A Spreadsheet-Based User Interface for Managing Plural Relationships in Structured Data

by

Eirik Bakke

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2011

Author ....
Department of Electrical Engineering and Computer Science
October 28, 2010

Certified by ...[illegible]...
David R. Karger
Professor
Thesis Supervisor

Certified by ...[illegible]...
Robert C. Miller
Associate Professor
Thesis Supervisor

Accepted by ...[illegible]...
Prof. Terry P. Orlando
Chairman, Committee on Graduate Students

A Spreadsheet-Based User Interface for Managing Plural Relationships in Structured Data

by

Eirik Bakke

Submitted to the Department of Electrical Engineering and Computer Science
on October 28, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

A key feature of relational database applications is managing *plural* relationships—one-to-many and many-to-many—between entities, be they customers and invoices, parts and suppliers, or meetings and conference rooms. However, since it is often infeasible to adopt or develop a new database application for any given schema at hand, information workers instead turn to spreadsheets, a general and more familiar data management tool which, unfortunately, lends itself poorly to schemas requiring multiple related entity sets. In this thesis, we propose to reduce the cost-usability gap between spreadsheets and tailor-made relational database applications by extending the spreadsheet paradigm to let the user establish relationships between rows in related worksheets as well as view and navigate the hierarchical cell structure that arises as a result. We present Related Worksheets, a spreadsheet-like prototype application, and evaluate it with a study involving 36 regular Excel users. First-time users of our software were able to solve most lookup-type query tasks without instruction, in one case 40% faster than on Excel.

Thesis Supervisor: David R. Karger
Title: Professor

Thesis Supervisor: Robert C. Miller
Title: Associate Professor

Acknowledgments

I would like to thank Paul Grogan and Yod Watanaprakornkul, who helped me design and build the first prototype of Related Worksheets during our project for the User Interface Design and Implementation course at MIT in the spring of 2009. Second, I thank my advisors for encouraging me to take the prototype through a formal user study; this has been a great learning experience for me, and also resulted in the development of the User Study Console, a broadly applicable tool for conducting online user studies. Third, I thank the reviewers who provided their comments on an early version of this document. Last, I thank the Mechanical Turk workers who participated in my user study, and who provided useful suggestions and critical bug reports both for the Related Worksheets prototype and the User Study Console tool.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Structured data is everywhere, and information workers need to manage it. On the storage side, the problem has largely been solved through the advent of relational databases; an off-the-shelf relational database management system will serve as a back-end for any database application. Yet, a back-end alone provides no user interface, and a database without a user interface is a database without users (see the commentary in The Lowell Database Research Self-Assessment [2]).

Certain kinds of task-specialized database applications may be useful to a large number of people and organizations. For these, high-quality free or commercial GUI implementations are likely to be available. Examples include Personal Information Managers (PIMs), Customer Relationship Management (CRM) software, and business accounting software. Other applications are far more domain-specific, and have to be developed from scratch, possibly by an organization's IT department or an external consulting firm. Real-world examples include oddities such as software used by Norwegian immigration authorities to manage applications for political asylum[1], software used by the Tunisian National Order of Pharmacists to manage documents and member affiliations[2], and software used by a small seafood trading business to keep track of sales and inventory of perishable goods[3]. In such cases, and even in less extreme ones, development costs may be very high relative to the

[1]Unique Flyktning by Visma AS. `http://www.visma.com`

[2]`http://netbeans.dzone.com/news/tunisian-national-order`

[3]WinFish, a piece of software encountered by the author. Like many of these applications, it is not publicly available.

number of potential target users.

1.1 Spreadsheets as Database Tools

When the effort required to either adopt or develop a new *tailor-made* relational database application is too high, information workers instead turn to a general and more familiar tool: the spreadsheet. One survey [8] shows that "sorting and database facilities" are the most commonly used spreadsheet features, with 70% of business professionals using them on a frequent or occasional basis. In contrast, less than half use "tabulation and summary measures such as averages/totals"—one of the original design goals of the original VisiCalc spreadsheet—or more advanced features. Furthermore, spreadsheet users "shun enterprise solutions" [10] and "do not appear inclined to use other software packages for their tasks, even if these packages might be more suitable" [3]. Presumably these ostensibly suitable software packages include specialized database applications; after word processors, database software is the second kind of software most commonly encountered by spreadsheet users, with 60% of spreadsheet users reporting some use of them [8].

Once it is established that spreadsheets are being used for database tasks, we should ask what functionality is lost when we compare them to corresponding tailor-made relational database applications. In this thesis we focus on the ability to maintain *plural* relationships—one-to-many and many-to-many—between different kinds of entities in a dataset, a key feature of any relational database application [4]. The world of tailor-made database applications has seen some fairly standard user interface paradigms for viewing and editing such relationships (see Related Work). We consider two aspects of these user interfaces to be crucial to their relationship management capabilities. First, they can provide the user with joined views of related tables in the database (albeit hard-coded for the particular schema in question). Second, tailor-made interfaces are not restricted to simple tabular views, but can expose relationships in a hierarchical fashion. This is, in particular, crucial for being able to visualize entities related to other entities through multiple distinct relationships; we will give an example of this in the next section. Neither joins nor hierarchical views are present in spreadsheets.

1.2 Joined Hierarchical Views

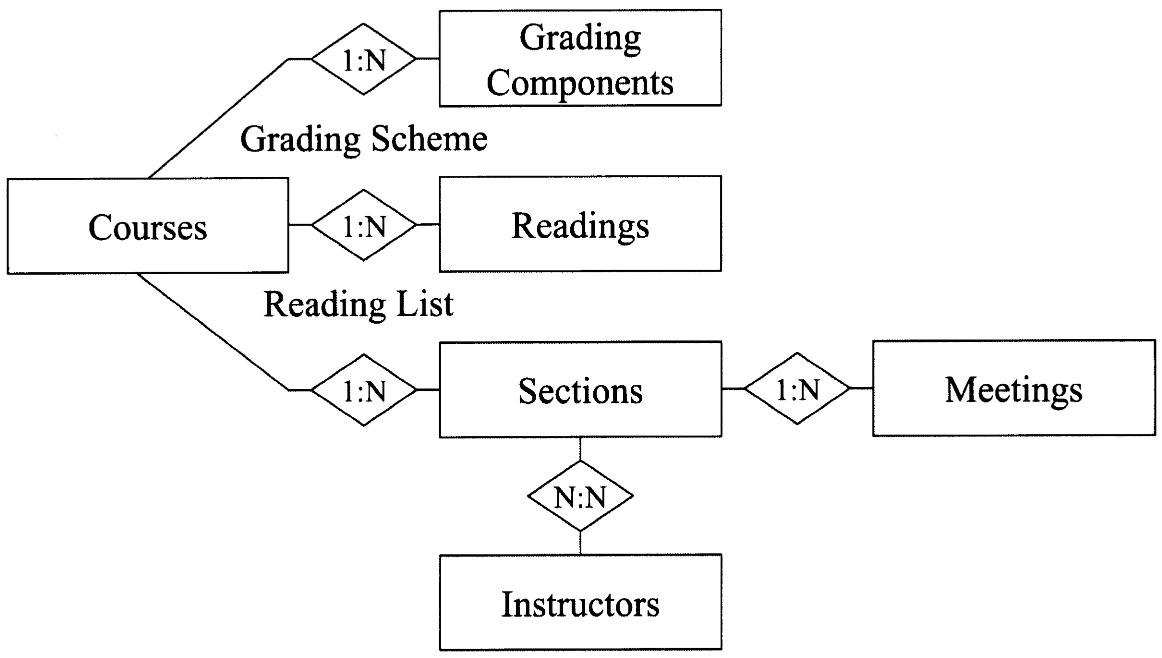
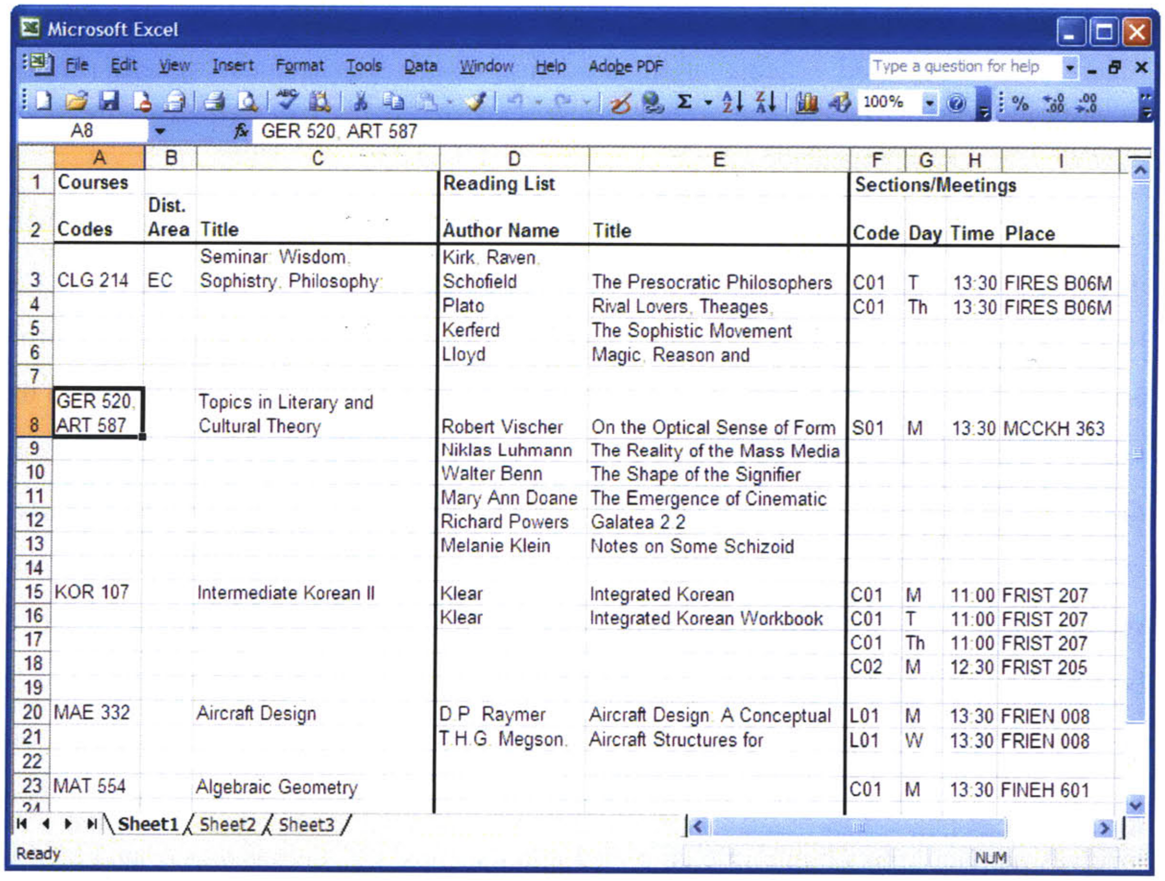


Figure 1.1: Simplified schema of an academic course management system (Entity-Relationship diagram).

As a running example, we consider a dataset derived from a real-world deployment of PeopleSoft Enterprise Campus Solutions, a database application for academic course management. The schema is shown in Figure 1.1. Here, each Course entity has multiple Sections (lectures, precepts, labs, etc.), each Section has multiple Instructors, and so on. In the original application, a web-based interface exposes many different views of the underlying data, for instance, "a list of sections by instructors, each section showing its associated course", or "a list of courses, each course showing its reading list, grading scheme, and sections, each section showing its meetings and assigned instructors". As described above, both of these views are hierarchical, since they call for lists contained within lists. The first view could just as easily be displayed as a flat table, like that returned by a simple join query in SQL. This is not the case for the second view, however. Since the second view involves parallel joins on multiple unrelated relationships, i.e. Grading Scheme and Reading List, such joins would produce an extremely large number of rows of mostly repeated data. Whenever we wish to represent an entity together with related entities from two or more plural relationships, hierarchical views become a necessity.

Because traditional spreadsheet UIs provide no easy way to create joined views of related tables, it becomes impractical to follow good practices of schema normalization, which call for tables of redundantly represented data to be decomposed into multiple smaller ones. This in turn exposes all the usual problems associated with managing improperly normalized data [5].

While hierarchical views can often be represented in spreadsheets using clever formatting tricks (e.g. indentation, skipped cells, or comma-separated lists-in-cells), this strategy has a number of problems. First, such views must be manually created and, if more than one view of the same data is desired, kept in sync with the original data. Second, it does not generalize well, with the sheet structure quickly growing unwieldy as relationships fan out and cascade. For instance, while a comma-separated list in a cell might be easy enough



A8 GER 520, ART 587

	A	B	C	D	E	F	G	H	I
1	Courses			Reading List		Sections/Meetings			
2	Codes	Dist. Area	Title	Author Name	Title	Code	Day	Time	Place
3	CLG 214	EC	Seminar: Wisdom, Sophistry, Philosophy:	Kirk, Raven, Schofield	The Presocratic Philosophers	C01	T	13:30	FIRES B06M
4				Plato	Rival Lovers, Theages,	C01	Th	13:30	FIRES B06M
5				Kerferd	The Sophistic Movement				
6				Lloyd	Magic, Reason and				
7									
8	GER 520, ART 587		Topics in Literary and Cultural Theory	Robert Vischer	On the Optical Sense of Form	S01	M	13:30	MCCKH 363
9				Niklas Luhmann	The Reality of the Mass Media				
10				Walter Benn	The Shape of the Signifier				
11				Mary Ann Doane	The Emergence of Cinematic				
12				Richard Powers	Galatea 2.2				
13				Melanie Klein	Notes on Some Schizoid				
14									
15	KOR 107		Intermediate Korean II	Klear	Integrated Korean	C01	M	11:00	FRIST 207
16				Klear	Integrated Korean Workbook	C01	T	11:00	FRIST 207
17						C01	Th	11:00	FRIST 207
18						C02	M	12:30	FRIST 205
19									
20	MAE 332		Aircraft Design	D.P. Raymer	Aircraft Design: A Conceptual	L01	M	13:30	FRIEN 008
21				T.H.G. Megson,	Aircraft Structures for	L01	W	13:30	FRIEN 008
22									
23	MAT 554		Algebraic Geometry			C01	M	13:30	FINEH 601

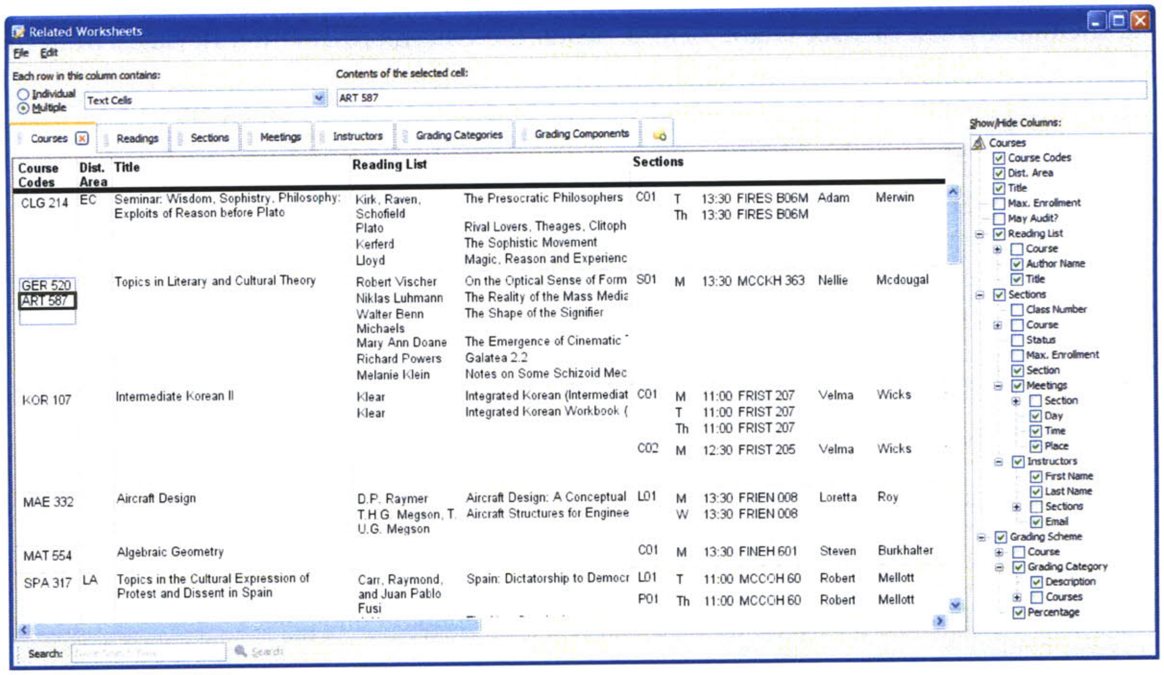
Figure 1.2: A particular hierarchical view of a dataset with the schema shown in Figure 1.1, simulated in a traditional spreadsheet using various formatting tricks.

to maintain, a list of lists is not. Third, when the data representation deviates from the standard tabular format, also known as first normal form (1NF), key spreadsheet features such as sorting, inserting, charting, filtering, or even simple navigation between individual units of data become hard or impossible to apply correctly. For instance, the structure of the spreadsheet shown in Figure 1.2 makes it hard to properly delete or insert items in the Reading List subtables, impossible to automatically sort courses by department or distribution area, and dangerous to hide columns from the view as it might make the user unaware of layout constraints that must be preserved when the data is updated. In the end, spreadsheet users are forced to choose between long-term manageability, achieved by storing data in properly normalized form, and short-term productivity, achieved by accepting whatever compromises are necessary to be able to view related data in one place.

