



**Sudan University of Science and
Technology**

**College of computer science and
information technology**

**Auto Database Schema
Generator**

(ADBG)

**التوليد التلقائي لهيكليه قاعده
البيانات**

October 2016

**THIS IS SUBMITTED AS A PARTIAL REQUIREMENT
OF B.SC. (HONOR) DEGREE IN COMPUTER AND
INFORMATION SYSTEM**

بسم الله الرحمن الرحيم

**Sudan University of Science and
Technology**

**College of computer science and
information technology**

Auto Database Schema Generator

التوليد التلقائي لهيكليه قاعده البيانات

Date: October 2016

By:

Esraa Eltom Elsadig Ali

Esraa Abdellatief Mohammed Ahmed

Eman Elyasa Abdelrahman Elhadi

Shaza Mahmoud Mohamed Daoud

Supervisor:

AL Sharif Hago Almugadam Yusuf

الآية القرآنية

قال تعالى: (وَإِذْ قَالَ إِبْرَاهِيمُ رَبِّ أَرِنِي كَيْفَ
تُحْيِي الْمَوْتَى قَالَ أُولِمُ تَوْمِينَ قَالَ بَلَىٰ وَلَٰكِن لِّيَطْمَئِنَّ قُلُوبُكَ قَالَ فَاخُذْ أَرْبَعَةً مِنَ الطَّيْرِ
فَصُرَّهُنَّ إِلَيْكَ ثُمَّ اجْعَلْ عَلَىٰ كُلِّ جَبَلٍ مِنْهُنَّ
جُزْءًا ثُمَّ ادْعُهُنَّ يَأْتِينَكَ سَعْيًا وَاعْلَمْ أَنَّ اللَّهَ
عَزِيزٌ حَكِيمٌ)

Dedication

Every challenging work needs self-efforts as well as guidance of elders especially those who were very close to our heart.

Our humble effort we are dedicate to our sweet and loving Fathers & mothers Whose affection, love, encouragement and prays of day and night make us able to get such success and honor

We are dedicate this work and give special thanks to our friends and our family.

ACKNOWLEDGEMENTS

First of all, we are grateful to the almighty god for establishing us to complete this research

The present study was supervised by AL Sharif Hago Almugadam Yusuf, for his suggestions, guidance, encouragement and supervision of this project.

We would like also to owe a deep sense of gratitude and thanks to department of computer science where the work was conducted.

Last but not least we are immensely grateful to our families for the unconditional support.

Table of Contents

الآية القرآنية	4
Dedication.....	5
ACKNOWLEDGEMENTS.....	5
Abstract.....	11
Chapter One.....	13
Chapter One.....	14
Introduction.....	14
1.1. Introduction.....	14
1.2Problem Statement.....	14
1.3. Research Questions.....	14
1.4. Objectives.....	15
1.5. Scope.....	15
Chapter Two.....	16
Chapter Two.....	17
Background to Generation Database Schema.....	17
2.1. Introduction.....	17
2.2. Database Schema.....	17
2.3. Database modeling	17
2.3.1. Conceptual data model.....	17
2.3.2. Logical data model.....	17
2.3.3. Physical data model.....	18
2.4 Entity Relationship Diagram ERD	19
2.4.1. ERD elements.....	19
2.5. Related Studies.....	22
2.6. Summary of previous studies.....	23
Chapter Three.....	25
Work Environment and Proposed System Analysis.....	25
Chapter Three.....	26
Work Environment and Proposed System Analysis.....	26

3.1. Introduction.....	26
3.2. Techniques and programming languages used in the system.....	26
3.2.1. Visual studio.....	26
3.2.2. Dot Net Framework.....	26
3.2.3. ASP.NET.....	26
3.2.4. C# Programming Language.....	27
3.2.5. HTML.....	27
3.2.6. CSS.....	27
3.2.7. JQuery.....	27
3.2.8. JavaScript	27
3.2.9. Microsoft SQL.....	27
3.2.10. UML technology.....	27
3.3. System Analysis.....	28
3.3.1 Use Case Diagram.....	28
3.3.2 Sequence Diagram.....	28
3.3.3 Activity Diagram.....	28
Chapter Four.....	31
Implementation.....	31
4.1. Introduction.....	31
4.2. How the system works	31
4.2.1. Draw ERD.....	32
4.2.2 Save Diagram.....	37
4.2.3 Generate Database.....	37
4.3. Case study.....	38
4.3.1. Library system.....	38
4.3.2. System Scenario.....	38
4.3.3. Result for case study.....	39
Chapter Five.....	41
Results	41
5.1. Introduction.....	41

Chapter Six.....	42
Chapter Six.....	43
Conclusion and Recommendations.....	43
6.1 Conclusion.....	43
6.2 Recommendations.....	43
References:.....	44

List of table:

Table (2.1) conceptual, logical and physical model compare.....	7
---	---

List of figure:

Figure (2.1): Entity Example in ER diagrams.....	8
Figure (2.2): Weak Entity Example in ER diagrams.....	8
Figure (2.3): Attributes Example in ER diagrams.....	9
Figure (2.4): Multivalued Attributes Example in ER diagrams.....	9
Figure (2.5): One-to-one Example in ER diagrams.....	10
Figure (2.6): One-to-Many in ER diagrams.....	11
Figure (2.7): Many-to-One in ER diagrams.....	11

Figure (2.8): Many-to-Many in ER diagrams.....	11
Figure (2.9): Many-to-Many in ER diagrams.....	12
Figure (3.1): Use case diagram for system.....	22
Figure (3.2): Sequence diagram for system.....	23
Figure (3.3): Activity diagram for system.....	24
Figure (4.1): flow chart show how ADBG work	26
Figure (4.2): The interface for ADBG	27
Figure (4.3): Type of entities in the system.....	28
Figure (4.4): Entity modal.....	28
Figure (4.5): Entity error message	28
Figure (4.6): Entity success message.....	29
Figure (4.7): Type of attributes.....	29
Figure (4.8): Attribute modal.....	30
Figure (4.9) Attribute error message.....	30
Figure (4.10): Attribute success message.....	31
Figure (4.11): Attribute connected to entity....	31
Figure (4.12): Composite attribute.....	31
Figure (4.12): Composite attribute modal for the first tap.....	32
Figure (4.13): Composite attribute modal for the second tap.....	32
Figure (4.14): Type of relationships.....	33

Figure (4.15): Relationship modal.....	34
Figure (4.16): relationship between two entities.....	34
Figure (4.17): Edit entity.....	35
Figure (4.17): modal for update entity.....	35
Figure (4.17): modal for delete entity.....	36
Figure (4.17): entity success message for update.....	36
Figure (4.17): entity success message for delete.....	36
Figure (4.18): Save image.....	36
Figure (4.18): Save modal.....	37
Figure (4.19) Database schema generate place.....	37
Figure (4.20): <i>Database generate modal</i>	38
Figure (5.1): The ERD for library system.....	39
Figure (5.2) database result of library system.....	40

Table of Terms

Abbreviation	Description	Page
ADBG	Auto Database generation	1
ER	Entity Relationship	1
ERD	Entity Relationship Diagram	1
RDBMS	Relational Database Management System	5
CODASYL	Conference/Committee on Data Systems Languages	12
SQL	Structure Query Language	13
DBMS	Database Management System	13
ADDS	Auto Database Design Schema	14
ISD	Information Structure Diagram	14
DSD	Data Structure Diagram	14
MS-DOS	Microsoft Disk Operating System	14
IDE	Integrated Drive Electronics	15
UML	Unified Modeling Language	15
GUI	Graphical User Interface	15
CLR	Common Language Runtime	16
XML	Extensible Markup Language	16
ISO	International Organization for Standardization	17
IEC	Internal Error Code	17
HTML	Hypertext Markup Language	18
CSS	Cascading Style Sheet	18
API	Application Programming Interface	18
OO	Object Oriented	19

Abstract

Conceptual modeling is a very important phase in designing a successful database application. A database is defined as a collection of data which is accessed by more than one person and probably used for more than one purpose.

The aim of this study is to Available software tools which take the database designer from the data analysis stage, through to physical database design and generate the database schema automatically by drawing entity relationship diagram.[4]

To study the problem and test the proposed solution has been to apply the following steps: analysis requirement and determine the (Entity, Attributes, Relationships).Then drawing entity relationship and testing the three phases of structural databases: conceptual (Includes the important entities and the relationships among them.), logical (describes the data in as much detail as possible, without regard to how they will be physical implemented in the database) and physical (represents how the model will be built in the database.). Then generate the database schema automatically, the diagram will be saved in the database allocated to the program, which enable you to retrieve the diagram from the database to make any edits.

