.

1 ConditionedReinforcement

1.1 About ConditionedReinforcement

Purpose

Acquisition of a new response with conditioned reinforcement.

Software requirements

Requires Whisker v2.0 or greater.

Data storage

- Text-based output to disk.
- ODBC data storage to a database (supplied).

Author

Rudolf Cardinal (rudolf@pobox.com).

Copyright

Version history

- Version 0.1: 25 October 2004.
- Version 0.2: 11 November 2005. CS0 and discriminative stimulus conditioning schedule.
- Version 0.3: 21 February 2005. Many improvements; new and more flexible schedules.
- Version 0.4: 5 May 2005. "Prequel" phase.
- Version 1.0: 7 March 2007. Improved ease of user compilation.
- Version 2.0: 12 Jan 2009. Server default changed from "loopback" to "localhost" (Windows Vista compatibility and more general standardization).
- Version 2.1: 14 Apr 2015. Update for WhiskerClientLib v4.62 socket code.

1.2 Required devices

The program requires to claim devices in groups named **box0, box1, box2...** with device names as listed below in bold:

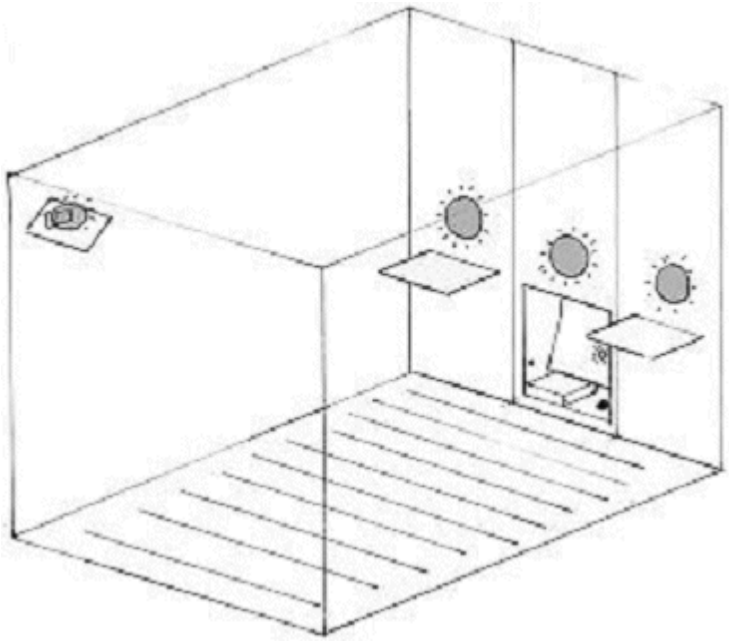
```
// Names of lines the program expects to be able to claim
NOSEPOKE           // input
LEFTLEVER          // input
RIGHTLEVER         // input
HOUSELIGHT         // output
PUMP               // output
DIPPER             // output
LEFTLEVERCONTROL   // output
RIGHTLEVERCONTROL  // output
LEFTLIGHT          // output
RIGHTLIGHT         // output
PELLET             // output

// Aliases used while the program is in full flight, which it therefore expects
```

```
not to be present on the server:
ActiveLever
InactiveLever
ActiveLeverControl
InactiveLeverControl
LeverLight
OppositeLight
```

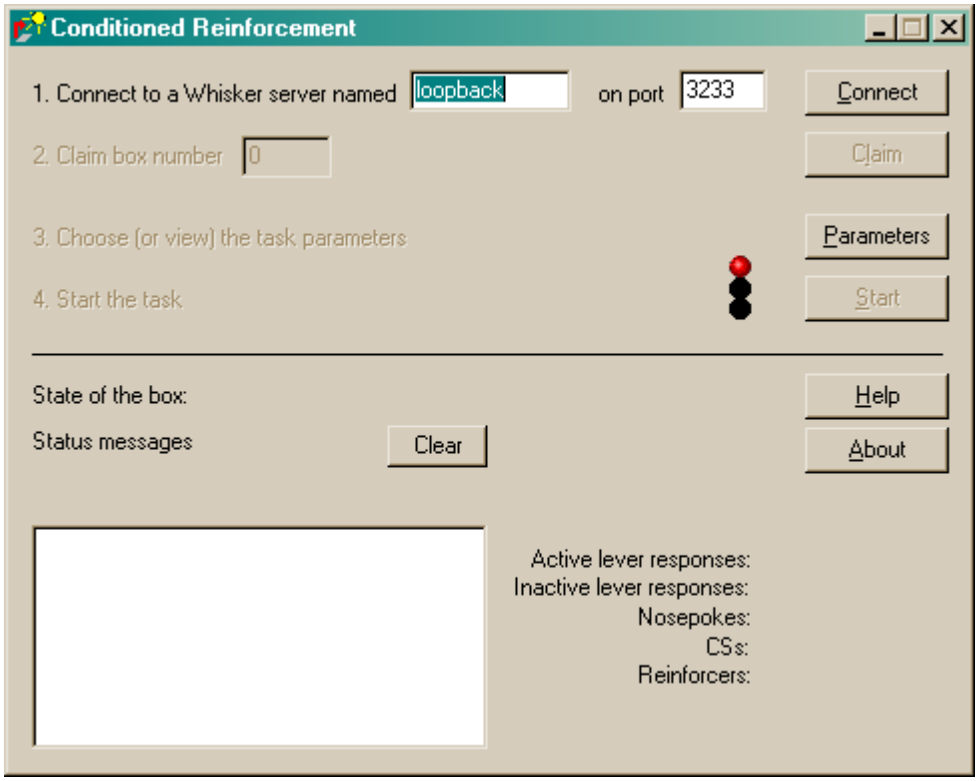
Please ensure that these devices are available and listed in the device definition file in use by the server.

