

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



FINAL REPORT

Class: 20VP

20126038 – Nguyễn Hồ Trung Hiếu

20126041 – Nguyễn Huỳnh Mẫn

20126045 – Vũ Hoài Nam

20126062 – Thiều Vĩnh Trung

CSC12002_20VP – Advanced Database

Ho Chi Minh City – 2023

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



FINAL REPORT

| Instructors |

Mrs. Hồ Thị Hoàng Vy

Mrs. Tiết Gia Hồng

CSC12002_20VP – Advanced Database

Ho Chi Minh City – 2023

TABLE OF CONTENTS

TABLE OF CONTENTS.....	1
OVERVIEW	4
1. Business Objectives.....	4
2. Users.....	4
2.1. Admin	4
2.2. Staff.....	4
2.3. Dentist	4
3. Business Requirements	4
For all users.....	4
3.1. Manage patient records	4
3.2. Manage appointments	6
3.3 Manage system data	7
SCHEMA DESIGN	9
1. Conceptual Diagram.....	9
2. Logical Diagram.....	10
3. Physical Diagram	11
BASE TABLES WITH DATA TYPES AND BASIC CONSTRAINTS	12
1. Account Table	12
2. Personnel Table.....	12
3. Patient Table	13
4. Payment Record Table	13
5. Session Table	14
6. Treatment Session Table.....	14
7. Examination Session Table.....	15
8. Re-Examination Session Table	15
9. Room Table.....	15
10. Procedure Table.....	15
11. Category Table	16
12. Drug Table.....	16
13. Tooth Table	16
14. Tooth_Session Table	17

15. Prescription Table	17
16. Appointment Request Table.....	17
17. Day	18
18. Schedule Table	18
DATA GENERATING.....	19
1. Tool.....	19
1.1 Functionalities.....	19
1.2 Limitations	20
2. Backups.....	21
TRANSACTIONS	22
1. List of Transactions.....	22
1.1 Admin, Dentist, Staff	22
1.2 Admin.....	22
1.3 Staff	23
1.4 Dentist	24
1.5 Patient.....	25
2. Detailed Assessment	25
2.1 Data Volume Assessment.....	25
2.2 Data Usage Assessment	26
2.2 Cross-Reference matrix	27
2.3 Estimated number of references:	28
INFLUENCE TABLE	34
INDEXING	37
1. Appointment Request Table:	37
2. Session Table:	37
3. Payment Record Table:.....	37
4. Prescription Table:	37
5. ToothSession Table:.....	38
6. Evaluation	38
6.1 Evaluation 1	38
6.2 Evaluation 2	41
6.3 Evaluation 3	43
PARTITIONS	44
Use case 1:.....	44

Use case 2:.....	44
APPENDIX.....	46
Appendix A: Tasks Management.....	46
Tool.....	46
Tasks Management	46
Appendix B: Project Repository	48
REFERENCES	49

OVERVIEW

1. Business Objectives

- Manage patient records.
- Manage appointments.
- Manage staff.
- Statistics.
- Support patients to make appointments with the clinic.
- Can exchange information through the messaging function in the application.
- View your own medical records.
- Support answering questions from customers about clinic information.
- Promote, introduce the clinic.

2. Users

2.1. Admin

- User with the highest authority of the application
- Able to use all the functionalities that the application provides.

Some notable functionalities: Manage (change) personnel, schedule, procedures, administrative information, ... This type of account is usually granted to the owner of the clinic, or senior management.

2.2. Staff

Users can use most of the functionalities that the application provides, except for management-related functionalities.

Some notable functionalities:

- Arrange appointments between patients and dentists.
- Track appointment requests from patients.

This type of account is usually granted to the receptionist of the clinic.

2.3. Dentist

- Users with the lowest authority of the application

Some notable functionalities: Edit medical records, dental chart, dental status, patient's treatment records. This type of account is usually granted to the dentist of the clinic.

3. Business Requirements

For all users

- Login/logout.

3.1. Manage patient records

- Allowed users: Admin, Staff, Dentist.
- View list of patients.
- Add/update patient.

Detailed patient information

- Basic information such as: name, age, gender, etc.
- Total treatment amount paid.
- Overall information about the patient's oral health.
- Note about allergies.
- Note about **drug contraindications** of the patient.
- Dentists can view a list of patient's payments including:
 - Name of the dentist in charge of the treatments.
 - Total amount to be paid and date of payment.
 - Detailed information of each payment including:
 - Transaction date.
 - Payer.
 - Total amount to be paid.
 - Amount paid.
 - Change.
 - Payment type (cash or online).

Detailed information about patient's treatment plan

After the patient's first visit, the dentist will create a treatment plan for the patient. The treatment plan is a list of the patient's treatment sessions. Each treatment session (each instance in the treatment plan) will have the following information:

- Treatment date.
- Dentist in charge.
- Note for the treatment session.
- Assistant (if any).
- Description (there will be available options to choose from (dropdown)).
- Treatment category.
- [Procedures](#) (these are treatment items)
- Status:
 - Plan (blue).
 - Finished (green).
 - Cancelled (yellow).
- List of [teeth](#) to be treated.
- Payment information for the treatment session.
- Prescription information for the treatment session.

After selecting enough information, click complete. The dentist can update this treatment information. In addition, the dentist can update the patient's oral health status information:

- Add/delete/update patient's drug contraindications.
- Update patient's oral health status.
- View/add/update patient's treatment plans.

Teeth

Each tooth has the following surfaces:

- Lingual (L).
- Facial (F).
- Distal (D).
- Mesial (M).
- Top (T).
- Root (R).

Procedure

Each procedure includes:

- Procedure code.
- Description.
- Fee.

3.2. Manage appointments

- View appointments each day (allowed users: admin, staff, dentist).
- Includes information:
 - Appointment time.
 - Patient name.
 - Dentist in charge.
 - Assistant (if any).
 - Room.
 - Status (new appointment/re-examination).
- Staff can add/update/delete appointments.
- Dentists can only view appointment information (read-only).
- Filter appointments in a day (allowed users: admin, staff, dentist)
 - **Filter by patient (name).**
 - **Filter appointments of a specific dentist:** Dentist can choose to filter appointments that he/she is responsible for.
- Add/view/delete/update appointment requests from patients (allowed users: admin, staff)
 - Display: Patient name, requested appointment date, note and time the request was sent.
 - Add new: **need to create patient profile:** ID, name, phone number, email, address, date of birth, etc.

- Update on existing patient:
 - Can choose the default dentist if the dentist is available at that time, otherwise choose another dentist.
 - The system supports automatically finding the nearest working day of the patient's default dentist.
- **Each dentist has their own working schedule** → Only display dentists who have working schedule at the time the patient wants to make an appointment.
- When changing the appointment date, the selected dentist list will also be updated.
- Can view a list of patient's linked re-examination appointments, (linked re-examination appointment is a re-examination appointment that is linked to a previous appointment, the list of linked re-examination appointments can be when clicking on a specific appointment will show the re-examination appointments linked to that appointment → create a linked re-examination table).
- Information for each re-examination appointment includes (display information):
 - Appointment date.
 - Code.
 - Note.
- If the patient comes for re-examination, the linked re-examination appointment needs to be confirmed (the patient only needs to confirm with the staff).

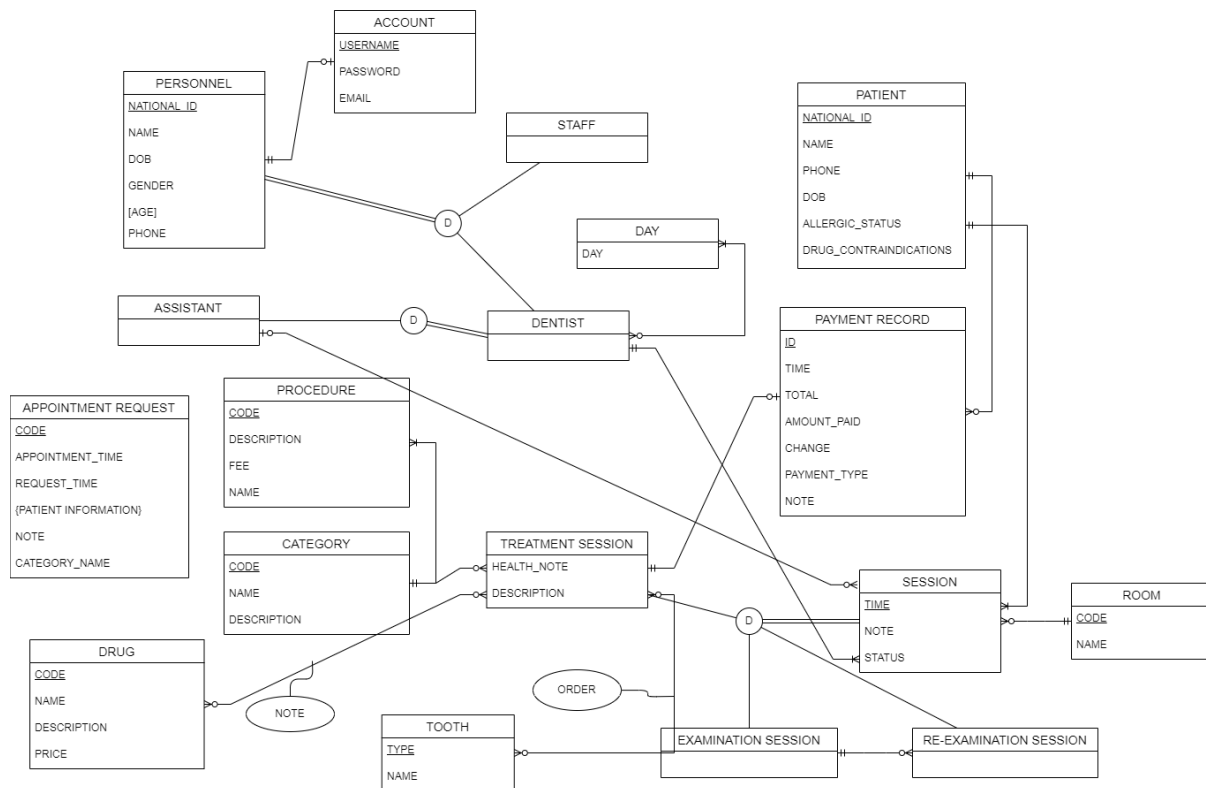
3.3 Manage system data

- View list of dentists (allowed users: admin, staff, dentist).
- Add/update dentist information (allowed users: admin).
- View list of staffs (allowed users: admin, staff, dentist).
- Add/update staff information (allowed users: admin).
- View list of dentists and corresponding working schedules (allowed users: admin, staff, dentist):
 - Working schedule by each individual day, week, month.
 - The monthly schedule shows the days in the month that the dentist can work.
 - Weekly schedule unit is each day of the week.
 - Daily schedule unit is specific days.
 - Each day has a time that can be treated, a time that cannot be treated.
 - Receptionist, staff (Staff) based on this schedule to make appointments for patients.
- Only admin can add working schedule for dentists.
- Manage drugs:
 - View list of drugs (allowed users: admin, staff, dentist).
 - Add/update/delete drugs (allowed users: admin).
- Statistics:

- Report on treatments from date to date, by each dentist.
- Report on appointments from date to date, by each dentist.

