A Database system for storage and analysis of neural stem cell data

Jonas Ladenfors

Sammanfattning

Denna exjobbsrapport beskriver planering och implementering av ett fleranvändaresystem för lagring av cellodlings data kring en databas med ett användargränssnitt av webbtyp. Verktyg med öppen källkod har använts, MySQL som databashanterare och Perl till användargränssnittet. Systemet kan användas på alla operativsystem med stöd för Perl och MySQL. Systemet har utvecklats under Linux men används även på andra typer av operativsystem. Systemet utgör idag en av delarna i NeuroNova AB’s forskningsinriktade programvaruserie.

KEYWORDS
Open-source, Linux, Apache, MySQL, Perl, RDBMS, Biotechnology, Software engineering, LIMS, Tissue cultivation

Examensarbete 20 p vid institutionen för informationsteknologi
Uppsala Universitet våren 2004
Contents

1 INTRODUCTION ........................................................................................................... 4
   1.1 PROJECT OUTLINE ......................................................................................... 5
   1.2 BIOINFORMATICS, A NEW COMPUTER SCIENCE FIELD .................................. 6

2 BACKGROUND .............................................................................................................. 7
   2.1 THE NEURONOVA RESEARCH ENVIRONMENT ............................................. 7
      2.1.1 Users of the tissue cultivation system ....................................................... 10
   2.2 TISSUE CULTIVATION .................................................................................. 11
      2.2.1 Workflow analysis .................................................................................... 11
   2.3 THE CURRENT BIOINFORMATICS ENVIRONMENT ...................................... 13
   2.4 LIMS, A BIOINFORMATICS SYSTEM DESCRIPTOR ........................................ 14
      2.4.1 Developing or investing in a new LIMS system ........................................ 14
      2.4.2 Data storage solutions ............................................................................. 15

3 TISSUE CULTIVATION SYSTEM DESIGN & IMPLEMENTATION .......... 16
   3.1 SYSTEM REQUIREMENTS ............................................................................. 16
   3.2 SYSTEM ARCHITECTURE ............................................................................ 17
      3.2.1 Code modulization .................................................................................. 20
      3.2.2 Cell batch naming standard ................................................................. 21
   3.3 DATABASE LAYER ....................................................................................... 22
   3.4 FRONT-ENDS ............................................................................................... 22
   3.5 SCHEMA & NORMALIZATION ...................................................................... 23
   3.6 KEY GENERATION ...................................................................................... 25
   3.7 DATA LOCKING ........................................................................................... 27
   3.8 TRANSACTIONS .......................................................................................... 28
   3.9 SECURITY & INTEGRITY ............................................................................. 28
   3.10 RECOVERY FUNCTIONALITY ..................................................................... 29
      3.10.1 Backup woes ....................................................................................... 29
      3.10.2 Recovery test ....................................................................................... 30
   3.11 CPAN MODULES ....................................................................................... 30
   3.12 GRAPHICAL USER INTERFACE ................................................................ 31
   3.13 SYSTEM REPORTS .................................................................................... 32
   3.14 AUTHORIZATION & USER ACCESS ............................................................ 34
   3.15 SOFTWARE ENGINEERING ...................................................................... 36

4 USAGE EXAMPLE ....................................................................................................... 39
   4.1.1 General .................................................................................................. 40
   4.1.2 Search .................................................................................................... 41
   4.1.3 Add ......................................................................................................... 42
   4.1.4 Reports .................................................................................................. 43

5 SUMMARY .................................................................................................................. 45
   5.1 SYSTEM DATABASE RESULTS .................................................................. 45
   5.2 FUTURE ENHANCEMENTS ........................................................................... 45
5.3 AN OBJECT RELATIONAL APPROACH ................................................................. 46
  5.3.1 Adopting the schema.................................................................................. 46
  5.3.2 ORDBMS conclusion................................................................................ 47

6 ACKNOWLEDGEMENTS ....................................................................................... 48

7 REFERENCES .......................................................................................................... 49

8 ABBREVIATIONS AND DEFINITIONS ............................................................. 51

APPENDICES ............................................................................................................. 53

APPENDIX 1 - PICTURES ......................................................................................... 53
APPENDIX 2 – REPLICATION GUIDE................................................................. 56
APPENDIX 3 – CELL DATABASE RESTORE SCRIPT ....................................... 58
APPENDIX 4 – DATABASE CREATE SCRIPT ..................................................... 60
1 INTRODUCTION

NeuroNova AB is a privately held, Stockholm based, biotech-company. The company’s main focus is finding new treatments for degenerative disorders in the central nervous system like Alzheimer’s disease, Parkinson’s disease and stroke. Today drugs to treat patients with degenerative CNS disorders can only slow down the process and the patient is never restored to the state he was in before the onset of the disease. NeuroNova’s aim is to pioneer stem cell treatment for neuro-degenerative diseases.

Dr Jonas Frisén and Dr Ann Marie Janson founded NeuroNova AB in 1998. They are both researchers at Karolinska Institutet in Stockholm, Sweden. They, and fellow researchers, showed that the ependymal layer of the brain contains stem cells that have the ability to mature into any of the major cell types of the brain and thus replace neurons, neurogenesis, which is an ongoing process even in the adult brain. This discovery is the foundation for much of the research done at NeuroNova. Research is conducted in several different ways but focuses on two different strategies for treatment of disorders in the central nervous system, Therapeutic Neurogenesis and Cell Therapy.

Therapeutic neurogenesis is a strategy in which you stimulate the endogenous growth of nerve cells by treating the patient with a substance. The idea is that the cells will migrate from the proliferative zones in the brain to the damaged areas, in Parkinson’s disease and repopulate it with fresh neurons. This regeneration of lost tissue would allow the patient to recover and regain functional properties lost in the disease.

Cell therapy is a strategy wherein you grow neural progenitor cells (stem cells) in vitro. You then differentiate them into the desired cell type, e.g. neurons in Parkinson’s disease, the stem cells or differentiated cells are then transplanted into relevant brain region of the patient.
1.1 Project outline

This master thesis project aims at developing a tracking system for the data generated in Neuronova’s tissue cultivation activities. This system will replace the earlier flat file system. The system should decrease workload and serve as a tool for communication and analysis. NeuroNova should benefit from this system in their target and lead development projects by providing the existing in-house systems with data and traceability. The system should also serve as a warehouse database for cells in cultivation or in storage. By gathering scientific tissue information, progress- and status data the system should be able to increase the employee’s interest and involvement in the tissue cultivation process.

The tissue culture group isolates and cultivates adult stem cells in vitro from mice or other brains for scientific experiments. The workflow between isolation and experiment should be covered by this system. Several different parameters are recorded including experimental setup, growth conditions and cellular characteristics at different time points. The information is later used to deduce knowledge about cellular behavior, experiment results and cellular phenotype. As a software developer you face a challenging task aiming to capture the rapidly changing world of biology. Conflicts arise when trying to adapt the software to a predefined laboratory work process. You are required to understand company goals and methods as well as the scientist demands and requirements.

The tissue cultivation system will be incorporated into the existing LIMS environment at NeuroNova as shown in figure 1. The known tools already used at NeuroNova will be used in the construction process for compatibility issues. The system will be developed together with the main user group, i.e. the tissue cultivation group. The tissue cultivation database will only handle the culturing activities and documentation conducted by the tissue culture group. Experiments and other post cultivation tissue manipulation will not be dealt with by the system. Any data sharing aspects between existing systems at NeuroNova AB is not developed but prepared for. Only a limited support for report generation is present in the system because of time limitations. Report generation should be done through third party
software or by the simple methods described in chapter 3.13. The projected will be conducted at NeuroNova in Stockholm, Sweden with existing computer and human resources at NeuroNova.

1.2 Bioinformatics, a new computer science field

Bioinformatics is a generally described as a mix between computer science and biology and chemistry. A successful bioinformatician must be able to understand all of these fields if he is going to be able to understand and solve any bioinformatic problems. Therefore it is today very common that biotech companies have a dedicated bioinformatics department that only deals with development and maintenance of systems that capture or store bioinformatics data. Historically chemistry is the most important part of bioinformatics but in the last couple of years biology has become an equally important part. For example DNA sequence data is now routinely captured as well as data from biological literature or clinical trials.

Documentation of research data is very important in the bioinformatics environment no matter the size of a company. The research data must also be quality controlled, traceable, storable and retrievable in a structured fashion. As biotech companies grow larger, with more employees and more research results the basic data acquisition methods become inadequate. Storing large quantities of data in lab books or Excel spreadsheets eventually reaches a point where it is no longer possible to guarantee the scientific data integrity. It is here bioinformatics is needed, research results are the most valuable assets found in a biotech company and therefore a high quality data storage system that guarantees data integrity and includes intelligent acquisition methods is vital for such a company’s survival.
BACKGROUND

2.1 The NeuroNova research environment

Today’s research environment at NeuroNova consists of three research sections the *Target discovery*, *Lead discovery* and *In-vitro* section. Through individual work and collaboration in between these groups a potential drug candidate is created. Each section has a specific role and place in the drug candidate chain. This is shown in figure 2 and described in the text below.

The first group in the drug candidate chain is as shown in figure 2 the *Target discovery section*. This section is responsible for finding the initial stem cell origin for a potential drug. The discovery can originate from almost any biologic material in the world. Needless to say this is an information intense environment. Much of the work in this section is performed by searching external and internal publication- and gene- databases. A trained scientist looks for certain characteristics in the search results and can through them find possible candidates for future drugs. The amount of data that needs to be presented and handled is huge and that is why bioinformatics is something very important for the target discovery section. This is also why the bioinformatics department at NeuroNova is closely connected to the Target discovery section. The efforts in the target discovery section results in a suggestion for a suitable biological material that could lead to a candidate drug. The suggestion is passed on to the next step in the drug candidate chain *the lead discovery section* as seen in figure 2.

*The lead discovery section* receives their material from either the *target section* or external resources. Their main objective is to see if a possible drug could be constructed out of the biological material. They check patents, competitor analysis, collaborations and known side effects. The lead section also specifies which tests should be performed on the substance. When the lead section decides that the biomaterial is suitable for drugs they inform the *in-vitro group*.
The *in-vitro section* then goes to work on the suggested substance. The substance is tested in-vitro (outside of the body) often on small groups of cells in laboratory dishes. Figure 3 shows what scientists from the in-vitro section wants from the tissue cultivation system:

1. Is there any tissue available?
2. Report tissue that has been used in experiments.

A scientist from the *in-vitro* section also needs to know what type of tissue he is using to be able to verify his experiments. In case of ambiguous experiment results data traceability is important, for example when a cell needs to be traced to its origin.

