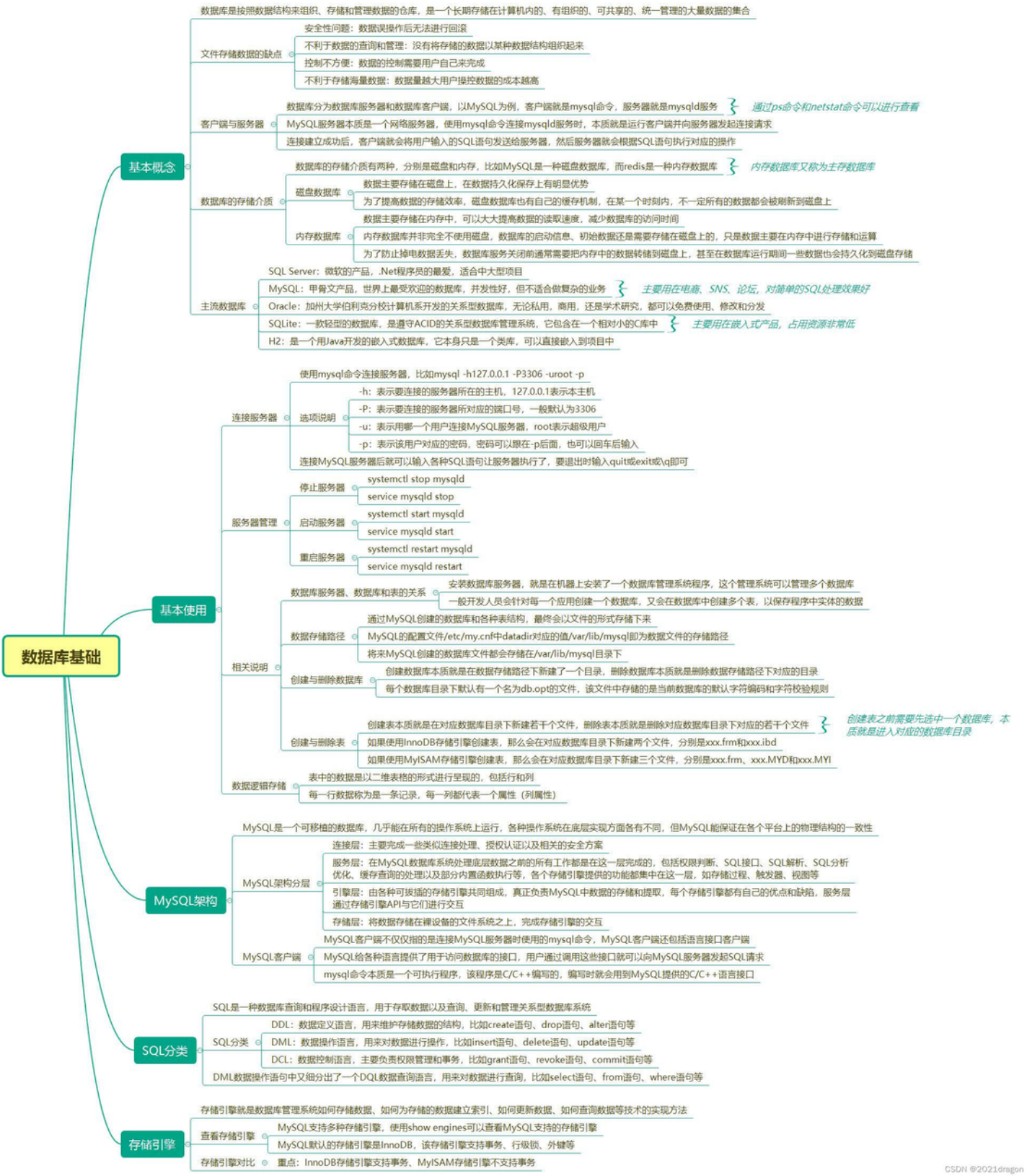
大佬的MySQL复习导图

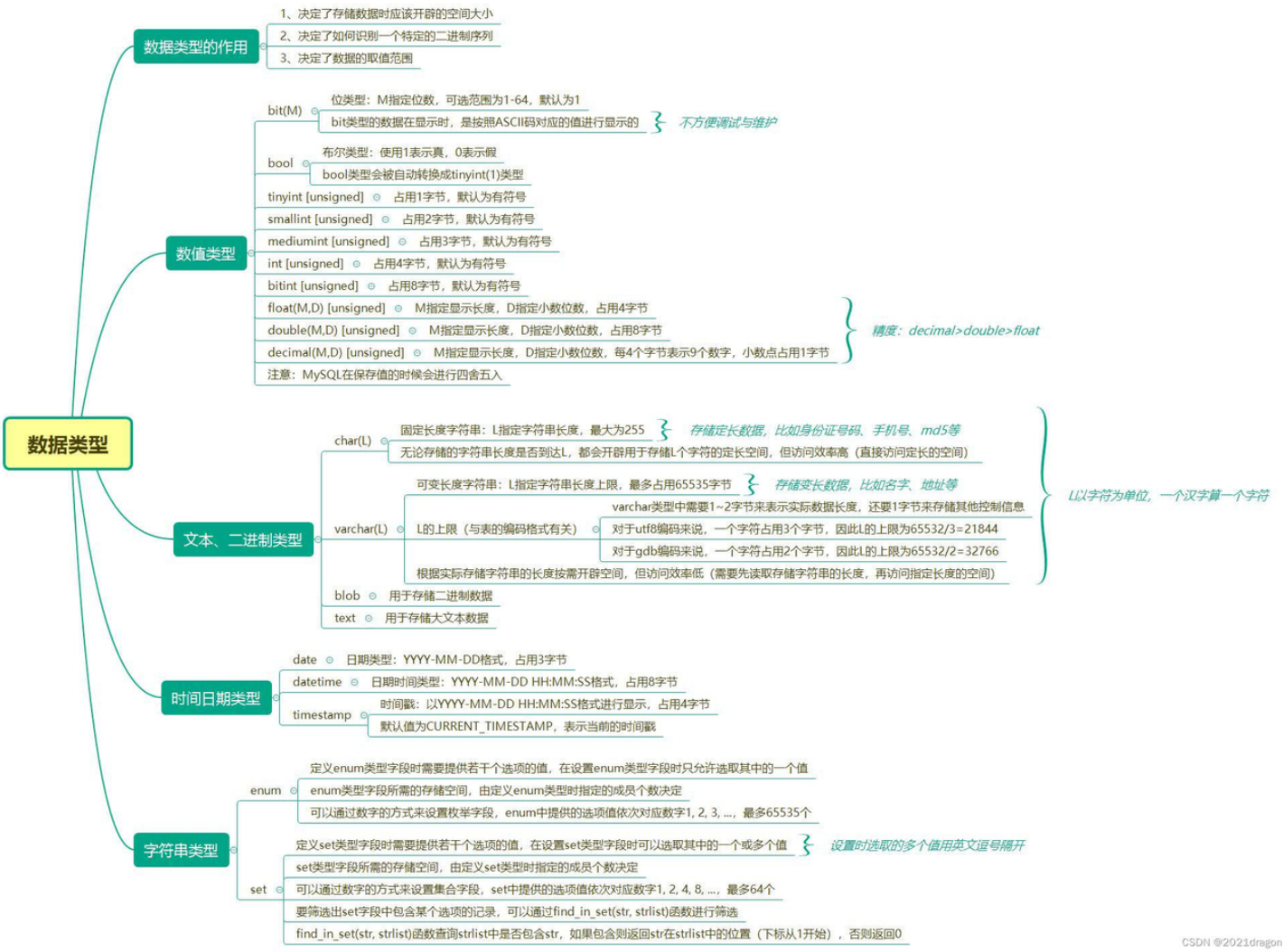
一、数据库基础


数据库基础
基本概念
数据库是按照数据结构来组织、存储和管理数据的仓库，是一个长期存储在计算机内的、有组织的、可共享的、统一管理的大量数据的集合
文件存储数据的缺点
安全性问题：数据误操作后无法进行回滚
不利于数据的查询和管理：没有将存储的数据以某种数据结构组织起来
控制不方便：数据的控制需要用户自己来完成
不利于存储海量数据：数据量越大用户操控数据的成本越高
客户端与服务器
数据库分为数据库服务器和数据库客户端，以MySQL为例，客户端就是mysql命令，服务器就是mysqld服务
通过ps命令和netstat命令可以进行查看
MySQL服务器本质是一个网络服务器，使用mysql命令连接mysqld服务时，本质就是运行客户端并向服务器发起连接请求
连接建立成功后，客户端就会将用户输入的SQL语句发送给服务器，然后服务器就会根据SQL语句执行对应的操作
数据库的存储介质
数据库的存储介质有两种，分别是磁盘和内存，比如MySQL是一种磁盘数据库，而redis是一种内存数据库
内存数据库又称为主存数据库
磁盘数据库
数据主要存储在磁盘上，在数据持久化保存上有明显优势
为了提高数据的存储效率，磁盘数据库也有自己的缓存机制，在某一个时刻内，不一定所有的数据都会被刷新到磁盘上
内存数据库
数据主要存储在内存中，可以大大提高数据的读取速度，减少数据库的访问时间
内存数据库并非完全不使用磁盘，数据库的启动信息、初始数据还是需要存储在磁盘上的，只是数据主要在内存中进行存储和运算
为了防止掉电数据丢失，数据库服务关闭前通常需要把内存中的数据转储到磁盘上，甚至在数据库运行期间一些数据也会持久化到磁盘存储
