1. Analysis of Timetable Management System

Data corresponding to the following would be readily available:

1. A university/college has several degree programs like BE, MCA, ME going on simultaneously. In each of this degree, there are sections from different years. For example: BE 1st year, BE 2nd year and so on..
2. Each semester, a new timetable is to be made according to the available faculty and rooms, sections and courses.
3. A particular teacher teachers a particular course to 1 or many sections.
4. The rooms have some capacity which can be measured in terms of number of sections that can sit in simultaneously. (Strength variation among various sections could be ignored.)
5. Some courses are such which can only be held in only some particular room(s). For example: DBMS LAB in CCCT LAB or CCMX LAB. The order of rooms could be treated as given in decreasing order of preference.

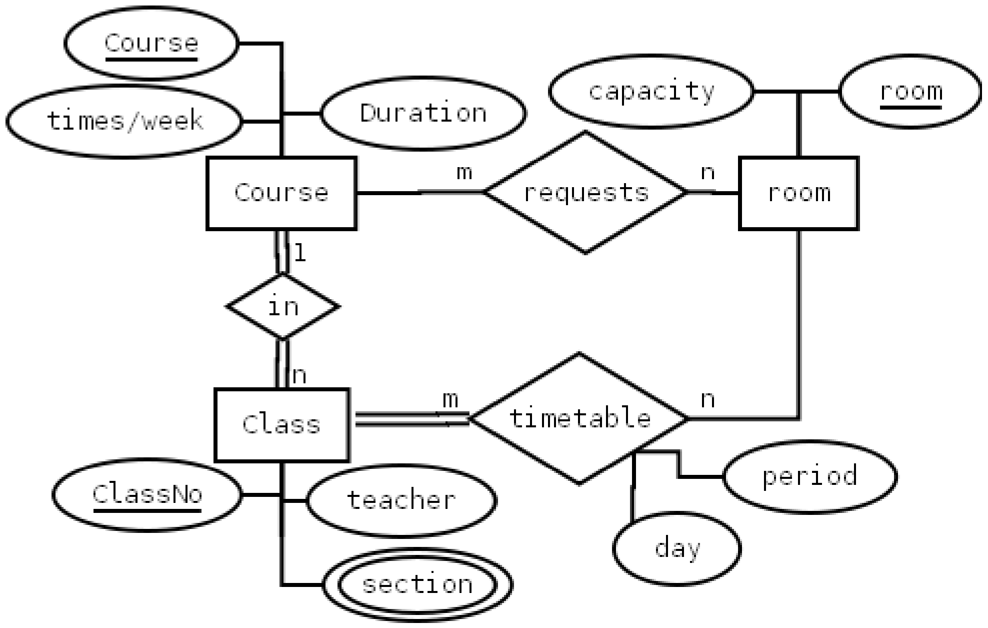
Constraints which would be provided in the form of the check boxes, user may choose the required:

1. No two consecutive sessions could be scheduled for a teacher.
2. For a teacher, no more than 1 class for a particular course should be scheduled.
3. Every section must have either 12:00 to 1:00pm or 1:00 to 2:00pm for lunch. (atleast 1)

Following output is required:

1. Which class/session will held in which room (Room no. or lab) and at what time (day and period)?

2. Entity Relationship Diagram



3. Tables for Timetable Project (along with sample data)

1. CLASS

classNo	teacher	section	course
1	AV	COE1	CS001L
1	AV	COE2	CS001L
1	AV	COE3	CS001L
2	PB	COE4	CS001L
2	PB	COE5	CS001L
2	PB	COE6	CS001L
3	AV	COE3	CS001P

2. COURSE

course(PK)	Duration (in no. of periods)	NoOfTimes/ week
CS001L	1	3
CS001L	1	3
CS001P	2	1
	and so on....	

3. ROOM

SerialNo	RoomNo(PK)	Capacity (in terms of no. of sections)
1	F101	1
2	F102	6
3	D115	6
4	CCCT1 LAB PART1	1
5	CCCT1 LAB PART2	1
	and so on...	