This is the type of operant chamber we're talking about:



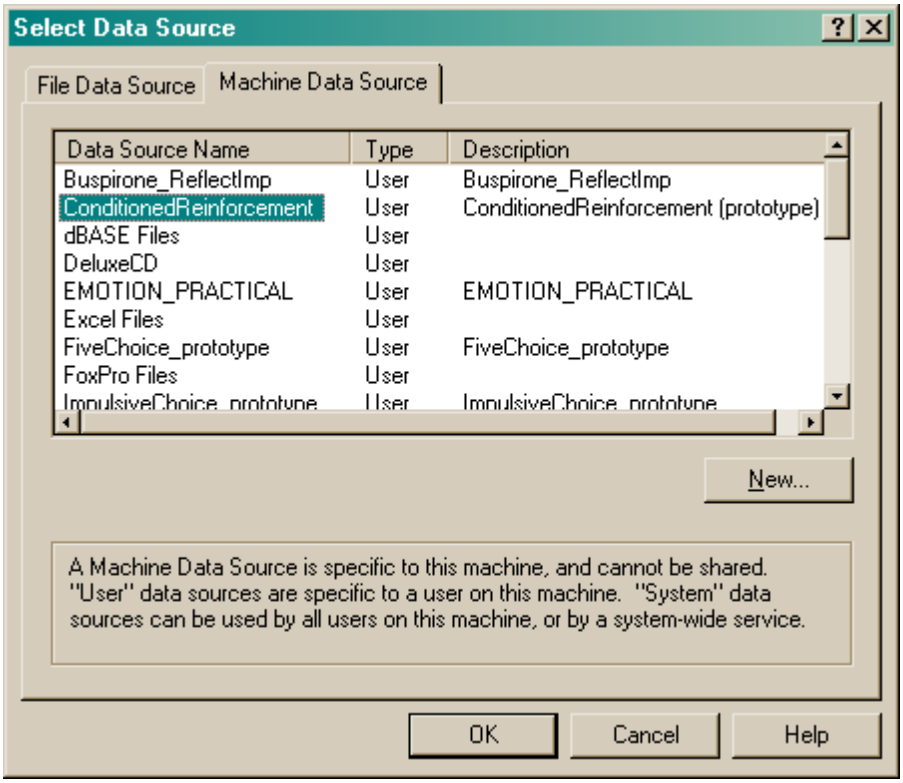
1.3 Using the task

When you run the task, the main screen looks as follows:



