Nathan Gong ng2695

A. Module I

1.1

1. The conceptual layer focuses on the structure of the data and not the implementation. It is intended for stakeholders and data analysts. The modeler specifies entity names and relationships.
The logical layer includes more detailed data elements and relationships but is independent of any specific DBMS. It is intended for database designers and developers. The modeler specifies entity names and relationships, attributes, and primary and foreign keys.
The physical layer represents the implementation of a database on a specific platform. It is intended for database administrators. The modeler specifies primary and foreign keys, table names, and column names and data types.

2. The degree of a relationship is the number of entity sets involved, and the cardinality of a relationship is the number of entity sets to which another entity can be associated with.

3. An alternate key is a unique key that is not the primary key, and a surrogate key is an artificial system-generated identifier.

1.2

4. In the latter schema with ID underlined, ID is the primary key, while it is not in the former schema without the underline.

5. Codd's rule 4 states that a system must provide a dynamic catalog that is accessible to authorized users. SQL implements this rule by having a data dictionary that can be accessed using:

select <column> from <information_schema> where table_schema = " ";

1.3

6. One example of such an SQL statement is using "GROUP BY".

7. Three examples are primary key, foreign key, and not null constraints.

8. Cascading actions refer to automatic updates to the child table when the parent table is updated. This helps maintain data consistency across tables.

2. Practical

2.1

9.

```
create table SalesRep (
    ID          text    not null,
    last_name   text,
    first_name  text
    primary key (ID));

create table Customer (
    ID          text    not null,
    last_name   text,
    first_name  text,
    primary key (ID));

create table CustomerSalesRep (
    customer_ID     text    not null,
    sales_rep_ID    text    not null,
    start_date      text,
    end_date        text,
    primary key (customer_ID, sales_rep_ID),
    foreign key (customer_ID) references Customer (ID),
    foreign key (sales_rep_ID) references SalesRep (ID));

create table Comment (
    ID          text    primary key not null,
    customer_ID text    not null,
    sales_rep_ID text   not null,
    comment_value text,
    foreign key (customer_ID) references Customer (ID),
    foreign key (sales_rep_ID) references SalesRep (ID));
```

2.2

10.

alpha	beta	charlie	delta
4	d	f	200
5	d	b	200
null	f	null	400
null	g	null	120

2.3

11. a)

- Make name_basics. nconst a primary key as shown in the diagram
- Separate name_basics. primaryName into two fields name_basics. firstName and name_basics. lastName to ensure atomicity
- Replace name_basics_knownForTitles with another table that contains a foreign key reference to name_basics. nconst and an additional field to hold the tconst values
- Make name_basics_nconst not null since it is a primary key
- The title_basics entity in the diagram has field nconst but it should be tconst.
- Make title_basics. tconst not null since it is a primary key
- Make title_principals. nconst a primary key, foreign key, and not null as shown in the diagram
- Make title_principals. tconst a primary key, foreign key, and not null as shown in the diagram

11. b)

```
create table if not exists w4111_f24_final.name_basics (
    nconst text primary key not null,
    firstName text null,
    lastName text null);
create table if not exists w4111_f24_final.known_for_titles (
    nconst text primary key not null,
    tconst text primary key not null,
    foreign key (nconst) references name_basics (nconst),
    foreign key (tconst) references title_basics (tconst));
create table if not exists w4111_f24_final.title_basics (
    tconst text primary key not null,
    primary_title text null);
create table if not exists w4111_f24_final.title_principals (
    nconst text primary key not null,
    tconst text primary key not null,
    foreign key (nconst) references name_basics (nconst),
    foreign key (tconst) references title_basics (tconst));
```

12. a)

```
with one as (
    select * from orders join orderdetails using (orderNumber)),
two as (
    select * from customers join one using (customerNumber)),
three as (
    select country, SUM(quantityOrdered * priceEach) as total_revenue
    from two group by country order by country where
    total_revenue >= 250000),
select country, total_revenue,
    case
        when total_revenue >= 1000000 then 'Tier 1'
        when total_revenue < 1000000 and total_revenue >= 400000
            then 'Tier 2'
        else 'Tier 3'
    end as revenue_tier
    from three;
```

12. b) The first with clause joins orders with their details based on orderNumber. The second with clause joins this with customers based on customerNumber. The third with clause selects specific fields, performs an aggregation to calculate total_revenue, groups by country, sorts by country, and limits the results based on total_revenue >= 250000. Finally, these two fields are selected and used to calculate revenue_tier based on the specified criteria.

B. Module II

13. Logical block addressing specifies data location based on a single unique logical address. Cylinder-head-sector block addressing specifies data based on the physical sector of the track of the cylinder of the disk that it is stored in.

14. The 4 primary approaches are heaped/unordered, sequential/ordered, hashed, and clustered.

15. Columnar storage is useful for aggregation and filtering queries.

16. LRU is extremely poor in scenarios in which only a small subset of the data is accessed very frequently.

17. A clustering index has a search-key that specifies the sequential order of the file. A table can have one clustering index.