![Figure 3. The usage of the tissue cultivation system.](image)

If the tests performed by the *in-vitro* group gives satisfying results the tissue is transformed into a candidate drug substance and preparation for in-vivo (inside the body) experiments starts. In-vivo experiments are not currently performed in-house at NeuroNova.

Beside these three research groups there exists a section at NeuroNova that only deals with tissue cultivation, the *tissue cultivation section*. This group does not perform any tissue experiments. The group is only responsible for tissue cultivation and tissue availability. This group represents the main user group in this project, all other system user groups are described later in chapter 2.1.1. The name tissue cultivation is sometimes referred to as cell cultivation. In this report tissue cultivation is most frequently used.
The tissue cultivation section sees to that each week somewhere in-between 10-20 batches of cells are grown. Each batch is sufficient for one experiment. The batch of cells comes from a donor tissue and is after a tissue purification process ready for experiments. Information on both the donor and the purification process is stored in the system as well as information regarding each batch of cells. This relationship between the donor and batches of cells are explained more in detail in chapter 2.2.1.

What does a tissue cultivation member expect from the system?

- How many cells are there currently available
- The condition of these cells
- Tracking diseases or other things that have a negative effect on the cultivation process
- Knowledge on what other members of the tissue cultivation group has done
- Workflow documentation
As shown in figure 4 a tissue cultivator cultivates cells and retrieves data from the cultivated cells. The tissue cultivator then goes to the tissue cultivation system with the new data. He adds the information to an existing cultivation project or starts a new one. He then returns to the cultivation and repeats the process until the cell is used in an experiment or thrown away. This process is described in greater detail in chapter 2.2.

![Figure 4: Usage of the tissue cultivation system](image.png)

**2.1.1 Users of the tissue cultivation system**

Four different user groups:

1. Tissue culture group
2. Other scientists
3. Management group
4. Bioinformatics group

The groups are presented in priority order. The tissue cultivation group represents the main users of the system and was involved in the system development together with the bioinformatics group. The software is designed to fit the way that the tissue cultivation group works. The tissue culture group’s main objectives are production of cells for experiments. This includes information gathering and maintaining tissue quality. Secondly the group also conducts tissue experiments. The tissue culture group works in a cell laboratory and considers the tissue cultivation data as valuable information and can clearly see the benefits of the forthcoming system for testing of experimental drugs and candidate drugs. The tissue cultivation scientists are well educated in the use of web-based applications. Some users have previous experience with commercial LIMS solutions. The users have explicitly expressed wishes that the new system should decrease the workload and not as some of them fear make every new report time consuming and complicated.

The other scientists perform experiments on cultivated tissue and tracks tissue origin, keeps traceability throughout and between experiments. The system should also group and present relevant data that are important for this group.
The management group needs reports and correctness in the program to be able to analyze and manage the overall situation and see that the company goals are being fulfilled. The bioinformatics group performs administration, updates, and bug fixes. This group needs documentation and ability to perform updates of the program. This group is also responsible for installing third party software.

2.2 Tissue cultivation

The tissue cultivation process can be seen as an expanding tree structure where the root of the tree is either a mouse or other donor as seen in Figure 5. A group of donors of the same species can together form a tree root, if and only if the extracted tissue is gathered from this one group. The next step consists of extracting a piece of tissue from the donor. One or several different tissue types can be extracted from the donor. From the extracted tissue a group of cells are isolated, from one tissue sample several isolations can be performed. These three processes together form a combination referred to as the tree root or the unique key.

When the scientist has completed the first three steps (donor, tissue, isolation) the cells reach passage zero. Passage zero contains several isolated cells (referred to as a batch of cells or a cell-batch). The cells are derived from the isolation step and are orphans in regard to previous batches of cells. As cell cultivation is performed the cells pass different time points. These are referred to as different passages. The passage informs the cultivator and people using the cell on the age of the cell. A cell often never survives more than 10-15 passages. Each batch of cells is unique and can have one or several parent batches. The cell can also form one to many parent relationships with any cell batch from the same donor.

2.2.1 Workflow analysis

The workflow can be separated into two different parts, part X, the isolation of primary cells from donor, and part Y, the expansion of primary cells in vitro. The two parts are always done in order where X always precedes Y. Part X can also include several sub-parts, which by themselves can form foundations for unique Y parts. An X sub-part consists of different tissue extracts from a specific donor or group of donors.
X: Retrieval of primary cells from donor

Part X contains three steps.

1. Donor
2. Tissue
3. Isolation

Establishing donor information is the first step in the tissue cultivation workflow. The scientist starts by extracting tissue from either a mouse or a other donor. The donor information is registered in the system and a unique key is generated. The donor can only be identified through the information saved in the system. The scientist registers tissue information into the system when the ventricular wall or other suitable tissue is extracted from the donor’s brain. After this step the scientist isolates stem cells from the extracted tissue. The isolation process is conducted by placing the extracted tissue in an enzyme solution that disrupts the cell bands between the stem cells. After finishing the isolation step the cell batch has reached passage zero and part Y has begun.

Passage zero can be seen as the connecting layer between Part X and Part Y. This passage contains all cells available in the isolated tissue. The cell batch in passage zero does not differ information wise from cell batches in part Y apart from the fact that the batch of cells in passage zero does not have a parent cell batch. All cell batches present in later passages have either one or more parent cell batches. All of the cell batches in passage zero are derived from the isolation process and therefore do not have a batch of cells as parent.

The program needs to capture the following information in the X part.

• Donor information, species, strain, time, date and whether the donor is a disease carrier or not
• Tissue information, extraction date, kind of tissue, frozen/thawed
• Stem cell information, isolation enzyme solution, seeding density
• Passage zero information, Number of cells, Number of growth containers

Y: Expansion of primary cells in vitro

Part Y is derived from part X and cannot be created without an existing X part. Each phase in the cultivation is separated by a passage. Depending on the cells expansion-rate a suitable cell split is made. In a cell split the growing cells are separated into one or more containers for further expansion. Beside expansion a cell split can also be placed in a freezer to later be used in an experiment or given as a handout to a scientist.

Handouts are split cells or tissue material given to different scientists at NeuroNova AB. The handout can be used for an experiment of continued expansion. The system notifies the scientist involved in a handout procedure by email and allows the scientist to notify the system that the handout has been picked up.

The program needs to capture the following information in the above procedure.
• Batch information, number of cells, cell status, date, relationship information, protocol, container
• Status information, freezer data, picture data
• Handout information between scientists

2.3 The current bioinformatics environment

When this project started NeuroNova had two large research systems. Each system was created especially for one section at NeuroNova and none of the systems shared any data in between each other. All systems share the same physical user group but each system has its own separate login procedure. The target discovery system helps the target discovery section to fulfill its goal in finding a suitable biological material for a candidate drug. The system helps a scientist to search from several different data sources and through those search results the target discovery system creates a score. The scoring mechanism is very complex and includes many different stem cell specific parameters. The other system is used by the lead section and is a data warehouse house for biological material information. A data warehouse is defined as a set of tools, technologies, and methodologies that allow for the construction, usage, management, and maintenance of the hardware and software used for a data warehouse, as well as the actual data itself [35]. In NeuroNovas data warehouse biological data is captured and stored to later be used in scientific reports, conclusions or comparisons. Today partial data sharing is possible between the lead system and the tissue cultivation system. Any cell found in the tissue cultivation system that was used in experiments by the lead group can be traced to the tissue cultivation system by a specially developed module in the new lead system. This module performs queries on the tissue database to connect substance experiments with cultivated cells. This is possible through the key generation standard explained in chapter 3.2.2. The tissue cultivation system does not

![Diagram](image_url)
fetch any data from the lead system. Work is currently being conducted to interconnect all of the interfaces and databases at NeuroNova AB. This umbrella system is referred to as the NeuroNova DiscoverySuite. The DiscoverySuite will feature and interconnect all of NeuroNova’s research tools where the tissue cultivation database system built in this project is one of the cornerstones. A complete system overview can be found in figure 6. The dotted lines represent future work and the solid arrows represent current connections between the in-house systems.

From a user perspective all of these systems serve the same purpose, helping research. In that aspect sharing data for global research report generation or advanced queries is important.

2.4 LIMS, A bioinformatics system descriptor

What is a LIMS? The abbreviation stands for Laboratory Information Management System. Originally a LIMS system collected raw data from research machines and assays. It often accompanied research machines. Today a LIMS is a general descriptor for computer systems used in biological or chemical research environments. It is often sold as only software and is not restricted to one type of research machine. A modern LIMS system often works with several different types of research equipment or by itself. LIMS systems today can contain anything from a single data storage with an interface to a complex system with analysis tools and report generation for a complete company.

2.4.1 Developing or investing in a new LIMS system

On today’s commercial LIMS market there exists quite a few systems similar to the one created in this report. The market is largely dominated by large companies such as MDL[11], SoftwarePoint[12], IDBS[6] and Accelrys[7] to name a few. These companies develop systems that are generally very sophisticated and highly advanced, with a matching price tag. To name an example IDBS has a solution called ActivityBase[6], that besides EDC and warehousing has sophisticated molecular structure applications with report and administration features. ActivityBase has been developed for more than 10 years and is very well tested and supported. This is a system that can satisfy several needs in several different departments in a biotech company. ActivityBase is a typical commercial system where a total research solution is the main focus. These type of total solution systems can seldom be incorporated into existing systems. Which means that all previously developed software needs to be replaced. But there are other examples of more flexible commercial solutions that work in the same field as this project system.

SoftwarePoint has with their solution WilabLIMS/LimsBOSS come one step closer to what NeuroNova needs. Their product can be bought in small modules which can satisfy small parts of a bigger system environment. Another example is Open BioLIMS from the Aldrix group. Open BioLIMS is constructed in a very similar way to this project system. It is also based on the open-source licensing model which allows the user to customize the system in any way he wants to without breaking any copyright restrictions.

Solutions like ActivityBase are generally more suitable for companies with 100 or more employees and cannot be seen as a system competing with the one constructed in this project.
WilabLIMS/LimsBOSS or Open BioLIMS can on the other hand be used and in Open BioLIMS case the software can even be customized to fulfill custom requirements not originally found in the system.

