SendNodesMarkII Bookwork Phase 2

5.1)
Domain - All unique values permitted for an attribute.
Attribute - An attribute refers to a database component or a characteristic of a class.
N-tuple - A tuple exists as a row or one record in a database. A N-tuple is a finite set of tuples in a given database. N-tuple is an ordered list of n values.
Relation schema - A set of sentences or formulas imposed on a database.
Relation state - A relation state, or a relation, is the r of the relation schema as a set of tuples.
Degree of a relation - The degree of a relation is the number of participating entity types.
Relational database schema - A set of relation schemas S and a set of integrity constraints.
Relational database state - DB of S is a set of relation states DB such that each r is a state of R and such that the r relation states satisfy the integrity constraints specified in IC.

5.2)
Tuples in a relation don't have any order in a set. Tuples in a relation are basically logical definitions which attempt to represent facts. Therefore a relation is not sensitive to the ordering of tuples. A row is the same thing as a tuple.

5.5)
We designate one of the candidate keys of a relation to be the primary key due to common convention. The primary key will be used to identify tuples in a relationship. The primary key also makes it easier to handle the database when one key is used rather than all the candidates.

5.9)
A foreign key is a set of attributes in a relation schema R1 that references the primary key attributes of the referenced relation R2. The foreign key concept is used for a referential integrity constraint, which is needed to maintain the consistency among tuples in the two relations.

5.12)
a) We begin by checking the availability of seats on the plane using the Number_of_available_seats attribute in the LEG_INSTANCE entity. We choose the specified flight number and date of flight, then check if Number_of_available_seats is more than 0. We then reserve the seat and insert the values into the SEAT_RESERVATION entity attributes.

b) We need to check if the seat number the customer is asking for is available at the specified date and time. We should also check whether the Number_of_available_seats for the flight is more than 0.

c) The Seat_number attribute has an entity integrity constraint. The LEG_INSTANCE entity does not have an entity or referential integrity constraint.

d) Referential Integrity Constraints Hold - Foreign Keys that reference keys

- FLIGHT_LEG(Flight_number) references FLIGHT(Flight_number)
- FLIGHT_LEG(Departure_airport_code) references AIRPORT(Airport_code)
- FLIGHT_LEG(Arrival_airport_code) references AIRPORT(Airport_code)
- FARE(Flight_number) references FLIGHT(Flight_number)
- SEAT_RESERVATION(Flight_number) references Flight(Flight_number)
- SEAT_RESERVATION(Flight_number) references LEG_INSTANCE(Flight_number)
- SEAT_RESERVATION(Leg_number) references LEG_INSTANCE(Leg_number)
- SEAT_RESERVATION(Date) references LEG_INSTANCE(Date)
- LEG_INSTANCE(Flight_number) references FLIGHT_LEG(Flight_number)
- LEG_INSTANCE(Leg_number) references FLIGHT_LEG(Leg_number)
- LEG_INSTANCE(Departure_airport_code) references AIRPORT(Airport_code)
- LEG_INSTANCE(Arrival_airport_code) references AIRPORT(Airport_code)
- LEG_INSTANCE(Airplane_id) references AIRPLANE(Airplane_id)
- LEG_INSTANCE(Flight_number) references FLIGHT(Flight_number)
- CAN_LAND(Airplane_type_name) references AIRPLANE_TYPE(Airplane_type_name
- CAN_LAND(Airport_code) references AIRPORT(Airport_code)

6.7)

CREATE TABLE BOOK

FOREIGN KEY(Publisher_name) REFERENCES PUBLISHER(Name)

ON DELETE REJECT: can not delete Name because it is referenced by Publisher_name

ON UPDATE CASCADE: if the PUBLISHER(Name) is updated then BOOK(Publisher_name) is propagated to it.

