GREAT LAKES FISHERY COMMISSION

1995 Project Completion Report[1]

Integrated Management of Sea Lamprey Database Review

by:

Miroslaw Kuc and Lorne Greig
ESSA Technologies Ltd.
#308, 9555 Yonge Street
Richmond Hill, ON L4C 9M5

March 1995

[1]Project completion reports of Commission-sponsored research are made available to the Commission's Cooperators in the interest of rapid dissemination of information that may be useful in Great Lakes fishery management, research, or administration. The reader should be aware that project completion reports have not been through a peer review process and that sponsorship of the project by the Commission does not necessarily imply that the findings or conclusions are endorsed by the Commission.

Integrated Management of Sea Lamprey Database Review

Prepared for:

The Great Lakes Fisheries Commission

Prepared by:

Miroslaw Kuc and Lorne Greig
ESSA Technologies Ltd.
#308, 9555 Yonge Street
Richmond Hill, ON L4C 9M5

March 31, 1995

DRAFT

Integrated Management of Sea Lamprey Database Review

Prepared for:

The Great Lakes Fisheries Commission

Prepared by:

Miroslaw Kuc and Lorne Greig
ESSA Technologies Ltd.
#308, 9555 Yonge Street
Richmond Hill, ON L4C 9M5

March 31, 1995

Table of Contents

1.0 Introduction & Objectives

This brief report documents the findings and recommendations of a review of the IMSL/LCSS database structure. The review was undertaken as a part of the second phase of development of the IMSL/LCSS during February 1995.

The objectives of the review were to:

1) identify inconsistencies, if any, in database naming and data element definitions across the set of tables currently in the IMSL/LCSS database;
2) identify opportunities for simplification of the IMSL/LCSS database structure that would increase the ease of database maintenance and/or accelerate performance during LCSS simulations; and
3) identify strategic options for incorporation of design features for the LCSS which are yet to be implemented (e.g. implementation of multi-site consolidation and security).

Consistent with the second objective, several changes have been identified which should lead to improvement in overall database performance. It should be noted however, that the review does not represent a thorough and exhaustive effort to optimize performance of the LCSS as this would entail additional effort especially with respect to the coding and data retrieval/storage strategies employed within the LCSS software.

2.0 Overview of Existing DB Structures

The review of existing structures was guided by the objective of identifying opportunities for improving consistency and/or simplification of the database. In many respects, the objective of simplifying the database structures arose not from specific technical design concerns but rather from the early difficulties which LCSS users were experiencing in working with some of the concepts employed within the design. In particular, three areas were targeted as foci for review:

1) the concept of reach sets (i.e. alternative definitions of the physical structure {sets of reaches} of an individual stream);
2) barrier efficiency hypotheses (closely related to reach sets); and
3) lamprey basins.

Each of these three components of the database design deal with the representation of spatial structures within the IMSL/LCSS database. They are central to the overall utility of the system and their implementation is moderately to highly complex with regard to the relationships between database tables.

In addition, the database structures related to budget groups and reporting basins were of special concern. These features are only partially implemented in the current system and there are outstanding issues related to their definition and utility within the overall system.

2.1 Reach Sets

The primary focus of the review of the implementation of reach sets was to determine whether the structure of the database could be re-engineered so as to dispense with the concept and terminology of 'reach sets' or whether an alternative implementation could buffer general users of the system from having to work with the reach set concept. A clear understanding of the concept and rationale for our recommendations requires at least an overview of the current implementation of the reach set concept.

The basic function of the reach set concept within the IMSL/LCSS database is to provide a mechanism for representing scenarios which would explore proposed (theoretical) barrier dam placements for lamprey control. When barriers are placed on a stream, their intended purpose is to restrict access to spawning and potentially rearing habitat. Barrier placement therefore potentially influences the number of reaches in which it is necessary to simulate lamprey populations within the stream (multi-reach streams only), habitat characteristics of the 'reach' where the barrier is placed, and the selection/definition of chemical, trap and SMRT options for treatment of any residual lamprey populations. The related concept of barrier efficiency hypotheses deals with the efficiency of a barrier in blocking upstream passage of lamprey.