While spreadsheets fall short when it comes to managing multiple related tables of data, they are great for managing individual ones. Spreadsheets afford a large range of streamlined facilities for working with any data that can be arranged in a grid of cells, including multiple selection, copy/paste, find/replace, undo/redo, inserting and deleting, extending data values, sorting and filtering on arbitrary fields, navigating and selecting cells with the keyboard, and so on. Tailor-made database UIs seldom reach this level of sophistication and maturity, due to the limited resources available to develop an application for only one specific schema. When it comes to general editing tasks on tabular data, spreadsheet systems have an advantage even over most tailor-made applications.

1.3 Our Contribution

In this thesis, we propose to reduce the cost-usability gap between spreadsheets and tailor-made relational database applications by extending the spreadsheet paradigm to let the user establish relationships between rows in related worksheets as well as view and navigate the hierarchical cell structure that arises as a result. Implementing this idea, we present Related Worksheets, a spreadsheet-like desktop application which provides relationship editing facilities similar to those often developed with tools like FileMaker or Microsoft Access, but which work from a single generic user interface rather than requiring a new one



Course Codes	Dist. Area	Title	Reading List		Sections					
CLG 214	EC	Seminar: Wisdom, Sophistry, Philosophy: Exploits of Reason before Plato	Kirk, Raven, Schofield Plato Kerferd Lloyd	The Presocratic Philosophers Rival Lovers, Theages, Clitoph The Sophistic Movement Magic, Reason and Experienc	C01	T Th	13:30 13:30	FIRES B06M FIRES B06M	Adam	Merwin
GER 520 ART 587		Topics in Literary and Cultural Theory	Robert Vischer Niklas Luhmann Walter Benn Michaels Mary Ann Doane Richard Powers Melanie Klein	On the Optical Sense of Form The Reality of the Mass Medi The Shape of the Signifier The Emergence of Cinematic Galatea 2.2 Notes on Some Schizoid Mec	S01	M	13:30	MCCKH 363	Nellie	Mcdougal
KOR 107		Intermediate Korean II	Klear Klear	Integrated Korean (Intermediat Integrated Korean Workbook (	C01	M T Th	11:00 11:00 11:00	FRIST 207 FRIST 207 FRIST 207	Velma	Wicks
					C02	M	12:30	FRIST 205	Velma	Wicks
MAE 332		Aircraft Design	D.P. Raymer T.H.G. Megson, T. U.G. Megson	Aircraft Design: A Conceptual Aircraft Structures for Enginee	L01	M W	13:30 13:30	FRIEN 008 FRIEN 008	Loretta	Roy
MAT 554		Algebraic Geometry			C01	M	13:30	FINEH 601	Steven	Burkhalter
SPA 317	LA	Topics in the Cultural Expression of Protest and Dissent in Spain	Carr, Raymond, and Juan Pablo Fusi	Spain: Dictatorship to Democr	L01	T	11:00	MCCOH 60	Robert	Mellott
					P01	Th	11:00	MCCOH 60	Robert	Mellott

Figure 1.3: A hierarchical view similar to that of Figure 1.2, as presented by the Related Worksheets system.

to be developed for every conceivable database schema. Unlike in a regular spreadsheet, the user can model complex database schemas in their proper normalized form without losing the ability to view and edit related data in one place, ensuring long-term managebility of the data. Figure 1.3 shows one of the hierarchical views mentioned earlier, as handled by Related Worksheets. While our prototype provides a simple Excel-style search function, future designs should include more sophisticated selection and filtering capabilities. Future versions should also be able to present hierarchical views using alternative layouts such as the record views often found in tailor-made database applications.

We evaluate our Related Worksheets system through a user study on 36 regular Microsoft Excel users. First-time users of our software were consistently able to solve lookup-type query tasks with the same or higher accuracy than subjects using Microsoft Excel, in one case 40% faster on average.

In Chapter 2, we give an overview of several categories of related work in commercial as well as academic contexts: the user interface tradition associated with tailor-made database applications, tools that have been created to ease development of such applications, and previously proposed spreadsheet approaches. In Chapter 3, we describe our

Related Worksheets system in detail. In Chapter 4, we present the user study we designed to evalute the system and discuss its results. We provide our final conclusions in Chapter 5.

Chapter 2

Related Work

We observe two general strategies for helping the average information worker solve highly domain-specific database tasks. The first assumes that a new application must be built for every unique database schema, and aims to reduce this effort by providing better development tools. The second aims instead to improve the functionality of generic data management applications—spreadsheets, for instance—such that these can replace tailor-made software in a larger number of cases. While the system presented in this paper follows the latter strategy, it is worthwhile to first mention how traditional database UIs have been shaped by the development tools that are used to make them.

2.1 Database UI Builder Tools

Visual tools for development of tailor-made database applications first emerged in the early 1980s, and have since become a standard genre of office productivity software on par with spreadsheets, word processors, and presentation software. These tools provide visual form designers, wizards, and such to help speed up the development process. Among the first such tools were 4th Dimension[1] (1984) and FileMaker[2] (1985); Microsoft Access[3] appeared later (1992). These three applications are still on the market and are maybe the most well-known examples of their kind today. They are all very similar in design and op-

[1]`http://www.4d.com`
[2]`http://www.filemaker.com`
[3]`http://office.microsoft.com/access`

eration (see [1]). There also exist open source clones such as Kexi[4] and OpenOffice Base[5], as well as Software-as-a-Service (SaaS) products like App2You [7] and Intuit QuickBase[6], which provide similar functionality in a web-based interface.

Especially in the more recent web-based development tools, we see some potentially significant usability improvements on the application developer's side, in particular with respect to how well the development environment manages to hide technical details such as join queries, foreign key relationships, and the underlying relational schema. For instance, the application design processes of both AppForge [13] and App2You are form-driven rather than schema driven, that is, the systems use the forms (*pages*) designed by the developer to derive the underlying relational schema rather than requiring them to be defined explicitly upfront. Another innovation is a blurring of the traditional distinction between editable forms and read-only *reports*; in both of the above systems it is possible to produce multi-level hierarchical tables that support at least limited editing of the data originally used to generate them. AppForge has a particularly flexible visual interface for defining such hierarchical views. However, these systems are application builders rather than general data manipulation tools: a new application must be built for every schema, and the actual applications produced are just as schema-dependent as any other tailor-made database application. Furthermore, neither project attempts to create a spreadsheet-like environment; these are purely form- and widget-based user interfaces.

2.2 Traditional Database UIs

While we have seen both improvements and increased diversity among the tools used to develop tailor-made database UIs, the class of UIs produced has itself remained quite homogeneous. Again and again we see the same paradigm, outlined in Figure 2.1. A main menu (*switchboard*) or tab row allows the user to navigate to one of several *table* views, each usually corresponding to a table in the underlying relational database schema or an entity set in the schema's entity-relationship diagram [4]. The table views typically have

[4]`http://www.kexi-project.org`
[5]`http://www.openoffice.org/product/base.html`
[6]`http://quickbase.intuit.com`

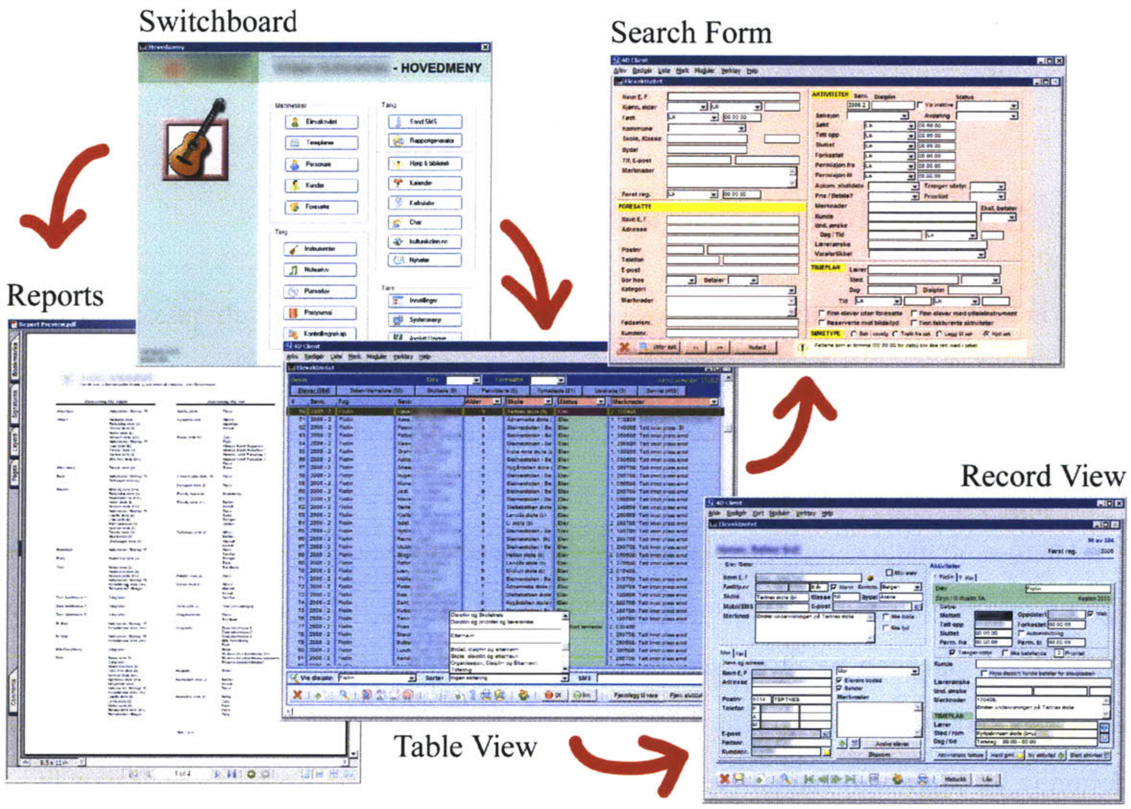


Figure 2.1: The stereotypical user interface of a database application made with FileMaker, Microsoft Access, or similar software development tools. (The screenshots are from a Norwegian CRM system for public music schools.)

a very limited set of editing facilities relative to those found in a spreadsheet, if they are editable at all. From the table view, the user can enter a *search* form to restrict the displayed results, or enter a *record* view to edit and display more detailed information about a single item in the table. The search form and the record view typically include a hard-coded selection of fields corresponding to the columns of the underlying relational table and some of its directly or indirectly related tables. Plural relationships are typically edited through one or more small table views contained within a record view—*subforms* or *portals*. Depending on the capabilities of the development tool used, it may or may not be possible to view and edit data more than a single level away from the currently viewed table or record. For instance, neither FileMaker nor Access allow subforms to contain other subforms; effectively, the hierarchical views that can be created are limited to a depth of one. This is not the case for 4th Dimension, which nevertheless limits editing to the top level. Still, each system allows its applications to provide a basic mechanism for viewing and editing plural relationships.

2.3 Spreadsheet Approaches

It is now widely acknowledged that user errors in spreadsheets are as ubiquitous as spreadsheets themselves, and that these ultimately lead to costly mistakes in organizations worldwide [9]. Much of existing spreadsheet research is motivated by a desire to eliminate such errors.

There is an increasing tendency among commercial spreadsheet applications to encourage users to organize data in simple, flat tables like those found in a relational database. Such tables can then be processed in various mechanical ways with less of a chance of errors caused by misinterpreted data (e.g. sorting only part of a table due to a skipped column). In Microsoft Excel[7] (versions 2007 and later), the user can create designated table areas within a spreadsheet by selecting a range of tabular data an invoking a Format as Table command. Specifying a table in such an explicit manner improves the visual formatting of the table and, more importantly, enables a number of affordances for sorting and filtering the table as well as for selecting entire columns or rows of data in the table. If the user scrolls down while the cursor is inside the table area, the labels in the table heading (first row) will replace the standard column letters (A B C etc.) that are usually shown immediately above the worksheet. A number of other table-related features are similarily enhanced by the presence of explicitly defined table areas. The spreadsheet application in Google Docs[8] enables a single frozen header row by default in new spreadsheets, suggesting that the user should fill the sheet with simple tabular data starting with a heading for each column in the first row. This enables features like sorting and automatic survey-style form generation without any further configuration. Apple iWork Numbers[9] changes the spreadsheet paradigm more radically by adding a level of indirection. Whereas in Excel and Google Docs, each spreadsheet document (*workbook*) consists of one or more *work-*

[7]`http://office.microsoft.com/excel`
[8]`http://docs.google.com`
[9]`http://www.apple.com/iwork/numbers`

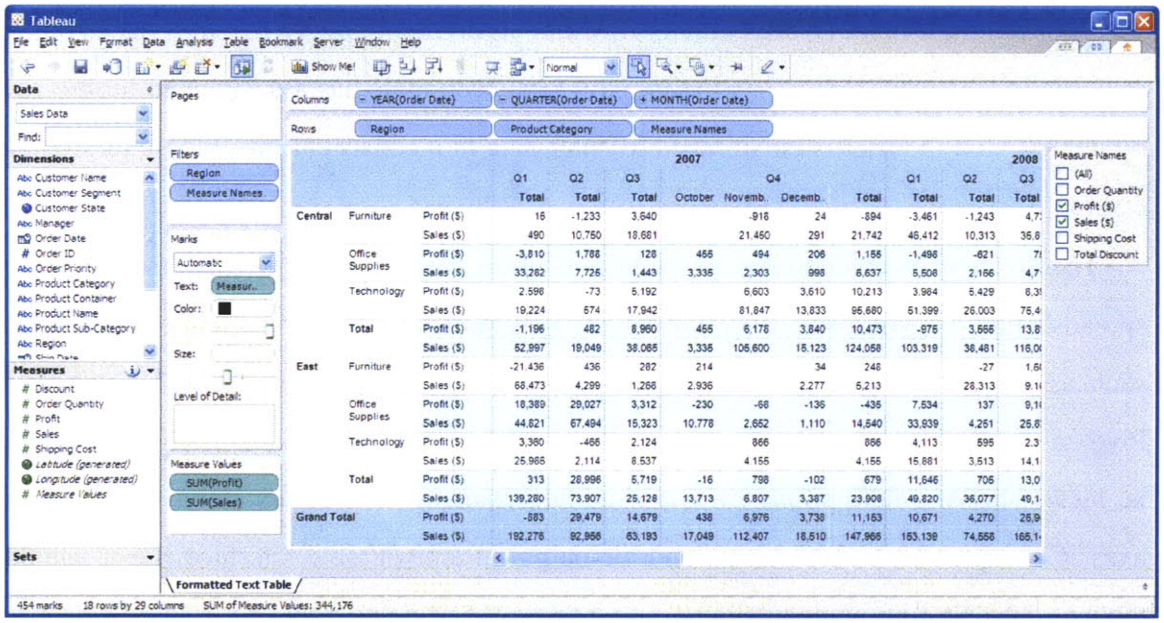
sheets of infinite extent, iWork Numbers documents consist of *pages* that may each contain multiple independent *tables*. As in Google Docs, each table has a designated header row by default, and the user is encouraged to enter data in a consistent, machine-interpretable fashion.

