Database Systems(ACCA7002)_Abertawe Leisure Centre Design & Implementation_Lang Yan(2200468)

by Lang Yan

Submission date: 05-Jan-2023 05:35PM (UTC+0000)

Submission ID: 193721868

File name: 40672_Lang_Yan_Database_Systems_ACCA7002__Abertawe_Leisure_Centre_Design__Implementation_Lang_Yan_2200468__474126_275673317.docx (864.21K)

Word count: 2560

Character count: 18913

Abertawe Leisure Centre Design & Implementation

Module: Database Systems

Module code: ACCA7002

Lecturer: Dr Stephen Hole & Mr Nathan Morgan

Student Name: Lang Yan

Student Number: 2200468

Abstract

This paper focuses on the design and implementation of the ALC database and the testing of ALC requirement scenarios. It mainly covers the design and implementation of the database. The design part is described through the ERD, by abstracting the descriptions in the requirement scenarios, selecting the appropriate data types for the attributes and establishing the relationships. The implementation part contains the creation of the database tables, the loading of data, the implementation of functions, triggers and procedures, index \ transaction. The designed scenarios are also tested for functionality using test data.

Catalogue

Abstract — 2

Task 1. Database Creation — 4

- ERD — 4
- Database implementation — 5
  - Normalized to 3NF — 5
  - Tables and primary keys — 5
  - Referential integrity constraints (foreign keys) — 8
  - At least three check constraints — 8
  - Populate basic data into tables — 9

Task 2. Performance Tuning Enhancement — 12

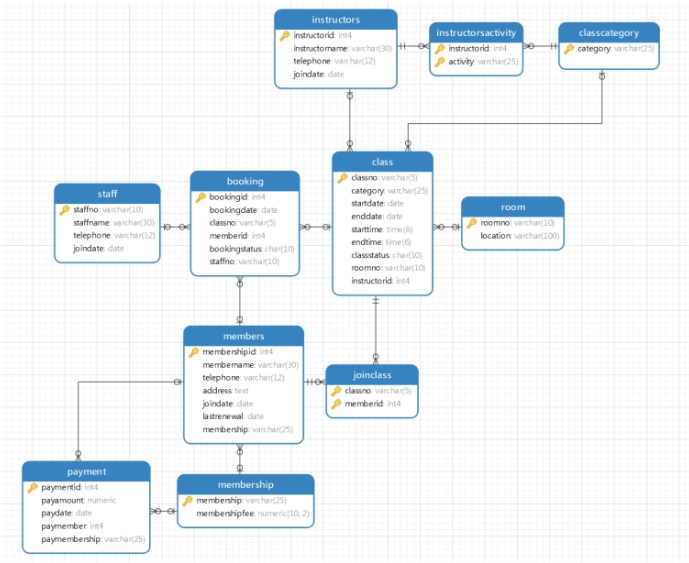
- Indexes — 12
- Functions — 13
- Triggers with an auditing feature — 15
- Stored Procedures — 16
- Test Implementation — 19
- Transaction — 23

Evaluation of Design and Implementation — 25

References — 26

Task 1. Database Creation

ERD



Database implementation

Normalized to 3NF

For the following, underscores indicate primary keys and italics indicate foreign keys.

membership (membership,membershipFee);
members
(membershipId,memberName,telephone,address,joinDate,lastRenewal,*membership*);

classCategory (category);

instructors (instructorId,instructorName,telephone,joinDate);

instructorsActivity (*instructorId*,activity);

payment (paymentId,payAmount,payDate,*payMember,payMembership*);

room (roomNo,location);

class
(classNo,*category*,startDate,endDate,startTime,endTime,classStatus,*roomNo,instructorId*);

staff (staffNo,staffName,telephone,joinDate);

booking (bookingId,bookingDate,*classNo,memberId*,bookingStatus,*staffNo*);

joinClass (*classNo,memberId*);