The interdependence of the various tables in the database can best be described by presenting the modifications involved in adding a reach set to the present database structure. Some of the tasks presented here can be accomplished using LCSS, but the changes have been described for the sake

of completeness. While reviewing this description you should fold out the chart entitled "Figure 1: Reach Set Links" in Appendix B to help you follow the discussion of changes referred to below.

Reaches represent distinct areas of ammocete habitat to be simulated. These may or may not represent the configuration found in a stream. In most cases it is not necessary to simulate all of the distinct areas of ammocete habitat. If a stream is always referred to as a single unit, all of the areas can be dealt with together. It is important to distinguish between areas if a density of ammocetes in one area will change differently than in others, due to some external factors (such as a barrier, treatments which sometimes treat only a part of the stream, etc.).

Reach sets are sets of reaches which taken together represent the whole stream. There can only be one reach set per stream for any given simulation run although its behaviour may change (see Barrier Efficiency Hypotheses). Alternative reach sets are used to represent different configurations of a stream (eg. with presence of a barrier simulated and with no barrier throughout the simulation).

Adding reach sets is necessary if a new representation of a stream is required (eg. a new barrier is planned). Let's suppose we would like to add a barrier to an existing simple one reach stream. Suppose that the new barrier will split the reach in two. (If it did not we could easily utilize the existing reach structure.)

The first step in adding a barrier is to add a new entry into the **ReachSet** table. This step automatically generates a new ReachSetID value. This value will be used throughout the database to refer to the new stream configuration.

Next we will add new reach definitions in the **Reach** table. It is up to you to select values for the Reach field. These cannot conflict with other values of Reach field for the same Lake and Stream. In our example assuming that the original reach value was 1, the new values could be reaches 2 and 3. Reach 2 represents the area below the barrier and reach 3 represents the area above the barrier. Most of the attributes of the new reaches are the same as for reach 1. The main difference is the habitat area (HabArea) which is now split into two. A description on how to initialize the population structure in each of the reaches can be found later in the *Results Database* section.

One situation that may arise when adding a barrier is that the barrier will block lampreys' access to all of the spawning habitat. This situation can be handled in one of the following ways:

1) Do not allocate any lamprey to any of the new reaches (**SpawnAllocFtr** set to 0). This will simulate lamprey staying at the mouth of the river and not swimming up the stream. In this approach the barrier trap, if present, will not catch any lamprey.

2) Create a "dummy" reach below the barrier and set the egg survival rate (**EggSurvRt** in **ModelReachParms**) to 0. This will simulate lamprey swimming to the barrier, spawning at the site with no eggs surviving. This approach will allow the barrier trap to catch lamprey swimming to the barrier. When the barrier is not active the allocation to this reach should be set to 0.

The new reaches should also be added to the **ModelReachParms** and **ModelReachAgeParms** tables. The entries will most likely be simply copies of the original reach number 1.

The two new values must also be added to the table **ReachSetReachLL** along with the ReachSetID of the new reach set. This step provides a formal link between the new reaches and the reach configuration. The value of DownStrReach indicates which reach is down-stream and which is up-stream. The reach flowing directly into a lake should have the value of DownStrReach set to its reach number. In our example the stream may have reach 3 flowing into 2 and reach 2 into the lake (set to flow into 2). Some of the reaches can be present in more than one reach set. For example, if the original stream configuration was composed of two reaches, the unaffected reach would be present in both reach sets.

Next we should add the actual barrier. The information describing the barrier is stored in the **BarrDef** table. As with the ReachSet table, adding an entry here generates another reference number: BarrID. This number will be used as a reference to the barrier itself. To link the barrier to the reach set we must also add a record to the **ReachSetBarrLL** table along with the location of the barrier (the reach immediately above the barrier).

Some of the old chemical options will apply, so we should refer to them in the **ReachSetChemOptLL** table with the new ReachSetID. (All of the chemical options relevant to the original (one reach) stream configuration should be applicable since the stream must be able to behave with the barrier active and inactive (i.e. not present).) The **ChemEff** table's most important purpose is to specify which reaches are affected by a given chemical option. A presence of a record with the appropriate reach number signifies that a reach is affected. A chemical option can affect reaches from multiple alternative configurations for a given stream. Only the reaches included in the currently selected stream configuration are actually used. In our example, we should add reaches 2 and 3 to all of the chemical options relevant to the original stream configuration.