4. RoomRequests

Course	RoomNo(FK)
CS001P	CCCT1 LAB
CS001P	CCMX LAB
<EGRA TUT>	B107
<EGRA TUT>	F206
<COM. SKILLS TUT>	E204
and so on...	

5. RoomsAllocs

RoomNo(FK)(PK)	Day(PK)	Period(PK)	ClassNo (default=0, means free room)
F102	MON	1	1
F103	MON	1	2
E206	MON	1	3
F102	MON	2	0
and so on..			

4. Normalized Tables for Timetable Project (along with sample data)

CLASS-I

ClassNo(PK)(FK)	**Section(PK)**
1	COE1
1	COE2
1	COE3
2	COE4
2	COE5
2	COE6
3	COE3
And so on..	

CLASS-II

classNo(PK)	**Teacher**	**course**
1	AV	CS001L
2	PB	CS001L
3	AV	CS001P
and so on....		

COURSE

course(PK)	**Duration (in no. of periods)**	**NoOfTimes/ week**
CS001L	1	3
CS001L	1	3
CS001P	2	1
	and so on....	

ROOM

SerialNo<sequence>	**RoomNo(PK)**	**Capacity (in terms of no. of sections)**
1	F101	1
2	F102	6
3	D115	6
4	CCCT1 LAB PART1	1
5	CCCT1 LAB PART2	1
	and so on...	

RoomRequests

Course(FK ON COURSE)	**RoomNo(FK ON ROOM)**
CS001P	CCCT1 LAB
CS001P	CCMX LAB
<EGRA TUT>	B107
<EGRA TUT>	F206
<COM. SKILLS TUT>	E204
and so on...	

RoomsAllocs

RoomNo(FK ON ROOM) (PK)	**Day(PK)**	**Period(PK)**	**ClassNo (FK)**

F102	MON	1	1
F103	MON	1	2
E206	MON	1	3
F102	MON	2	0
and so on..			

5. Functional Requirements of Timetable Management System

Procedure or working: In the computation of timetable, a loop is run to schedule classes specified by classNo. In each iteration, the following query is run:

1. SELECT * FROM class-I WHERE classNo = <counter>;
2. SELECT * FROM class-II WHERE classNo = <counter>;
3. SELECT * FROM course WHERE course = <course retrieved above>;

Now the course retrieved is checked in the roomRequests table using the following query:

4. SELECT roomNo FROM roomRequest WHERE course = <retrieved above>

If present, a room, day and period is chosen on FCFS(first come first serve basis). The selected combination of room, day and period is checked for constraints. Few constraints are mentioned below as illustrations:

- capacity (no. of sections)
- duration (is the selected room available for full duration specially in case of 2-hr or 3-hr labs)
- Is the room empty for the day and period selected?
- And other constraints selected by the user of the software.

If the room, day, period satisfies the constraints, it would be added to Timetable table else the combination of room, day, period is blacklisted for the current classNo and a new combination of room, day, period is chosen and so on… till a satisfactory room is not found.

Functional Requirements:

1. Here the first three queries, retrieves information about the selected classNo. This effort can be reduced to one-third by merging these three tables. Retrieving data from only 1 table would be faster than retrieving the same information by 3 SELECT queries.
2. For the fourth query, the roomRequest table can be merged with the table having course value which will make the work done without the need to access another table i.e. roomRequest.

Result: Only first FR is considered for De-Normalisation. The implementation of second FR results in a single bulky table. Instead an Index is implemented to fasten this operation.

6. Revisiting Database Design
De-normalized Tables for Timetable Project (along with sample data)

1. CLASS

classNo	teacher	section	course	Duration (in no. of periods)	NoOfTimes/ week
1	AV	COE1	CS001L	1	3
1	AV	COE2	CS001L	1	3
1	AV	COE3	CS001L	1	3
2	PB	COE4	CS001L	1	3
2	PB	COE5	CS001L	1	3
2	PB	COE6	CS001L	1	3
3	AV	COE3	CS001P	2	1
				and so on....	