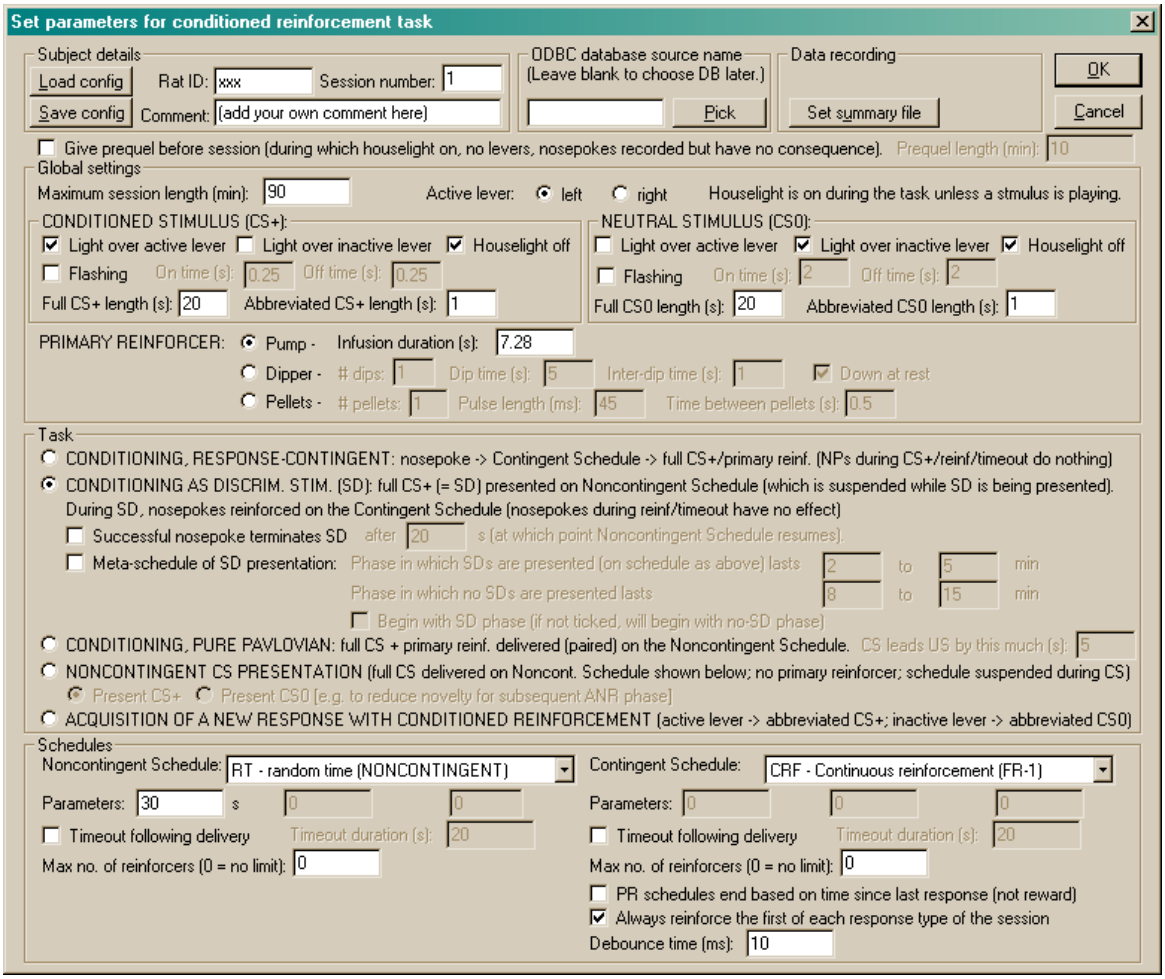
You must connect to a Whisker server, claim an operant chamber (box), and set up the parameters for your task. Once that's done, the traffic lights will turn amber. When you are ready, press *Start* to begin the task.

When the task finishes, it saves data to disk and pops up a new dialogue box for you to select a database to store the data to. (The data sources are configured under *Control Panel* → *ODBC*.) If you previously specified an ODBC data source in the parameters, that data source is used automatically and you will only see a dialogue box if something goes wrong and the program needs your input.



1.4 Parameters

The parameters dialogue box looks like this:



Particular things to note:

- Reinforcer limits (specified in the schedules) apply to whatever is being delivered by that schedule - so in the SD task, the "max no. of reinforcers" for the Noncontingent Schedule sets the total maximum number of SDs to be delivered.
- The "always reinforce the first of each response..." is *superimposed* upon the underlying schedule. So in an FR10 schedule, if this is ticked, then responses 1 and 10 and 20 will be reinforced (not 1 and 11 and 21).
- When a reinforcer limit has been reached, the task ends.
- In the ANR phase, any timeout (triggered by either lever) applies to both levers.
- In the ANR phase, responses during an ongoing stimulus are not reinforced.
- When the session time limit expires, any ongoing CSs are cut off, but the task waits for any ongoing primary reinforcement to finish.
- "Debouncing" applies especially to levers (though debouncing is applied to all onset/offset events from levers and nosepokes, if selected). Levers tend to bounce mechanically, generating spurious but rapid electrical signals. "Debouncing to 10 ms" means ignoring any response that occurs within 10 ms of a previous response (on the same device). 10 ms is behaviourally extremely short, yet electrically quite long, so it serves to discriminate true from false responses quite well.