Adding values to the **ChemEff** table generates new values of the ChemEffKey field. These should be added to the **ChemEffAge** with the appropriate chemical efficiencies by age.

Next we must check the table **BarrEffHyp** for existing barrier efficiency hypotheses relevant to the new reach set. All of the barrier efficiency hypotheses relevant to the original (one reach) stream configuration should be applicable since the stream must be able to behave with the barrier active and inactive (or not present). There may be new barrier efficiency hypotheses which describe activity with the barrier active, which should be added as well. See the following section for a detailed description of **BarrEffHyp** and related tables.

All of relevant existing barrier efficiency hypotheses should be added to the **BarrBarrEffHypLL** table along with the new ReachSetID and BarrID. The value of field IsActive should most probably be false since the original stream configuration did not contain a barrier. One of the new **BarrEffHyp** should become the default for the Reach Set. The most likely candidate is the one equivalent to the default in the original stream configuration.

The next step is setting up spawner allocation within the stream. Information about that is stored

in **SpawnAllocFtr** table. The table combines ReachSetID, Reach, and BarrEffHyp values determined earlier.

If applicable, trap information should be added to the **TrapDef** and the **TrapType** tables.

If there were any entries in the **ScenarioSMRT**, **ScenarioTrap**, **ScenarioEffHyp**, or **ScenarioEffHyp** these may also have to be duplicated.

As the last step, if the new stream configuration should become a default for the stream, the field ReachSetID in the table **SimStream** should be updated to reflect the change.

2.2 Barrier Efficiency Hypotheses

Barrier efficiency hypotheses are used to represent the various ways in which a stream can act. Some examples of these activities are appearance of a barrier, or a partial or a full failure of a barrier. There can be multiple barrier efficiency hypotheses defined for any configuration (Reach Set) and the behaviour can change during a simulation run.

To illustrate relationships within the database of barrier efficiency hypotheses, let us manually add a new barrier efficiency hypothesis to the database. The new barrier efficiency hypothesis will describe an active barrier within the stream added in the Reach Set section above.

Before we proceed with adding a barrier efficiency hypothesis we will add a chemical option which would be relevant to the new stream activity.

First we should add an entry into the **ChemOpt** table. The chemical option will treat only the part of the stream below the barrier, ie. only reach 2. A new entry will automatically acquire a value of ChemOpt. We use this value to add entries to the **ChemEff** table with only reach 2 (reach 3 is not treated) which in turn generates a new value of **ChemEffKey** which is used for entering chemical efficiencies in **ChemEffAge** table.

Next we should link the new chemical option with the reach configuration by adding an entry containing the new ChemOpt and the appropriate ReachSetID into the **ReachSetChemOptLL** table.

Now we can add a new entry in the **BarrEffHyp** table. The newly created ChemOpt value will become the default option for this barrier efficiency hypothesis. If there are also values for the default annual release of sterile males, they should be added as well.

The new barrier efficiency hypothesis will have a new set of spawner allocation values. These should be added to the **SpawnAllocFtr** with all of the reaches present. The reaches where there will be no spawners should receive an allocation of 0. All entries for a particular BarrEffHyp should add up to 1, but values less than 1 can be used to simulate lower nesting success (for more information about spawner allocation and nesting success see section about adding reaches to the reach sets).

It there are any traps placed on the stream which are active year after year (eg. a barrier trap) those should be added to the **TrapDef** table.

With the new activity set up we can turn the new barrier on. An entry in the **ScenarioEffHyp** table will turn the barrier on at the appropriate time in the simulation. The spawner allocation will automatically change to the new one we just set up. In addition, should the stream be scheduled for treatment the default will be the new chemical option. An entry in the table should only be placed there once. It will automatically be carried forward until another change is encountered. For example, should there be a time where we wish to simulate a complete barrier failure, we would need to add another value to the **ScenarioEffHyp** table with the old barrier efficiency hypothesis. The spawner allocation and the default chemical option will revert back to the old ones.

2.3 Lamprey Basins

Lamprey basins have a function in the Great Lakes similar to the function of a reach set in a streams; lamprey basins partition the Great Lakes into distinct sections. These sections can represent whole lakes or the parts to be simulated separately. Not all lakes need to be present in all of the configurations. If only part of the Great Lakes basin needs to be simulated, it is more efficient to set up a lamprey basin configuration containing only those parts needed for simulation. As with the reach sets, there are many ways in which the Great Lakes can be partitioned, although only one can be used in any one simulation.