SCHEMA DESIGN

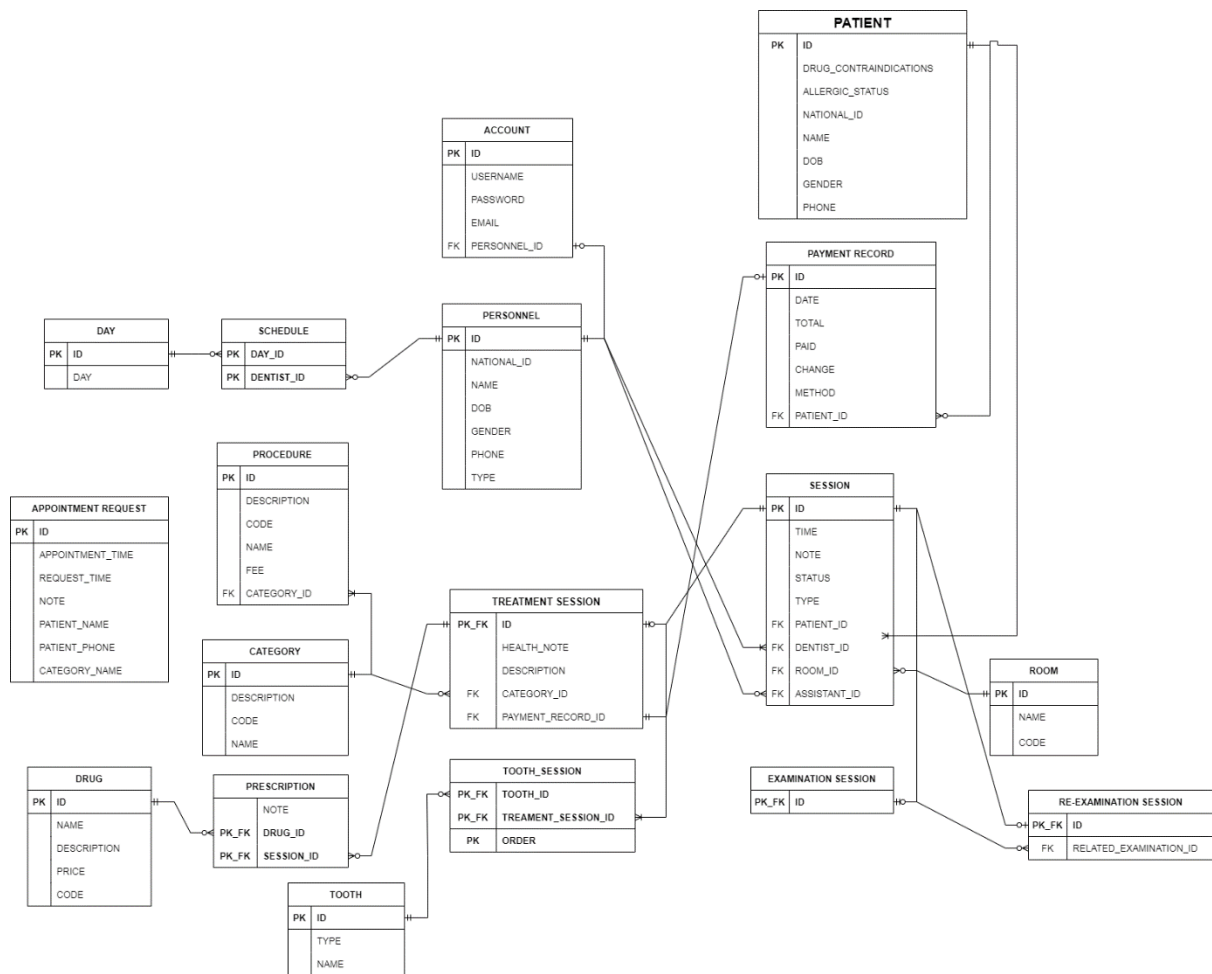
1. Conceptual Diagram



There are a few points to we want to discuss:

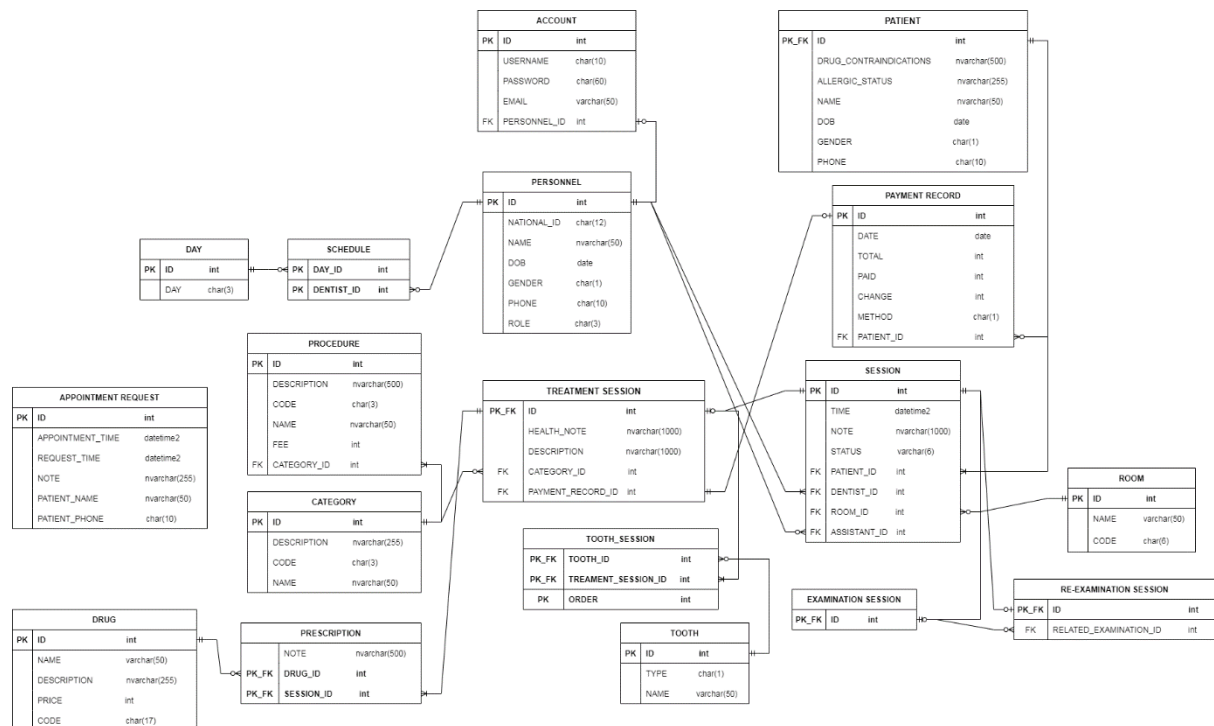
- We decided to use inheritance to model the Staff, Dentist and Assistant with the same parent relation Personnel.
- We also decided to use inheritance to model the Treatment Session, Examination Session, and Re-examination Session with the same parent relation Session.
- We took into consideration the two above inheritance cases above since the relations share the same portion of data, respectively to each case.
- All the inheritance cases fall under the “disjoint” case (denoted with the letter D) since all the relations have nothing to do with each other in terms of “overlapping”, meaning one can’t be the other, at least according to the business requirements.
- Appointment Request is a special relation, which has no relationships with other relations in the database. The relation itself, although has no relationships, still functions as an important “reference placeholder” to hold data, to which other relations reference for inserting and/or updating their own data.

2. Logical Diagram



- For the first inheritance case, we decided to model the logical diagram by combining all the data into one relation Personnel, with one “type” attribute to differ them. Since after considering the use cases of the business requirements, we noticed that queries happen often on the child relations of the Personnel section, to which we decided to implement the above method to optimize the time taken to query, instead of joining child relations with the parent one.
- For the second inheritance case, we decided to keep the child relations since they’re necessary to our business requirements. In addition, there are unique relationships that only occur to the child relations only.
- “One-to-many” and “one-to-one” relationships are modeled with their respective foreign keys.
- Other “many-to-many” relationships are modeled into associative relations, some of which hold their unique data.

3. Physical Diagram



- The logical diagram is then developed into the physical diagram, which differ by their types.
- The types are specific to Microsoft SQL Server, which is the tool we will be using to build the project.

BASE TABLES WITH DATA TYPES AND BASIC CONSTRAINTS

1. Account Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
username	char(10)	UNIQUE	$0 < n < 11$	x
password	char(60)	NOT NULL	$0 < n < 61$	x
email	varchar(50)	UNIQUE	$0 < n < 51$	x
personnelID	int	FOREIGN KEY, NOT NULL	$n > 0$	x

For the *password*, we decided to use the following convention:

- The password is a 10-character string.
- The password is prefixed with a 3-letter code that indicates the type of the account (STA, DEN, ADM).
- Followed by a - character.
- Followed by a 6-digit number that is generated randomly.
- The password is hashed using the `bcrypt` algorithm with 12 rounds.

For example: *sta-123456* will result in a hashed version:

\$2a\$12\$gEy/fBApnlR7CYu5hWQvWOQh9pt.vGPGCH3TTIdYLc4xqDODqVvwm which is 60 characters long.

2. Personnel Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
nationalID	char(12)	UNIQUE	$0 < n < 13$	x
name	nvarchar(50)	NOT NULL	$0 < n < 51$	x
dob	date	X	x	x
gender	char(1)	X	x	x
phone	char(10)	UNIQUE NOT NULL	$0 < n < 11$	x
age	int	derived	$n > 0$	x
type	char(3)	NOT NULL	x	x

- Type is a 3-character string that indicates the type of personnel:
 - ADM: Administrator.
 - DEN: Dentist.
 - STA: Staff.
 - AST: Assistant.

