**The following represents a relational schema:**

Building(buildingNo, buildingName, postCode)

Room(buildingNo, roomNo, floor, maxOccupancy, owningDeptCode)

User(userNo, userName, deptCode)

Dept(deptCode, deptName)

Booking(buildingNo, roomNo, date, startTime, duration, userNo)

**What are the primary keys:**

Building =buildingNo

Room =buildingNo, roomNo (composite key)

User =userNo Dept =deptCode

Booking =buildingNo, roomNo, date, startTime (composite key)

**Are there any other candidate keys:**

BuildingName for Building?

Username for User?

deptName for Dept?

**What are the foreign keys:**

Building has no foreign keys Room has two foreign keys: buildingNo (key to Building) and owningDept (key to Dept)

User has deptCode (key to Dept)

Dept has no foreign keys

Booking has two foreign keys: userNo (key to User) and buildingNo,roomNo (key to Room)

**What constraints would be useful for this schema:**

Possible constraints needed

Building: postcode standard format or lookup

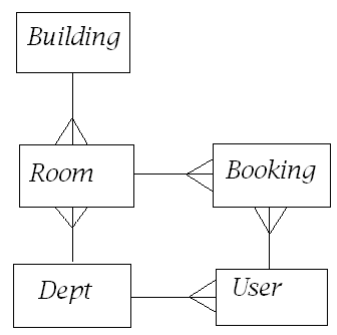
Room: range for floor and maxOccupancy

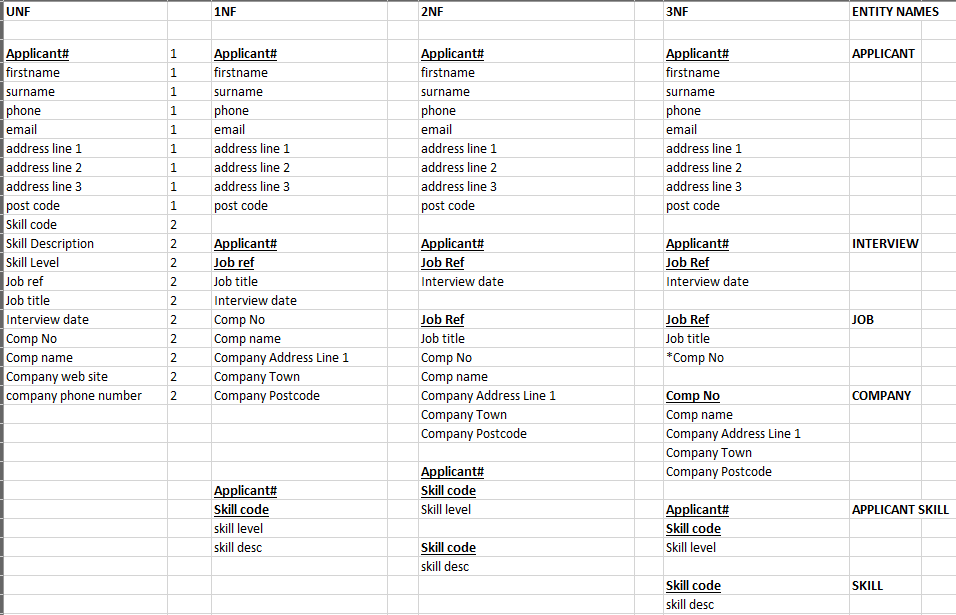
User: any?