Tests results in this study showed that the system ADBG (Auto Database Generator) works Effectively in drawing entity relationship diagram and saves this scheme accurately, the results of tests also showed that the program available software tools which take the database designer from the data analysis stage, through to physical database design and generate the database schema automatically by drawing entity relationship diagram and maintains the progress of the drawing process and the generation of error-free, and finally generates database schema an accurate.

المستخلص

النمذجة المفاهيمية هو مرحلة مهمة جدا في تصميم تطبيق قاعدة بيانات ناجحة. يتم تعريف قاعدة البيانات على شكل مجموعة من البيانات التي يتم الوصول إليها من قبل أكثر من شخص واحد، وربما تستخدم لأكثر من غرض واحد. الهدف من هذه الدراسة هو توفير أدوات البرمجيات والتي تأخذ مصمم قاعدة البيانات من مرحلة تحليل البيانات، وصولاً إلى تصميم قاعدة البيانات البدني وتوليد مخطط قاعدة البيانات تلقائياً عن طريق رسم تخطيطي علاقة الكيان. [4]

لدراسة المشكلة و اختبار الحل المقترح تم تطبيق الخطوات التالية: رسم مخطط علاقة الكيان و تحديد مكوناته من (كينونات Entity، صفات Attributes، وعلاقات تربط بينهم Relationships) ، اختبار المراحل الثلاثة لهيكلية قواعد البيانات المفاهيمية conceptual (يشمل المنشآت الهامة و العلاقات فيما بينها.)، المنطقية logical (تصف البيانات في تفاصيل أكبر قدر ممكن، بغض النظر عن الكيفية التي سوف تكون البدني تنفيذها في قاعدة البيانات) و المادية physical (يمثل كيف سيتم بناء نموذج في قاعدة البيانات.)، من

ثم توليد مخطط قاعدة البيانات تلقائياً ، سيتم حفظ المخطط في قاعدة البيانات المخصصة للبرنامج ، مما يمكنك من استرجاع المخطط لاجراء اي تعديلات.

نتائج الإختبارات في هذه الدراسة أظهرت أن النظام ADBG (التوليد التلقائي

لهيكلية قاعدة البيانات (Auto Database Generator) يعمل بشكل فعال في رسم تخطيطي علاقة الكيان يحفظ هذا المخطط بصورة دقيقة .وأظهرت نتائج الاختبارات أيضا أن البرنامج يقوم بتوفير أدوات البرمجيات التي تأخذ مصمم قاعدة البيانات من مرحلة تحليل البيانات، وصولاً إلى تصميم قاعدة البيانات البدني وتوليد مخطط قاعدة البيانات تلقائياً عن طريق رسم تخطيطي علاقة الكيان ويحافظ على التقدم المحرز في عملية الرسم والجيل من خالية من الأخطاء، وأخيراً يولد مخطط قاعدة بيانات بصورة دقيقة.

Chapter One

Introduction

Chapter One

Introduction

1.1. Introduction

Conceptual modeling is a very important phase in designing a successful database application. Generally, the term **database application** refers to a particular database and the associated programs that implement the database queries and updates. [1]

Entity-Relationship-Diagram (ERD) is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of database applications. [1]

- You can translate the entity sets and relationship sets of ERD in a consistent way into relational schemas that represent the contents of the database:
 - A database that is modeled by an ERD can be represented as a set of schemas.
- For each entity set and each relationship set you generate a unique schema
 - Name the schema after the corresponding entity or relationship set in the ERD.
- Each schema has a number of columns with names that are unique in this schema
 - These usually correspond to the attributes in the ERD.

So the database developer takes a lot of time in planning and create database schema clearly. This research will be the solution for the problems in database schema created by generation of database schema automatically when using modeling by ERD ,by clicking on (image button) the specific shape for entity, attributes, and relationships and check the physical and logical data independence.

1.2 Problem Statement

The problems of this thesis are as follows:

There are no software tools available which take the database designer from the data analysis stage, through to physical database design. Some software tools are available, but they are limited in the facilities they provide, frequently they only run on certain machines, are very expensive and none of them will actually create a CODASYL schema [4]. so we want to available software tools which take the database designer from the data analysis stage, through to physical database design and generate the database schema automatically by drawing entity relationship diagram.

1.3. Research Questions

- How available tools which take the database designer from the data analysis stage, through to physical database design?
- How to generate database schema automatically by using the ERD?
- How to avoid problems that appear during creating ERD?

1.4. Objectives

The objectives of this thesis are as follows:

- To available tools which take the database designer from the data analysis stage, through to physical database design.
- To design a system that generates the database schema from drawing (ERD).

- To ensure that all ERD phases (conceptual, logical and physical) are error free before creating database schema.

1.5. Scope

For the database developers this research project provides ability to draw ERD and check the three phases of database, the conceptual data model is the first phase which includes the important entities and the relationships among them no attributes or primary key is specified, the logical data model is the second phase which designing to specify primary keys for all entities and find the relationships between them, then specify all attributes for each entity, divides larger tables to smaller tables and link them using relationships(normalization in 1NF and 2NF), then physical data model is the last phase , which is where automatically convert entities into tables , relationships into foreign keys and attributes into columns In the sense that is modifying the actual data model based on physical constraints / requirements and generate database schema automatically using SQL server.

Chapter Two

Background to Generation
Database Schema

Chapter Two

Background to Generation Database Schema

2.1. Introduction

In pervious chapter we show problem statement, research question and objective of this research. And this chapter is divided into two sections, the first section gives general description of how to create database schema that modeling using ERD, the second section describes the related studies to research project.

2.2. Database Schema

A database is defined as a collection of data which is accessed by more than one person and probably used for more than one purpose. Therefore the term database today can be applied not to only data held on a computer but also to any collection of manual records.

Designing a computer database is a very complex task. It is a lengthy and time consuming process which involves investigating the business thoroughly, understanding the transactions that will be applied against the database and then translating this into the physical design. One of the major problems with this task is that once the information has been collected, it must then be cross-referenced, collated, documented and then used to formulate the design. When a system comprises hundreds of data items and tens of transactions, this is a very difficult manual task.

All of the database management systems currently available are based on a data model, which simply represents the information held in the database as it is viewed by the users of the database. There are a number of different models are available for different

purposes. In database work three types of data model are generally available, the hierarchical, network and relational model.

All of these models are defined using entities, attributes and relationships. An entity is defined as the basic unit about which data can exist e.g. a client and it is described in terms of its attributes. Typical attributes of the client entity would be name, age and date of birth. A relationship is then defined between entities. Possible relationships are one-to-one, one-to-many and many-to-many e.g. One client may have many policies, therefore a one-to-many relationship exists between the entities client and policy. This research limited for just ERD.

2.3. Database modeling

The three level of data modeling:-

2.3.1. Conceptual data model

The conceptual data model identifies the highest-level relationships between the different entities. Features of conceptual data model include:

- Includes the important entities and the relationships among them.
- No attribute is specified.
- No primary key is specified. [2]

2.3.2. Logical data model

The logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Features of a logical data model include:

- Includes all entities and relationships among them.
- All attributes for each entity are specified.
- The primary key for each entity is specified.

- Foreign keys (keys identifying the relationship between different entities) are specified.
- Normalization occurs at this level.

The steps for designing the logical data model are as follows:

- Specify primary keys for all entities.
- Find the relationships between different entities.
- Find all attributes for each entity.
- Resolve many-to-many relationships.
- Normalization. [2]

2.3.3. Physical data model

The physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables. Features of a physical data model include:

- Specification all tables and columns.
- Foreign keys are used to identify relationships between tables.
- De-normalization may occur based on user requirements.
- Physical considerations may cause the physical data model to be quite different from the logical data model.
- Physical data model will be different for different RDBMS. For example, data type for a column may be different between Oracles, DB2 etc.

The steps for physical data model design are as follows:

- Convert entities into tables.
- Convert relationships into foreign keys.

