BIOINFORMATICS ORIGINAL PAPER
Vol. 28 no. 10 2012, pages 1390–1397
doi:10.1093/bioinformatics/bts140

Databases and ontologies
Advance Access publication March 29, 2012

AutoLabDB: a substantial open source database schema to support a high-throughput automated laboratory

Andrew Sparkes and Amanda Clare*
Department of Computer Science, Aberystwyth University, Penglais, Aberystwyth, SY23 3DB, UK
Associate Editor: Martin Bishop

ABSTRACT

Motivation: Modern automated laboratories need substantial data management solutions to both store and make accessible the details of the experiments they perform. To be useful, a modern Laboratory Information Management System (LIMS) should be flexible and easily extensible to support evolving laboratory requirements, and should be based on the solid foundations of a robust, well-designed database. We have developed such a database schema to support an automated laboratory that performs experiments in systems biology and high-throughput screening.

Results: We describe the design of the database schema (AutoLabDB), detailing the main features and describing why we believe it will be relevant to LIMS manufacturers or custom builders. This database has been developed to support two large automated Robot Scientist systems over the last 5 years, where it has been used as the basis of an LIMS that helps to manage both the laboratory and all the experiment data produced.

Availability and implementation: The database schema has been made available as open source (BSD license), so that others may use, extend and improve it to meet their own needs. Example software interfaces to the database are also provided. http://autolabdb.sourceforge.net/

Contact: afc@aber.ac.uk

Received on November 11, 2011; revised on March 12, 2012; accepted on March 20, 2012

1 INTRODUCTION

1.1 Automation and data in laboratories

Automation in modern laboratories is becoming commonplace. Many organizations own one or more automated systems, each designed to optimize a specific chemical or biological assay or process. These systems may later get reused in whole or in part to perform other tasks, when the laboratory refocuses on new goals or projects. The quantity of data produced by such systems usually far exceeds that produced by non-automated methods. The quality of the data is also improved; small effects become detectable due to the reliability of the hardware and the repeatability of experiments in large numbers. At Aberystwyth University we have been early adopters of large automated laboratory systems, and so have also had early exposure to the data-handling required to make the best use of them.

An LIMS (Laboratory Information Management System) is a software system for handling experimental data and meta-data, and is a vital component in a modern automated laboratory. While commercial off-the-shelf LIMS exist, they are necessarily designed to record specific tasks. It is highly unlikely that any automated laboratory can use an existing LIMS without extensive modification, to cover the specifics of their working environment, project aims and data recording requirements. For this reason, an open source LIMS with a clearly specified data structure is highly desirable, so that it can be adapted or extended by any experienced software engineer.

In the field of Computer Science, good data structures are known to be fundamental to producing quality systems. Many programming languages or formal systems emphasize data structures as the key to producing good software. Modules on Data Structures form the theoretical basis of undergraduate computer science programming courses. In spite of this we find that most existing LIMS systems, whose primary purpose it is to manage data, are typically geared towards intuitive and aesthetically pleasing user interfaces rather than having a solid core data design. When working in an automated laboratory, where data collection, entry, analysis and reporting are produced by machines such user interfaces are helpful, but not nearly as important as having a good underlying data structure.

1.2 Robot Scientists and AutoLabDB

The laboratory that has inspired the development of the AutoLabDB database is perhaps even more automated than most modern research laboratories, as it is an environment for Robot Scientists. A Robot Scientist is a robotic laboratory system that uses methods from machine learning to automatically generate and execute cycles of scientific experimentation (King *et al.*, 2009; Sparkes *et al.*, 2010). It generates hypotheses, designs the experiments to test them, physically executes the experiments on automated laboratory equipment, analyses and interprets the results, and then repeats the cycle. One major advantage of this approach to scientific study over standard human experimentation or even the latest high-throughput techniques, is the ability of a Robot Scientist to record in great detail every step of its reasoning and execution, allowing for greater transparency, repeatability of experiments and reusability of results.

The Robot Scientist project comprises two large integrated laboratory systems, named ‘Adam’ and ‘Eve’, both designed for systems biology research using assays based on *Saccharomyces cerevisiae* yeast running in microtitre plates. The Adam system (commissioned in 2006) performs functional genetics experiments and is capable of executing ~1000 experiments a day, whereas the Eve system (commissioned in 2009) performs intelligent screening campaigns aimed at improving primary pharmaceutical drug screening and can execute over 2000 experiments a day.

*To whom correspondence should be addressed.

AutoLabDB has been in operation since 2006 to support these systems.

As the aim of the research is to automate and formally record as much of the scientific process as possible, we attempt to capture details in the database from each stage of the process, whether that stage is manual or fully automatic.

- Organism and chemical libraries are manually prepared, and their content and location stored to facilitate automated selection.
- Experiments may be manually or automatically designed, and a job submitted and stored.
- High-level experiment content details are automatically generated from the jobs and stored.
- Individual assay content is automatically generated and plate layouts created and stored.
- Technicians manually prepare the required chemical and media solutions using stored recipe protocols and stock information, and record their creation and allocation for automated experiments.
- Experiments are then executed under automation on the robotic systems. Use of physical consumables and solutions, as well as movements of microtitre plates are tracked.
- Experiment observation data, and environment and instrument meta-data are automatically recorded as the experiments are executed.
- Observations are then automatically analyzed and derived results generated, stored and summarized for users.
- Further cycles of experimentation may then follow.
- Users may manually check the state of the systems and the experiments, and view the experiment observations at any time.

