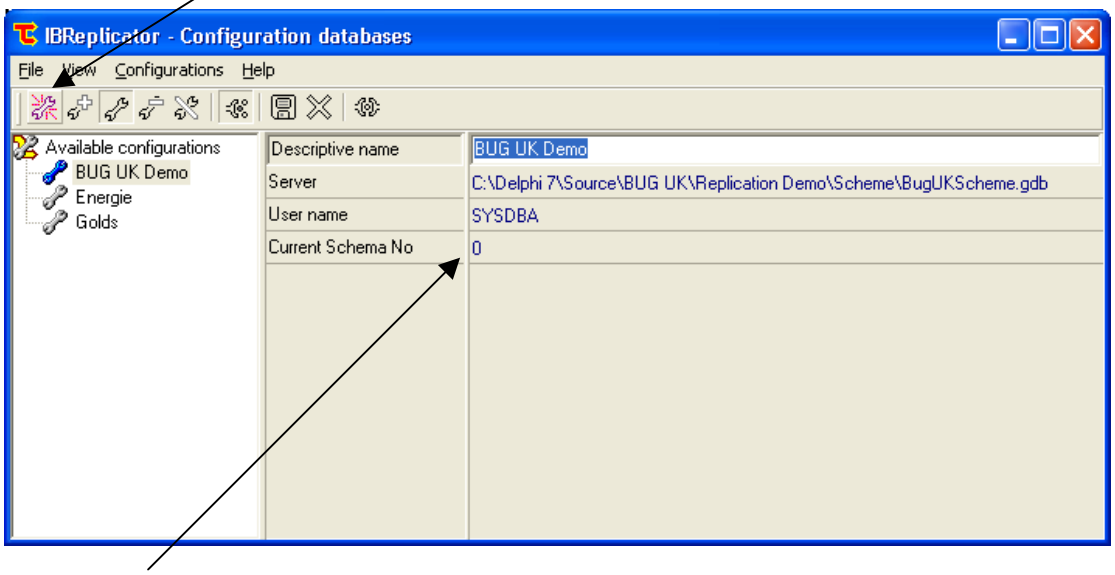
IBReplicator an Introduction

Notes to the talk.

- The talk covered the use of replication, terms and what it can do.
- To give a real world feel, the talk looked at case study of a Fitness Club replicating with a head office or other sites in the chain.

Open Replication Manager : File >> Configuration.

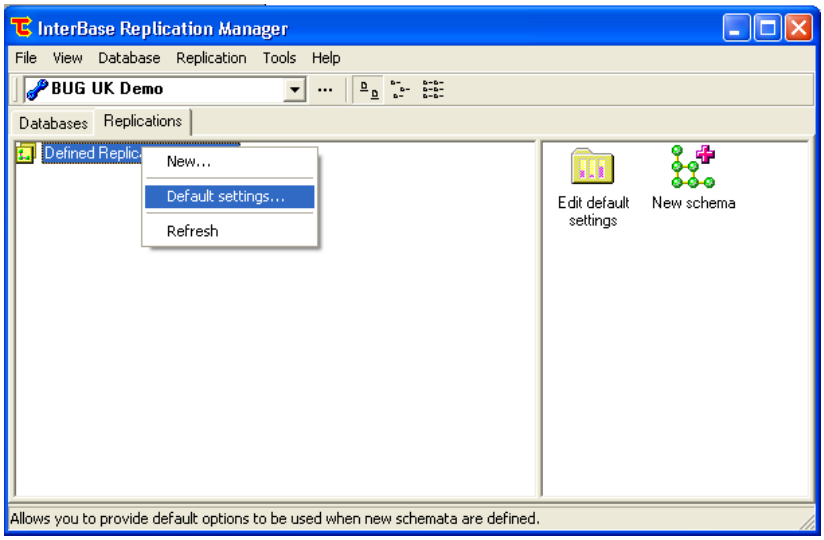
Start by creating a Scheme.

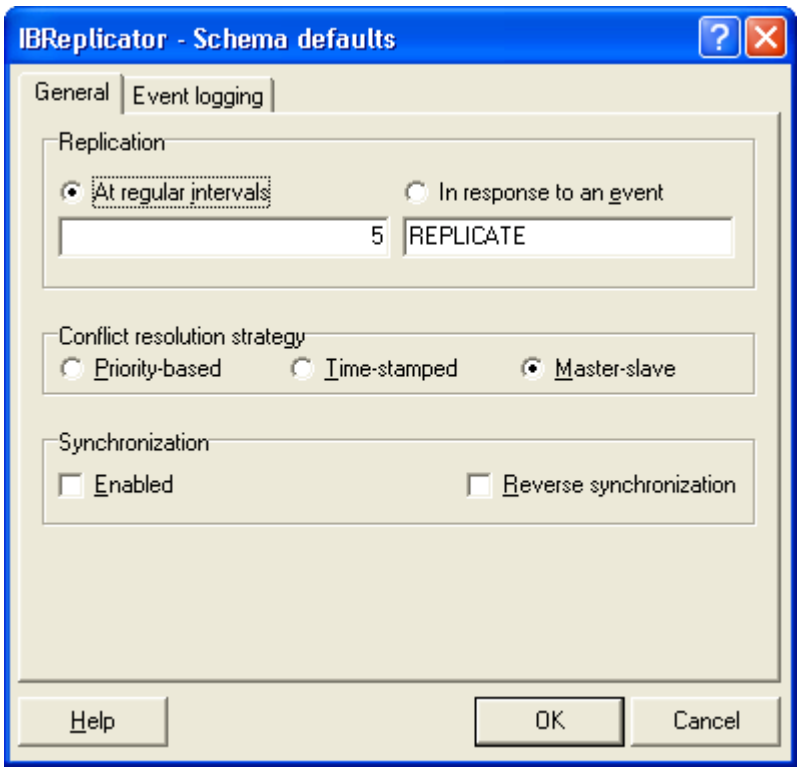


Schema No: This is the running internal number, each scheme has a unique number to enable multiple schemes to run in one database (eg in a head office solution with multiple site) This increments as we build the scheme.