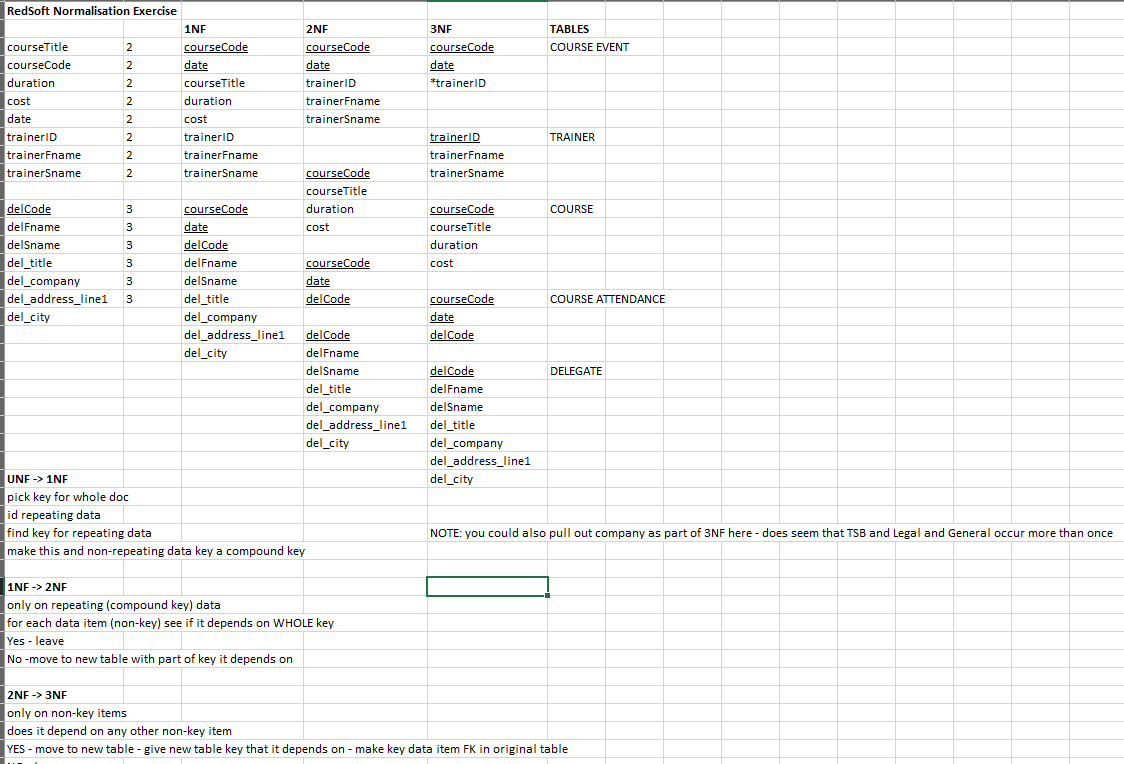
Dept: any?

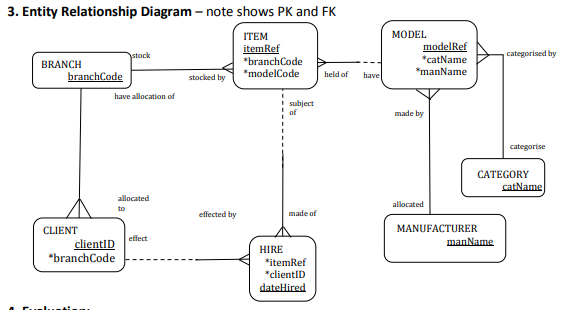
Booking: limits on startTime, format for duration

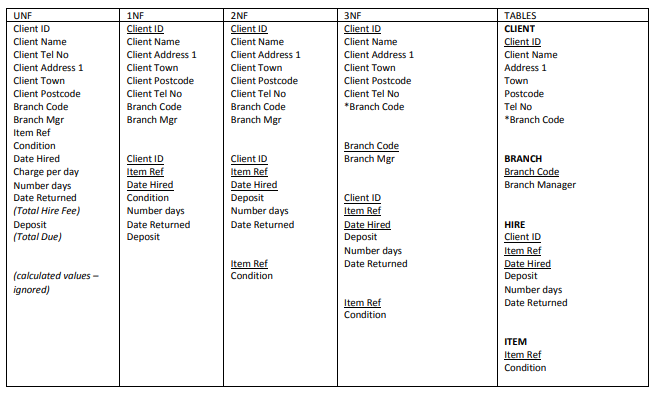
**Draw (on paper) an ER model that represents this schema:**











*The SELECT statement provides the means of data retrieval in SQL. The full syntax is rather complex, reflecting the power of the statement. A simplified form of the syntax is shown below:*

SELECT *select\_list* FROM *table\_source*

[ WHERE *search\_condition* ]

[ GROUP BY *group\_by\_expression* ] [ HAVING *search\_condition* ]