AutoLabDB includes tables for: results data that have been parsed from files produced by instruments (such as plate readers, liquid handlers and tube scanners), environmental sensor data, records of physical transfers of media and yeast inoculum into assay plates, job creation descriptions, experiment plans, the data needed to prioritize the optimal queuing of experiments, experiment microtitre plate layouts, stock control data, and data that tracks consumable movements and events.

Views of this data can be derived to show: results data and meta-data from both automated Robot Scientist systems, content of the yeast strain library and its status, laboratory environment sensor data, the definitions, suppliers, laboratory stocks and usage of chemicals and chemical mixtures, the current contents of automated freezers, incubators and consumable stacks in the robotic systems, experiment contents, chemical stock levels, and the content of chemical compound and yeast libraries.

In this work, we present the substantial database that has been developed to support an automated laboratory. The data design decisions that we have taken should also be applicable to other laboratories. AutoLabDB can be used in part or in whole to assist other developers in LIMS design and development. Our interface code to interact with this database (both for automated and human access) is also provided as open source, to serve as an example.

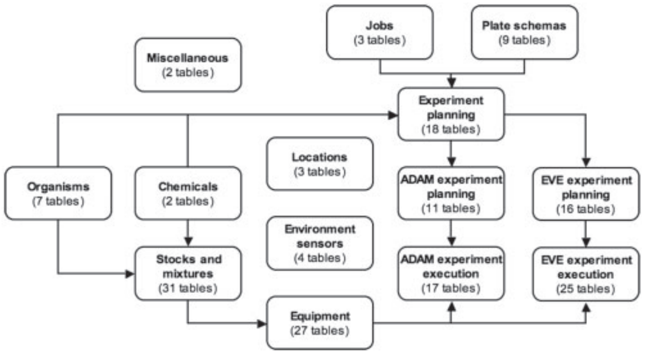

Fig. 1. A high-level diagram of regions in the database schema.

2 DESIGN

2.1 Database schema

The AutoLabDB schema is the result of over 5 years continuous development, and has evolved in parallel with the manual and automated biological experimentation performed in the Robot Scientist laboratory during that time. Initially, in 2006, the schema was limited in scope and included tables to hold only the simplest of physical yeast library and assay plate information, and the primary observed results data generated by the Adam Robot Scientist. Over the following years the schema was extended to hold details from more stages of the scientific process, including information related to experiment plans, stock control, equipment and consumables, derived results, and to store the chemical compound libraries and meet other requirements of the second Robot Scientist, Eve, commissioned in 2009.

See Figure 1 for an overview of the database schema. The schema has been visually sub-divided into a number of regions that contain related tables, to aid both in understanding the relationships between entities and in maintenance. Full details of the schema including a detailed diagram, descriptions of each table, the table creation script, MySQL Workbench file and other related information is included on the AutoLabDB website.

The most interesting design decisions made during the development of this schema are described in the following sections. We also describe how these decisions allow AutoLabDB to be reusable/extensible for other laboratories.

2.2 Abstraction and data hierarchies

AutoLabDB's design priority was to support the work of the high-throughput automated experiments being executed during the Robot Scientist project, as this work developed and changed. The schema is currently designed to support two complex high-throughput automated systems, as well as to help manage laboratory consumables, and to hold information about both organism and chemical compound libraries. It was designed to be extensible, such that further automated systems could be incorporated, with different experiments, equipment and consumables.

2.2.1 Separation of abstract and physical information The AutoLabDB schema explicitly separates abstract information from physical information, to minimize repetition of data and to allow more of the scientific process to be recorded. Abstract information includes: experiment plans, stock mixture recipes, planned assay and library microtitre plate layouts, sampling plans and liquid transfer plans. Physical information includes: observed results, current equipment status, biological and chemical library content, location and status, environment sensor data, stock control data, experiment events, and physical tube and well contents. This separation allows multiple unique physical instances to be connected to the corresponding abstract descriptions; experiments can be planned in advance, revised, and then executed once or repeated many times (or maybe not at all if the reason for performing the experiment is superseded).

A simple example of this is the relationship between 'plate model' and 'plate instance' (see Fig. 2a). The *plate_model* table (along with its super-type table *consumable_equipment_model*) holds the abstract definition of the type of plate; how many wells it contains, the working volume, the manufacturer, its product code for reorder etc. This allows for easy definition of new plate models as required. Whereas the *plate_instance* table holds information about specific instances of plates (each with a unique barcode) that can be individually tracked and against which observed results and other meta-data can be recorded. E.g. a single Greiner 384-well assay plate model may have hundreds of uniquely barcoded instances.

A second example is the abstract 'stock mixture protocol' as compared with the physical 'stock mixture' (see Fig. 2b). The *stock_mixture_protocol* table defines any recipe for making up a chemical mixture. In contrast, the *stock_mixture* table holds information about the specific instances of mixtures, each with a unique identifier that allows tracking of usage in experiments. This is all designed to be flexible, to allow any number of chemicals and/or existing mixtures to be combined in a recipe.