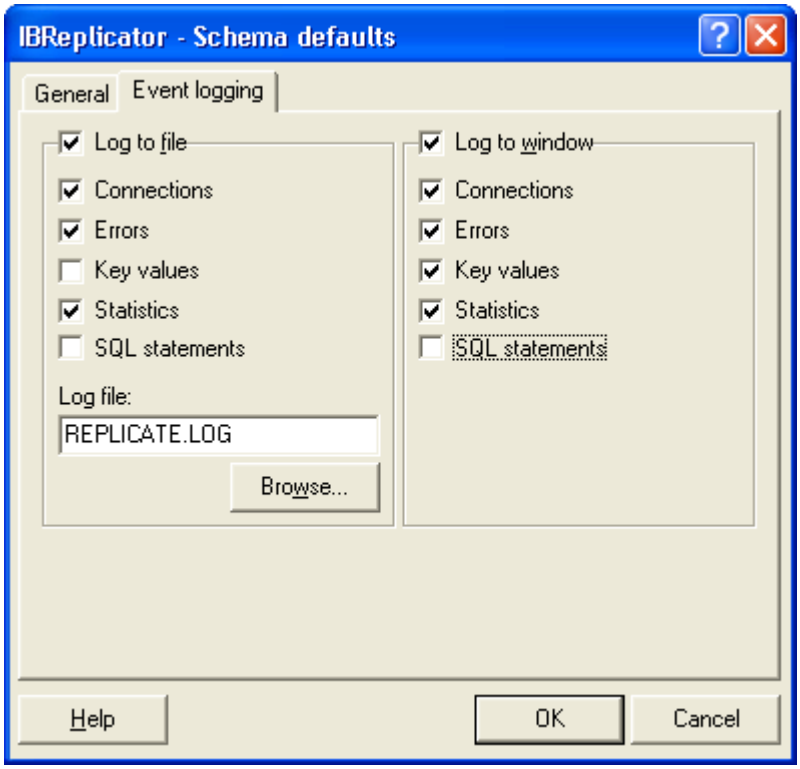
IT IS VERY IMPORTANT THIS DOES NOT OVER LAP EVER.
If you have multiple schemes set up that run on the same database it is a good idea to separate them but 100 or so for each Scheme (ie 1st Scheme starts from 1, 2nd from 101, 3rd from 201 and so on)

Set up Defaults (as other wise you will spend time doing it and checking it later.





Recommend Master-Slave for our set-up Time-Stamp leads to errors when closing. Priority-based (never used) but is setup on the database tab.



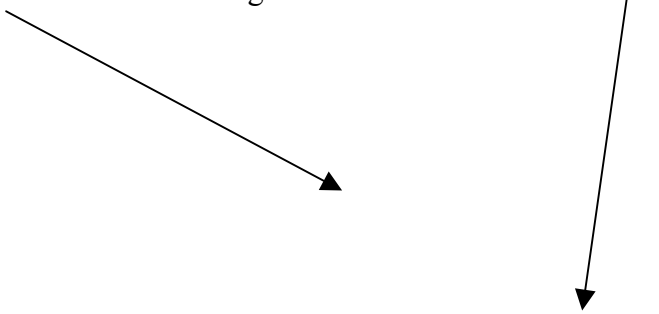
Log connection info, any errors and stats, the rest you only need when debugging. Log Key values to see which records have been replicated. (but it explodes the log file) and there is no auto limitation on size.

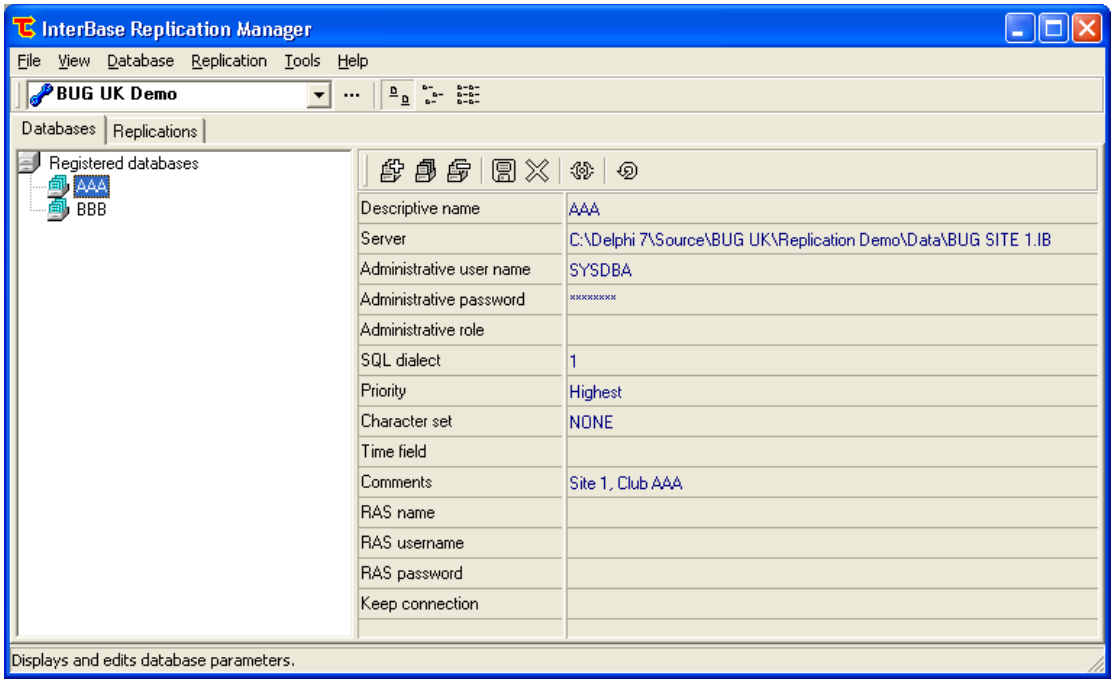
Set-up Databases.