18. An advantage is that hash indexes are good for exact match queries, and a disadvantage is that hash indexes are bad for range based queries. B+ trees resolve this by being a sorted data structure that allows for efficient range and non-equality queries.

19. select * from instructor join department using (department_name);

20. select * from (
    select * from instructor where dept_name = 'Comp. Sci.'
    join
    select * from teaches where teaches.year = 2018);

21. Two evils of redundancy are data inconsistency and update/insert/delete anomalies.

22. A fractional dependency is a relationship between two attributes in a database that satisfies a real-world constraint. Decomposition consists of breaking down a relation into smaller relations to preserve information while remaining in normal form.

23. Steal/no steal refers to whether the database is allowed to write modified pages before a transaction commits. Force/no force refers to whether the database is required to force changes to disk before a transaction commits. The write ahead logging technique uses steal/no force.

24. 2PL has a growing phase where transactions may obtain/not release locks and a shrinking phase where transactions may release/not obtain locks. Strict 2PL states that transactions must hold all exclusive locks until a commit/abort. The benefit of strict 2PL is that it guarantees serializability, prevents cascading aborts, and ensures recoverability.

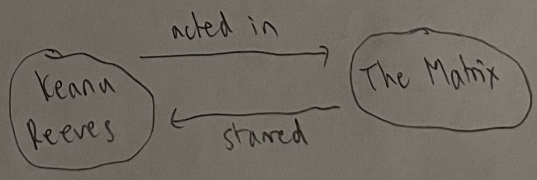
25. A deadlock occurs when multiple transactions are waiting for each other to release locks. This can be resolved using total or partial rollback.

C. Module III

26. Example MongoDB document:

```
{
    "_id" : 1,
    "title" : "The Matrix",
    "cast" : [
        {
            "name" : "Keanu Reeves"
        }
    ]
}
```

Example Neo4j drawing:



27.
```
Match (p:Actor)-[:Acted_In]->(m:Movie)<-[:Acted_In]-(q:Actor)
Where p.name = 'Tom Hanks'
Return distinct q.name
```

28.
```
db.orders.aggregate([
  {
    $unwind: "$orderlines"
  },
  {
    $project: {
      orderNumber: 1,
      orderDate: 1,
      status: 1,
      orderlines.productCode: 1,
      orderlines.quantityOrdered: 1,
      orderlines.priceEach: 1
    }
  }
])
```

0. Module IV

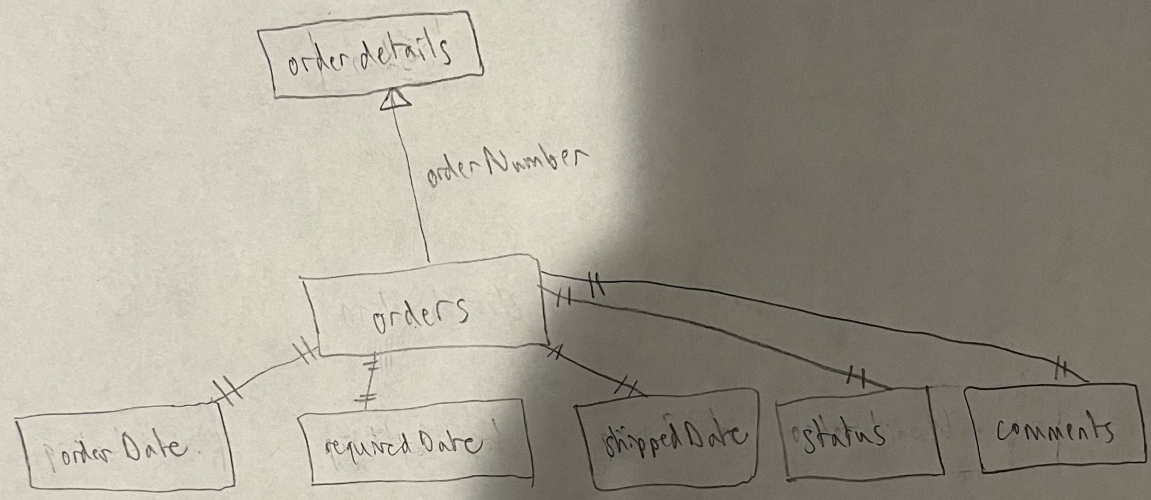
29. Algebraic operations in Spark are used to convert reliable distributed datasets and dataframes to new RDDs and dataframes. These offer much more flexibility than the operations in MapReduce, which just consist of two main operations: map and reduce.

30. Fact tables store quantitative data/metrics. Example: orders table

Dimension tables describe the context of the data in fact tables.
Example: order details table

Drill down and roll up allow for summarizing data at higher/lower detail.
Example: order details.quantityOrdered → order.orderDate

31.



32. volume, variety, velocity, veracity, value