2. ROOM

RoomNo<Sequence>	RoomNo(PK)	Capacity (in terms of no. of sections)
1	F101	1
2	F102	6
3	CCCT1 LAB PART1	1
4	CCCT1 LAB PART2	1
	and so on...	

3. RoomRequests

Course<index>	RoomNo(FK)
CS001P	CCCT1 LAB
CS001P	CCMX LAB
<EGRA TUT>	B107
<EGRA TUT>	F206
<COM. SKILLS TUT>	E204

4. RoomsAllocs or Timetable

RoomNo(FK)(PK)	Day(PK)	Period(PK)	ClassNo (default=0, means free room)
F102	MON	1	1
F103	MON	1	2
E206	MON	1	3
F102	MON	2	0
and so on..			

7. Database Implementation

1. Database Logical Objects used

- **Table** Class: to contain the attributes of room cl and to implement the relation between Class and course as shown in ER diagram.
- **Table** Room: to contain the attributes of room entity
- **Table** RoomRequests: to implement the relation between course and room as shown in ER diagram.
- **Table** TimeTable: To apply the relationship between class table and room table
- **Trigger**: Primary Key on RoomNo in Room table.
- **Trigger**: RoomRequests(RoomNo) references Room(RoomNo)
- **Trigger**: RoomNo,Day,Period composite primary key in TimeTable Table
- **Trigger**: TimeTable(RoomNo) references Room(RoomNo)
- **Trigger**: Set Delete cascade Trigger on TimeTable(RoomNo) references Room(RoomNo)
- **View** coe1: to view timetable for coe1 section.
- **View** AV: to view timetable for teacher AV
- **Package** package_timetable: It contains functions and procedures
- **Function** func_cal_hrs: to calculate number of hours/Week of a section using cursor
- **Procedure** proc_cal_hrs: to calculate number of hours/Week of a section using cursor
- **Sequence** sr_no_room: on SerialNo in Room table which would help to have a count on no. of rooms provided no room will be deleted from this room, once inserted.
- **Index** course_roomrequest: Index on roomrequest table used because searching is done on this field as mentioned in query 4 in Functional Requirements.

2. Details Of Logical Objects

- Table: Class

```
CREATE TABLE class(

classno NUMBER(4),

teacher VARCHAR(5),

section VARCHAR(5),
course VARCHAR(9),
duration NUMBER(1),
nooftimesperweek NUMBER(1)
);
```

- Table: Room

```
CREATE TABLE room(
roomno VARCHAR(20),
capacity NUMBER(1)
);
```

- Table: RoomRequests

```
CREATE TABLE roomrequests(
course VARCHAR(9),
roomno VARCHAR(20)
);
```

- Table: TimeTable

```
CREATE TABLE timetable(
roomno VARCHAR(20),
day CHAR(3),
period NUMBER(10),
classno NUMBER(4)
);
```

- Trigger: Primary Key on RoomNo in Room table.

```
CREATE OR REPLACE TRIGGER pk_roomno
BEFORE INSERT OR UPDATE ON Room FOR EACH ROW
DECLARE
r Room.RoomNo%TYPE;
BEGIN
IF :NEW.RoomNo IS NULL THEN
```

```
raise_application_error(-20005, 'primary key constraint violated');
END IF;
SELECT RoomNo INTO r FROM Room WHERE RoomNo = :NEW.RoomNo;
IF SQL%ROWCOUNT = 1 THEN
raise_application_error(-20005, 'primary key constraint violated');
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE(‘okay.. row inserted');
END;
```

- Trigger: RoomRequests(RoomNo) references Room(RoomNo)

```
CREATE OR REPLACE TRIGGER fk_RoomRequests
BEFORE INSERT OR UPDATE ON RoomRequests
FOR EACH ROW
DECLARE
r RoomRequests.RoomNo%TYPE;
BEGIN
Select RoomNo INTO r from Room where RoomNo = :NEW.RoomNo;
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20005, 'foreign key constraint violated');
WHEN TOO_MANY_ROWS THEN
raise_application_error(-20006, 'UNIQUE constraint of parent table is violated');
END;
```