Schedule selection

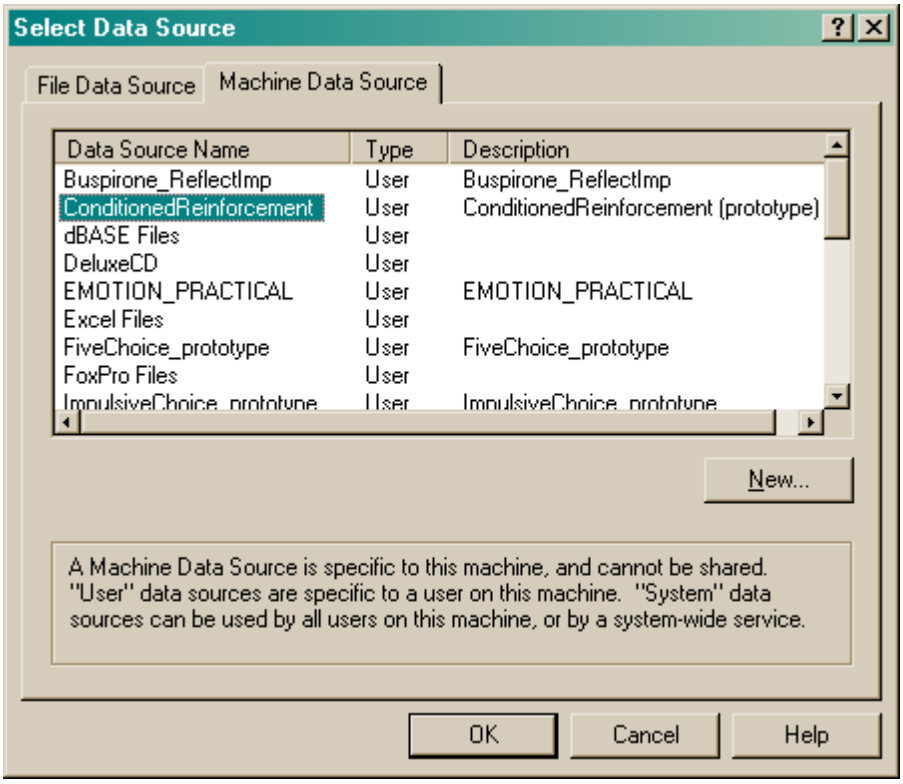
The schedules (both response-contingent and noncontingent) are:

- **CRF - continuous reinforcement (FR-1).** One reinforcer per response.
- **EXT - extinction.** No reinforcers.
- **FR *x* - fixed ratio.** One reinforcer per *x* responses.
- **VR *x* to *y* - variable ratio (specifying min, max).** After a variable number of responses (randomly chosen from *min* to *max* inclusive), one reinforcer is delivered.
- **RR *x* - random ratio.** P(reinforcer | response) = 1/*x*.
- **PROB *p* - probabilistic.** P(reinforcer | response) = *p*.
- **FI *x* - fixed interval.** The first response after *x* seconds is reinforced. The *first response* of the schedule is also reinforced.
- **RI *x* - random interval.** Reinforcement is set up on a random-time schedule (see below); after reinforcement has been set up, the next response is reinforced.
- **VI *x* to *y* - variable interval (specifying min, max).** After a variable time (from *min* to *max* seconds), the next response is reinforced.
- **FT *x* - fixed time (NONCONTINGENT).** No lever is present. Reinforcement is delivered every *x* seconds.
- **VT *x* to *y* - variable time (specifying min, max) (NONCONTINGENT).** No lever is present. The schedule waits for between *min* and *max* seconds, then delivers a reinforcer, then repeats.
- **RT *x* - random time (NONCONTINGENT).** Every second, p(reinforcer delivered this second) = 1/*x*. Thus, on average, reinforcement is delivered once every *x* seconds, but the subject cannot predict the likelihood of reinforcement based on how long it has waited (unlike a typical VT schedule).
- **PR - progressive ratio - add one (1,2,3,4...)** - progressive ratio schedule, adding one to the ratio requirement at each step. The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
- **PR - progressive ratio - double (1,2,4,8...) -** progressive ratio schedule, doubling the ratio requirement at each step. The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
- **PR - progressive ratio - Fibonacci (1,1,2,3,5...) -** progressive ratio schedule with a Fibonacci progression. The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
- **PR - progressive ratio - Roberts exponential (A * exp(reinfnum * B) - A) -** progressive ratio schedule with an exponential progression, based on Roberts DCS & Richardson NR (1992), Self-administration of psychomotor stimulants using progressive ratio schedules of reinforcement, *Neuromethods* 24: 233-269 (eds Boulton A, Baker G, Wu PH; Humana Press). The ratio requirement is (A * exp(reinforcer number * B)) - A, rounded to the nearest integer. Typically, A is 5. A typical schedule might have B=0.2; these values yield ratio requirements {1, 2, 4, 6, 9, 12, 15, 20, 25, 32, 40, 50, 62, 77, 95, 118, 145, 178, 219, 268, 328, 402, 492, 603, 737, 901, 1102, 1347, ...}. A steeper PR schedule is obtained with B=0.25, giving {1, 3, 6, 9, 12, 17, 24, 32, 42, 56, 73, 95, 124, 161, 208, 268, 346, 445, 573, 737, 948, 1218, 1566, 2012, 2585, 3321, 4265, 5478, ...} The schedule termination is determined by the other parameter (on the left, labelled (min)); if this parameter is >0, then when this many minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
- **DELAYFR1 - FR1 with delayed reinforcement.** This is an FR1 schedule, but there is a delay between responding and reinforcement. This delay is the sole parameter (specified in seconds).
- **PR - progressive ratio - double increment every A reinforcers.** The increment starts at 1, and doubles every A reinforcers. If A is 8, then the ratio requirements are 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 18, 20, 22, 24, 28, 32, 36... The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.