- Convert attributes into columns.
- Modify the physical data model based on physical constraints/requirements. [2]

Here we compare these three types of data models. The table (2.1) compares the different features:

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	✓
Foreign Keys		✓	✓
Table Names			✓
Column Names			✓
Column Data Types			✓

Table (2.1) Conceptual, logical and physical model compare.[3]

2.4 Entity Relationship Diagram ERD

Is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts. We describe the basic data-structuring concepts and constraints of the ERD model and discuss their use in the design of

conceptual schemas for database applications. We also present the diagrammatic notation associated with the ERD model, known as **ERD**. [3]

2.4.1. ERD elements

There are three basic elements in an ERD: entity, attribute, relationship. There are more elements which are based on the main elements. They are weak entity, multivalued attribute, derived attribute, weak relationship and recursive relationship. Cardinality and ordinarily are two other notations used in ERDs to further define relationships. [3]

2.4.1.1. Entity

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ERDs by a rectangle and named using singular nouns. [3]



Figure (2.1): Entity Example in ER diagrams. [3]

2.4.1.2. Weak Entity

A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attributes to form the primary key. An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of order. [3]



Figure (2.2): Weak Entity Example in ER diagrams. [3]

2.4.1.3. Attribute

An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own specific attributes. For example, the attribute “customer address” can have the attributes number, street city, and state. These are called composite attributes. Note that some top level ER diagrams do not show attributes for the sake of simplicity. In those that do, however, attributes are represented by oval shapes. [3]

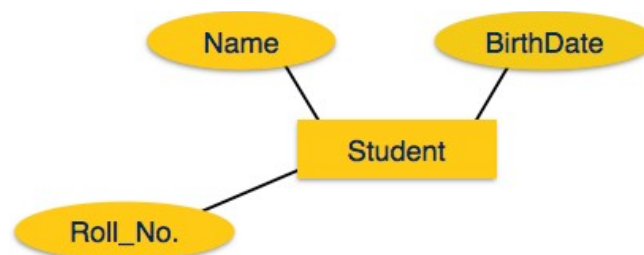
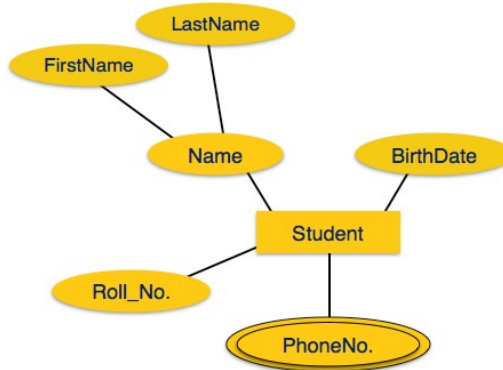


Figure (2.3): Attributes Example in ER diagrams [3]

2.4.1.4. Multivalued attribute

If an attribute can have more than one value it is called a multivalued attribute. It is important to note that this is different to an attribute having its own attributes. For example a teacher entity can have multiple subject values. [3]

Figure (2.4): Multivalued Attributes Example in ER diagrams. [3]



2.4.1.5. Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line. [3]

2.4.1.5.1. Binary Relationship and Cardinality

A relationship where two entities are participating is called a binary relationship. Cardinality is the number of instance of an entity from a relation that can be associated with the relation:

- **One-to-one** –When only one instance of an entity is associated with the relationship; it is marked as '1:1'. The following image reflects that only one instance of each

entity should be associated with the relationship. It depicts one-to-one relationship. [3]

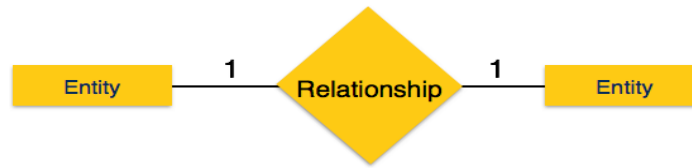


Figure (2.5): One-to-one Example in ER diagrams. [3]

- **One-to-many** –When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship. [3]

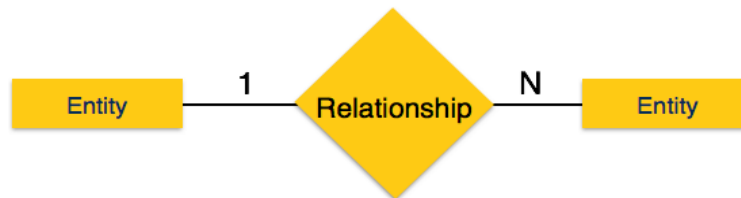


Figure (2.6): One-to-Many in ER diagrams. [3]

- **Many-to-one** –When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship. [3]

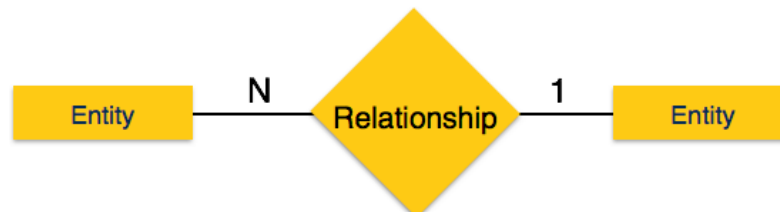


Figure (2.7): Many-to-One in ER diagrams. [3]

- **Many-to-many** – The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship. [3]

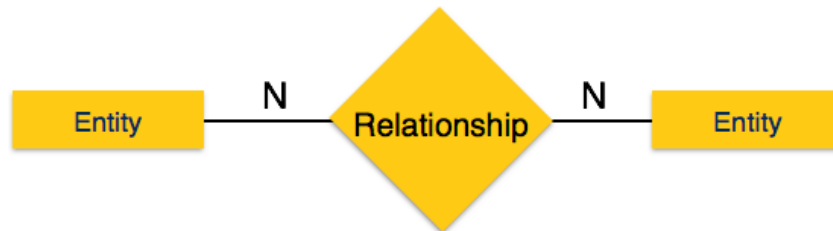


Figure (2.8): Many-to-Many in ER diagrams. [3]

2.4.1.5.2. Participation Constraints

- **Total Participation** – Each entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation** – Not all entities are involved in the relationship. Partial participation is represented by single lines. [3]

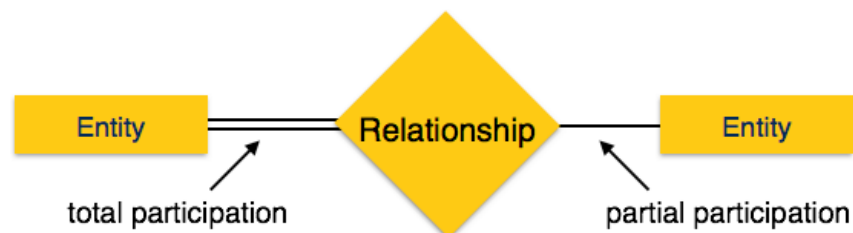


Figure (2.9): Many-to-Many in ER diagrams. [3]

2.5. Related Studies

2.5.1 Automatic generation of database schema

“This research shows is that it is possible to develop a computer system which performs all of these tasks. Using the entity model as the starting point for the design, the attributes are defined. Then the transaction are described as text and verified, using a natural language processor against the information already entered into the system. Finally the schemas for a CODASYL compliant database are created. The objective of this thesis was to show that it was possible to develop a computer system that could be used from Data Analysis through to Database Design, the final product being a set of schemas that would be used to actually create the database, a requirement that as one can see has been shown to be feasible.

The system creates the schemas necessary to build the database, and it has been found that the schemas generated only required minor amendment. Multiple versions of the schemas may be retained so it is possible to see the changes that have been applied. The only restrictions with the schemas is that it is not possible to specify the order in which the attributes will appear on the schema. If a set is sorted the rules for duplicates cannot be specified and there is no ability to define a check clause between records and sets.

However, these problems are minor compared to the work that the system has done. These schemas were never envisaged as being the master database definitions that would never require amendment. Instead their purpose is to ensure that nothing important has been omitted and they are there for the designer to use his skills to amend to suit the needs of the application. As was stated at the outset, this software does not set out to replace the database designer, it is merely a tool used to assist during database design. “[4]

2.5.2. SQL Mutation: A tool to generate mutants of SQL database queries

“This paper presents a tool to automatically generate mutants of SQL database queries. The SQL Mutation tool is available on the Web and it can be accessed using two different interfaces: A Web application to interactively generate the mutants and a Web service that allows it to be integrated with other applications developed using different platforms. “[5]

2.5.3. Automatic generation of database queries

“A computer program for automatically generating queries to a database management system (DBMS) is disclosed. The computer program receives a high level specification of the data sought to be retrieved from an application program. The high level specification includes the columns from which data is sought, and any constraints on the data to be retrieved from those columns such as filter constraints. The computer program also receives a context for the columns to be queried, so that it can be determined which tables of the DBMS the columns are associated with. The computer program further receives a specification of the schema of the DBMS, with the schema specifying the relationship between the various data storage entities of the DBMS. Using the high level specification and the schema, the computer program automatically generates queries to the DBMS seeking to reduce the complexity of the queries to speed execution, and to reduce the number of round trips between the computer program and the DBMS, also to enhance speed. For this purpose, queries may be separated into primary and secondary queries, and batched together, with secondary queries using data retrieved in a primary query in order to retrieve further data.” [6]

2.5.4. ADDS: A System for automatic database schema design based on the binary-relationship model

“This paper presents the system ADDS that has been developed to assist the database designer designing a database schema. A distinction is made between the stage of information structure analysis in which the information structure of the system is defined according to its user information needs, and the stage of database schema design in which the record types of the database and the relationships between them are defined. In the first stage a conceptual schema is obtained, represented as an information structure diagram (ISD), and in the later stage the ISD is used to derive the database schema in the form of a data structure diagram (DSD). ADDS automatically creates the database schema out of a conceptual schema which is expressed as an ISD of the binary-relationship data mode. The resulting schema consists of normalized record types, according to the relation model, along with hierarchical/set relationships between ‘owner’ and ‘member’ record types, as in the CODASYL/Network model. ADDS applies algorithms to convert the conceptual schema into the database schema. It is implemented on a micro-computer under MS-DOS using dBASE III. “[7]

2.6. Summary of previous studies

According to studies (2.5.1, 2.5.2, 2.5.3, 2.5.4) that are mentioned above, analysis the user's requirement (the entities and attributes are defined, then the transaction are described as text and verified) it is the starting point to design and create the database schema. They use class diagram as start point and create the quires of database.