There are four tables representing the lamprey basins. **LampBasinDef** contains a general description of the definition itself. This description allows for the identification of the lamprey basin and aids in the selection of which basin configuration is most appropriate for the simulation.

The table **LampBasinSubDef** lists all of the sub-basins included in the configuration. If we were to simultaneously simulate all of the Great Lakes we could define an entry for each lake in this table.

Tables **LampBasinToStream** and **LampStreamToBasin** are used to list all of the basins contributing spawners to streams and all of the streams contributing transformers to basins. There can be multiple basins from which spawners migrate up a single stream, just as a single stream can contribute transformers to multiple basins. An example of this is the St. Mary's river which contributes and receives lamprey from Lake Huron and Lake Michigan.

It is important to ensure that the sum of proportions of spawners for each basin and transformers for each stream add up to one, otherwise lamprey may be miscounted.

2.4 Database Files

The IMSL database was designed to protect static data and to provide a facility for keeping

multiple sets of results generated by minor modifications to a single scenario database (e.g. budget size, crew combination, etc.). This led to the creation of three separate component databases for each type of data:

1) Static and rarely changing parameters are stored in the options database;
2) Dynamic parameters subject to change between individual runs are stored in the scenario database; and
3) Results of the runs are stored in the results database.

Options Database

The options database (OPTIONS.MDB) contains data which rarely changes between runs. The database contains the following tables: **BarrDef**, **Criteria**, **Lake**, **Reach**, **ReachSetBarrLL** and **Stream**.

The tables **BarrDef** and **ReachSetBarrLL** define the barriers and contain the links between the stream configurations and the barriers (which barriers are placed on which streams).

Tables **Lake**, **Reach**, and **Stream** contain the names and the information about the size and location of streams.

The **Criteria** table contains the descriptions of the four evaluation criteria.

Scenario Database

The scenario database (e.g. SHIST.MDB) contains dynamic data, potentially subject to change from run to run. It contains the following tables: **BarrBarrEffHypLL**, **BarrEffHyp**, **BaseCamp**, **BaseCampDeplSiteLL**, **BudgetChem**, **BudgetCrew**, **BudgetGen**, **ChemEff**, **ChemEffAge**, **ChemOpt**, **CummInfRt**, **DeplSite**, **DeplSiteStreamLL**, **LampBasin2Stream**, **LampBasinDef**, **LampBasinSubDef**, **LampStream2Basin**, **ModelBasinParms**, **ModelGenParms**, **ModelReachAgeParms**, **ModelReachParms**, **NetTrapEff**, **ReachSet**, **ReachSetChemOptLL**, **ReachSetReachLL**, **ScenarioAction**, **ScenarioBarr**, **ScenarioDescription**, **ScenarioEffHyp**, **ScenarioFish**, **ScenarioSMRT**, **ScenarioTrap**, **SimStream**, **SpawmAllocFtr**, **TrapDef**, **TrapSummary**, **TrapType**, **TripLL**, and **UserEventOpt**.

Descriptions for most of these tables were presented in Section 2.1 and Section 2.2 of this report.

In addition, the tables **BaseCamp**, **BaseCampDeplSiteLL**, **DeplSite**, and **DeplSiteStreamLL** provide information on the three base camps and the deployment sites used from those base camps.

Tables **BudgetChem**, **BudgetCrew**, **BudgetGen** hold the information about resource availability.

Tables **LampBasin2Stream**, **LampBasinDef**, **LampBasinSubDef**, **LampStream2Basin** control the stream-to-basin and basin-to-stream allocations

Tables **ScenarioAction**, **TripLL** allow for the building of treatment schedules, while **UserEventOpt** holds user-defined options.

Tables **ScenarioBarr**, **ScenarioEffHyp**, **ScenarioFish**, **ScenarioSMRT**, **ScenarioTrap** provide non-default selections for specifications.

Results Database

The results database (e.g. RHIST.MDB) contains the individual age structured populations, basin populations and the stream rank list. All of this data is stored in five tables: **LampByBasin**, **LampByStream**, **LampByReach**, **AmmReachPop**, and **RankStream**.