Special case: the first response on contingent interval schedules (FI, RI, VI) is always reinforced.

Database selection

To pick an ODBC database **in advance** of finishing, click *Pick* and you will be offered the ODBC Data Source picker (below). Your choice will be recorded and will apply to this subject from now on (or until you specify a different source).



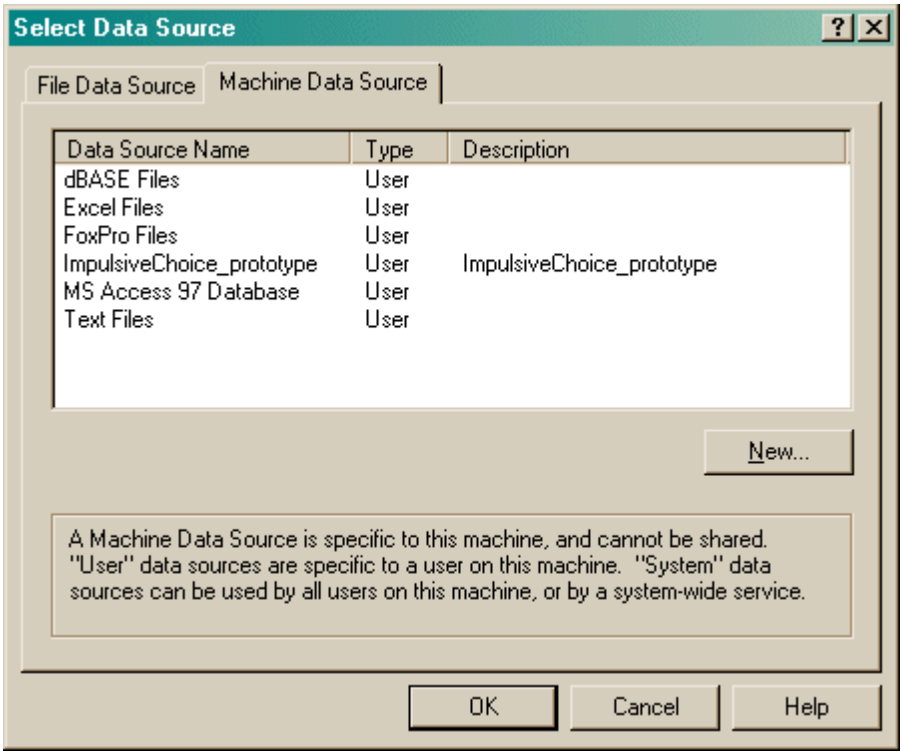
If you don't specify an ODBC data source now, or you delete the value in the "ODBC data source name" box, you'll be asked to choose when the task ends (and that choice will only apply to the session in progress).

What happens if you can't find an appropriate ODBC source?