Reviewing all these systems certainly indicates that there is a need for a tool that will provide all of these tasks and create physical database schema.

So this system a useful tool to analysis the database requirements and check the three phases of database from drawing ERD. The first check is checking the conceptual model (from defined the entities as tables and the relationships between them by drawing them at the sketch side), the second one is checking the logical model (check the database constraints as validations about defining the attributes for these entities), then the last check it's for physical model (creating the database schema in SQL server).

Chapter Three

Work Environment and Proposed System Analysis

Chapter Three

Work Environment and Proposed System Analysis

3.1. Introduction

In pervious chapter we show general description of how to create database schema that modeling using ERD and related studies to research project. In this chapter we describes specification of operating system, programming language, and techniques used to build the system. Then describes the system analysis using UML Diagrams.

3.2. Techniques and programming languages used in the system

3.2.1. Visual studio

Visual Studio.NET is a Microsoft-integrated development environment (IDE) that can be used for developing consoles, graphical user interfaces (GUIs), Windows Forms, Web services and Web applications.

Visual Studio is used to write native code and managed code supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight. Visual Studio .NET's code editor supports IntelliSense and code refactoring, while the Visual Studio .NET integrated debugger supports both source and machine-level debugging. Visual Studio .NET includes other built-in tools, like a form designer, which is useful when building GUI applications; a Web designer that creates

dynamic Web pages; a class designer that is used to create custom libraries, and a schema designer for database support.[8]

3.2.2. Dot Net Framework

The .NET Framework is a technology that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives

To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.

- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code. [9]

3.2.3. ASP.NET

ASP.NET is a unified web development model integrated with .NET framework, designed to provide services to create dynamic web applications and web services. It is built on the Common Language Runtime (CLR) of the .NET framework and includes those benefits like multi-language interoperability, type safety, garbage collection and inheritance. [10]

3.2.4. C# Programming Language

C# (pronounced “See Sharp”) is a simple, modern, object-oriented, and type-safe programming language. C# has its roots in the C family of languages and will be immediately familiar to C, C++, and Java programmers. C# is standardized by ECMA International as the **ECMA-334** standard and by ISO/IEC as the **ISO/IEC 23270** Standard.

Microsoft’s C# compiler for the .NET Framework is a conforming implementation of both of these standards C# is an object-oriented language, but C# further includes support for **component-oriented** programming. Contemporary software design increasingly relies on software components in the form of self-contained and self-describing packages of functionality. Key to such components is that they present a programming model with properties, methods, and events; they have attributes that provide declarative information about the component, and they incorporate their own documentation. C# provides language constructs to directly support these concepts, making C# a very natural language in which to create and use software components. [11]

3.2.5. HTML

HTML is stand for Hypertext Markup Language is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page. The markup tells the Web browser how to display a Web page's words and images for the user. Each individual markup code is referred to as an element (but many people also refer to it as a tag). Some elements come in pairs that

indicate when some display effect is to begin and when it is to end. [12]

3.2.6. CSS

CSS stands for Cascading Style Sheets, describe the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments. This is referred to as the separation of structure (or: content) from presentation. [13]

3.2.7. JQuery

JQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, JQuery has changed the way that millions of people write JavaScript. [14]

3.2.8. JavaScript

JavaScript is a programming language commonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced by Java, the syntax is more similar to C and is based on ECMAScript, a scripting language developed by Sun Microsystems. [15]

3.2.9. Microsoft SQL

SQL Server 2016 is a version of Microsoft's relational database management system (RDBMS) that first became available in preview releases during 2015, with general availability on June 1, 2016. SQL Server 2016 is a SQL-based database designed to support a mix of

transaction processing, data warehousing and analytics applications in enterprise environments. [16]

3.2.10. UML technology

Unified Modeling Language (UML) is the industry standard language for describing, visualizing, and documenting object-oriented (OO) systems. UML is a collection of a variety of diagrams for differing purposes. Each type of diagram models a particular aspect of OO design in an easy to understand, visual manner. The UML standard specifies exactly how the diagrams are to be drawn and what each component in the diagram means. UML is not dependent on any particular programming language, instead it focuses on the fundamental concepts and ideas that model a system. Using UML enables anyone familiar with its specifications to instantly read and understand diagrams drawn by other people [17].