The concept of *pivot tables*, pioneered in Lotus Improv (1993), exists today both in spreadsheets like Microsoft Excel as well as Online Analytical Processing (OLAP) tools such as Tableau [11] (formerly Polaris) and Oracle Business Intelligence Discoverer[10]. Pivot tables are interactive user interfaces for producing cross-tabulated views of data, that is, views showing data points grouped and aggregated in categories along two different axes. Conceptually, both the input and the output of a pivot table interface is a single flat table of data. Starting with an empty output table, the user can drag one or more attributes originally found in the input table from a palette onto each of three axes on the output table. Attributes dropped onto the row or column axes of the output table are used to group and aggregate the attributes dropped onto the third *measure* axis. The former attributes become *dimensions* and the latter become *measures*. If more than one attribute is dropped onto either the row or the column axis, the groups become more specific, and may be labeled in a hierarchical fashion in the axis label. The data in the output table, however, remains in its flat table form; pivot tables do not have the ability to provide hierarchical views of the user's data.

Depending on the tool used, the input data to a pivot table may be stored as a table in a spreadsheet, as one or more tables in a relational database, or as a data cube in an OLAP database. If the data is stored across multiple tables in a relational datbase, the user must select the tables and configure the appropriate join conditions before the pivot table interface can be used, possibly through a custom SQL query, but usually through some sort of wizard provided by the analysis tool.

As an example of a pivot table interface, consider the Tableau system, shown in Figure 2.2. In the configuration shown, sales and profit measures are aggregated in groups along the sub-divided time dimension on the horizontal axis and along the Region and Product Category dimensions on the vertical axis. The axis labels appear hierarchical, but the actual

[10]`http://www.oracle.com/technetwork/developer-tools/discoverer`


Columns: YEAR(Order Date), QUARTER(Order Date), MONTH(Order Date)
Rows: Region, Product Category, Measure Names

			2007								2008		
			Q1	Q2	Q3	Q4				Q1	Q2	Q3
			Total	Total	Total	October	Novemb..	Decemb..	Total	Total	Total	Total
Central	Furniture	Profit ($)	16	-1,233	3,640		-918	24	-894	-3,461	-1,243	4,7
		Sales ($)	490	10,750	18,681		21,450	291	21,742	46,412	10,313	35,8
	Office Supplies	Profit ($)	-3,810	1,788	128	455	494	206	1,155	-1,496	-621	7
		Sales ($)	33,262	7,726	1,443	3,335	2,303	998	6,637	5,508	2,166	4,7
	Technology	Profit ($)	2,596	-73	5,192		6,603	3,610	10,213	3,984	5,429	8,3
		Sales ($)	19,224	674	17,942		81,847	13,833	95,680	51,399	26,003	75,4
	Total	Profit ($)	-1,196	482	8,960	455	6,178	3,840	10,473	-975	3,565	13,8
		Sales ($)	52,997	19,049	38,065	3,335	105,600	15,123	124,058	103,319	38,481	116,0
East	Furniture	Profit ($)	-21,436	436	282	214		34	248		-27	1,6
		Sales ($)	68,473	4,299	1,268	2,936		2,277	5,213		28,313	9,1
	Office Supplies	Profit ($)	18,389	29,027	3,312	-230	-68	-136	-435	7,534	137	9,1
		Sales ($)	44,821	67,494	15,323	10,778	2,652	1,110	14,540	33,939	4,251	25,8
	Technology	Profit ($)	3,360	-466	2,124		866		866	4,113	595	2,3
		Sales ($)	25,965	2,114	8,537		4,155		4,155	15,881	3,513	14,1
	Total	Profit ($)	313	28,996	5,719	-16	798	-102	679	11,646	706	13,0
		Sales ($)	139,280	73,907	25,128	13,713	6,807	3,387	23,908	49,820	36,077	49,1
Grand Total		Profit ($)	-883	29,479	14,679	438	6,976	3,738	11,153	10,671	4,270	26,9
		Sales ($)	192,276	92,956	63,193	17,049	112,407	18,510	147,966	153,139	74,558	165,1

Measure Names: ☐ (All) ☐ Order Quantity ☑ Profit ($) ☑ Sales ($) ☐ Shipping Cost ☐ Total Discount

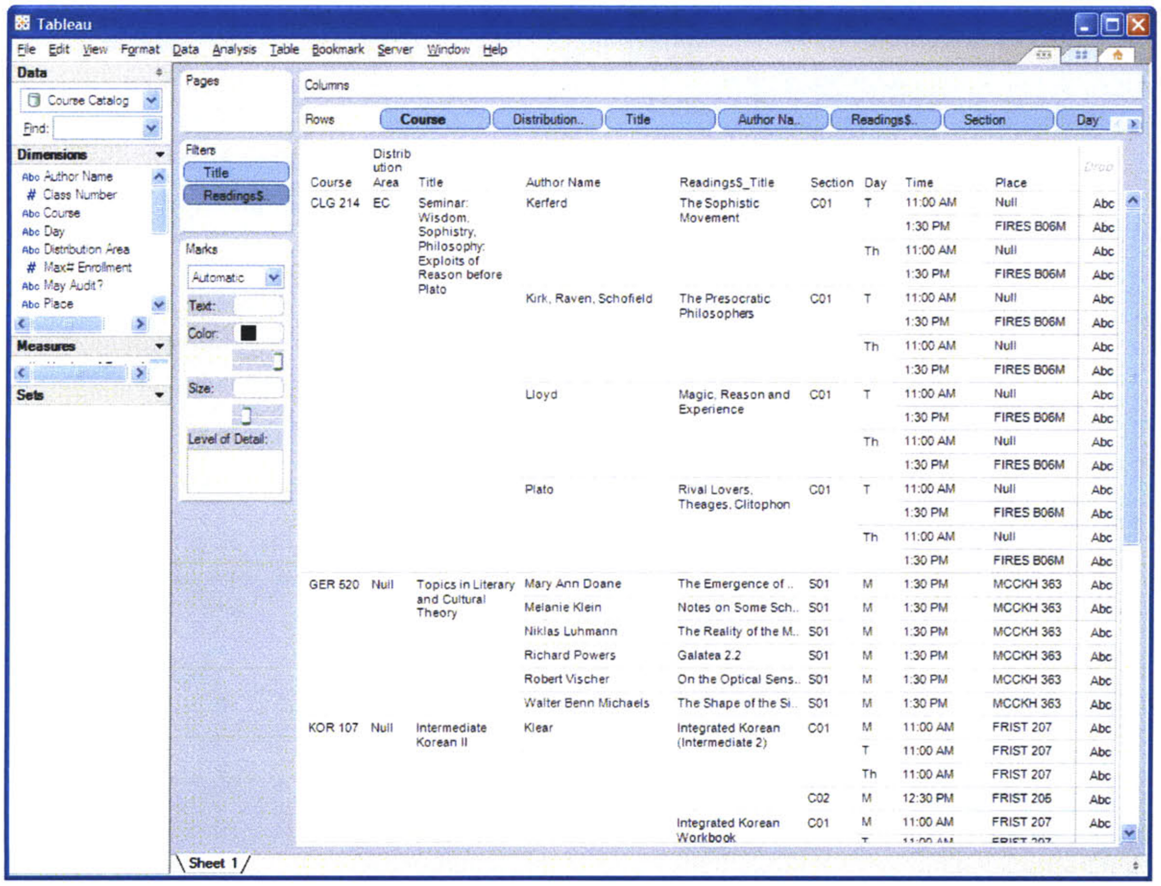
454 marks 18 rows by 29 columns SUM of Measure Values: 344,176

Figure 2.2: A typical pivot table interface and use case, as exposed by the Tableau system. Data is cross-tabulated along two axes. While the group labels are hierarchical, the actual data is presented in a standard, flat table.

data is in flat table form. Like any query system that limits its results to flat tables, pivot tables are thus insufficient for presenting entities relating to other entities through multiple parallel one-to-many relationships. This is illustrated in Figure 2.3, where we have attempted to recreate the course catalog view of Figures 1.3 and 1.2 using a pivot table in Tableau. The view may appear hierarchical, but is in fact simply a flat table with adjacent duplicate values hidden; this does not prevent duplicates from appearing elsewhere. In the example given, the entire section list of each course is repeated once for every item in the course's reading list, and vice versa.

Another class of proposed spreadsheet extensions relaxes the restriction that worksheets must be simple two-dimensional grids of cells in rows and columns (first normal form). In IceSheets [12], cells in a worksheet can recursively contain entire new worksheets, allowing data to be addressed and organized in a hierarchical fashion. However, the content of the hierarchy is static and must be built manually by the user, and the system has no concept of joins or relationships, making it unsuitable for the kind of database tasks we are interested in.

PrediCalc [6] approaches the problem of maintaining consistency between multiple



Course	Distribution Area	Title	Author Name	Readings$_Title	Section	Day	Time	Place	
CLG 214	EC	Seminar: Wisdom, Sophistry, Philosophy: Exploits of Reason before Plato	Kerferd	The Sophistic Movement	C01	T	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
						Th	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
			Kirk, Raven, Schofield	The Presocratic Philosophers	C01	T	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
						Th	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
			Lloyd	Magic, Reason and Experience	C01	T	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
						Th	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
			Plato	Rival Lovers, Theages, Clitophon	C01	T	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
						Th	11:00 AM	Null	Abc
							1:30 PM	FIRES B06M	Abc
GER 520	Null	Topics in Literary and Cultural Theory	Mary Ann Doane	The Emergence of ..	S01	M	1:30 PM	MCCKH 363	Abc
			Melanie Klein	Notes on Some Sch..	S01	M	1:30 PM	MCCKH 363	Abc
			Niklas Luhmann	The Reality of the M..	S01	M	1:30 PM	MCCKH 363	Abc
			Richard Powers	Galatea 2.2	S01	M	1:30 PM	MCCKH 363	Abc
			Robert Vischer	On the Optical Sens..	S01	M	1:30 PM	MCCKH 363	Abc
			Walter Benn Michaels	The Shape of the Si..	S01	M	1:30 PM	MCCKH 363	Abc
KOR 107	Null	Intermediate Korean II	Klear	Integrated Korean (Intermediate 2)	C01	M	11:00 AM	FRIST 207	Abc
						T	11:00 AM	FRIST 207	Abc
						Th	11:00 AM	FRIST 207	Abc
					C02	M	12:30 PM	FRIST 206	Abc
				Integrated Korean Workbook	C01	M	11:00 AM	FRIST 207	Abc

Figure 2.3: An attempt to recreate the view of Figures 1.2 and 1.3 using Tableau's pivot table interface. Section information is materialized once for every item in the courses' reading lists, and vice versa. While the system hides adjacent duplicates, the data is still presented in a flat table.

tables in a spreadsheet when the items in the tables are related by plural relationships, including the case where there are temporary or partial inconsistencies in the constraints. It does not, however, extend the structure of the spreadsheet itself beyond two-dimensional cell grids, and thus does not support hierarchical views.

Chapter 3

The Related Worksheets System

We now describe the Related Worksheets user interface and prototype implementation.

3.1 Worksheet Interface

Related Worksheets is a desktop application which, in its initial state, appears largely like a regular spreadsheet. The user can create, open, and save documents known as *workbooks*, and each workbook contains a number of infinitely sized *worksheets* that can be created and selected through a row of tabs. Empty worksheets start out looking like those of Excel or any other spreadsheet application, and can be navigated and edited using standard mouse and keyboard commands (click cells to select, navigate cells with arrow keys, type to edit, drag to resize columns etc.). Since our system's novel features depend on the creation of multiple worksheets, and since worksheets are referred to by name in various parts of the user interface, we force the user to create the first worksheet explicitly rather than providing one by default when the application starts. See Figures 3.1 and 3.2.