User names and passwords are stored encrypted in the scheme database. This is needed to create system objects. They need to be the Admin user name password (ie that you use to connect to and manage the database)

(Hint…. Ctrl+C does not work, but Ctrl+X ,Ctrl+V does use this if you need to copy the user name from one to the other.)

Add the source and target database into the Scheme





To learn, we are going to use a scenario.
In this scenario we are running a members club. One peace of good practice to help replicate is to identify who the records belong to. To do this we will have club codes on the database. Here is the example database. As we are using the “Club Code” on each table I’ve set it up as a domain. We are also using a unique way of making a key for each record (I’ll cover this later)

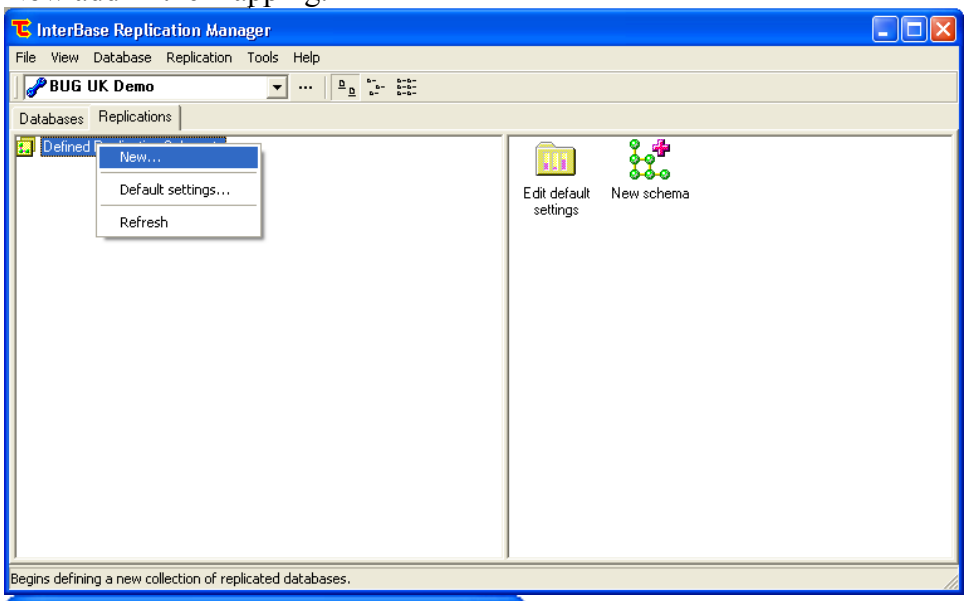
```
SET SQL DIALECT 1;

/* CREATE DATABASE 'C:\Delphi 7\Source\BUG UK\Replication
Demo\Data\BUG Site 1.ib' PAGE_SIZE 4096

 DEFAULT CHARACTER SET  */
/* Domain definitions */
CREATE DOMAIN DOMAINCLUBCODE AS VARCHAR(3) NOT NULL;
CREATE DOMAIN DOMAINUID AS VARCHAR(20) NOT NULL;

/* Table: ATTENDANCE, Owner: SYSDBA */

CREATE TABLE ATTENDANCE
(
  UID   DOMAINUID,
  MEMBER_UID       DOMAINUID,
  NAME         VARCHAR(40),
  VISIT_DATE_TIME          TIMESTAMP,
  CLUB_CODE         DOMAINCLUBCODE,
 PRIMARY KEY (UID)
);
```

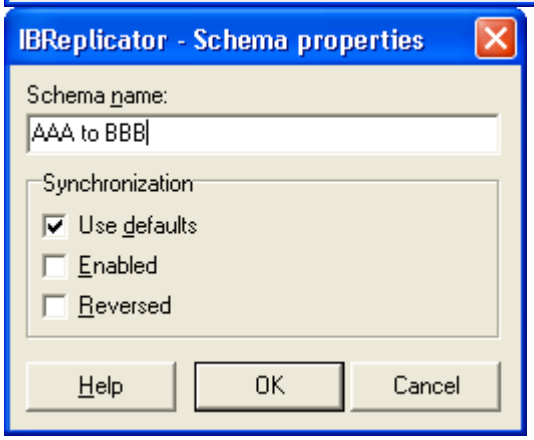
```
/* Table: MEMBERS, Owner: SYSDBA */

CREATE TABLE MEMBERS
(
  UID   DOMAINUID,
  CLUB_CODE         DOMAINCLUBCODE,
  NAME         VARCHAR(40),
  MEMBERSHIP_TYPE       VARCHAR(15),
  LAST_VISIT          TIMESTAMP,
  ADDRESS_LINE_1 VARCHAR(30),
  ADDRESS_LINE_2 VARCHAR(30),
  CITY_STATE_ZIP  VARCHAR(50),
 PRIMARY KEY (UID)
);
```

In this example there is 2 sites. AAA and BBB.
The relationship between these sites will change as the demo progresses.