1.5 Creating a new ODBC source

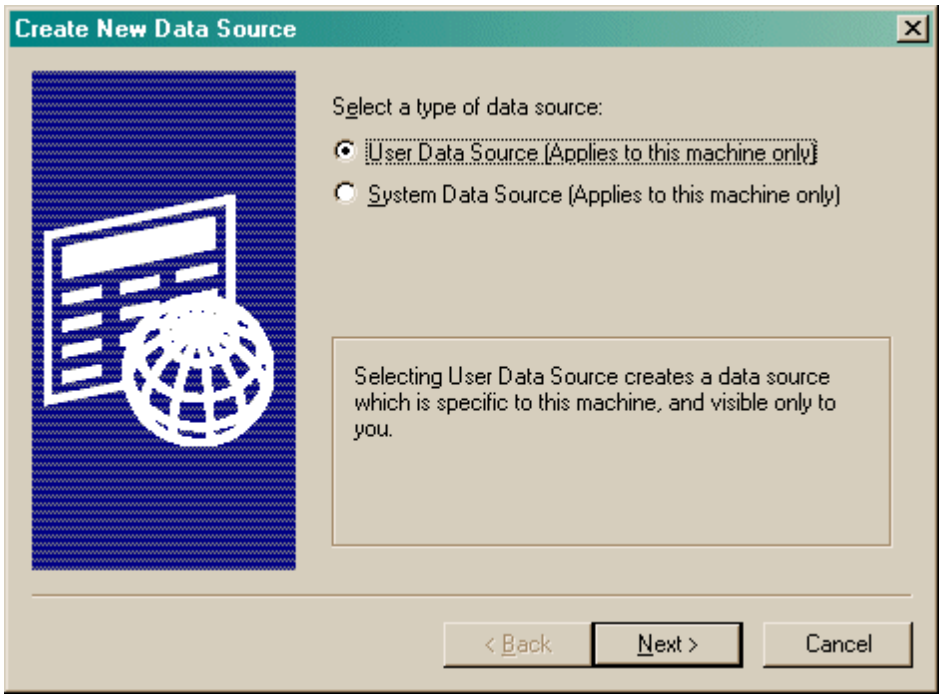
What happens if you can't find an ODBC source? You should configure it via *Control Panel → ODBC* . Alternatively, you can set one up "on the fly", as explained here.

Suppose you're looking for a PIT database. But there isn't one...

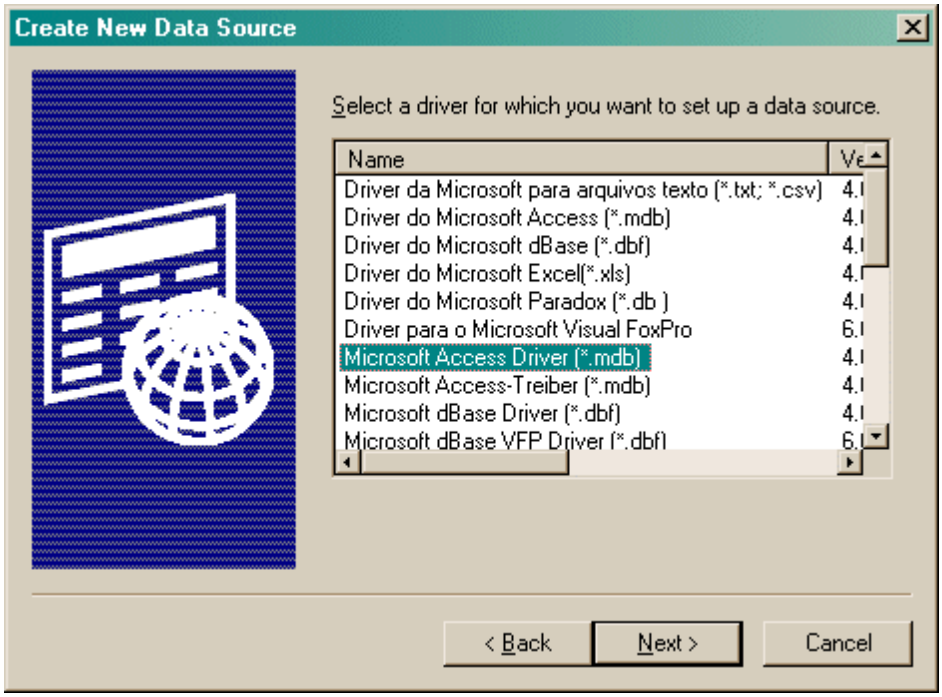


Let's assume that you have **already made a working copy of the prototype database supplied with the task.** How do we go about setting this up as an ODBC data source?