Tables and primary keys

```
drop table if exists joinClass;
drop table if exists booking;
drop table if exists class;
drop table if exists room;
drop table if exists payment;
drop table if exists instructorsActivity;
drop table if exists instructors;
drop table if exists staff;
```

```
drop table if exists members;
drop table if exists membership;
drop table if exists classCategory;

create table membership(
membership varchar(25) primary key,
membershipFee decimal(10,2)
);


create table members(
membershipId serial primary key,
memberName varchar(30) not null,
telephone varchar(12) not null,
address text,
joinDate date,
lastRenewal date,
membership varchar(25)
);

create table classCategory(
category varchar(25) primary key
);

create table instructors(
instructorId serial primary key,
instructorName varchar(30) not null,
telephone varchar(12) not null,
joinDate date
);

create table instructorsActivity(
instructorId int,
activity varchar(25),
primary key (instructorId, activity)
);


create table payment(
paymentId serial primary key,
payAmount decimal,
payDate date,
payMember int,
payMembership varchar(25)
```

```
);

create table room(
roomNo varchar(10) primary key,
location varchar(100)
);


create table class(
classNo varchar(5) primary key,
category varchar(25),
startDate date,
endDate date,
startTime time,
endTime time,
classStatus char(10),
roomNo varchar(10),
instructorId int
);

create table staff(
staffNo varchar(10) primary key,
staffName varchar(30) not null,
telephone varchar(12) not null,
joinDate date
);

create table booking(
bookingId serial primary key,
bookingDate date,
classNo varchar(5),
memberId int,
bookingStatus char(10),
staffNo varchar(10)
);

create table joinClass(
classNo varchar(5) not null,
memberId int not null,
primary key (classNo, memberId)
);
```

(Bopp et al., 2019)

Referential integrity constraints (foreign keys)

alter table members add constraint foreignkey_members foreign key (membership) references membership(membership);
alter table instructorsActivity add constraint foreignkey_activity foreign key (activity) references classCategory(category);
alter table instructorsActivity add constraint foreignkey_instructor foreign key (instructorId) references instructors(instructorId);
alter table payment add constraint foreignkey_payMember foreign key (payMember) references members(membershipId);
alter table payment add constraint foreignkey_paymemship foreign key (payMembership) references membership(membership);
alter table class add constraint foreignkey_instructor foreign key (instructorId) references instructors(instructorId);
alter table class add constraint foreignkey_category foreign key (category) references classCategory(category);

alter table class add constraint foreignkey_room foreign key (roomNo) references room(roomNo);
alter table booking add constraint foreignkey_bookclass foreign key (classNo) references class(classNo);
alter table booking add constraint foreignkey_bookmember foreign key (memberId) references members(membershipId);
alter table joinClass add constraint foreignkey_joinclass foreign key (classNo) references class(classNo);
alter table joinClass add constraint foreignkey_joinmember foreign key (memberId) references members(membershipId);

alter table booking add constraint foreignkey_staff foreign key (staffNo) references staff(staffNo);
(Ben et al., 2021)

At least three check constraints.

alter table booking add constraint check_bookstatus check (bookingStatus in ('ON','OFF'));
alter table class add constraint check_classstatus check (classStatus in ('Booking','Opening','Closing','Canceled'));
alter table payment add constraint check_payamount check (payAmount in (550,800,1200));

alter table members add constraint check_renewal check(lastRenewal>=joinDate); (Obrenović et al., 2019)

Populate basic data into tables

insert into membership values('Week Day',550);
insert into membership values ('Weekend',800);
insert into membership values ('All week',1200);

```
select * from membership;
```

Data Output | Explain | Messages | Notifications

	membership [PK] character varying (25)	membershipfee numeric (10,2)
1	Week Day	550.00
2	Weekend	800.00
3	All week	1200.00

insert into classCategory values ('Squash');
insert into classCategory values ('Tennis');
insert into classCategory values ('5 a Side Football');
insert into classCategory values ('Badminton');

insert into classCategory values ('Netball');

```
select * from classCategory;
```

