



NYU

**TANDON SCHOOL
OF ENGINEERING**

Relational Database & Pair Trading

4/12/2019

1



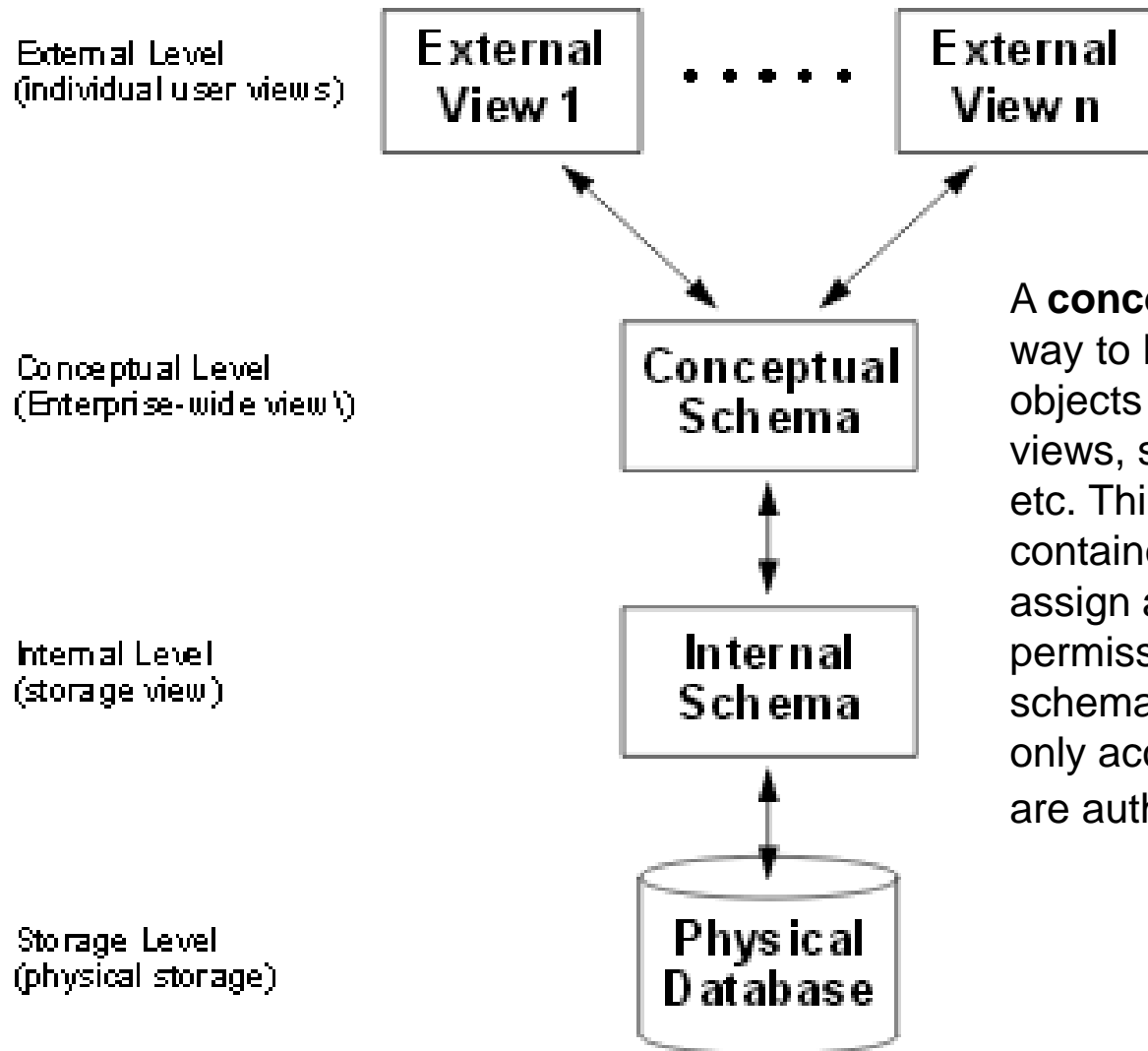
NEW YORK UNIVERSITY

Leading invention, innovation
and entrepreneurship



- **Database**
 - *A database is a persistent, logically coherent collection of inherently meaningful data, relevant to some aspects of the real world*
- **Database Management System (DBMS)**
 - *A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. According to the ANSI/SPARC DBMS Report, a DBMS should be envisioned as a multi-layered system.*
- **Relational Algebra**
 - *Special-purpose programming language*

Database Management System (DBMS)

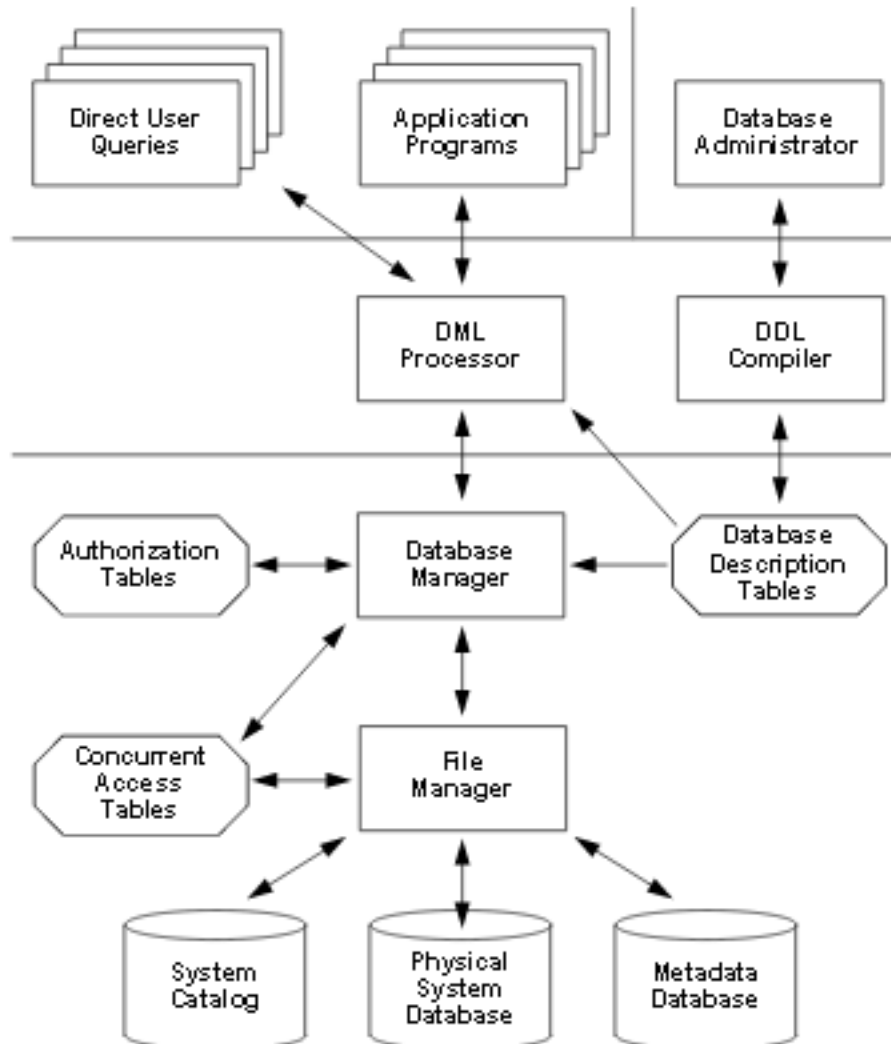


A **conceptual schema** is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects. You can assign a user login permissions to a single schema so that the user can only access the objects they are authorized to access.