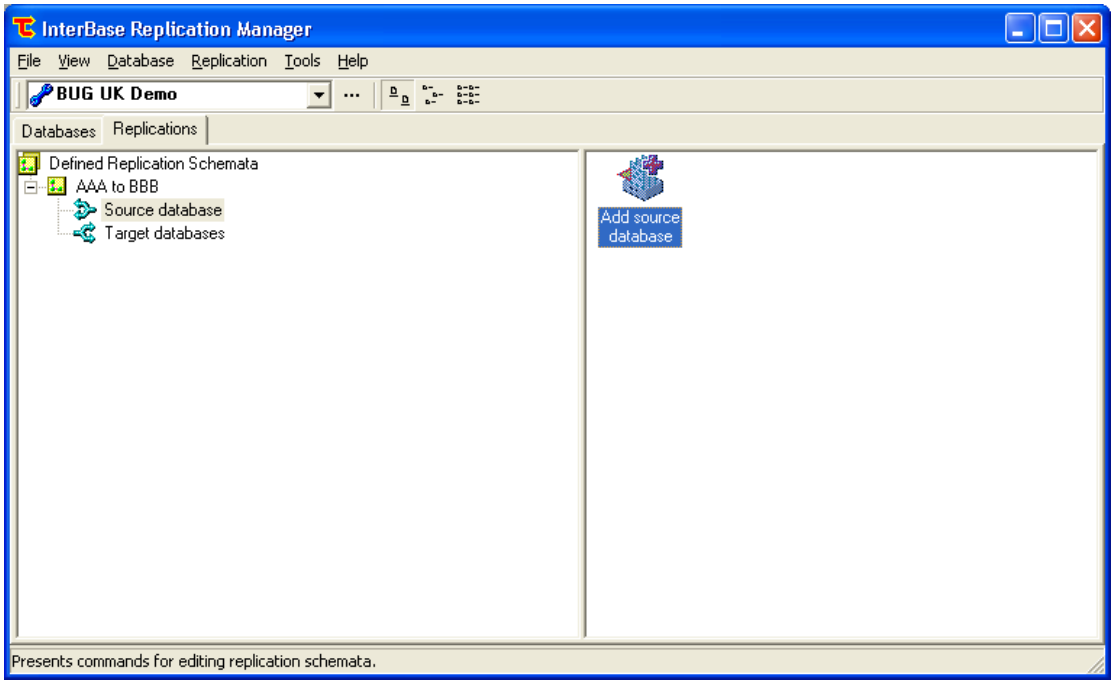
For now they are Pear-to-Pear clubs. Each club manages their own members and members are replicated to each other's club so they can attend either club.

Now add in the mapping.

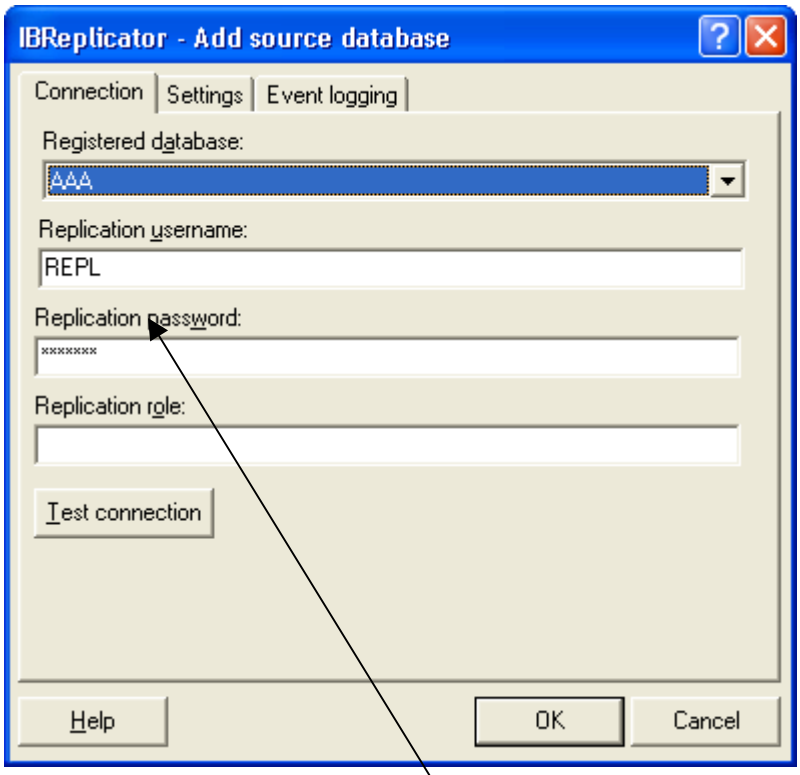




Simple names really help here. Lets start with moving changes from site AAA to site BBB



Now Source database is the source of the changes, target is where you want to move the changes too. In this example the as we are moving from AAA to BBB, AAA is the source. And BBB is the target