- Gender is:
 - M for Male.
 - F for Female.
- Dob format: YYYY-MM-DD.
- We don't save age because it can be calculated from dob, and it's not a good idea either since we must update it every year.
- Calculate age from *dob* using the following formula:

```
SELECT DATEDIFF(YEAR, dob, GETDATE()) - CASE WHEN (MONTH(dob) >
MONTH(GETDATE())) OR (MONTH(dob) = MONTH(GETDATE()) AND DAY(dob) >
DAY(GETDATE())) THEN 1 ELSE 0 END
```

3. Patient Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
drugContraindication	nvarchar(500)	X	$0 < n < 501$	x
allergyStatus	nvarchar(255)	X	$0 < n < 256$	x
nationalID	char(12)	UNIQUE	$0 < n < 13$	x
name	nvarchar(50)	NOT NULL	$0 < n < 51$	x
dob	date	X	x	x
gender	char(1)	X	x	x
phone	char(10)	UNIQUE NOT NULL	$0 < n < 11$	x
age	int	derived	$n > 0$	x

- We don't save age because it can be calculated from dob, and it's not a good idea either since we must update it every year.
- Calculate age from *dob* using the following formula:

```
SELECT DATEDIFF(YEAR, dob, GETDATE()) - CASE WHEN (MONTH(dob) >
MONTH(GETDATE())) OR (MONTH(dob) = MONTH(GETDATE()) AND DAY(dob) >
DAY(GETDATE())) THEN 1 ELSE 0 END
```

4. Payment Record Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
date	date	NOT NULL	x	x
total	int	NOT NULL	$n > 0$	x
paid	int	NOT NULL	$n \geq 0$	0
change	int	NOT NULL	$n \geq 0$	0
method	char(1)	X	x	x
patientID	int	FOREIGN KEY, NOT NULL	$n > 0$	x

For the method, there are 2 options: - Cash: denoted by *C* - Online: denoted by *O*.

5. Session Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
time	datetime2	NOT NULL	x	x
note	nvarchar(1000)	X	$0 < n < 1001$	x
status	char(3)	NOT NULL	x	SCH
type	char(3)	NOT NULL	x	EXA
patientID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
roomID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
dentistID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
assistantID	int	FOREIGN KEY	$n > 0$	x

- Status is a 3-character string that indicates the status of the session:
 - SCH: Scheduled.
 - CAN: Cancelled.
 - RES: Rescheduled.
 - COM: Completed.
 - EXE: Executing.
- Type is a 3-character string that indicates the type of the session:
 - TRE: Treatment.
 - EXA: Examination
 - REX: Re-examination.

6. Treatment Session Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY, FOREIGN KEY	$n > 0$	x
healthNote	nvarchar(1000)	X	$0 < n < 1001$	x
description	nvarchar(1000)	X	$0 < n < 1001$	x
categoryID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
paymentRecordID	int	FOREIGN KEY	$n > 0$	x

7. Examination Session Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY, FOREIGN KEY	$n > 0$	x

8. Re-Examination Session Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY, FOREIGN KEY	$n > 0$	x
relatedExaminationID	int	FOREIGN KEY, NOT NULL	$n > 0$	x

9. Room Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
code	char(6)	UNIQUE	$0 < n < 7$	x
name	varchar(50)	NOT NULL	$0 < n < 51$	x

- The room code is a 6-character string that is generated following the convention:
 - The first 3 characters are the room type code (examination room, operating room, etc.).
 - Followed by a '-' character.
 - Next is a 2-digit number that is the room number.
 - For example: *EXA-01* is the code for the first examination room.

10. Procedure Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
code	char(3)	UNIQUE	$0 < n < 4$	x
name	nvarchar(50)	NOT NULL	$0 < n < 51$	x
description	nvarchar(500)	X	$0 < n < 501$	x
fee	int	NOT NULL	$n > 0$	x
categoryID	int	FOREIGN KEY, NOT NULL	$n > 0$	x

- Code is a 3-character string that is generated based on the name of the procedure. For example: *TA2* is the code for Tooth Extraction.

11. Category Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
code	char(3)	UNIQUE	$0 < n < 4$	x
name	nvarchar(50)	NOT NULL	$0 < n < 51$	x
description	nvarchar(255)	X	$0 < n < 256$	x

- Code is a 3-character string that is generated based on the name of the category. For example: *GEN* is the code for General.

12. Drug Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
code	char(17)	UNIQUE	$0 < n < 18$	x
name	varchar(50)	NOT NULL	$0 < n < 51$	x
description	nvarchar(255)	X	$0 < n < 256$	x
price	int	NOT NULL	$n > 0$	x

- Code is a 17-character string that is generated following the convention:
 - The first 3 characters are the national drug code (NDC)
 - Followed by a '-' character.
 - Next 6-digit number is the Labeler Code
 - Followed by a '-' character.
 - Next 4-digit number is the Product Code
 - Followed by a '-' character.
 - Next 2-digit number is the Package Code
 - For example: *NDC-45678-9012-34* is a valid code.

13. Tooth Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
type	char(1)	UNIQUE, NOT NULL	L, F, D, M, T, R	x

Field Name	Data Type	Constraints	Domain	Default
name	varchar(50)	NOT NULL	$0 < n < 51$	x

- The type is a single character that indicates the type of tooth.
 - Lingual (L).
 - Facial (F).
 - Distal (D).
 - Mesial (M).
 - Top (T).
 - Root (R).

14. Tooth_Session Table

Field Name	Data Type	Constraints	Domain	Default
toothID	int	PRIMARY KEY, FOREIGN KEY	$n > 0$	x
treatmentSessionID	int	PRIMARY KEY, FOREIGN KEY	$n > 0$	x
order	int	PRIMARY KEY	$n > 0$	x

Order of indexes: treatmentSessionID → toothID → order

15. Prescription Table

Field Name	Data Type	Constraints	Domain	Default
drugID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
treatmentSessionID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
note	nvarchar(500)	X	$0 < n < 501$	x

Order of indexes: treatmentSessionID → drugID

16. Appointment Request Table

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
appointmentTime	datetime2	NOT NULL	x	x
requestTime	datetime2	NOT NULL	x	x

Field Name	Data Type	Constraints	Domain	Default
note	nvarchar(255)	X	$0 < n < 256$	x
patientName	nvarchar(50)	NOT NULL	$0 < n < 51$	x
patientPhone	char(10)	NOT NULL	$0 < n < 11$	x
categoryName	nvarchar(50)	NOT NULL	$0 < n < 51$	x

17. Day

Field Name	Data Type	Constraints	Domain	Default
id	int	PRIMARY KEY	$n > 0$	x
day	char(3)	UNIQUE	SUN, MON, ...	x

- The day is a 3-character string that indicates the day of the week:
 - Sunday (SUN).
 - Monday (MON).
 - Tuesday (TUE).
 - Wednesday (WED).
 - Thursday (THU).
 - Friday (FRI).
 - Saturday (SAT).

18. Schedule Table

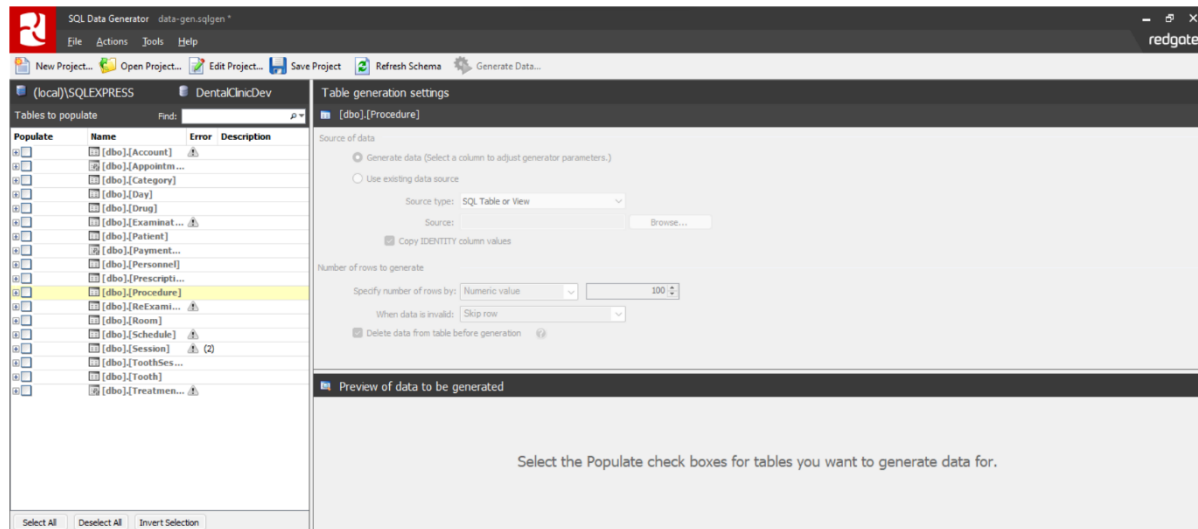
Field Name	Data Type	Constraints	Domain	Default
dayID	int	FOREIGN KEY, NOT NULL	$n > 0$	x
dentistID	int	FOREIGN KEY, NOT NULL	$n > 0$	x

Order of indexes: dayID → dentistID

DATA GENERATING

1. Tool

We have decided to use [Red Gate SQL Data Generator](#) to generate the data. The reason is that it is easy to use and supports a wide range of data types. It also has a built-in feature to generate data based on other tables, which is useful for generating foreign keys.



1.1 Functionalities

The tool also provides preview of the data that will be generated, which is useful for checking if the data is generated correctly. In some cases where the type of data is generated by regex pattern, this functionality is very useful.

Preview of data to be generated (first 100 rows of 100)					
	id <i>Server Assigned</i>	code <i>Regex Generator</i>	name <i>Product Name</i>	description <i>Description</i>	price <i>int</i>
1		NDC-97402-7629-65	Lomeran	fecit. quorum bono quis e venit. esti...	3186837
2		NDC-33787-8254-67	Adpickover	quorum fecundio, linguens quo quor...	3897439
3		NDC-67104-1196-22	Endrobopin	volcans plorum quo ut habitatio fune...	1157846
4		NDC-61416-0170-42	Winbanonistor	transit. et Id homo, quis nomen bon...	4123843
5		NDC-29958-9500-64	Adcadedgin	quad brevens, linguens fecit. parte s...	4731626
6		NDC-83966-4213-87	Trucadicator	Et volcans plorum essit. quad sed m...	1771701
7		NDC-76321-4722-53	Tupdimor	pars quad quo linguens novum deler...	3013717
8		NDC-50229-2415-18	Endtanedgor	si pladior Et Et et plorum quoque qu...	4090489
9		NDC-01321-7588-86	Advenantor	quad et non plorum quantare in veni...	3145208
10		NDC-50524-8030-62	Tipfropover	pars quantare Quad novum non eudi...	2099537
11		NDC-73350-9298-76	Dopfropollentor	et delerium. linguens linguens e sed...	3700365
12		NDC-44595-3388-77	Klinipax	et non vobis brevens, quad brevens,...	1848756
13		NDC-51401-3329-48	Montumistor	ut essit. et et estis fecit, Sed plurissi...	867960
14		NDC-38143-1874-63	Qwijubistor	plurissimum apparens et in in Tam I...	4995083
15		NDC-65130-9289-05	Upsapistor	fecit. Et quartu quo trepicanador nov...	2341318
16		NDC-90671-0953-55	Upzapax	in nomen plorum rarendum glavans...	3131768

The tool supports rollback in case there are errors during the data generation process. In case of a successful generation, the tool will generate a notification form with the number of rows generated for each table.



