Low Level Design

CS-370: Software Design and Development

Robert Bruce

High level versus low level design

- Low-level design includes detailed information on project implementation.
- Low-level design is more detailed than high-level design.
- Low-level design prepares programmers for project codification (i.e. writing the software).
- High-level design addresses the “what”.
- Low-level design addresses the “how”.

Source: pages 155, 180, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Implementing a Relational Database Management System

- One example of low-level design is preparing a database management system (DBMS) for implementation in a project.

What is a Database Management Systems (DBMS)?

- A Database Management System (DBMS) is software to store, retrieve, update, and delete data stored on disk.
- A DBMS utilizes various techniques for storage efficiency, retrieval speed, and possibly data integrity.
- A DBMS is comprised of one or more uniquely-named databases.
- Each database in a DBMS contains one or more uniquely named tables.
- Each table in a database contains one or more uniquely named columns (or fields) of data.
- A row of data represents one or more columns of data from a database query.

What is a Relational Database Management Systems (RDBMS)?

- A Relational Database Management System (RDBMS) is a common implementation of a DBMS which can manage data by building relations between columns of data.
- Structured Query Language (SQL) is the universal standard language for communicating with an RDBMS.
- The American National Standards Institute (ANSI) defined the SQL standard.
- With few exceptions, SQL is consistent and portable regardless of the RDBMS you utilize.

Structured Query Language (SQL): SELECT statement

- INSERT is an SQL statement for adding new data into a table in a database.
- **`INSERT INTO students (student_id, first_name, last_name) VALUES (912941821, 'Quimby', 'Blastonovich');`**

Structured Query Language (SQL): SELECT statement

- SELECT is an SQL statement for retrieving data from one or more tables in a database (depending on the complexity of the query)
- `SELECT (first_name, last_name) FROM students WHERE student_id = 912941821;`

Structured Query Language (SQL): SELECT statement

- UPDATE is an SQL statement for revising one or more columns of data in a table in a database.
- **`UPDATE courses SET enrollment_capacity = 28 WHERE (course_name = 'CS-370') AND (course_section = 2);`**

Structured Query Language (SQL): SELECT statement

- DELETE is an SQL statement for removing one or more rows of data in a table in a database.
- `DELETE FROM waitlist WHERE last_date_to_add < NOW();`

MySQL datatypes (abridged)

- Numeric data types:
  - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT
  - DECIMAL, NUMERIC
- String data types:
  - VAR, VARCHAR
- Date data types:
  - DATE, DATETIME, TIMESTAMP

Relational Table Design: data normalization

- Data normalization is a structural process to remove data duplication.

Source: pages 170, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Relational Table Design: First Normal Form (1NF)

- “Each column must have a unique name.”
- “The order of the rows and columns doesn’t matter.”
- “Each column must have a single data type.”
- “No two rows can contain identical values.”
- “Each column must contain a single value.”
- “Columns cannot contain repeating groups.”

Source: pages 170-171, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Weapons training sign-up sheet (un-normalized)

NAME	WEAPON	WEAPON
Shelly Silva	Broadsword	
Louis Christenson	Bow	
Lee Hall	Katana	
Sharon Simmons	Broadsword	Bow
Felipe Vega	Broadsword	Katana
Louis Christenson	Bow	
Kate Ballard	Everything	

Source: pages 171, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Weapons training sign-up sheet (1NF)

Tutorials	
TIME	NAME
9:00	Shelly Silva
9:30	Louis Christenson
10:00	Lee Hall
10:30	Sharon Simmons
11:00	Felipe Vega
11:30	Louis Christenson
12:00	Kate Ballard

Tutorial Weapons	
TIME	WEAPON
9:00	Broadsword
9:30	Bow
10:00	Katana
10:30	Broadsword
10:30	Bow
11:00	Broadsword
11:00	Katana
11:30	Bow
12:00	Broadsword
12:00	Bow
12:00	Katana

Source: pages 173, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Relational Table Design: Second Normal Form (2NF)

- “It is in 1NF.”
- “All non-key fields depend on all key fields.”

