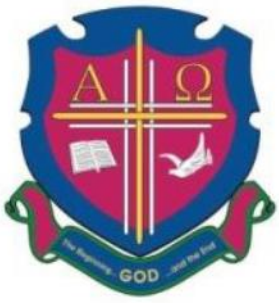


UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

FACULTY OF ENGINEERING DESIGN AND TECHNOLOGY

NAME: MOGA MUZAMIL ABDUL WAHAB

REG NO: S21B23/013

ACCESS NO: A94166

COURSE: BACHELOR OF SCIENCE IN COMPUTER SCIENCE (BSCS)

COURSE UNIT: DATABASE DESIGN AND APPLICATION

LECTURER: MR. SIMON FRED LUBAMBO

Take-home questions:

I. Define Normalization and how it has been applied in your project. **(5 Marks)**
II. Define the different transaction anomalies giving examples from your project and how they can occur. **(10 Marks)**
III. Suggest ways in which the security of your database can be enhanced **(5 Marks)**

Qn.I Define Normalization and how it has been applied in your project.

Normalization is the process of organizing data into tables in such a way that the results of using the database are always unambiguous and is as intended for use.

It reduces data redundancy and eliminates inconsistent dependency and undesirable characteristics like insertion, Update and Deletion Anomalies and makes the database more flexible.

Normalization brings the tables to a state where issues of update, deletion and insert anomalies can be avoided. In simple words it splits tables into multiple tables and defines relationships between them using keys.

HOW I HAVE APPLIED NORMALIZATION IN MY HOSPITAL DATABASE:

Since the table patient is used for the purpose of keeping patient records, and my patient table had patient_id, names, occupation, age, gender, disease, address_location, the admission_discharge_id, ward_no, date_of_admission, date_of_discharge, bill, amout_paid, balance, pharmacy_id, medicine_name and price.

I then split it into 3 three tables the table patient having patient_id, names, occupation, age, gender, disease and address_location.

The second table I created to contain a continuation of patient records is the admission_disharge table consisting of the admission_discharge_id, ward_no, date_of_admission, date_of_discharge, bill, amout_paid and balance of the patient without repeating the patient names and patient_ids and names in this table.

The Third table I created still to contain patient information is the pharmacy table consisting of pharmacy_id, medicine_name and price without repeating patient names and ids.

I then created a join table containing all the primary keys of the other four tables connecting and making it easy to join the tables and obtain results using queries. The tables treatment (Join table) has attributes patient_id,(primary key of table patient), pharmacy_id (primary key of table pharmacy), doctor_id(primary key of table doctor) and admission_discharge_id(primary key of table admission_discharge).

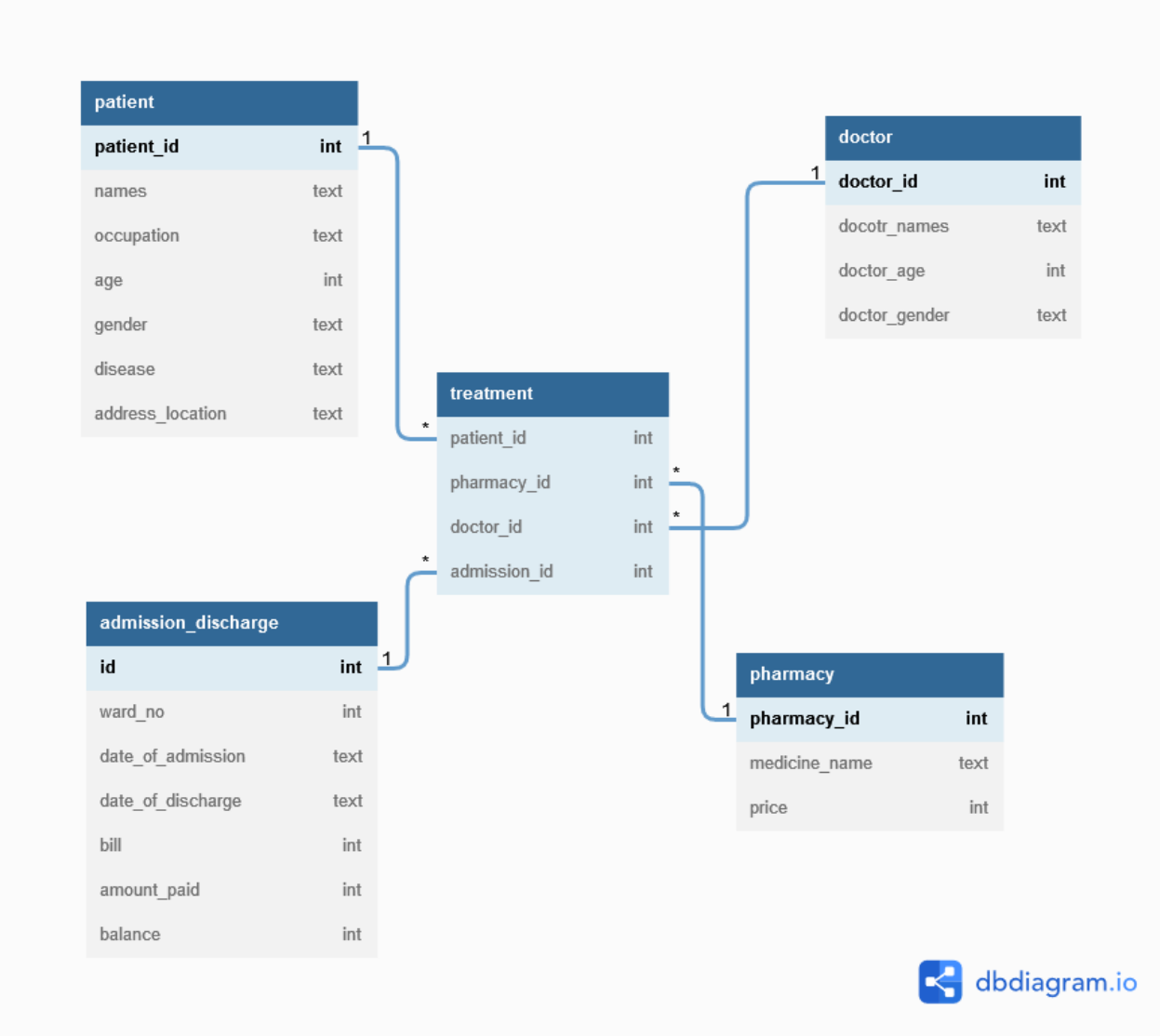
Anomalies prevented by Normalization:

Update Anomaly: is a data inconsistency that results from redundancy and a partial update. For example in my project, I have 7 columns in table patient out of which 2 are patient names and address_location. Now if one patient changes location, then I would have to update the table. But the problem would come in if the table is not normalized is that the patient will have multiple entries and while updating the entries some might get missed.

Insertion Anomaly: is the inability to add data to the database due to the absence of other data. For example, in my project, if a patient gets registered onto the database and does not get admitted then, the patient will have filled in all the attributes of the table patient but then will have NULL VALUES on the table admission_discharge and table pharmacy. This results in database inconsistencies due to omission.

Deletion Anomaly: This anomaly indicates unnecessary deletion of important information from the table or it is the unintended loss of data due to deletion of other data. For example, if a patient leaves the hospital, then the entry related to that patient will be deleted. However, that deletion will also delete ward_no of the hospital even if ward_no depends upon the hospital.

SCHEMA OF MY HOSPITAL DATABASE PROJECT



Qn. II

Define the different transaction anomalies giving examples from your project and how they can occur.

Transaction is a collection or a set of operations used to perform a logical set of work or that performs a single logical unit. A transaction means that the data in the database has changed.

A transaction anomaly refers to data change points that shouldn't normally occur in a system that generated data and they include; a dirty read, unrepeated read and Lost update(overwriting uncommitted Data) anomalies.

1.Dirty read:

Dirty read is a read of dirty data written by another transaction that has not yet committed.

Dirty data means uncommitted data written by another transaction.

It occurs or happens because of a Write Read (RW) conflict which results from reading uncommitted data from another transaction. For example: $w_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(A)$

has a sequence of actions in two transactions. Transaction 1 writes a value A, transaction 2 reads the value A, transaction 2 writes to A, and transaction 1 reads value from A.

Write by transaction 1 is read by transaction 2 and write by write by transaction 2 is read by transaction 1 and in so doing A is changed.

Risk of a dirt read: The transactions that wrote it might abort at a later time because the effect of it on the transaction persists through the uncommitted data that is read.

2. Unrepeated read:

It occurs or happens because of a Read Write (RW) conflict. For example:

$r_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(A)$;

Transaction 1 reads A, transaction 2 reads A and writes A which might be a different value after an increment or doing something with the value from A then transaction 1 reads again seeing or getting a different value from the previous read.

Therefore A is changed.

3. Lost update anomaly(Overwriting uncommitted Data)

Such a write is also called a blind write. It occurs because of a Write Write(WW) conflict. For example:

$w_1(A)$; $w_2(A)$; $w_2(B)$; $w_1(B)$;

Transaction I writes A, then transaction 2 writes A and writes B, then transaction 1 overwrites the value of B written by transaction 2.

Based on the order of commits that might happen, the database will end up with different outputs which might not happen in case of any of the transactions happening in serial.

Qn. III

Suggest ways in which the security of your database can be enhanced.

Creating regular back-ups of the database: This mitigates the risk of losing sensitive information due to malicious attacks or data corruption.

Deploying physical database security to prevent physical attacks by outsiders or even insider threats to prevent access to physical database server who can steal data, corrupt it or even insert harmful malware to gain remote access.

Avoid free hosting services because of possible lack of security and when choosing web hosting services make sure it is a company with a known track of taking security matters seriously.

Separating database servers and using real-time security information and event monitoring(SIEM) which is dedicated to database security and allows database owners to immediately take actions in the event of an attempted breach.

Avoid using default network ports (TCP and UDP) to transmit data between servers, the cyber attackers targeting your server try different port number variations with trial and error which discourages them from prolonging their attack attempts due to the additional work that is needed.

Setting up an HTTPS proxy sever: A proxy server evaluates requests sent from a workstation before accessing the database server. This server acts as a gatekeeper that aims to keep out non-authorized requests. Dealing with sensitive information such as passwords, payment information or personal information, set up an HTTPS server so that data traveling through the proxy server is also encrypted, giving you an additional security layer.

Using real-time database monitoring: Actively scanning your database for breach attempts bolsters your security and allows you to react to potential attacks. You can use monitoring software such as Tripwire's real-time File Integrity Monitoring (FIM) to log all actions taken on the database's server and alert you of any breaches. Furthermore, set up escalation protocols in case of potential attacks to keep your sensitive data even safer.

Regularly auditing the database security and organizing cybersecurity penetration tests. These allow you to discover potential security loopholes and patch them before a potential breach.

Deploying data encryption protocols: Encrypting your data which is important when moving or storing sensitive user information. Setting up data encryption protocols lowers the risk of a successful data breach. This means that even if cybercriminals get a hold of your data, that information remains safe.