Data Output | Explain | Messages | Notifications

	category [PK] character varying (25)
1	Squash
2	Tennis
3	5 a Side Football
4	Badminton
5	Netball

insert into instructors(instructorName, telephone, joinDate) values ('Jane Patrick','11+123125','2021-01-01');
insert into instructors(instructorName, telephone, joinDate) values ('Will Tunor','11+123125','2021-01-01');
insert into instructors(instructorName, telephone, joinDate) values ('Paul David','11+123125','2021-01-01');
insert into instructors(instructorName, telephone, joinDate) values ('Luis Wang','11+123125','2021-01-01');
insert into instructors(instructorName, telephone, joinDate) values ('Jay Chou','11+123125','2021-01-01');

```
select * from instructors;
```

Data Output Explain Messages Notifications

	instructorid [PK] integer	instructorname character varying (30)	telephone character varying (12)	joindate date
1	1	Jane Patrick	11+123125	2021-01-01
2	2	Will Tunor	11+123125	2021-01-01
3	3	Paul David	11+123125	2021-01-01
4	4	Luis Wang	11+123125	2021-01-01
5	5	Jay Chou	11+123125	2021-01-01

insert into instructorsActivity values (1,'Squash');
insert into instructorsActivity values (1,'Tennis');
insert into instructorsActivity values (2,'Squash');
insert into instructorsActivity values (3,'Squash');
insert into instructorsActivity values (3,'5 a Side Football');
insert into instructorsActivity values (4,'Badminton');
insert into instructorsActivity values (4,'Netball');

```
select * from instructorsActivity;
```

Data Output Explain Messages Notifications

	instructorid [PK] integer	activity [PK] character varying (25)
1	1	Squash
2	1	Tennis
3	2	Squash
4	3	Squash
5	3	5 a Side Football
6	4	Badminton
7	4	Netball

insert into room values ('R1001','R1001 Building1');
insert into room values ('R1002','R1002 Building1');

insert into room values ('R1003','R1003 Building1');
insert into room values ('R1004','R1004 Building1');
insert into room values ('R1005','R1005 Building1');

```
27  select * from room;
28
29
```

Data Output Explain Messages Notifications

	roomno [PK] character varying (10)	location character varying (100)
1	R1001	R1001 Building1
2	R1002	R1002 Building1
3	R1003	R1003 Building1
4	R1004	R1004 Building1
5	R1005	R1005 Building1

insert into staff values ('S101101', 'George Green','11+32031','2020-01-01');
insert into staff values ('S101102', 'George Armani','11+32032','2020-01-01');
insert into staff values ('S101103', 'Tom Fort','11+32033','2020-02-01');
insert into staff values ('S101104', 'Luis Larrun','11+32034','2020-03-01');
insert into staff values ('S101105', 'Frank Lee','11+32035','2020-04-01');

```
30  select * from staff;
```

Data Output Explain Messages Notifications

	staffno [PK] character varying (10)	staffname character varying (30)	telephone character varying (12)	joindate date
1	S101101	George Green	11+32031	2020-01-01
2	S101102	George Armani	11+32032	2020-01-01
3	S101103	Tom Fort	11+32033	2020-02-01
4	S101104	Luis Larrun	11+32034	2020-03-01
5	S101105	Frank Lee	11+32035	2020-04-01

Task 2. Performance Tuning Enhancement

Indexes

An index is a special data structure used to quickly query a data table for records that

have a particular value. With an index, you do not have to read all the information about the record when querying the data, but only the index column. Otherwise, the database system would read all the information for each record to match.

Indexes are the basis of database performance tuning techniques and are often used to achieve fast data retrieval. Therefore, the use of indexes can largely improve the query speed of the database, but also effectively improve the performance of the database system.

There are several other advantages regarding indexes.