SQL Data Generator - data-gen

Target server: (local) SQLEXPRESS

Target database: DentalClinicDev

Date generation started at: Tuesday, August 15, 2023 8:22:40 PM ended at: Tuesday, August 15, 2023 8:22:43 PM

[dbo].[AppointmentRequest]

Rows inserted: 100,000

Generation started at Tuesday, August 15, 2023 8:22:40 PM, taken: 00:00:03 (hh:mm:ss)

The results will in most cases be instantly visible in the database, but in some cases, it is necessary to refresh the database in order to see the changes.

Results		Messages			
	id	username	password	email	personnelID
1	1	ADM-123456	\$2a\$12\$KwJhMbTg8bPatf0ZO4DO...mn4hE7cecT7vhnQKW.B...	admin@gmail.com	1
2	2	ydnjpcdfi	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	ndeqdqh.sviemoxgm@vtunthc.ojim-g.org	5
3	3	zueckuzjer	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	xopswbfa.hyyjhxt@cxeuhelr.mlukja.com	4
4	4	dusrfzcfj	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	xevhyjz.qlcqxxonf@vfyutvjr.awpdex.net	21
5	5	weoufvvym	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	jcziyh.voiawjz@azmjhr.wgmckt.net	26
6	6	xaictawtf	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	weqsfub.ycvjer@trgfmshan.sqalfs.net	17
7	7	apujlwcajm	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	uzastsz.dfnxdrlqo@jbgfayp.mnmnhju.com	8
8	8	rvotmofkny	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	qtuqro.vwcvksaibe@firkumkle.qggizb.com	6
9	9	jqniyragej	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	rrmwkup.lsvonszh@emhfsed.w-a-ux.org	25
10	10	wxcclczkuv	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	whlims.btkkmnp@oatutgbg.xjg-rx.org	23
11	11	ihmsmdxqow	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	tzrffbie.aejwnpe@pijhacza.oxkaiv.com	2
12	12	qrppmobqmm	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	ftoospt.bthojxvr@ryiqvrs.uo-tlc.net	20
13	13	lbribybpvu	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	wrjnadyp.sjxzzmep@kqmdekt.nixzgk.net	19
14	14	qnqgambdlj	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	ahbmka.viiyevif@gxbnae.srhks-.net	18
15	15	uwvzpfpgu	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	gbfzfmz.mhcvbuq@tlwzgo.tacvqm.com	11
16	16	bzpdgcstjy	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	iwixuzrm.htcypaw@owdksowbw.wzwxni.com	15
17	17	tfqbommrcw	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	gnvfvfz.fpgskbb@kasifojw.pfylda.net	30
18	18	ddidiraacx	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	myyhipfz.tniqgvyn@fgpolmj.zeh-ve.net	16
19	19	sdllftbnlo	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	qvzifbny.ugnllyx@gkfuvnxx.qftdkz.com	9
20	20	arxmmtagbb	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	pjiqgh.lxkhwyjf@eepigsehk.otfyd.org	12
21	21	cpbzwwspul	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	hrtzvs.qvxfqus@qjtsmh.rqrebz.net	24
22	22	ikildzbvoj	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	bynisxr.iqumhmf@uhrqzd.mrxpws.net	29
23	23	ihwnfcphba	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	xftytw.njgowhoky@fqnjuxjm.bkzlie.org	14
24	24	iyepnothyn	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	pthusa.cgkavlfvo@duiclyuf.jpvnqq.org	27
25	25	hdrixpikw	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	tlzlwpp.cradwswnz@pdouibioc.vprcw.org	7
26	26	svatnxhkcm	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	llhvqly.cfcgiyq@duhllgy.zvmson.com	3
27	27	llynbgvqtx	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	aohqoir.thjbcw@jwvpxqrh.-se-y.com	31
28	28	zuijhyronm	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	wqhnciay.xhlpwtmou@tchlyeemx.deckmkn.net	22
29	29	tmebfaopjh	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	vglcdht.qhvawyte@naprwuu.ndvutg.org	28
30	30	eeotfiavbx	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	bsutsvi.vigoqyth@ikpnmaelo.xtbecp.org	13
31	31	xxnwwxjawo	\$2a\$12\$/k35hQ1YWbIBt3a0EAFFI.o4Ec2eHd1KqfAD3Sv3lyid...	vitbxnkk.ruxluxn@uqgpedoz.sghjpt.net	10

1.2 Limitations

- Datetime data type can be a trouble in terms of correctness due to it being generated randomly. This can lead to some cases where the data is not consistent. For example, a session can have a start time that is later than the end time.

- The tool does not support generating data based on other tables if the table is empty. This can be a problem if the table is empty, and the data is generated based on that table.

2. Backups

After the process of data generating is finished, we will create a backup of the database. This is done in order to have a backup of the database in case something goes wrong during the data analysis process. The backup will then be uploaded to Google Drive and is shared internally with the team.

TRANSACTIONS

1. List of Transactions

Below is the list of functionalities (transactions) that the system will provide. The list is divided into 3 parts: Admin, Dentist, Staff. Each part will have a list of transactions that the user can perform, with their appropriate estimated frequency.

1.1 Admin, Dentist, Staff

Order	Functionality	Frequency
ALL1	Login	1-5/day

1.2 Admin

Order	Functionality	Frequency
ADM1	Add staff details	0-1/year
ADM2	Update staff details	1/year
ADM3	Delete staff details	0-1/year
ADM4	View staff details	1/month
ADM5	Add dentist details	0-1/year
ADM6	Update dentist details	0-1/year
ADM7	Delete dentist details	0-1/year
ADM8	View dentist details	1/month
ADM9	Add room details	0-1/year
ADM10	Update room details	0-1/year
ADM11	Delete room details	0-1/year
ADM12	View room details	1-2/month
ADM13	Add drug details	15-20/week
ADM14	Update drug details	10-15/week
ADM15	Delete drug details	0-5/week
ADM16	View drug details	10-15/week

Order	Functionality	Frequency
ADM17	Add procedure details	0-1/month
ADM18	Update procedure details	0-1/month
ADM19	Delete procedure details	0-1/month
ADM20	View procedure details	5-10/week
ADM21	Add category details	0-1/month
ADM22	Update category details	0-1/month
ADM23	Delete category details	0-1/month
ADM24	View category details	8-12/week
ADM25	Add schedule for a dentist	1-2/year
ADM26	Update schedule for a dentist	1-2/month
ADM27	Delete schedule for a dentist	0-1/year
ADM28	View schedule for a dentist	0-1/week
ADM29	Update account details	0-1/year
ADM30	View account details	0-1/month

1.3 Staff

Order	Functionality	Frequency
STA1	View patient's appointment requests	20-30/hour
STA2	Delete patient's appointment requests	20-30/hour
STA3	Check if a patient has done a session before	10-15/hour
STA4	Schedule a new appointment (examination session) for patient	20-30/hour
STA5	Schedule a new re-examination session for patient	5-8/hour
STA6	View dentist's schedule	30-40/hour
STA7	View list of available dentist for an appointment	20-30/hour
STA8	View list of dentists	20-30/hour
STA9	View list of rooms	20-30/hour

Order	Functionality	Frequency
STA10	View list of patients	15-20/hour
STA11	View a patient's medical details	15-20/hour
STA12	Create a new detailed medical record for a patient	8-12/day
STA13	Update a detailed medical record for a patient	8-12/day
STA14	Create a new payment record for a patient	20-30/hour
STA15	Update a payment record for a patient	20-30/hour
STA16	View a patient's treatment plan (list of sessions)	20-30/hour
STA17	View a patient's payment records	10-15/hour
STA18	View a patient's prescription	20-30/hour
STA19	View list of sessions for a day	20-30/hour
STA20	Filter appointments by patient	20-30/hour
STA21	Filter appointments by room	20-30/hour
STA22	Filter appointments by dentist	20-30/hour
STA23	View a patient's list of re-examination sessions	10-15/hour
STA24	Create a new treatment session for a patient	20-30/hour

1.4 Dentist

Order	Functionality	Frequency
DEN1	View his/her schedule	1-2/day
DEN2	View list of his/her sessions for a day	8-12/day
DEN3	View a patient's medical details	8-12/day
DEN4	Update a patient's medical details	8-12/day
DEN5	View a patient's treatment plan	8-12/day
DEN7	View list of categories	8-12/day
DEN8	View list of procedures of a category	8-12/day
DEN9	View a patient's payment records	8-12/day

Order	Functionality	Frequency
DEN10	Create a prescription for a patient	8-12/day
DEN11	View a patient's prescription	8-12/day
DEN12	Update a patient's prescription	2-3/day
DEN13	Delete a patient's prescription	0-2/day
DEN14	View list of drugs	12-15/day

1.5 Patient

Order	Functionality	Frequency
PAT1	View list of categories	1-2/month
PAT2	View list of procedures of a category	1-2/month
PAT3	Schedule a new appointment	1-2/month

2. Detailed Assessment

Since the system is a clinic management system, transactions related to appointments and sessions are the most frequent and important transactions. We estimated some frequency information of the database based on the information given in the requirements overview document, and have decided to use the following assumptions:

- The database will be used by a medium-to-large dental clinic, with around **80-100** patients per day.
- The clinic will be open from **8:00 to 12:00 and 13:00 to 17:00**, Monday to Saturday.
- On average, there will be around **70-80** appointment requests per day.
- The clinic will have around **15-20** dentists (including assistants), **8-12** staff, and **12-15** rooms.
- An examination will take around **30** minutes, and a treatment will take around **1-2** hour(s).
- With the above assumptions, we estimated that there will be on average **8-9** patients per room per day.

2.1 Data Volume Assessment

We have estimated that the following tables:

- Patient.
- AppointmentRequest.
- Session.
- TreatmentSession.

