



Managing Data & Databases

Session 8
Linking your lists

Today's Dose of SQL

■ DML

- Alisases
- Subqueries
- INSERT INTO SELECT
- UNION
- JOIN
- EXISTS

■ DDL

- CREATE TABLE LIKE
- CREATE INDEX
- FOREIGN KEY

SQL Aliases

■ Use case

- You can give aliases to columns you select or the tables you select from. Aliases are valid only in the query where they are defined.

■ Simplified Syntax

- `SELECT `field1_name` AS `alias1`, ...
FROM `table_name` AS `alias2`;`

■ Example

- `SELECT Databasers.first_name AS `First Name`,
Databasers.last_name AS `Last Name`
FROM people AS Databasers;`
- `SELECT seniority*2 AS double_seniority
FROM people
WHERE double_seniority <= 2;`

SQL Subqueries

■ Use case

- By putting a query in parentheses and attributing an alias to its result set you can turn it into a subquery. Queries treat the results of their subqueries as in-memory tables. Using subqueries you can do multiple operations in a single query, and without saving the intermediary results.

■ Simplified Syntax

- `SELECT `field1_name`, ...
FROM (SELECT `field1_name`, ...
FROM `table_name`) AS `alias1`;`

■ Example

- `SELECT `First Name`
FROM (SELECT first_name AS `First Name`,
last_name AS `Last Name`
FROM people) AS Databasers;`

Copying tables in MySQL

- We need to have a table of old people in the class, with the exact same specification as the table “people”. MySQL lets you copy the entire structure of a table to a new table using the command `CREATE TABLE LIKE`. It lets you copy the data from one table to the other using `INSERT INTO SELECT`. So, all we have to do is:
 - `CREATE TABLE decrepits
LIKE people;`
 - `INSERT INTO decrepits
SELECT *
FROM people
WHERE seniority>=3;`
- You can also check the DDL of the new table to make sure the structure corresponds to what you need:
 - `SHOW CREATE TABLE decrepits;`

Indexes (Indices?!)

- Use case

- Indexes are used to speed up sorting, searching and joining. An index may be defined on one or more columns (depending on our needs).

- Syntax

- `CREATE INDEX `index_name`
ON `table_name`(`column1`, ...);`

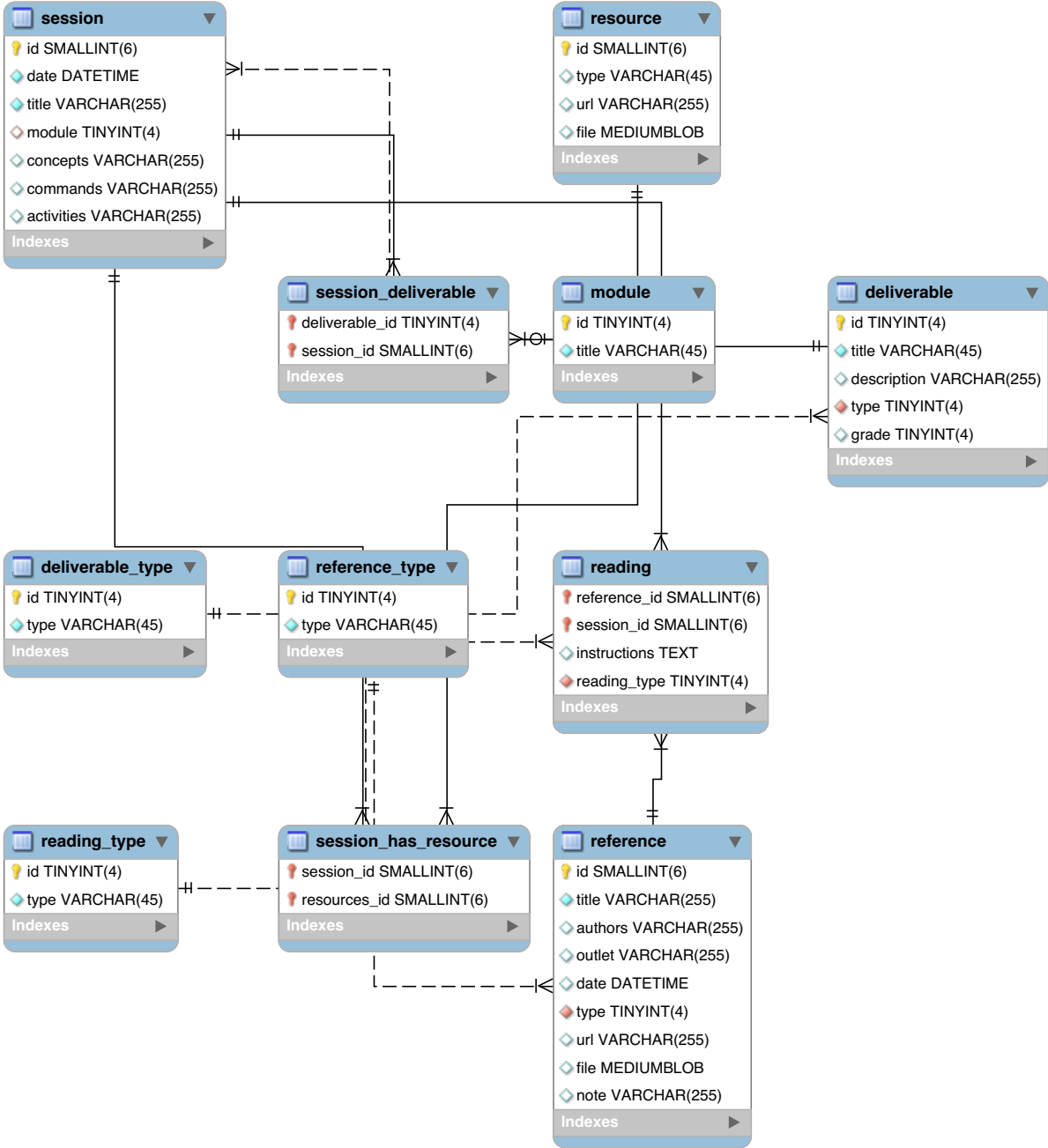
- Example

- `CREATE INDEX idx_person_name
ON people (last_name, first_name);`

- Since indexes are defined on tables, they should be removed from tables using ALTER TABLE:

- `ALTER TABLE people
DROP INDEX idx_person_name;`

Entity Relationship Diagrams (ERD)



SQL Command

FOREIGN KEY

■ Use case

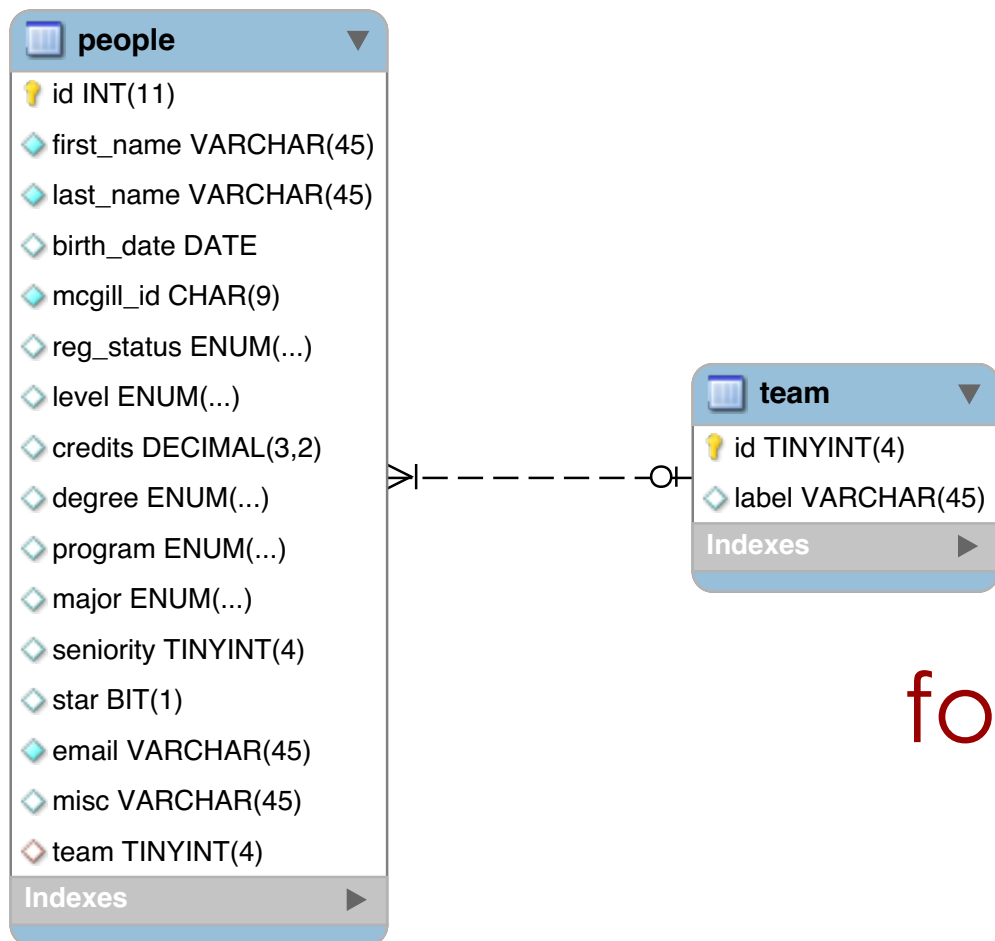
- Creates a link between two tables, making the primary key of one table the lookup list for the foreign key values of the other table.

■ Simplified Syntax

- `FOREIGN KEY `field_name`
REFERENCES `lookup_table_name` (`primary_key`);`

■ Example

- `CREATE TABLE team
(id TINYINT AUTO_INCREMENT UNIQUE PRIMARY KEY,
label VARCHAR(45));`
- `ALTER TABLE people
ADD team TINYINT,
ADD FOREIGN KEY(team)
REFERENCES team(id);`



What does a
foreign key look like?

SQL Command

JOIN

■ Use case

- JOIN is the perfect tool for putting back together all the data that we structure into different tables. JOIN integrates tables vertically.

■ Simplified Syntax

- `SELECT `some fields from table_1 and table_2`
FROM `table_1` JOIN `table_2`
ON `join_condition`;`

■ Example

- `SELECT first_name, last_name, label AS team_label
FROM people JOIN team
ON team=team.id;`

SQL Command UNION

- Use case

- Integrates tables horizontally

- Simplified Syntax

- `SELECT Query 1`
`UNION`
`SELECT Query 2`

- Example

- `SELECT id, last_name AS entity_name FROM people`
`UNION`
`SELECT id, label AS entity_name FROM team;`

- Attention

- The result sets of the two unionized queries must contain the same columns, with the same names and in the same order.

SQL Clause EXISTS

■ Use case

- Conditions the query to the existence of records within a subquery

■ Simplified Syntax

```
■ SELECT `column_name1`  
  FROM `table_name1`  
  WHERE EXISTS  
    (SELECT * FROM `table_name2`  
     WHERE `condition`);
```

■ Example

```
■ SELECT * FROM people  
  WHERE EXISTS  
    (SELECT * FROM team  
     WHERE team.id=people.team);
```