① Creating a unique index ensures that each row of data in a database table is unique

② Speeds up table-to-table joins in terms of code integrity

③ Significantly reduce the time spent on grouping and sorting during queries

-- The member staff has created a secondary index on the start time and end time in order to constantly check the status of the course (opening, closing, etc.) based on the start time and end time. (Tapio et al., 2005)

```
create index index_class_time on class(startTime, endTime);
```

-- When scheduling the courses booked by members, they need to register in a predetermined order and therefore often need to be sorted by predetermined time. To do this, a secondary index is created in the bookingDate field.

```
create index index_booking_date on booking(bookingDate);
```

-- The ALC has developed a rebate program for old members and needs to select the early members in order of how long they have been in the program. To do this, create a secondary index in the members table joinDate.

```
create index index_joindate on members(joindate);
```

Functions

Functions are an essential element of code integrity and readability. A function is in fact a unified and complete block of code that can represent a certain function. In an object-oriented process, the subdivision of functions and tasks into modules for implementation improves readability and rationality and makes the whole more concise.

-- update member's membership information(Shin'nosuke et al., 2022)

```
drop trigger if exists update_membership on payment;
drop function if exists update_membership;

create function update_membership()
returns trigger
as $$
declare
membership_name varchar(20);
begin
select membership into membership_name from membership where membershipFee=new.payAmount;
update members set membership=membership_name, lastrenewal=now() where membershipId=new.payMember;
return new;
end;
$$ language plpgsql;
```

-- when a member booking for a class, check the number of booking of the class and decide whether opening a class.

```
drop trigger if exists update_class on booking;
drop function if exists update_class;
create function update_class()
returns trigger
as $$
declare
booking_num int:=0;
bookingRecord booking;
act varchar(20);
iid int;
begin
select count(*) into booking_num from booking where classNo=new.classNo;
if booking_num>=6 then
update class set classStatus='Opening' where classNo=new.classNo;
select category into act from class where classNo=new.classNo;
```

```
select min(instructorId) into iid from instructorsActivity where activity=act;
update class set roomno=(select min(roomno) from room), instructorId=iid where
classNo=new.classNo;
for bookingRecord in
 select * from booking where classNo=new.classNo
loop
   insert into joinClass values(bookingRecord.classNo,bookingRecord.memberId);
end loop;
end if;
return new;
commit;
end;
$$ language plpgsql;
```

-- when a member wanted to cancel a booking and check the booking status and decide whether cancel a class.

```
drop trigger if exists cancel_class on booking;
drop function if exists cancel_class;

create function cancel_class()
returns trigger
as $$
declare
booking_num int:=-1;
begin
select count(*) into booking_num from booking where classNo=old.classNo and
bookingStatus='ON';
if booking_num=0 then
update class set classStatus='Canceled' where classNo=old.classNo;
end if;
return old;
commit;
end;
$$ language plpgsql;
```

Triggers with an auditing feature

A trigger is a special kind of stored procedure. It is not controlled by subjective behaviour, but is triggered by specific events. One very advantageous feature of triggers is that:triggers can disable or roll back changes that violate referential integrity, thus cancelling the attempted data modification. Obviously, it safeguards the integrity of the data.

-- update member's membership information when a pay is made.

```
create trigger update_membership after insert or update on payment
for each row execute procedure update_membership();
```

-- when a member booking for a class, check the number of booking of the class and decide whether opening a class.

```
create trigger update_class after insert on booking
for each row execute procedure update_class();
```

-- when a member wanted to cancel a booking and check the booking status and decide whether cancel a class.

```
create trigger cancel_class after update on booking
for each row execute procedure cancel_class();
```

(Anh et al., 2021)

Stored Procedures

Stored procedures are 'code encapsulation and reuse' at the database language level. It can transfer parameters and can implement complex logical functions. At the same time it is fast and, apart from the first compilation process, subsequent calls can be executed immediately.