Like Google Docs and Apple iWork Numbers, we intend the user to populate each worksheet with a single table of items under a designated row of column headers. Thus, the first row of each worksheet is automatically formatted as a heading (with a bolded font and a thick border underneath) as well pinned to remain visible at the top of the worksheet view during vertical scrolling of the rest of the worksheet. See Figure 3.3 for an example of a properly entered table of simple, primitive values. Columns can be renamed at any time

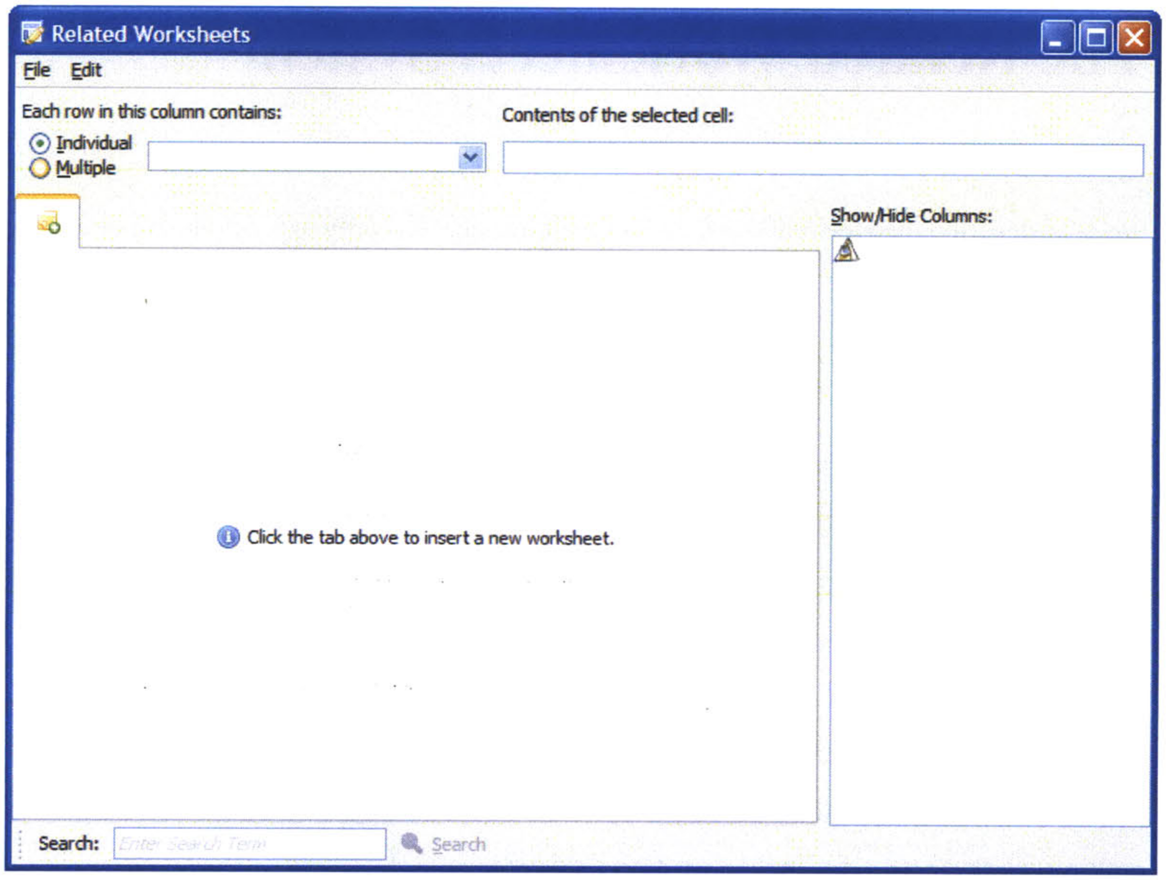

Related Worksheets
File
Edit
Each row in this column contains:
Individual
Multiple
Contents of the selected cell:
Show/Hide Columns:
Click the tab above to insert a new worksheet.
Search:
Enter Search Term
Search

Figure 3.1: Related Worksheets at startup.

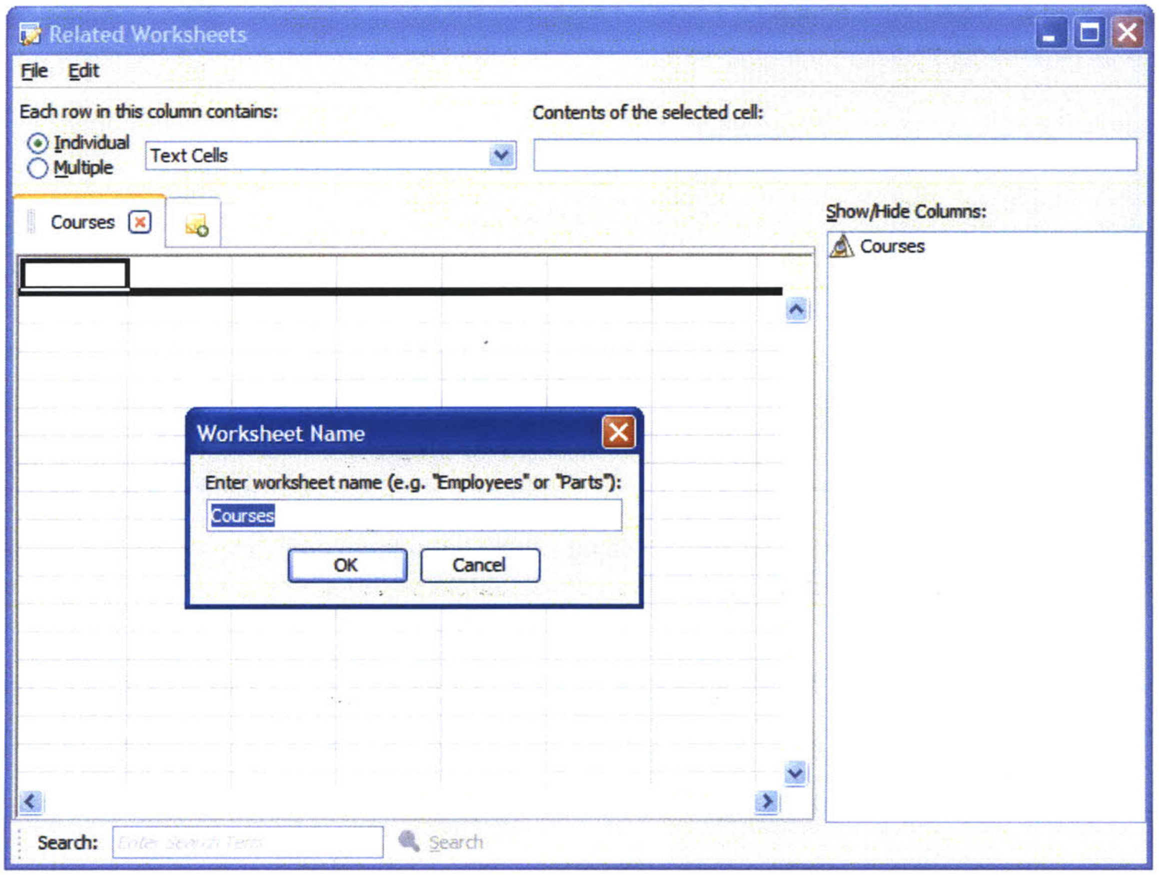
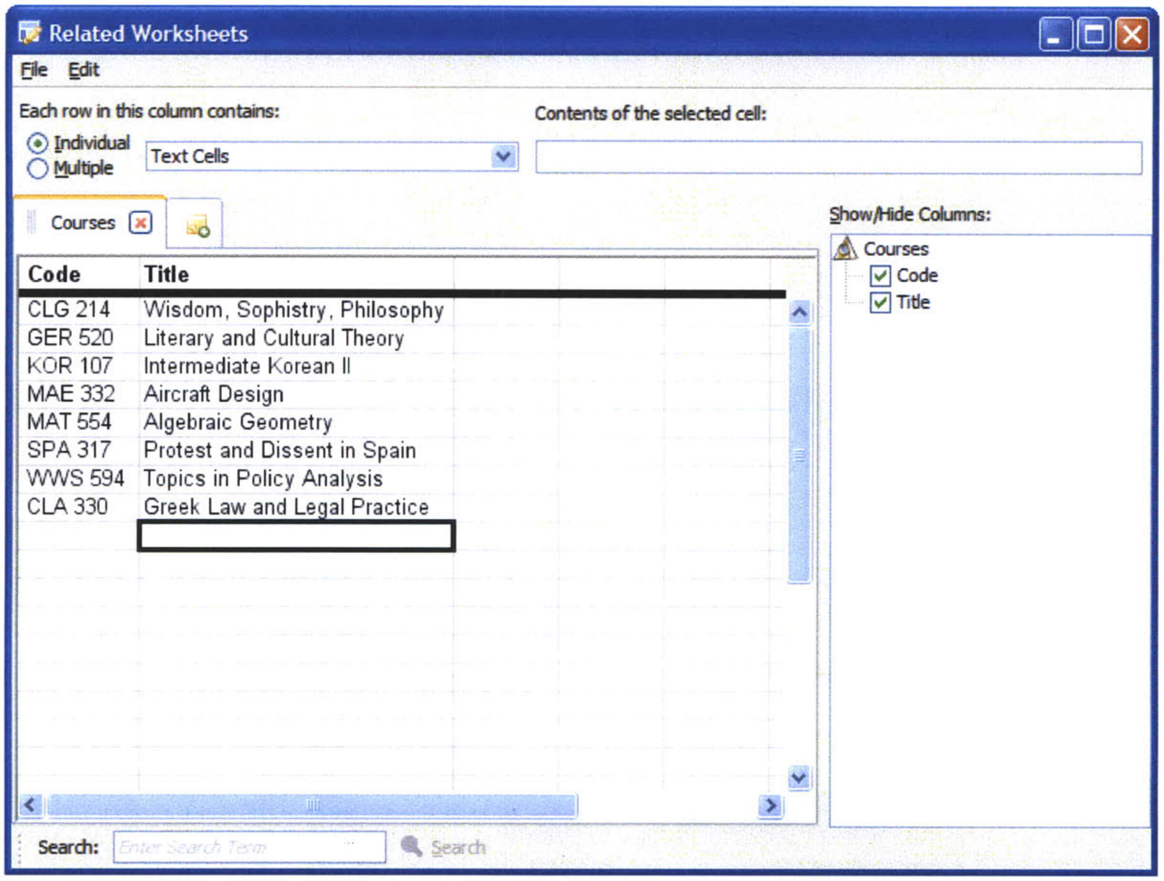


Figure 3.2: Related Worksheets after creating a single new worksheet, also showing the dialog used to set the name of the worksheet.



Each row in this column contains: [✓ Individual] Multiple

Text Cells

Contents of the selected cell:

Code	Title
CLG 214	Wisdom, Sophistry, Philosophy
GER 520	Literary and Cultural Theory
KOR 107	Intermediate Korean II
MAE 332	Aircraft Design
MAT 554	Algebraic Geometry
SPA 317	Protest and Dissent in Spain
WWS 594	Topics in Policy Analysis
CLA 330	Greek Law and Legal Practice

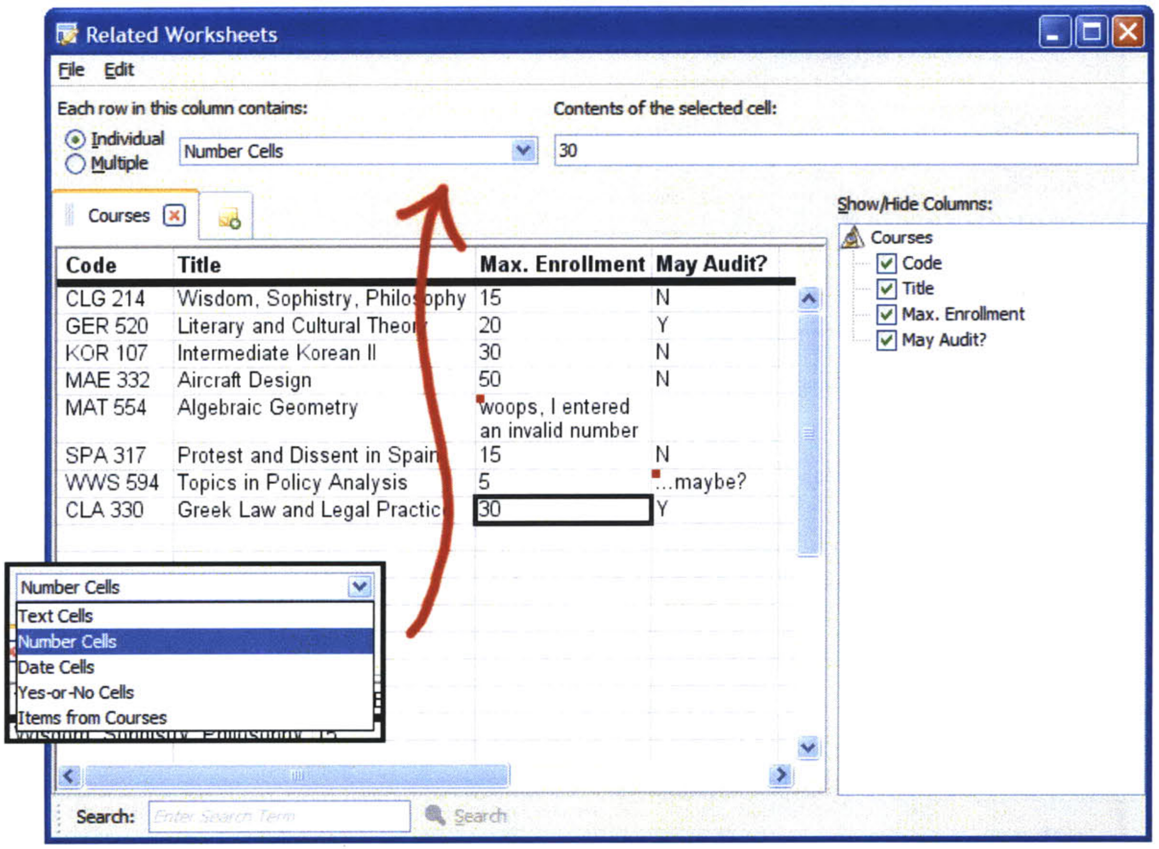
✓ Code
✓ Title

Figure 3.3: Related Worksheets after entering simple tabular data into a worksheet. Column widths have been adjusted to fit the data in each column without wrapping.

by editing the contents of their respective header row cells, and can be deleted or moved left or right through designated keyboard commands (an improved implementation would expose these features through the menu system and/or drag-and-drop).

3.2 Column Data Types and Cardinalities

In our system, each column in a worksheet has an explicit preferred data type associated with it. The user may change this type at any time using a drop-down menu in the toolbar area. If the value of a cell does not conform to its column's selected data type—e.g. text in a Number column or an invalid date in a Date column—the cell is rendered with a red warning box similar to those used by PrediCalc [6] and Excel to indicate other kinds of inconsistencies. See Figure 3.4. The user is always free to keep the default Text type, which



Each row in this column contains: [✓ Individual] Multiple — Number Cells

Contents of the selected cell: 30

Show/Hide Columns: Courses — ✓ Code, ✓ Title, ✓ Max. Enrollment, ✓ May Audit?

