CMPE 180B

Media Streaming Services

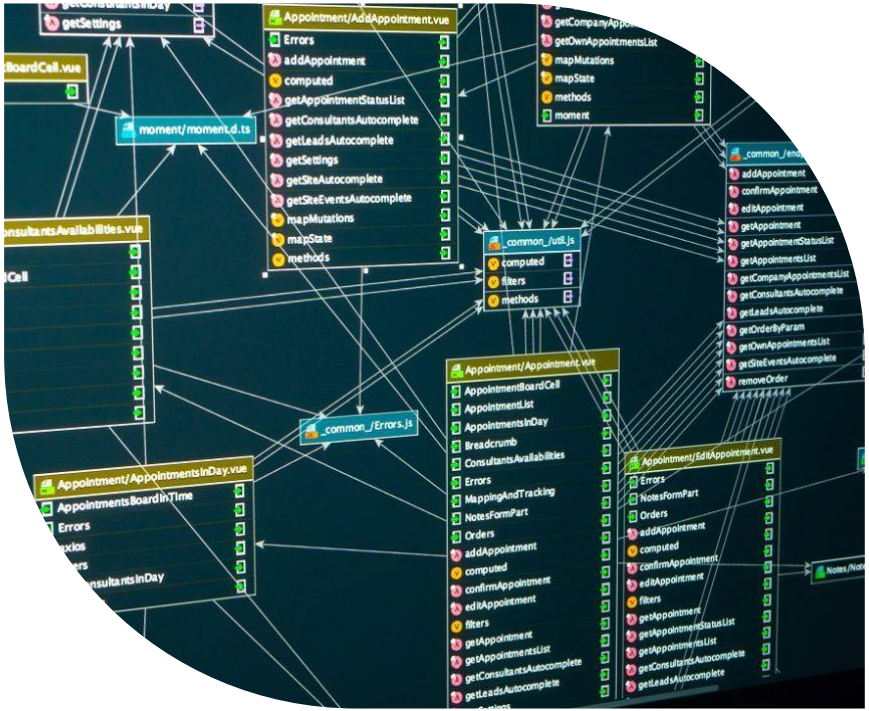
Database Implementation

Query Masters

- *Manish Maryala*
- *Jainam Chhatbar*
- *Venkat Sai Kedari Nath Gandham*
- *Kalyani Chitre*

1. Project Overview
2. Database Design and ER Diagram
3. Queries and Optimization
4. Transaction and Concurrency Handling
5. Backup and Recovery Plan

1. **Project Overview**
2. Database Design and ER Diagram
3. Queries and Optimization
4. Transaction and Concurrency Handling
5. Backup and Recovery Plan