What Does a DBMS Do?

- Database management systems provide several functions in addition to simple file management:
 - allow concurrency
 - control security
 - maintain data integrity
 - provide for backup and recovery
 - control redundancy
 - allow data independence
 - provide non-procedural query language
 - perform automatic query optimization

Components of a Database System



DDL (data definition language), such as Create, Drop, Alter, Rename

DML (data manipulation language), such as
SELECT ... FROM ... WHERE ...
INSERT INTO ... VALUES ...
UPDATE ... SET ... WHERE ...
DELETE FROM ... WHERE ...

Metadata is literally "data about data." This term refers to information about data itself -- perhaps the origin, size, formatting or other characteristics of a data item. In the database field, metadata is essential to understanding and interpreting the contents of a data warehouse.

Examples: The eXtensible Markup Language (XML) is a metadata format used to define data objects.

Relational Database Model

- What is a relational database?
 - a database that treats all of its data as a collection of relations
- What is a relation?
 - a kind of set
 - a subset of a Cartesian product
 - an unordered set of ordered tuple

Structured Information

- ***Field:*** An “atomic” unit of data.
 - *Number, string, true/false.*
- ***Record:*** A collection of related fields.
- ***Table:*** A collection of related records.
 - Each record is one row in the table.
 - Each field is one column in the table
- ***Primary Key:*** The field that identifies a record
 - Value of a primary key must be unique.
- ***Database:*** A collection of tables.

Relational Algebra

- Tables represent “relations”
 - Course, course description
 - Name, email address, department
- Named fields represent “attributes”
- Each row in the table is called a “tuple”
 - The order of the rows is not important
- Queries specify desired conditions
 - The DBMS then finds data that satisfies the queries.

Registrar Example

- Which students are in which courses?
- What do we need to know about the students?
 - First name, last name, email and department.
- What do we need to know about the courses?
 - Course ID, description, enrolled students, and grades.

A “Flat File” Solution: A bad approach

Student ID	Last Name	First Name	Department ID	Department	Course ID	Course description	Grades	email
1	Arrows	John	EE	EE	lpsc690	Information Technology	90	jarows@wam
1	Arrows	John	EE	Elec Engin	ee750	Communication	95	ja_2002@yahoo
2	Peters	Kathy	HIST	HIST	lpsc690	Information Technology	95	kpeters2@wam
2	Peters	Kathy	HIST	history	hist405	American History	80	kpeters2@wma
3	Smith	Chris	HIST	history	hist405	American History	90	smith2002@olue
4	Smith	John	CLIS	Info Sci	lpsc690	Information Technology	98	js03@wam

Goals of “Normalization”

- Save space
 - Save each fact only once
- More rapid updates
 - Every fact only needs to be updated once
- More rapid search
 - Finding something once is good enough
- Avoid inconsistency
 - Changing data once changes it everywhere

A Normalized Relational Database

Student Table

Student ID	Last Name	First Name	Department ID	email
1	Arrows	John	EE	jarrows@wam
2	Peters	Kathy	HIST	kpeters2@wam
3	Smith	Chris	HIST	smith2002@glue
4	Smith	John	CLIS	js03@wam

Department Table

Department ID	Department
EE	Electronic Engineering
HIST	History
CLIS	Information Stuides

Course Table

Course ID	Course Description
lbsc690	Information Technology
ee750	Communication
hist405	American History

Enrollment Table

Student ID	Course ID	Grades
1	lbsc690	90
1	ee750	95
2	lbsc690	95
2	hist405	80
3	hist405	90
4	lbsc690	98

Approaches to Normalization

- For simple problems
 - Start with “binary relationships”
 - Pairs of fields that are related
 - Group together wherever possible
 - Add keys where necessary
- For more complicated problems
 - Entity relationship modeling

Example of Join

Student Table

Student ID	Last Name	First Name	Department ID	email
1	Arrows	John	EE	jarrows@wam
2	Peters	Kathy	HIST	kpeters2@wam
3	Smith	Chris	HIST	smith2002@glue
4	Smith	John	CLIS	js03@wam

Department Table

Department ID	Department
EE	Electronic Engineering
HIST	History
CLIS	Information Stuides

“Joined” Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

- “Primary Key” uniquely identifies a record
 - Such as student ID in the student table.
- “Compound” primary key
 - Synthesize a primary key with a combination of fields.
 - Such as StudentID + CourseID in the enrollment table.
- “Foreign Key” is the primary key in the ***other*** table.
 - Note: it need not to be unique in this table.

Queries

New Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam



SELECT Student ID, Department

Student ID	Department
1	Electronic Engineering
2	History
3	History
4	Information Stuides

Filtration

New Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

WHERE Department ID = "HIST"

Student ID	Last Name	First Name	Department ID	Department	email
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue

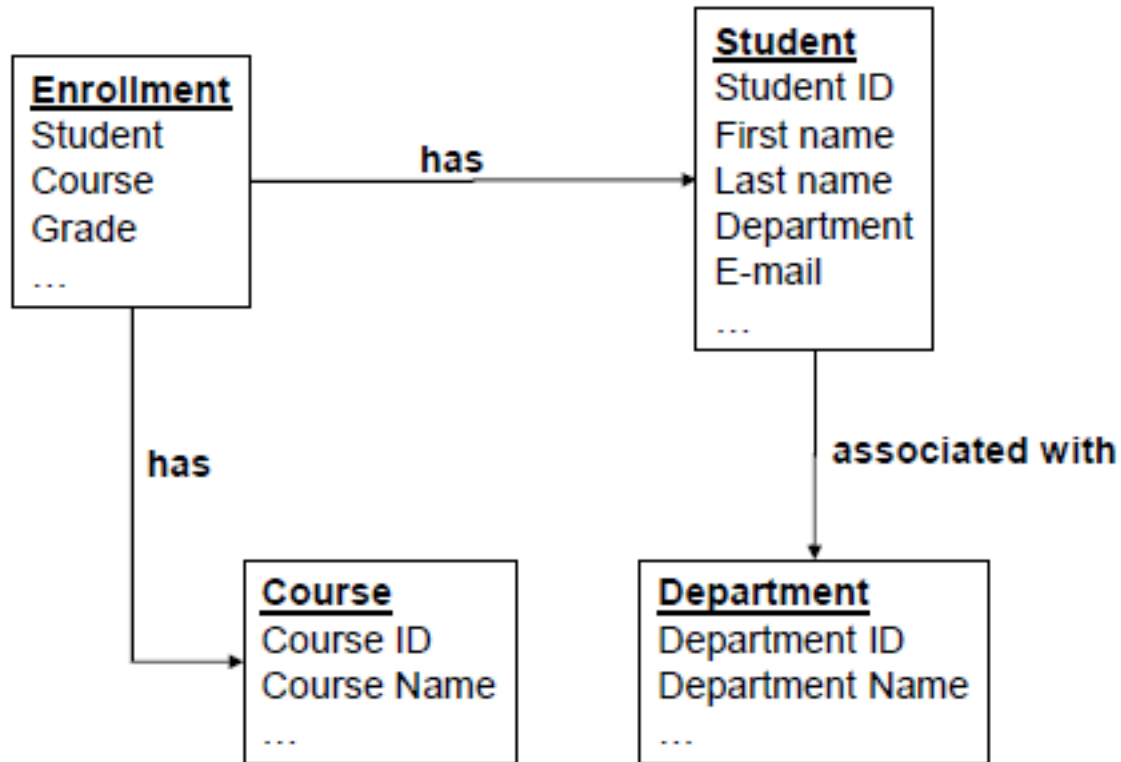
Entity-Relationship Diagram

- Graphical visualization of the data model
- Entities are captured in boxes.
- Relationships are captured using arrows.