Code	Title	Max. Enrollment	May Audit?
CLG 214	Wisdom, Sophistry, Philosophy	15	N
GER 520	Literary and Cultural Theo	20	Y
KOR 107	Intermediate Korean II	30	N
MAE 332	Aircraft Design	50	N
MAT 554	Algebraic Geometry	woops, I entered an invalid number	
SPA 317	Protest and Dissent in Spain	15	N
WWS 594	Topics in Policy Analysis	5	...maybe?
CLA 330	Greek Law and Legal Practic	30	Y

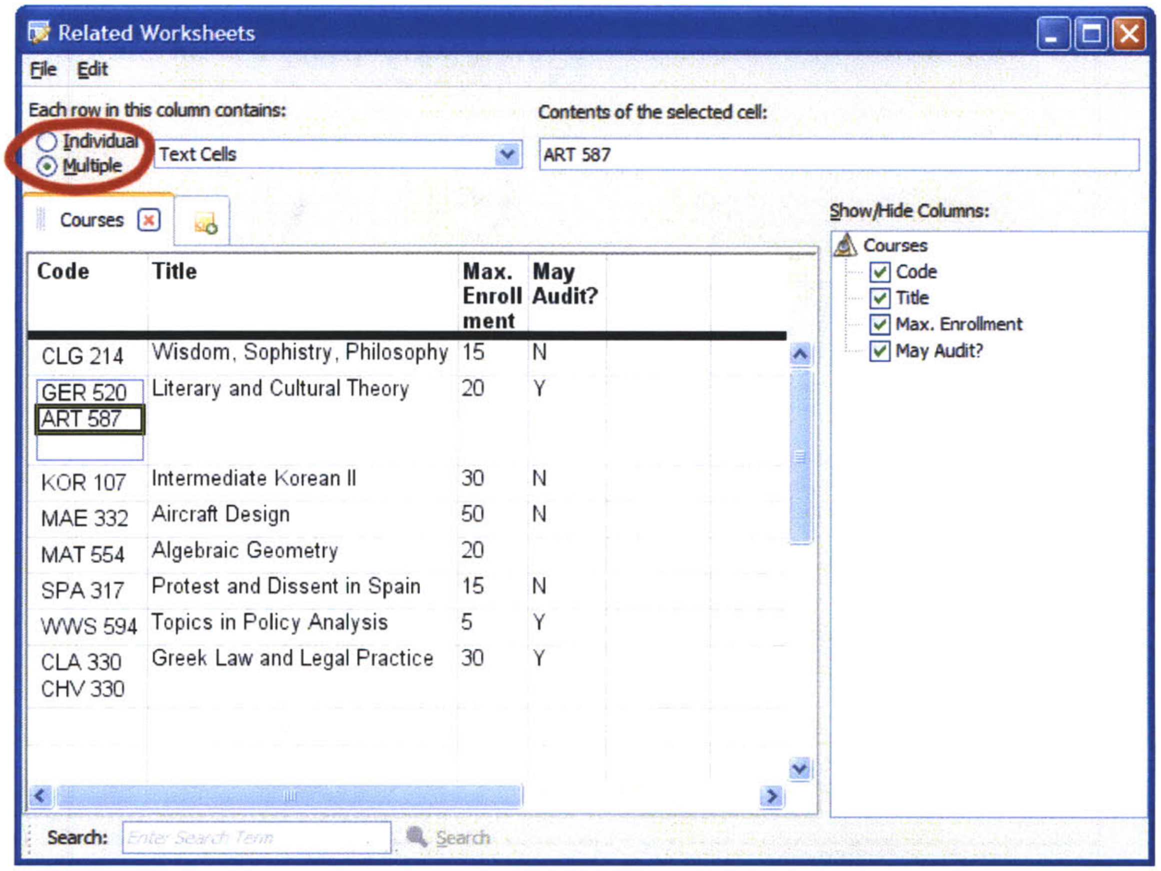
Number Cells
- Text Cells
- [✓ Number Cells]
- Date Cells
- Yes-or-No Cells
- Items from Courses

Figure 3.4: Related Worksheets after adding more columns of primitive data and selecting Number Cells and Yes-or-No Cells as the columns' preferred data types. Some of the values entered do not conform to their columns' data types, and have been marked with a red warning box by the software.

accepts any primitive value.

The system's weak form of type checking serves to allow temporary inconsistencies. For instance, it is possible to paste data from Excel without first having to set the appropriate column types, and the data type of existing columns can be changed without special conversion semantics. The type checking functionality applies not only to primitive data types, but also to reference types and column cardinalities, which we discuss next. Note that in the header row, cells always behave as they would in a Text column.

In addition to a preferred data type, each column also has a preferred cardinality, selected by a pair of radio buttons next to the data type drop-down. Columns preferring *individual* cells per row behave just like columns in a traditional spreadsheet; this is the default. The *multiple* cardinality, however, indicates that the column accepts multiple cells


Related Worksheets

File Edit

Each row in this column contains: Individual [✓ Multiple] Text Cells

Contents of the selected cell: ART 587

Courses

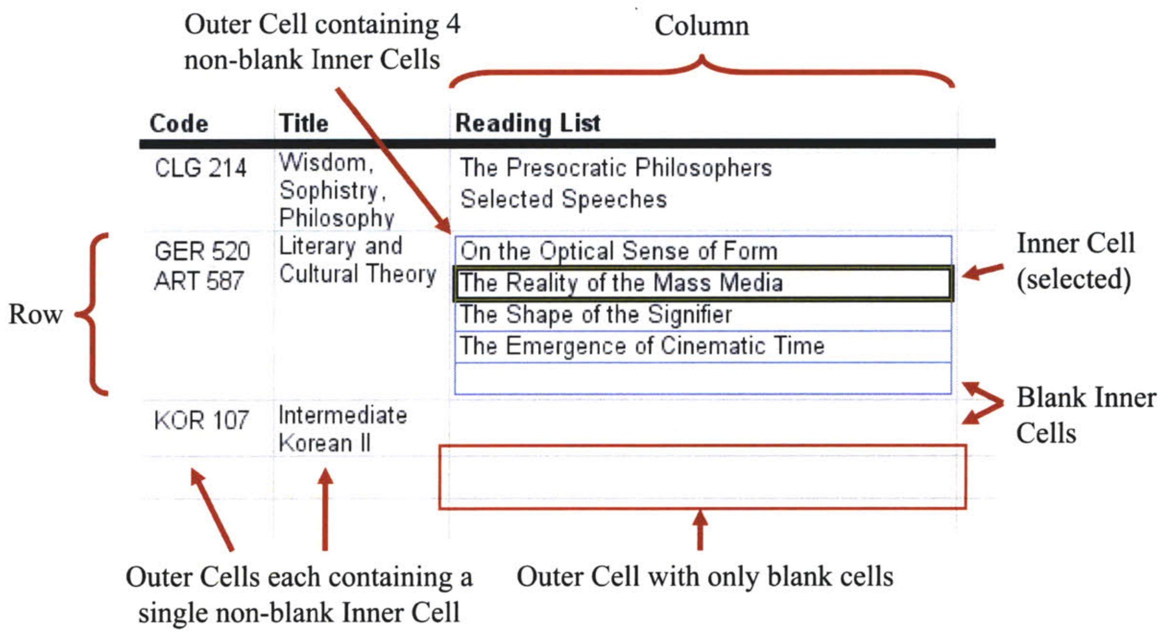
Code	Title	Max. Enrollment	May Audit?
CLG 214	Wisdom, Sophistry, Philosophy	15	N
GER 520 ART 587	Literary and Cultural Theory	20	Y
KOR 107	Intermediate Korean II	30	N
MAE 332	Aircraft Design	50	N
MAT 554	Algebraic Geometry	20	
SPA 317	Protest and Dissent in Spain	15	N
WWS 594	Topics in Policy Analysis	5	Y
CLA 330 CHV 330	Greek Law and Legal Practice	30	Y

Show/Hide Columns:
- Courses
  - ✓ Code
  - ✓ Title
  - ✓ Max. Enrollment
  - ✓ May Audit?

Search: Enter Search Term Search

Figure 3.5: Related Worksheets after the Code column has been set to allow multiple inner cells and some additional course codes have been entered. The outer cell containing the currently selected inner cell is rendered with a blue border, and with one extra blank inner cell as an affordance for adding additional entries.

per row. This situation is illustrated in Figure 3.5. It now becomes necessary to distinguish between *outer* and *inner* cells; in general, outer cells are defined by the intersection of a row and a column, while inner cells are contained within outer cells. Data values are contained in inner cells, and the cursor moves left/right/up/down between the inner cells that are visible on the screen at any time. Outer cells are always rendered to show the contents of each of their non-blank inner cells, and always with at least one inner cell visible, even if blank. For better scalability, large collections of inner cells might be rendered in a scrollable pane, though we did not implement this. In columns of multiple cardinality, a light gray border is added around inner cells as a visual indicator of the special behavior of these cells. When the cursor enters such a cell, the border is highlighted in blue and an extra blank inner cell is rendered within the same outer cell as an affordance to enter additional


Outer Cell containing 4 non-blank Inner Cells
Column
Inner Cell (selected)
Row
Blank Inner Cells
Outer Cells each containing a single non-blank Inner Cell
Outer Cell with only blank cells

Code	Title	Reading List
CLG 214	Wisdom, Sophistry, Philosophy	The Presocratic Philosophers Selected Speeches
GER 520 ART 587	Literary and Cultural Theory	On the Optical Sense of Form The Reality of the Mass Media The Shape of the Signifier The Emergence of Cinematic Time
KOR 107	Intermediate Korean II	

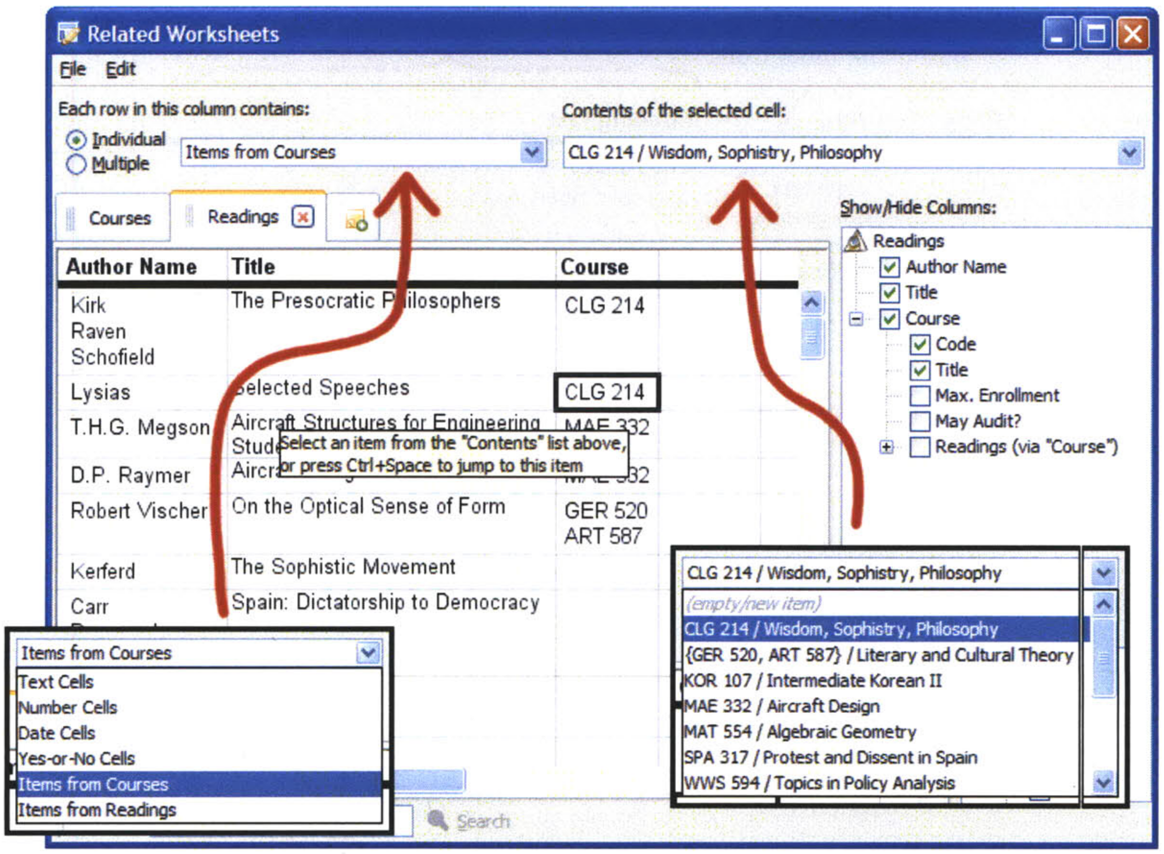
Figure 3.6: Terminology for describing worksheets in the Related Worksheets system.

values. See Figure 3.6.

3.3 Reference Types and Relationships

Besides the primitive column data types, our system provides reference types. For each worksheet in the open workbook, including the currently selected one, a reference type Items from *(name of worksheet to be referenced)* is available in the column data type drop-down, below the standard primitive types. Each inner cell in a column of such a type is allowed to contain, instead of a primitive value, a reference to a row in the target worksheet specified by the column type. When the cursor enters an inner cell in a reference column, the user can select a reference value from another drop-down list. The drop-down shows one entry per non-blank row in the referenced worksheet, plus an option to reset the cell to its default empty state.

As an example, suppose the user is modeling an academic course management system like that of Figure 1.1. In two separate worksheets, the user may have created tables of Courses and Readings, respectively, and now wishes to establish a relationship between the two in order to maintain a reading list for each course. The user can do this by creating



Each row in this column contains: (● Individual ○ Multiple) Items from Courses

Contents of the selected cell: CLG 214 / Wisdom, Sophistry, Philosophy

Author Name	Title	Course
Kirk Raven Schofield	The Presocratic Philosophers	CLG 214
Lysias	Selected Speeches	CLG 214
T.H.G. Megson	Aircraft Structures for Engineering Stud	MAE 332
D.P. Raymer	Aircr	MAE 332
Robert Vischer	On the Optical Sense of Form	GER 520 ART 587
Kerferd	The Sophistic Movement	
Carr	Spain: Dictatorship to Democracy	

Select an item from the "Contents" list above, or press Ctrl+Space to jump to this item

Show/Hide Columns:

- Readings
  - [✓] Author Name
  - [✓] Title
  - [✓] Course
    - [✓] Code
    - [✓] Title
    - [ ] Max. Enrollment
    - [ ] May Audit?
    - [ ] Readings (via "Course")

Items from Courses
- Text Cells
- Number Cells
- Date Cells
- Yes-or-No Cells
- Items from Courses
- Items from Readings

CLG 214 / Wisdom, Sophistry, Philosophy
- (empty/new item)
- CLG 214 / Wisdom, Sophistry, Philosophy
- {GER 520, ART 587} / Literary and Cultural Theory
- KOR 107 / Intermediate Korean II
- MAE 332 / Aircraft Design
- MAT 554 / Algebraic Geometry
- SPA 317 / Protest and Dissent in Spain
- WWS 594 / Topics in Policy Analysis

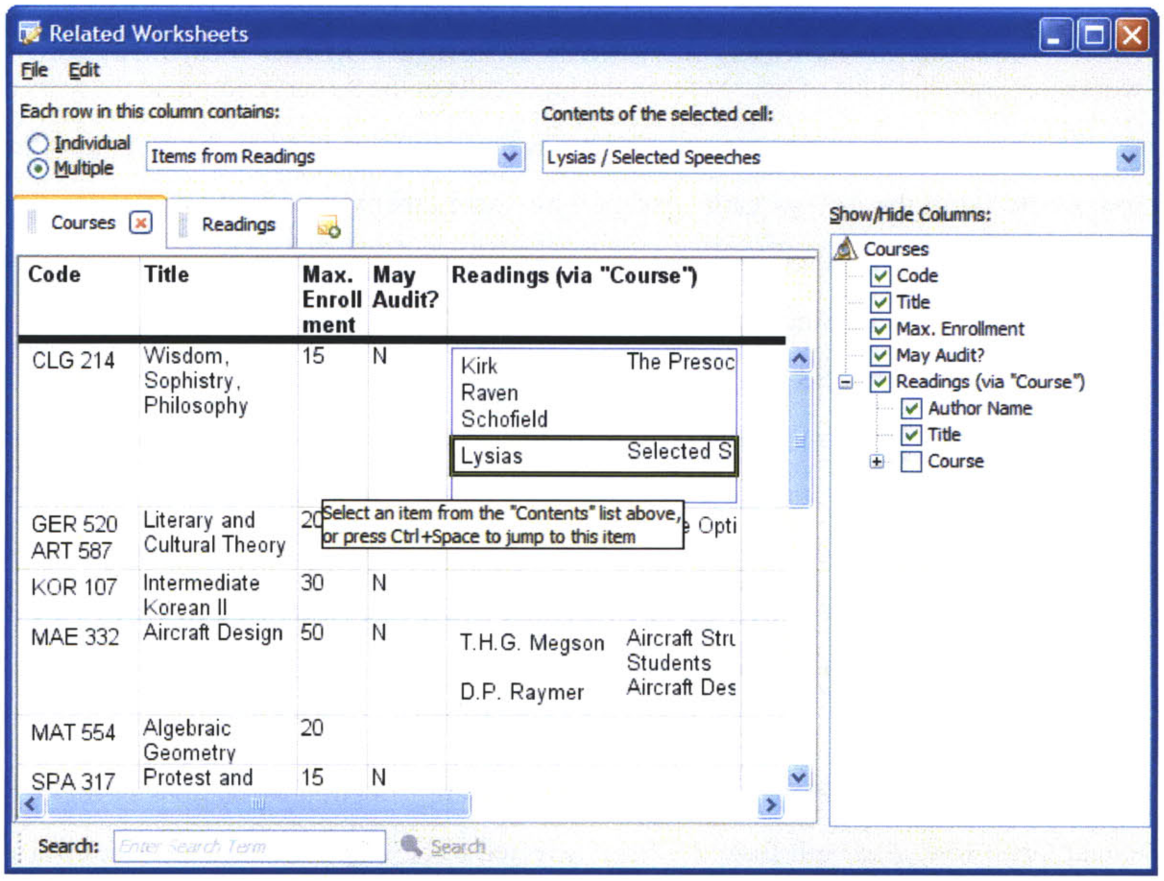
Figure 3.7: Related Worksheets after adding a new worksheet with data. The Course column has been set to contain references to the Courses worksheet by selecting Items from Courses as the column's preferred data type. Reference values for cells in this column can now be selected from a drop-down list (right) that shows one entry for each row in the Courses table.

a column Course of type Items from Courses to the Readings worksheets. This column will then contain, for each reading, the course containing the reading in its reading list, and the user can select the appropriate course from the reference selection drop-down. See Figure 3.7. Alternatively, the user can create a column of type Items from Readings in the Courses worksheet—we will explain this symmetry later. Note that *creating* a column simply means making use of one of the infinitely many available empty columns, or reusing an existing one, like in a regular spreadsheet.