3.3. System Analysis

3.3.1 Use Case Diagram

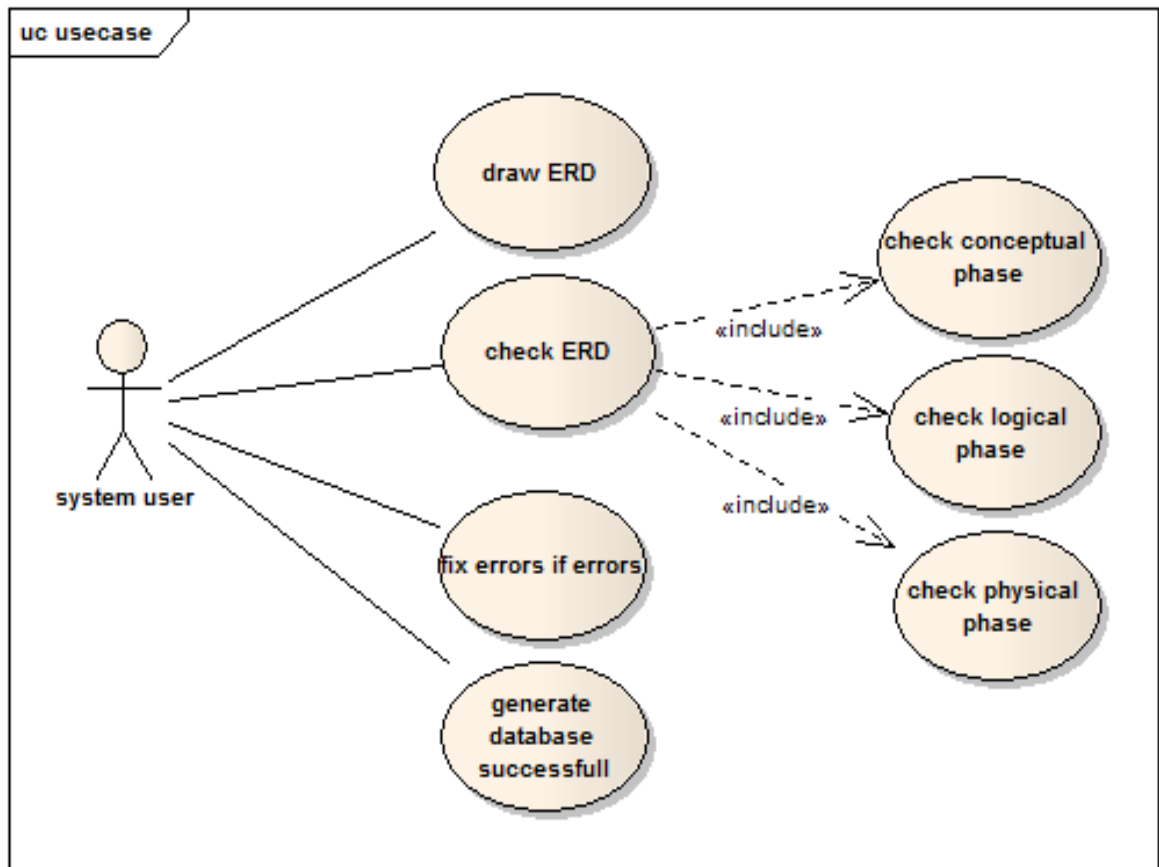


Figure (3.1): Use case diagram for system

3.3.2 Sequence Diagram

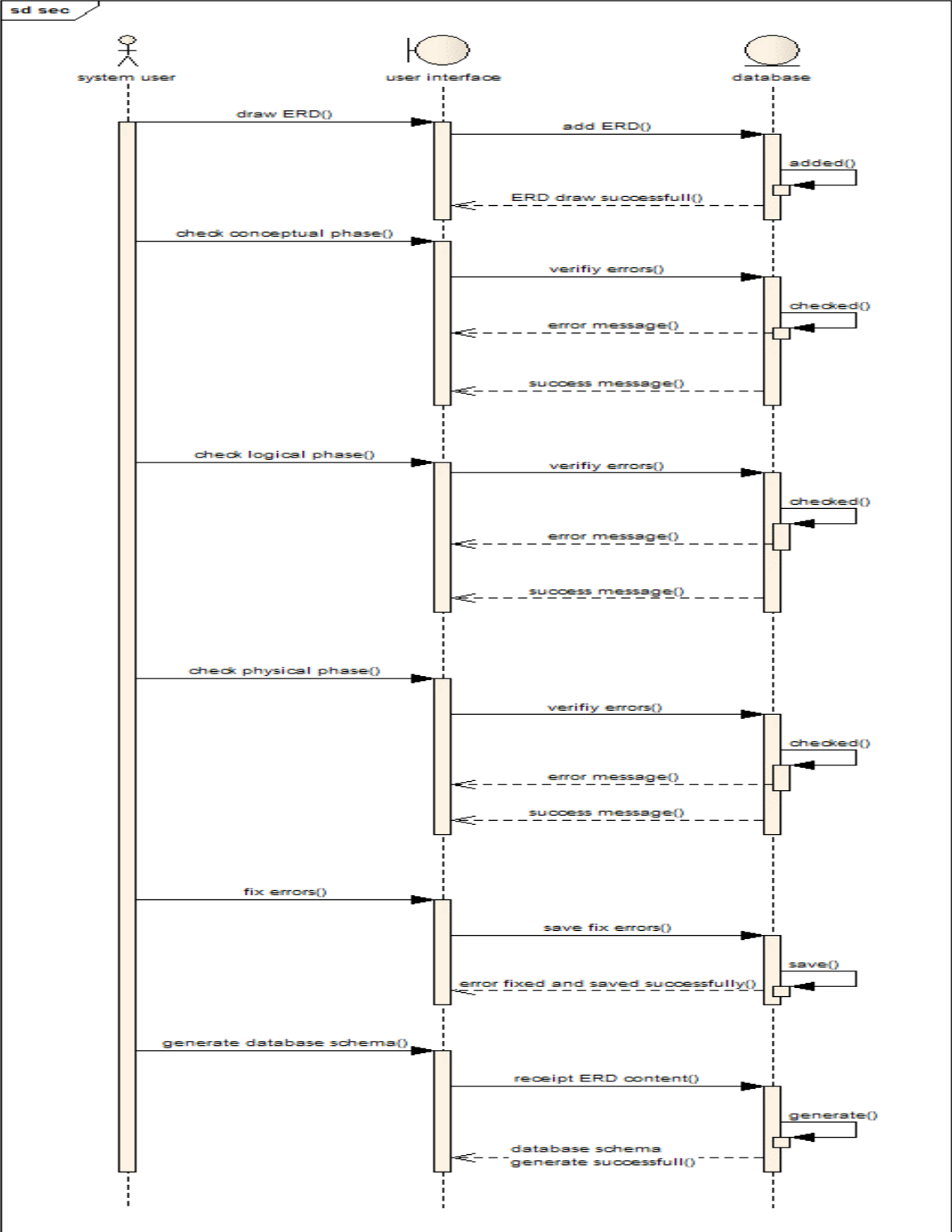
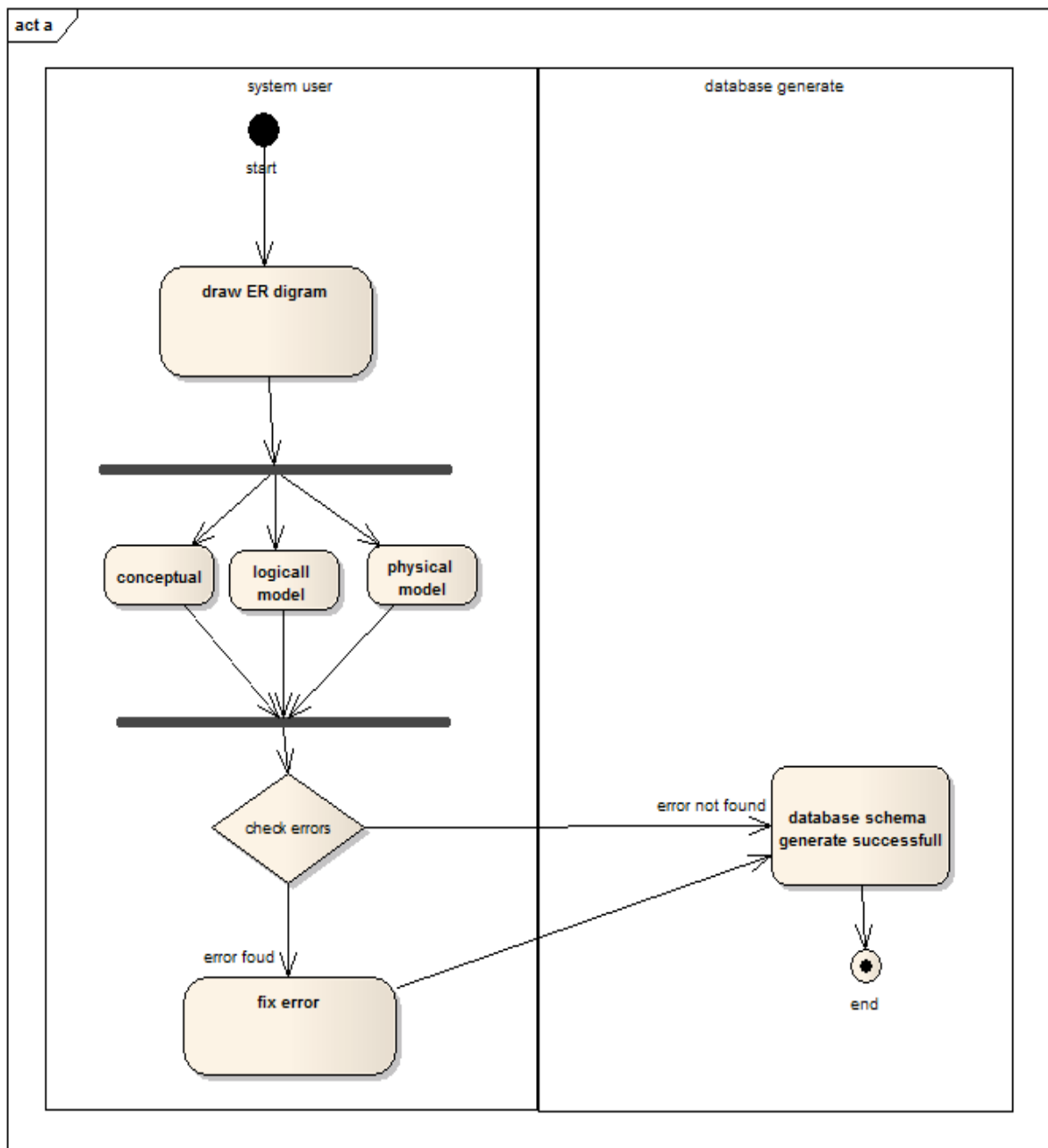


Figure (3.2): Sequence diagram for system



3.3.3 Activity Diagram

Figure (3.3): Activity diagram for system

Chapter Four

Implementation

Chapter Four

Implementation

4.1. Introduction

In previous chapter we show describes specification of operating system, programming language, and techniques used to build the system. Then describes the system analysis using UML Diagrams. And in this chapter we shows how the system works and includes the explanation stages of database generation and drawing the ERD with checking the database phases (conceptual, logical and physical) case study (Library system).

This website is creating ERD from scratch, since complex shapes and connectors are pre-built and easy to move throughout the modeling process. For the purposes of this guide, we'll proceed as though you're using ADBG. To describe how system work, follow these steps in Figure (4.1):

4.2. How the system works

This section shows how the system works and includes the explanation to how draw ERD and check the three phases of database .The first activity is draw ERD(contain entities, attributes, relationships) after you draw check for (conceptual, logical, physical)will be occur as follow conceptual data model is the first phase make sure it includes the important entities and the relationships among them no attributes or primary key is specified, the logical data model is the second phase you must specify primary keys for all entities and find the relationships between them, then specify all attributes for each entity, and it contain normalization (in1NF

and 2NF), then physical data model is the last phase , which is where automatically convert entities into tables , relationships into foreign keys and attributes into columns In the sense that is modifying the actual data model based on physical constraints / requirements and generate database schema automatically using SQL server. if no error appear the database schema will generate successfully , if any error appear message of error will

appear to make you fix the error and then generate the database schema.as we show in Figure (4.1).