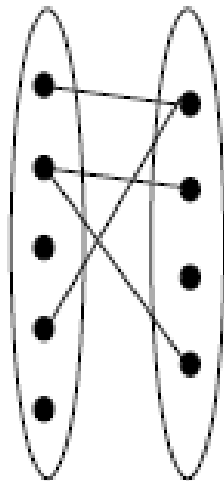
Getting started with E-R Modeling

- What questions must you answer?
- What data is needed to generate the answers?
 - Entities
 - Attributes of those entities
 - Relationships
 - Nature of those relationships
- How will the user interact with the system?
 - Relating the question to the available data
 - Expressing the answer in a useful form.

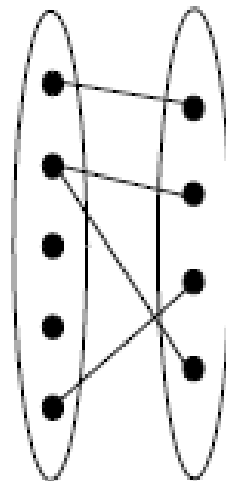
Registrar ER Diagram



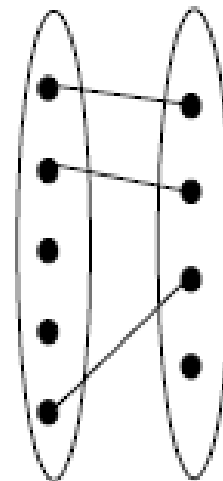
Types of Relationships



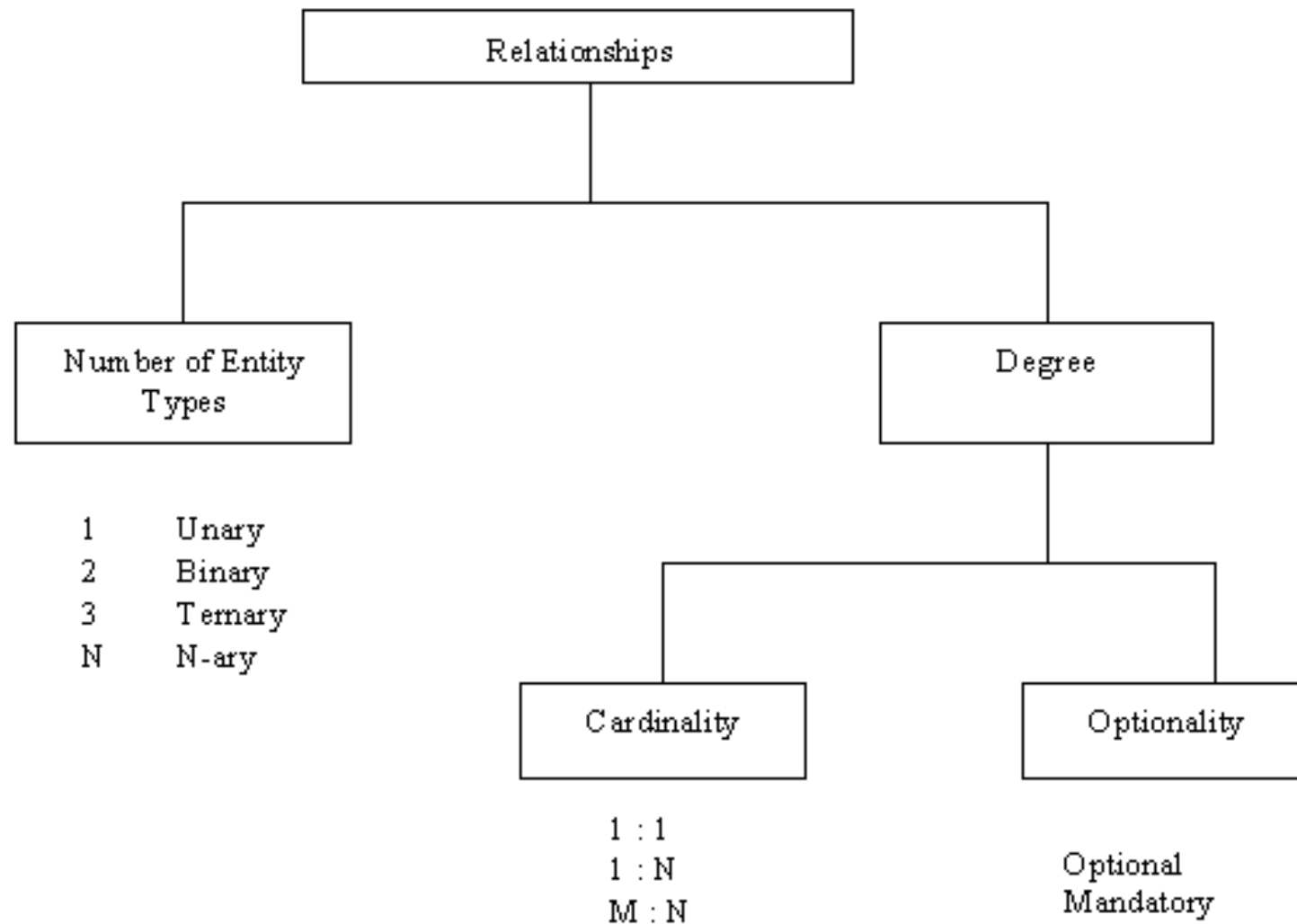
Many-to-Many



1-to-Many



1-to-1



Making Tables from E-R Diagrams

- Pick a primary key for each entity
- Build the tables
 - One per entity
 - Plus one per M:M relationship
 - Choose terse but memorable table and field names
- Check for parsimonious representation
 - Relational “normalization”
 - Redundant storage of computable values
- Implement using a DBMS