Source: page 174, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Camp games schedule (1NF)

TIME	GAME	DURATION	MAXIMUM PLAYERS
1:00	Goblin Launch	60 mins	8
1:00	Water Wizzards	120 mins	6
2:00	Panic at the Picnic	90 mins	12
2:00	Goblin Launch	60 mins	8
3:00	Capture the Castle	120 mins	100
3:00	Water Wizzards	120 mins	6
4:00	Middle Earth Hold’em Poker	90 mins	10
5:00	Capture the Castle	120 mins	100

Source: page 175, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Camp games schedule (2NF)

Scheduled Games	
TIME	GAME
1:00	Goblin Launch
1:00	Water Wizzards
2:00	Panic at the Picnic
2:00	Goblin Launch
3:00	Capture the Castle
3:00	Water Wizzards
4:00	Middle Earth Hold’em Poker
5:00	Capture the Castle

Games		
GAME	DURATION	MAXIMUMPLAYERS
Goblin Launch	60 min	8
Water Wizzards	120 min	6
Panic at the Picnic	90 min	12
Capture the Castle	120 min	100
Middle Earth Hold’em Poker	90 min	10

Source: page 177, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Relational Table Design: Third Normal Form (3NF)

- “It is in 2NF.”
- “It contains no transitive dependencies.”

Source: page 176, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Counselors' favorite books (1NF)

COUNSELOR	FAVORITEBOOK	AUTHOR	PAGES
Becky	Dealing with Dragons	Patricia Wrede	240
Charlotte	The Last Dragonslayer	Jasper Fforde	306
J.C.	Gil's All Fright Diner	A. Lee Martinez	288
Jon	The Last Dragonslayer	Jasper Fforde	306
Luke	The Color of Magic	Terry Pratchett	288
Noah	Dealing with Dragons	Patricia Wrede	240
Rod	Equal Rites	Terry Prachett	272
Wendy	The Lord of the Rings Trilogy	J. R. R. Tolkien	1178

Source: page 177, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Counselors' favorite books (3NF)

Counselor Favorites	
COUNSELOR	FAVORITEBOOK
Becky	Dealing with Dragons
Charlotte	The Last Dragonslayer
J. C.	Gil's All Fright Diner
Jon	The Last Dragonslayer
Luke	The Color of Magic
Noah	Dealing with Dragons
Rod	Equal Rites
Wendy	The Lord of the Rings Trilogy

