Bioinformatics Resources
- SQL -

Lecture & Exercises
Prof. B. Rost, Dr. L. Richter, J. Reeb

Institut für Informatik I12
SQL in use

- Syntax of MySQL server 5.7
- useful utilities: mysql, mysqladmin
- frequent tasks / operations
- command line
- Python integration
Prerequisites

- Know your server version: Even if SQL is a standard, different vendors implement different versions of add vendor/version specific features
- Have client programs installed (mysql, mysqladmin)
- Have a language driver/connector installed
Prerequisites

- command line clients come typically with the installation
- GUI clients are also available
- Connectors come in different flavors for many languages and may have to be installed
- Connectors may offer different APIs
(User) Administration

- Administration information is stored in the database ‘mysql’
- Users have to connect to the database server
- Users are managed via accounts:
  - username
  - hostname / IP address
  - optional password
(User) Administration

- Example: ‘dowj’@’myhost’
- special hostnames: localhost, 127.0.0.1, ::1
- ‘empty’ user name: anonymous
- hostnames may contain wildcards: % or _
- omission of user or host name allowed
- password can be and is set individually for each name/host combination
Example

| user    | host              | password |
|---------+-------------------+----------|
| root    | localhost         | *CABC7   |
| root    | phoenix.fritz.box | *CABC7   |
| root    | 127.0.0.1         | *CABC7   |
| root    | ::1               | *CABC7   |
| richter | localhost         | *75B62   |
|         | phoenix.fritz.box |          |

- Set / change password:
  SET PASSWORD for ‘dowj’@’localhost’ = PASSWORD(‘cleartext_password’)
More Administration

- default: only root account, might be unsecured
- databases: mysql, information_schema, performance schema, test
- further creation of user account and databases needed
More Administration

- use the mysqladmin tool to create a new, empty database:
  mysqladmin -u root -p create resource_db

- create a user (in a mysql session):
  CREATE USER dowj [identified by ‘clear_pw’]

- now you have to grant privileges on a certain database to the user:
  GRANT ALL ON resource_db.* TO ‘dowj’@’localhost’
Full Grant Syntax 1

GRANT

priv_type [(column_list)]

[, priv_type [(column_list)]] ...

ON [object_type] priv_level

TO user_specification [, user_specification] ...

[REQUIRE {NONE | tsl_option [[AND] tsl_option] ...}]

[WITH {GRANT OPTION | resource_option} ...]

GRANT PROXY ON user_specification

TO user_specification [, user_specification] ...

[WITH GRANT OPTION]

object_type: {

    TABLE
    | FUNCTION
    | PROCEDURE

}
Full Grant Syntax 2

priv_level: {
  *
  | *.*
  | db_name.*
  | db_name.tbl_name
  | tbl_name
  | db_name.routine_name
}

user_specification:
  user [ auth_option ]

auth_option: {
  IDENTIFIED BY 'auth_string'
  | IDENTIFIED BY PASSWORD 'hash_string'
  | IDENTIFIED WITH auth_plugin
  | IDENTIFIED WITH auth_plugin AS 'hash_string'
}
Full Grant Syntax 3

tsl_option: {
    SSL
    | X509
    | CIPHER 'cipher'
    | ISSUER 'issuer'
    | SUBJECT 'subject'
}