Other examples include: the various 'plate layout' tables which hold the planned content of the individual wells on assay or library plates (biological and/or chemical components), the 'screening plan' tables for the Eve system which define which biological strains are to be screened against which chemical compounds, and the 'experiment planning' tables for both the Adam and Eve systems, which define all the experiments to be run and the order in which assay plates to be made.

2.2.2 Data hierarchies The AutoLabDB schema contains numerous examples of hierarchically structured data, including those for the abstract versus physical data already discussed.

Hierarchies allow data types to be extended, so that future changes in lab equipment, consumables, experiment data etc. can easily be accommodated.

The difficulty of implementing hierarchical data structures within a relational database is a known issue (Faroult and Robson, 2006; Kamfonas, 1992), and while the database community has experimented with hierarchical database systems, most practitioners prefer to remain with the benefits of relational databases and use one of a handful of standard techniques (e.g. adjacency list model and nested set model).

2.2.3 Sub-class/super-class hierarchies (abstract and physical) Sub-class/super-class hierarchies can be represented using a single table, with one field to represent the type, or as multiple tables. Single table approaches provide better support for polymorphic queries, but potentially waste space when fields that are meaningful for one class are not needed for another, and so require that all classes are tightly coupled. A combination of a single super-class table and multiple sub-class tables is often the most practical solution, and is the solution adopted for AutoLabDB.

The equipment hierarchy (see Fig. 2c) is designed to allow for the tracking and stock keeping of disparate types of items in the laboratory (e.g. instruments, chemicals, microtitre plates, tubes, tips etc.). This starts with the abstract super-type 'purchasable item' at the top level. Purchasable items are linked directly to abstract 'supplier items' (an instrument, chemical or consumable may be supplied by several companies) and through to physical 'stock items' for tracking physical stock in the laboratory. Purchasable items are sub-typed into the abstract 'chemical', 'capital equipment model' and 'consumable equipment model' categories, and consumable models are further sub-typed into abstract 'plate', 'tube', 'tube rack' and 'tip box' models. These abstract sub-type tables are then each linked in turn to tables representing the physical object instances. For example, an abstract chemical description may have one or more abstract 'supplier items' (e.g. two suppliers for a chemical) which in turn can have one or more physical 'stock items' (e.g. three different batches of the same supplier chemical), or an abstract 'tube model' may have many physical 'tube instances' etc.

The physical object data may itself be further sub-typed. For example, the microtitre 'plate instance' data are further sub-divided into a variety of usage types (these are specific to our systems), e.g. 'Adam yeast library plate instance', 'Eve assay plate instance' etc. Results and meta-data tables then link off each of these usage-specific tables.

This design is easily extensible, for example, new definitions for plates or tubes can be easily created in the data, or if a new consumable model is needed (e.g. petri-dish) the relevant tables can be added without disruption to the existing data structure. The design also ensures that concepts that are common to all laboratories, such as 'purchasable item', do provide a common core structure, and can be specialized for each laboratory as needed.

2.2.4 Self-referencing hierarchies Adjacency list models were used in the schema where data in a table needed to be self-referencing to allow for a nested hierarchy.

An example of this is the *location* table, which stores positions within the laboratory and our robotic systems (see Fig. 2d). Chemical stocks, instruments and consumables all have a location reference, allowing users to find them, and the automated systems to track movements and usage during automated experiments (very useful for debugging assay problems). Each location may have a parent location, allowing nested self-referencing, potentially to many levels. For example, a specific batch of a chemical may be located in box 3, on the second shelf, in the fridge, in room 36 or in the biology building. Our stock control software can display this information so a technician can find it. Or, a specific compound library microtitre plate may be located in slot 7 of stack 3 in the dry store, in the Eve robotic system, allowing experiment planning software to schedule its use in compound screening.

Nested hierarchies allow flexibility in the depth of relationships. A laboratory may record simple locations without nesting, or may choose to nest racks in boxes on shelves in freezers in rooms, as required, to an arbitrary depth. AutoLabDB supports the full range of complexity, and more nesting levels can easily be added as the laboratory grows.

