Lecture 21: SQL DML (Basic SELECT)

SELECT, WHERE, ORDER BY, TOP, and Common Mistakes

Z2004: Database Management Systems

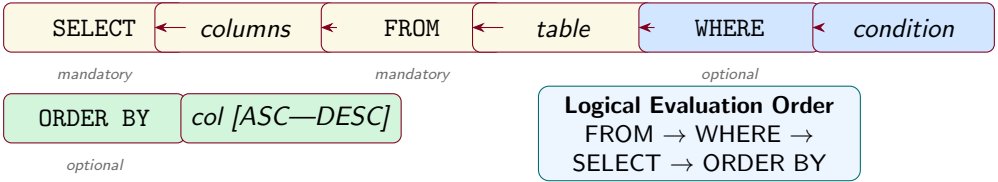


IIT Madras Zanzibar

Tuesday, 21 April 2026

Lecture Outline

Anatomy of a SELECT Statement



Full Clause Order (T-SQL)

`SELECT` *[TOP n] cols* `FROM` *tables* `[WHERE` *condition*`]` `[GROUP BY` *cols*`]` `[HAVING` *cond*`]`
`[ORDER BY` *cols*`]`

SELECT *, DISTINCT, and Column Aliases

```sql
-- All columns, all rows
SELECT * FROM Student;

-- Specific columns only
SELECT SID, Name, GPA
FROM   Student;

-- Eliminate duplicate departments
SELECT DISTINCT Dept
FROM   Student;

-- Column alias with AS
SELECT Name        AS StudentName,
       GPA         AS GradePoint,
       GPA * 25    AS ScoreOutOf100
FROM   Student;

-- Alias without AS (T-SQL allows)
SELECT Name StudentName
```

Avoid SELECT * in Production

`SELECT *` is fine for exploration but problematic in code:

- Column order may change
- Transfers unnecessary data
- Breaks views/stored procs on schema change

DISTINCT Behaviour

`DISTINCT` applies to the *combination* of all selected columns, not just the first one.

WHERE Clause — Operators

```sql
-- Comparison operators
SELECT * FROM Student WHERE GPA >=
    3.5;
SELECT * FROM Student WHERE Dept <> '
    CS';

-- BETWEEN (inclusive on both ends)
SELECT * FROM Student
WHERE  GPA BETWEEN 3.0 AND 3.9;

-- IN (membership)
SELECT * FROM Student
WHERE  Dept IN ('CS', 'EE', 'ME');

-- NOT IN
SELECT * FROM Student
WHERE  Dept NOT IN ('ME');
```

```sql
-- LIKE patterns
--  %  = zero or more characters
--  _  = exactly one character
SELECT * FROM Student
WHERE  Name LIKE 'A%';      --
    starts with A

SELECT * FROM Student
WHERE  Name LIKE '%ali';    -- ends
    with ali

SELECT * FROM Student
WHERE  Name LIKE '_a%';     -- 2nd
    char is a

-- NULL checks (never use = NULL)
SELECT * FROM Student
WHERE  GPA IS NULL;
```

WHERE — Logical Connectives

```sql
-- AND: both conditions must be true
SELECT * FROM Student
WHERE  Dept = 'CS'  AND  GPA > 3.5;

-- OR: at least one condition must be true
SELECT * FROM Student
WHERE  Dept = 'CS'  OR  GPA > 3.8;

-- NOT: negation
SELECT * FROM Student
WHERE  NOT (Dept = 'CS');

-- Combined: use parentheses to control precedence
SELECT * FROM Student
WHERE  (Dept = 'CS' OR Dept = 'EE')
  AND  GPA >= 3.0;
```

Operator Precedence in WHERE

ORDER BY — Sorting Results

```sql
-- Ascending (default)
SELECT Name, GPA
FROM   Student
ORDER BY GPA ASC;

-- Descending
SELECT Name, GPA
FROM   Student
ORDER BY GPA DESC;

-- Multiple columns
SELECT Name, Dept, GPA
FROM   Student
ORDER BY Dept ASC,
         GPA  DESC;

-- By column position (not recommended)
SELECT Name, GPA
```

ORDER BY Rules

- Default is `ASC` (ascending)
- `NULL` values sort *first* in ascending order (T-SQL)
- Can sort by columns not in the SELECT list
- Column position numbers (1, 2, ...) are valid but fragile — avoid in code
- Sorting is always the *last* logical step

NULL Ordering

In T-SQL, NULLs sort first in ASC, last in DESC. PostgreSQL allows `NULLS FIRST` /

Limiting Rows: TOP, LIMIT, FETCH FIRST

T-SQL (SQL Server):

```
-- Top 5 students by GPA
SELECT TOP 5 Name, GPA
FROM   Student
ORDER BY GPA DESC;

-- Top 10 percent
SELECT TOP 10 PERCENT Name, GPA
FROM   Student
ORDER BY GPA DESC;

-- With ties (include all ties at
    boundary)
SELECT TOP 5 WITH TIES Name, GPA
FROM   Student
ORDER BY GPA DESC;
```

PostgreSQL / MySQL:

ANSI SQL:2008+ (standard):

```
SELECT Name, GPA
FROM   Student
ORDER BY GPA DESC
FETCH FIRST 5 ROWS ONLY;

-- With offset (pagination)
SELECT Name, GPA
FROM   Student
ORDER BY GPA DESC
OFFSET 10 ROWS
FETCH NEXT 5 ROWS ONLY;
```

Pagination Pattern

`OFFSET` + `FETCH NEXT` is the standard way to implement page-by-page results (e.g.

Think-Pair-Share — Write the SQL Queries

Schema (reminder)

```
Student(SID, Name, Dept, GPA)
Course(CID, CName, Credits)
Enrolment(SID, CID, Grade)
```

1. List all student names sorted alphabetically.
2. Find distinct departments in the Student table.
3. List the names of students whose GPA is between 3.0 and 3.5 (inclusive).
4. Find all students whose name starts with the letter "M".

Pair up — 8 minutes

- Write SQL for each of the 5 queries
- Check each other's `WHERE` clauses carefully
- Verify `ORDER BY` column order and direction
- One pair will share each answer with the class

Sample Answers:

```sql
-- Q1
SELECT Name FROM Student
ORDER BY Name ASC;
-- Q3
SELECT Name FROM Student
```

Summary

Key Takeaways — L16 SQL DML: Basic SELECT

1. `SELECT` clause order: SELECT → FROM → WHERE → ORDER BY (logical evaluation is FROM first)
2. `SELECT *` is fine to explore; in production always name columns explicitly
3. `DISTINCT` eliminates duplicate result rows across all selected columns
4. Column aliases (`AS`) rename output columns; cannot use them in `WHERE`
5. `WHERE` operators: $=, \neq, <, \leq, >, \geq$, `BETWEEN`, `IN`, `LIKE` (%, _), `IS NULL`
6. `ORDER BY`: `ASC`/`DESC`, multiple columns, NULLs sort first in ascending T-SQL
7. Row limiting: `TOP n` (T-SQL), `LIMIT` (PostgreSQL), `FETCH FIRST` (ANSI standard)

Next Lecture — L17: SQL DML: INSERT, UPDATE, DELETE

L17 (7 Apr): Modifying data — `INSERT` (single row, multi-row, INSERT…SELECT), `UPDATE` with `SET`/`WHERE`, `DELETE` safety, common mistakes, and transaction basics (`BEGIN`