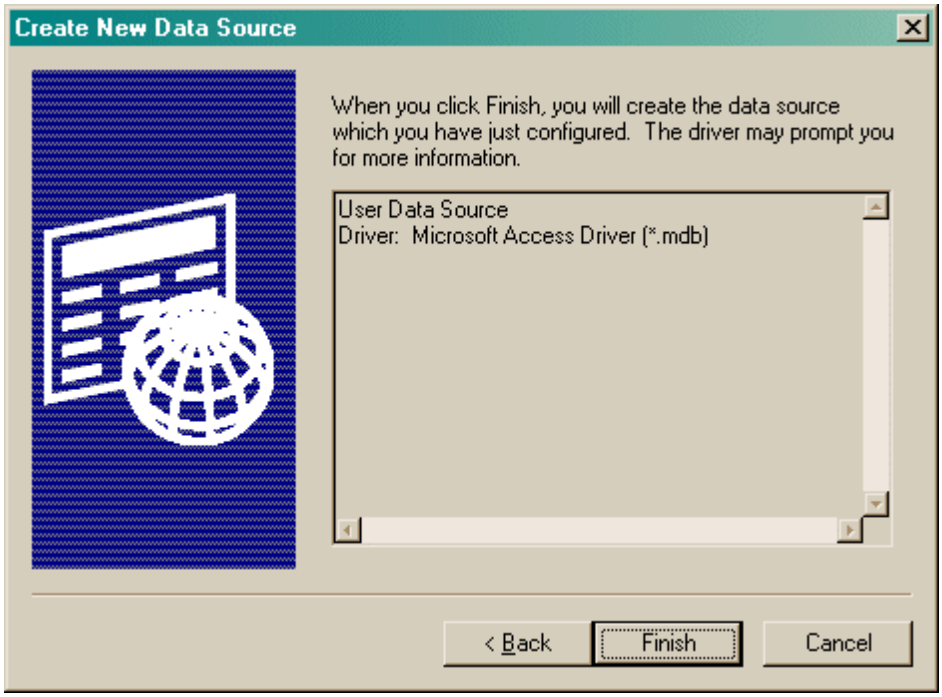
Click New.



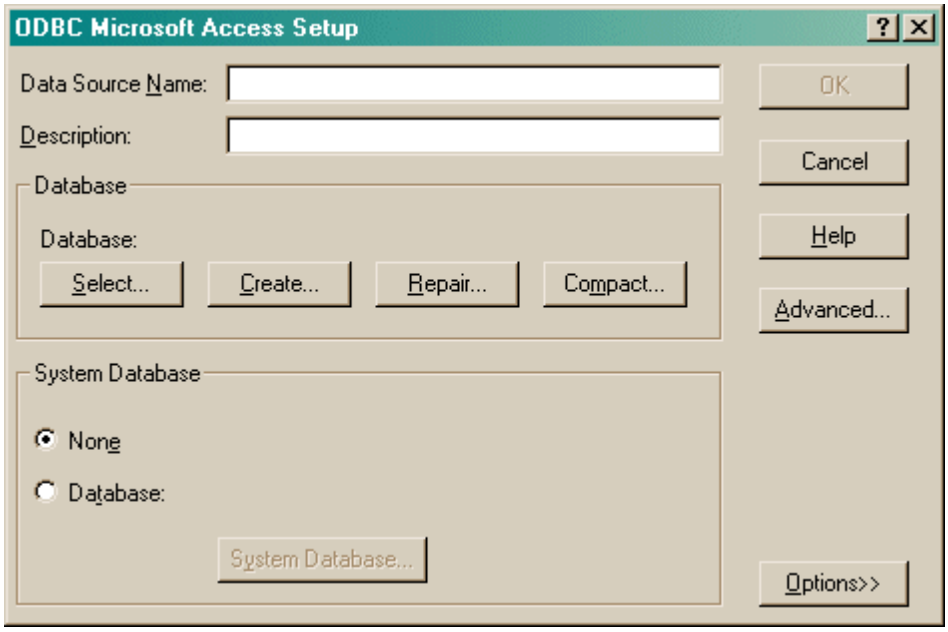
Choose a User or System data source. **User** is probably more sensible. Click Next.



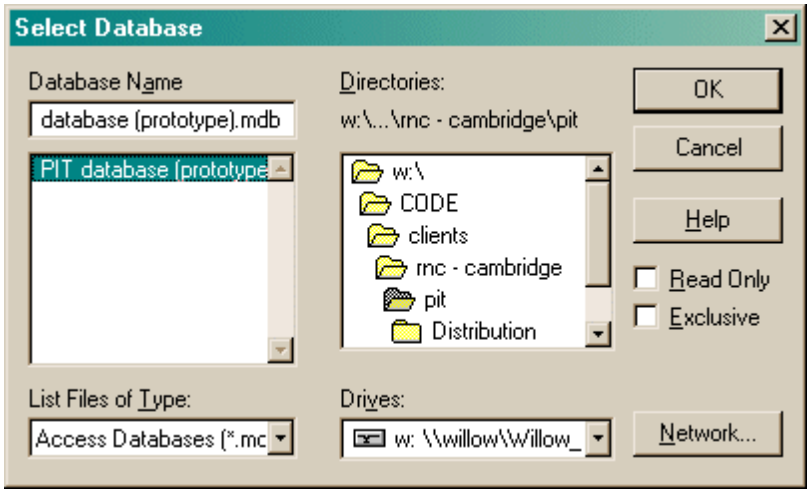
Choose your database driver. Click Next.



