1 2 9 0

UNIVERSIDADE D
COIMBRA

Faculty of Sciences and
Technology

HOSPITAL MANAGEMENT SYSTEM

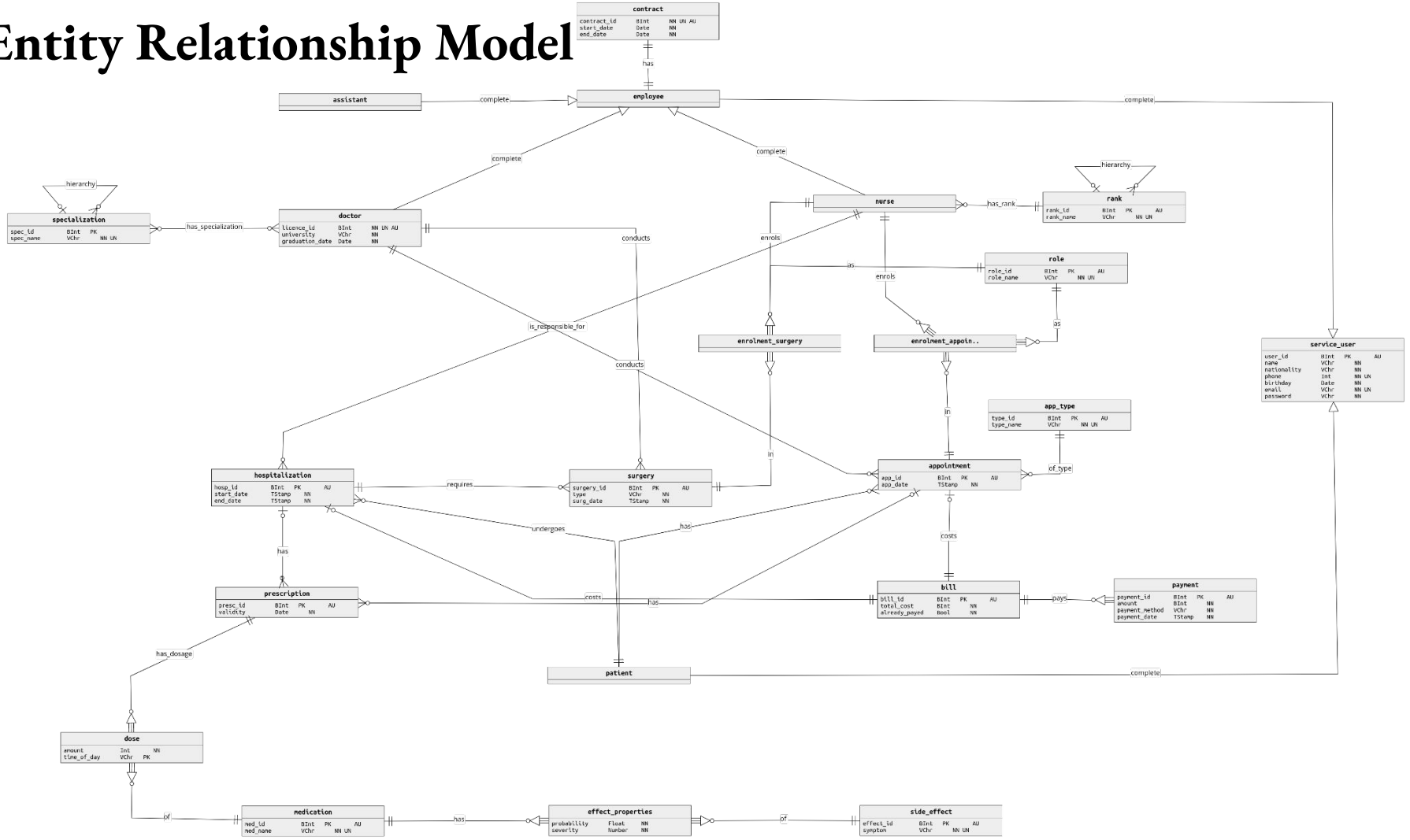
DATABASES - BACHELOR'S DEGREE IN INFORMATICS ENGINEERING

Overview

As this presentation is only 3 minutes long, it will be focused on the less obvious aspects that would probability not be noticed in the demo

- Entity Relationship Model
- Concurrency Conflicts
- Security
- Database Tuning

Entity Relationship Model



Concurrency Conflicts

- Update lock when scheduling services (availability check)
- Same order checking to avoid deadlocks
- Isolation to avoid dirty reads on the GET method endpoints
- Row-Level lock when executing payments