### 2.4.2 Data storage solutions

In the academic bioinformatics field MySQL is without a doubt the most common RDBMS in use, but competition is fierce. Oracle is a strong competitor claimed to be the world’s most comprehensive commercial RDBMS systems. But that strength comes with a substantial cost in licenses and education. Another strong contender is PostgreSQL. PostgreSQL comes with a lot of interesting SQL99 features that MySQL lacks like stored procedures, triggers, and views. PostgreSQL is freely available under a BSD license. A comparison between MySQL and PostgreSQL shows that PostgreSQL is the one with the most advanced features of the two whereas MySQL on the other hand is the one with highest usability and availability. MySQL can at a glance seem like a little brother in a function comparison. This is at a closer inspection not the case. MySQL has a wider base of administration and reliability tools than PostgreSQL. The MySQL package comes with built-in replication and extensive testing abilities. PostgreSQL needs more administration because of its transaction id implementation. The transaction implementation in PostgreSQL requires a manual cleaning (vacuuming) of the transaction id variable from time to time. Performing such a task is both time consuming and resource demanding.

Both PostgreSQL and MySQL are licensed under an open-source philosophy but still differ a lot in how development is conducted. MySQL is developed by a closed group of developers. The source code is mainly supplied to enable an advanced user to add own features or track bugs.

PostgreSQL is developed in a more open environment resembling the Linux development cycle. In the PostgreSQL development an open group of developers write source code and from that source code repository a core group select what should be included in the next release. These two methods share the open-source mentality but are still very different. PostgreSQL has a more rapid development cycle and MySQL a more well tested and documented source code base.

PostgreSQL and MySQL both perform well in different environments and cannot be seen as competitors but instead as different tools for different requirements.
3 TISSUE CULTIVATION SYSTEM DESIGN & IMPLEMENTATION

3.1 System requirements

The system architecture has one critical requirement.

- Use already available development tools.
  - PERL, MySQL, Javascript

Original system requirements

- Store information about donor and extracted tissue
  - Donor and tissue information can be used for backtracking failed experiments or statistics.
  - This data should be saved to the system.
    - Donor species
    - Company from where the donor was bought
    - Postmortem time span
    - Date and time of tissue extraction start
    - Date and time of tissue extraction end
    - Responsible person

- Store information on how the cells of interest were isolated from the donor material
  - When a suitable tissue is extracted from the donor the isolation process starts.
    The isolation process goal is to create a small batch of isolated cells that can be used for cell cultivation.
  - This data should be saved to the system.
    - Date and time
    - Responsible person
    - Isolation solvent used
    - Approximated number of cells in the isolated cell batch

- Store information on cell expansion
  - This is the main process of the tissue cultivation group. It is here cell batches are created for later use in experiments.
  - This data should be saved to the system
    - Date and time
    - Responsible person
    - Approximated number of cells in the current cell batch
    - Cell batch state: frozen, experiment, dead or expansion
    - Division for the next expansion phase: 1:1->1:N

- Store information about the protocols used
  - It exists a number of different protocols for each step of the tissue cultivation process. Each step should be tagged with the protocol used.
Present the stored data to allow the scientist to estimate the outcome when altering various expansion, isolation or experimental parameters.
   - Create a web interface.
   - Data in the web interface should be accessible and understandable.

Report generation
   - Create a report system, where donor and tissue information is gathered together with the all the cell batches connected to that donor tissue combination.

Create Authorization/Access module
   - At every system session start each user is required to leave a personal username and password.

System should perform backups in regular intervals
   - The information in the database should be sent to a tape station in regular intervals.
   - Interface and logic should be apart of a CVS system.

The original system requirements where initially specified in a requirement specification by Karl Nyberg [22].

New requirements

- The system should be compatible with the previous systems at NeuroNova
  - Use software development techniques already available at NeuroNova.
  - Make the new tissue cultivation system friendly to other existing systems.
- Extend data capturing abilities both in the donor/tissue phase and in the cell cultivation phase.
  - Different donor species has some different data requirements
    - It exists several different mice types
  - Digital pictures of the cell cultivation should be saved to the system.
  - A cell in cultivation should be allowed to switch responsible person.

- Interface design
  - Increase usability
  - Interfaces for inserts, searches, and manipulating of existing data.

- Documentation/Manual

- Test setup
  - Test phase setup with real lab data.

- Extend report generation
  - Tree report generation.
  - Excel reports.
  - Bar chart reports.

The new requirements were developed by me in corporation with tissue cultivation group and technical aspects were developed together with the bio-IT group.

3.2 System architecture
The system built in this thesis using a Semi-three-tier architecture. A true Three-tier architecture is any system which enforces a general separation between the following three parts:

- **Client Tier** or user interface
- **Middle Tier** or application logic
- **Data Storage Tier**, Database layer

True Three-tier architectures can today be found in most modern web development processes like J2EE. J2EE is an acronym for Java 2 Enterprise Edition which is a Java based web based system development process that uses java server pages (JSP), Javabeans, Java classes and Java Servlets. The Java parts connect to the different Three-tier architecture layers in the following way [35]:

- **Client Tier** - JSPs or **Servlets** responsible for creating HTML user interface pages
- **Middle Tier** - **Servlets** or **JavaBeans** responsible for application logic
- **Data Storage Tier** - **Servlets**, **JavaBeans**, or **Java classes** responsible for data access.

PERL/CGI resembles Java Server Pages (JSP) in the way that they are executed in the client tier and presented as HTML. PERL/CGI can also create modules in the middle tier which are a cross between Servlets, Javabeans and JSP pages, PERL/CGI modules are described further in chapter 3.2.1. A Middle-tier often resides on one or many application server/s which are separated from any web server. The system in this thesis lacks a dedicated application server and only uses a web server in Middle-tier for both applications and HTML page generation. Connections to the Data storage tier are done in a similar way in the Three-tier and Semi-three-tier architecture. In Three-tier it is performed through Servlets or JDBC, in Semi-three-tier the connection is performed through a module called DBI.

The most significant difference between Semi-three-tier and true Three-tier is the isolation between the different layers which is stricter in J2EE then PERL/CGI. A figure that describes a Semi-three-tier architecture is show below.
The Client-tier and Middle-tier are both executed on the web server. The difference between the two is that while the PERL/CGI pages construct HTML system interface pages the modules just perform application logic. The Modules uses the DBI API to connect to the Data storage tier when data needs to be physically stored or fetched. A user can connect to the system through a combination of several different PERL/CGI generated pages. System functionality will always be accompanied by a PERL/CGI generated interface. The interface architecture is arranged in the way described in the “A system map” figure.

For a more detailed view on how each of the PERL/CGI interfaces connects to each other see the interface architecture figure. All squares represent an interface page. See chapter 4 for usage examples.
3.2.1 Code modulization

Perl has a very simple object oriented functionality that does not include header files or inheritance abilities but it still enables a more structured way of software development organization. Classes in Perl are called packages and can be recognized be their .pm extension. Dividing the software into smaller blocks where each block is represented by a package makes the software easier to understand and debug. Each package acts as a small independent part of the system and can transparently be replaced with a completely different structured version. Software without one standardized language has a lot to gain by encapsulating code in smaller blocks. By encapsulating code you achieve the impression that only one language is used and you achieve a standardized output language. In this project I have tried to encapsulate all client-side scripting code. This has allowed me to keep a clean Perl module layout.

A package in Perl does not have a C style function header. Instead each Perl module takes an array of scalars as input. There are no length or type restrictions on the input array. The input array is called @_ where the first element in the array always points to the initialized object of the module, similar to the this variable in C++.

Perl’s non-existing method headers make method description necessary in a multi developer environment. Each method was therefore described in the package head. The methods were tagged with a number and a comment declaring what parameters the module used. This helped other developers to understand how a module could be used without knowing how it was built.

figure 8. The interface architecture
SQL modules
SQL modules were constructed to keep SQL language out of the web interface page files. Every module was developed with general recycling in mind.

- SelectSQL.pm - select statements
- InsertSQL.pm - insert statements
- UpdateSQL.pm - update & delete statements

SCRIPTS.PM
Encapsulation of JavaScript and VBScript and other obscure functions.

INTERFACE.PM
Page headers and footers. All web forms.

INTERFACEUPDATE.PM
Update interface functions were separated from other interface functions to keep the interface module in a reasonable size.

3.2.2 Cell batch naming standard

Each element in the cultivation process has a unique key. This key is derived from the cells origin and place in the tissue cultivation cycle. It is automatically generated by the system to keep a consequent naming standard. Each key has a unique root name extracted from the donor/tissue/isolation.

Example 1, batch naming standard
root name: MM240T4I2
From this root name you can see that the donor is a mouse [MM] and is donor number 240 of that strain in the system. You can also make out that at least three tissues have previously been extracted and that this is the fourth tissue [T4]. An isolation has also been made earlier from the [T4] tissue. This is understood from the isolation number [I2].

On a unique root name a group of passages can be built. The passage count starts at zero and is then increased with one at every passage switch. Passage naming always starts with the letter P followed by a number. A passage consists of one or several cell-batches. A cell batch is named from 1 to the amount of cell-batches in the passage.

Example 2
A passage & cell batch naming built upon a root name
MM240T4I2_P34.12

In this case the name describes the 12:th cell-batch in passage 34.

The underscore between passage and isolation is used to make it easier to read the name. The complete key is later colorized when displayed in the system were the strain and passage is shown in blue, tissue in green and isolation in red.
3.3 Database layer

MySQL was the RDBMS of choice at NeuroNova and therefore used in this thesis. The MySQL RDBMS engine supports several different table types like InnoDB, MyISAM and BDB. MyISAM is the standard table type in MySQL since version 4.x. The previous table type ISAM is deprecated and no longer used. InnoDB is a table type developed by PhD Marko Mäkelä at Helsinki University. Today InnoDB is a product of Innobas Oy. InnoDB is known for its transactional abilities and for its unique row-level locking. BDB, also known as BerkleyDB, is developed by Sleepycat software (Massachusetts, U.S.A) BDB is a transactional table type and was the first table type in MySQL to allow this.

In an environment where inserts and deletes often occur a transaction-based table-type is preferred. Transaction procedures manage concurrent connections but not database consistency. Consistency is managed by foreign key constraints. InnoDB handles both transactions and foreign key constraints, which makes it suitable for the tissue cultivation database at NeuroNova AB.

3.4 Front-ends

A client-user needs to be able to insert and retrieve data from the tissue cultivation system. The client-user is not aware of how the system is built and does not care about that either. There are several ways for a client-user to connect to a MySQL system were one of the quickest ways is using the accompanying console. This is a quick way to connect or debug a MySQL database but the MySQL console is not suitable for reports or inserting or receiving large sets of data.