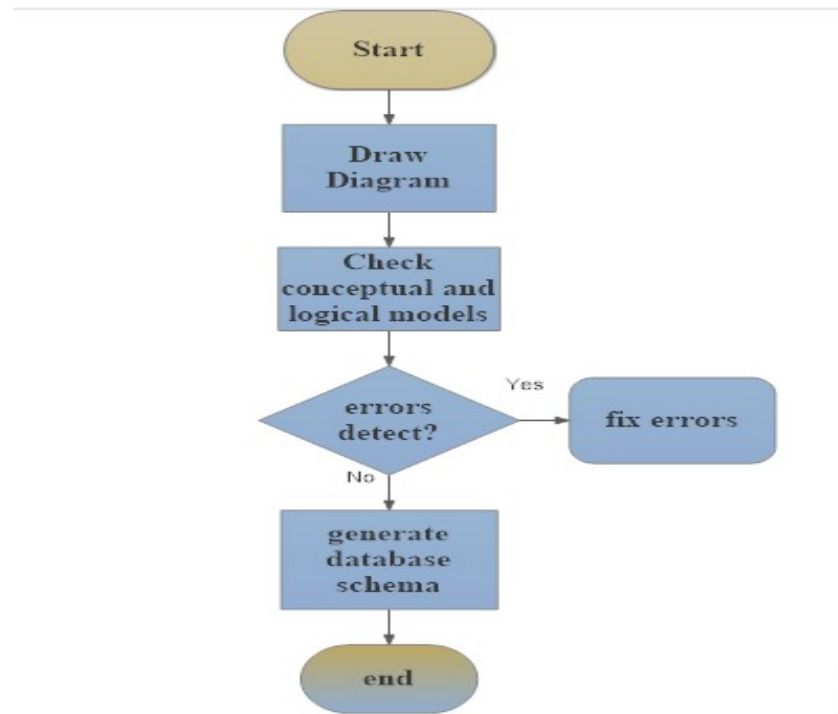


Figure (4.1): flow chart show how ADBG work

ADBG is drawing interface divided into three sides. The top of the interface is contain specific characteristics (opening new interface, save the model, open a model already saved, generating database, test whether the database was created or not and help).The left of the interface is contain ERD component (Entities, attributes and relationships) and edit (update and delete).



Figure (4.2): The interface for ADBG

4.2.1. Draw ERD

ADBG is built specifically for creating database schema automatically by drawing ERD.

Drawing ERD is first step in this system in creating database schema that define the required entities, attributes and relationships.

Before you start clicking on shapes and labeling them, be sure to differentiate between (entities, attributes and relationships), determine significant interactions between them, and analyze the nature of the interactions.

There's one more concept you should understand before creating an ERD: data model levels. An entity-relationship diagram will occupy one of three data modeling levels: conceptual, logical, or physical.

4.2.1.1. Draw Entities

This step is to draw the entities (tables in database) by clicking on entity shape from the toolbox, which represented by rectangle for simple entities and double rectangle for weak entities.

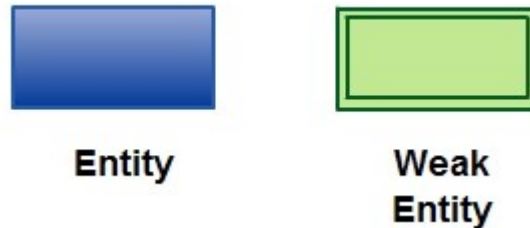


Figure (4.3): Type of entities in the system

When clicking on entity or weak entity shape modal that contain text field to name entity or weak entity will appear as shown in figure (4.4).

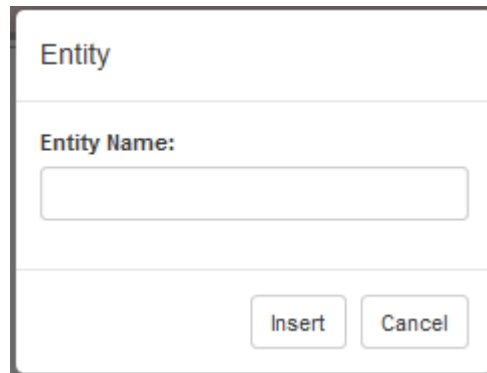
The image shows a modal dialog box titled 'Entity'. It contains a label 'Entity Name:' followed by a text input field. At the bottom right, there are two buttons: 'Insert' and 'Cancel'.

Figure (4.4): Entity modal

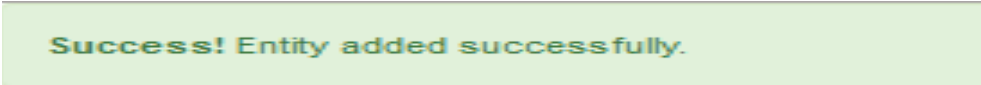
Be sure that the entity name is not empty, already used and start with (numbers, signs). If you did not abide by these constraints error message that contain the type of error will appear as shown in figure (4.5).

Example: when using name already used:

Error! Entity name cannot be duplicated.

Figure (4.5): Entity error message

If you abide by constraints entity or weak entity will be drew successfully!

A light green rectangular box with a thin black border containing the text "Success! Entity added successfully." in a dark green font.

Figure

(4.6): Entity success message

Starting with this step is very important; because drawing attributes and relationships required specifying which entity they belong to.

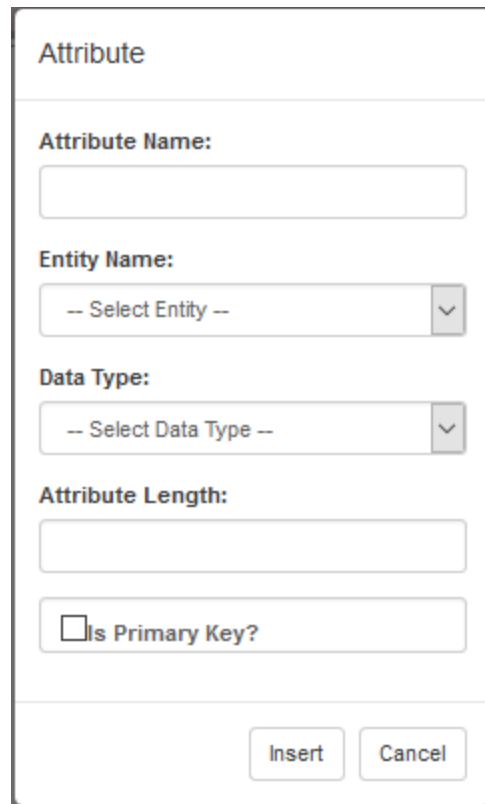
4.2.1.2. Draw Attributes

After you draw entities the next step it is draw the attributes by clicking on attribute shape from the toolbox, which represented by oval for simple attributes, dotted ovals for derived attributes and double ovals for multivalued attributes.



Figure (4.7): Type of attributes

When clicking on attribute or derived attribute or multivalued shape modal will appear as shown in figure (4.8) contain text field for attribute name, entity that the attribute belong to (drop down list filled automatically of entities that added before) , data type, attribute length field and check box that determine if attribute is

A modal form titled "Attribute" with the following fields: "Attribute Name:" (text input), "Entity Name:" (dropdown menu with "-- Select Entity --"), "Data Type:" (dropdown menu with "-- Select Data Type --"), "Attribute Length:" (text input), and "Is Primary Key?" (checkbox). At the bottom are "Insert" and "Cancel" buttons.

primary key or not. The derived and multivalued attributes will never be primary key.

Figure (4.8): Attribute modal

Be sure that the attribute name is not empty, already used and start with (numbers, signs). If you did not abide by these constraints error message that contain the type of error will appear as shown in figure (4.9).

Example: when naming two attributes with the same name within same entity:

Error! Attribute name cannot be duplicated in the same entity.

Figure (4.9) Attribute error message

If you abide by constraints attribute or derived attribute will be drew and connected automatically with entity that selected from entities list by flexible line.

Success! Attribute added successfully.

Figure (4.10): Attribute success message

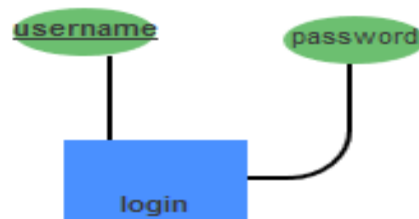


Figure (4.11): Attribute connected to entity

4.2.1.3 Draw composite attribute

To draw composite attribute clicking on the shape from the toolbox

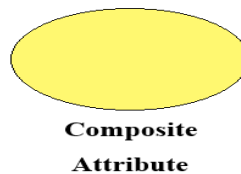


Figure (4.12): Composite attribute

When clicking on composite attribute shape modal that contain two taps tap for main attribute contain text field for name attribute and the other tap contain child attribute name, composite attribute

name data type and attribute length will appear as shown in figure (4.13).