-- using procedure populate the members

```
drop procedure if exists proc_member;
create procedure proc_member(mName varchar(30), phone varchar(12), addr text, joinDate
date)
as $$
begin
insert into members(memberName, telephone, address, joinDate, lastRenewal)
values(mName,phone,addr, joinDate, joinDate);
commit;
end;
$$ language plpgsql;
```

-- populate the payments.

```
drop procedure if exists proc_paying;
create procedure proc_paying(mName varchar(30),
payDate date,
payAmount decimal,
payMembership varchar(25))
as $$
declare
mid int:=-1;
begin
select membershipId into mid from members where memberName=mName;
if mid=-1 then
rollback ;
end if;
insert into payment(payAmount, payDate,payMember, payMembership) values( payAmount,
payDate,mid, payMembership);
commit;
end;
$$ language plpgsql;
```

-- populate the classes.

```
drop procedure if exists proc_class;
create procedure proc_class(
classNo varchar(5),
category varchar(25),
startDate date,
endDate date,
startTime time,
endTime time,
classStatus char(10)
)
as $$
begin
insert into class(classNo,category, startDate, endDate, startTime, endTime, classStatus)
values (classNo,category,startDate, endDate, startTime, endTime, classStatus);
commit;
end;
$$ language plpgsql;
```

-- populate the bookings.

```
drop procedure if exists proc_booking;
create procedure proc_booking(bookingDate date,
classNo varchar(5),
mName varchar(30),
bookingStatus char(10),
sName varchar(30))
as $$
declare
mid int:=-1;
sid varchar(10):='staff';
begin
select membershipId into mid from members where memberName=mName;
if mid=-1 then
rollback ;
end if;
select staffNo into sid from staff where staffName=sName;
if sid='staff' then
rollback ;
```

```
end if;
insert into booking(bookingDate, classNo, memberId, bookingStatus, staffNo) values
(bookingDate, classNo, mid, bookingStatus, sid);
commit;
end;
$$ language plpgsql;
```

-- update the booking

```
drop procedure if exists proc_update_booking;
create procedure proc_update_booking(bDate date,
cNo varchar(5),
mName varchar(30),
status char(10),
sName varchar(30))
as $$
declare
mid int:=-1;
sid varchar(10):='staff';
begin
select membershipId into mid from members where memberName=mName;
if mid=-1 then
rollback ;
end if;
select staffNo into sid from staff where staffName=sName;
if sid='staff' then
rollback ;
end if;
update booking set bookingStatus=status where bookingDate=bDate and classNo=cNo
and memberId=mid and staffNo=sid;
commit;
end;
$$ language plpgsql;
```
(Bin et al., 2020)

Test Implementation

The staff's is load the members into the database, call the procedure proc_member

call proc_member('Ross Geller','11+123456','New York','2022-12-01');
call proc_member('Monica Geller','11+323456','New York','2022-12-01');
call proc_member('Chandler Bing','11+423456','New York','2022-12-01');
call proc_member('Pheobe Buffey','11+623456','New York','2022-12-01');
call proc_member('Rachel Green','11+723456','New York','2022-12-01');
call proc_member('Joey Tribbone','11+823456','New York','2022-12-01');
call proc_member('Emily Luis','11+923456','London','2022-12-01');

```
select * from members;
```

Data Output Explain Messages Notifications

	membershipid [PK] integer	membername character varying (30)	telephone character varying (12)	address text	joindate date	lastrenewal date	membership character varying (25)
1	2	Ross Geller	11+123456	New York	2022-12-01	2022-12-01	[null]
2	3	Monica Geller	11+323456	New York	2022-12-01	2022-12-01	[null]
3	4	Chandler Bing	11+423456	New York	2022-12-01	2022-12-01	[null]
4	5	Pheobe Buffey	11+623456	New York	2022-12-01	2022-12-01	[null]
5	6	Rachel Green	11+723456	New York	2022-12-01	2022-12-01	[null]
6	7	Joey Tribbone	11+823456	New York	2022-12-01	2022-12-01	[null]
7	8	Emily Luis	11+923456	London	2022-12-01	2022-12-01	[null]