Therefore MySQL has a wide variety of third party client software that enables a more hassle-free access to the system. The most common one is probably the database PHP interface MyAdmin which has excellent support for MySQL. MyAdmin is a script that presents a web interface with elegant shortcuts to many of MySQLs features. Most user administration, inserts and retrieval tasks can be performed through MyAdmin but it still contains an abstraction level that is too complicated for most normal users. MySQL administrator is graphical administration software constructed by MySQL AB. This software is not included in the MySQL database distribution but is freely available elsewhere through the GNU GPL license. MySQL administrator has almost all available features as its console counterpart. But in MySQL administrator the features are presented in a graphical environment. The software is available for both Linux and Windows.
Both these software’s are useful in development but not for normal client usage. They cannot be configured to create custom made reports or complicated data manipulation and has a simple access control which is not sufficient to this project. To satisfy the aim of this thesis I created a custom made interfaces (GUI). The interfaces are based on Perl for layout a general sketch is shown in figure 10. The tissue cultivation system interfaces are built using Perl mainly because it is a familiar and stable language combined with extensive documentation. Perl is a language that is never interpreted by the client as both JavaScript and VBScript are. All Perl code is executed on the web server. This technique is known as server-side scripting. Perl has a wide range of modules freely available on CPAN[4]. A large part of the report tools created in the tissue cultivation system uses freely available modules from CPAN. The famous GD library and Spreadsheet::WriteExcel makes report construction fast, easy and reliable. The interface creation is discussed further in chapter 3.11.

The amount of different web techniques (server side scripting, client side scripting) available on the web makes strict use of only one language hard and a combination of languages in software can create confusion and make the debug process difficult. By using an object oriented programming technique the language combination difficulties were overcome as mentioned in chapter 3.2.1.

### 3.5 Schema & normalization

After finalizing the requirements for the system a data storage was created. The data is stored in the data storage tier of the architecture as shown in figure 7. The database tier is made out a database schema on a RDBMS system. When creating a database schema I followed the steps in the normalization process which was first proposed by Codd in 1972 [13]. The normalization process requires the database schema to undertake a series of tests. There exist at least six different normalization tests, 1NF, 2NF, 3NF, 3NF Boyce-Codd, 4NF.
and 5NF. The purpose of the normalization tests is to minimize redundancy and minimize insert, update, and deletion anomalies. Normalization is something that serves a purpose but is something that is hard to achieve on a live system. Minimizing redundant data is an important factor that keeps integrity and clarity of the database schema. A normalization process in the initial design phase gives you a firm database foundation.

The database schema found in figure 11 below is initially normalized in 3NF but as development moved on some 3NF rules were broken. This is something I believe is hard to avoid or at least it would have been extremely time consuming redesigning the schema at every update to it. What I did was to create a good foundation and then keep a close eye on redundant data. This worked in my case but might be questionable in larger projects.
3.6 Key generation
Creation of database keys is based on which type of donor the user is creating. In the database table **CD_donor** as presented in figure 11 above each key can either be of type mouse or human. They differ only by the two first letters in the key, the so-called key prefix. The key prefix for *Homo-Sapiens* is **HS** and for *Mus-Musculus** **MM**. After the key prefix is set the system checks how many keys exists prior to the current key. The system takes that number and increases it with one. The system then concatenates the key prefix with that value to form a complete key.

**Example key construction**
If a key in the system was of type "Mus-Musculus 100" the key would be written like "MM100". If a new key was to be generated the system would first determine which key prefix was going to be used, and then find the number of keys of this type and increase that value with one. The new key in this case would be "MM101".

Besides adding or changing data some users with administrative privileges can delete data. Deletes can be performed through a restricted set of options. A user can perform a clean sweep in the system by deleting a donor. This delete will transparently erase everything in the system associated with that donor. This method should be used with caution. The user also has the ability to delete a single cell batch at a time. Only one batch at a time can be deleted and only a tissue batch that does not have any child batches. This restriction is implemented by using the foreign key constraints *on delete restrict*.

Foreign keys enable the user to set restricted dependencies on table relationships.

MySQL supports the following restriction types on Delete or Update.
- **CASCADE**, if one of the datasets in the relationship is deleted/updated see to that the other dataset is treated the same way.
- **RESTRICT**, if a delete/update occurs on the restricted dataset in the relationship, restrict that action.
- **SET NULL**, if a delete/update occurs on a dataset in the relationship, set the dataset to null.
- **NO ACTION**, Do nothing.

To keep data consistency these tools are essential. Creating a relationship with foreign keys keeps the constraint controls at a very low level in your system. The database schema should always be transparent to a database user. The user should be able to delete/update or insert information into the database without knowing the relationship between tables. Foreign keys also help to avoid database schema destruction through schema alternations by the database administrator.
Example, foreign keys

Cell batches always except batches in passage zero form a relationship with other cell batches. If a user would delete a cell batch he could ruin the cell batch relationship structure and in that way traceability of the system. By using foreign keys a deletion of a cell batch would cause all cells in that relationship to be deleted. This is achieved through a so-called on delete cascade. This is transparent from a user perspective but necessary in a database environment if deletes should be possible and data integrity maintained.

3.7 Data locking

MyISAM and BDB tables uses table locks that effectively stop possible deadlocks and run on relatively small resources. In an environment where inserts and deletes are frequent table locks are inefficient. A complete table should not have to be locked for only a single insert since this would result in a lot of locking overhead in an environment with frequent inserts. InnoDB presents a more efficient solution with row-locking. This is a better solution in a RDBMS with a lot of frequent data inserts. Row-level locking never locks a complete table, only a single row or a combination rows at each insert. This enables several concurrent inserts in each table.

InnoDB is the only table format currently able to perform a row-level lock in MySQL. The row-level locking is performed by creating a lock index table were each row receives a shared or exclusive state. Row level locking only locks rows that are explicitly part of the SQL statement that locks the table.

Example 3
SELECT * FROM child WHERE id > 100 FOR UPDATE;

This statement would lock all rows were the attribute id is greater than 100. Other rows will be stored in a shared state. The row-locking method has one flaw, deadlocks. A deadlock occurs when two different inserts or updates wants to access each other’s row for an insert or update. The manipulations become dependant on each other and a dead lock might occur. InnoDB solves this problem by using a strategy that uses deadlock detection. If a deadlock is detected a rollback is performed and the transaction is redone.
3.8 Transactions

InnoDB is a transactional table-type that enables a developer to perform several insert/update/delete SQL statements in one transaction. You start a transaction by specifying start with a BEGIN statement. Perform any SQL statements you desire and then end the transaction with the COMMIT statement. If the transaction receives an error from any of the SQL statements in the transaction a ROLLBACK is performed. The user can also perform a ROLLBACK explicitly by using the ROLLBACK command at any point in a transaction. MySQL is by default configured to AUTOCOMMIT. In AUTOCOMMIT mode MySQL automatically performs a commit on every SQL statement directly after it has been executed. In this mode a transaction will never be longer than one query.

A transaction is always performed completely or not at all. Transactions were used in the tissue cultivation database through the Perl DBI transaction functions.

- Begin()
- Commit()
- Rollback()

These methods call the corresponding SQL functions and enables transactions in the current database connection. It is important to understand that each open database handle must receive its own transaction methods. These transaction methods have enabled complicated procedures to run in a single transaction. This transactional ability have proven useful at several occasions and helped to keep the database in a consistent state.

Example 4
When creating a tissue batch combined with a picture upload you first insert the batch data and then perform a picture upload. The upload routine performs a file type check and refuses the upload if the file type is not one of the supported picture types. This failure would leave you with a half complete tissue batch insertion and the database consistency would fail. Using transactions this failure will not occur. An erroneous file format will generate a rollback and a correct one will perform a commit on the complete insert.

3.9 Security & integrity

MySQL has an accompanying tool for table backup called mysqldump. Mysqldump creates backup data in a human readable format. When using mysqldump you lock all databases that are currently dumped. This limits the usage of mysqldump to applications with few concurrent users. In the latest version of MySQL a hot backup tool is present. This tool enables non locked dumping with MyISAM tables. Innobase Oy has a tool called InnoDBHotBackup that is able to create a real-time dump of your database without forcing a lock on any table in the database. This dump is performed inside a transaction and the dump becomes an exact replicate of the database state at the time of the transaction initialization. InnoDBHotBackup has a commercial license whereas mysqldump has a GPL license.

A simple example can be found below that describes how mysqldump dumps a database with drop tables added before every create table, it doesn’t stop at errors and the
dump process is encapsulated in a single transaction that guarantees data consistency. Database transactions are discussed further in chapter 3.8.

Example MySQLDump
```bash
mysqldump -add-drop-table -force -single-transaction db > /dev/st0
```

An alternative method of backing up the database is done through `tar` or other archive software where you archive the database at file system level. The MySQL database is saved in the MySQL data directory specified in the configuration file `my.cnf`. Each database has its own directory in the data directory were every file associated with the database is stored. Archiving the database directory can only be done when no users are logged on.

### 3.10 Recovery functionality

MySQL can be configured to automatically perform table specific recovery processes at startup. The InnoDB recovery protocol performs an automatic roll-forward on all present logs and then a rollback on all uncommitted transactions present at the time of the crash. The user can adjust the granularity of this process but not it’s content. There exist six different InnoDB recovery protocols levels. You specify which levels you want to use with the `innodb_force_recovery` switch in the MySQL configure file `my.cnf`.

1. (SRV_FORCE_IGNORE_CORRUPT) Let the server run even if it detects a corrupt page; try to make SELECT * FROM table jump over corrupt index records and pages, which helps in dumping tables;
2. (SRV_FORCE_NO_BACKGROUND) Prevent the main thread from running. If a crash would occur in purge, this prevents it;
3. (SRV_FORCE_NO_TRX_UNDO) Do not run transaction rollbacks after recovery;
4. (SRV_FORCE_NO_IBUF_MERGE) Prevent also insert buffer merge operations. If they would cause a crash, better not do them; do not calculate table statistics;
5. (SRV_FORCE_NO_UNDO_LOG_SCAN) Do not look at undo logs when starting the database: InnoDB will treat even incomplete transactions as committed;
6. (SRV_FORCE_NO_LOG_REDO) Do not do the log roll-forward in connection with recovery.