主流数据库
SQL Server：微软的产品，.Net程序员的最爱，适合中大型项目
MySQL：甲骨文产品，世界上最受欢迎的数据库，并发性好，但不适合做复杂的业务
主要用在电商、SNS、论坛，对简单的SQL处理效果好
Oracle：加州大学伯利克分校计算机系开发的关系型数据库，无论私用，商用，还是学术研究，都可以免费使用、修改和分发
SQLite：一款轻型的数据库，是遵守ACID的关系型数据库管理系统，它包含在一个相对小的C库中
主要用在嵌入式产品，占用资源非常低
H2：是一个用Java开发的嵌入式数据库，它本身只是一个类库，可以直接嵌入到项目中
基本使用
连接服务器
使用mysql命令连接服务器，比如mysql -h127.0.0.1 -P3306 -uroot -p
选项说明
-h：表示要连接的服务器所在的主机，127.0.0.1表示本主机
-P：表示要连接的服务器所对应的端口号，一般默认为3306
-u：表示用哪一个用户连接MySQL服务器，root表示超级用户
-p：表示该用户对应的密码，密码可以跟在-p后面，也可以回车后输入
连接MySQL服务器后就可以输入各种SQL语句让服务器执行了，要退出时输入quit或exit或\q即可
服务器管理
停止服务器
systemctl stop mysqld
service mysqld stop
启动服务器
systemctl start mysqld
service mysqld start
重启服务器
systemctl restart mysqld
service mysqld restart
相关说明
数据库服务器、数据库和表的关系
安装数据库服务器，就是在机器上安装了一个数据库管理系统程序，这个管理系统可以管理多个数据库
一般开发人员会针对每一个应用创建一个数据库，又会在数据库中创建多个表，以保存程序中实体的数据