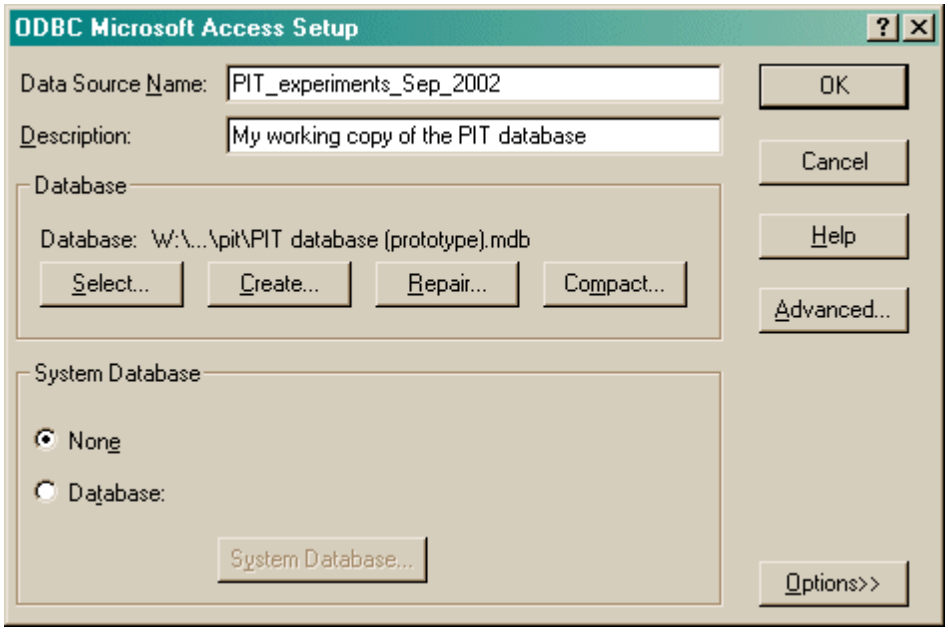
Click Finish.



You should fill in the **Data Source Name (no spaces)** and the **description,** and **Select** a database. When you click Select, this dialogue box appears:



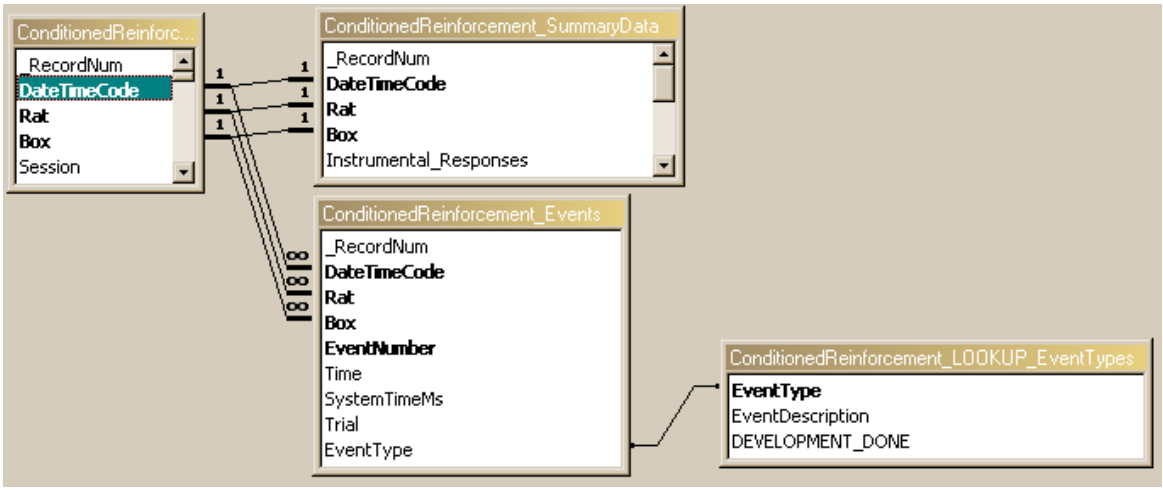
Choose your database here and click OK. Your ODBC data source fields should now all be set up:



Click OK. You will be returned to the ODBC selection screen with your new data source now available.

1.6 Database structure

This is the structure of the ConditionedReinforcement database:



Index

- C -

ConditionedReinforcement
about 2
database structure 12
parameters 6
required devices 2
using 4

- O -

ODBC source
creating a new 8