Figure (4.13): Composite attribute modal for the first tap

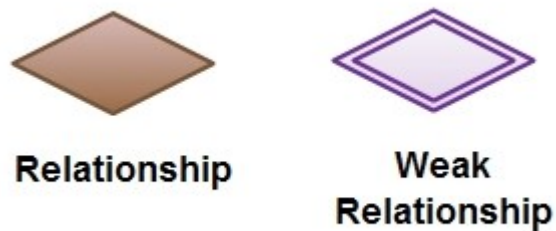
The screenshot shows a modal window titled "Attribute". At the top, there are two tabs: "Attribute" (selected) and "Child Attribute". Below the tabs, there is a text input field labeled "Attribute Name:". Below the input field is an "Insert" button. At the bottom right of the modal is a "Cancel" button.

Figure (4.14): Composite attribute modal for the second tap

The screenshot shows a modal window titled "Attribute". At the top, there are two tabs: "Attribute" and "Child Attribute" (selected). Below the tabs, there is a text input field labeled "Child Attribute Name:". Below that is a dropdown menu labeled "Composite Attribute Name:" with the text "-- Select Entity --". Below that is another dropdown menu labeled "Data Type:" with the text "-- Select Data Type --". Below that is a text input field labeled "Attribute Length:". Below the input field is an "Insert" button. At the bottom right of the modal is a "Cancel" button.

4.2.1.4 Draw Relationships

Link shapes with specialized connectors that express both cardinality and ordinary. Cardinality specifies how many instances of an entity relate to another instance of an entity, while ordinary describes the relationship as either mandatory or optional. In ADBG, cardinality can be shown by drawing a diamonds and changing its name shown in the shape to indicate a one-to-one relationship, one-to-many relationship, or any other relationship. You can demonstrate ordinary by drawing either a single line, the former indicates a relationship between entities, while the latter shows a constraint that



forces total participation in the relationship.

Figure (4.15): Type of relationships

When click on relationship or weak relationship shapes modal that contain relationship name, primary table/attribute, foreign/attribute and relationship type will appear as shown in figure (4.16).

Relationship

Relationship Name:

Primary Table/ Attribute:

Foriegn Table/ Attribute:

Relationship Type:

Figure (4.16): Relationship modal

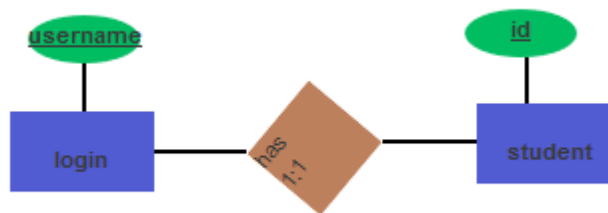


Figure (4.17): relationship between two entities

You can edit (update, delete) diagram during draw diagram by clicking on image of edit of shape that you want to edit it.as we show in Figure (4.18)

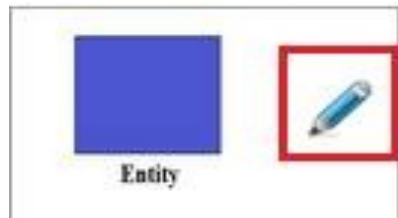


Figure (4.18) edit entity

When click on edit shapes modal that contain two tapes will appear as shown in figure (4.19), figure (4.20). Tap for update contain(drop down list of entity that found, text to insert new entity) and tap for delete contain (drop down list of all found entity) .

The image shows a modal window titled 'Edit Entity'. At the top, there are two buttons: 'Update' and 'Delete'. Below them is a 'Select Entity:' label followed by a dropdown menu showing 'login'. Underneath is a 'New Name:' label followed by a text input field containing 'employee'. At the bottom left is a 'Modify' button, and at the bottom right is a 'Cancel' button.

Figure (4.19) modal for update entity

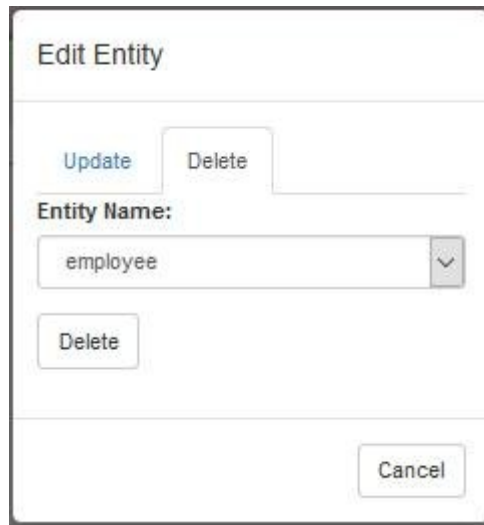


Figure (4.20) modal for delete entity

When edit (update, delete) finish message that show you the update or delete successful will be appear as we shown in figure (4.21), figure (4.22)



Figure (4.21) entity success message for update



Figure (4.22) entity success message for delete

4.2.2 Save Diagram

After you draw the ERD you can save the diagram by clicking on the image of save that appear as shown in figure (4.23):



Figure (4.23): Save place

When clicking on save modal that contain diagram name will appear as shown in figure (4.24).

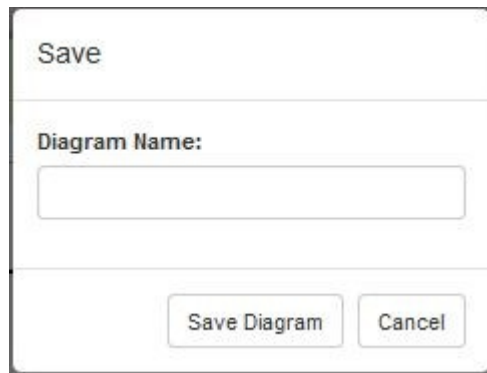


Figure (4.24): Save modal

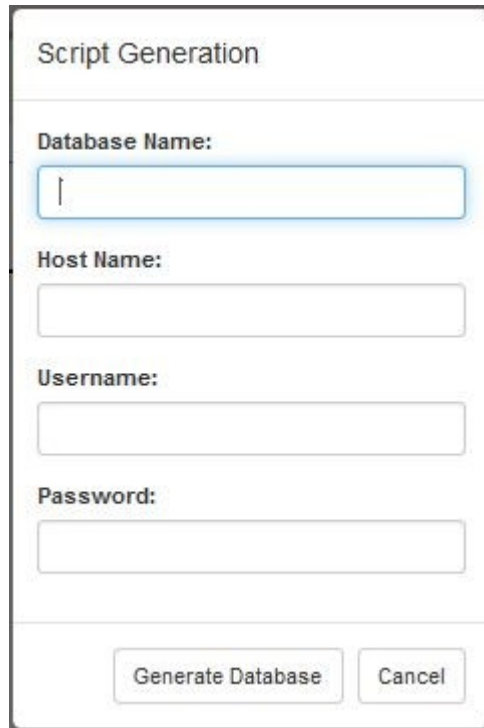
4.2.3 Generate Database

After you draw the whole ERD now this step about generate automatically database schema by clicking on image button for database generation that appear as shown in figure (4.25).



Figure (4.25) Database schema generate place

When clicking on generate database image button modal that contain database name, host name, username and password will appear as shown in figure (4.26).



The image shows a modal window titled "Script Generation". It contains four input fields: "Database Name:" (with a cursor), "Host Name:", "Username:", and "Password:". At the bottom, there are two buttons: "Generate Database" and "Cancel".

Figure (4.26): Database generate modal

4.3. Case study

4.3.1. Library system

Library management system is basically updating the manual library system into an internet-based application so that the librarian can know the details of availability of books and maximum limit for borrowing.

The librarian will be acting as the controller and he will have all the privilege of an administrator.