数据存储路径
通过MySQL创建的数据库和各种表结构，最终会以文件的形式存储下来
MySQL的配置文件/etc/my.cnf中datadir对应的值/var/lib/mysql即为数据文件的存储路径
将来MySQL创建的数据库文件都会存储在/var/lib/mysql目录下
创建与删除数据库
创建数据库本质就是在数据存储路径下新建了一个目录，删除数据库本质就是删除数据存储路径下对应的目录
每个数据库目录下默认有一个名为db.opt的文件，该文件中存储的是当前数据库的默认字符编码和字符校验规则
创建与删除表
创建表本质就是在对应数据库目录下新建若干个文件，删除表本质就是删除对应数据库目录下对应的若干个文件
创建表之前需要先选中一个数据库，本质就是进入对应的数据库目录
如果使用InnoDB存储引擎创建表，那么会在对应数据库目录下新建两个文件，分别是xxx.frm和xxx.ibd
如果使用MyISAM存储引擎创建表，那么会在对应数据库目录下新建三个文件，分别是xxx.frm、xxx.MYD和xxx.MYI
数据逻辑存储
表中的数据是以二维表格的形式进行呈现的，包括行和列
每一行数据称为是一条记录，每一列都代表一个属性（列属性）
MySQL架构
MySQL是一个可移植的数据库，几乎能在所有的操作系统上运行，各种操作系统在底层实现方面各有不同，但MySQL能保证在各个平台上的物理结构的一致性
MySQL架构分层
连接层：主要完成一些类似连接处理、授权认证以及相关的安全方案
服务层：在MySQL数据库系统处理底层数据之前的所有工作都是在这一层完成的，包括权限判断、SQL接口、SQL解析、SQL分析优化、缓存查询的处理以及部分内置函数执行等，各个存储引擎提供的功能都集中在这一层，如存储过程、触发器、视图等
引擎层：由各种可拔插的存储引擎共同组成，真正负责MySQL中数据的存储和提取，每个存储引擎都有自己的优点和缺陷，服务层通过存储引擎API与它们进行交互