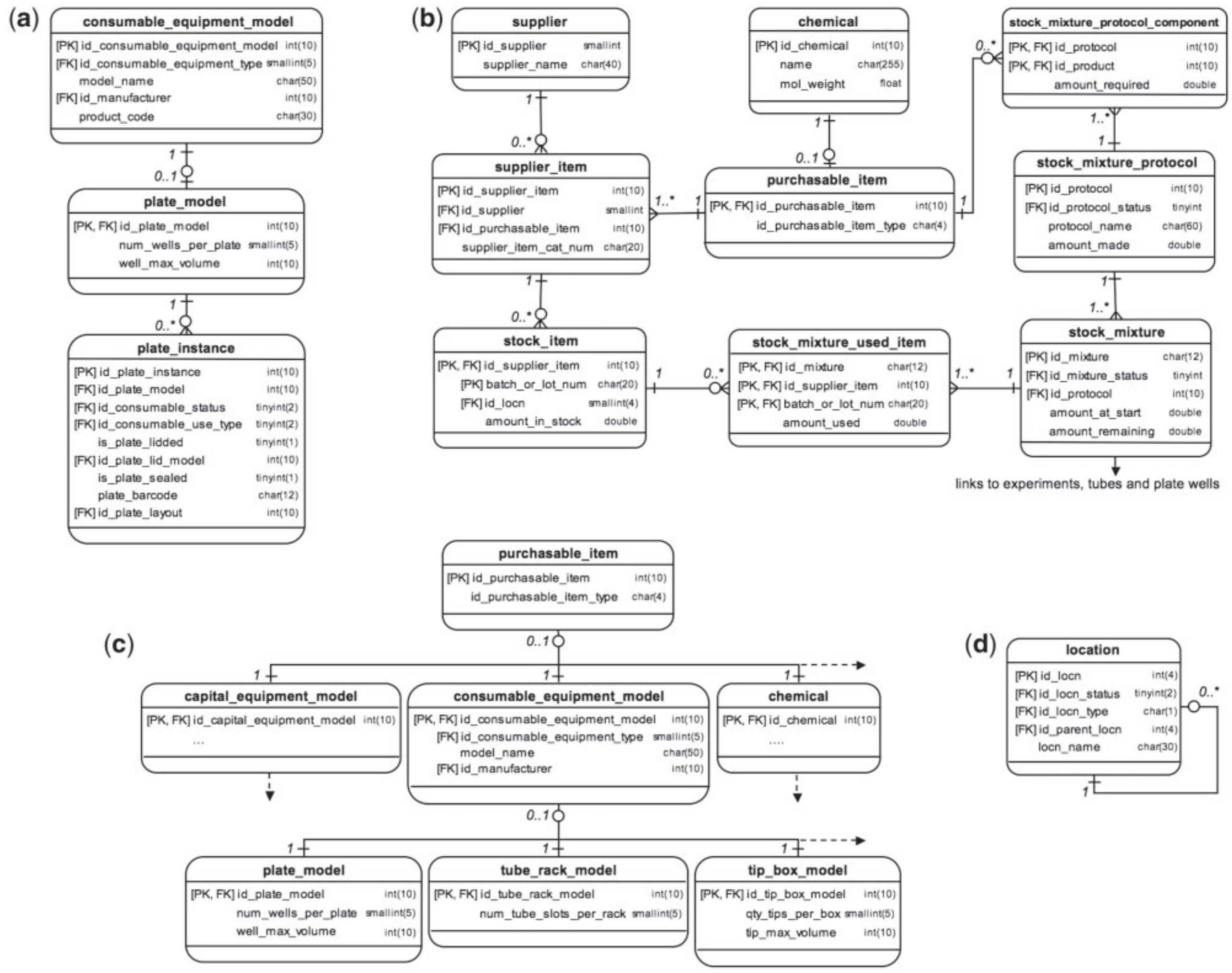
(a)
consumable_equipment_model
[PK] id_consumable_equipment_model int(10)
[FK] id_consumable_equipment_type smallint(5)
model_name char(50)
[FK] id_manufacturer int(10)
product_code char(30)
1
0..1
plate_model
[PK, FK] id_plate_model int(10)
num_wells_per_plate smallint(5)
well_max_volume int(10)
1
0..*
plate_instance
[PK] id_plate_instance int(10)
[FK] id_plate_model int(10)
[FK] id_consumable_status tinyint(2)
[FK] id_consumable_use_type tinyint(2)
is_plate_lidded tinyint(1)
[FK] id_plate_lid_model int(10)
is_plate_sealed tinyint(1)
plate_barcode char(12)
[FK] id_plate_layout int(10)
(b)
supplier
[PK] id_supplier smallint
supplier_name char(40)
1
0..*
supplier_item
[PK] id_supplier_item int(10)
[FK] id_supplier smallint
[FK] id_purchasable_item int(10)
supplier_item_cat_num char(20)
1..*
1
chemical
[PK] id_chemical int(10)
name char(255)
mol_weight float
1
0..1
purchasable_item
[PK, FK] id_purchasable_item int(10)
id_purchasable_item_type char(4)
1
0..*
stock_mixture_protocol_component
[PK, FK] id_protocol int(10)
[PK, FK] id_product int(10)
amount_required double
1..*
1
stock_mixture_protocol
[PK] id_protocol int(10)
[FK] id_protocol_status tinyint
protocol_name char(60)
amount_made double
1
1..*
1
0..*
stock_item
[PK, FK] id_supplier_item int(10)
[PK] batch_or_lot_num char(20)
[FK] id_locn smallint(4)
amount_in_stock double
1
0..*
stock_mixture_used_item
[PK, FK] id_mixture char(12)
[PK, FK] id_supplier_item int(10)
[PK, FK] batch_or_lot_num char(20)
amount_used double
1..*
1
stock_mixture
[PK] id_mixture char(12)
[FK] id_mixture_status tinyint
[FK] id_protocol int(10)
amount_at_start double
amount_remaining double
links to experiments, tubes and plate wells
(c)
purchasable_item
[PK] id_purchasable_item int(10)
id_purchasable_item_type char(4)
0..1
1
1
1
capital_equipment_model
[PK, FK] id_capital_equipment_model int(10)
...
consumable_equipment_model
[PK, FK] id_consumable_equipment_model int(10)
[FK] id_consumable_equipment_type smallint(5)
model_name char(50)
[FK] id_manufacturer int(10)
chemical
[PK, FK] id_chemical int(10)
....
0..1
1
1
1
plate_model
[PK, FK] id_plate_model int(10)
num_wells_per_plate smallint(5)
well_max_volume int(10)
tube_rack_model
[PK, FK] id_tube_rack_model int(10)
num_tube_slots_per_rack smallint(5)
tip_box_model
[PK, FK] id_tip_box_model int(10)
qty_tips_per_box smallint(5)
tip_max_volume int(10)
(d)
location
[PK] id_locn int(4)
[FK] id_locn_status tinyint(2)
[FK] id_locn_type char(1)
[FK] id_parent_locn int(4)
locn_name char(30)
0..*
1