CREATE TABLE BOOK_AUTHORS

FOREIGN KEY(Book_id) REFERENCES BOOK(Book_id)

ON DELETE CASCADE: If the BOOK(Book_id) is deleted then BOOK_AUTHORS(Book_id) is propagated to it.

ON UPDATE CASCADE: If the BOOK(Book_id) is updated then BOOK_AUTHORS(Book_id) is propagated to it.

CREATE BOOK_COPIES

FOREIGN KEY(Book_id) REFERENCES BOOK(Book_id)

ON DELETE CASCADE: If the BOOK(Book_id) is deleted then BOOK_COPIES(Book_id) is propagated to it.

ON UPDATE CASCADE: If the BOOK(Book_id) is updated then BOOK_COPIES(Book_id) is propagated to it.

FOREIGN KEY(Branch_id) REFERENCES LIBRARY_BRANCH(Branch_id)

ON DELETE CASCADE: If the LIBRARY_BRANCH(Branch_id) is deleted then BOOK_COPIES(Branch_id) is propagated to it.

ON UPDATE CASCADE: If the LIBRARY_BRANCH(Branch_id) is updated then BOOK_COPIES(Branch_id) is propagated to it.

CREATE BOOK_LOANS
FOREIGN KEY(Book_id) REFERENCES BOOK(Book_id)
ON DELETE CASCADE: If the BOOK(Book_id) is deleted then BOOK_LOANS(Book_id) is propagated to it.
ON UPDATE CASCADE: If the BOOK(Book_id) is updated then BOOK_LOANS(Book_id) is propagated to it.

FOREIGN KEY(Brach_id) REFERENCES LIBRARY_BRANCH(Branch_id)
ON DELETE CASCADE: If the LIBRARY_BRANCH(Branch_id) is deleted then BOOK_LOANS(Branch_id) is propagated to it.
ON UPDATE CASCADE: If the LIBRARY_BRANCH(Branch_id) is updated then BOOK_LOANS(Branch_id) is propagated to it.

FOREIGN KEY(Card-no) REFERENCES BORROWER(Card_no)
ON DELETE CASCADE: If the BORROWER(Card_no) is deleted then BOOK_LOANS(Card_no) is propagated to it.
ON UPDATE CASCADE: If the BORROWER(Card_no) is updated then BOOK_LOANS(Card_no) is propagated to it.

7.1)
SELECT <Attribute and Function List> (Required)
FROM: Table List> (Required)
WHERE: <Condition> (Optional)
GROUP BY: <Grouping Attributes> (Optional)
HAVING: <Group Condition> (Optional)
ORDER BY: <Attribute List> (Optional)

7.3

In SQL, NULL is treated as an UNKNOWN value. For comparison operators in SQL, NULL can be compared using IS or IS NOT operator. SQL treats each NULL as distinct value, so =,<,> cannot be used for comparison. In general, NULL values are discarded when aggregate functions are applied to a particular column. In case of grouping attribute, there will be a separate group created for all tuples with a NULL value in the grouping attribute.

8.1

· SELECT- Used to obtain a subset of tuples of a relation bases on a condition. The symbol used to denote SELECT is sigma
· PROJECT- used to obtain certain attributes/columns of a relation. The attributed to be retrieved must be specified as a list separated by commas. The symbol us Pi.

· THETA JOIN- used to combine related tuples from two relations and outputs as a single tuple.
· EQUI JOIN- used to combine all tuples of relations R1 and R2 that satisfy the condition. The operator is =.
· NATURAL JOIN= Same as EQUI JOIN except that the join attributes of R2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.
· UNION- Produces a relation that includes all the tuples in R1 or R2 or both R1 and R2; R1 and R2 must be union compatible.
· INTERSECTION- Produces a relation that includes all the tuples in both R1 and R2; R1 and R2 must be union compatible.
· DIFFERENCE- Produces a relation that includes all the tuples in R1 that are not in R2; R1 and R2 must be union compatible.
· CARTESIAN PRODUCT- Produces a relation that has the attributes of R1 and R2 and includes as tuples all possible combinations of tuples from R1 and R2.
· DIVISION- Produces a relation R(X) that includes all tuples t[X] in R1(Z) that appear in R1 in combination with every tuple from R2(Y), where Z=X U Y.