[MySQL manual ch. 16.9.1 Forcing Recovery] [3]

### 3.10.1 Backup woes

Database backup proved somewhat problematic. After restoring the database we realized that the backup data was corrupt. All data in `longblob` variable types had been corrupted which meant that all our images were destroyed. This problem seemed to originate from a combination of InnoDB tables and the 5.x.x series of MySQL. Both Linux and OpenBSD had the same problem. To solve this problem I created a Perl script that first dumps the complete database with `mysqldump` and then dumps the image data explicitly. The `longblob` table was dumped using the “SELECT INTO OUTFILE” SQL statement. Data Restoration is then performed through another Perl script that uses “LOAD DATA INFILE” to restore the `longblob` table. Both these scripts are present in the appendices (Ch. 0, 0). This is a temporary solution that will be dropped when the `mysqldump` problems are resolved.
Today this problem has been finally been resolved. The corrupt data problem originated in different charsets in the database and the mysqldump client. The problem wasn’t visible on normal characters because UTF-8 and Latin1 share a few characters but the special binary characters became corrupt. Running the mysqldump client in Latin1 solved the problems.

### 3.10.2 Recovery test

Database recovery is tested by dumping the MySQL data and then restoring it using the special method described in Ch 3.10.1. The restoration is conducted on another machine running the same version of MySQL. The backup data is taken from data on the production server. The backup data in this test is normally saved to a tape station.

<table>
<thead>
<tr>
<th>Date</th>
<th>Size</th>
<th>Restoration passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-05-04</td>
<td>240 Kb</td>
<td>OK</td>
</tr>
<tr>
<td>2004-05-18</td>
<td>412 Kb</td>
<td>OK</td>
</tr>
<tr>
<td>2004-06-01</td>
<td>446 Kb</td>
<td>OK</td>
</tr>
</tbody>
</table>

### 3.11 CPAN Modules

Comprehensive Perl Archive Network (CPAN) is a repository of Perl modules freely available on the internet. The repository is maintained by hundreds of voluntary developers from all over the world. That is a one of the big advantages with Perl. [4]

#### Graphics draw – GD

GD is a Perl interface for Thomas Boutell's GD graphics library (GD, graphics draw). GD allows you to create color images using a large set of tools and also to save the image date in the PNG file format. *GD::ThumbNail* uses image:GD library to resize an image to any specified size. *GD::Graph* is a perl5 module to create charts using the GD module. GD::Graph contains several different methods for creating graphs. The most common types like pie chart, bar chart and histogram are available.

#### Spreadsheet::WriteExcel

The Spreadsheet::WriteExcel module can be used to create a cross-platform Excel binary file. Multiple worksheets can be added to a workbook and formatting can be applied to cells. Text, numbers, formulas, hyperlinks and images can be written to the cells.
Common gateway interface (CGI)

The CGI library uses Perl5 objects to make it easy to create Web fill-out forms and parse their contents. This package defines CGI objects, entities that contain the values of the current query string and other state variables. Using a CGI object's methods, you can examine keywords and parameters passed to your script, and create forms whose initial values are taken from the current query (thereby preserving state information). The module provides shortcut functions that produce correct HTML syntax, reducing typing and coding errors. It also provides functionality for some of the more advanced features of CGI scripting, including support for file uploads, cookies, cascading style sheets, server push, and frames.

3.12 Graphical user interface

The GUI's are built using Perl, JavaScript and VBScript that are familiar and stable languages combined with extensive documentation and in-depth NeuroNova AB knowledge. Perl’s key advantages is its ability to compile data before it is presented on the web and the wide range of modules freely available on CPAN[4]. By using a web interface as GUI the client can connect to the system from any type of system supporting web browsing. (Worth noting is that some client-side scripting (VBScripts, JavaScript) techniques used in this system are not supported by all browser). Through the GUI the user has the ability to search, add and update data. Each of these three different methods is for clarity distinguished by a color. Green for search, orange for add and blue for update. Each step in a user session is colored in the correct way to help the user navigate through the software. Navigating with the help of colors makes the system easier to understand. Cascading style sheets were frequently used from both a design and usage point of view. A style sheet is a plain text file with special tags describing the look of certain web objects. The style sheet pages are linked to Perl scripts and called locally or globally for each web object. A style sheet can for instance define font face, table or hyperlink appearances. Every web object created in a Perl page can call a style sheet. Changes to the design can later be done by only changing a single line in the style sheet object. The style sheet helps to create standardized GUI interfaces. Each batch of cells can exist in one of four different states; frozen, experiment, expansion and dead. For each one of these states an icon was created.

Icons

<table>
<thead>
<tr>
<th>Frozen</th>
<th>Experiment</th>
<th>Expansion</th>
<th>Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td><img src="image2.png" alt="Icon" /></td>
<td><img src="image3.png" alt="Icon" /></td>
<td><img src="image4.png" alt="Icon" /></td>
</tr>
</tbody>
</table>

The icons represent the activity they symbolize. The iceberg for frozen tissue, the hand for experiment tissue, the ok text for expanding tissue and finally the skull for dead tissue.

The input logic and transaction control is handled through Perl. The GUI input data is processed by the software using HTML forms. The form data is sent unencrypted through POST or GET requests to the server. Data is sent in a binary form when pictures are sent in forms.
A picture or any other binary data can be sent through HTML forms by using a multipart type of HTML form header. Before the picture data is added to the database a mime and picture type check is performed. Only JPEG or PNG picture types are allowed in the software. GIF is not supported because of copyright restrictions. If the picture is valid the data is parsed into a thumbnail sized image (50x50 pixels). This is performed by the GD::ThumbNail module. The original picture is there after added to the database. The data integrity control is maintained by a Perl algorithm, which if an erroneous picture file is found performs a rollback on the inserted picture data. Input errors are reported to the user through JavaScript pop-up windows. The pop-up window describes the error and offers the user to restart the procedure if possible or otherwise end it. In the event of a fatal error a descriptive message is printed directly to screen before the software breaks.

### 3.13 System reports

The ability to create reports from data stored in the database is an important feature not only for the primary users but for all users of the system. The scientist can at any passage starting from passage one create reports based on data from the tissue isolation to the present state. The software can create four different report types.

**Excel spreadsheet** report as seen in figure Excel report. The spreadsheet contains all data gathered on the donor, tissue and isolation. Additionally, the total number of expanding cells in each passage is reported as well as the total number cells in any of the following states dead, experiment or frozen. The spreadsheet is generated through a Perl module called Spreadsheet::WriteExcel available at CPAN[4].

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CellDatabase growth information</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td>Donor/Tissue Information</td>
<td>Isolation Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3</strong></td>
<td>Species: Mus Musculus</td>
<td>Mobility: 44</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>Strain: C57Bl/6J</td>
<td>Time: 2.00 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td>Donor: 21 Female 10 Male</td>
<td>No Cells: 2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>6</strong></td>
<td>Age: 4.00 Hours</td>
<td>Protocol: P3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>7</strong></td>
<td>Postmortem: Yes, 6hr 2200 Hours</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>8</strong></td>
<td>Disease</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>9</strong></td>
<td>Virulent EBV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>10</strong></td>
<td>Volume: 19.58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>11</strong></td>
<td>Structure: Hippocampus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>12</strong></td>
<td>Acronym: 203 JEC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>13</strong></td>
<td>Protocol: P102</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>14</strong></td>
<td>Cell MR201HT1_pZ4 growth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>15</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>16</strong></td>
<td>Passages</td>
<td>Expanding No Cells</td>
<td>Dead/Handout/Frozen Cells</td>
<td></td>
</tr>
<tr>
<td><strong>17</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>18</strong></td>
<td>P1</td>
<td>9909</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td><strong>19</strong></td>
<td>P2</td>
<td>10300</td>
<td>01</td>
<td>101</td>
</tr>
<tr>
<td><strong>20</strong></td>
<td>P3</td>
<td>10000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><strong>21</strong></td>
<td>P4</td>
<td>10073</td>
<td>4</td>
<td>401</td>
</tr>
<tr>
<td><strong>22</strong></td>
<td>P5</td>
<td>0</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td><strong>23</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>24</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>25</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>26</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**figure 15, Excel report**
**Tree relationship** report as seen in figure tree-like report shows each batch of cells relationship in a tree like structure. It also shows the current cell batch status. The tree structure algorithm was developed using the GD module.

![Tree relationship diagram](image)

**Figure 16, An example of a tree-like report**

**Growthchart** as seen in figure growthchart is another report tool that report the total number of cells in each passage. The report is presented in a barchart were each bar in the barchart consists of one or two colors; blue for expanding cells and pink for dead, frozen or finished cells. The bar chart is generated through a Perl module called GD::Graph::bars available at CPAN[4].

![Growthchart](image)

**figure 17, Tissue cultivation growth chart**
**HTML report** as seen in “figure HTML report” reports each cell as a parent with all its children attached to it. Each parent and child cell is presented with its corresponding number of cells. The total number of cells in each parent’s child batches is also reported. These report tools give the scientist a way to communicate with other users and enables an overview of the complete process from the first passage to the last.

![Figure 18](image)

**3.14 Authorization & User access**

The authentication capabilities of Perl and the lack of grouping capabilities in the MySQL protocol is a problem. To overcome these problems a security protocol was created. The protocol specifies that two different user groups are sufficient for the software. The first user group **normal** was given the ability to **INSERT, SELECT, UPDATE, DELETE** on the tissue database and the privileges **SELECT** on the MySQL user database. The second group **super** inherited all of the **normal** groups’ privileges but was also given **INSERT, UPDATE and DELETE** on the MySQL user database. That enabled the **super** group to remove, add or grant/revoke super user privileges on users of the system. The system is able to keep track of user

![Figure 13](image)
privileges, authentication information and group membership through a custom-made security protocol. The user data uses the MySQL internal security system as a base protocol for all user access. The custom-made security protocol is a mix of Perl and MySQL internal security. The logon procedure starts at the tissue cultivation login page where the user is asked to provide a username and password. The username and password are passed on through a form and verified against the MySQL user database by a select statement. If the user is present in the MySQL user database a cookie is written. The cookie will include the username and password that was supplied at the login page both encrypted with MD5. The cookie information is later used whenever a connection to the database is established. Every time a page loads an authentication control is performed that checks whether a valid cookie is written. If a valid cookie is not found the user is redirected back to the login page and asked to provide a valid username and password. The time to live (TTL) of the cookie is set to 10 hours. This might seem like an excessive TTL but the NeuroNova AB workflow documentation requires it and on trusted workstations this is applicable. The software is able to recognize super and normal group users through a control method written in Perl. To be able to use his privileges, a super user has access to the administrative part of the software. In the administration part shown in figure 13 above the super user can add, delete, change password or grant/revoke super privileges to other users in the system. All of these features are presented in a web based form that doesn’t require an in depth knowledge of SQL or any database structures.

figure 14, System security overview
3.15 Software engineering

Software engineering is always conducted in an iterative development style. No requirement specification can include everything the end user expects of a system. What I as a software engineer strived to accomplish was to maximize the initial iterative development period to as early as possible be able to capture the end-user needs. In Paul J. Morris article [27] a common development progress life cycle is presented which is shown in figure 19. It describes the development process in a similar project.