Fig. 2. (**a**) Plate models (abstract) and plate instances (physical). (**b**) In stock control chemical mixture protocols (abstract) are used to create chemical mixtures (physical). (**c**) Sub-class/super-class hierarchies used in the representation of laboratory equipment, consumables and chemicals. (**d**) The self-referencing hierarchy for nested item locations.

2.2.5 Flexibility versus practicality The AutoLabDB schema was designed to be flexible to allow for the addition of further high-throughput systems in the future, or modification of experiments performed on the existing systems. We also intend it to be used in some form by other laboratories. But consideration was also given that it be practical to use. There is a trade-off between complete flexibility and deliberate limitation to restrict relationships and so to simplify the requirements of the interacting software. For example, the schema is focused on high-throughput microtitre plate experiments, and some schema relationships have been deliberately de-normalized to make querying easier. For example, there is a link table between the assay instance and physical well tables to avoid an extensive multi-table join query that would otherwise be necessary.

2.3 Data volumes

A database schema for a high-throughput automated laboratory needs to cope with high volumes of data, and must make that data efficient to input and to access. For example, when a 384-well assay plate on Eve is observed with three different fluorescent filters every 90 min for 40 h during a screen; this generates 30 720 observed readings. If just 10 such plates are created and screened, this results in 307 200 readings in under 2 days. Such volumes of data require efficient table indices, with very fast insert and select times. This was taken into consideration when creating the AutoLabDB schema. For example, when designing the plate readings table we considered the format of the instrument data files being parsed, how the observed data was connected to the experiments, and how the data would later be accessed and displayed.

2.4 Database organization and version control

The database schema currently comprises 175 tables arranged into 14 regions (see Fig. 1 for an overview and the AutoLabDB website for full schema details). The majority of these regions should be generally reusable by all microtitre plate laboratories. The regions that are specific to the laboratory at Aberystwyth are labelled with ‘Adam’ and ‘Eve’. Links between regions are typically via foreign keys on just one table. This compartmentalization allows development in one region without adversely affecting others, while avoiding the complications to software interfaces and queries that arise if regions are separated into distinct databases. Version control of all schema changes provides a historical record of how the database evolves over time. Schema modifications requiring alterations to existing software interfaces were developed and system-tested in a parallel replica environment before implementation.

The requirement to re-factor software after making schema design changes is an important consideration. Our schema has evolved continuously since its inception and our software interfaces have evolved in line with these changes. Persistence frameworks (Ireland *et al.*, 2009) can help to reduce the impact on software when making additions or minor modifications to a schema, but we found that significant software changes were usually needed to take advantage of the new features in the schema design. We found it more efficient and less disruptive to make large scheduled step changes to the schema and associated software rather than many small changes.

3 IMPLEMENTATION

3.1 Database Server Architecture

There are many forms of database technology for storing data. These include relational databases (Codd, 1970), object oriented databases (Eilbeck *et al.*, 1999), array-based databases (The SciDB Development Team, 2010), write-one-read-often data storage such as the Google File System (Ghemawat *et al.*, 2003) and RDF triplestore databases (Miles *et al.*, 2010; Ruttenberg *et al.*, 2009). We chose to use relational database technology for AutoLabDB because it is mature, stable and well supported by a large user community. While the others each offer attractive advantages, they are still mainly Niche technologies.

In particular we chose MySQL to host our schema. MySQL is a robust, scalable, high performance open source database server with significant community support. It can handle the large data volumes required, and has comprehensive open source tools for designing schemata, for querying and for administration tasks. Insert and access times are very fast, allowing, recording and viewing of high-throughput experiment data in near realtime. MySQL has also proven to be very well supported, allowing us to develop software interfaces in many different programming languages and on multiple operating systems. Backing up data are also straightforward; incremental backups are taken automatically every night, and full backups whenever a significant change is made to either the schema or the data.

3.2 Database content

The following sub-sections describe key areas of the AutoLabDB schema and their content.

3.2.1 Stock control Detailed stock control is useful in most laboratories, but becomes essential for an automated laboratory. Knowledge of the precise components of any chemical solution and their origin is important for documenting, understanding and analyzing experiment results. Awareness of available stock levels is vital during planning; choice of experiment may depend on the availability and cost of components as well as the likely value of the information to be gained.

AutoLabDB allows the recording of: abstract chemical information, suppliers, placed orders, deliveries, locations and stock levels by chemical batch, and usage of chemicals and mixtures for both manual and automated experiments. Recipes for chemical mixtures are stored and can then be viewed by users. These recipes describe the amount of each component needed to make the requested volume, and the available batch stock levels and locations. When stock is recorded as having been used, the stock levels can be automatically updated. As chemical mixtures are created they are assigned a unique identifier.