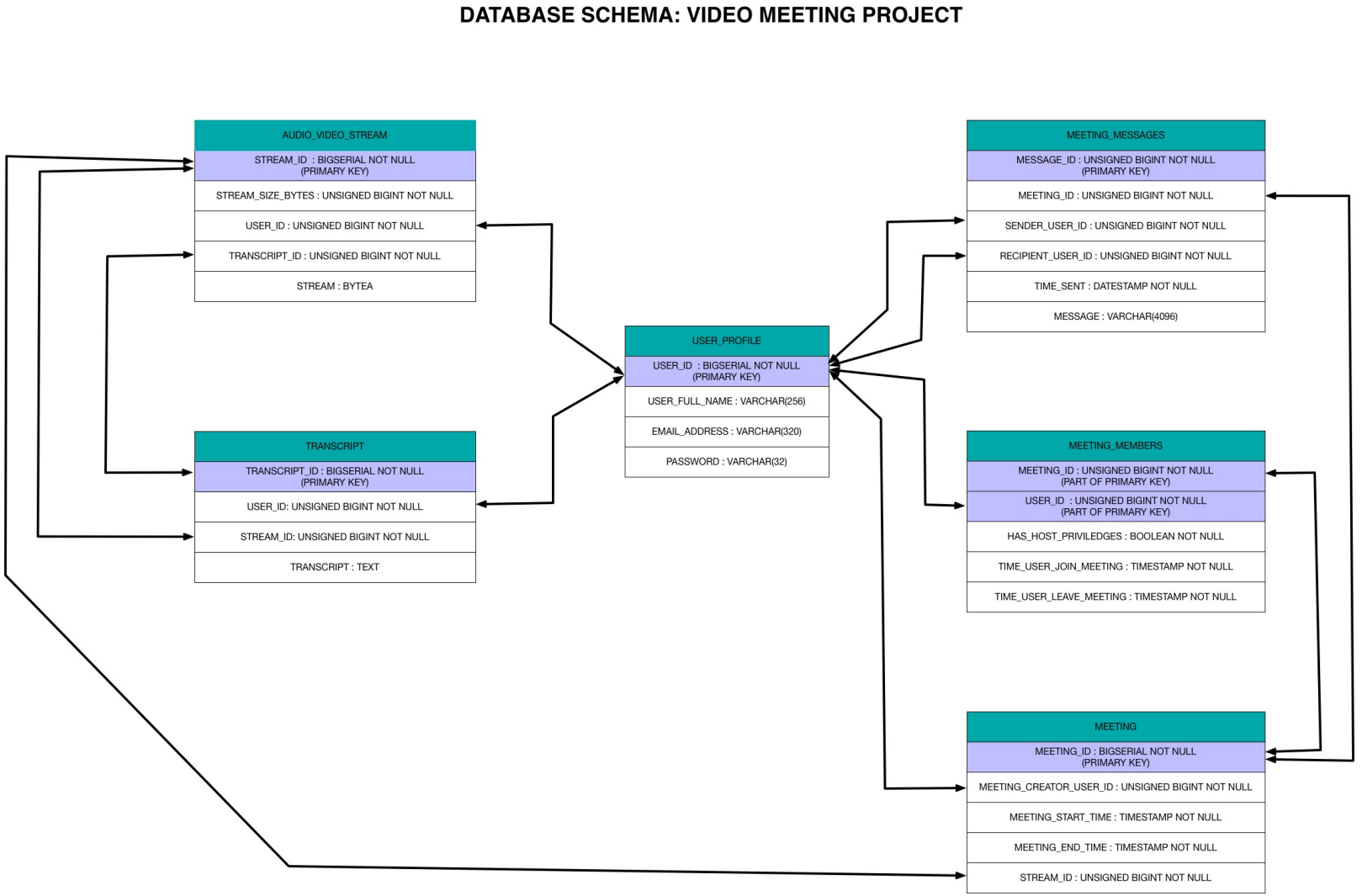
BookInfo		
BOOK	AUTHOR	PAGES
Dealing with Dragons	Patricia Wrede	240
The Last Dragonslayer	Jasper Fforde	306
Gil's All Fright Diner	A. Lee Martinez	288
The Color of Magic	Terry Pratchett	288
Equal Rites	Terry Pratchett	272
The Lord of the Rings Trilogy	J. R. R. Tolkien	1178

Source: page 178, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Defining the database Schema

- The database schema is a diagram which defines the structure of the entire database.
- Each table in the database should include the names of each column (field) in that table as well as the datatype for each column (field).
- The database schema defines the relationship between tables in the database (i.e. how the tables are tied together).
- I typically draw arrow diagrams pointing to columns (fields) within each table to indicate how two tables are tied together.

Example Schema



Creating MySQL databases

- Before creating tables, one must create a database (a container) to store the tables:
- `CREATE DATABASE auction;`
- If you accidentally create the wrong database, you can delete it. **This is a very dangerous command if you have data inside:**
- `DROP DATABASE auction;`

Choosing a database to utilize in MySQL: USE

- After creating the database, enter the USE command to choose a database:
- `USE auction;`