The staff continues to handle the member's payment for the membership. Call the procedure proc_paying.

call proc_paying('Ross Geller',now()::date, 550, 'Week Day');
call proc_paying('Monica Geller',now()::date, 800, 'Weekend');
call proc_paying('Chandler Bing',now()::date, 1200, 'All week');
call proc_paying('Pheobe Buffey',now()::date, 550, 'Week Day');
call proc_paying('Rachel Green',now()::date, 800, 'Weekend');
call proc_paying('Joey Tribbone',now()::date, 550, 'Week Day');
call proc_paying('Emily Luis',now()::date, 550, 'Week Day');

then the payments are:

Query Editor Query History

```
select * from payment;

select * from members;
```

Data Output Explain Messages Notifications

	paymentid [PK] integer	payamount numeric	paydate date	paymember integer	paymembership character varying (25)
1	1	550	2023-01-03	2	Week Day
2	2	800	2023-01-03	3	Weekend
3	3	1200	2023-01-03	4	All week
4	4	550	2023-01-03	5	Week Day
5	5	800	2023-01-03	6	Weekend
6	6	550	2023-01-03	7	Week Day
7	7	550	2023-01-03	8	Week Day

And the members membership just updated:

```
select * from payment;

select * from members;
```

Data Output Explain Messages Notifications

	membershipid [PK] integer	membername character varying (30)	telephone character varying (12)	address text	joindate date	lastrenewal date	membership character varying (25)
1	2	Ross Geller	11+123456	New York	2022-12-01	2023-01-03	Week Day
2	3	Monica Geller	11+323456	New York	2022-12-01	2023-01-03	Weekend
3	4	Chandler Bing	11+423456	New York	2022-12-01	2023-01-03	All week
4	5	Pheobe Buffey	11+623456	New York	2022-12-01	2023-01-03	Week Day
5	6	Rachel Green	11+723456	New York	2022-12-01	2023-01-03	Weekend
6	7	Joey Tribbone	11+823456	New York	2022-12-01	2023-01-03	Week Day
7	8	Emily Luis	11+923456	London	2022-12-01	2023-01-03	Week Day

The staff then continues to add the class information which are used to booked by the members.

call proc_class('c0001','Squash','2023-01-31','2023-06-30','9:00:00','18:00:00','Booking');
call proc_class('c0002','Tennis','2023-02-10','2023-05-30','12:00:00','18:00:00','Booking');

```
select * from class;
```

Data Output Explain Messages Notifications

	classno [PK] character varying (5)	category character varying (25)	startdate date	enddate date	starttime time without time zone	endtime time without time zone	classstatus character (10)	roomno character varying (10)	instructorid integer
1	c0001	Squash	2023-01-31	2023-06-30	09:00:00	18:00:00	Booking	[null]	[null]
2	c0002	Tennis	2023-02-10	2023-05-30	12:00:00	18:00:00	Booking	[null]	[null]

The staff handles the class booking information:

call proc_booking('2023-01-01','c0001','Ross Geller','ON','George Armani');
call proc_booking(now()::date,'c0001','Monica Geller','ON','George Armani');
call proc_booking(now()::date,'c0001','Chandler Bing','ON','George Armani');
call proc_booking('2023-01-02','c0001','Pheobe Buffey','ON','George Armani');
call proc_booking('2023-01-04','c0001','Rachel Green','ON','George Armani');
call proc_booking('2023-01-02','c0001','Joey Tribbone','ON','George Armani');
call proc_booking('2023-01-01','c0002','Emily Luis','ON','Tom Fort');

```
select * from booking;
```

Data Output | Explain | Messages | Notifications

	bookingid [PK] integer	bookingdate date	classno character varying (5)	memberid integer	bookingstatus character (10)	staffno character varying (10)
1	1	2023-01-01	c0001	2	ON	S101102
2	2	2023-01-03	c0001	3	ON	S101102
3	3	2023-01-03	c0001	4	ON	S101102
4	4	2023-01-02	c0001	5	ON	S101102
5	5	2023-01-04	c0001	6	ON	S101102
6	6	2023-01-02	c0001	7	ON	S101102
7	7	2023-01-01	c0002	8	ON	S101103