resource_option: {
    | MAX_QUERIES_PER_HOUR count
    | MAX_UPDATES_PER_HOUR count
    | MAX_CONNECTIONS_PER_HOUR count
    | MAX_USER_CONNECTIONS count
}
## Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Meaning and Grantable Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL (PRIVILEGES)</td>
<td>Grant all privileges at specified access level except GRANT OPTION</td>
</tr>
<tr>
<td>ALTER</td>
<td>Enable use of ALTER TABLE. Levels: Global, database, table.</td>
</tr>
<tr>
<td>ALTER ROUTINE</td>
<td>Enable stored routines to be altered or dropped. Levels: Global, database, procedure.</td>
</tr>
<tr>
<td>CREATE</td>
<td>Enable database and table creation. Levels: Global, database, table.</td>
</tr>
<tr>
<td>CREATE ROUTINE</td>
<td>Enable stored routine creation. Levels: Global, database.</td>
</tr>
<tr>
<td>CREATE TABLESPACE</td>
<td>Enable tablespaces and log file groups to be created, altered, or dropped. Level: Global.</td>
</tr>
<tr>
<td>CREATE TEMPORARY TABLES</td>
<td>Enable use of CREATE TEMPORARY TABLE. Levels: Global, database.</td>
</tr>
<tr>
<td>CREATE USER</td>
<td>Enable use of CREATE USER, DROP USER, RENAME USER, and REVOKE ALL PRIVILEGES. Level: Global.</td>
</tr>
<tr>
<td>CREATE VIEW</td>
<td>Enable views to be created or altered. Levels: Global, database, table.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Enable use of DELETE. Level: Global, database, table.</td>
</tr>
<tr>
<td>DROP</td>
<td>Enable databases, tables, and views to be dropped. Levels: Global, database, table.</td>
</tr>
<tr>
<td>EVENT</td>
<td>Enable use of events for the Event Scheduler. Levels: Global, database.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Enable the user to execute stored routines. Levels: Global, database.</td>
</tr>
<tr>
<td>FILE</td>
<td>Enable the user to cause the server to read or write files. Level: Global.</td>
</tr>
<tr>
<td>GRANT OPTION</td>
<td>Enable privileges to be granted to or removed from other accounts. Levels: Global, database, table, procedure, proxy.</td>
</tr>
<tr>
<td>INDEX</td>
<td>Enable indexes to be created or dropped. Levels: Global, database, table.</td>
</tr>
<tr>
<td>INSERT</td>
<td>Enable use of INSERT. Levels: Global, database, table, column.</td>
</tr>
<tr>
<td>LOCK TABLES</td>
<td>Enable use of LOCK TABLES on tables for which you have the SELECT privilege. Levels: Global, database.</td>
</tr>
<tr>
<td>PROCESS</td>
<td>Enable the user to see all processes with SHOW PROCESSLIST. Level: Global.</td>
</tr>
<tr>
<td>PROXY</td>
<td>Enable user proxying. Level: From user to user.</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>Enable foreign key creation. Levels: Global, database, table, column.</td>
</tr>
<tr>
<td>RELOAD</td>
<td>Enable use of FLUSH operations. Level: Global.</td>
</tr>
<tr>
<td>REPLICATION CLIENT</td>
<td>Enable the user to ask where master or slave servers are. Level: Global.</td>
</tr>
<tr>
<td>REPLICATION SLAVE</td>
<td>Enable replication slaves to read binary/log events from the master. Level: Global.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Enable use of SELECT. Levels: Global, database, table, column.</td>
</tr>
<tr>
<td>SHOW DATABASES</td>
<td>Enable use of SHOW DATABASES to show all databases. Level: Global.</td>
</tr>
<tr>
<td>SHOW VIEW</td>
<td>Enable use of SHOW CREATE VIEW. Levels: Global, database, table.</td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td>Enable use of mysqldadmin shutdown. Level: Global.</td>
</tr>
<tr>
<td>SUPER</td>
<td>Enable use of other administrative operations such as CHANGE MASTER TO, kill, purge binary logs, set global, and mysqldadmin debug command. Level: Global.</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>Enable trigger operations. Levels: Global, database, table.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Enable use of UPDATE. Levels: Global, database, table, column.</td>
</tr>
<tr>
<td>USAGE</td>
<td>Synonym for &quot;no privileges&quot;</td>
</tr>
</tbody>
</table>

Useful / Important MySQL programs

- `mysqld`: database server (demon)
- Typically started via `mysqld_safe` or `mysql.server`
- `mysql_install_db`: initializes the MySQL data directory and the grant tables and executes once upon installation
- Client programs: `mysql`, `mysqladmin`, `mysqldump`, `mysqlimport`, `mysqlshow`
- More program for logging and self-checks
mysqladmin

- tool to perform administrative tasks
- create a new database
- drop a database
- flush-commands (forced write to disc)
- retrieve status information
Hacking your MySQL database

- in case you forgot your database root account


- assumes you have admin privileges on your computer:
  - stop the server
  - restart the server with `--skip-grant-tables` and `--skip-working`
  - connect with the mysql client
  - enable privileges with `FLUSH PRIVILEGES`
Hacking your MySQL database

- depending on your version:
  - `ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass';` (version 5.7.6 and later)
  - `SET PASSWORD FOR 'root'@'localhost' = PASSWORD('MyNewPass');` (version 5.7.5 and earlier)
  - stop and restart the server, now without the `skip-` clauses
**mysqlshow**


- access to various show commands like:
  - available databases
  - accessibel tables
  - ...

