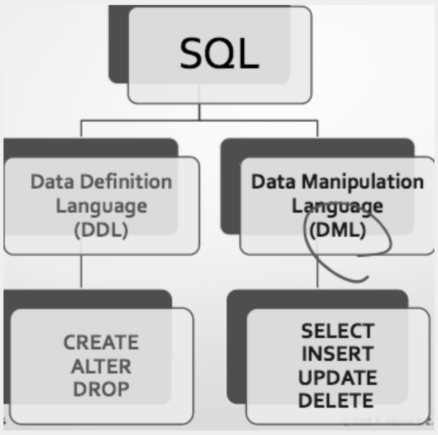
Cheatography

CS 411 Midterm 1 Cheat Sheet

by etau97hi1 via cheatography.com/32178/cs/18949/

SQL (Structured Query Language)


SQL
Data Definition Language (DDL)
Data Manipulation Language (DML)
CREATE
ALTER
DROP
SELECT
INSERT
UPDATE
DELETE

Relational Database Model

- Data is structured as **relations**

- Defines a limited set of **operations (query and modification)** to interact with the data

- Allow for defining **constraints** on columns (attributes), a table (relation), relationships among tables (foreign keys).

Syntax: Patterns and "LIKE"

- Used in a WHERE clause

- General form: `<Attribute> LIKE <pattern>` or `<Attribute> NOT LIKE <pattern>`

- Pattern: quoted string with % for any string, _ for any character

Closed

Query language is **closed** if we can use the answer from one query as input to another query

SQL Subquery Example

```
SELECT *
FROM (SELECT * FROM Customer WHERE
  name LIKE 'A%') as temp
WHERE temp.phone LIKE '5%';
```

SQL Subqueries that Return Scalar

Subquery can be used as value if guaranteed to produce one tuple with one component

- “Single” tuple often guaranteed by key constraint

- A run-time error if not scalar

SQL: Boolean Operators

<tuple> IN <relation> is true if and only if the tuple is a member of the relation.

EXISTS(<relation>) is true if and only if the <relation> is not empty.

x = ANY(<relation>) is a boolean condition meaning that x equals at least one tuple in the relation.

Similarly, x <> ALL(<relation>) is true iff for every tuple t in the relation, x is not equal to t

Ex: Aggregations

```
SELECT AVG(price) FROM Sells
WHERE drink = ‘Mocha’;
```

Aggregations

SUM, AVG, COUNT, MIN, and MAX can be applied to a column in a SELECT clause to produce that aggregation on the column.

Ex: Grouping

```
SELECT customer, AVG(price)
FROM Frequents, Sells
WHERE drink = ‘Mocha’ AND
Frequents.cafe = Sells.cafe
GROUP BY customer
```

Single User Assumptions:

- Each **operation** (UPDATE ... SET ... WHERE) is **executed one at a time**

- **ISOLATION** - one op exec, maybe change DB, then next op exec

- **ATOMIC** - op exec entirely or not at all

Transactions Definitions

- Group the SFW and USW into a **transaction**

- Transaction is a sequence of statements considered a "unit of operation" on DB

- **Serializability of transactions** - Either user1 transaction exec first or user2's, but not in parallel

Transactions

Transaction: sequence of read/write ops on the DB w/ the property that either all or none of actions complete.
- May either succeed (COMMIT), or fail (ABORT or ROLLBACK)

ACID Properties

Atomicity	either all ops exec or none
Consistency	trans exec in isolation keeps DB in consistent state
Isolation	trans isolated from effects of other concurrently exec trans
Durability	updates stay in DBMS

Transaction Manager

Ensure that transactions that exec in parallel don’t interfere with each other.

Concurrent Execution Problems

Write-Read conflict	dirty/inconsistent read
Read-Write conflict	Unrepeatable read
Write-Write conflict	lost update