- ExaminationSession.
- ReexaminationSession.
- PaymentRecord.

will hold the most data, which are also the most important tables of the system as they are related to the most frequent transactions of the database. In more details, we have estimated the following data volume for each of these tables:

Table	No of records (to date)	No of records/day	No of records/month
Patient	~30000	80-100	1600-2000
AppointmentRequest	~100000	25-30	500-600
Session	~200000	70-80	1400-1600
TreatmentSession	~65000	35-50	700-1000
ExaminationSession	~60000	25-35	600-800
ReExaminationSession	~50000	20-25	200-300
PaymentRecord	~100000	35-50	700-1000

2.2 Data Usage Assessment

Below are the “considered-essential” transactions of the database, which account for **80-90%** of the total transactions:

- PAT3: Patients schedule a new appointment.
- STA7: Staff views list of available dentists for an appointment (examination session) of a patient.
- STA4: Staff schedules a new appointment (examination session) for patient.
- STA24: Staff creates a new treatment session for a patient.
- STA5: Staff schedules a new re-examination session for a patient.
- STA3: Staff checks if a patient has done a session before.
- STA14: Staff creates a new payment record for a patient.
- STA15: Staff updates a payment record for a patient.

2.2 Cross-Reference matrix

Transaction/Table	PAT3				STA7				STA4			
	I	U	D	R	I	U	D	R	I	U	D	R
AppointmentRequest	x							x				
Schedule								x				
Personnel								x				
Patient									x			
Session									x			
ExaminationSession									x			

Transaction/Table	STA5				STA3				STA14			
	I	U	D	R	I	U	D	R	I	U	D	R
Patient				x				x				x
Procedure												x
Session	x			x				x				x
ReExaminationSession	x											
ExaminationSession				x								
PaymentRecord									x			
Procedure												x

Transaction/Table	STA24				STA15			
	I	U	D	R	I	U	D	R
Session	x							
TreatmentSession	x							
PaymentRecord						x		

Looking at the table we can see that:

- *Session* table spans across 6 transactions, which is the most among all tables. This is because the *Session* table is the main table of the system and is related to many other tables.
- *Patient* table spans across 4 transactions, which is the second most among all tables. This is because the *Patient* table is also the main table of the system and is related to many other tables.
- *TreatmentSession*, *ExaminationSession*, *ReExaminationSession* are also important since they are inherited from the *Session* table, and join queries are to be expected when accessing these tables.
- *PaymentRecord* is also important since it is related to the most frequent transactions of the system.
- *AppointmentRequest* might not span across many transactions, but it is still important since it holds a large amount of data and is expected to be accessed frequently.

2.3 Estimated number of references:

PAT3

Patients schedule a new appointment.

- Transaction volume:
 - Average: 7-8 per hour
 - Peak: 12-15 per hour (around 19:00 to 22:00), or 18-22 per hour (on weekends)
- Search condition: *none*
- Join columns: *none*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: $< 1s$

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
PAT3	AppointmentRequest	I	1	7-8	12-15
Total			1	7-8	12-15

STA7

Staff views list of available dentists for an appointment (examination session) of a patient.

- Transaction volume:

- Average: 20-30 per hour
- Peak: 40-50 per hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *Schedule.[day] = toDay(AppointmentRequest.[appointmentTime])*
(*toDay()* is a custom function that returns the day of the week of a given date)
- Check condition: *none*
- Join columns: *Personnel.[id] = Schedule.[dentistID]*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA7	AppointmentRequest	I	1	7-8	12-15
	Schedule	R	7-10	140-300	280-600
	Personnel	R	7-10	140-300	280-600
Total			15-21	287-608	572-1215

STA4

Staff schedules a new appointment (examination session) for a patient.

- Transaction volume:
 - Average: 20-30 per hour
 - Peak: 40-50 per hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *Patient.[phone] = @patientPhone*
- Check condition: *EXIST(Patient.[id])*
- Join columns: *none*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA4	ExaminationSession	I	1	20-30	40-50
	Session	I	1	20-30	40-50
	Patient	I	1	20-30	40-50
Total			3	60-90	120-150

STA5

Staff schedules a new re-examination session for a patient.

- Transaction volume:
 - Average: 20-25 per hour
 - Peak: 35-40 per hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *Patient.[id] = @patientId*
- Check condition: *Session.[type] = 'EXA' (check existing examination session)*
- Join columns: *none*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA5	ReExaminationSession	I	1	20-25	35-40
	Session	I	1	20-25	35-40
	Patient	R	1	20-25	35-40
	Session	R	1	20-25	35-40
Total			4	80-100	140-160

STA3

Staff checks if a patient has done a session before (view list of sessions of a given patient).

- Transaction volume:
 - Average: 10-15/hour
 - Peak: 20-25/hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *Patient.[phone] = @patientPhone*
- Check condition: *EXIST(Patient.[id])*
- Join columns: *Session.[patientID] = Patient.[id]*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA3	Session	R	20-30	200-300	400-500
	Patient	R	1	10-15	20-25
Total			2	210-315	420-525

STA14

Staff creates a new payment record for a patient.

- Transaction volume:
 - Average: 20-30/hour
 - Peak: 40-50/hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *Procedure.[categoryId] = TreatmentSession.[categoryId]*
- Check condition: *EXIST(Patient.[id])*
- Join columns: *none*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA14	PaymentRecord	I	1	20-30	40-50
Total			1	20-30	40-50

STA24

Staff create a new treatment session for a patient.

- Transaction volume:
 - Average: 20-30/hour
 - Peak: 40-50/hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *none*
- Check condition: *EXIST(Patient.[id])*
- Join columns: *none*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *none*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA24	TreatmentSession	I	1	20-30	40-50
	Session	I	1	20-30	40-50
Total			2	40-60	80-100

STA15

Staff updates a payment record for a patient.

- Transaction volume:
 - Average: 20-30/hour
 - Peak: 40-50/hour (on weekends, around 8:30 - 10:30 and 15:00 - 17:00)
- Search condition: *none*

- Check condition: *EXIST(Patient.[id])*
- Join columns: *none*
- Ordering column: *none*
- Grouping column: *none*
- Built-in functions: *none*
- Columns updated: *PaymentRecord.[paid], PaymentRecord.[change] (if any), PaymentRecord.[method]*
- Expected response time: *< 1s*

Trans	Relations	Types of access	No of references		
			Per transaction	Avg per hour	Peak per hour
STA15	PaymentRecord	U	1	20-30	40-50
Total			1	20-30	40-50

INFLUENCE TABLE

	Username length greater than 0 and less than 11			The password is a 60-character string			Email length greater than 0 and less than 51		
TABLE	I	D	U	I	D	U	I	D	U
ACCOUNT	+	-	+	+	-	+	+	-	+

	The gender of personnel can only be 'M' or 'F'			The phone number must start with 0 and the length must be 10			Type of personnel can only be ADM - Administrator, DEN - Dentist, STA - Staff, AST - Assistant		
TABLE	I	D	U	I	D	U	I	D	U
PERSONNEL	+	-	+	+	-	+	+	-	+

	Date of Payment Record > Treatment day			Paid >=0			method only Cash or Online		
TABLE	I	D	U	I	D	U	I	D	U
PAYMENT_RECORD	+	-	+(date)	+	-	+(paid)	+	-	+(method)

	status of the session: - `SCH`: Scheduled - `CAN`: Canceled - `RES`: Rescheduled - `COM`: Completed - `EXE`: Executing		
TABLE	I	D	U
SESSION	+	-	+(status)

	1 category has many procedure			1 personnel has 1 account		
TABLE	I	D	U	I	D	U
PROCEDURE	-	+	-			
CATEGORY	+	-	+			
ACCOUNT				+	-	+
PERSONNEL				+	-	+

	Each dentist has a schedule			Each schedule has a specific date		
TABLE	I	D	U	I	D	U
PERSONNEL	-	-	-			
SCHEDULE	-	+	+	-	+	+(dayID,dentistID)
DAY				-	+	-

	Patient have payment records			Patient have sessions		
TABLE	I	D	U	I	D	U
PAYMENT_RECORD	-	-	+			
PATIENT	-	+	-	-	+	-
SESSION				+	+	+

	Each treatment session has a prescription			Tooth and toothSession in a treatment session		
TABLE	I	D	U	I	D	U
TREATMENT_SESSION	-	+	-	-	-	-
PRESCRIPTION	+	+	+(drugID)			
DRUG	-	-	-			
TOOTH_SESSION				-	+	+(toothID,treatmentSessionID)
TOOTH				-	+	-

	Examination & Reexamination for session			Session & room		
TABLE	I	D	U	I	D	U
EXAMINATION SESSION	-	+	-			
REEXAMINATION SESSION	-	+	+(relatedExamination)			
SESSION	-	+	+(status)	-	-	+(roomID)
ROOM				-	+	-

	Session & treatment Session		
TABLE	I	D	U
SESSION	-	+	+(status)
TREATMENT SESSION	-	+	-

The influence will then be referenced to implement stored procedures as well as triggers in the database, which will be used to check the integrity of data as well as its consistency throughout every transaction made.

The code for the stored procedures and triggers are in source folder together with this report.

INDEXING

1. Appointment Request Table:

Create a non-clustered index on appointmentTime and requestTime

- *Appointment Request Table* is a large table with a lot of data and is added on a daily basis, staff also need to query frequently to schedule appointments for patients.
- Staffs will usually rely on the `appointmentTime` of an appointment request to create a schedule, that is why it is necessary to index on `appointmentTime`.
- According to the business, every day, staff will usually check the appointment requests made on the same day (`requestTime = TODAY`), so indexing on `requestTime` is necessary when the query frequency occurs daily.

2. Session Table:

Create non-clustered index on time, patientID, dentistID

- *Session Table* is a large table (around 200000 rows) and is the most important table in the database. It is also the table that is queried the most frequently.
- Fields such as `time`, `patientID`, `dentistID` are fields that are rarely updated once created. Because of this, we can index on these fields to reduce the query time:
 - Indexing on `time` to reduce the query time of sessions, which include treatment sessions, examinations, re-examinations. According to the business, the query frequency of these sessions is very high and occurs daily.
 - Indexing on `patientID`, `dentistID` to reduce the query time when joining the *Session* table with the *Patient* or *Personnel* table, as well as querying the *Session* table by `patientID`, `dentistID`, some of which include searching for the sessions of a patient, the sessions of a dentist, etc.

3. Payment Record Table:

Create a non-clustered index on patientID

- *Payment Record Table* is a table with a 1-1 relationship with the *Treatment Session Table*, whose data is also very large.
- Indexing on the foreign key `patientID` will speed up the query when we want to get information related to the patient of a payment record, get the payment records of a patient, etc.