- `mysqlshow [options] [db_name [tbl_name [col_name]]]`
mysqldump

- Used to create a logical backup of a database
- Output in SQL, CSV or XML format
- Simple dump & restore:
  
  ```
  mysqldump db_name > backup-file.sql
  mysql db_name < backup-file.sql
  ```

- Copy from one server to the other:
  
  ```
  mysqldump --opt db_name | mysql --host=remote_host -C db_name
  ```
mysqlimport

- [Link](https://dev.mysql.com/doc/refman/5.7/en/mysqlimport.html)

- Interface to the `LOAD DATA INFILE` command

- `mysqlimport [options] db_name textfile1 [textfile2 ...]`

- Data from `textfileX` is imported into table `textfileX`

- Option `--local` allows to import local client data

- Non-local load operations need `FILE` privileges
LOAD DATA INFILE

LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[PARTITION (partition_name,...)]
[CHARACTER SET charset_name]
{{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
}
[LINES]
  [STARTING BY 'string']
  [TERMINATED BY 'string']
}
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]
mysql

- main client to interact with the database
- graphical alternatives are also available e.g. MySQLWorkbench
- used for most of the interactive work with the database
Important SQL Command

- from https://dev.mysql.com/doc/refman/5.7/en/

- Data Definition Language (DDL):
  - to create and to modify the table features
  - create
  - drop
  - alter ...

- Data Manipulation Language (DML):
  - access the data
  - insert, update, delete
  - select
  - join
CREATE TABLE


CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  (create_definition,...)
  [table_options]
  [partition_options]

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  [(create_definition,...)]
  [table_options]
  [partition_options]
  [IGNORE | REPLACE]
  [AS] query_expression

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  { LIKE old_tbl_name | (LIKE old_tbl_name) }
CREATE TABLE

create_definition:
col_name column_definition
| [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
  [index_option] ...
| {INDEX|KEY} [index_name] [index_type] (index_col_name,...)
  [index_option] ...
| [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
  [index_name] [index_type] (index_col_name,...)
  [index_option] ...
| {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...)
  [index_option] ...
| [CONSTRAINT [symbol]] FOREIGN KEY
  [index_name] (index_col_name,...) reference_definition
| CHECK (expr)
CREATE TABLE

column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
    [COMMENT 'string']
    [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
    [STORAGE {DISK|MEMORY|DEFAULT}]
    [reference_definition]
    | data_type [GENERATED ALWAYS] AS (expression)
        [VIRTUAL | STORED] [UNIQUE [KEY]] [COMMENT comment]
    [NOT NULL | NULL] [[PRIMARY] KEY]
**CREATE TABLE**

<table>
<thead>
<tr>
<th>data_type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT[(length)]</td>
</tr>
<tr>
<td>TINYINT[(length)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>SMALLINT[(length)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>INT[(length)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>INTEGER[(length)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>BIGINT[(length)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>REAL[(length, decimals)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>DOUBLE[(length, decimals)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>FLOAT[(length, decimals)] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>DECIMAL[(length[, decimals])] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>NUMERIC[(length[, decimals])] [UNSIGNED] [ZEROFILL]</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>TIME[(fsp)]</td>
</tr>
<tr>
<td>TIMESTAMP[(fsp)]</td>
</tr>
<tr>
<td>DATETIME[(fsp)]</td>
</tr>
<tr>
<td>YEAR</td>
</tr>
<tr>
<td>CHAR[(length)] [BINARY]</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>VARCHAR(length) [BINARY]</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>BINARY[(length)]</td>
</tr>
<tr>
<td>VARBINARY[length]</td>
</tr>
<tr>
<td>TINYBLOB</td>
</tr>
<tr>
<td>BLOB</td>
</tr>
<tr>
<td>MEDIUMBLOB</td>
</tr>
<tr>
<td>LONGBLOB</td>
</tr>
<tr>
<td>TINYTEXT [BINARY]</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>TEXT [BINARY]</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>MEDIUMTEXT [BINARY]</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>LONGTEXT [BINARY]</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>ENUM(value1,value2,value3,...)</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>SET(value1,value2,value3,...)</td>
</tr>
<tr>
<td>[CHARACTER SET charset_name] [COLLATE collation_name]</td>
</tr>
<tr>
<td>JSON</td>
</tr>
<tr>
<td>spatial_type</td>
</tr>
</tbody>
</table>
CREATE TABLE

index_col_name:
  col_name [(length)] [ASC | DESC]

index_type:
  USING {BTREE | HASH}

index_option:
  KEY_BLOCK_SIZE [=] value
  index_type
  WITH PARSER parser_name
  COMMENT 'string'

reference_definition:
  REFERENCES tbl_name (index_col_name,...)
    [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
  RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
CREATE TABLE

```
index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
    | index_type
    | WITH PARSER parser_name
    | COMMENT 'string'

reference_definition:
    REFERENCES tbl_name (index_col_name,...)
    [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```
### CREATE TABLE