As we have already set the defaults, they will be set for us on the settings and event logging tabs. (so no lovely picture now)

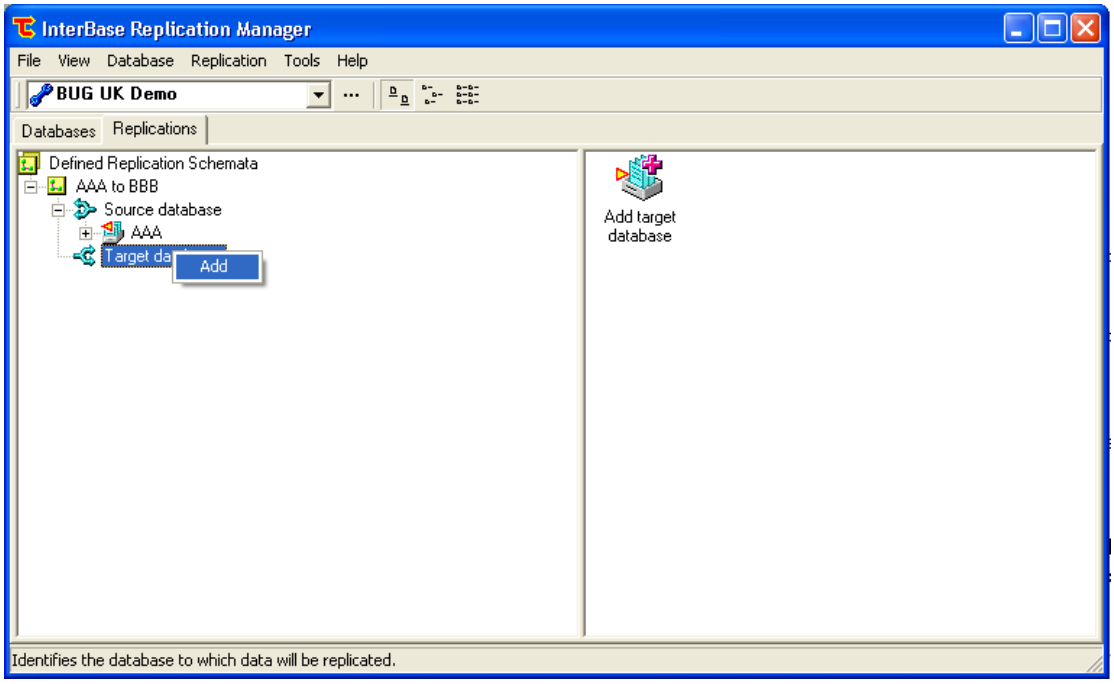
We have selected AAA as the source, however the replication user name and password are not the admin names here. This is a common pitfall.

When we make a change in the database it gets logged to move.
A little later the replicator connects to the source and moves it to the target database. However when the replicator makes a change in the target, how do we make sure that it does not bounce back so we avoid an ever running circle of changes?

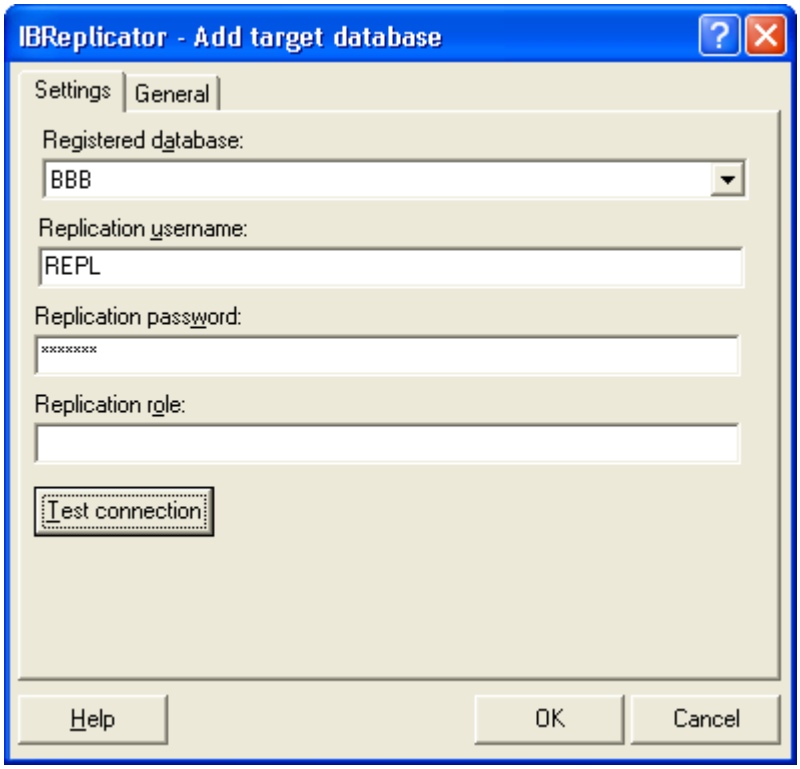
The answer to the question and this common pit fall is to use the REPL user. This is a special user that you need to add to Interbase and both ends, with a password of your choice. When the user REPL makes a change, the insert/update/delete is not logged to replicate back.

This is essential on the target database, however is good practice on the source database as well.