[ ORDER BY *order\_expression* [ ASC | DESC ] ]

*Key words in CAPITALS*

**Eliminating duplicate rows:**

* The result of a SELECT may contain duplicate rows

SELECT author FROM tBook

*returns 89 rows with duplicates where an author has several books In the library*

SELECT DISTINCT author FROM tBook

*Returns 50 unique rows*

Conditions:

* The WHERE clause in a SELECT statement is used to specify a condition
* The statement will return rows for which the condition is TRUE and ignore the rest
* The condition itself is of arbitrary complexity

**Comparison operators:**

SELECT \* FROM tLoan WHERE memberID <= 1002

SELECT title FROM tBook WHERE author >= 'V'

*Picks out the books by authors with names alphabetically from V onwards*

SELECT \* FROM Loans WHERE dueBack > '2004-03-21'

*Picks out the books that are due back after the specified date*

***Adjust these queries to make them work*** *(and give a response) in your database.*

*Use Object Explorer to check table and field names.*

* The BETWEEN operator provides a shorthand for inclusive range comparisons

SELECT surname FROM tMember

WHERE memberNo BETWEEN 1010 AND 1700

*Is the same as:*

SELECT surname FROM tMember

WHERE memberNo >= 1010 AND memberNo <= 1700

**Classification of physical storage media:**

Speed with which data can be accessed

Cost per unit of data

Reliability

data loss on power failure or system crash

physical failure of the storage device

Can differentiate storage into:

**volatile storage:** loses contents when power is switched off

**non-volatile storage**:

Contents persist even when power is switched off.

Includes secondary and tertiary storage, as well as batter-backed up main-memory.

**Cache** – fastest and most costly form of storage; volatile; managed by the computer system hardware.

**Main memory**:

fast access (10s to 100s of nanoseconds; 1 nanosecond = 10–9 seconds)

generally too small (or too expensive) to store the entire database – but changing all the time (see [this](https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/))

capacities to many Gigabytes widely used currently

Capacities have gone up and per-byte costs have decreased steadily and rapidly (roughly factor of 2 every 2 to 3 years)

**Volatile** — contents of main memory are usually lost if a power failure or system crash occurs.

**Flash memory**:

Data survives power failure

Data can be written at a location only once, but location can be erased and written to again

Can support only a limited number of write/erase cycles.

Erasing of memory has to be done to an entire bank of memory

Reads are roughly as fast as main memory

But writes are slow (few microseconds), erase is slower

Cost per unit of storage roughly similar to main memory

Widely used in embedded devices such as digital cameras

also known as EEPROM (Electrically Erasable Programmable Read-Only Memory)

A database file is partitioned into fixed-length storage units called **blocks**. Blocks are units of both storage allocation and data transfer.

Database system seeks to minimize the number of block transfers between the disk and memory. We can reduce the number of disk accesses by keeping as many blocks as possible in main memory.

**Buffer** – portion of main memory available to store copies of disk blocks.

**Buffer manager** – subsystem responsible for allocating buffer space in main memory.

**Heap** – a record can be placed anywhere in the file where there is space

**Sequential** – store records in sequential order, based on the value of the search key of each record

**Hashing** – a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed

**Indexed Sequential** – Combines Indexed and Sequential file organization

Records of each relation may be stored in a separate file. In a **clustering file organization**  records of several different relations can be stored in the same file

Write an SQL statement to produce first name and last name of all customers who are currently renting a car:

SELECT FNAME, LNAME FROM CUSTOMER C JOIN RENTAL R ON

C.DRLICENCE=R.DRLICENCE

Write an SQL statement to show make and colour of all cars currently rented out:

SELECT MAKE, COLOUR FROM CAR C JOIN RENTAL R

ON C.CARID=R.CARID AND TO\_DATE IS NULL

Write an SQL statement to produce the make of cars that have never been rented:

SELECT DISTINCTMAKE FROM CAR WHERE MAKE NOT IN

(SELECT DISTINCT MAKE FROM CAR C JOIN RENTAL R ON C.CARID=R.CARID)

Write an SQL statement to produce the rental price and rental\_id for each (completed) rental:

SELECT SUM(CHARGE), RENTALID FROM

(SELECT RENTALID, AMOUNT\*COUNT FROM RENT\_RATE JOIN RENTAL ON

RENT\_RATE.DURATION=RENTAL.DURATION AS T (RENTALID, CHARGE)

GROUP BY RENTALID

The ACID properties are main characteristics of a database transaction and every modern DBMS enforces those properties. Discuss what effect on a database there would be if DBMS didn’t implement only one of the ACID properties. Support your answer with examples:

understanding of Atomicity, Consistency, Isolation and Durability

An example that without Atomicity, a whole transaction wouldn’t be completed

An example that without Consistency, a database wouldn’t be in a valid state

An example that without Isolation transactions would interfere with each other and violate the Consistency

An example that without Durability changes to database would be lost

Compare and contrast Pessimistic and Optimistic concurrency control mechanisms for managing transactions in a multi user DBMS:

address 2 phase locking and time stamping mechanism as pessimistic approaches to concurrency control, providing examples. In discussing Optimistic approach, they should show understanding that transactions are free to execute their read/write operations and that the validation for consistency is performed afterwards.

come up with a comprehensive disaster recovery plan for the university’s systems. Summarize the main issues that you would include in your written plan:

Locations – where should backups be stored to facilitate quick recovery? Examine or propose options such as Cloud or mirror sites

Personnel - who are the people responsible and what are their roles?

Recovery procedures and scripts for all system software, applications and data: These should be detailed step-by-step procedures required for recovery.

In a university library database the following schema is used:

Book (ISBN, title, authors, category)

BookCopy(ISBN, copynumber, DatePurchased)

Classification(ClassID, classname)

Member (MemberID, name, address, status)

Loan (MemberID, copynumber, dateborrowed, loanLength, datereturned)

Based on the following requirements, write SQL statements to grant privileges to users to access objects:

1. Suppose that user Anna is the owner of the table BOOK and wishes to grant privileges to John and Maria to SELECT queries on the BOOK table and to authorize someone else to do the same.
2. Assume that John creates a view named NewBOOK and grants it to Andy and Maria where they can execute only SELECT statements on the view and are not authorised to pass the given privilege to others.

Referring to the previous scenario, explain the implication(s) on the privilege access of all the users if user Anna revokes the SELECT privilege on BOOK from John.

1. GRANT SELECT ON BOOK TO John, Maria WITH GRANT OPTION
2. GRANT SELECT ON NewBOOK TO Andy, Maria

Revoking John’s privilege by Anna will result in John losing the SELECT privilege on BOOK.

The NewBOOK view created by John will be dropped as it is derived from the table BOOK, to which John loses the SELECT privilege.

Since Andy and Maria received the SELECT privilege on NewBOOK from John, they also lose their privileges.

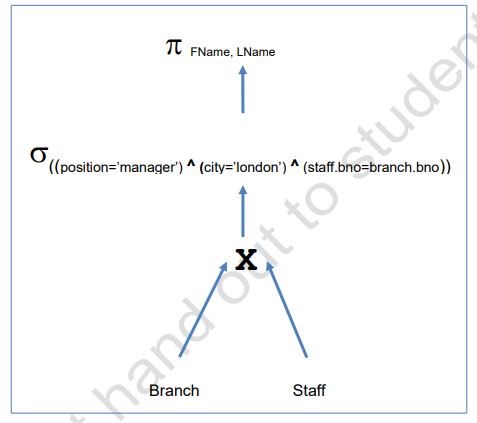
However, Maria still has the SELECT privilege on BOOK which was granted to her by Anna.

**A database consists of two tables:**

Branch(Bno, Street, City, PostCode, Tel\_No)

Staff(Sno, FName, LName, Adress, Tel\_No, Position, Sex, DOB, Salary, NI\_No, Bno)

When an Sql query is submitted to DBMS for execution, the following Relational Algebra tree is generated:



For the given Relational Algebra tree, write the corresponding Relational Algebra expression:

π FName, Name(σ(position=’manager’ ^ city=’london’ ^ staff.bno=branch.bno) (BranchXStaff))