```
table_option:
    ENGINE [ = ] engine_name
    AUTO_INCREMENT [ = ] value
    AVG_ROW_LENGTH [ = ] value
    [DEFAULT] CHARACTER SET [ = ] charset_name
    CHECKSUM [ = ] { 0 | 1 }
    [DEFAULT] COLLATE [ = ] collation_name
    COMMENT [ = ] 'string'
    COMPRESSION [ = ] {'ZLIB'|'LZ4'|'NONE'}
    CONNECTION [ = ] 'connect_string'
    DATA DIRECTORY [ = ] 'absolute path to directory'
    DELAY_KEY_WRITE [ = ] { 0 | 1 }
    ENCRYPTION [ = ] {'Y' | 'N'}
    INDEX DIRECTORY [ = ] 'absolute path to directory'
    INSERT_METHOD [ = ] { NO | FIRST | LAST }
    KEY_BLOCK_SIZE [ = ] value
    MAX_ROWS [ = ] value
    MIN_ROWS [ = ] value
    PACK_KEYS [ = ] { 0 | 1 | DEFAULT }
    PASSWORD [ = ] 'string'
    ROW_FORMAT [ = ] { DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT }
    STATS_AUTO_RECALC [ = ] { DEFAULT | 0 | 1 }
    STATS_PERSISTENT [ = ] { DEFAULT | 0 | 1 }
    STATS_SAMPLE_PAGES [ = ] value
    TABLESPACE tablespace_name [ STORAGE { DISK | MEMORY | DEFAULT } ]
    UNION [ = ] ( tbl_name[], tbl_name[], ...)
```
CREATE TABLE Examples

- Create a table with the same layout as an existing table:
  CREATE TABLE new_tbl LIKE orig_tbl;

- Create as a copy of another table:
  CREATE TABLE new_tbl AS SELECT * FROM orig_tbl;

- CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));

- CREATE TABLE animals ( id MEDIUMINT NOT NULL AUTO_INCREMENT, name CHAR(30) NOT NULL, PRIMARY KEY (id) );
DROP TABLE

- quite simple:
  DROP [TEMPORARY] TABLE [IF EXISTS]
  tbl_name [, tbl_name] ...
  [RESTRICT | CASCADE]

- other DROP statements are analogous
ALTER TABLE

- ALTER TABLE tbl_name
  [alter_specification [, alter_specification] ...]
  [partition_options]

- alter_specifications: mostly of following type:
  - ADD
  - DROP
  - RENAME
INSERT

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name,...)]
[(col_name,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name,...)]
SET col_name={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name,...)]
[(col_name,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
    [, col_name=expr] ... ]
INSERT Examples

- INSERT INTO tbl_name (col1,col2) VALUES(15,col1*2);
- INSERT INTO tbl_name (a,b,c) VALUES(1,2,3), (4,5,6), (7,8,9);
UPDATE

- UPDATE t1 SET col1 = col1 + 1;
- UPDATE items,month SET items.price=month.price WHERE items.id=month.id;
Delete

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
[PARTITION (partition_name,...)]
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]

- DELETE FROM t1, t2 USING t1 INNER JOIN t2
  INNER JOIN t3 WHERE t1.id=t2.id AND
  t2.id=t3.id;

- DELETE t1 FROM t1 LEFT JOIN t2 ON
  t1.id=t2.id WHERE t2.id IS NULL;
SELECT

[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[Straight_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]

select_expr [ , select_expr ... ]

[FROM table_references
 [PARTITION partition_list]

[WHERE where_condition]

[GROUP BY {col_name | expr | position}]
[ASC | DESC], ... [WITH ROLLUP]]

[HAVING where_condition]

[ORDER BY {col_name | expr | position}]
[ASC | DESC], ...]

[LIMIT {[offset,] row_count | row_count OFFSET offset}]

[PROCEDURE procedure_name(argument_list)]

[INTO OUTFILE 'file_name'
 [CHARACTER SET charset_name]
 export_options
 | INTO DUMPFILE 'file_name'
 | INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
Expression

expr:
   expr OR expr
| expr || expr
| expr XOR expr
| expr AND expr
| expr && expr
| NOT expr
| ! expr
| boolean_primary IS [NOT] {TRUE | FALSE | UNKNOWN}
| boolean_primary

boolean_primary:
   boolean_primary IS [NOT] NULL
| boolean_primary <= predicate
| boolean_primary comparison_operator predicate
| boolean_primary comparison_operator {ALL | ANY} (subquery)