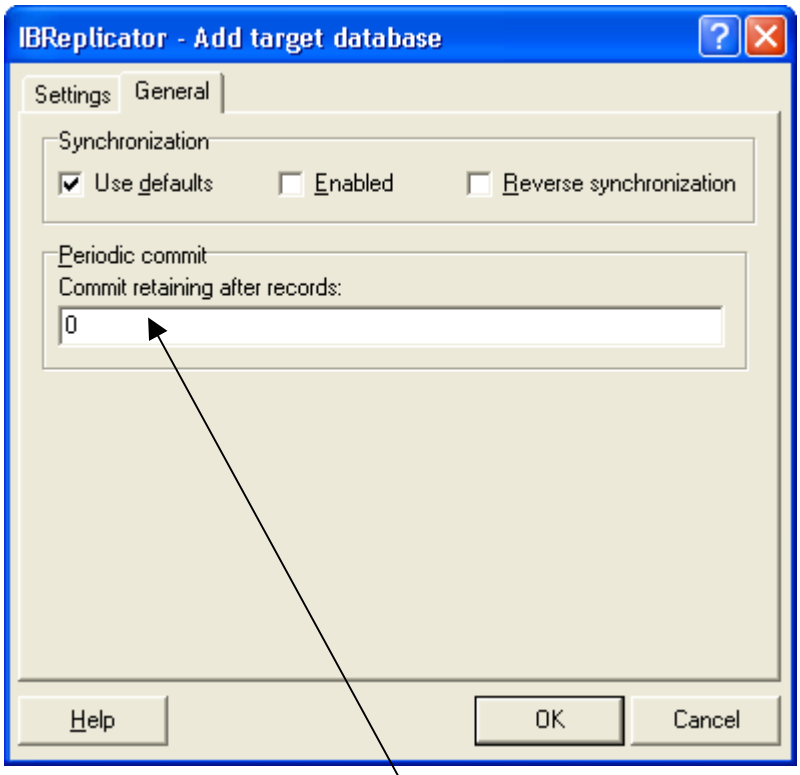
.



So lets add the target and make sure we use the correct user name here.



Test connection only means you can connect to the database, it doesn't mean you have rights to do anything on the database!!! It's a good way to confirm you have the user name and password correct, but don't forget to grant REPL rights to everything.

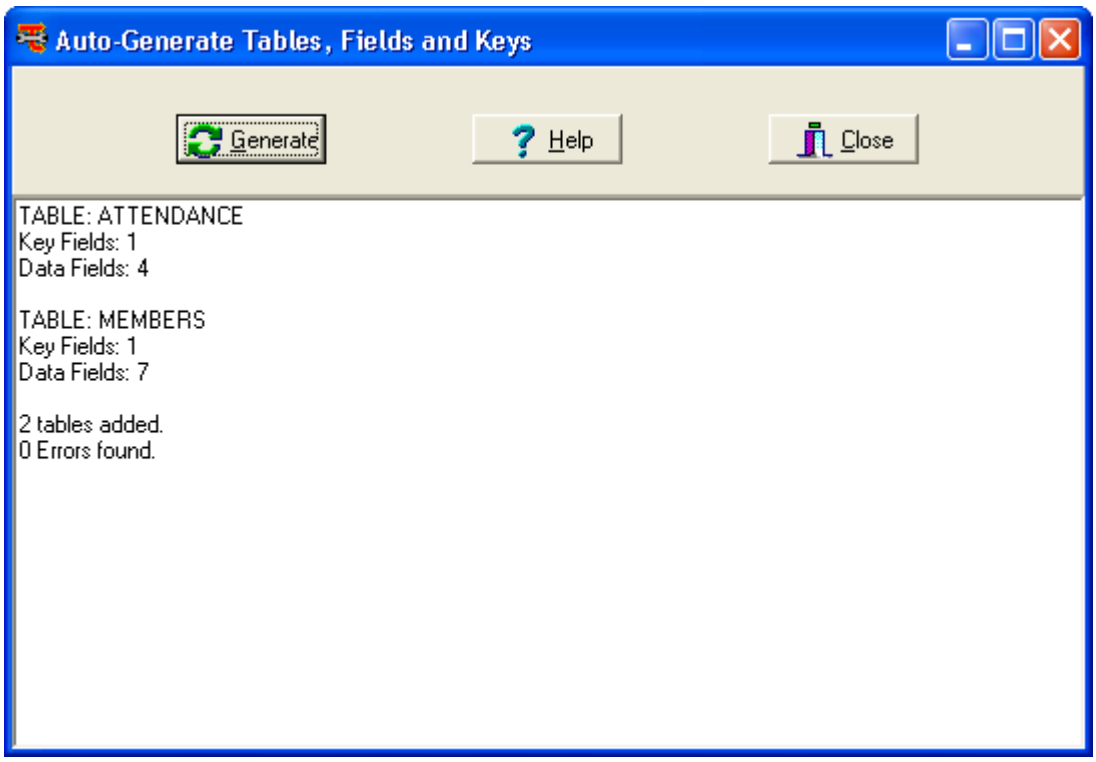


If you use dial up or have "not so stable" connections (be it due to routers or what ever). It is a good idea to use the periodic commit. That way, once the x number of changes has been replicated, the replicator completes the 2-stage commit process and