4.3.2. System Scenario

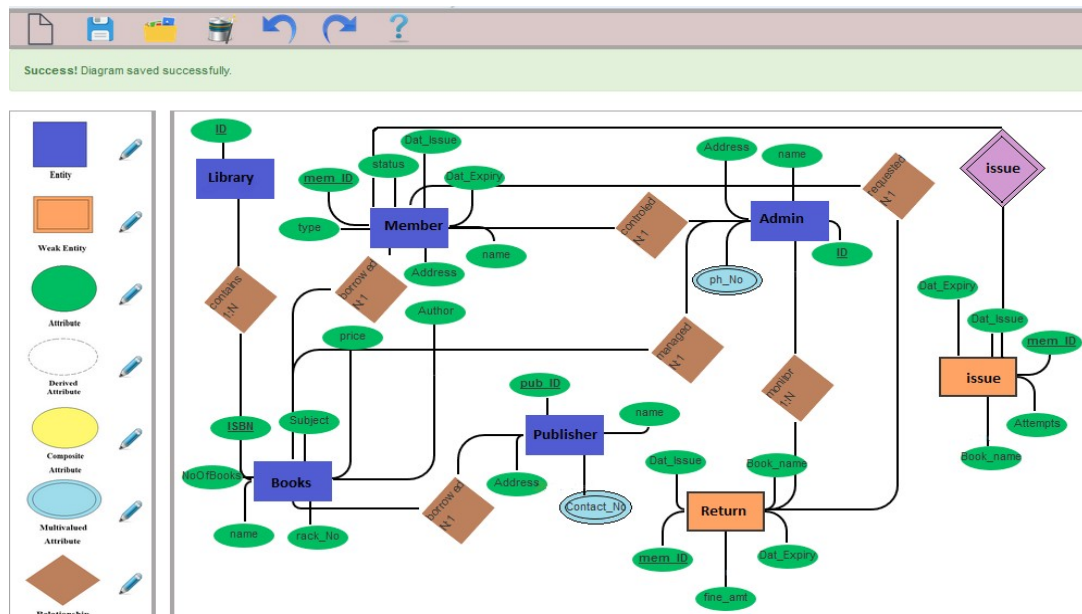
This system was applied to the Sudan University of Science and Technology Library, which consists a number of books and each book contains (the subject of the book, the book name , the author's name, book number, number of rack it is located, the price). The members and librarian interact with system, system can provide information about member (Name, Type, Member ID, Status, Address, Date_Issue, Date_Expiry) and librarian (ID, Name, Address).

The system contain tow interfaces one for student and the second for librarian, the first one contain information about books and student can reserve the book and determines borrowing and retrieval time. And the other one contain all the information about books and reserved books and retrieval time, to impose a fine if it did not retrieve the book at the same specified when booking process.

The member's status of issue/return is maintained in the library database. The member's details can be fetched by the librarian from the database as and when required.

We have been using ADBG system to generate database schema automatically for the above system.

Figure (4.27): The ERD for library system



4.3.3. Result for case study

The last output of this system is that the database schema has been generated in SQLServer.

In SQLServer after creating the database a database diagram will be generated automatically as shown in figure (4.28).

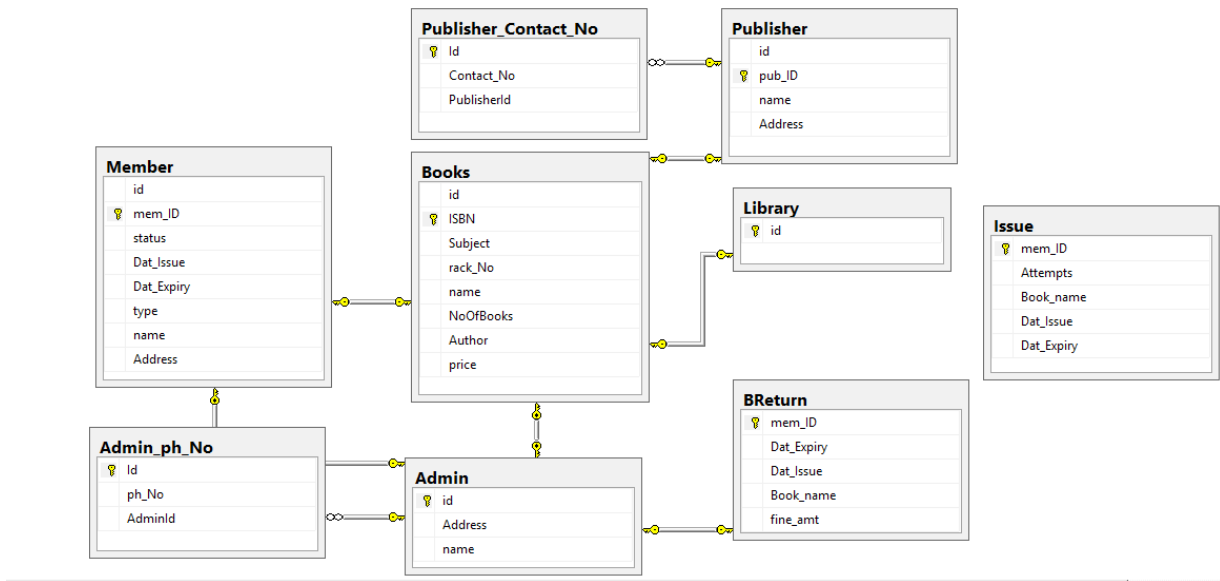


figure (4.26)database result of library system.

Chapter Five

Result and Discussion

Chapter Five

Results

5.1. Introduction

In pervious chapter we show how the system works and the case study (Library system).And finally in this chapter we describes result of ADBG system.

From this implementation we extracted the following results:

- 1- available tools which take the database designer from the data analysis stage, through to physical database design.
- 2- Reduce user time in the processes of drawing ERD and generating database schema.
- 3- Check for three phases of database (conceptual, logical and physical), and ensure that the diagram free of errors.
- 4- Generating the database schema automatically is successful.

Chapter Six

Conclusion and Recommendation

Chapter Six

Conclusion and Recommendations

6.1 Conclusion

In pervious chapter we show the result of ADBG system. And in this chapter we show the conclusion and recommendation.

System has been developed to analyze user requirements by drawing ERD and insure that the diagram is error-free then generate the database schema automatically.

The main idea of this research is to solve the problems that appear in ERD that drew manually and help the user to create database in easy way.

6.2 Recommendations

To make this system more usable and scalable we recommend to do the following:

- Generate database schema from all other types of database models (hierarchical model and network models).
- Use other techniques to draw the (entities, attributes and relationships) to enhance moving resizing the shapes.
- Support database generation using (Oracle, Microsoft Access, PostgreSQL, MySQL etc.).
- Security of the database.
- Add Binary Relationship and Cardinality (Many-To-Many).

- Add Participation Constraints (total and partial participation).
- Add Normalization in logical phase.

References:

- [1] Ramez Elmasri, Shamkant B. Navathe "fundamentals of database systems" 20 December 2015.
- [2][Online] <http://www.1keydata.com/datawarehousing/data-modeling-levels.html> 1 october 2016 ,at 10:18AM
- [3][Online] https://www.tutorialspoint.com/dbms/er_diagram_representation.htm ER digram 2 march 2016 ,at 19:20 pm.
- [4][Online] [Hobbs, L. M. Automatic generation of database schema. Diss. University of Southampton, 1987](#) 29 January 2016 ,at 16:26 pm.
- [5] [Online]<https://www.computer.org/csdl/proceedings/mutation/2006/2897/00/28970001-abs.html> SQL mutation 29 January 2016 ,at 14:19 pm.
- [6][Online] [Tabbara, Bassam. "Automatic generation of database queries." U.S. Patent No. 6,148,296. 14 Nov. 2000](#) 29 January 2016 ,at 14:17 pm.
- [7][Online] <http://www.sciencedirect.com/science/article/pii/S0169023X87900188> adds 29 January 2016 ,at 14:22 pm.
- [8][Online] <https://www.techopedia.com/definition/15740/visual-studio-net> 10 may 2016 ,at 10:39 PM
- [9][Online] [https://msdn.microsoft.com/enus/library/zw4w595w\(v=vs.110\).aspx](https://msdn.microsoft.com/enus/library/zw4w595w(v=vs.110).aspx) 10 may 2016 ,at 11:30 AM
- [10] [Online] <https://www.techopedia.com/definition/3213/asp-net> 10 may 2016 ,at 12:10 AM
- [11] [Online] [The C# Programming Language Anders Hejlsberg Scott Wiltamuth Peter Golde](#) 10 may 2016 ,at 12:11 PM.

[12][Online] <http://searchsoa.techtarget.com/definition/HTML> 10 may 2016 ,at 12:50 pm.

[13][Online] <http://www.w3schools.com> 2016 ,at 11:30 AM.

[14][Online] <https://jquery.com> 10 may 2016, at 1:30 pm._____

[15][Online] <http://techterms.com/definition/javascript> 10 may 2016 ,at 1:40 pm.

[16] [Online]
<http://searchsqlserver.techtarget.com/definition/Microsoft-SQL-Server-2016> 10 may 2016 ,at 3:33 PM

[17] [Online]
<http://www.ibm.com/developerworks/rational/library/769.html> 11 may 2016 , at 10:12 AM.