| predicate

comparison_operator: = | >= | > | <= | < | <> | !=

bit_expr:
   bit_expr | bit_expr
| bit_expr & bit_expr
| bit_expr << bit_expr
| bit_expr >> bit_expr
| bit_expr + bit_expr
| bit_expr - bit_expr
| bit_expr * bit_expr
| bit_expr / bit_expr
| bit_expr DIV bit_expr
| bit_expr MOD bit_expr
| bit_expr % bit_expr
| bit_expr ^ bit_expr
| bit_expr + interval_expr
| bit_expr - interval_expr
| simple_expr

simple_expr:
   literal
| identifier
| function_call
| simple_expr COLLATE collation_name
| param_marker
| variable
| simple_expr || simple_expr
| + simple_expr
| - simple_expr
| simple_expr
| simple_expr BINOP simple_expr
| (expr [, expr] ...)
| ROW (expr, expr [, expr] ...)
| (subquery)
| EXISTS (subquery)
| {identifier expr}
| match_expr
| case_expr
| interval_expr
SELECT Examples

- SELECT * FROM t1 INNER JOIN t2 ...
- SELECT t1.*, t2.* FROM t1 INNER JOIN t2 ...
- SELECT AVG(score), t1.* FROM t1 ...
- SELECT CONCAT(last_name,', ',first_name) AS full_name FROM mytable ORDER BY full_name;
- SELECT t1.name, t2.salary FROM employee AS t1, info AS t2 WHERE t1.name = t2.name;
- SELECT t1.name, t2.salary FROM employee t1, info t2 WHERE t1.name = t2.name;
SELECT Examples / GROUP/ORDER BY

- SELECT college, region, seed FROM tournament
  ORDER BY region, seed;

- SELECT college, region AS r, seed AS s FROM tournament
  ORDER BY r, s [ASC];

- SELECT a, COUNT(b) FROM test_table GROUP BY a DESC;

- SELECT COUNT(col1) AS col2 FROM t GROUP BY col2 HAVING col2 = 2;

- SELECT user, MAX(salary) FROM users
  GROUP BY user HAVING MAX(salary) > 10;
JOIN

table_references:
    escaped_table_reference [, escaped_table_reference] ...

escaped_table_reference:
    table_reference
    | { OJ table_reference }

table_reference:
    table_factor
    | join_table

table_factor:
    tbl_name [PARTITION (partition_names)]
    | ([AS] alias) [index_hint_list]
    | table_subquery [AS] alias
    | ( table_references )

join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
    | table_reference STRAIGHT_JOIN table_factor
    | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
    | table_reference {LEFT | RIGHT} [OUTER] JOIN table_reference join_condition
    | table_reference NATURAL [{LEFT | RIGHT} [OUTER]] JOIN table_factor

join_condition:
    ON conditional_expr
    | USING (column_list)

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX | KEY}
    | [FOR {JOIN | ORDER BY | GROUP BY}] ([index_list])
    | IGNORE {INDEX | KEY}
    | [FOR {JOIN | ORDER BY | GROUP BY}] ([index_list])
    | FORCE {INDEX | KEY}
    | [FOR {JOIN | ORDER BY | GROUP BY}] ([index_list])

index_list:
    index_name [, index_name] ...
JOIN Examples

- SELECT * FROM t1 LEFT JOIN (t2, t3, t4) ON (t2.a = t1.a AND t3.b = t1.b AND t4.c = t1.c)

- SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4) ON (t2.a = t1.a AND t3.b = t1.b AND t4.c = t1.c)

- SELECT t1.name, t2.salary FROM employee AS t1 INNER JOIN info AS t2 ON t1.name = t2.name;

- SELECT t1.name, t2.salary FROM employee t1 INNER JOIN info t2 ON t1.name = t2.name;
JOIN Examples

- a LEFT JOIN b USING (c1, c2, c3)

- Does not work:
  
  ```sql
  SELECT * FROM t1, t2 JOIN t3 ON (t1.i1 = t3.i3);
  ```

- Confusing:
  
  ```sql
  SELECT * FROM (t1, t2) JOIN t3 ON (t1.i1 = t3.i3);
  ```

- Better:
  
  ```sql
  SELECT * FROM t1 JOIN t2 JOIN t3 ON (t1.i1 = t3.i3);
  ```
DESCRIBE, SHOW, EXPLAIN, LIMIT

- **DESCRIBE/EXPLAIN** (synonymous):
  - **DESCRIBE**: information about a table definition
  - **EXPLAIN**: execution plan info

- **SHOW**: command used to create a table ([https://dev.mysql.com/doc/refman/5.7/en/show-create-table.html](https://dev.mysql.com/doc/refman/5.7/en/show-create-table.html))