Then the 2-7 members are booking class c0001, c0001 is opening.

```
select * from class;
```

Data Output | Explain | Messages | Notifications

	classno [PK] character varying (5)	category character varying (25)	startdate date	enddate date	starttime time without time zone	endtime time without time zone	classstatus character (10)	roomno character varying (10)	instructorid integer
1	c0002	Tennis	2023-02-10	2023-05-30	12:00:00	18:00:00	Booking	[null]	[null]
2	c0001	Squash	2023-01-31	2023-06-30	09:00:00	18:00:00	Opening	R1001	1

And they are all joined the class c0001.

```
select * from joinclass;
```

Data Output | Explain | Messages | Notifications

	classno [PK] character varying (5)	memberid [PK] integer
1	c0001	2
2	c0001	3
3	c0001	4
4	c0001	5
5	c0001	6
6	c0001	7

When a member wanted to cancel the booking.

call proc_update_booking('2023-01-01','c0002','Emily Luis','OFF','Tom Fort');

```
select * from booking;
```

Data Output | Explain | Messages | Notifications

	bookingid [PK] integer	bookingdate date	classno character varying (5)	memberid integer	bookingstatus character (10)	staffno character varying (10)
1	1	2023-01-01	c0001	2	ON	S101102
2	2	2023-01-03	c0001	3	ON	S101102
3	3	2023-01-03	c0001	4	ON	S101102
4	4	2023-01-02	c0001	5	ON	S101102
5	5	2023-01-04	c0001	6	ON	S101102
6	6	2023-01-02	c0001	7	ON	S101102
7	7	2023-01-01	c0002	8	OFF	S101103

When there's no booking is on, so the class is canceled.

```
select * from class;
```

Data Output | Explain | Messages | Notifications

	classno [PK] character varying (5)	category character varying (25)	startdate date	enddate date	starttime time without time zone	endtime time without time zone	classstatus character (10)	roomno character varying (10)	instructorid integer
1	c0001	Squash	2023-01-31	2023-06-30	09:00:00	18:00:00	Opening	R1001	1
2	c0002	Tennis	2023-02-10	2023-05-30	12:00:00	18:00:00	Canceled	[null]	[null]

Transaction

Transactions were created to simplify the programming model when applications access the database, eliminating the need for people to consider a wide range of potential errors and concurrency issues. So transactions are essentially for the application layer. They do not come with the database system.

-- Member Ross Geller wants to booking c0002 but he found the time is not very suitable, so the staff didn't submit.

```
start transaction;

insert into booking(bookingDate, classNo, memberId, bookingStatus, staffNo) values (now()::date, 'c0002', (select membershipId from members where memberName='Ross Geller'), 'ON', (select staffNo from staff where staffName like 'Tom%'));

rollback;

select * from booking;
```

(Jerry et al., 2016)

-- Member Ross Geller still wants to booking c0002

```
start transaction;

insert into booking(bookingDate, classNo, memberId, bookingStatus, staffNo) values (now()::date, 'c0002', (select membershipId from members where memberName='Ross Geller'), 'ON', (select staffNo from staff where staffName like 'Tom%'));

commit;
```

bookingid [PK] integer	bookingdate date	classno character varying (5)	memberid integer	bookingstatus character (10)	staffno character varying (10)
1	2023-01-01	c0001	2	ON	S101102
2	2023-01-03	c0001	3	ON	S101102
3	2023-01-03	c0001	4	ON	S101102
4	2023-01-02	c0001	5	ON	S101102
5	2023-01-04	c0001	6	ON	S101102
6	2023-01-02	c0001	7	ON	S101102
7	2023-01-01	c0002	8	OFF	S101103
9	2023-01-03	c0002	2	ON	S101103

Evaluation of Design and Implementation

The ERD in this design conforms to 3NF and provides an accurate description of the one-to-many and many-to-many relationships of entities in an ALC scenario. It conforms to the principle of reducing redundancy for data storage.