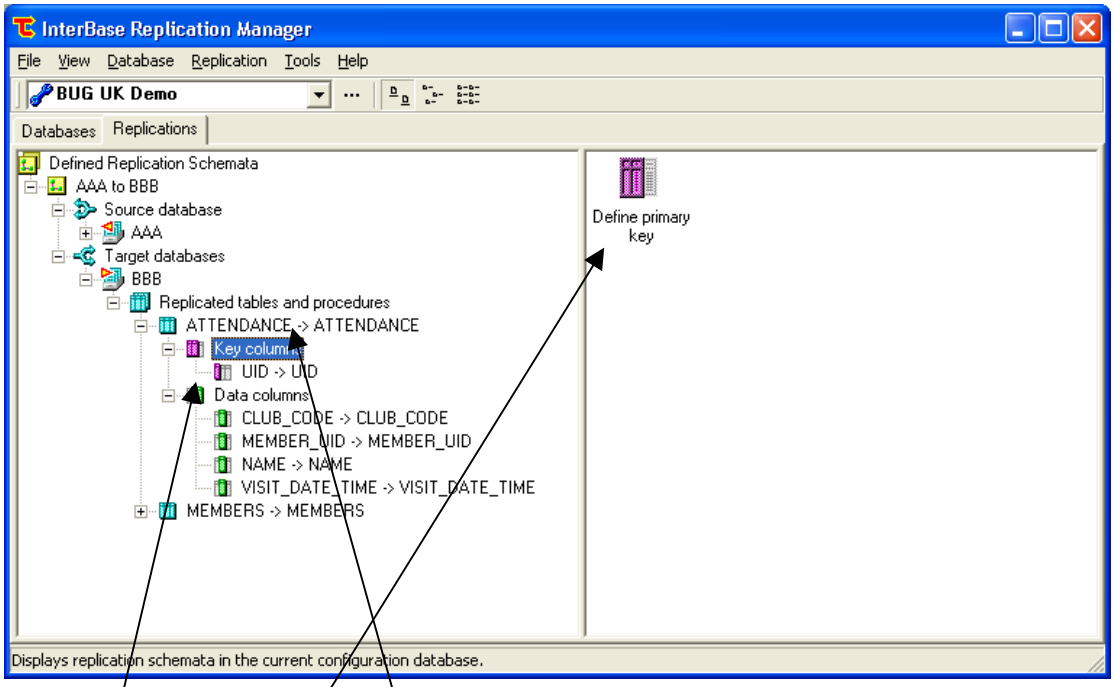
continues on. This way if the link goes down, a limited amount of work needs to be redone. Past experience has set a guide of 500 for dial up (56k analogue) and 1000 for an ISDN (64k) line or above.

Now we have the source and target, we have to say what to move from where to where.

There is a really nice option here to auto “Generate” the mappings now. However make sure your primary keys are defined in the tables, if not your need to go back and map them manually before you can create system objects later (what?? System objects. I’ll get to that in a minute, but it’s important - in short its what the replictor uses the admin db password for, to alter the database by adding tables and triggers to record the changes that happen so it knows to move them)



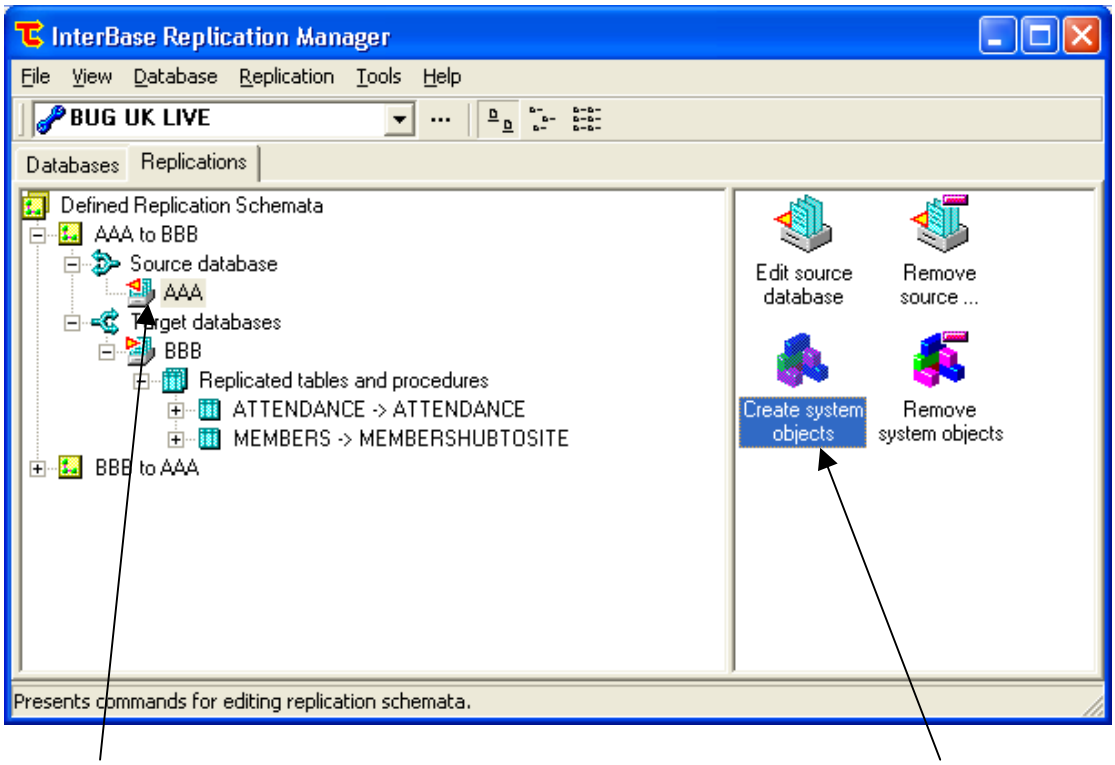
Afterwards have a look at the target database mapping



It shows the source db field on the left and the target field on the right.
It shows we replicate from AAA.ATTENDANCE to BBB.ATTENDANCE and AAA.UID is moved to target BBB.UID as the primary key (pink) and so on with the data columns. (in green)

To manually change the mappings here select Key or Data Columns and use the define option

Now create the system objects and replicate.