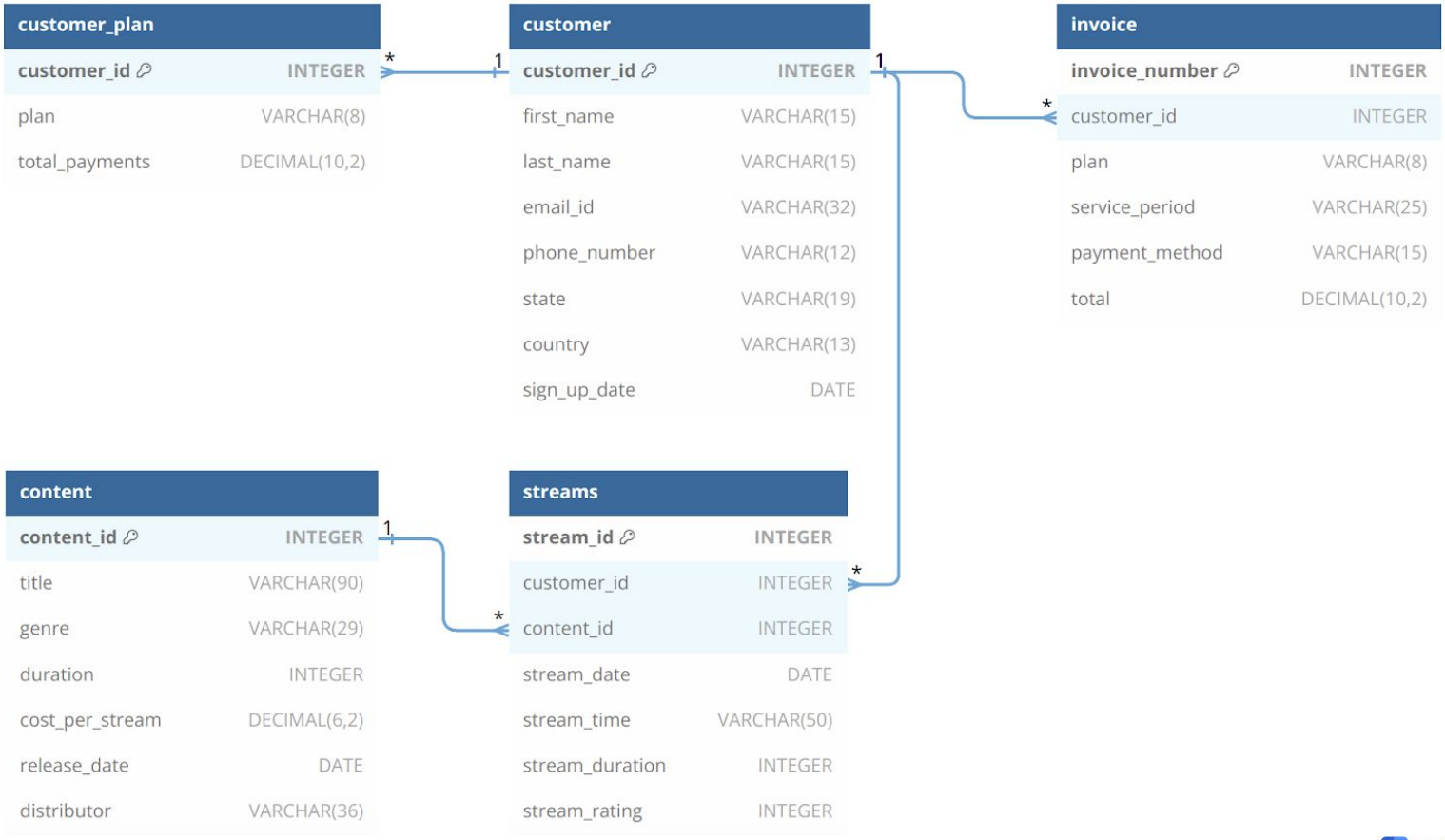


A dynamic media streaming database that seamlessly organizes customer, content, streaming, and invoice data, supporting essential business queries for strategic insights.

1. Project Overview
2. **Database Design and ER Diagram**
3. Queries and Optimization
4. Transaction and Concurrency Handling
5. Backup and Recovery Plan

Database Design

- Customer: Stores detailed information about streaming service users.
- Streams: Tracks individual streaming sessions, including content, customer details, and viewing times.
- Content: Contains metadata for all streaming content, such as title, genre, and cost.
- Invoice: Records customer subscriptions, payments, and service plans.
- Customer Plan: Manages subscription plans.


customer_plan
customer_id INTEGER
plan VARCHAR(8)
total_payments DECIMAL(10,2)
customer
customer_id INTEGER
first_name VARCHAR(15)
last_name VARCHAR(15)
email_id VARCHAR(32)
phone_number VARCHAR(12)
state VARCHAR(19)
country VARCHAR(13)
sign_up_date DATE
invoice
invoice_number INTEGER
customer_id INTEGER
plan VARCHAR(8)
service_period VARCHAR(25)
payment_method VARCHAR(15)
total DECIMAL(10,2)
content
content_id INTEGER
title VARCHAR(90)
genre VARCHAR(29)
duration INTEGER
cost_per_stream DECIMAL(6,2)
release_date DATE
distributor VARCHAR(36)
streams
stream_id INTEGER
customer_id INTEGER
content_id INTEGER
stream_date DATE
stream_time VARCHAR(50)
stream_duration INTEGER
stream_rating INTEGER
1
*
dbdiagram

1. Project Overview
2. Database Design and ER Diagram
3. **Queries and Optimization**
4. Transaction and Concurrency Handling
5. Backup and Recovery Plan

```sql
SELECT s.customer_id,
       c.first_name,
       c.last_name,
       COUNT(s.stream_id) AS total_streams,
       AVG(s.stream_rating) AS avg_rating
FROM streams s
JOIN customer c ON s.customer_id = c.customer_id
GROUP BY s.customer_id, c.first_name, c.last_name
ORDER BY total_streams DESC;
```

```sql
SELECT co.title,
       COUNT(s.stream_id) AS total_views
FROM streams s
JOIN content c ON s.content_id = c.content_id
GROUP BY co.title
ORDER BY total_views DESC
LIMIT 10;
```

```sql
SELECT co.title,
       SUM(i.total) AS total_revenue,
       (co.cost_per_stream * COUNT(s.stream_id)) AS total_cost
FROM content co
JOIN streams s ON co.content_id = s.content_id
JOIN invoice i ON cp.customer_id = i.customer_id
GROUP BY co.title, co.cost_per_stream
HAVING total_revenue > total_cost
ORDER BY total_revenue DESC;
```

How do we optimize it?