- Trigger: RoomNo,Day,Period composite primary key in TimeTable Table

```
CREATE OR REPLACE TRIGGER pk_roomno
BEFORE INSERT OR UPDATE ON Room FOR EACH ROW
DECLARE
```

```
r Room%ROWTYPE;

BEGIN

IF :NEW.RoomNo IS NULL or :NEW.Day IS NULL or :NEW.Period IS NULL then

raise_application_error(-20005, 'primary key constraint violated');

END IF;

Select * INTO r from Room where RoomNo = :NEW.RoomNo and Day = :NEW.Day and Period
= :NEW.Period;

if SQL%ROWCOUNT = 1 then

raise_application_error(-20005, 'primary key constraint violated');

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('OKAY ROW INSERTED');

END;
```

- <u>Trigger: TimeTable(RoomNo) references Room(RoomNo)</u>

```
CREATE OR REPLACE TRIGGER fk_TimeTable

BEFORE INSERT OR UPDATE ON TimeTable

FOR EACH ROW

declare

r TimeTable.RoomNo%TYPE;

begin

Select RoomNo INTO r from Room where RoomNo = :NEW.RoomNo;

EXCEPTION

WHEN NO_DATA_FOUND THEN

raise_application_error(-20005, 'foreign key constraint violated');

WHEN TOO_MANY_ROWS THEN

raise_application_error(-20006, 'UNIQUE constraint of parent table is violated');
```

```
END;
```

- Set Delete cascade Trigger: TimeTable(RoomNo) references Room(RoomNo)

```
CREATE OR REPLACE TRIGGER del_cascade AFTER DELETE ON Room FOR EACH ROW

DECLARE

BEGIN

IF DELETING THEN

DELETE FROM TimeTable WHERE RoomNo= :OLD.RoomNo;

END IF;

END;
```

- View : to view timetable for coe1 section.

```
CREATE OR REPLACE VIEW coe1 AS

SELECT * FROM timetable WHERE classNo IN (SELECT classNo FROM class WHERE
LOWER(section) LIKE ‘coe1%’);
```

- View : to view timetable for AV teacher.

```
CREATE OR REPLACE VIEW AV AS

SELECT * FROM timetable WHERE classNo IN (SELECT classNo FROM class WHERE
LOWER(teacher) LIKE ‘AV%’);
```

- Package package_timetable: To calculate number of hours of teaching for a section on week basis

```
CREATE OR REPLACE PACKAGE package_timetable AS
        PROCEDURE pack_proc_sec_cal_hrs(section IN VARCHAR, total_hrs OUT NUMBER);
        FUNCTION  pack_func_sec_cal_hrs(section  VARCHAR) RETURN NUMBER;
END;
CREATE OR REPLACE PACKAGE BODY package_timetable IS
        PROCEDURE pack_proc_sec_cal_hrs(section IN VARCHAR, total_hrs OUT NUMBER)
AS
                cur CURSOR IN SELECT * FROM class WHERE (LOWER(section))=section;

                rec class%ROWTYPE;

                BEGIN

                        total_hrs := 0;

                        FOR cur IN rec LOOP
```

```
                    total_hrs = total_hrs + (rec.nooftimesperweek*rec.duration);

                END LOOP;

            EXCEPTION

                WHEN NO_DATA_FOUND THEN

                DBMS_OUTPUT.PUT_LINE('No such Section');

            END pack_proc_sec_cal_hrs;
        FUNCTION  pack_func_sec_cal_hrs(section  VARCHAR) RETURN NUMBER AS
            total_hrs NUMBER;

            cur CURSOR IN SELECT * FROM class WHERE (LOWER(section))=section;

            rec class%ROWTYPE;

            BEGIN

                total_hrs := 0;

                FOR cur IN rec LOOP

                    total_hrs = total_hrs + (rec.nooftimesperweek*rec.duration);

                END LOOP;

                RETURN(total_hrs);

            EXCEPTION

                WHEN NO_DATA_FOUND THEN

                DBMS_OUTPUT.PUT_LINE('No such Section');

            END pack_func_sec_cal_hrs;

END package_timetable;
```