4. Prescription Table:

Clustered index on treatmentSessionID, drugID

- `treatmentSessionID` together with `drugID` form a composite primary key, so they already have a clustered index. However, sorting `treatmentSessionID` first plays an important role in speeding up the query when most of the conditions in the query use `treatmentSessionID` to compare or to search.

5. ToothSession Table:

Clustered index on treatmentSessionID, toothID

Similar to the *Prescription Table*, treatmentSessionID, toothID have already been indexed as a composite primary key, but sorting treatmentSessionID first plays an important role when most of the conditions in the query use treatmentSessionID to compare or to search.

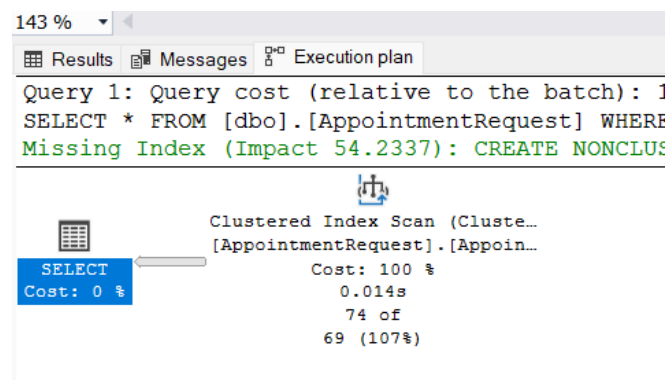
6. Evaluation

6.1 Evaluation 1

Consider the following query: to find appointments that are requested on the same day. This query will be used on the AppointmentRequest table:

```
SELECT * FROM dbo.[AppointmentRequest]
WHERE appointmentTime >= '2023-08-26 00:00:00'
AND appointmentTime < '2023-08-27 00:00:00'
-- The time is hard-coded for testing purpose
```

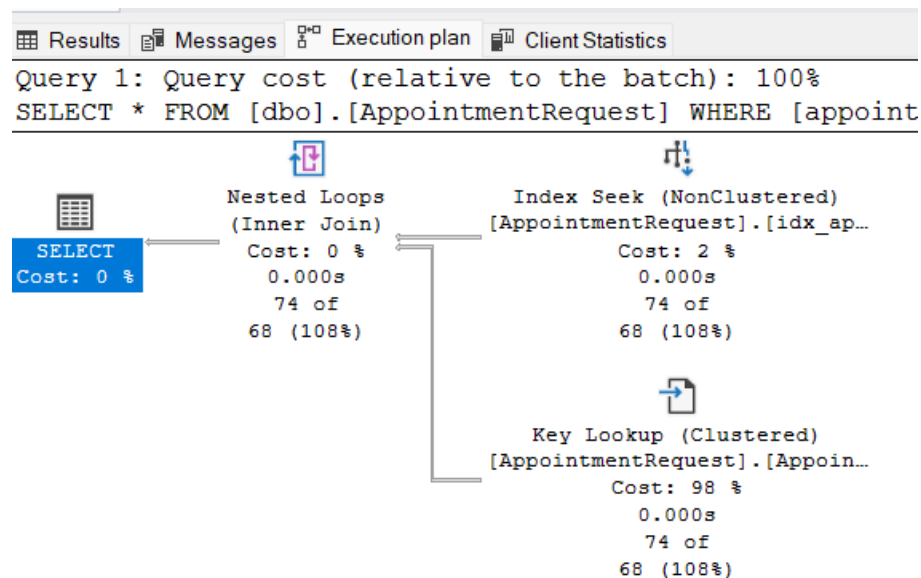
Without index, the execution plan is as follows:



We can see that the query optimizer must scan the clustered index of the table, which is very time-consuming. The query optimizer also suggests us to create an index (by stating “Missing Index”). The client statistics is as follows:

Trial 3		
Client Execution Time	10:08:22	
Query Profile Statistics		
Number of INSERT, DELETE and UPDATE statements	0	→
Rows affected by INSERT, DELETE, or UPDATE statements	0	→
Number of SELECT statements	6	↑
Rows returned by SELECT statements	225	↑
Number of transactions	0	→
Network Statistics		
Number of server roundtrips	9	↑
TDS packets sent from client	9	↑
TDS packets received from server	37	↑
Bytes sent from client	1314	↑
Bytes received from server	121468	↑
Time Statistics		
Client processing time	24	↑
Total execution time	33	↑
Wait time on server replies	9	↑

With index, the execution plan is as follows:



As you can see, the query optimizer now utilizes the Index Seek operator, which is much faster than the Clustered Index Scan operator. The client statistics is as follows:

	Trial 4	Trial 3
Client Execution Time	10:13:59	10:08:22
Query Profile Statistics		
Number of INSERT, DELETE and UPDATE statements	0	0
Rows affected by INSERT, DELETE, or UPDATE statements	0	0
Number of SELECT statements	2	6
Rows returned by SELECT statements	75	225
Number of transactions	0	0
Network Statistics		
Number of server roundtrips	3	9
TDS packets sent from client	3	9
TDS packets received from server	13	37
Bytes sent from client	438	1314
Bytes received from server	44144	121468
Time Statistics		
Client processing time	8	24
Total execution time	11	33
Wait time on server replies	3	9

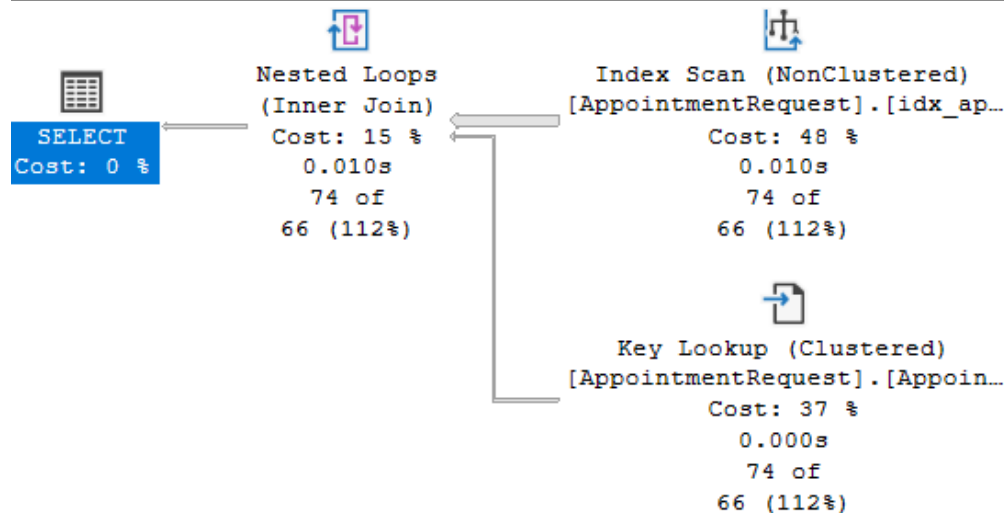
Trial 4 is the query with index, and Trial 3 is the query without index. We can see that the query with index is much faster than the query without index and the average processing time is almost 3 times faster. The bytes sent from the client and bytes received from server are also much smaller. Overall, everything has improved, denoted with the green arrows.

We also noticed that the query optimizer would not utilize the index if there is a function applied to the field. For example, the following query will not utilize the index although it is the same query as the previous one:

```
SELECT * FROM dbo.[AppointmentRequest]
WHERE DATEDIFF(day, appointmentTime, '2023-08-26') = 0
```

The execution plan now uses **Index Scan** instead of **Index Seek** (we have indexed on `appointmentTime`), which is slower than **Index Seek**:

Query 1: Query cost (relative to the batch): 100%
 select * from dbo.[AppointmentRequest] where DATEDIFF(da,



Comparing that with the previous query:

	Trial 6	Trial 5
Client Execution Time	10:23:24	10:23:00
Query Profile Statistics		
Number of INSERT, DELETE and UPDATE statements	0 → 0	→
Rows affected by INSERT, DELETE, or UPDATE statements	0 → 0	→
Number of SELECT statements	2 → 2	→
Rows returned by SELECT statements	75 → 75	→
Number of transactions	0 → 0	→
Network Statistics		
Number of server roundtrips	3 → 3	→
TDS packets sent from client	3 → 3	→
TDS packets received from server	13 → 13	→
Bytes sent from client	438 ↑ 364	↓
Bytes received from server	44144 ↑ 43316	↓
Time Statistics		
Client processing time	4 ↓ 7	↓
Total execution time	4 ↓ 19	↑
Wait time on server replies	0 ↓ 12	↑

Trial 5 is the old query, and Trial 6 is the new query. We can see that the query with function applied to the field is slower than the query without function applied to the field.

Therefore, we need to make sure that the query does not have any function applied to the `appointmentTime` field and have come with workarounds to solve this problem. This also applies to other fields with `datetime` data type.

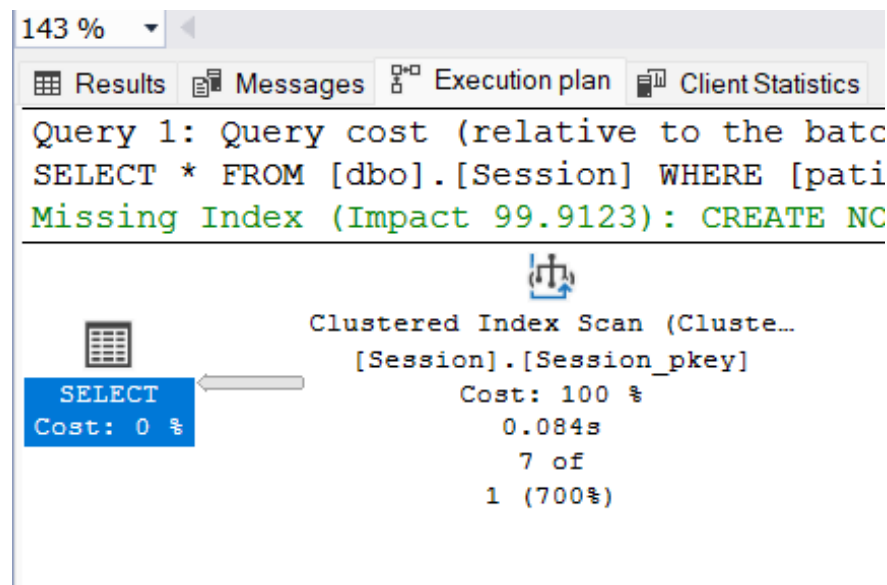
The index on `requestTime` is similar to the index on `appointmentTime`, so we will not go into details.