- Indices: additionally, composite indices
- Query caching
- Avoiding `SELECT *` queries
- Using GROUP BY and ORDER BY
- Materialized view

1. Project Overview
2. Database Design and ER Diagram
3. Queries and Optimization
4. **Transaction and Concurrency Handling**
5. Backup and Recovery Plan

Transactions and Concurrency

- This step focuses on maintaining ACID properties by designing transactions that maintain database integrity through atomicity, consistency, isolation, and durability.
- Concurrency issues like dirty reads and lost updates are resolved using appropriate isolation levels such as READ COMMITTED and REPEATABLE READ.
- Ensures reliable and consistent database operations even with multiple users accessing the system simultaneously.

Project demonstration

1. Project Overview
2. Database Design and ER Diagram
3. Queries and Optimization
4. Transaction and Concurrency Handling
5. **Backup and Recovery Plan**

Backup Plan

Automated scheduled backups

Weekly full backups –
Incremental daily/hourly backups –
Differential backups –

Tools

cron –
mysqldump –
mysqlpump –

Recovery Plan

Full System Failure

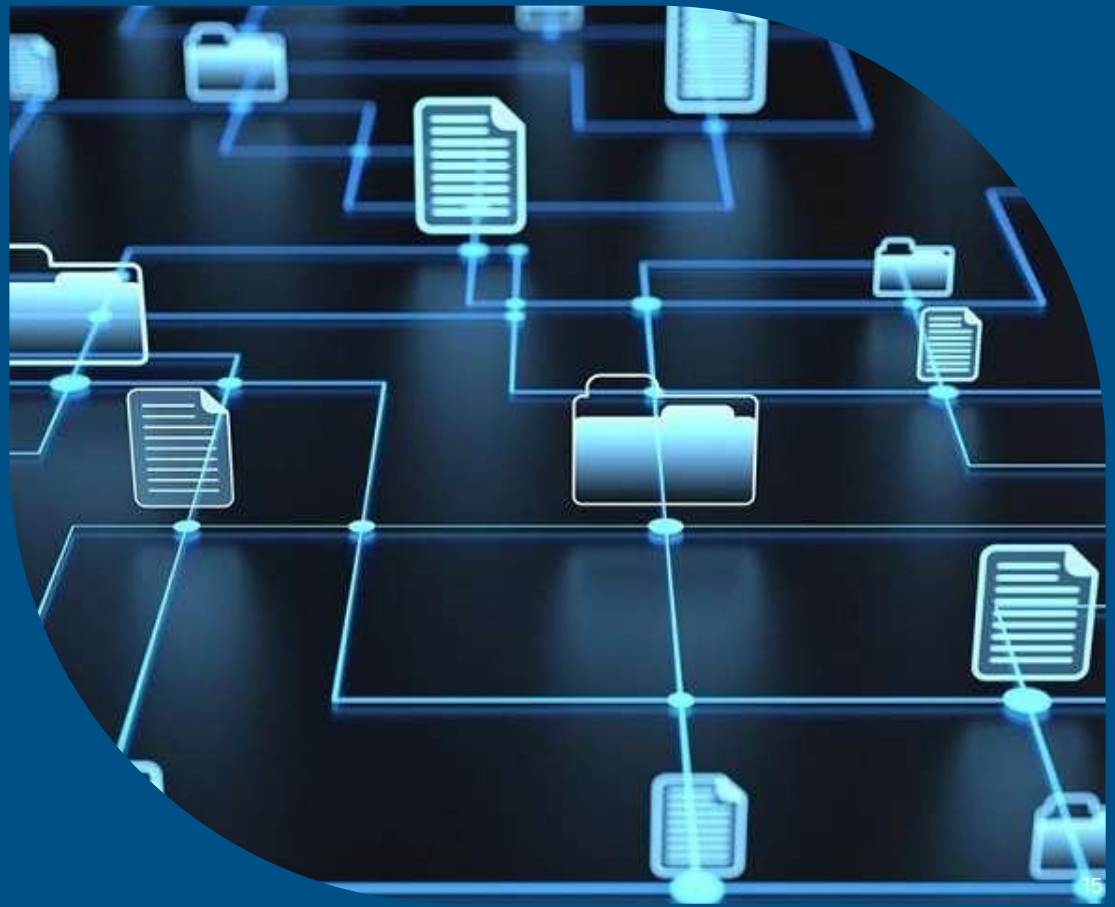
- Restore most recent full backup
- Apply incremental backups or binary logs to recover the latest changes

In Case of Data Corruption/Accidental Deletion

- Identify the affected database or table.
- Restore the specific backup and replay binary logs for minimal data loss.

Conclusion

This comprehensive schema optimizes scalability, minimizes redundancy, and empowers efficient data retrieval for enhanced operational and analytical decision-making.




Thank
you