<u>We can call the procedure in the following manner:</u>

```
DECLARE

        section VARCHAR := ‘COE1’;

        total_hrs NUMBER := 0;

BEGIN

        PACK_TIMETABLE.pack_proc_sec_cal_hrs(section, total_hrs);

        DBMS_OUTPUT.PUT_LINE(‘TOTAL HOURS FOR ’ || section || ‘ ARE ‘ ||  total_hrs);

END;
```

We can call the function in the following manner:

```
DECLARE

        section VARCHAR := ‘COE1’;

        total_hrs NUMBER := 0;

BEGIN

        total_hrs := PACK_TIMETABLE.pack_func_sec_cal_hrs(section);

        DBMS_OUTPUT.PUT_LINE(‘TOTAL HOURS FOR ’ || section || ‘ ARE ‘ ||  total_hrs);

END;
```

- Function func_cal_hrs: to calculate number of hours/Week of a section

```
FUNCTION  func_sec_cal_hrs(section  VARCHAR) RETURN NUMBER AS
        count NUMBER;

        duratn NUMBER;

        total_hrs NUMBER;

        cur CURSOR IN SELECT * FROM class WHERE (LOWER(section))=section;

        rec class%ROWTYPE;

        BEGIN

                total_hrs := 0;

                FOR cur IN rec LOOP

                        total_hrs = total_hrs + (rec.nooftimesperweek*rec.duration);

                END LOOP;

                RETURN(total);

        EXCEPTION

                WHEN NO_DATA_FOUND THEN

                DBMS_OUTPUT.PUT_LINE('No such Section');

        END;
```

We can call the function in the following manner:

```
DECLARE

        section VARCHAR := ‘COE1’;
        total_hrs NUMBER := 0;
BEGIN
        total_hrs := func_sec_cal_hrs(section);
        DBMS_OUTPUT.PUT_LINE('TOTAL HOURS FOR ' || section || ' ARE ' ||  total_hrs);
END;
```

- Procedure pro_cal_hrs: to calculate number of hours/Week of a section

```
PROCEDURE proc_sec_cal_hrs(section IN VARCHAR, HrsPerWeek OUT NUMBER) AS
total_hrs NUMBER;
cur CURSOR IN SELECT * FROM class WHERE (LOWER(section))=section;
rec class%ROWTYPE;
BEGIN
        total_hrs := 0;
        FOR cur IN rec LOOP
                total_hrs = total_hrs + (rec.nooftimesperweek*rec.duration);
        END LOOP;
        RETURN(total);
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No such Section');
END;
```

We can call the procedure in the following manner:

```
DECLARE
        section VARCHAR := 'COE1';
        total_hrs NUMBER := 0;
BEGIN
        proc_sec_cal_hrs(section, total_hrs);
        DBMS_OUTPUT.PUT_LINE('TOTAL HOURS FOR ' || section || ' ARE ' ||  total_hrs);
END;
```

- <u>Sequence sr_no_room: on SerialNo in Room table</u>

```
CREATE SEQUENCE sr_no_room
        START WITH 1
        INCREMENT BY 1
        MINVAL 1
        MAXVAL 1000;
```

<u>INSERT QUERY WOULD BE:</u>

```
        INSERT INTO roomRequests VALUES(sr_no_room.NEXTVAL, <course>, <room>);
```

<u>TO CHECK THE CURRENT VALUE OF SEQUENCE:</u>

```
        SELECT sr_no_room.CURVAL FROM DUAL;
```

- <u>Index course_roomrequest: Index on roomrequeset table</u>

```
CREATE INDEX course_roomrequest ON roomRequest (course);
```