Defining the structure of a MySQL table: CREATE

- After creating the database, enter the CREATE command to define the structure of a table in that database:
- `CREATE auction (auction_id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY, item_id BIGINT NOT NULL, auction_expiration_date DATETIME NOT NULL);`
- An alternative way to create the table structure (above) is to issue three separate commands:
- `CREATE TABLE auction (auction_id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY);`
- `ALTER TABLE auction ADD item_id BIGINT NOT NULL;`
- `ALTER TABLE auction ADD auction_expiration_date DATETIME NOT NULL;`
- If you need to delete a whole table (**a very dangerous move if there is data inside the table**) you can do so with the following command:
- `DROP TABLE auction;`

Using MySQL’s proprietary function: AUTO_INCREMENT

- AUTO_INCREMENT (with an underscore) is a proprietary function within MySQL to automatically increment a non-negative integer when inserting a new row of data within a MySQL table.
- The AUTO_INCREMENT function is atomic. This means the function is impervious to race-conditions (i.e. when two rows are inserted at the table at the exact same time, the automatic increment feature will still work singularly to ensure each row of data has a UNIQUE non-negative integer.
- AUTO_INCREMENT should be used when a column is defined as a PRIMARY KEY and must contain a unique value in that column.
- *Rob’s note: AUTO_INCREMENT is a useful feature but is not defined in the ANSI SQL standard. Consequently, this MySQL feature will need to be changed when porting MySQL to another RDBMS such as Postgres (use BIGSERIAL in Postgres – same functionality but also proprietary) or Oracle (use triggers in Oracle).*

Next step: Accessing MySQL through an API

- After creating a database in MySQL and at least one table in that database, the next step is connecting to MySQL through your programs.

Accessing MySQL through an API

- Typically, programmers access database management systems through a library or module. The library or module provides a standard (consistent) interface for issuing database commands.
- MySQL provides an Application Programming Interface (API) for compiled languages C and C++.
  - *Note: writing your applications with C or C++ means you will need to include a MySQL includes header file and link to a MySQL library (shared object or dynamic linked library) into your compiled application. This is something you would include in a Makefile.*
  - *I will provide example programs in C (with Makefile), C++ (with Makefile), Python, Perl, and PHP. These will be uploaded to the files folder under Canvas by next week.*
- Scripting languages such as Python, Perl, or PHP also provide a standard database interface to connect to MySQL.

Example: Connecting to MySQL through Python

```
#!/usr/bin/python3

import mysql.connector

cnx = mysql.connector.connect(user='username',
password='password',host='127.0.0.1',database='students')

cnx.close()
```

Example: Connecting to MySQL through Perl

```
#!/usr/bin/perl

use strict;
use warnings;
use DBI;

# Connect to the database.
my $dbh = DBI→connect("DBI:mysql:database=students;host=localhost",
"username", "password", 'RaiseError' => 1});

# Disconnect from the database.
$dbh->disconnect();
```

Example: Connecting to MySQL through PHP

```
<?php

// Connecting, selecting database
$link = mysql_connect('hostname', 'username', 'password')
    or die('Could not connect: ' . mysql_error());

// Closing connection
mysql_close($link);

?>
```

Example: Connecting to MySQL through C (abridged)

```
/* connect to server with the CLIENT_MULTI_STATEMENTS option */
if (mysql_real_connect (mysql, host_name, user_name, password,
    db_name, port_num, socket_name, CLIENT_MULTI_STATEMENTS) == NULL)
{
  printf("mysql_real_connect() failed\n");
  mysql_close(mysql);
  exit(1);
}

mysql_close(mysql);
```

Example: Connecting to MySQL through C++ (abridged)

```
MYSQL *conn;
MYSQL_RES *res;
MYSQL_ROW row;

conn = mysql_init(NULL);
if (conn == NULL)
{
  std::cerr << "mysql_init() failed\n";
  return EXIT_FAILURE;
}
if (mysql_real_connect(conn, "server_host", "username", "password", "database", 0, NULL, 0) == NULL)
{
  std::cerr << "mysql_real_connect() failed\n";
  mysql_close(conn);
  return EXIT_FAILURE;
}
mysql_close(conn);
```