The database includes tables to track usage of the uniquely identifiable chemical mixtures on the robotic systems, linking them and the volumes transferred during the liquid handling steps to the individual wells in microtitre plates. This information is then available for automated anaysis of results, for the production of human-viewable visualizations of experiment results, and for the production of reports. Mixture identities are also an effective way to track the stock details of human-conducted laboratory experiments. This data proved very useful in diagnosing causes when biological assays behaved unexpectedly due to missing or flawed components. We also found the process of recording mixture identities for all experiments greatly reduces human error, while also providing detailed historical records of the precise contents of experiments.

3.2.2 Job creation and experiment planning Experiment descriptions and plans for high-throughput experiments or test runs on the automated robotic systems are known as ‘jobs’. These jobs can be created either automatically or manually, and are treated identically.

The job descriptions are at the abstract level, containing experiment meta-data, biological and chemical descriptors, replicate numbers and a choice of template for plate layout. Jobs can be created by humans or automatically by intelligent software. After job details have been entered into the database, they can be automatically ordered by intelligent software (Byrne, 2007) to optimize execution time and efficiency of access to stock and equipment.

3.2.3 Sample selection Selection of appropriate sources for biological and chemical components for automated experiments is a factor that can have enormous influence over the time experiments take to prepare and run, their cost, and the quality of results. We find that by using the information stored in the database these sample selection processes are made far more reliable; this reduces waste and failure rates in experiments and improves the quality of results.

Automated sample selection interfaces are used on both Adam and Eve robotic systems, for yeast strain inoculation and compound screening, respectively. For yeast experiment inoculation in Adam database tables regarding frozen yeast library plate wells include fields describing their relative locations, their ages and the counts of previous picks from those wells, all of which enables humans or intelligent software to make suitable decisions about which plates to select and which wells to sample. Yeast strains are stored in replicates across multiple plates in the library, to increase selection possibilities. The resulting sample ‘picking list’ is optimized for the time to select and return successive plates to the freezer, and to use stocks of similar age, and thus viability. The optimized picking list is recorded in the database and is then available for use in experiments on Adam.

Additionally, during the automated inoculation step on the Adam robotic system, the depth of sampling from the frozen yeast wells can be determined as a factor from the recorded counts of previous picks. This greatly improves the reliability of inoculation and growth, because repeated picking gouges a hole in the frozen sample. Old or heavily used yeast plates can be ascertained from the age of samples and number of picks recorded, and these plates can then be replaced.

For chemical compound screening in Eve, there is a list of specific compounds to be cherry-picked from the library. Records of volumes remaining from the latest acoustic survey can be used to determine the most suitable choice of compound wells from which to sample. The locations of the compounds determine the most efficient picking order. Compounds from the same source library plate can be grouped together during assay plate creation. Reports can be produced describing the content and volumes remaining in the master library tubes. For daughter libraries the latest acoustic survey results give well volume and hydration information. Change in volume or hydration over time can be plotted (compounds are dissolved in di-methyl sulfoxide, DMSO, which absorbs water from the air). This aids compound library management, allowing monitoring of effects from usage, evaporation and hydration.

3.2.4 Experiment execution The database holds a realtime record of the current state of the experiment processes. This allows a robotic system to be restarted after a failure and be continued, minimizing disruption, reducing human error and lowering the skill requirements for the operator.

For compound screening the abstract plans stored in the database are used to direct the flow of experiments. The control software protocols running on the robotic system query this data to determine what actions to perform next, creating assay plates by specifying source library plates and all the liquid handling details.

Users can request data about the realtime state of the experiments as they progress through the automated systems. For example, of 10 assay plates planned, 6 have already been successfully created and are in the reading cycle, while a 7th is being generated and the other 3 have not been started. This status information allows users to plan their workday around the system's progress, and to monitor system state and progress without needing to be physically in the laboratory.

3.2.5 Consumable tracking Tracking consumable movements and events can have significant advantages when attempting to identify the causes of systematic noise or other anomalies in observed experiment results. For example, when we noticed a periodic crenelation affect on some observed growth curve results in Adam, we were able to trace the problem back to a difference in the length of time between unloading the assay plate from each of the two shaking incubators before reading it. This time difference allowed more cell settling after the plate was unloaded from one incubator than from the other, increasing observed optical density. Tracking data allowed diagnosis of this issue, enabling us to identify the cause and fix the problem by adding a plate shaker step prior to reading the plates.

AutoLabDB records movements of consumables around the systems and all the events that happen to them. For example, an assay plate is created in the database when it is first taken from a consumable stack (by a robot arm) and scanned (unique barcode), then it may get placed into a nest and be de-lidded, then be moved to a liquid handler, have compounds dispensed into it, then moved to a second liquid handler where a biological assay is dispensed into it, then be moved back to the nest to be re-lidded, before entering an incubate and read cycle. All these movements and events are stored in the database, so we have a complete historical record of everything that has happened to each assay plate in every experiment performed.

There is also the capability to record consumable meta-data (type and model of consumable), and by recording locations of consumables, reports can be generated showing the current contents of equipment such as incubators, freezers and drystores, and the remaining free capacity of the system.