Associative EMPLOYEE_PROJECT table that resolves the M:M relationship

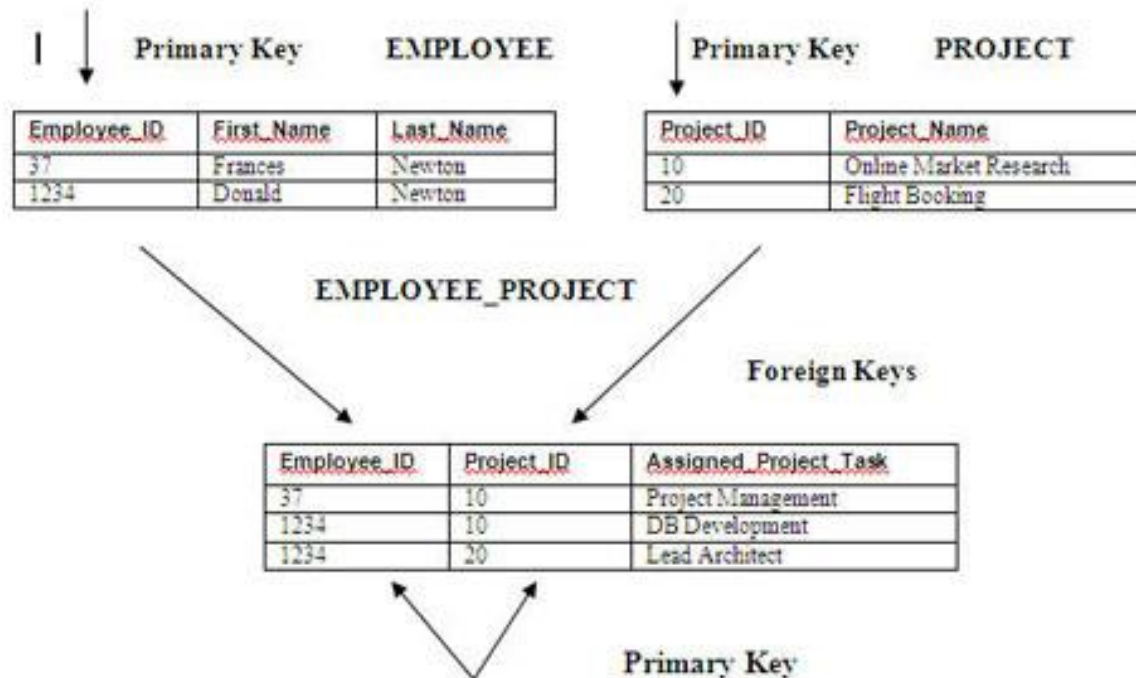


Figure 4: Associative EMPLOYEE_PROJECT table that resolves the M:M relationship

Structured Query Language (SQL)

- Consistent, unambiguous interface to any DBMS
- Simple command structure:
 - e.g., `SELECT Last name FROM Students WHERE Dept=CLIS`
- Useful standard for inter-process communications

The SQL SELECT Command

- Chooses columns
 - Based on their label
- Restrict chooses rows
 - Based on their contents, such as department ID = “HIST”
- These can be specified together
 - SELECT Student ID, Dept WHERE Dept = “History”
- Restrict Operators
 - Each SELECT contains a single WHERE
 - Numeric comparison
 - <, >, =, <>, e.g., grade<80
 - Boolean operations, e.g., Name = “John” AND Dept <> “HIST”

QueryPairPriceRatio

```
SELECT Pairs.Id, Pairs.Ticker1, Pairs.Ticker2,  
       Format(StDev(Prices1.Adj_Close/Prices.Adj_Close),"Fixed") AS  
       PriceRatio  
FROM Prices1, Prices, Pairs  
WHERE (((Prices1.Symbol)=[Pairs].[Ticker1]) AND  
       ((Prices.Symbol)=[Pairs].[Ticker2]) AND ((Prices1.Date)=[Prices].[Date]))  
GROUP BY Pairs.Id, Pairs.Ticker1, Pairs.Ticker2  
ORDER BY Pairs.Id, Pairs.Ticker1, Pairs.Ticker2;
```

Class Practice

FRE7831 Spring 2019 Part 2 Practice

Symbol	CUSIP	Company Name	TraderID	Last Name	First Name	Email	Phone	OrderID	Price	Quantity	Side	Order Type
GOOG	38259P508	Alphabet Inc.	1234	ABC	Joe	JA123@gs.com	212-111-1234	GSJA000111122015	717.01	200	Buy	Limit
GOOG	38259P508	Alphabet Inc.	1245	BCD	Andy	AB124@jp.com	212-222-1234	JPAB000111132015	0	125	Sell	Market
IBM	459200101	International Business Machines Corporation	2345	ABC	Smith	SA234@citi.com	212-333-1234	CTSA000111132015	131.75	1000	Buy	Limit
C	312072001	Citigroup Inc	1234	ABC	Joe	JA123@gs.com	212-111-1234	GSJA000211122015	53.17	10000	Buy	Limit
IBM	459200101	International Business Machines Corporation	1245	BCD	Andy	AB124@jp.com	212-222-1234	JPAB000211132015	131.7	2000	Sell	Limit
C	312072001	Citigroup Inc	2345	ABC	Smith	SA234@citi.com	212-333-1234	CTSA000211132015	0	200	Buy	Market

(1) Convert the above flat table into tables in a relational database with 4 tables: symbol table, trader table, order table and execution table.

(2) Draw an E-R diagram for your database. Show or Label proper primary key and foreign key (if have) for each table. Show all the fields of each table in your diagram

- Open the Excel sheet from our class Web site and complete 2 tasks.

Pair Trading Project

- Design, populate, and query an sqlite3 database for pairs trading.
- Provide SQL queries for pair trading statistics.
- Write an Object-Oriented program in C++ to run the back-test for a given portfolio over a given time period and store the results in the database.
- Design the algorithm to implement real time “grey box” trading using the database and objects you used for the back tester.

SQLite

- SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.



History of SQLite

- SQLite is an open source embedded database. The original implementation was designed by D. Richard Hipp.
- Hipp was designing software used on board guided missile systems and thus had limited resources to work with.
- The resulting design goals of SQLite were to allow the program to be operated without a database installation or administration.

Owens, Michael (2006). The Definitive Guide to SQLite. Apress. doi:10.1007/978-1-4302-0172-4_1. ISBN 978-1-59059-673-9. http://en.wikipedia.org/wiki/SQLite#cite_note-1

Notable Users

- Adobe Photoshop,
- Acrobat\Adobe Reader
- Apple Mail, Safari Web Brower
- Mozilla Firefox Web Browser
- Google Desktop
- Android OS
- McAfee Anti-Virus

<http://www.sqlite.org/famous.html>

Unique Features

- No configuration. Just drop in the C library and go.
- No server process to administer or user accounts to manage.
- Easy to backup and transmit database (just copy the file)
- Dynamic typing for column values, variable lengths for column records
- Query can reference multiple database files
- A few non-standard SQL extensions (mostly for conflict resolution)

<http://www.sqlite.org/different.html>

Disadvantages

- **High concurrency** – reader/writer locks on the entire file
- **Dataset Size Limitation** – DB file can't exceed file system limit or 2TB
- **Access control** – there isn't any

<http://www.sqlite.org/different.html>

Condition and Assumption

- “Going short” – the first stock of the pair is short and the other is long.
- “Going long” - the first stock of the pair is long and the other is short.
- Always trade 10,000 shares for the first stock (S_1) and determine the shares for the other (S_2) accordingly: $N_1P_1 + N_2P_2 = 0$
- N_1 and N_2 are the numbers of shares of S_1 and S_2 , and P_1 and P_2 are the prices of S_1 and S_2

Pair Trading Algorithm (1/2)

- Compute the standard deviation, σ_p , of the ratio of the two adjusted closing stock prices in each pair from 1/2/2008 to 12/31/2018. Store this standard deviation in the database.
- The variable, k , has default value 1, but could be changed by user in the pair trading program.
- Get *Close1d1, Close2d1, Open1d2, Open2d2, Close1d2, and Close2d2*, where *Close1d1* and *Close2d1* are the closing prices on day $d - 1$ for stocks 1 and 2, respectively, *Open1d2, Open2d2* are the opening prices for day d .