When an inner cell in one worksheet contains a reference to a row in another, a representation of the referenced row is itself rendered within the referencing cell. This representation is a recursive selection of columns from the referenced row which the user can configure through the Show/Hide Columns tree of checkboxes, similar to the Schema Navigation Menu found in AppForge [13]. For instance, in Figure 1.3, each row in the Courses worksheet contains a list of references in the inner cells of the Sections column to the rows of the Sections worksheet. Each section reference is then rendered just like the corresponding row would be in the Sections table, except with some otherwise visually distracting borders removed, and with some of its columns hidden according to the user's selection in the Show/Hide Columns tree. Since each section also has a list of references to Meetings as well as to Instructors, this process is applied recursively until primitive values are rendered in the base case. Effectively, the user is able to perform arbitrary nested joins on the relationships between the worksheets, starting from any given worksheet as a base table. Note that each worksheet has its own recursive Show/Hide Columns configuration; for instance, a course can be shown with its full title when viewed under the Courses worksheet, while only showing its six-letter course code when viewed through several levels of references from the Instructors worksheet.

The entries in the reference selection drop-down are string representations of each potential target row, also built from the fields selected to be visible in the Show/Hide Columns tree, including nested ones, or whatever portion of the string there is horizontal space enough to display. Though not implemented in our prototype system, users should be able to pick items from this drop-down list by typing the first letters of this string representation, using a combo box instead of a list box. Thus, for instance, users could choose to pick a reference to the Course table either by starting to type in a course code, a course title, or even a related field such as the instructor of the course (in cases where there is only one instructor per course). See again Figure 3.7. An alternative design of the drop-down menu could be to render each entry as it would appear in the actual reference cell. The string representation, however, has the advantage that each entry in the drop-down list can be displayed in the space of a single line of text. Thus, the list can display more items at a time, and will behave like the standard list or combo box widgets that users are already familiar with.

Relationships are bidirectional. When the user creates a reference column in one worksheet in order to maintain a relationship with another worksheet, a corresponding reference column is automatically created in the referenced worksheet, refering back to the original



Related Worksheets

File Edit

Each row in this column contains: ○ Individual [✓ Multiple] Items from Readings

Contents of the selected cell: Lysias / Selected Speeches

Courses | Readings

Code	Title	Max. Enrollment	May Audit?	Readings (via "Course")
CLG 214	Wisdom, Sophistry, Philosophy	15	N	Kirk / Raven / Schofield — The Presoc; Lysias — Selected S
GER 520 ART 587	Literary and Cultural Theory	20		Opti
KOR 107	Intermediate Korean II	30	N	
MAE 332	Aircraft Design	50	N	T.H.G. Megson — Aircraft Stru Students; D.P. Raymer — Aircraft Des
MAT 554	Algebraic Geometry	20		
SPA 317	Protest and	15	N	

Select an item from the "Contents" list above, or press Ctrl+Space to jump to this item

Show/Hide Columns:
- Courses
  - [✓] Code
  - [✓] Title
  - [✓] Max. Enrollment
  - [✓] May Audit?
  - [✓] Readings (via "Course")
    - [✓] Author Name
    - [✓] Title
    - [ ] Course

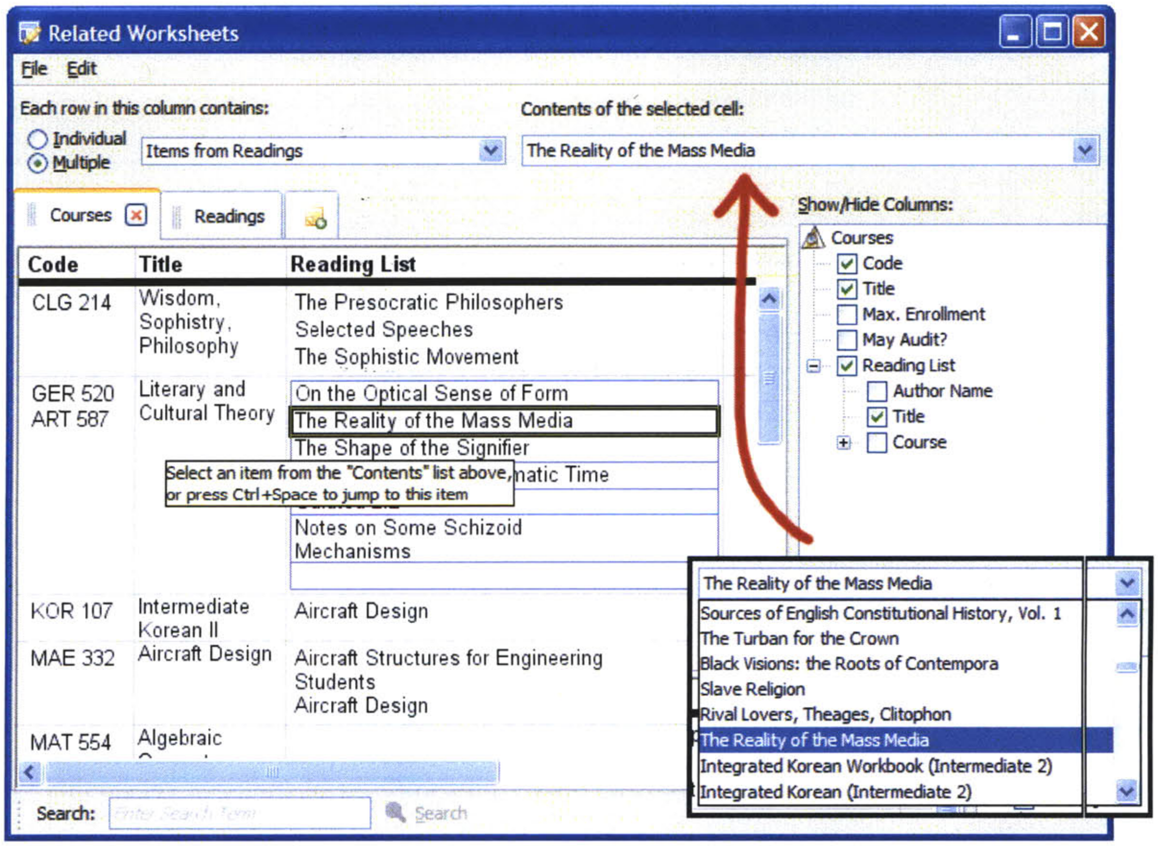
Search: Enter Search Term Search

Figure 3.8: Related Worksheets showing the reverse reference column that was automatically created in the Courses table when the Course column of the Readings table was set to refer to the former. By pressing Ctrl+Space, the user can teleport between the cursor location selected here versus that of Figure 3.7.

worksheet. For instance, if each Reading is assigned to a Course, then each Course has a set of Readings. When the type Items from Courses is selected for the Course column of the Readings worksheet, this automatically creates a corresponding column of type Items from Readings in the Courses worksheet. See Figure 3.8.

When a reverse reference column is first created, it is assigned a default name of the form *(name of the other worksheet)* (via "*(name of the corresponding reference column in the other worksheet)*"). In our example, the name would be Readings (via "Course"). This name is only an initial default and, as for any other column, can be changed at any time. Like any other column, the reverse reference column can be hidden from the view using the Show/Hide Columns tree. Deleting the reverse reference column deletes the original referencing column as well. Note that the system itself makes no distinction between what we have been referring to as *forward* and *reverse* reference columns; we are simply using these terms to distinguish between the columns explicitly created by the user versus those automatically made available in the other direction.

By setting the cardinalities of a related pair of forward and reverse reference columns, the user can choose between one-to-one, one-to-many, many-to-one, and many-to-many relationships. By default, an automatically created reverse reference column is set to have the multiple cardinality, resulting in a one-to-many relationship if the forward reference column was of the individual cardinality, or a many-to-many relationship if the forward reference column was of the multiple cardinality. Note that our system is able to model many-to-many relationships directly, without an intermediate join table, as long as no attibutes are required on the relationship itself. This is an important usability feature, as the



Each row in this column contains:
Individual
[✓ Multiple]
Items from Readings

Contents of the selected cell:
The Reality of the Mass Media

Code	Title	Reading List
CLG 214	Wisdom, Sophistry, Philosophy	The Presocratic Philosophers Selected Speeches The Sophistic Movement
GER 520 ART 587	Literary and Cultural Theory	On the Optical Sense of Form The Reality of the Mass Media The Shape of the Signifier matic Time Notes on Some Schizoid Mechanisms
KOR 107	Intermediate Korean II	Aircraft Design
MAE 332	Aircraft Design	Aircraft Structures for Engineering Students Aircraft Design
MAT 554	Algebraic	

Select an item from the "Contents" list above, or press Ctrl+Space to jump to this item

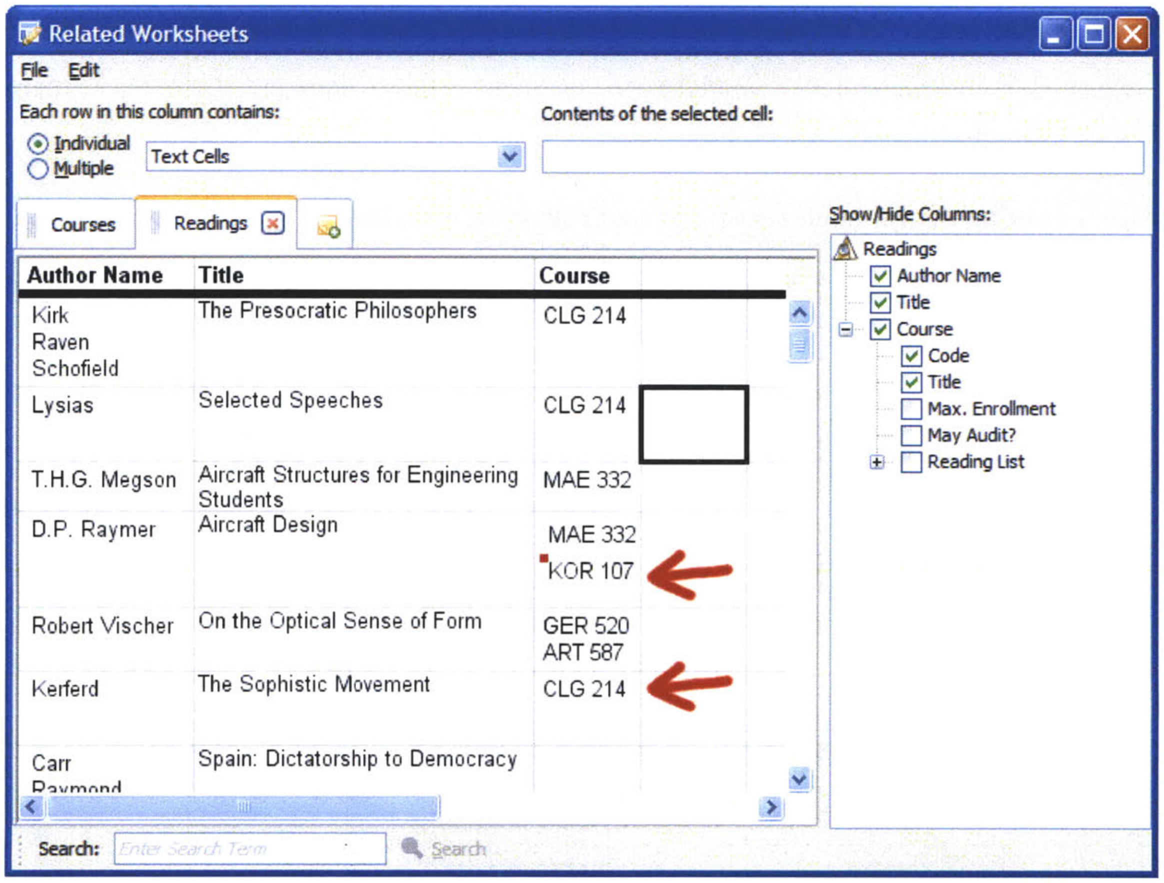
Courses
- [✓] Code
- [✓] Title
- [ ] Max. Enrollment
- [ ] May Audit?
- [✓] Reading List
  - [ ] Author Name
  - [✓] Title
  - [ ] Course

Search: Enter Search Term

The Reality of the Mass Media

Sources of English Constitutional History, Vol. 1
The Turban for the Crown
Black Visions: the Roots of Contempora
Slave Religion
Rival Lovers, Theages, Clitophon
The Reality of the Mass Media
Integrated Korean Workbook (Intermediate 2)
Integrated Korean (Intermediate 2)

Figure 3.9: Related Worksheets after editing the worksheet shown in Figure 3.8. The reference column has been renamed and resized, the Author Name column of the referenced Readings worksheet has been hidden, some top-level columns have been hidden, and some additional entries have been added to the reading lists of the different courses.



Each row in this column contains: [✓ Individual] Multiple — Text Cells

Contents of the selected cell:

Author Name	Title	Course
Kirk Raven Schofield	The Presocratic Philosophers	CLG 214
Lysias	Selected Speeches	CLG 214
T.H.G. Megson	Aircraft Structures for Engineering Students	MAE 332
D.P. Raymer	Aircraft Design	MAE 332 KOR 107
Robert Vischer	On the Optical Sense of Form	GER 520 ART 587
Kerferd	The Sophistic Movement	CLG 214
Carr Raymond	Spain: Dictatorship to Democracy	

Show/Hide Columns:

- Readings
  - ✓ Author Name
  - ✓ Title
  - ✓ Course
    - ✓ Code
    - ✓ Title
    - ☐ Max. Enrollment
    - ☐ May Audit?
    - ☐ Reading List

Figure 3.10: Related Worksheets showing the Readings worksheet again as it appears after the edits in Figure 3.9 have been carried out in the other worksheet. The references to the Readings worksheet that were added from the Courses worksheet are now visible in the reverse direction as well. However, since the Course column has been set to allow only a single entry per row, a red warning box is shown for the incorrect assignment of Aircraft Design to both courses MAE 332 and KOR 107.

concept of join tables is abstract and may make little sense to a spreadsheet user without experience in schema design for relational databases.