[Software engineering’s example]

1. Plan (planning, analysis, requirements collection)
2. Design (Conceptual database design, leading to information model, physical database design, user interface design)

   In my case these two steps lead to a prototype that were iterated before step three was begun.

3. Implement (Database implementation, user interface implementation)
4. Load legacy data (Clean legacy data, transform legacy data, load legacy data)
5. Test (test implementation)
6. Put the database into production use and perform operational maintenance.
7. Repeat this cycle

Extracted from:
Relational Database Design and Implementation for Biodiversity Informatics by Paul J. Morris

What I did to in this project was to isolate and prolong the first two steps of the development process to be able to shorten the last five steps. I did this by developing a prototype of the system. The prototype was used in a very early phase of development. This helped me to understand the user’s needs and also to create the system the user required in shorter time. The further back in the development chain you need to iterate the longer the development period will be needed. Prototyping lessens the need to go back by splitting the development workflow in two. Developing a prototype for engineering purposes can be a tiresome startup effort but creates a mutual understanding between the end-user and developer. Letting the end-user see a product before it has been built gives the user an opportunity to influence the development process at an early state. The user has an opportunity to express ideas and suggestions from a perspective based on the development teams limitations.

Creating a dummy GUI early in the project to let all participants in the development phase express an opinion speeds up the later development phase of the project. Letting the end-user and other non-technical staff use the prototype in their own pace and situation decreases the amount of time spent in an iterative work process. A prototype also creates anticipation and interest for the project and this develops a more focused end-user group. Domain knowledge within the group will be shared when a visualization of the problem is presented.
The tissue cultivation group has been part of the development process from the start. This primary user group has been given a quick introduction and position in the development process that has shortened the iterative development work process cycle. The user group has been briefed of work progress approximately once a month. On each of these briefings the software has been reviewed and evaluated. This has helped the development process to move closer towards the user needs even before the product was fully developed. The user evaluation process has saved a lot of development time. This evaluation method has incorporated the work performed by the primary user groups into the project. To keep the primary user group satisfied is extra important. Their opinions will form foundation for opinions for the remaining user groups. If the first user group produces successful results through the software it will help related user groups to perform more precise experiments.

Finding a group of users that has time to review and test a system in its construction phase is important. This provides software that meets the end users user needs instead of developer defined user needs. Requirement specifications and other pre-construction documentation can be kept at a high level and user input will guide the development process. The requirement specification should only point in the correct direction and limit the project scope and duration.

Another software engineering factor to keep in mind is how development in a shared environment will be performed. Here it is important to keep the software in a consistent state. CVS is an open-source solution that enables developers to share development files in a central source code repository. Developers uses local file trees under development but can at any chosen time either update his own local source tree from the central repository or commit his recent changes to the repository. At a commit the developer’s file is compared with original file in the central repository. If no changes has been made to the file between the current commit and the last commit the developer is allowed to commit his changes to the repository. If the local file that the developer is about to commit has changed he is notified of this and can choose to add his new changes or not. The CVS repository method is well documented and used by large development groups all over the world. CVS will make sure that no code is ever erased without notifying the developer first. A complete backtrack option on all software changes is also possible.

When working in a software environment that combines languages with different execution time like JavaScript and Perl you can easily come across timing issues. A JavaScript bundled into a Perl module will not execute at the same time as the Perl module is executed. The Perl module is executed server side while the JavaScript is executed client side. This makes the placement of such a module function important. Placing a JavaScript function that is going to be called later in the program forces the placement to occur earlier than the function call or the software will crash. This is not the case with Perl modules were placement of the function call can be done in any order. Perl scripts are executed on the web server before the data is presented to the client in HTML format. Sending parameters between JavaScript and Perl functions require caution and should as a rule be avoided as often as possible. When working in a mixed environment it is wise to always use lowercase letters. In my replication experiments I have used a MySQL database running on Windows and with that tried replicating a database running on Linux. I was at first unable to establish a connection between the two. But after some debugging I finally figured out that the connection errors occurred because Windows was unable to call the correct database on
Linux. I saw that Windows automatically calls all databases in lowercase letters. Linux (or UNIX) distinguishes between mixed capital and lowercase letter database names. After rewriting the database with only lowercase letters I solved the replication problems.

A developer should always avoid mixing capital and lower-case letters. In this way you will not constrain yourself to one architecture. The replication experiment step-by-step guide is included in the appendices.
4 Usage example

To better understand system usage I have created a short system tutorial in the form of usage examples. This tutorial shows four different interfaces that may contain several different functions from a user perspective. I have tried to cover the most common usage of the system.

- Add new cells,
- Keep track of cells (through searches)
- Handing them out to other scientists or cultivators (cell batch owner changes).

The private part in the general section described below gives the user quick overview to these more common functions. More obscure functions such as updates or deletes on cell data is not covered here because they are very rarely performed.

System tutorial sections
1. General
2. Search
3. Add
4. Reports
4.1 General

First of all each user has to authenticate himself. The user is then presented to the main page. The main page lets the user do a simple search and also presents a small private section to the user. The user can also navigate to other parts of the system via the horizontal top menu. The simple search has four options where you either search for a donor or a cell batch. The search keyword can either be a absolute or partial key name. The private section presents the last 20 cell batches this user has added and if any tissue has been given to the logged on user. This is shown in the “you have X awaiting handouts, click here” section were X shows the number of new cells. If a cell is waiting for the user he must pick it up. This procedure is shown in section Add 4.2.1
4.2 Search

A user can do either a basic search as shown in the figure main page above. If this search method doesn’t generate enough precision the user can choose to use the advance search option. The advance search interface is shown in figure advance search. The advance search lets the user specify the following parameters in their search query.

- Type of donor
- In what passage the batch is found in
- Sex
- Creator (Owner)
- Infection
- Frozen
- Tissue type
- Free text search
The result from any type of search (basic/advance) is presented in the form shown in *figure search results*. Both these right hand pictures are presented in larger versions in the appendix.

4.2.1

4.3 Add

This interface is divided into three areas that are separated by black boxes as shown in the figure 23. The user can manipulate information in one of the following ways.

- Add a new cell
- Receive a handout
- Add new batches of cells
  - complete passage, add all batches to a passage at once.
  - single batch of cells, add one cell batch at a time to a passage.

When clicking the “add new cell” button the user is presented to an interface where he can add donor, tissue and isolation information. This is the first step in any tissue cultivation process. The second option “receive a handout” directs the user to a page where all currently available handouts for him are presented. The user can choose any of these hand-outs and do a pick-up. The pick-up notifies the system that the handout has been both given and received.
and that the owner change is now complete. The third part of the add menu is used when a user wants to add one or several cell batches to the system.

![Image of NeuroNova interface]

**figure 23, The add new page**

The complete or partial key name of the cell batch that the cultivated batch of cells was expanded (cultivated) from in the text field found next to the single batch button. If the user used cell batch MM204T1I1_P3.4 to expand the current cell batch he could either type the complete key name or a part of it. After the user typed the key name and pressed the single batch button the system would search for keys matching the query and present them to the user in an interface similar to the one shown in *figure search results*. The user would actively choose the cell batch that the current cells originated from and he would then be presented to an add interface for a single cell. After filling out the text fields on that page the cell batch would have been added to the system.

The procedure above would be almost identical if the user would like to add several cell batches at one time with the difference that the user would choose the *complete passage* button and text field.

### 4.4 Reports

The tissue cultivation system also has ways to generate reports. The most common report is the HTML report that is shown below in *figure info page*. This page shows all information that has been added to a single cell batch and information on its donor, tissue and isolation process. The user can also see how many cell batches that exist in the same passage as the current cell batch. The report functionality has as mentioned before not been focused on. It is therefore not as well developed as it should be. In my opinion this is otherwise a very important feature in a data warehouse system.
figure 24, the information page
5 SUMMARY

The tissue cultivation software is now used at NeuroNova AB and all features defined in the requirement specification have been completed. Both scientists and other staff at NeuroNova AB have expressed satisfaction over the software. Besides the features covered in the requirement specification a few additional features have been added.

- Report-tools
- User administration
- Reliability solutions

The report functions are so far simple demonstrations on how data can be used in the software but they still cover basic demands from the users. User administration is conducted through the GUI. The graphical administration setup was appreciated and will in the forthcoming months be expanded to include a professional LDAP user directory solution. Reliability has been tested and controlled through backup, replication and the MySQL test suite. A fully functional replication setup has been developed for the database. The replication is not currently used but a step-by-step setup-guide is supplied in this report and the database is adjusted to allow replication whenever the demand occurs. All in all it has been an instructive project with challenges in a wide range of fields. The final result became something that everybody involved in the project was happy with and the software is today used in the everyday work at NeuroNova AB.

5.1 System database results

The tissue cultivation software will, as soon as MySQL 5.0.0 reaches production status, move to the central bioinformatics server, the bioinformatics server will by that time be upgraded with the production version of MySQL. It is estimated that MySQL will reach production status in about six to ten months (sometime in early 2005) by which time the tissue cultivation database should have matured into stable and reliable software. Most bugs and user suggestions in the tissue cultivation database should by then have been fixed. A map describing the system in general can be found below (figure 9, A system map). The figure describes the system architecture at a high level. Each green box represents a unique system page. A system page is written in CGI and each page includes some or all of the various tools and techniques described in the report e.g. RDBMS access, security checks, interface data and CGI logic. A system page is viewable by the system user through the Internet explorer web browser. A system page is not a RDBMS, the database system is found at a lower description level. All data found in the system pages are fetched from the lower database system level through specially developed CGI modules.

5.2 Future enhancements

One-login authentication. The network login process at NeuroNova AB is today performed with Microsoft’s Active Directory. Active Directory is a LDAP compatible highly advanced user directory service. This directory service can be used to centralize the login procedure for all in-house database projects at NeuroNova AB. This would decrease user administration workload. By using OpenLDAP, an open-source LDAP protocol, a
connectivity program can easily be created. Scalability and security would increase and no additional software licenses would be needed.

5.3 An object relational approach

In retrospect a different database schema structure could have been constructed. Object relational database management systems (ORDBMS) are interesting complements to standard relational database systems. The ORDBMS resembles more the standard software engineering development way (e.g. Java’s object oriented style) with its object way of thinking. The object relational structure was not possible with the chosen RDBMS (MySQL) but with for example Oracles system this idea could have been realized.

The current database schema can in any case be adopted into an object-oriented schema. The following UML database schema found below forms an idea on how such a system would look like. The schema is a rough sketch but visualizes some of the advantages found in object relational database systems, e.g. the possibility to use inheritance.