Table **LampByBasin** contains the information about the size and the composition of all of the basin populations for the current configuration (definition). This table needs to include data at least for one year just prior to the first year of the simulation (new data is added during the simulation).

Table **LampByStream** contains the spawning run information for each of the streams in the simulation. Entries in this table are generated from the basin population sizes and the allocation found in the **LampBasinToStream** table described earlier.

The **AmmReachPop** table contains the age structure for all of the reaches in simulation. The initial population structure for a particular year of simulation is stored in records dated one year prior to that simulation year. Entries for each subsequent year are generated automatically. If new reaches (and new reach sets) are added to the scenario database the initial population structures must be added to the **AmmReachPop** table. This is the largest of all tables in all of the IMSL databases.

Table **LampByReach** contains the size of the spawner run and the summary information on the transformer production for the reach.

3.0 Revisions to Existing Database Structures

Based on our review of the current structure of the IMSL database we conclude that a number of changes in its format could improve the database speed, ease of maintenance, and overall consistency. Changes related to the relocation of database tables or to relatively minor changes in data elements and table composition are listed below. Changes that address more substantive modifications to the structure of the system that are primarily related to the implementation of new functionality are dealt with in following sections of this report.

3.1 Database merge:

Consolidation of Scenario and Options Database tables

Recommended Change: Most of the tables currently found in the scenario database can be moved to the options database. All of the tables in question contain relatively stable data eg. Chemical Options, or Model Parameters.

Tables Affected by the Merge: BarrBarrEffHypLL, BarrEffHyp, ChemEff, ChemEffAge, ChemOpt, LampBasinDef, LampBasin2Stream, LampBasinSubDef, LampStream2Basin, ModelBasinParms, ModelGenParms, ModelReachAgeParms, ModelReachParms, ReachSet, ReachSetChemLL, ReachSetReachLL, SimStream, SpawnAllocFtr, TrapDef, and TrapType.

Rationale: This move will make it easier to maintain and store data, and will decrease the overall requirement for storage space since all of the data will be shared by all of the scenario databases. This modification will also enable merging of some of the tables currently stored in the two databases.

Consolidation of Results and Scenario Databases

Recommended Change: In the original design, multiple results databases for a single scenario database could be maintained. Use of LCSS over the last year has proved that this feature is not used and will likely be not needed. It is recommended that the two databases be merged.

Tables Affected by the Merge: AmmReachPop, LampByBasin, LampByReach, LampByStream, RankStream.

Rationale: Merging the two databases will make the maintenance easier since there will be no need to maintain a separate pointer to a results database. This move, once fully implemented, may also improve the performance of the system since there will be no need to coordinate reading of the results data (ammocete

population, ranked list) with parameters in the scenario database.

3.2 Table Structure Changes

The following notes describe a number of changes to table structures within the IMSL/LCSS database. These changes typically extend the users ability to document the database (e.g. rationale for data derivations) by adding additional comment fields, or consolidate existing table entries to simplify/streamline database maintenance. In some cases the changes may lead to improvements in run-time efficiency of the LCSS.

ModelReachParms

Recommended Change: Once tables are moved from the current Scenario DB to the Options DB it will be possible to move the parameters from the ModelReachParms to the Reach table.

Rationale: This move will simplify the data structure by eliminating the table and it may improve the performance by simplifying the data queries unifying the reach parameters.

ModelBasinParms

Recommended Change: All of the data from this table can be stored in the LampBasinSubDef table eliminating the need for ModelBasinParms.

Rationale: This simplifies and streamlines the data structure.

ModelGenParms

Recommended Change: The data currently stored in the ModelGenParms table can be stored in 2 existing tables: LampBasinSubDef (once it contains the ModelBasinParms data) for the biological parameters L_inf and AvgLen1stTrans, and ScenarioDescription table for the rest of the parameters. These changes will eliminate the need for ModelGenParms table.

Rationale: This simplifies and streamlines the data structure.

ChemEff and ChemEffAge

Recommended Change: Replace the "artificial" common key ChemEffKey with ChemOpt and Reach.

Rationale: Using a combination of ChemOpt and Reach instead will make the structure and referencing to the data in those tables simpler and more consistent with the general principles of database design.

ScenarioComment — ***Recommended Change:*** Create a new table that will contain a date and a comment field to store time-sensitive comments about the current state of the scenario.