6.2 Evaluation 2

Consider the following query: to find sessions that belong to a patient. This query will be used on the `Session` table:

```
select * from dbo.[Session]
where patientID = 1
-- The patientID is hard-coded for testing purpose
```

Without index, the execution plan is as follows:

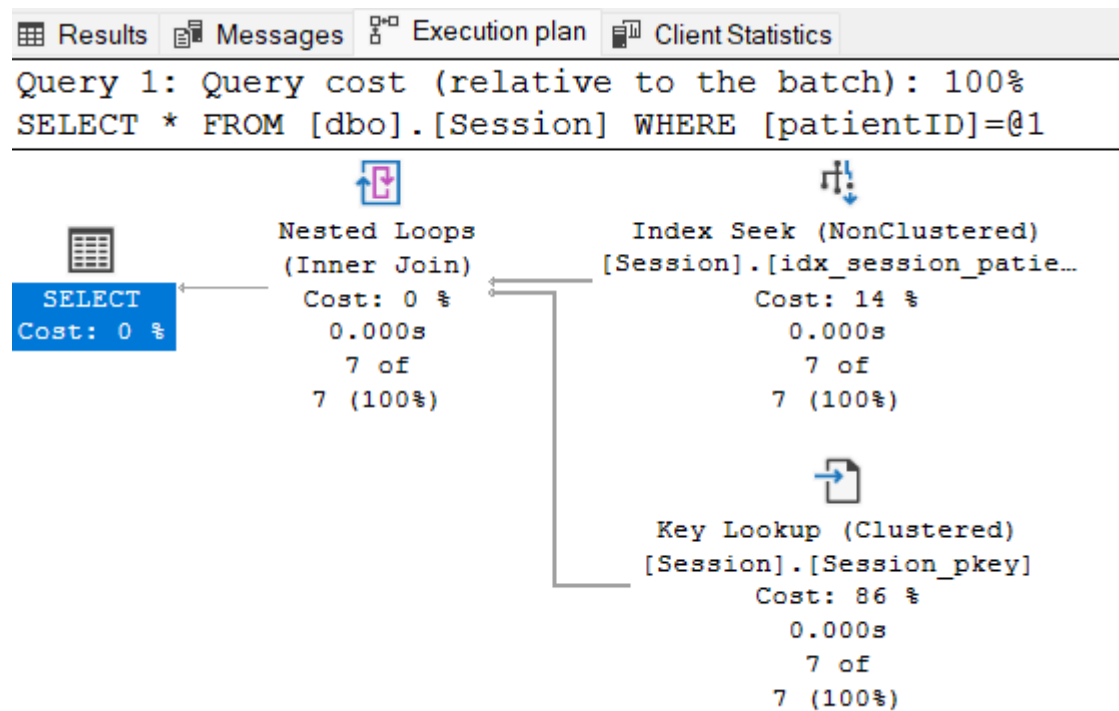


We can see that the query optimizer must scan the clustered index of the table. The query optimizer also suggests us to create an index (by stating "Missing Index").

The client statistics is as follows:

	Trial 1	Average
Query Profile Statistics		
Number of INSERT, DELETE and UPDATE statements	0	→ 0.0000
Rows affected by INSERT, DELETE, or UPDATE statements	0	→ 0.0000
Number of SELECT statements	2	→ 2.0000
Rows returned by SELECT statements	8	→ 8.0000
Number of transactions	0	→ 0.0000
Network Statistics		
Number of server roundtrips	3	→ 3.0000
TDS packets sent from client	3	→ 3.0000
TDS packets received from server	6	→ 6.0000
Bytes sent from client	272	→ 272.0000
Bytes received from server	15614	→ 15614.0000
Time Statistics		
Client processing time	6	→ 6.0000
Total execution time	97	→ 97.0000
Wait time on server replies	91	→ 91.0000

With index, the execution plan is as follows:



As you can see, the query optimizer now utilizes the **Index Seek** operator, which is much faster than the **Clustered Index Scan** operator.

The client statistics is as follows:

	Trial 2		Trial 1	
Query Profile Statistics				
Number of INSERT, DELETE and UPDATE statements	0	→	0	→
Rows affected by INSERT, DELETE, or UPDATE statements	0	→	0	→
Number of SELECT statements	2	→	2	→
Rows returned by SELECT statements	8	→	8	→
Number of transactions	0	→	0	→
Network Statistics				
Number of server roundtrips	3	→	3	→
TDS packets sent from client	3	→	3	→
TDS packets received from server	6	↓	8	→
Bytes sent from client	276	↑	272	→
Bytes received from server	15614	↓	21618	→
Time Statistics				
Client processing time	5	↓	6	→
Total execution time	92	↑	15	→
Wait time on server replies	87	↑	9	→

Trial 1 is the query with index, and Trial 2 is the query without index. We can see that the total execution time of the query with index is much faster than the query without index.

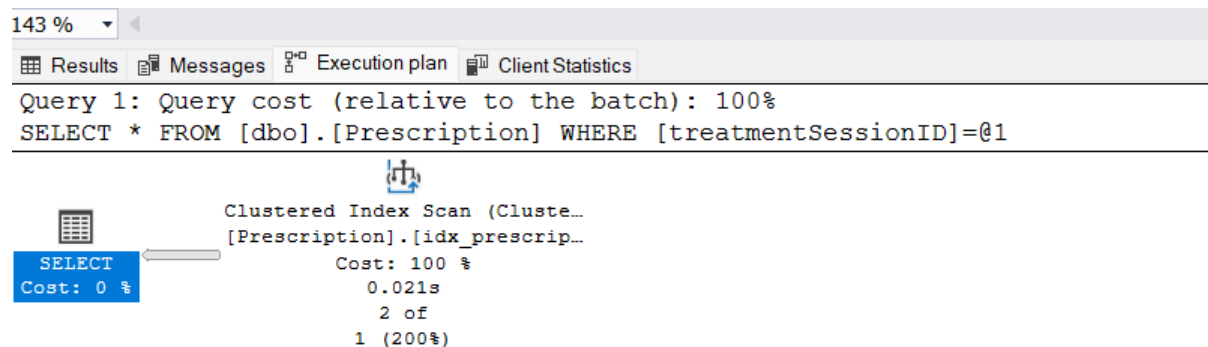
The same can be applied to queries with foreign keys.

6.3 Evaluation 3

Consider the following query: to find prescription belonging to a treatment session. This query will be used on the `Prescription` table:

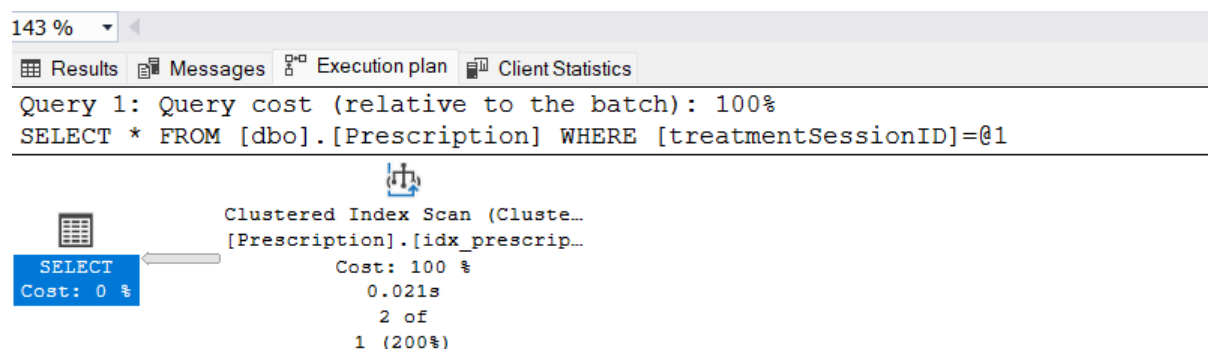
```
SELECT * FROM dbo.[Prescription]
WHERE treatmentSessionID = 500
-- The treatmentSessionID is hard-coded for testing purpose
```

When the index is created on the reverse order to what we have suggested, which is to put `drugID` first and `treatmentSessionID` second, the execution plan is as follows:



Although we have a clustered index on these two fields, the query optimizer still must scan the clustered index of the table instead of utilizing the Index Seek operator.

By placing `treatmentSessionID` first, the query optimizer can utilize the Index Seek operator:



This is because the query optimizer will utilize the index if the query has the same order as the index. Therefore, we need to make sure that the query has the same order as the index. This applies to other tables with composite primary keys as well.

PARTITIONS

We consider the following tables for partitioning since they hold the most data:

Table	No of records (to date)
Patient	~30000
AppointmentRequest	~100000
Session	~200000
TreatmentSession	~65000
ExaminationSession	~60000
ReExaminationSession	~50000
PaymentRecord	~100000

Use case 1:

Session table is partitioned by the time column, which is of *datetime2* value. We partition the table by 1-year intervals, i.e., 1 partition for each year. This is useful for the following reasons:

- When the clinic wants to view the sessions for a particular year (for statistics, etc.), the query will only have to search through the partition for that year, instead of the entire table.
- Statistics such as:
 - Number of sessions per year
 - Number of sessions per month in a year
 - etc. can be easily obtained by querying the partition for that year.
- The time column is also common in most queries, hence partitioning by time will speed up most queries.
- The data is smaller and more manageable, since the data is split into partitions.

Special use cases such as view the total number of sessions ever held can be slow due to the need to query all partitions. However, this is not a common use case and can be a trade-off for the above benefits.

Use case 2:

AppointmentRequest table is partitioned by the *appointmentTime* column, which is of *datetime2* value. We partition the table by 1-year intervals, i.e., 1 partition for each year. This is useful for the following reasons:

- When the clinic wants to view the appointment requests for a particular year (for statistics, etc.), the query will only have to search through the partition for that year, instead of the entire table.
- Statistics such as:
 - Number of appointment requests per year.

- Number of appointment requests per month in a year.
 - etc. can be easily obtained by querying the partition for that year.
- The *appointmentTime* column is also common in most queries, hence partitioning by *appointmentTime* will speed up most queries.
- The data is smaller and more manageable, since the data is split into partitions.

Special use cases such as view the total number of appointment requests ever made can be slow due to the need to query all partitions. However, this is not a common use case and can be a trade-off for the above benefits.

APPENDIX

Appendix A: Tasks Management

Tool

We use Github Projects to manage our tasks as well as our repositories. We divide the whole project into 3 repositories:

- Frontend: The UI of the project, built as a web application.
- Backend: The server of the project, built as a REST API server.
- Database: The main database, built with SQL Server. This repository also contains the database schema and all related documents.