Pair Trading Algorithm (2/2)

— Open Trade:

- If $\text{abs}(\text{Close1d1}/\text{Close2d1} - \text{Open1d2}/\text{Open2d2}) > k\sigma$,
 - short the pair;
- Else if $\text{abs}(\text{Close1d1}/\text{Close2d1} - \text{Open1d2}/\text{Open2d2}) < k\sigma$,
 - go long the pair.
- $N1 = 10,000$ shares, traded at the price Open1d2 ,
- $N2 = N1 * (\text{Open1d2}/\text{Open2d2})$, traded at the price Open2d2 .

— Close Trade:

- The open trades will be closed at the closing prices and P/L for the pair trade will be calculated as:
$$(\pm N1 * [\text{Open1d2} - \text{Close1d2}]) +$$
$$(\pm N2 * [\text{Open2d2} - \text{Close2d2}])$$

Database Implementation Details (1/6)

- The database PairTradingDb has 5 main tables: Pairs, Tickers, Prices, TestPrices and Trades.
- PairPriceRatio Table, is created by using the SQL query, QueryPairPriceRatio, on Pairs and Prices tables:
 - ***SELECT Pairs.Id, Pairs.Ticker1, Pairs.Ticker2, Format(StDev(Prices1.Adj_Close/Prices.Adj_Close), "Fixed") AS PriceRatio***
 - ***FROM Prices1, Prices, Pairs***
 - ***WHERE Prices1.Symbol=Pairs.Ticker1 and Prices.Symbol=Pairs.Ticker2 and Prices1.Date=Prices.Date***
 - ***GROUP BY Pairs.Id, Pairs.Ticker1, Pairs.Ticker2***
 - ***ORDER BY Pairs.Id, Pairs.Ticker1, Pairs.Ticker2;***

Database Implementation Details (2/6)

PairPriceRatio				
	Id	Ticker1	Ticker2	PriceRatio
	1	AAPL	HPQ	7.14
	2	ABX	GG	0.09
	3	ABX	IAU	0.64
	4	ABX	NEM	0.07
	5	APC	CHK	0.70
	6	APC	DVN	0.21
	7	AXP	COF	0.13
	8	BAC	JPM	0.22
	9	BHP	FCX	0.39
	10	CAT	DE	0.14
	11	CENX	FCX	0.39
	12	CENX	NUE	0.31
	13	CHK	DVN	0.06
	14	COP	CVX	0.07
	15	CS	DB	0.11
	16	CSX	NSC	0.03
	17	CVX	XOM	0.14
	18	DAL	UAL	0.33
	19	DRYS	GNK	0.40
	20	DSX	GNK	0.47
	21	EGLE	GNK	0.40
	22	GG	ABX	0.08
	23	GG	GLD	0.05
	24	GG	KGC	0.88

Record: 1 of 31 No Filter Search

Database Implementation Details (3/6)

- PairTestPrices Table is created by using the SQL query, QueryPairTestPrices, on Pairs and TestPrices tables:
 - *SELECT Pairs.Ticker1 AS Ticker1, Pairs.Ticker2 AS Ticker2, TestPrices1.Open AS Ticker1Open, TestPrices1.Close AS Ticker1Close, TestPrices.Open AS Ticker2Open, TestPrices.Close AS Ticker2Close, TestPrices.Date AS PriceDate*
 - *FROM TestPrices1, TestPrices, Pairs*
 - *WHERE TestPrices1.Symbol=Pairs.Ticker1 and TestPrices.Symbol=Pairs.Ticker2 and TestPrices1.Date=TestPrices.Date*
 - *ORDER BY Pairs.Ticker1, Pairs.Ticker2, TestPrices1.Date;*

Database Implementation Details (4/6)

PairTestPrices						
Ticker1	Ticker2	Ticker1Open	Ticker1Close	Ticker2Open	Ticker2Close	PriceDate
AAPL	HPQ	615.91	606.81	18.31	17.66	8/1/2012
AAPL	HPQ	602.84	607.79	17.61	17.55	8/2/2012
AAPL	HPQ	613.63	615.7	17.83	18.26	8/3/2012
AAPL	HPQ	617.29	622.55	18.29	18.69	8/6/2012
AAPL	HPQ	622.77	620.91	18.56	18.96	8/7/2012
AAPL	HPQ	619.39	619.86	19.48	19.41	8/8/2012
AAPL	HPQ	617.85	620.73	19.4	19.41	8/9/2012
AAPL	HPQ	618.71	621.7	19.3	19.7	8/10/2012
AAPL	HPQ	623.39	630	19.69	19.62	8/13/2012
AAPL	HPQ	631.87	631.69	19.76	19.36	8/14/2012
AAPL	HPQ	631.3	630.83	19.29	19.29	8/15/2012
AAPL	HPQ	631.21	636.34	19.43	19.52	8/16/2012
AAPL	HPQ	640	648.11	19.52	19.52	8/17/2012
AAPL	HPQ	650.01	665.15	19.55	20.09	8/20/2012
AAPL	HPQ	670.82	656.06	20.22	19.93	8/21/2012
AAPL	HPQ	654.42	668.87	19.5	19.2	8/22/2012
AAPL	HPQ	666.11	662.63	18.05	17.64	8/23/2012
AAPL	HPQ	659.51	663.22	17.65	17.58	8/24/2012
AAPL	HPQ	679.99	675.68	17.65	17.21	8/27/2012
AAPL	HPQ	674.98	674.8	17.13	16.9	8/28/2012
AAPL	HPQ	675.25	673.47	16.86	16.94	8/29/2012
AAPL	HPQ	670.64	663.87	16.87	16.78	8/30/2012
AAPL	HPQ	667.25	665.24	16.88	16.88	8/31/2012
AAPL	HPQ	665.76	674.97	16.82	16.99	9/4/2012

41

Database Implementation Details (5/6)

Pairs				
	ID	Ticker1	Ticker2	Add New Field
+	1	AAPL	HPQ	
+	2	ABX	GG	
-	3	ABX	IAU	

	TradeID	TradeDate	Profit	Add New Field
	105	8/1/2012	0	
	106	8/2/2012	-1157.51	
	107	8/3/2012	-1798.55	
	108	8/6/2012	2466.46	
	109	8/7/2012	3414.71	
	110	8/8/2012	-4482	
	111	8/9/2012	-200.5	
	112	8/10/2012	3131.18	
	113	8/13/2012	-4167.11	
	114	8/14/2012	881.44	
	115	8/15/2012	5062.32	
	116	8/16/2012	8856.24	
	117	8/17/2012	-1810.76	
	118	8/20/2012	1461.18	
	119	8/21/2012	-3766.12	
	120	8/22/2012	3422.61	
	121	8/23/2012	-1239.41	
	122	8/24/2012	-1833.47	
	123	8/27/2012	-2135.94	
	124	8/28/2012	700	