Updates made to a reverse reference column are reflected back to the original referencing column. See Figures 3.9 and 3.10. One-to-many relationships may be violated by selecting the same reference more than once in the column of multiple cardinality. In this case, the reverse column with the individual cardinality marks the references beyond the first with the red warning box seen before, see Figure 3.10. In future versions of the software it would also make sense to show a warning in the forward reference cell, and to explain such warnings in a tooltip pop-up.

For the purposes of selecting columns to display in the Show/Hide Columns tree, forward and reverse references can be followed indefinitely. For instance, it is possible to show, for each instructor, the courses taught by that instructor (via the Sections worksheet), then for those courses, what other instructors teach the course, then for those instructors, the complete set of courses taught by those instructors, and so on. Infinite recursion is avoided by expanding the Show/Hide Columns tree lazily and hiding, by default, columns that have already been instantiated as an ancestor in the tree. Like any other column, these can be shown or hidden again by checking or unchecking the appropriate checkbox in the tree.

While references are rendered recursively and the user can easily see across many cascading and parallel relationships simultaneously, we note that the cell cursor moves across the top level only. This means that in order to edit a particular value, whether it be a primitive or a reference, the user must first select the worksheet that actually contains that value directly, that is, the worksheet that contains the value in one of its inner cells. For instance, in Figure 3.7, it is not possible to edit the course code or title of a course without first switching to the Courses worksheet tab. This is a similar restriction to those found in tailor-made database applications made with FileMaker or Microsoft Access. However, generalizing a standard idiom often found in such applications, we introduce a *teleport* feature to make such navigation quick and convenient.

The teleport feature allows the user to quickly jump between multiple related worksheets via the references that exist between them. With the cursor located on a cell containing a reference to a row in another worksheet, the user can press Ctrl+Space to switch to the related worksheet, with the cursor placed on the inner cell containing the corresponding reverse reference in the referenced row. After teleporting, the user is free to move the cursor around in the newly selected worksheet, for instance to edit values that were previously too deep in the hierarchy to select with the cursor, or even to teleport another degree away from the original worksheet. To get back to the original point, the user can move the cursor back to the original reverse reference that was selected after the teleport, and teleport again. Consequently, teleporting twice without changing the cursor location will always lead back to the original cell. We believe a regular set of back/forward buttons, not implemented in our prototype, would complement this feature well. Moreover, the teleport feature should be accessible through a mouse-driven affordance as well as the existing keyboard shortcut.

As an example of the teleport feature, consider the situation in Figure 3.7. A user who wishes to edit the title of the course CLG 214 can press Ctrl+Space to get to the state shown in Figure 3.8. The user can now move the cursor to and edit the cell containing the Wisdom, Sophistry, Philosophy primitive value. When done, the user can move the cursor back to the cell containing the Lysias / Selected Speeches reference and teleport to get back to the original location in Figure 3.7.

Last, we provide a simple search feature accessible from a search bar similar to that found in the Mozilla Firefox[1] web browser. By entering a text string and clicking the Search button, the cursor will move to the next cell whose string representation contains the search term. The string representation is the same as that used by the reference selection drop-down item, minus special characters; it contains the primitive values of all the columns that have been recursively selected for display in the Show/Hide Columns tree.

3.4 Refactorization Features

We now consider the problem of converting data between the flat table format used in relational databases and the reference-based format preferred in the Related Spreadsheets system. We used the techniques described below to convert a sample MySQL database to the Relational Spreadsheets format in preparation of our user study.

In a relational database, data is stored in multiple flat tables of primitive values, and relationships are represented by foreign key relationships, where primitive values in one table identify rows in another. It is trivial to import such tables from a relational database as worksheets in Related Worksheets; however, the system will not be aware of any relationships between these tables until foreign keys have been converted to reference values. To perform this conversion, in either direction, we have implemented two special column manipulation commands which seem likely to have significant utility beyond testing and development of the software itself. One command, Detect References, replaces each value in a column with a reference to a row in a target worksheet that contains the value in question

[1]`http://www.mozilla.com/firefox`

in its first visible column (as selected in the Show/Hide Columns tree), if a unique such row exists. This is typically used to convert a column of primitive values to a column of reference values, for instance after having imported two flat tables related through a foreign key relationship from a relational database. The other command, Replace References, works oppositely; it replaces each reference value in a column with the value of the first visible column in the referenced row. Besides flattening existing Related Worksheet databases, the Replace References can be used to convert the join tables commonly found in relational database schemas into natively supported many-to-many relationships.

Consider the task of converting a column of primitive values to a column of references to another worksheet, as we did ourselves to convert the foreign key relationships of a flat MySQL database into the relationships naturally supported by Related Worksheets. We first change the data type of the column from the primitive type to the appropriate reference type, e.g. from Text to Items from Courses. This does not not have any immediate effect on the data in the column, except for the appearance of red warning boxes in the cells containing primitive values to indicate that these values have not yet been converted to the new preferred data type. However, as for any reference column, it now becomes possible to choose a selection of subcolumns from the target table, e.g. Courses, in the Show/Hide Columns selection tree. The Detect References command uses the first visible column from the related worksheet as a field to join on, so the user should now make the primary key column of the target table the first visible one, and then apply the command. At this point, the command goes through each inner cell of the column to be converted and replaces the existing primitive values with reference values by looking for a unique matching value in the related worksheet's first visible column. If no value or multiple values are found, the original value is left as it is, with the warning box still showing. Otherwise, it is replaced by a reference to the row in the related worksheet containing the matching column value.

The Replace References command works oppositely: to convert a column of reference values to a column containing the contents of one of the referenced rows' values, we first select the column to remain in the Show/Hide Columns selection tree, then invoke the command, and finally change the type of the column to the appropriate new one. Depending on the column selected to remain, the values can be of either a primitive type or a reference type. As an example of a use case for the Replace References command, other than simply reversing the effect of a previously applied Detect References command, consider the problem of eliminating a join table Instructors-Sections(Instructor Name, Section Number) representing a many-to-many relationship between the Instructors(Name, Email, Salary) and Sections(Number, Course, Status) tables, after having imported these tables from a relational database as three separate worksheets. First, we would convert the primitive values in each of the columns of the Instructors-Sections table into reference values using the Detect References command. The Instructors and Sections tables would now each have a reverse reference column called Instructors-Sections (via "Instructor Name") and Instructors-Sections (via "Section Number"), respectively. By invoking the Replace References command on one of these reverse reference columns, all references to the join table are replaced by references to the actual Sections and Instructors worksheet, respectively. As a side-effect of this, the command results in a new reverse reference column in the worksheet opposite of the one to which the command was applied; this column contains the appropriate references in the reverse direction. The join table worksheet can now be deleteted in its entirety without loss of information.

More general versions of the Detect References and Replace References commands should allow references to be detected and replaced using more than one column, though we did not implement this in our prototype. For instance, a worksheet might have a First Name and a Last Name column, and it would be desireable to be able to combine these two columns into a single column of references to another worksheet by joining on both of these fields, since a single field may not be sufficient to uniquely identify a row in the referenced table. Likewise, the Replace References command should be able to convert a single column of references into multiple columns containing values from a selection of columns originally found in the referenced table.

Chapter 4

User Study

We wish to demonstrate that databases containing a multitude of plural relationships can be more efficiently managed in our system than in a regular spreadsheet. That is, we hypothesize that users will be able to perform common database tasks faster when the database, in its appropriate normalized form, is made available through our proposed user interface. Furthermore, we would like to evaluate the learnability of the system; we wish to show that typical spreadsheet users are able to figure out how to perform such tasks without special instructions. We have not studied the process of setting up a new spreadsheet database from scratch.

4.1 Study Design

To test our claims, we conducted an online user study using Amazon Mechanical Turk[1] and the User Study Console, a Java-based tool we developed to allow test subjects to stream timestamped recordings of their computer screens to our server with a minimum of effort. The User Study Console would also automatically download and open sample spreadsheets we had prepared as well as allow the user to submit changed versions of these in exchange for a confirmation code to be pasted into the Mechanical Turk HIT (Human Intelligence Task, a survey form). Depending on the stage and parameters of the experiment, sample spreadsheets would be opened in either Microsoft Excel, assumed to be already installed

[1]`http://www.mturk.com`

on the user's computer, or in our own Related Worksheets application, distributed as part of the same Java Web Start executable.

Our study was a between-subjects design in two stages. The first stage was a recruiting task offered to the general population of Mechanical Turk workers, intended to let us select a group of workers who (1) were willing and able to successfully run the User Study Console on their machines, (2) happened to have Microsoft Excel or a compatible spreadsheet application installed, (3) had demonstrated good faith in solving an otherwise unrelated spreadsheet task, and (4) had provided us with answers to some background and demographic questions. We offered \$0.25 for workers to launch the User Study Console, make a chart based on sample data we provided in an Excel spreadsheet, and fill in our survey. A previous HIT approval rate of 80% or greater was required to accept the HIT, as is the default.

The survey part of the recruiting HIT included questions about gender, age, occupation, previous spreadsheet experience, and previous experience with Microsoft Access, FileMaker, or other database software, if any. Multiple-choice questions relating to previous experience were accompanied by qualitative questions to encourage accurate answers; for instance, people reporting previous experience with database software were asked to give examples of tasks they had performed in this context. The actual data we provided for the recruiting task was a collection of daily currency exchange rates for three currencies since 2006, formatted in a single table such that Excel's charting wizards would be able to produce a nice chart in about three clicks. Since we were not actually interested in the charting task itself, we told workers that "Just about any chart will be good!"

Once workers had submitted the recruiting HIT, we reviewed their submissions, including screencasts, log files, and uploaded spreadsheet files, to ensure we had meaningful answers to both the survey and the charting parts of the task, and to ensure that there had been no technical problems with the User Study Console during the execution of the tasks.

In the second stage of our experiment, Mechanical Turk workers with complete and technically unproblematic submissions for the recruiting task were divided randomly into two groups of equal size, and invited by e-mail to do a longer follow-up HIT, worth \$3.00. Workers in the control group would be given a link to a HIT which, like before, asked them to launch the User Study Console and solve tasks using a Microsoft Excel spreadsheet that we provided. Workers in the treatment group would, on the other hand, be given a link to a different HIT that would ask them to use our Related Worksheets application. From a technical point of view we achieved this separation by assigning a custom Mechanical Turk Qualification with a randomly chosen value between 1 and 99 to each worker, and then setting specific ranges as requirements for both previewing and accepting the HIT (<50 for the Related Worksheets version and ≥ 50 for the Excel version). The qualification was labeled "Spreadsheet Study (randomly assigned qualification)" to reassure users that the numerical value of the qualification was not a measure of their past performance. Since Mechanical Turk does not support the creation of completely unlisted HITs, the metadata (title, description, reward amount, etc.) for the two HITs was visible to the public, but this data was identical for the two HITs (except for the qualification value range).

The tasks and instructions in the two main HITs were identical except in the description of the tool used to solve the tasks. Most significantly, the treatment group HIT had the following note added: "In this part of the study, we will not actually be using Microsoft Excel, but instead a spreadsheet-like application which has some similarities with and some differences from a regular spreadsheet." No actual instructions were provided on how to perform the tasks, neither in the Excel nor the Related Worksheets versions of the HIT. Workers from one group were unable to see the contents of the other HIT, and vice versa.

The actual database tasks modeled in our user study were based on the course management system example introduced earlier. The data was a subset of an actual course database, containing 37 courses and their complete set of related entities. See Figure 1.1. Some attributes, like the lengthy course descriptions, were omitted.

In the Excel version of the tasks, the database was represented as a set of normalized relations, each stored as a table in its own worksheet. See the Introduction for a discussion of why this particular mapping is a natural one for our investigation. We used natural rather than surrogate primary keys, e.g. course codes in the form "KOR 107", to identify the various entities involved. The full normalized schema of the database is shown in Figure 4.1.

For the Related Worksheets version of the task, we imported the same normalized

Courses(Course, Distribution Area, Title, Max. Enrollment, May Audit)
Readings(Course, Author Name, Title)
Sections(Class Number, Course, Status, Max. Enrollment, Section)
Meetings(Section Class Number, Day, Time, Place)
Instructors(First Name, Last Name, Email)
Grading Components(Course, Grading Category, Percentage)
Instructors-Sections(Instructor Name, Section Class Number)
Cross-Listings(Crosslisted Course Code, Primary Course Code)

Figure 4.1: The normalized schema as it appeared in the Excel version of the tasks. Primary keys that participate in foreign key relationships are underlined, foreign keys are double-underlined.

database tables into our system as individual worksheets, converted foreign key columns into reference columns using the Detect References and Replace References operations described earlier, and made default selections in the Show/Hide Columns tree corresponding to what we believed would be seen in a real course management system. In particular, the Courses worksheet was shown with all fields and subfields fully expanded, excluding reverse reference columns; this corresponds to the way entities would be displayed in a real course catalog (each course showing its reading list, grading scheme, sections, meetings, instructors etc.). Each of the other worksheets were shown with all their immediate fields visible, and some selection of important identifying subfields for each related entity (e.g. course code, distribution area, and course title whenever a course was referenced). Effectively, it would not be necessary for a user to show additional columns or subcolumns to solve the tasks given, though depending on window size and screen resolution it might be necessary to scroll horizontally or hide some columns to see the desired information.

The exact tasks given are listed in Table 4.1. The actual instructions given to users also specified the exact form expected of the answers and gave an example of a correctly specified answer. These tasks are lookup (read-only) tasks of the kind we would imagine to be frequently performed on a real-world course management database. We originally included a number of update tasks as well; however, these uncovered a number of discoverability bugs that would have to be fixed before realistic update performance could be measured.

After the expiration of the main HIT, we again reviewed workers' submissions and associated screencasts, log files, and uploaded spreadsheet files. We graded each of the

#	*Description*
1	Find a course that is taught by Harry Morrill.
2	In the course "MUS 105: Music Theory Through Performance and Composition", what percentage of the final grade is derived from the Midterm Exam?
3	Find a course that satisfies the "LA" (Literature and the Arts) distribution area, with a lecture (meetings belonging to a section denoted "L01") that starts after noon (after 12.00).
4	What is the e-mail address of the instructor who teaches "KOR 107: Intermediate Korean II"?
5	Who teaches the precept section of "HIS 383: The United States Since 1920" that meets on Wednesdays at 10am?

Table 4.1: Individual tasks given in the user study.

individual tasks on a pass/fail basis. Last, we manually reviewed all screencasts and estimated the amount of time spent on each individual task using video frame timestamps and observations of the subjects' actions.

4.2 Results

The recruiting stage of the study yielded 54 subjects. The subjects were divided into two groups of 26 subjects each and invited to participate in the main part of the study. Of these, 18 recruits from the Excel group and 18 recruits from the Related Worksheets group completed the main HIT. An additional 4 recruits from the Excel group and 4 recruits from the Related Worksheets group started doing a few of the tasks in the HIT, but soon closed the User Study Console and never submitted the HIT. The remaining recruits never opened the User Study Console.

Of the 36 subjects participating in the main part of the study, 25 were male (69%). Subjects were between 19 and 48 years of age, with first, second, and third quartiles at 26, 30, and 35 years, respectively. 16 subjects (44%) described having some prior experience using "database software such as FileMaker or Microsoft Access". Most users had significant previous spreadsheet experience; Table 4.2 shows the levels of typical spreadsheet use as reported by subjects themselves. The subjects' self-reported occupations are categorized in Table 4.3.