Rationale: This will allow users to track the evolution of a scenario.

Reach Table — ***Recommended Change:*** Add a new comment field to the Reach table to store information about how some of the numerical data (eg. habitat size) was arrived at.

Rationale: Documentation of the data sources will help users understand its significance within the model.

Consistent Naming — ***Recommended Change:*** Eliminate naming inconsistencies. For example, in the **BaseCampDeplSiteLL** table, the travel time field is named "TravelTime", and in the **DeplSiteStreamLL** table it is named "TravelTo".

Rationale: Using consistent naming will make it easier to develop the programs interacting with the database (eg. LCSS) and will decrease the maintenance required for the database.

3.3 Extensions to the Database Structure

3.3.1 Budget Groups

The budget groups can be used to partition the budget into separate sections and to operate from those sections separately. For example, budgets could be created for a Canadian and a US section, or individually for each of the Great Lakes.

Issue 1: What they will be used for

The current budget group structure (see Figure 2 in Appendix B) separates the budgets with a completely independent structures for budgets, crews and chemicals. All of the streams are separated into independent groups which are treated by the appropriate crews. This structure is best suited for separating the budget into Canadian and the US sections, or for individual base camps (Sault S. Marie, Marquette, and Ledington). Thus while it is appropriate for the operational side of the program, it does not allow a more "relaxed" approach in which money is constrained to individual lakes, but not the crews.

An alternative structure does not separate the budgets completely. It allocates a proportion of the overall budget to groups of streams (eg. Lakes). Technically, any crew can treat those streams. The cost of the treatment would be tallied up by the budget groups. This approach allows users to address greater policy questions, but lacks the operational accounting by crew.

Issue 2: How will they be applied

There are two ways of using either one of the above described database designs: as a constraint or as an indicator.

The constraint mode is most useful in the long term runs examining how various management strategies would affect the populations of lamprey in the various Great Lakes.

This constraint, however, would not be desirable when LCSS is used interactively to schedule streams. Budget groups should be used there as indicators to show the distribution of resources among the various groups of streams (Canada/US, Lakes) or crews.

3.3.2 Database Security & Multi-Site Coordination

There are several objectives related to database security, and coordinating the use of the database across multiple sites. These objectives include:

a) ensure the integrity of 'static'/'officially sanctioned data' (historical, chemical options, stream inventory, parameter estimates etc.);
b) provide a mechanism to combine information developed at more than one site. This is likely most relevant to short-term actual program planning where different agencies/offices may be responsible for portions of the program (e.g. Canada/US agents, treatment specialists/assessment specialists);
c) permit users of the system who will be exploring the model relationships to improve model predictions to virtually change anything they want; and
d) facilitate (c) by making recovery from mistakes or backing up to some previous system state relatively easy. This last objective may be the most difficult and complex to achieve, and is perhaps not as important as the first three objectives.

There are several ways of addressing these objectives. Possible alternatives are presented below, along with their associated advantages and disadvantages.

I. Every table in the database is modified to carry an up-date flag. In order to make it relatively easy to recover from mistakes (Objective (d)) this would have to be a cycle number (incremented when a run is made) or possibly a date field. Users could then specify the level to back up to. For this alternative to work, whenever a change is made to a record, the update flag is set to the cycle number or date and the record must be written out to an update database which has an identical structure but only update/changed records.

Advantage: Addresses objectives (a), (c) and (d).
Disadvantage: Complex and costly to implement. Does not address (b).

II. Use the native security features of MS Access to assign read/change/add authorities to different classes of system user.

Advantage: Helps to address (a) and may be useful in preventing unintentional changes to 'static' data.

Disadvantage: Does not address (b), (c) or (d). Authorities are assigned at the table level (and not for records within tables). In the original design, 'static' and 'scenario' data were completely separate. The lack of SQL table unions in the early version of Access forced a redesign and several tables were combined to enable development on schedule. This partly compromises the ability to fully use the native security features and means that very careful analysis will be needed to avoid frustrating system users.

III. Standardized operational procedures and conventions for recording updates or additions to the DB.

Advantage: This is easy to develop and change. If kept very simple and straight-forward, this alternative can be successful particularly with regard to objective (a).

Disadvantage: It would require a highly disciplined and somewhat bureaucratic mind-set to implement the complexity that would be required to meet all of the objectives. A high probability of failure is likely due to the inconvenience imposed on the user.