3.2.6 Realtime recording of observations Raw observed experiment data can be parsed and uploaded to the database in near realtime, and linked to specific consumable instances using the scanned barcodes. For the Robot Scientist systems this includes output from multi-functional plate readers, liquid handlers and scanners and constitutes the bulk of the results data (tens of millions of individual observations). Having clear views on data from still ongoing experiments is a huge advantage, especially in experiments that take several days to complete. Users can monitor the experiments as they progress, and potentially fix issues before experiments are ruined. The selection of growth data and on-the-fly generation of growth plots (time series data) for all 384 wells of a plate is fast using MySQL, and occurs without noticeable delay in a web browser.

4 DISCUSSION

Every laboratory has different priorities, and so has different needs from their information management system. The information structure of such a system must keep abreast of changes in the laboratory, as new equipment is purchased and new workflows are made possible. Useful and flexible laboratory data management software is difficult to achieve and maintain. Commercially available LIMS providers and systems include: IDBS's ActivityBase, CambridgeSoft's BioAssay and Accelrys's Assay Explorer. Open source LIMS include BioRails, Open-LIMS and BikaLIMS (www.biorails.org/projects/core; www.open-lims.org and www.bikalabs.com), but none of these make their core database explicit or describe it in public documentation. Each of these attempts to cover some aspects of the processes performed in a typical laboratory. However, very few open source LIMS are flexible yet detailed enough to be capable of assisting an automated laboratory in a full scientific work cycle.

In order to allow others to modify and extend AutoLabDB we have taken care to use hierarchical data structures which can be extended or sub-classed, to use generic concepts wherever possible, and to compartmentalize regions of the database so that future changes to one part are isolated from the rest. We provide full schema diagrams and descriptions of the purpose of tables to enable engineers to plan and make suitable changes.

Screensaver (Tolopko *et al.*, 2010), a recently published open source LIMS, is perhaps the closest alternative to AutoLabDB. It provides a web-based solution for high-throughput screening applications and, just as AutoLabDB was originally developed to support the laboratory at Aberystwyth, Screensaver was developed to provide support for the screening facility at Harvard Medical School. Screensaver uses a PostgreSQL relational database of 77 tables.

Screensaver and AutoLabDB currently support different features. For example, Screensaver includes user permissions for data, roles of users in the activities that were performed and a description of who the screen was performed for and why. This gives a level of user accountability that AutoLabDB does not currently have. AutoLabDB, on the other hand, supports stock control, consumable location tracking, chemical mixtures and recipes, plate layout plans, high level job descriptions and the capture of raw plate reader results, which allows the display of realtime results. Both systems keep records of library history, track liquid volumes, distinguish physical copies of plates from the underlying library details and record data to manage cherry picking workflow. Screensaver is more specific in scope, aimed at high-throughput screening applications only. AutoLabDB has been used for high-throughput screening, but also for other applications, and has been designed with the aim of widening its usage to a variety of automated laboratories. It has been used to study yeast functional genomics using single gene deletion yeast strains, supporting the Robot Scientist Adam (King *et al.*, 2009). It has also been used to investigate the utility of different gel/gum materials for long term suspension of frozen yeast samples, and the uptake of different carbon and nitrogen sources for yeast metabolism (Lu *et al.*, 2010), among other tasks. These applications all require stock control, experiment design and layout in microtitre plates, recording of optical density measurements taken automatically over a 5-day growth period, management of locations and history of stored samples, and tracking volumes, batches and protocols of media, chemicals and organisms added to the experiment.

Other LIMS worthy of comparison include Brunn (Alvarsson *et al.*, 2011), which is a Bioclipse plugin comprising a relational database and graphical front-end. Its data model is based around microplate screening experiments, and shares some similarities with AutoLabDB. Brunn includes abstract plate layouts as well as records of physical plates (layouts can be re-used to create many plates), it includes data hierarchies (plate types and samples), it links plate wells to their contents (including drug samples, patient samples and cell samples), and it links wells to kinetic results to allow the plotting of curves.

SLIMS (Van Rossum *et al.*, 2010) 'Sample-based Laboratory Information Management System' is a web application designed for users to record sample information in genotyping laboratories. It allows users to create and share lists of sample information linking human subjects, biological samples and the containers they are stored in. The data model is relatively small (24 data objects), and it does not hold any stock control, experiment, or results data, nor does it hold abstract information. It does log modifications to the data to provide traceability, which AutoLabDB does not.

An older system with the same acronym, SLIMS, 'Small Laboratory Information Management System' (Kelley *et al.*, 2004), is a Python and Metakit database application which has been designed for the storage and analysis of chemical screening data. The data model is relatively simple, but again has some similarities with AutoLabDB including handling microplate types and results data, physical locations and details of compounds. It also holds simple plate layout information that describes the position of controls for use when analyzing results data. It does not include stock control, experiment hierarchy, well content or other features found in AutoLabDB.

QuickLIMS (Kokocinski *et al.*, 2003) is a Microsoft Access and VBA-based LIMS designed to aid in the manufacture of microarray plates. Its data model is again relatively small, and is specialized to hold information about microarray plate creation. It does have a plate hierarchy structure and plate content information, but not stock control, experiment or results data.