Record: 7 of 52	No Filter	Search
-----------------	-----------	--------

Database Implementation Details (6/6)

- SQL queries for trade statistics.
 - QueryProfitableTrades:
 - *SELECT Pairs.ID, Pairs.Ticker1, Pairs.Ticker2,*
 - *sum(iif(Trades.Profit > 0, 1,0)) AS ProfitableTrades,*
 - *sum(iif(Trades.Profit < 0, 1,0)) AS LossTrades,*
 - *sum(iif(Trades.Profit <>0, 1, 0)) AS TotalTrades,*
 - *Format(ProfitableTrades/LossTrades, "Fixed") AS ProfitableRatio*
 - *FROM Pairs, Trades*
 - *WHERE Pairs.ID=Trades.PairID*
 - *GROUP BY Pairs.ID, Pairs.Ticker1, Pairs.Ticker2;*

Pair Trading Program Implementation (1/9)

- class Trade
- {
- private:
- int iPairID;
- char sTicker1[TICKER_LEN];
- char sTicker2[TICKER_LEN];
- float fTicker1Open;
- float fTicker1Close;
- float fTicker2Open;
- float fTicker2Close;
- char sDate[DATE_LEN];
- float fProfit;

Pair Trading Program Implementation (2/9)

- public:
- Trade(void);
- ~Trade(void);
- Trade(int id, char *pTicker1, char *pTicker2, float open1, float close1, float open2, float close2, char *pDate, float profit);
- int iGetPairID(void);
- void SetPairID(int id);
- char * sGetTicker1(void);
- void SetTicker1(char *pTicker1);
- char * sGetTicker2(void);

Pair Trading Program Implementation (3/9)

- `void SetTicker2(char *pTicker2);`
- `char * sGetDate(void);`
- `void SetDate(char *pDate);`
- `float fGetTicker1Open(void);`
- `float fGetTicker1Close(void);`
- `float fGetTicker2Open(void);`
- `float fGetTicker2Close(void);`
- `float fGetProfit(void);`
- `void SetProfit(float profit);`
- `friend ostream & operator<<(ostream &out, const Trade & aTrade);`
- `};`

Pair Trading Program Implementation (4/9)

- STL map<pair<string, string>, PairPriceStd>
PairPriceStdTable
 - Store information retrieved from PairPriceRatio table. The keys of the map are the pairs of tickers. The elements of the map are instances of class PairPriceStd.
- STL map< pair<string, string>, vector<Trade> > TradeTable
 - Store retrieved price information from PairTestPrices table as well as P/L for each trade for the back test. The keys of the map are the pairs of tickers. The elements of the map are STL vectors of trade instances. Each vector holds all the trades for the pair of tickers from the back test.

Pair Trading Program Implementation (5/9)

- The program PairTrading.cpp has 5 functions:
 - RetrievePairPriceStdTable(...)
 - RetrieveTradeTable(...)
 - UpdateTradeTable(...)
 - PairTrade(...)
 - EnterPairTrade(...)

Pair Trading Program Implementation (6/9)

- The function EnterPairTrade() allows an investor to enter a pair trade. For example:
 - *Enter a pair trade:*
 - *ticker1: AAPL*
 - *ticker2: HPQ*
 - *Ticker 1 Previous Day Close Price: 606.81*
 - *Ticker 2 Previous Day Close Price: 17.66*
 - *Ticker 1 Open Price: 602.84*
 - *Ticker 2 Open Price: 17.61*
 - *Ticker 1 Close Price: 607.79*
 - *Ticker 2 Close Price: 17.55*
 - *AAPL HPQ*
 - *delta = 7.14 k = 1*
 - *vol1 = 10000 vol2 = 342328 P/L = 70039.7*

Pair Trading Program Implementation (7/9)

- The PairTrading program offers users a menu of options:
 - 1 - *Select k (default k = 1).*
 - 2 - *Run Back Test.*
 - 3 - *Enter a Pair Trade.*
 - 4 - *Exit.*
- The results from option 1 and 2 are saved in outputs.txt file

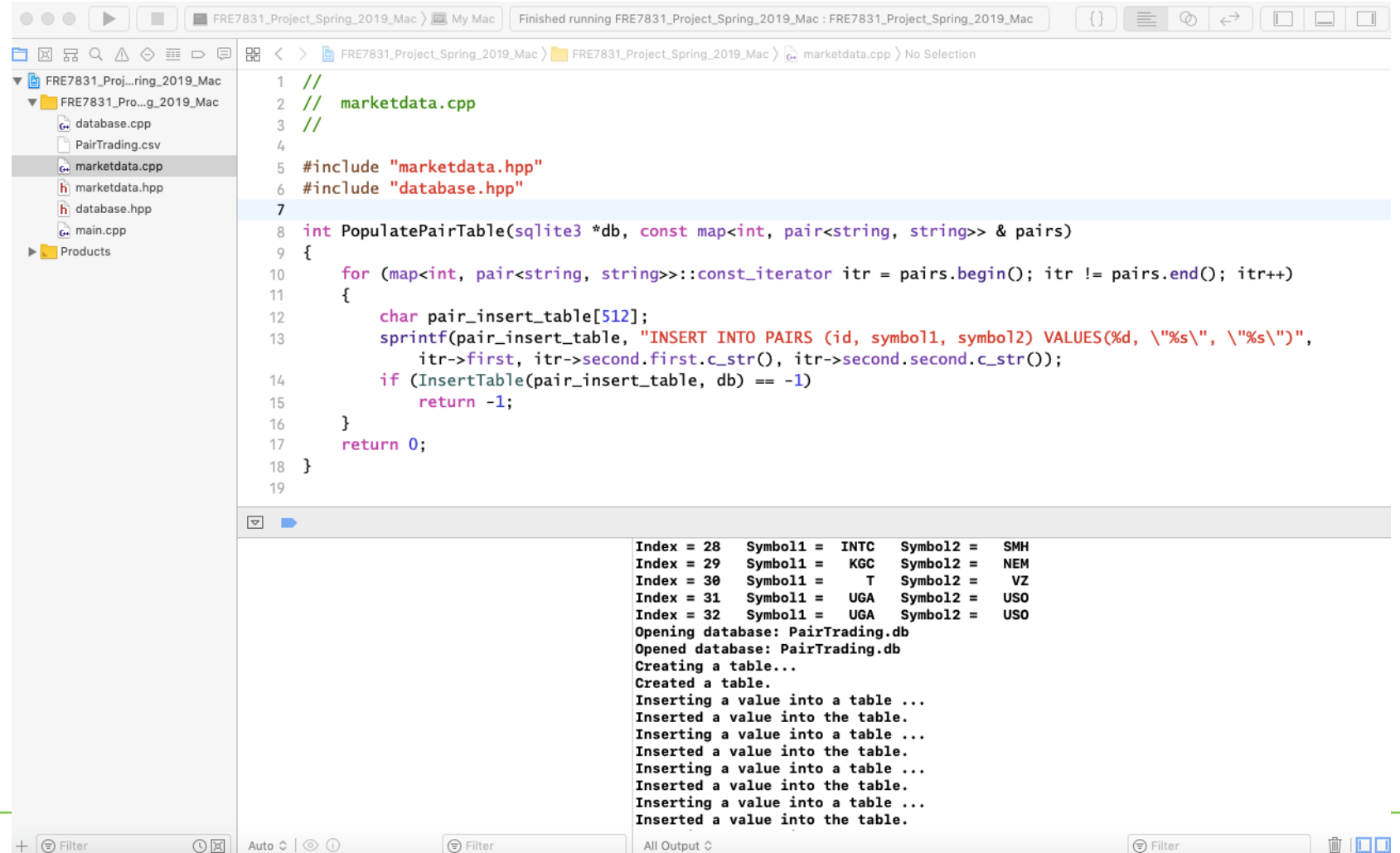
Pair Trading Program Implementation (8/9)