5.3.1 Adopting the schema

An isolated batch of cells consists of four different sub-parts. The donor is the only tissue part not in a many to one relationship with the isolated batch of cells. Besides the donor a batch of cells can have several isolations, tissues and belong to one or several different passages.
5.3.2 ORDBMS conclusion

On today’s market there exist specially developed database structures for common bioinformatics data structures available with the Oracle ORDBMS. The data structures enable indexing over complex data structures among other interesting things. In this project none of these pre-defined structures were interesting but in other similar bioinformatics data capture projects these special structures could definitely serve a purpose [28].

The schema above can be seen as an alternative database system scenario and briefly summarizes some ideas and limitations (e.g. inheritance) that surfaced during the development of the system found in this report. This current system will never be converted into an object oriented database system schema but future systems might.
6 Acknowledgements

I would like to thank my company supervisor Karl Nyberg and Rikard Erlandsson. I would also like to thank my examiner Kjell Orsborn and last but not least all the friendly people at NeuroNova AB.
7 References

[19] community press
[20] Larry Wall, Tom Christiansen & Jon Orwant, 2000, Programming Perl, O’Reilly
[21] Rafe Colburn, 2000, Using SQL, Que
[22] Alligator Descartes & Tim Bunce, 2000, Programming the Perl DBI, O’Reilly
[23] Kevin Kline, Daniel Kline, 2001, SQL in a nutshell, O’Reilly
[33] Eva Toller, Tore Risch, 2000: An Object-Relational Model for Musical Data, Research
[34] report
[37] Bruce Blackwell, Siva Ravada, 2003 Oracle’s technology for bioinformatics and future
directions, Australian Computer Society, Inc.
[38] Sudeshna Adak, Vishal S Batra, Deo N Bhardwaj, PV Kamesam, Pankaj Kankar, Manish
[39] P Kurhekar, Biplav Srivastava, 2002, A System for Knowledge Management in
[40] Bioinformatics, Conference on Information and Knowledge Management
[42] 2002, Labrat LIMS: An extensible framework for developing Laboratory information
8 Abbreviations and definitions

VPN, *Virtual Private Network*. VPN is an encrypted connection method that enables people to establish a connection to a private network from the outside.

IPSEC is a common encryption method. [1]

LIMS, *Laboratory Information Management System*. Systems in the Life Science sphere often refer to their systems for EDC, EDM and automatic control as LIMS. These are the systems from which bioinformaticians collect their raw data for further analysis.

EDC, Electronic data capture. The capture of data from complex activities into electronic format to improve storage, mining and retrieval

BSD & GNU LICENSES, *Berkley and the Free Software Foundation licenses*. Open-source licenses. Both are used extensively in open-source software.

TTL, *Time To Live*. A common abbreviation used in Internet traffic. Sets the maximum lifetime of an IP-package or a cookie.

CVS, * Concurrent Versioning System*. CVS is a development tool. The system applies a consistency control to source code in shared development environments.


LAMP, An abbreviation for Linux, Apache, MySQL and Perl. A common toolkit used when creating web-enabled systems.

Open-source, A software licensing model originating from the Free software alliance.

CNS, Central Nervous System. The CNS includes the brain and the spinal cord.

Neurogenesis, The formation of new neurons.

Differentiation, The specialization of multi-potent cells, e.g. the transition of neural progenitor cells into glia or neurons.

Progenitor cell, an immature cell with the possibility to differentiate into many different cell types

Proliferative, Something with ability to induce growth
Neural stem cell, A basal cell type with the ability to divide asymmetrically i.e. to produce one copy of itself and one progenitor cells. The stem cells are the origin of all new CNS cells.

Neuron, A cell that conduct nerve impulses.

Cell batch, A collection of cells.

Passage, Term used in tissue cultivation to describe in what growth cycle a cultivation process belongs to.
Appendices

Appendix 1 - pictures

Tissue database schema
### SEARCH RESULTS

<table>
<thead>
<tr>
<th>Date</th>
<th>BatchID</th>
<th>Creator</th>
<th>Pass</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-02-25</td>
<td>MMM204711_P0_1</td>
<td>jonas.ladenfor</td>
<td>0</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P1_1</td>
<td>jonas.ladenfor</td>
<td>1</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P1_2</td>
<td>jonas.ladenfor</td>
<td>1</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P1_3</td>
<td>jonas.ladenfor</td>
<td>1</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P1_4</td>
<td>jonas.ladenfor</td>
<td>2</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P2_1</td>
<td>jonas.ladenfor</td>
<td>2</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P2_2</td>
<td>jonas.ladenfor</td>
<td>2</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P2_3</td>
<td>jonas.ladenfor</td>
<td>2</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P2_4</td>
<td>jonas.ladenfor</td>
<td>2</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P3_1</td>
<td>jonas.ladenfor</td>
<td>3</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P3_2</td>
<td>jonas.ladenfor</td>
<td>3</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P3_3</td>
<td>jonas.ladenfor</td>
<td>3</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P3_4</td>
<td>jonas.ladenfor</td>
<td>3</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P4_1</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P4_2</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P4_3</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P4_4</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-06-03</td>
<td>MMM204711_P4_5</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-08-23</td>
<td>MMM204711_P4_6</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-09-03</td>
<td>MMM204711_P4_7</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
<tr>
<td>2004-09-03</td>
<td>MMM204711_P4_8</td>
<td>jonas.ladenfor</td>
<td>4</td>
<td>[DEMO]</td>
</tr>
</tbody>
</table>

Next page >>
Advance search page
Appendix 2 – Replication guide

Replication Guide

This is a small guide on how to set up a replication environment between one client(slave) and one server(master). Circular dependencies are allowed just pretend your slave has become your master and your master your slave. Perform each step in the order they are presented and you should be alright.

You **cannot** replicate MySQL databases with mixed capital and lower-case letters in a combined Windows and Linux architecture.

If you are going to create a circular dependency you must include this line in both your my.ini.

```plaintext
Log-slave-updates
```

Create a replication user on the master

```plaintext
GRANT SELECT, PROCESS, FILE, SUPER, REPLICATION CLIENT, REPLICATION SLAVE, RELOAD ON *.* TO repl@'%' IDENTIFIED BY 'repl_pass';
```

Put this in your master servers MY.CNF

```plaintext
Log-bin
Server-id = 1
```

Install a appropriate MySQL server on slave.

<table>
<thead>
<tr>
<th>Slave</th>
<th>Master</th>
<th>Master</th>
<th>Master</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.23.33 and up</td>
<td>yes</td>
<td>4.0.3 and up or any 4.1.x</td>
<td>no</td>
</tr>
<tr>
<td>4.0.3 and up</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>5.0.0</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Create a MY.INI on your slave similar to this
Duplicate replication information on master to slave. This can be done in a couple of different ways.

execute LOAD DATA FROM MASTER on the slave, which does a “hot” backup of the replication data. Only usable with MyISM tables.

Lock tables or stop database. Package database structure (TAR or GZIP) and extract in the slaves data dir.

Start slave

    mysql> start slave;

Unlock possible table locks on master.

Finished…

To debug use “Show slave status”
Appendix 3 – Cell database restore script

DB restorescript

{-#START-CELLDB-RESTORE-ALGORITHM------------------

my $dumpDir = '/home/jonas/backup/';
my $mysqlPath = '/usr/local/bin/mysql/bin/';
my $databaseName = 'celldb';
my $bakFile = '';
my $picFile = '';

if ($#ARGV ne -1){
    $bakFile = $ARGV[0];
    $picFile = $ARGV[0].'-cd_picture.txt';
} else{
    print "usage: restorescript.pl filename\n";
    exit 0;
}

my $loadInto = "echo \"set foreign_key_checks=0;load data infile \\
".". $dumpDir.$picFile." replace into table celldb.cd_picture\" |  \\
".".$mysqlPath

print "\nRestorescript.pl\n\nRemember that you need a file named dumpfile- 
\ncd_picture.txt. This file is automatically generated by dumpscript.p

print " + Restoring table data\n";
system($mysqlPath."mysql < ". $dumpDir.$bakFile);
print " + Restoring picture data\n";
system($loadInto);

{-#END-CELLDB-RESTORE-ALGORITHM-------------------

DB dumpscript

{-#CELLDB-BACKUP-ALGORITHM------------------------

my $time = localtime;
my $dumpDir = '/home/jonas/backup/';
my $mysqlPath = '/usr/local/mysql/bin/';
my $databaseName = 'celldb';
my $bakFile = '';
my $picFile = '';

if ($#ARGV ne -1){
    $bakFile = $ARGV[0];
    $picFile = $ARGV[0].'-cd_picture.txt';
} else{
    print "usage: dumpscript filename\n";
    exit 0;
system("echo 'use ".$databaseName.;' >> ".$bakFile);
system("echo 'SET FOREIGN_KEY_CHECKS=0;' >> ".$bakFile); # No foreign key checks while we are restoring data

# Dump the database!
system("$mysqlPath."mysqldump --add-drop-table --force --single-transaction ".$databaseName." >> ".$dumpDir.$bakFile);

# Fix for binary blob dumping!
system("rm -rf ".$dumpDir.$picFile);
system("echo \"SELECT * INTO OUTFILE "."$dumpDir.$picFile." FROM cd_picture\" | ".$mysqlPath."mysql celldb");
system("echo 'SET FOREIGN_KEY_CHECKS=1;' >> ".$bakFile); # Enable foreign key checks after we've add all restored data.