	"How often do you typically use spreadsheet software?"	*N*
1	Never	1
2	Once a year or less	1
3	A couple of times per year	1
4	A couple of times per month	12
5	A couple of times per week	5
6	Almost daily	10
7	Multiple times per day	6
	Total	36

Table 4.2: Number of subjects reporting particular levels of previous spreadsheet experience. The numbers in the left column are used to calculate averages in Table 4.3.

Occupation	*N*	*average spread-sheet experience*
Student, business/finance	2	6.0
Student, mechanical engineering	1	4.0
Student, unknown	1	2.0
Teacher	2	4.0
Business administration	5	5.6
Business process outsourcing (BPO)	3	6.0
Homemaker	5	3.2
Programmer	8	5.8
Other IT	3	5.0
Other technical	5	5.6
Other non-technical	1	4.0
Total	36	5.1

Table 4.3: Number of subjects and average level of previous spreadsheet experience by occupations reported. Averages are based on the 7-step scale in Table 4.2.

		Task #				
	N	*1*	*2*	*3*	*4*	*5*
Excel	18	18	18	14	17	12
RS	18	18	18	14	17	15

Table 4.4: Number of subjects who solved each task correctly.

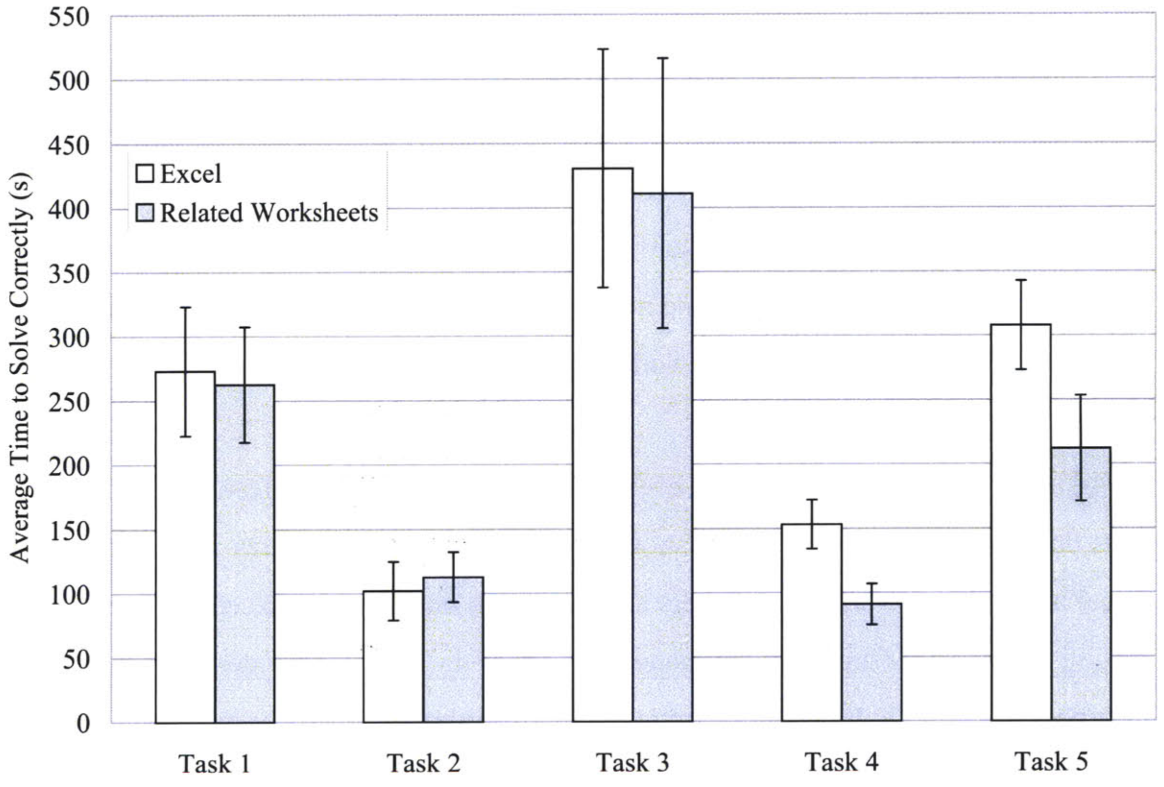


Figure 4.2: Timing measurements for the user study. The error bars show the standard error of the mean.

	Excel			*Related Worksheets*		
Feature	*n*	*N*	*n/N*	*n*	*N*	*n/N*
Search	13	17	76%	9	17	53%
Mark Rows	3	15	20%	2	17	12%
Filter	11	15	73%			
Sort	1	15	7%			
Teleport				8	18	44%
Show/Hide Columns				3	17	18%

Table 4.5: Number of users observed to make use of specific features or techniques during the user study.

Of the subjects who submitted the main HIT, the number who passed each task is shown in Table 4.4. The average times taken to complete each task, for subjects who solved each task in question correctly, are shown in Figure 4.2. Due to intermittent network problems, 5 screencasts from the Excel group and 2 screencasts from the Related Worksheets group were incompletely streamed and could not be used for timing measurements. (With regards to the User Study Console, we have workarounds planned to avoid such problems in future studies.)

Using uploaded screencasts and event logs, we summarized the features used by subjects during the study. This data is shown in Table 4.5. Some features are found in both Excel and Related Worksheets, others are specific to one system. Except for the teleport feature, the numbers are based on manual review of the screencast videos; the number of data points is slightly less than the total number of users due to some incomplete screencasts. The teleport command was invoked 1, 1, 2, 5, 5, 7, 12, and 13 times by each respective user. Of the Related Worksheets subjects making use of the search feature, 3 discovered it only after having progressed about halfway through the study. Of the Excel subjects making use of the filter feature, 3 did not use the sort feature at all. "Mark Rows" refers to subjects visibly working to keep track of rows in a worksheet; this was done in various ways by different users. Two Excel users colored marked rows yellow with the highlighting tool, one of these also at one point wrote down the primary keys of marked rows in an external text editor. A third Excel user marked rows by typing an "x" in the first empty column available next to the table. Two Related Worksheets users marked rows as well; these both used an external editor. Note that we do not know how many of the subjects may have taken notes on paper while working on the tasks.

4.3 Discussion

For each submission we watched the screencasts and measured the time taken to solve each individual task. We then, for each individual task, applied Welch's unpaired two-tailed t-test to the times taken to complete the tasks by those subjects who got them right only. Only Task 4 showed a statistically significant ($p<0.05$) improvement in the Related Worksheets case; it was 41% faster than Excel. Task 5 showed a 31% improvement but was not statistically significant ($p=0.09$). In all other cases the results were insignificant.

4.3.1 Task Analysis

We can analyze the different tasks in terms of the joins involved and tables traversed in the normalized database schema used in the Excel version of the database in each case. It is useful for this purpose to refer back to the Entity-Relationship diagram that was shown in Figure 1.1.

Task 2 involves only the Courses and Grading Components tables. Additionally, since only the primary key is needed from the Courses table, it suffices to look at the normalized Grading Components table to solve the task. Thus, Task 2 is expected to be the simplest one for the Excel group, and not much difference would be expected between the Related Worksheets and Excel groups. This was indeed the case.

Tasks 1 and 4 each involve the Courses, Sections, and Instructors tables. Because Sections and Instructors are related through a many-to-many relationship, a join table Instructors-Sections must also be traversed in the normalized Excel version of the database. In Task 1, only the primary key is needed from the Instructors and Courses tables, so only two tables, Sections and Instructors-Sections, must actually be consulted. In Task 4, the primary key from the Courses table is sufficient, but the e-mail address of an instructor must be looked up, requiring the same two tables to be used plus an additional lookup in the Instructors table. We believe the fact that subjects spent a relatively large amount of time on Task 1 compared to, for instance, Task 4 to be due to the learning time needed to get acquainted with the database and, in the case of the Related Worksheets group, the software itself. For Task 4, we saw a large performance gain for subjects using our system versus those using Excel. We can attribute this to the manual join that must be done between the Sections and Instructors-Sections tables followed by the lookup in the Instructors table in the Excel case. In the Related Spreadsheets case, solving the task was a simple matter of finding the course in the Courses worksheet, which showed all related entities in one place.

Like Task 4, Task 3 involved two joins and three tables. In Task 3, the tables involved are Courses, Sections, and Meetings. Task 3 is harder than Task 4, since there are both multiple meetings per section as well as multiple sections per course; in Task 4, only a single lookup was required in the Instructors table. To solve Task 3 in Excel typically requires the user to keep track of numbers of several sections to look for in the Meetings table while browsing the Sections table. In Related Worksheets, one can simply browse for the appropriate course, though since we do not provide any filter-like capabilities, the user must manually look through the entire list of courses.

Task 5 involved the Sections, Meetings, and Instructors-Sections tables. Since there is often only a single instructor per section, the task is simpler than Task 3. Like in Task 4, we see indications of a performance advantage among users using the Related Worksheets application over those using Excel. While we did not have enough timing data points for the last task to show statistical significance, it seems that the advantage is large but slightly smaller than that of Task 4. This effect could potentially be due to Excel users getting more familiar with the manual process of joining tables.

4.3.2 User Behavior

The most frequently used feature in both the Excel and the Related Worksheets version of the tasks was search, available in both systems. Typically, users of both systems would use a combination of searching and manual browsing—scrolling back and forth, switching between worksheets, and scanning the data visually—to complete the tasks. This was anticipated: the chief advantage of the Related Spreadsheets system is, in this context, the relevancy of the information seen on the screen at any time.

Among the Excel users, we were surprised to find that just about as many subjects made use of the filter feature as made use of the search feature. The filter feature allows the user to select individual items for exclusive display in the table by selecting values from a drop-down list above each table column. Subjects used the filter feature both to search for specific rows in each table and to mark multiple rows of potential interest for later review. Since the drop-down list provided by the filter feature is always sorted, users could quickly locate a specific item in an otherwise unsorted table without using the search feature. As mentioned, three users even made extensive use of the filter function without ever making use of the search feature. Given these observations, we consider a filter feature to be an important potential addition to future versions of our system.

As predicted by the task analysis, users sometimes found it necessary to temporarily mark rows of interest in order to solve tasks involving more than a single join between tables. Some of the users did this in ways visible on the screencasts; see our observations in the Results section. Two Related Worksheets users wrote down primary keys in an external editor, despite this not being necessary for completing the tasks in our system. We assume these users must have believed it would be faster to approach the problem in the same way as they would in a standard spreadsheet rather than spending time learning to understand how the hierarchical views worked.

In the Related Worksheets version of the tasks, it was not necessary to modify the Show/Hide Columns configuration, and most users rightly ignored the feature. A few subjects experimented with the feature, along with column resizing, to hide unnecessary data from view. The teleport feature was tried by 8 of the 18 Related Worksheets subjects; at least 5 of these subjects seemed to make productive use of it throughout the progression of the study, indicating significant potential value in this feature.

Chapter 5

Conclusion

Since the early years of database management systems, we have kept building a new database application for every schema that came along. These resulting *tailor-made* database applications are expensive to develop, hard to maintain, hard to use, and hardly much more than graphical front-ends to their underlying database. By looking to the success of spreadsheets as a general-purpose data management tool, and by developing new visual interfaces to let end-users access the relationship management capabilities traditionally offered only by tailor-made relational database applications from a single unified interface, we can eliminate the pains of using either spreadsheets or tailor-made applications for database tasks, and get the best of both worlds.

We have presented the Related Worksheets system, a spreadsheet-like application that lets the user establish relationships between rows in related worksheets as well as view and navigate the hierarchical cell structure that arises as a result. Notably, the system allows cells in worksheets to contain multiple values, and allows bidirectional relationships to be established between worksheets through the use of reference values. Reference values are rendered using a user-definable recursive selection of visible fields. Together, these features allow database schemas with a multitude of plural relationships to be modeled and managed naturally from within a spreadsheet-like user interface.

Finally, we presented the results of a Mechanical Turk-based user study on 36 regular Excel users in a between-subjects design. We found that first-time users of our system were able to solve lookup-style query tasks with the same or better accuracy than subjects in a control group using Excel, in one case 40% faster on average.

5.1 Future Work

In future work, we plan to extend the user interface paradigm presented in this thesis in three aspects, in order to further approach the level of functionality found in traditional tailor-made database UIs. First, the system must incorporate a more expressive visual query language that allows a larger class of hierarchical views to be built by the user. This will include the ability to apply arbitrary sorting, filtering, nesting, aggregation, and formulas to the entity sets in the database. Second, the system must be able to render the hierarchical views in question using a selection of different layout configurations beyond the simple tabular one. In particular, it must be possible to produce the standard kind of *record view* that is ubiquitous in FileMaker-style applications, in which entities are shown with their attributes vertically arranged and labeled individually. Third, the system must provide a more flexible interface for managing the views themselves; the current model of providing a single view per entity set is insufficient and somewhat arbitrary.

Bibliography

[1] 4th Dimension, MS Access, and FileMaker Pro: A comparison. `ftp://ftp.4d.com/aci_product_reference_library/4d_product_white_papers/4D_Access_FMP_Comparison.pdf`, 2001. 4D marketing white paper.

[2] Serge Abiteboul, Rakesh Agrawal, Phil Bernstein, Mike Carey, Stefano Ceri, Bruce Croft, David DeWitt, Mike Franklin, Hector Garcia Molina, Dieter Gawlick, Jim Gray, Laura Haas, Alon Halevy, Joe Hellerstein, Yannis Ioannidis, Martin Kersten, Michael Pazzani, Mike Lesk, David Maier, Jeff Naughton, Hans Schek, Timos Sellis, Avi Silberschatz, Mike Stonebraker, Rick Snodgrass, Jeff Ullman, Gerhard Weikum, Jennifer Widom, and Stan Zdonik. The Lowell database research self-assessment. *Commun. ACM*, 48(5):111–118, 2005.

[3] Yolande E. Chan and Veda C. Storey. The use of spreadsheets in organizations: determinants and consequences. *Information & Management*, 31(3):119–134, 1996.

[4] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[5] E. F. Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[6] Michael Kassoff and Michael R. Genesereth. PrediCalc: a logical spreadsheet management system. *The Knowledge Engineering Review*, 22(03):281–295, 2007.

[7] Keith Kowalzcykowski, Alin Deutsch, Kian Win Ong, Yannis Papakonstantinou, Kevin Keliang Zhao, and Michalis Petropoulos. Do-It-Yourself database-driven web applications. In *CIDR*, 2009.

[8] J. D. Pemberton and A. J. Robson. Spreadsheets in business. *Industrial Management & Data Systems*, 200(8):379–388, 2000.

[9] Stephen G. Powell, Kenneth R. Baker, and Barry Lawson. A critical review of the literature on spreadsheet errors. *Decision Support Systems*, 46(1):128–138, 2008.

[10] Neil Raden. Shedding light on shadow IT: Is Excel running your business? Technical report, Hired Brains, Inc., January 2005.

[11] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[12] Brian C. Whitmer. Improving spreadsheets for complex problems. Master's thesis, Brigham Young University, August 2008.

[13] Fan Yang, Nitin Gupta, Chavdar Botev, Elizabeth F Churchill, George Levchenko, and Jayavel Shanmugasundaram. WYSIWYG development of data driven web applications. *Proc. VLDB Endow.*, 1(1):163–175, 2008.