The screenshot shows the DB Browser for SQLite window with the following table structure:

Name	Type	Schema
Tables (1)		
PAIRS	CREATE TABLE PAIRS (id INT NOT NULL, symbol1 CHAR (20) NOT NULL, symbol2 CHAR (20) NOT NULL)	
id	INT	'id' INT NOT NULL
symbol1	CHAR (20)	'symbol1' CHAR (20) NOT NULL
symbol2	CHAR (20)	'symbol2' CHAR (20) NOT NULL
Indices (0)		
Views (0)		
Triggers (0)		

The 'Edit Database Cell' dialog is open, showing the 'symbol1' column selected. The 'Type of data currently in cell: NULL' and '0 byte(s)' are displayed. The 'Remote' tab is also visible.

Pair Trading Program Implementation (9/9)



```
1 //
2 // marketdata.cpp
3 //
4
5 #include "marketdata.hpp"
6 #include "database.hpp"
7
8 int PopulatePairTable(sqlite3 *db, const map<int, pair<string, string>> & pairs)
9 {
10     for (map<int, pair<string, string>>::const_iterator itr = pairs.begin(); itr != pairs.end(); itr++)
11     {
12         char pair_insert_table[512];
13         sprintf(pair_insert_table, "INSERT INTO PAIRS (id, symbol1, symbol2) VALUES(%d, \"%s\", \"%s\")",
14             itr->first, itr->second.first.c_str(), itr->second.second.c_str());
15         if (InsertTable(pair_insert_table, db) == -1)
16             return -1;
17     }
18     return 0;
19 }
```

Index = 28 Symbol1 = INTC Symbol2 = SMH
Index = 29 Symbol1 = KGC Symbol2 = NEM
Index = 30 Symbol1 = T Symbol2 = VZ
Index = 31 Symbol1 = UGA Symbol2 = USO
Index = 32 Symbol1 = UGA Symbol2 = USO
Opening database: PairTrading.db
Opened database: PairTrading.db
Creating a table...
Created a table.
Inserting a value into a table ...
Inserted a value into the table.
Inserting a value into a table ...
Inserted a value into the table.
Inserting a value into a table ...
Inserted a value into the table.
Inserting a value into a table ...
Inserted a value into the table.
Inserting a value into a table ...
Inserted a value into the table.

Algorithmic Trading

- Buy 50,000 shares of SPY using VWAP on 5/21/2012
- For VWAP:
 - Compute the weighted average volume distribution from 5/1/2012-5/20/2012
 - Use it as a guide for trade sizing
 - Different trading desks start at different times.
- Report execution statistics:
 - Duration of execution
 - Realized size-weighted average price
 - Price distribution
 - Order size distribution

Algorithm Development

- When we used MAY 2012 ticker data for calculating weighted average volume distribution,
 - Used only the traded quantities and their corresponding prices. We did not consider any quote.
 - For intraday day, we combined trading quantities into each 15min interval according to the timestamp of each trade.
 - For the quantity of each 15 min interval, we summed up corresponding quantity for each trading day from 5/1/12 to 5/20/12, and then calculated the average for each time interval by dividing the sum by the number of trading dates, shown in the following table.
 - For the execution prices, we used the trading prices on 5/21/12. We assumed the price of 1st trade in each time interval as the execution price for the entire interval, shown in the following table.

Time Interval	Quantity (Avg Qty from 5/1-5/20)	Price (Trading Prices from 5/21)
9:30:00	6459	130.6
9:45:00	16145	130.35
10:00:00	7968	130.42
10:15:00	7135	130.4
10:30:00	35281	130.42
10:45:00	34101	130.57
11:00:00	144182	130.62
11:15:00	156987	130.59
11:30:00	191750	130.52
11:45:00	167884	130.53
12:00:00	526524	130.39
12:15:00	625083	130.2
12:30:00	1045698	130.33
12:45:00	581429	130.18
13:00:00	1007854	130.17
13:15:00	705572	130.25
13:30:00	9983530	130.16
13:45:00	8314013	130.48
14:00:00	9923065	130.35
14:15:00	7437595	130.5
14:30:00	6642507	130.93
14:45:00	5890572	131.12
15:00:00	6362905	131.35
15:15:00	5351374	131.2
15:30:00	5760619	131.14
15:45:00	6794683	131.08

Data

- Data for May 2012 is a csv file, over 2 GB in size
- Data contain best quotes and trades only
- The data file could not be opened by Excel or text editor software products due to its size.
- Have to use C/C++ to write a program to retrieve the information directly from the file