IV. Establish a mechanism which would permit defining the 'ownership' of parts of the database: a specific table (e.g. budget); a spatial domain (e.g. US or Canadian streams); functional domain (stream structure data, chemical option data, or barrier hypotheses - e.g. all affected tables); or a temporal domain (e.g. historical treatment data). This is somewhat similar to Alternative II except that it allows us to define whether there would be any 'sub-ownership' within a table. A combination of the alternative domain types (spatial, functional, temporal) could also be implemented.

Advantage: Addresses (b), and to some degree (a) and (c). It might also be used to partly address (d).

Disadvantage: This alternative is labour-intensive and complex to implement. It would be difficult to change once sub-ownerships and domain types were established.

Recommendation:

None of the alternatives explored above address all objectives.We recommend an incremental approach to implementing database security and coordination across sites. This approach combines elements from all of the alternatives described above. As the first step we recommend:

1. Establish an on-line backup of all of the databases that contain "official"data. The

default scenario and results database can currently be utlized for this purpose. The only database requiring a new backup os the options database.

2. Establish a set of guidelines on who should update which data and implement these guidelines as a set of utilities for comparing and merging databases.

3. Where necessary, set up password-level protection to databases.

4. Monitor database use and record the most common points of contention with respect to the data. Implement any necessary modifications to the procedures and tools.

We expect that the ways in which the system is used will change over time. For this reason, we recommend keeping the complexity of the security/coordination features to a minimum, as this will make it easier to maintain the overall system.

3.3.3 Integrating Assessment Data with the IMSL/LCSS DB

Integrating the assessment summary data with the LCSS database will allow us to compare the model and sampling-derived estimates, and thus contribute to the ongoing improvement of the model parameters. It will also ground the development of treatment plans in the "reality" of observational data.

There are two basic alternatives to achieve the above objectives:

I. Incorporate tables of basic assessment summary data: I) estimates of the relative abundance/size structure of larvae populations and transformers/transformation rates; ii) estimates of the relative abundance of adult spawners; and iii) estimates derived from (I) and (ii) of the overall transformer/spawner populations at a basin level. Provide standardized queries/reports/graphs to facilitate comparison.

II. Implement Alternative I above and in addition provide a means to force the 'modelled' populations to conform to the assessment estimates.

Recommendation:
The need for implementation at the second level is not immediately obvious and consequently the additional effort is likely not justified. Implementing the first level of integration between the assessment data and the LCSS database is an essential first step needed to determine whether the second level implementation may be necessary. We recommend proceeding with Alternative I, followed by an extensive effort to review the parameterization of the model before any efforts are made with respect to Alternative II.

The assessment database should be implemented as a separate database with individual tables linked to the scenario database. This approach will ensure that the assessment database can be maintained independently of the rest of the system, while also providing the flexibility of viewing

the scenario and results data together.

3.3.4 Reporting Basins

The main use of reporting basins is to group various data for presentation. The two categories of data most desirable for grouping are resources used (crew time, chemicals, money, etc.) and lamprey populations (ammocete size/density, transformer productions, parasite population sizes, damage to fish stocks).

Originally, we anticipated the need for a separate structure for reporting from that being simulated (lamprey basins, budget groups), although this may no longer be necessary. In our opinion, the simulated structures and their alternative settings can be used to group data for reports.

Appendix A: Initial Effort estimates

These effort estimates include only those changes that are required to the existing structure. They assume that design for sections not yet implemented are flexible enough to accommodate the new database structure.

ScenarioDB => Options DB 1d
- moving tables and linking
- no LCSS code changes
- updating database macros to link Options DB

Results DB => Scenario DB 14d +
- moving tables
- Scenario DB front end modifications (?)
- enabling the code (minimum changes only: opening the right DB, closing DB before compacting; sections affected: budget, long term runs, new DB interface, model, scheduler)
- integrating the changes, first step (combining the queries to read data)
- more time may be needed

Assessment DB => Options DB ? d
- does not exist yet

ModelReachParms 1.5d
- model
- interface (Biological Paramters)
- summary screen

ModelBasinParms 1.5d

ModelGenParms 3d
- DB
- model (allocations, etc)
- interface (Biological Parameters , Gen Opt)

ChemEff and ChemEffAge 1d
- model
- ChemOpt and ChemEffAge interface

TrapDef and TrapType 1d
- DB
- barrier cost/activity/spawner allocation

ScenarioTrapDef 0d
- interface does not exist yet

ScenarioComment 0.5d
- Scenario Description interface

Reach 0d
- no LCSS interface to physical parameters yet
- no DB front end yet (?)