Concurrency Conflicts - Examples

top3.sql

```sql
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

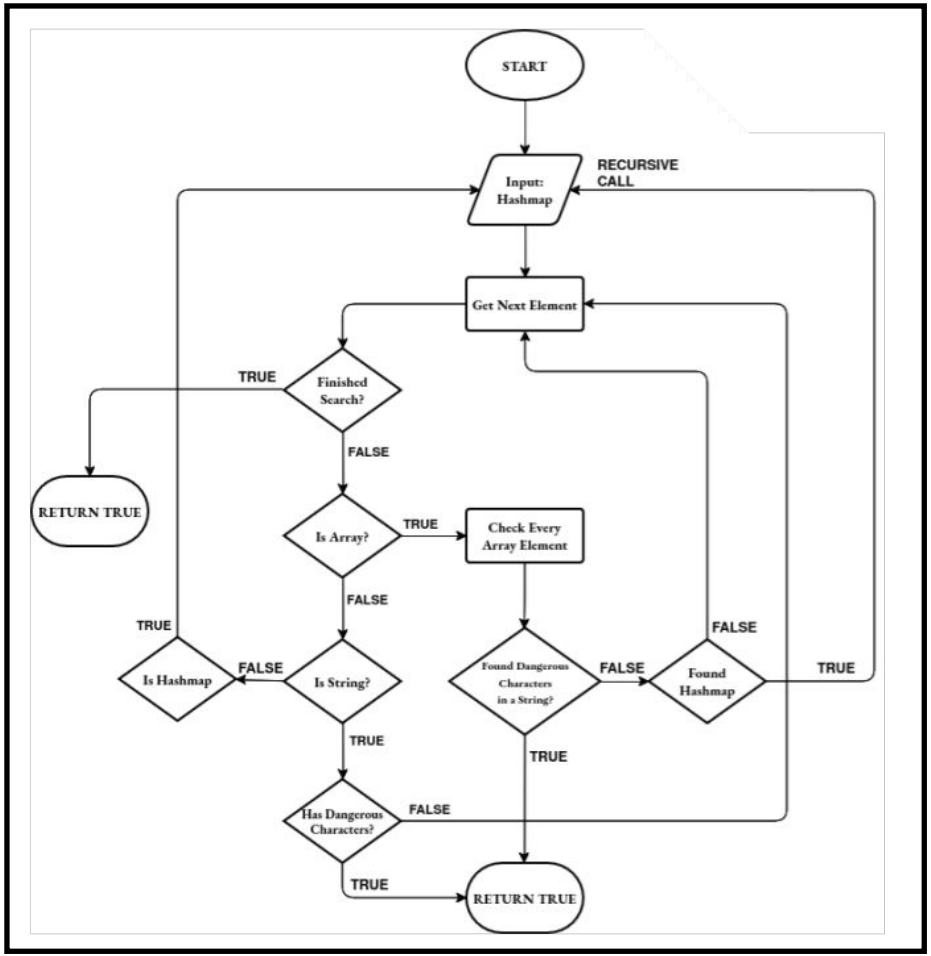
Top 3 query isolation

```python
def check_doctor_appointment_availability(cur, doctor_id, date):
    # Update lock (no other transaction can update appointment until this transaction is finished)
    cur.execute("""
                SELECT * FROM appointment
                WHERE doctor_employee_contract_service_user_user_id = %s AND
                (app_date BETWEEN %s::timestamp - INTERVAL %s AND %s::timestamp + INTERVAL %s)
                FOR UPDATE
                """, (doctor_id, date, APPOINTMENT_DURATION, date, APPOINTMENT_DURATION))
```

Update lock when checking doctor availability

Security

- Password hashing
- Query parameterization
- Dangerous characters detection



Dangerous character search flowchart

Security - Examples

```python
ITERATIONS = 100000
ALGORITHM = 'sha256'

def hash_password(password):
    # Generate salt
    salt = os.urandom(32)

    key = hashlib.pbkdf2_hmac(
        ALGORITHM, # The hash digest algorithm for HMAC
        password.encode('utf-8'), # Convert the password to bytes
        salt, # Provide the salt
        ITERATIONS # Use 1e6 iterations
    )

    return salt + key

def verify_password(stored_key, provided_password):
    # Convert the stored key to bytes
    stored_key = bytes.fromhex(stored_key.replace("\\x", ""))
    # Get the salt from the stored password
    salt = stored_key[:32]
    # Get the key from the stored password
    stored_key = stored_key[32:]
    # Hash the provided password
    new_key = hashlib.pbkdf2_hmac(
        ALGORITHM,
        provided_password.encode('utf-8'),
        salt,
        ITERATIONS
    )

    # Compare the stored key with the new key
    return new_key == stored_key
```

Password hashing function

```python
# Search dangerous characters in a user input string
def string_contains_dangerous_chars(input_str):
    # Check for SQL injection characters
    dangerous_chars = [';', '--', '/*', '*/']
    for char in dangerous_chars:
        if char in input_str:
            return True
    return False

# Check if a payload contains dangerous characters
def payload_contains_dangerous_chars(payload):
    for key, value in payload.items():
        # If it's a dictionary, recursively call the function
        if isinstance(value, dict):
            if payload_contains_dangerous_chars(value):
                return True
        # If it's a list or tuple, iterate over the elements and check each one
        elif isinstance(value, (list, tuple)):
            for element in value:
                if isinstance(element, str) and string_contains_dangerous_chars(element):
                    return True
                elif isinstance(element, (dict, list, tuple)) and payload_contains_dangerous_chars({0: element}):
                    return True
        # If it's a string, check for dangerous characters
        elif isinstance(value, str):
            if string_contains_dangerous_chars(value):
                return True
    return False
```

Dangerous character search function

Tuning

- Indexing attributes commonly used in WHERE clauses

- Denormalization of the *bill* table

Tuning - Examples

```
CREATE INDEX idx_surgery_surg_date
ON surgery (surg_date);

CREATE INDEX idx_hospitalization_start_date
ON hospitalization (start_date);

CREATE INDEX idx_hospitalization_end_date
ON hospitalization (end_date);

CREATE INDEX idx_payment_payment_date
ON payment (payment_date);
```

Indexing dates

bill			
bill_id	BInt	PK	AU
total_cost	BInt	NN	
already_payed	Bool	NN	

Bill denormalization