To create system objects simply select the source database and choose Create system objects

Have a play and see how data is moving.

You can also try adding the stored procedures to the databases and mapping from Members >> MembersHubToSite and back the other way from Members >> MembersSiteToHub. We use the Table name and then the direction in the demo as we are emulating a Head office (HUB) and a number of clubs/sites in the chain.

Eg Stored procedures…..

```
set term ^ ;
CREATE PROCEDURE MEMBERSHUBtoSITE (
  ADDRESS_LINE_1 CHAR(30),
  ADDRESS_LINE_2 CHAR(30),
  CITY_STATE_ZIP CHAR(50),
  CLUB_CODE CHAR(3),
  LAST_VISIT TIMESTAMP,
  MEMBERSHIP_TYPE CHAR(15),
  NAME CHAR(40),
  UID CHAR(20),
  THE_TYPE CHAR(1) )
  RETURNS
 (RESULT INTEGER) AS
  declare variable COUNTER INTEGER;

  BEGIN
  RESULT=0;
  select count(*) from MEMBERS where UID = :UID
  into counter;
  if (THE_TYPE = 'I' and COUNTER > 0) then
  begin
    RESULT = 1;
    exit;
  end
  if (THE_TYPE = 'U' and counter = 0) then
  begin
    RESULT = 1;
    exit;
  end
  if (THE_TYPE = 'D' and counter = 0) then
  begin
    RESULT = 1;
    exit;
  end
if (THE_TYPE = 'I') then
 begin
  insert into MEMBERS (UID,
  ADDRESS_LINE_1,
  ADDRESS_LINE_2,
  CITY_STATE_ZIP,
  CLUB_CODE,
  LAST_VISIT,
  MEMBERSHIP_TYPE,
  NAME
 ) Values (
  :UID,
  :ADDRESS_LINE_1,
  :ADDRESS_LINE_2,
  :CITY_STATE_ZIP,
  :CLUB_CODE,
  :LAST_VISIT,
  :MEMBERSHIP_TYPE,
  :NAME
);
  exit;
 end
if (THE_TYPE = 'U') then
```

```
 begin
  update MEMBERS set
    ADDRESS_LINE_1 = :ADDRESS_LINE_1,
    ADDRESS_LINE_2 = :ADDRESS_LINE_2,
    CITY_STATE_ZIP = :CITY_STATE_ZIP,
    CLUB_CODE = :CLUB_CODE,
    MEMBERSHIP_TYPE = :MEMBERSHIP_TYPE,
    NAME = :NAME
   where UID = :UID;
  exit;
 end
if (THE_TYPE = 'D') then
 begin
   Delete from MEMBERS where UID = :UID;
 exit;
end
END
^
 GRANT EXECUTE ON PROCEDURE MEMBERSHUBtoSITE TO REPL;
^
SET TERM ^ ;
```

```
set term ^ ;
CREATE PROCEDURE MEMBERSSITEtoHUB (
  ADDRESS_LINE_1 CHAR(30),
  ADDRESS_LINE_2 CHAR(30),
  CITY_STATE_ZIP CHAR(50),
  CLUB_CODE CHAR(3),
  LAST_VISIT TIMESTAMP,
  MEMBERSHIP_TYPE CHAR(15),
  NAME CHAR(40),
  UID CHAR(20),
  THE_TYPE CHAR(1) )
  RETURNS
 (RESULT INTEGER) AS
  declare variable COUNTER INTEGER;

  BEGIN
  RESULT=0;
  select count(*) from MEMBERS where UID = :UID
  into counter;
  if (THE_TYPE = 'I' and COUNTER > 0) then
  begin
    RESULT = 1;
    exit;
  end
  if (THE_TYPE = 'U' and counter = 0) then
  begin
    RESULT = 1;
    exit;
  end
  if (THE_TYPE = 'D' and counter = 0) then
  begin
    RESULT = 1;
    exit;
  end
if (THE_TYPE = 'I') then
 begin
  insert into MEMBERS (UID,
  ADDRESS_LINE_1,
  ADDRESS_LINE_2,
  CITY_STATE_ZIP,
  CLUB_CODE,
  LAST_VISIT,
  MEMBERSHIP_TYPE,
  NAME
 ) Values (
  :UID,
  :ADDRESS_LINE_1,
  :ADDRESS_LINE_2,
  :CITY_STATE_ZIP,
  :CLUB_CODE,
  :LAST_VISIT,
  :MEMBERSHIP_TYPE,
  :NAME
```

```
);
  exit;
 end
if (THE_TYPE = 'U') then
 begin
  update MEMBERS set
    LAST_VISIT = :LAST_VISIT
   where UID = :UID;
  exit;
 end
if (THE_TYPE = 'D') then
 begin
   Delete from MEMBERS where UID = :UID;
 exit;
end
END
^
 GRANT EXECUTE ON PROCEDURE MEMBERSSITEtoHUB TO REPL;
^
SET TERM ^ ;
```

EOD

mailto: Steve@Designer-software.net 2004-07-18

Speaker notes from BUG UK meeting - July 2004