9.1

A

Correspondence

ER Model	**Relational Model**
Entity type	Entity relation
1:1 or 1:N relationship type	Foreign key
M:N relationship type	Relationship relation and two foreign keys.
n-ary relationship type	Relationship relation and n foreign keys.
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain

Key attribute	Primary (or secondary) key

Method to map from ER model Relational model

1. Mapping of all strong Entities.

· Map all strong entities into tables. Create a separate relation for each strong entity including all simple attributes in the ER diagram and choose key attribute of ER diagram as primary key of relation.
· Assume an entity type T in the ER model E, create a relation R including all simple attributes of T, also choose unique attribute as a primary key of relation R.

2. Mapping of weak Entities.

· Map all weak entities into tables. Create a separate relation for each weak entity including all simple attributes. Include all primary keys of the relations to which weak entity is related as foreign key, to establish connection among the relations.

· Weak entity does not have its own candidate key. Here candidate key of the relation R is composed of the primary key(s) of the participating entity(s) and the partial key of the weak entity.

3. Binary 1:1 Mapping.

· For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

1.Foreign Key (2 relations) approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
2.Merged relation (1 relation) option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
3.Cross-referenceor relationship relation (3 relations) option: The third alternative is to set up a third relation R for the purpose of
cross-referencing the primary keys of the two relations S and T representing the entity types.

4. Binary 1: N Mapping.

· Identify all 1: N relationships in ER diagram. For each binary 1: N relationship in relation R, the primary key present on the 1-side of the relationship becomes a foreign key on the N-side relation.

· Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.

· Include any simple attributes of the 1:N relation type as attributes of S.

5. Binary M: N Mapping.

· For each regular binary M:N relationship type R, create a new relation S to represent R. This is a relationship relation.

· Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

· Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

6. Mapping of Multivalued attributes.

· Create a new relation R, corresponding to each multivalued attribute A present in the ER diagram. All simple attributes corresponding to A, would be present in relation R.

· The relation will comprise of primary key attribute K, such that the attribute K belongs to the relation representing the relationship type containing A as a multivalued attribute. The primary key of R would be the combination of A and K.

7. Mapping of N-ary relationship.

· For each n-ary relationship type R, where n>2, create a new relationship S to represent R.

· Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

· Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

B.

EER to Relational model

We can use the 8 steps in map EER to Relational model. In addition to that, there are some steps.

Mapping specialization or generalization.

· A number of subclasses, that constitutes a specialization, can be mapped into relational schema using several options.

First option is to map the whole specialization into a single table. Second option is to map it into multiple tables. In each option, variations may occur that depends upon the constraints on the specialization or generalization.

· Each specialization containing m subclasses and generalized super class C, having the primary key k and the attributes are converted into relational schemas using one of the following options:

Option 8a: Multiple relations-superclass and subclasses.

· Create a relation table for the superclass that includes all the components (attributes, domains, constraints). For each subclass, have another table with the remaining attributes with a matching primary key.

Option 8b: Multiple relations-subclasses relations only.

· Create a different relational table for each subclass. The participation must be total, such that the primary key of the superclass is not duplicated. And no case of an independent superclass exists.

Option 8c: Single relation with one type attribute.

· The subclass has only one attribute different than the superclass. Create a relation table for the superclass that includes all the components (attributes, domains, constraints) of all the subclasses. The attribute can be checked to find the subclass type.

Option 8d: Single relation with multiple type attribute.

· Create a relational table for the superclass that includes all the components (attributes, domains, constraints) of all the subclasses. Include an additional Boolean flag for each kind of subclass in the model. This flag is set true for the subclass.