The design of the database for ALC can be used in multi-user scenarios. When different users operate on the database, they can do so simultaneously and without interfering with each other. The isolation level of the database should be set above repeatable reads to avoid problems such as lost updates, dirty reads and non-repeatability.

References

https://www.postgresql.org/docs/14/trigger-definition.html

Anh, L.H., To, V.K. & Truong, T.N..(2021).Formal Analysis of Database Trigger Systems Using Event-B. *International Journal of Software Innovation (IJSI)*(4). DOI:10.4018/IJSI.20211001.OA1.

Ben, S., AlZain A. Mohammed,Li S. Alice & Samra E. Halima.(2021). Review of Contemporary Database Design and Implication for Big Data. *International Journal of Smart Education and Urban Society (IJSEUS)*(4). DOI:10.4018/IJSEUS.2021100101.

Bin, M.M.H. & Chan, P.H..(2020).Development of database and stored procedures for E-commerce systems. *Journal of Advanced Marine Engineering and Technology*(1). DOI:10.5916/jamet.2020.44.1.55.

Bopp, C., Lehn M. Benjamin & Amy Voida.(2019).The Coerciveness of the Primary Key. *Proceedings of the ACM on Human-Computer Interaction*(CSCW). DOI:10.1145/3359153.

Jerry Chun-Wei Lin,Qiankun Liu,Philippe Fournier-Viger & Tzung-Pei Hong.(2016).PTA: An Efficient System for Transaction Database Anonymization.. *IEEE Access*.

Obrenović, N., Luković, I. & Ristić, S..(2019).Consolidation of database check constraints.. *Software & Systems Modeling*(3). DOI:10.1007/s10270-017-0637-2.

Shin'nosuke, Y., Hideki, K., Yoshimasa, O. & Kazunori, N..(2022).Design of Question-and-Answer Interface using Moodle DATABASE Function. *Procedia Computer Science.* DOI:10.1016/J.PROCS.2022.09.153.

Tapio, L. & Michael, L..(2005).*Relational Database Index Design and the Optimizers:DB2, Oracle, SQL Server, et al*..John Wiley & Sons, Inc.. DOI: 10.1002/0471721379

Database Systems(ACCA7002)_Abertawe Leisure Centre Design & Implementation_Lang Yan(2200468)

GRADEMARK REPORT

FINAL GRADE

89 /100

GENERAL COMMENTS

Instructor

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills .	10	5
Selection of relevant and credible sources. Very good use of referencing, with no inaccuracies. You need to discuss the importance of the concepts of rollback, commit, functions, triggers, procedures, transactions and indexes and support your decisions with citations.		
2. Knowledge and Understanding Skills	20	18
An excellent Entity Relationship Diagram (ERD) has been produced. The ERD demonstrates an excellent mastery of a complex and specialised area of knowledge.		
3. Cognitive and Intellectual Skills	20	26
You have clearly demonstrated your ability to analyse the case study problem scenario and propose an appropriate, normalised to 3NF, database schema. Exceptional critical evaluation/ critique of case study scenario.		

4. Practical Application Skills	30	26
You have taken the ERD and produced and excellent database structure with Primary and Foreign Keys, indexes, procedures and functions. The database has been tested with appropriate SQL data manipulations statements. An advanced application of knowledge, methods, materials, tools and/or techniques. The context of the application is well considered, and insightful.		
5. Transferable Skills for Life and Professional Practice	20	14
Work is coherent, very fluent and is presented professionally. Can work autonomously with initiative. You need to discuss the importance of the concepts of rollback, commit, functions, triggers, procedures, transactions and indexes and support your decisions with citations.		

Assessment Mark (Assessment marks are subjec ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guida only until they are confirmed.)

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26