Name	#RIC	Date[G]	Time[G]	GMT Offset	Type	Price	Volume	Market	VWAP	Bid Price	Bid Size	Ask Price	Ask Size	Qualifiers
	SPY	01-MAY-2012	00:00:00.097	-4	Quote	140.07	1	140.08	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	00:00:00.177	-4	Quote	140.05	1	140.08	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	00:00:00.451	-4	Quote	140.07	1	140.08	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	00:00:00.518	-4	Quote	0	0	0,NQ	[PRC_QL3];NQ	[PRC_QL_CD]				
	SPY	01-MAY-2012	08:00:00.190	-4	Quote	139.75	1	140.19	1,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:01:15.666	-4	Quote	139.53	1	140.25	2,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:01:16.503	-4	Quote	139.53	1	140.19	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:02:42.364	-4	Quote	139.54	160	140.19	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:02:42.364	-4	Quote	139.54	160	140.18	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:02:56.684	-4	Quote	139.54	160	140.18	163,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:02:56.684	-4	Quote	139.54	160	140.18	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:02:56.684	-4	Quote	139.54	160	140.17	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:16.772	-4	Quote	139.55	50	140.17	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:16.772	-4	Quote	139.56	160	140.17	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:34.303	-4	Quote	139.55	50	140.18	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:34.690	-4	Quote	139.56	160	140.18	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:34.690	-4	Quote	139.56	160	140.17	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:43.959	-4	Quote	139.56	160	140.17	163,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:43.959	-4	Quote	139.56	160	140.17	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:43.959	-4	Quote	139.56	160	140.16	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:48.149	-4	Quote	139.57	7	140.16	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:48.149	-4	Quote	139.58	160	140.16	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:51.095	-4	Quote	139.58	160	140.16	163,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:51.095	-4	Quote	139.58	160	140.16	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:51.095	-4	Quote	139.58	160	140.15	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:55.828	-4	Quote	139.57	7	140.16	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:55.828	-4	Quote	139.58	160	140.16	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:55.828	-4	Quote	139.58	160	140.15	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:57.978	-4	Quote	139.58	160	140.15	163,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:57.978	-4	Quote	139.58	160	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:03:58.010	-4	Quote	139.58	160	140.14	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:07.291	-4	Quote	139.57	7	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:07.411	-4	Quote	139.58	160	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:07.411	-4	Quote	139.58	160	140.14	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:15.058	-4	Quote	139.57	7	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:18.796	-4	Quote	139.58	160	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:18.796	-4	Quote	139.58	160	140.14	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:21.987	-4	Quote	139.57	7	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:22.082	-4	Quote	139.58	160	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:22.082	-4	Quote	139.58	160	140.14	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:23.672	-4	Quote	139.57	7	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:24.930	-4	Quote	139.58	160	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:24.930	-4	Quote	139.58	160	140.14	160,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:27.014	-4	Quote	139.57	7	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			
	SPY	01-MAY-2012	08:04:27.152	-4	Quote	139.58	160	140.15	3,R	[PRC_QL3];R	[PRC_QL_CD]			

Date modified: 11/9/2012 7:16 PM

Date created: 11/7/2014 10:35 AM

na Separated Val...

Size: 2.76 GB



POLYTECHNIC INSTITUTE OF NEW YORK UNIVERSITY

Name	SPY,01-MAY-2012,09:18:36.193,-4,Quote,,139.87,255,139.94,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:36.193,-4,Quote,,139.87,255,139.94,410,R [PRC_QL3];R [PRC_QL_CD]
SPY_May_2012	SPY,01-MAY-2012,09:18:36.193,-4,Quote,,139.87,5,139.94,410,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:48.694,-4,Quote,,139.87,5,139.94,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:48.694,-4,Quote,,139.87,5,139.93,250,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:48.694,-4,Quote,,139.87,5,139.92,250,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:50.104,-4,Quote,,139.87,5,139.94,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:50.210,-4,Quote,,139.87,5,139.94,410,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:50.447,-4,Quote,,139.87,5,139.94,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:18:50.618,-4,Quote,,139.87,5,139.94,410,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.172,-4,Quote,,139.87,5,139.94,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.172,-4,Quote,,139.87,5,139.93,250,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.172,-4,Quote,,139.87,5,139.92,250,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.771,-4,Quote,,139.87,5,139.91,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.786,-4,Quote,,139.87,5,139.91,410,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.786,-4,Quote,,139.87,5,139.9,250,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:18.786,-4,Quote,,139.87,5,139.91,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:28.957,-4,Quote,,139.87,5,139.91,410,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:32.093,-4,Quote,,139.87,5,139.9,250,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:33.995,-4,Quote,,139.87,5,139.91,410,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:36.950,-4,Quote,,139.87,5,139.91,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:19:36.950,-4,Quote,,139.87,5,139.89,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:21:55.071,-4,Quote,,139.87,10,139.89,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:21:57.798,-4,Quote,,139.87,5,139.89,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:22:03.707,-4,Trade,139.87,500,,,,," T [GV4_TEXT];131[IRGCOND];NBL [PRC_QL2]"
	SPY,01-MAY-2012,09:22:15.136,-4,Trade,139.87,100,,,,," T [GV4_TEXT];131[IRGCOND];NBL [PRC_QL2]"
	SPY,01-MAY-2012,09:22:43.593,-4,Trade,139.87,500,,,,," T [GV4_TEXT];131[IRGCOND];NBL [PRC_QL2]"
	SPY,01-MAY-2012,09:22:47.037,-4,Trade,139.87,3900,,,,," T [GV4_TEXT];131[IRGCOND];NBL [PRC_QL2]"
	SPY,01-MAY-2012,09:22:47.037,-4,Quote,,139.8,160,139.89,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:22:47.037,-4,Quote,,139.8,160,139.86,121,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:22:47.037,-4,Quote,,139.8,410,139.86,121,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:22:47.116,-4,Quote,,139.8,410,139.86,160,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:23:50.527,-4,Quote,,139.8,440,139.86,190,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:52.477,-4,Quote,,139.8,440,139.86,30,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:52.477,-4,Quote,,139.8,440,139.87,30,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:52.477,-4,Quote,,139.8,440,139.87,280,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:52.477,-4,Quote,,139.8,440,139.87,260,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:52.519,-4,Quote,,139.8,440,139.87,280,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:54.061,-4,Quote,,139.83,160,139.87,280,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:54.061,-4,Quote,,139.83,160,139.87,30,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:26:54.235,-4,Quote,,139.83,190,139.87,10,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:29:53.631,-4,Quote,,139.83,30,139.87,10,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:29:53.631,-4,Quote,,139.8,440,139.87,10,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:29:53.631,-4,Quote,,139.8,440,139.87,260,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:29:53.631,-4,Quote,,139.8,440,139.87,10,R [PRC_QL3];R [PRC_QL_CD]
	SPY,01-MAY-2012,09:29:53.649,-4,Quote,,139.8,190,139.87,10,R [PRC_QL3];R [PRC_QL_CD]

Date modified: 11/9/2012 7:16 PM

Date created: 11/7/2014 10:35 AM

na Separated Val...

Size: 2.76 GB

Homework Assignment 3

- Using SPY_May_2012.csv for the following tasks:
 - Used only the traded quantities and their corresponding prices. We did not consider any quote.
 - For intraday day, we combined trading quantities into each 15min interval according to the timestamp of each trade.
 - For the quantity of each 15 min interval, we summed up corresponding quantity for each trading day from 5/1/12 to 5/20/12, and then calculated the average for each time interval by dividing the sum by the number of trading dates, shown in the following table.
 - For the execution prices, we used the trading prices on 5/21/12. We assumed the price of 1st trade in each time interval as the execution price for the entire interval, shown in the following table.
 - Save your results in outputs.csv file.

References

- *Database systems: The Complete Book*. Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom. Prentice Hall, 2008, ISBN 0-131-87325-3
- *Database Fundamentals*, Robert Robbins, John Hopkins University, 1995
- *Relational Database*, INFM 603 Week 9, College of Information Studies, University of Maryland, 2014
- *NYU Polytechnic School, FRE6883, Financial Computing, Lecture Notes*, Fall 2014
- *Pairs Trading, Quantitative Methods and Analysis*, Ganapathy Vidyamurthy, Wiley, 2004, ISBN 0-471-46067-2
- *Pairs Trading: A Bayesian Example*, Stefan Hools & J.Richard Hollos, Abrazol Publishing, 2012
- *High-Frequency Trading: A Practical Guide to Algorithmic Strategies & Trading Systems*, 2nd Ed, 2013, Irene Aldridge, ISBN: 1-118-34350-6