Human-friendly interfaces to AutoLabDB could be provided by a variety of methods, and we provide examples of some of our Java/Tomcat interfaces at the AutoLabDB website. WIST (Huang *et al.*, 2011) is a recent Perl toolkit for building workflows that generate web-based forms to accept and view lab data, which assumes that the user has already designed a back-end database and could be used to provide another means of interfacing to AutoLabDB.

To summarize, we have developed a substantial relational database schema and associated software interfaces (LIMS) to both record and display the data and meta-data produced in a laboratory performing automated high-throughput biological experiments. This work currently supports two large automated systems at Aberystwyth University. The majority of the relational database schema and software interfaces are generic in nature and can be re-used in other laboratories. We make the schema and software available as open source with the aim that others in the community may find it useful and contribute to its development. The AutoLabDB schema comprises 187 individual tables, and we are currently using it to store over 50 million rows of data.

Future development plans for AutoLabDB include: enhancement of the 'organisms' section to include genotype data about the strains, enhancement of the 'chemicals' section to distinguish isomers, tautomers etc. and store 3D structure [we store InChI, KEGG, CAS, Smiles and IUPAC identifiers, and recognize that the accurate storage of chemical data is a subject of active research (Bolton *et al.*, 2008; Degtyarenko *et al.*, 2008)], integration of AutoLabDB data with barcode printers for further consumable tracking, user authentication, and additional user-friendly interfaces.

We strongly encourage the reuse of the AutoLabDB schema, in whole or in part, to assist in the design of LIMS infrastructures for other laboratories.

ACKNOWLEDGEMENTS

We would like to thank our technician colleagues for helping us to test and improve AutoLabDB (Magdalena Markham, Kate Martin and Ron Pateman) and the Computational Biology group (past and present) at Aberystwyth University for general help and support. We also thank the anonymous reviewers for their helpful comments and recommendations.

Funding: Biotechnology and Biological Sciences Research Council [BB/D00425X/1, BB/F008228/1] and Royal Academy of Engineering/Engineering and Physical Sciences Research Council fellowship to A.C.

Conflict of Interest: none declared.

REFERENCES

Alvarsson,J. *et al*. (2011) Brunn: an open source laboratory information system for microplates with a graphical plate layout design process. *BMC Bioinformatics*, **12**, 179.

Bolton,E. *et al*. (2008) PubChem: integrated platform of small molecules and biological activities. In *Annual Reports in Computational Chemistry*. Vol. 4. American Chemical Society.

Byrne,E.L. (2007) Optimising the flow of experiments to a robot scientist with multi-objective evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. London, UK.

Codd,E. (1970) A relational model of data for large shared data banks. *Commun. ACM*, **13**, 377–387.

Degtyarenko,K. *et al.* (2008) ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res.*, **36**, D344–D350.

Eilbeck,K. *et al*. (1999) Interact: an object oriented protein-protein interaction database. In *ISMB '99*. Heidelberg, Germany.

Faroult,S. and Robson,P. (2006) *The Art of SQL*. O'Reilly Media.

Ghemawat,S. *et al.* (2003) The Google File System. In *19th ACM Symposium on Operating Systems Principles*. New York, USA.

Huang,Y. *et al*. (2011) WIST: toolkit for rapid, customized LIMS development. *Bioinformatics*, **27**, 437–438.

Ireland,C. *et al*. (2009) A classification of object-relational impedance mismatch. In *First International Conference on Advances in Databases, Knowledge, and Data Applications*, Cancun, Mexico, pp. 36–43.

Kamfonas,M.J. (1992) Recursive hierarchies: the relational taboo! *Rel. J*. **4**.

Kelley,B.P. *et al*. (2004) A flexible data analysis tool for chemical genetic screens. *Chem. Biol.*, **11**, 1495–1503.

King,R. *et al.* (2009) The automation of science. *Science*, **324**, 85–89.

Kokocinski,F. *et al*. (2003) QuickLIMS: facilitating the data management for DNA-microarray fabrication. *Bioinformatics*, **19**, 283–284.

Lu,C. *et al*. (2010) Constraint-based optimisation tools for semi-automated refinement of genome-scale yeast metabolic models. In *Poster presentation at the 11th International Conference of Systems Biology (ICSB2010)*. Edinburgh, UK.

Miles,A. *et al*. (2010) OpenFlyData: an exemplar data web integrating gene expression data on the fruit fly *Drosophila melanogaster*. *J. Biomed. Inform.*, **43**, 752–761.

Ruttenberg,A. *et al*. (2009) Life sciences on the Semantic Web: the Neurocommons and beyond. *Brief. Bioinform.*, **10**, 193–204.

Sparkes,A. *et al.* (2010) Towards Robot Scientists for autonomous scientific discovery. *Autom. Exp.*, **2**, 1.

The SciDB Development Team (2010) Overview of SciDB, large scale array storage, processing and analysis. In *SIGMOD'10*. Indianapolis, USA.

Tolopko, A. N. *et al.* (2010). Screensaver: an open source lab information management system (LIMS) for high throughput screening facilities. *BMC Bioinformatics*, **11**, 260.

Van Rossum,T. *et al*. (2010) SLIMS – a user-friendly sample operations and inventory management system for genotyping labs. *Bioinformatics*, **26**, 1808–1810.