#-END-CELLDB-BACKUP-ALGORITHM----------------
Appendix 4 – Database create script

Database schema

CREATE TABLE cd_user (  
  user_name_id VARCHAR(150) NOT NULL,  
  user_lastname VARCHAR(150) NOT NULL,  
  mail VARCHAR(200) NULL, 
  lastlogin DATE NULL,  
  title VARCHAR(100) NULL,  
  PRIMARY KEY(user_name_id, user_lastname)  
) 
TYPE=InnoDB;

CREATE TABLE cd_freeze (  
  freeze_id VARCHAR(10) NOT NULL, 
  location VARCHAR(150) NULL, 
  size VARCHAR(150) NULL,  
  PRIMARY KEY(freeze_id), 
  INDEX freeze_id(freeze_id)  
) 
TYPE=InnoDB;

CREATE TABLE cd_protocol (  
  protocol_id VARCHAR(10) NOT NULL, 
  description VARCHAR(255) NULL, 
  part CHAR(2) NULL DEFAULT B, 
  PRIMARY KEY(protocol_id), 
  INDEX protocol_id(protocol_id)  
) 
TYPE=InnoDB;

CREATE TABLE cd_structure (  
  structure_id VARCHAR(100) NOT NULL,  
  PRIMARY KEY(structure_id)  
) 
TYPE=InnoDB;

CREATE TABLE cd_flask (  
  flask_id INTEGER(11) NOT NULL, 
  name VARCHAR(100) NOT NULL, 
  size VARCHAR(150) NOT NULL, 
  coating VARCHAR(150) NULL,  
  PRIMARY KEY(flask_id)  
) 
TYPE=InnoDB;

CREATE TABLE cd_vendor (  
  vendor_id VARCHAR(150) NOT NULL,  
  PRIMARY KEY(vendor_id)  
) 
TYPE=InnoDB;

CREATE TABLE cd_cond (  
  cond_id VARCHAR(100) NOT NULL,  
  PRIMARY KEY(cond_id)  
) 
TYPE=InnoDB;

CREATE TABLE cd_donor (  
  donor_id VARCHAR(10) NOT NULL,  
  date DATE NULL,  
  pmtime INTEGER(11) NULL, 
)
CREATE TABLE cd_donor (
    donor_id VARCHAR(150) NOT NULL,
    age BIGINT(20) NULL,
    comments LONGTEXT NULL,
    species VARCHAR(200) NULL,
    prefrozen ENUM('Yes', 'No') NULL DEFAULT No,
    strain VARCHAR(100) NULL,
    vendor VARCHAR(200) NULL,
    creator VARCHAR(100) NULL,
    bdate DATE NULL,
    orgin VARCHAR(150) NULL,
    cond VARCHAR(150) NULL,
    PRIMARY KEY(donor_id)
) TYPE=InnoDB;

CREATE TABLE cd_disease (
    disease_id VARCHAR(150) NOT NULL,
    PRIMARY KEY(disease_id)
) TYPE=InnoDB;

CREATE TABLE cd_sex (
    sex_nr_id VARCHAR(80) NOT NULL,
    sex_sex_id VARCHAR(80) NOT NULL,
    sex_donor_id VARCHAR(10) NOT NULL,
    PRIMARY KEY(sex_nr_id, sex_sex_id, sex_donor_id),
    INDEX sex_donor_id(sex_donor_id)
) TYPE=InnoDB;

CREATE TABLE cd_tests (
    tests_name_id VARCHAR(80) NOT NULL,
    tests_found_id VARCHAR(80) NOT NULL DEFAULT Negative,
    tests_donor_id VARCHAR(10) NOT NULL,
    PRIMARY KEY(tests_name_id, tests_found_id, tests_donor_id),
    INDEX tests_donor_id(tests_donor_id)
) TYPE=InnoDB;

CREATE TABLE cd_tissue (
    tissue_id VARCHAR(10) NOT NULL,
    date DATE NULL,
    amount FLOAT NULL,
    structure VARCHAR(200) NULL DEFAULT -,
    volume FLOAT NULL,
    creator VARCHAR(100) NULL DEFAULT NotEntered,
    t_protocol_id_fk VARCHAR(10) NULL,
    t_freeze_id_fk VARCHAR(10) NULL,
    t_donor_id_fk VARCHAR(10) NULL,
    time INTEGER(11) NULL,
    PRIMARY KEY(tissue_id),
    INDEX structure(structure),
    INDEX t_protocol_id_fk(t_protocol_id_fk),
    INDEX t_freeze_id_fk(t_freeze_id_fk),
    INDEX t_donor_id_fk(t_donor_id_fk),
    INDEX tissue_id(tissue_id)
) TYPE=InnoDB;

CREATE TABLE cd_isolation (
    isolation_id VARCHAR(10) NOT NULL,
    date DATE NULL,
    nocells INTEGER(11) NULL,
    time INTEGER(11) NULL,
    viability INTEGER(11) NULL,
CREATE TABLE isolation (  isolation_id INTEGER(11) NOT NULL,  i_protocol_id_fk VARCHAR(10) NULL,  i_tissue_id_fk VARCHAR(10) NULL,  volume FLOAT NULL,  PRIMARY KEY(isolation_id),  INDEX i_protocol_id_fk(i_protocol_id_fk),  INDEX i_tissue_id_fk(i_tissue_id_fk),  INDEX isolation_id(isolation_id) )  TYPE=InnoDB;

CREATE TABLE cd_batch (  batch_id VARCHAR(80) NOT NULL,  date DATE NULL,  nocells INTEGER(11) NULL,  growth ENUM('Suspension','Adherent','Unknown') NULL DEFAULT Suspension,  creator VARCHAR(100) NULL DEFAULT Nobody,  enzyme VARCHAR(200) NULL DEFAULT -,  seeddensity FLOAT NULL,  comments LONGTEXT NULL,  commited ENUM('Yes','No') NULL DEFAULT No,  dead ENUM('Yes','No') NULL DEFAULT No,  b_freeze_id_fk VARCHAR(10) NULL,  b_flask_id_fk INTEGER(11) NULL,  b_protocol_id_fk VARCHAR(10) NULL,  b_isolation_id_fk VARCHAR(10) NULL,  PRIMARY KEY(batch_id),  INDEX b_freeze_id_fk(b_freeze_id_fk),  INDEX b_flask_id_fk(b_flask_id_fk),  INDEX b_protocol_id_fk(b_protocol_id_fk),  INDEX b_isolation_id_fk(b_isolation_id_fk),  INDEX batch_id(batch_id) )  TYPE=InnoDB;

CREATE TABLE cd_template (  template_id VARCHAR(100) NOT NULL,  donid VARCHAR(10) NULL,  isoid VARCHAR(10) NULL,  tisid VARCHAR(10) NULL,  batid VARCHAR(80) NULL,  PRIMARY KEY(template_id),  INDEX donid(donid),  INDEX isoid(isoid),  INDEX tisid(tisid),  INDEX batid(batid),  INDEX template_id(template_id) )  TYPE=InnoDB;

CREATE TABLE cd_parent (  id INTEGER(11) NOT NULL,  child_id VARCHAR(80) NULL,  parent_id VARCHAR(80) NULL,  PRIMARY KEY(id),  INDEX parent_id(parent_id),  INDEX child_id(child_id) )  TYPE=InnoDB;

CREATE TABLE cd_picture (  picture_id VARCHAR(100) NOT NULL,  p_batch_id VARCHAR(80) NOT NULL,  data LONGBLOB NULL,
CREATE TABLE cd_handout (  handout_id INTEGER(11) NOT NULL,  pdate DATE NULL,  recvname VARCHAR(200) NOT NULL,  pboknr VARCHAR(100) NULL,  pusage TEXT NULL,  h_batch_id VARCHAR(80) NULL,  h_isolation_id VARCHAR(10) NULL,  h_tissue_id VARCHAR(10) NULL,  PRIMARY KEY(handout_id),  INDEX h_batch_id(h_batch_id),  INDEX h_isolation_id(h_isolation_id),  INDEX h_tissue_id(h_tissue_id),  INDEX recvname(recvname) ) TYPE=InnoDB;

Constraint schema
ALTER TABLE cd_isolation ADD FOREIGN KEY (i_protocol_id_fk) REFERENCES cd_protocol(protocol_id) ON DELETE RESTRICT;
ALTER TABLE cd_isolation ADD FOREIGN KEY (i_tissue_id_fk) REFERENCES cd_tissue(tissue_id) ON DELETE CASCADE;
ALTER TABLE cd_sex ADD FOREIGN KEY (sex_donor_id) REFERENCES cd_donor(donor_id) ON DELETE CASCADE;
ALTER TABLE cd_tests ADD FOREIGN KEY (tests_donor_id) REFERENCES cd_donor(donor_id) ON DELETE CASCADE;
ALTER TABLE cd_tissue ADD FOREIGN KEY (t_protocol_id_fk) REFERENCES cd_protocol(protocol_id) ON DELETE CASCADE;
ALTER TABLE cd_tissue ADD FOREIGN KEY (t_donor_id_fk) REFERENCES cd_donor(donor_id) ON DELETE CASCADE;
ALTER TABLE cd_tissue ADD FOREIGN KEY (t_freeze_id_fk) REFERENCES cd_freeze(freeze_id) ON DELETE RESTRICT;
ALTER TABLE cd_batch ADD FOREIGN KEY (b_protocol_id_fk) REFERENCES cd_protocol(protocol_id) ON DELETE RESTRICT;
ALTER TABLE cd_batch ADD FOREIGN KEY (b_flask_id_fk) REFERENCES cd_flask(flask_id) ON DELETE RESTRICT;
ALTER TABLE cd_batch ADD FOREIGN KEY (b_freeze_id_fk) REFERENCES cd_freeze(freeze_id) ON DELETE RESTRICT;
ALTER TABLE cd_batch ADD FOREIGN KEY (b_isolation_id_fk) REFERENCES cd_isolation(isolation_id) ON DELETE CASCADE;
ALTER TABLE cd_picture ADD FOREIGN KEY (p_batch_id) REFERENCES cd_batch(batch_id) ON DELETE CASCADE;
ALTER TABLE cd_handout ADD FOREIGN KEY (h_batch_id) REFERENCES cd_batch(batch_id) ON DELETE CASCADE;
ALTER TABLE cd_handout ADD FOREIGN KEY (h_isolation_id) REFERENCES cd_isolation(isolation_id) ON DELETE CASCADE;
ALTER TABLE cd_handout ADD FOREIGN KEY (h_tissue_id) REFERENCES cd_tissue(tissue_id) ON DELETE CASCADE;
ALTER TABLE cd_parent ADD FOREIGN KEY (child_id) REFERENCES cd_batch(batch_id) ON DELETE RESTRICT;
ALTER TABLE cd_tissue ADD FOREIGN KEY (structure) REFERENCES cd_structure(structure_id) ON DELETE NO ACTION;
ALTER TABLE cd_template ADD FOREIGN KEY (donid) REFERENCES cd_donor(donor_id) ON DELETE CASCADE;
ALTER TABLE cd_template ADD FOREIGN KEY (tisid) REFERENCES cd_tissue(tissue_id) ON DELETE CASCADE;
ALTER TABLE cd_template ADD FOREIGN KEY (isoid) REFERENCES cd_isolation(isolation_id) ON DELETE CASCADE;
ALTER TABLE cd_template ADD FOREIGN KEY (batid) REFERENCES cd_batch(batch_id) ON DELETE CASCADE;
ALTER TABLE cd_tests ADD FOREIGN KEY (tests_name_id) REFERENCES cd_disease(disease_id) ON DELETE RESTRICT;