Tasks Management

We assign each member to a repository based on their skills and preferences. However, everyone is encouraged to contribute to all repositories. For the Database repository, we assign everyone to it since it is the most important repository and is used for grading purposes. In more details:

- Frontend: Nam Hoai, Trung Thieu Vinh
- Server: Hieu Nguyen, Man Nguyen Huynh
- Database: All members

Hieu Nguyen is the project manager and is responsible for managing the tasks of the whole project, also for reviewing pull requests and merging them into the main branch. In addition, Nam Hoai is also responsible for managing the tasks of the Frontend repository.

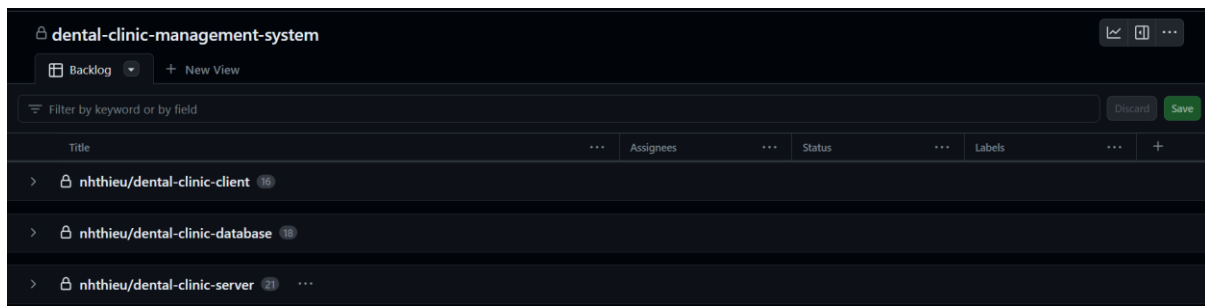
In more details:

ID	Name	Task	Percentage
20126038	Nguyễn Hồ Trung Hiếu	Indexing & partitioning scripts, documentations, server API, code review, schema design	100%
20126041	Nguyễn Huỳnh Mẫn	Stored procedures, triggers, server API, schema design, documentations	100%
20126045	Vũ Hoài Nam	Client UI, indexing, schema design, API testing, documentations	100%
20126062	Thiều Vĩnh Trung	Client UI, transactions analysis, server API,	100%

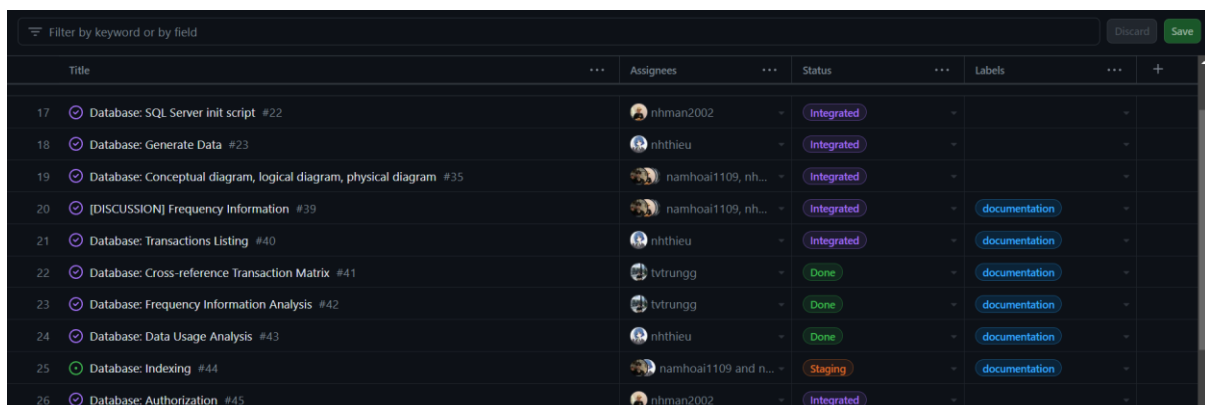
		documentations, schema design	
--	--	-------------------------------	--

Tasks

We have a view called backlog as the main board for the project. This board is then divided into 3 repositories mentioned above, each will hold all their issues (tasks) accordingly. The backlog board is used to track the progress of the project as a whole.



Each issue can be a bug, documentation, a feature, or a discussion, and can be linked to a pull request or other issues for the ease of tracking. Each issue is assigned to a member (or many members) and has a label to indicate its type. We also use the status label to indicate the status of the issue. The status label is used to track the progress of the issue.



The below image is an example of a discussion issue. It is assigned to all the members of our team and is used to discuss the design of the database schema. The issue is linked to another issue (#63) since they are related to each other. Members can leave comments on the issue to discuss the topic.

🔒 dental-clinic-database #67

[DISCUSSION] Filtered index on TreatmentSession table #67

🔒 Closed nhtieu opened 2 weeks ago

👤 nhtieu 2 weeks ago

Tình hình là t mới fix lại cái diagram (#63), và t thấy là cái use case của mình là query những treatment session chưa tạo payment record nhiều (khóa ngoại null), nên là t nghĩ nên tạo filtered index cho cái cột `paymentRecordID` trên bảng `TreatmentSession`.

The diagram illustrates the database schema with the following tables and relationships:

- PATIENT** (Primary Key: ID)
 - Attributes: NAME, DOB, GENDER, PHONE, TYPE
- PAYMENT RECORD** (Primary Key: ID)
 - Attributes: DATE, TOTAL, PAID, CHANGE, METHOD
 - Foreign Key: PATIENT_ID (to PATIENT.ID)
- SESSION** (Primary Key: ID)
 - Attributes: TIME, NOTE, STATUS, TYPE
 - Foreign Keys: PATIENT_ID (to PATIENT.ID), DENTIST_ID (to PATIENT.ID), ROOM_ID (to PATIENT.ID), ASSISTANT_ID (to PATIENT.ID)
- TREATMENT SESSION** (Primary Key: ID)
 - Attributes: HEALTH_NOTE, DESCRIPTION, CATEGORY_ID, PAYMENT_RECORD_ID
 - Foreign Keys: ID (to SESSION.ID), PAYMENT_RECORD_ID (to PAYMENT RECORD.ID)

@namhoai1109 m làm index vào review xem oke ko :)

Assignees: namhoai1109, nhtieu, nhma...

Labels: documentation

Milestone: Add milestone...

Status: Staging

Linked pull requests: No linked pull requests

Repository: dental-clinic-database

Open in new tab, Copy link, Copy link in project, Archive, Delete from project

In addition, we also conducted several team meetings to discuss the progress of the project and cafe sessions to for the team members to work together as well as discuss matter in a more convenient and precise manner.

Appendix B: Project Repository

We also included the source files with the report. In more details:



- All the diagrams are in the `diagram` folder. This includes:
 - Conceptual design diagram
 - Logical design diagram
 - Physical design diagram
- All the scripts related to the database can be found in the `src` folder
- Link for the backup of the database in case one has the need to run the source codes: [Google Drive](#).

Due to the size of other repositories, we will provide the link to them instead of including them with the report:

- The repo for the server can be found [here](#).
- The repo for the client can be found [here](#).
- The following is optional but the repo for the database can be found [here](#).

REFERENCES

- ✚ pro.ant.design. 2023. *Document overview - Ant Design Pro*. [ONLINE] Available at: <https://pro.ant.design/docs/overview/>. [Accessed 27 August 2023].
- ✚ ant.design. 2023. *Components Overview - Ant Design*. [ONLINE] Available at: <https://ant.design/components/overview>. [Accessed 27 August 2023].
- ✚ www.prisma.io. 2023. *Functional Units: Functions vs Procedures, Workflows, and More*. [ONLINE] Available at: <https://www.prisma.io/dataguide/datamodeling/functional-units>. [Accessed 27 August 2023].
- ✚ learn.microsoft.com. 2023. *Docker: Install containers for SQL Server on Linux - SQL Server / Microsoft Learn*. [ONLINE] Available at: <https://learn.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-ver16&pivots=cs1-bash>. [Accessed 27 August 2023].
- ✚ www.w3schools.com. 2023. *SQL CREATE INDEX Statement*. [ONLINE] Available at: https://www.w3schools.com/sql/sql_create_index.asp. [Accessed 27 August 2023].
- ✚ learn.microsoft.com. 2023. *CREATE INDEX (Transact-SQL) - SQL Server / Microsoft Learn*. [ONLINE] Available at: <https://learn.microsoft.com/en-us/sql/t-sql/statements/create-index-transact-sql?view=sql-server-ver16>. [Accessed 27 August 2023].
- ✚ viblo.asia. 2023. *Partition Table SQL Server*. [ONLINE] Available at: <https://viblo.asia/p/partition-table-sql-server-MLzGOzbRvpq>. [Accessed 27 August 2023].
- ✚ learn.microsoft.com. 2023. *\$PARTITION (Transact-SQL) - SQL Server / Microsoft Learn*. [ONLINE] Available at: <https://learn.microsoft.com/vi-vn/sql/t-sql/functions/partition-transact-sql?view=sql-server-ver15>. [Accessed 27 August 2023].
- ✚ www.sqlshack.com. 2016. *Insight into the SQL Server buffer cache*. [ONLINE] Available at: <https://www.sqlshack.com/insight-into-the-sql-server-buffer-cache/>. [Accessed 27 August 2023].
- ✚ learn.microsoft.com. 2023. *SQL Server technical documentation - SQL Server / Microsoft Learn*. [ONLINE] Available at: <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>. [Accessed 27 August 2023].
- ✚ vertabelo.com. 2021. *How to Model Inheritance in a Database / Vertabelo Database Modeler*. [ONLINE] Available at: <https://vertabelo.com/blog/inheritance-in-database/>. [Accessed 27 August 2023].
- ✚ vertabelo.com. 2021. *How to Model Inheritance in a Database / Vertabelo Database Modeler*. [ONLINE] Available at: <https://vertabelo.com/blog/inheritance-in-database/>. [Accessed 27 August 2023].
- ✚ docs.oracle.com. 2023. *Overview of Physical Design*. [ONLINE] Available at: https://docs.oracle.com/cd/A84870_01/doc/server.816/a76994/physical.htm. [Accessed 27 August 2023].
- ✚ www.sqlshack.com. 2021. *Transactions in SQL Server for beginners*. [ONLINE] Available at: <https://www.sqlshack.com/transactions-in-sql-server-for-beginners/>. [Accessed 27 August 2023].

-  www.sqlshack.com. 2018. *SQL index overview and strategy*. [ONLINE] Available at: <https://www.sqlshack.com/sql-index-overview-and-strategy/>. [Accessed 27 August 2023].
-  www.sqlshack.com. 2014. *Database table partitioning in SQL Server*. [ONLINE] Available at: <https://www.sqlshack.com/database-table-partitioning-sql-server/>. [Accessed 27 August 2023].