Consistent Naming 3d +
- DB
- code
- may require more time

Appendix B: Graphical Representations of Database Structures

Figure 1: Reach Set Links

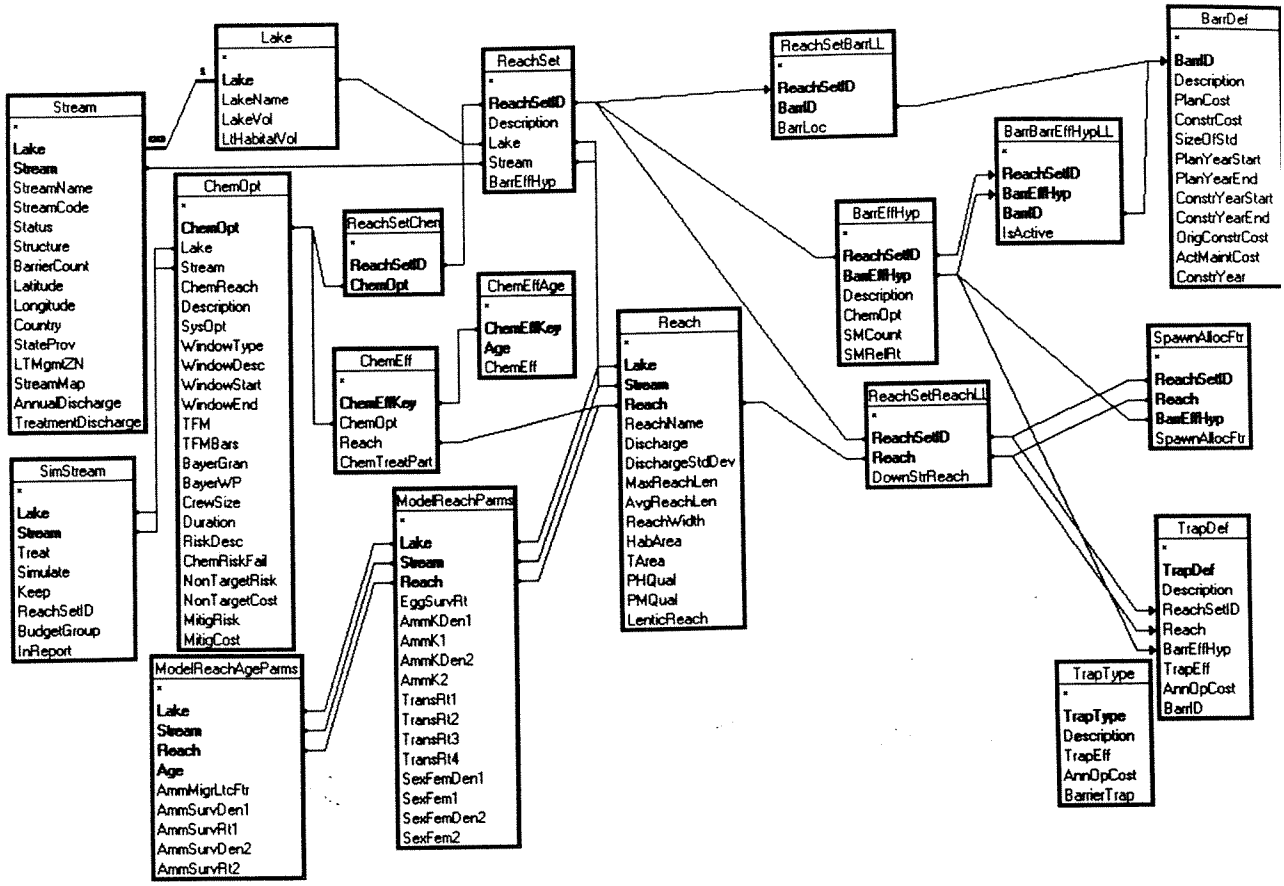

Figure 2: Budget Group Structure