存储层：将数据存储在裸设备的文件系统之上，完成存储引擎的交互
MySQL客户端
MySQL客户端不仅仅指的是连接MySQL服务器时使用的mysql命令，MySQL客户端还包括语言接口客户端
MySQL给各种语言提供了用于访问数据库的接口，用户通过调用这些接口就可以向MySQL服务器发起SQL请求
mysql命令本质是一个可执行程序，该程序是C/C++编写的，编写时就会用到MySQL提供的C/C++语言接口
SQL分类
SQL是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系型数据库系统
SQL分类
DDL：数据定义语言，用来维护存储数据的结构，比如create语句、drop语句、alter语句等
DML：数据操作语言，用来对数据进行操作，比如insert语句、delete语句、update语句等
DCL：数据控制语言，主要负责权限管理和事务，比如grant语句、revoke语句、commit语句等
DML数据操作语句中又细分出了一个DQL数据查询语言，用来对数据进行查询，比如select语句、from语句、where语句等
存储引擎
存储引擎就是数据库管理系统如何存储数据、如何为存储的数据建立索引、如何更新数据、如何查询数据等技术的实现方法
查看存储引擎
MySQL支持多种存储引擎，使用show engines可以查看MySQL支持的存储引擎
MySQL默认的存储引擎是InnoDB，该存储引擎支持事务、行级锁、外键等
存储引擎对比
重点：InnoDB存储引擎支持事务、MyISAM存储引擎不支持事务
CSDN @2021dragon

二、库的操作

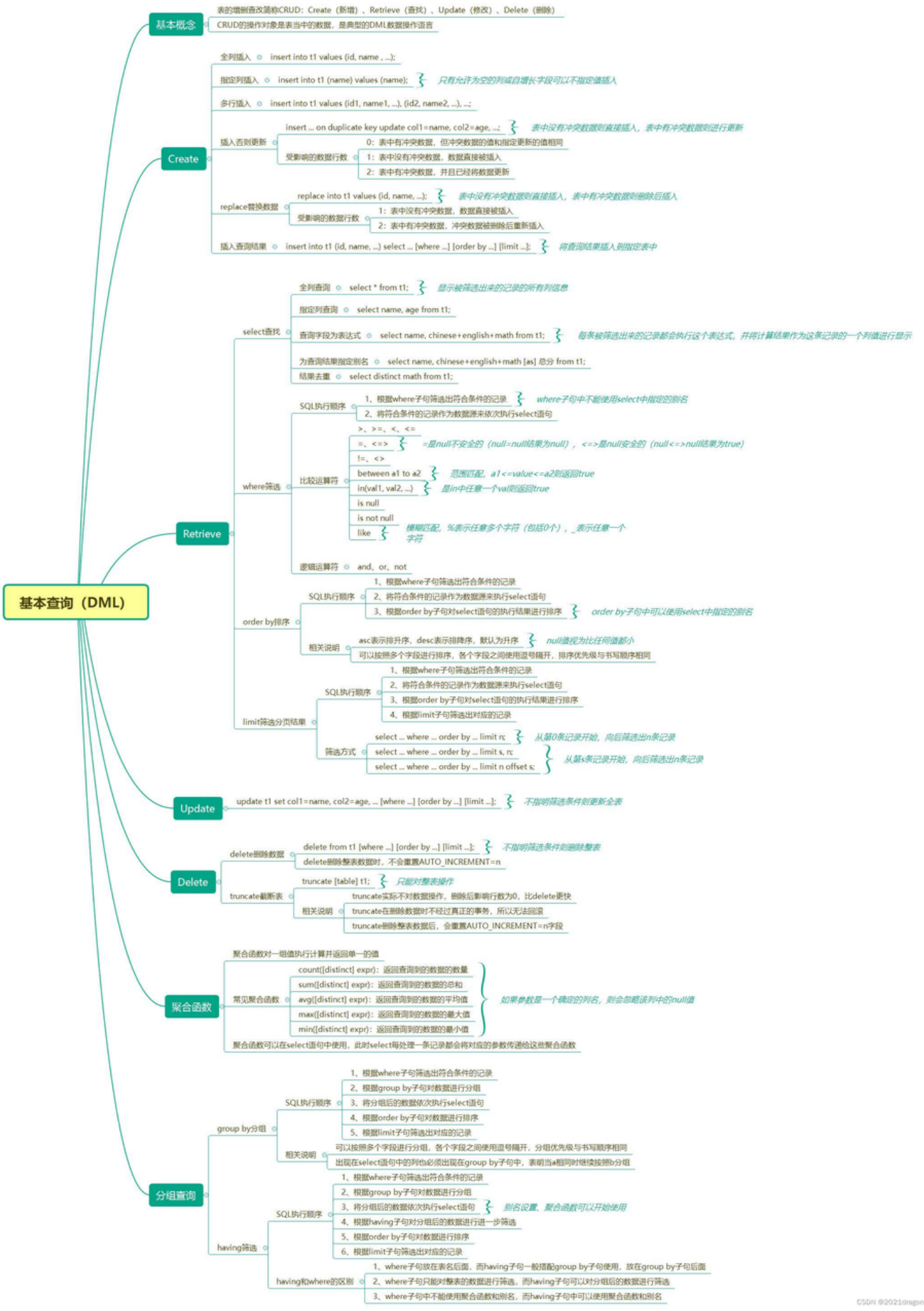
三、表的操作（DDL）

四、数据类型

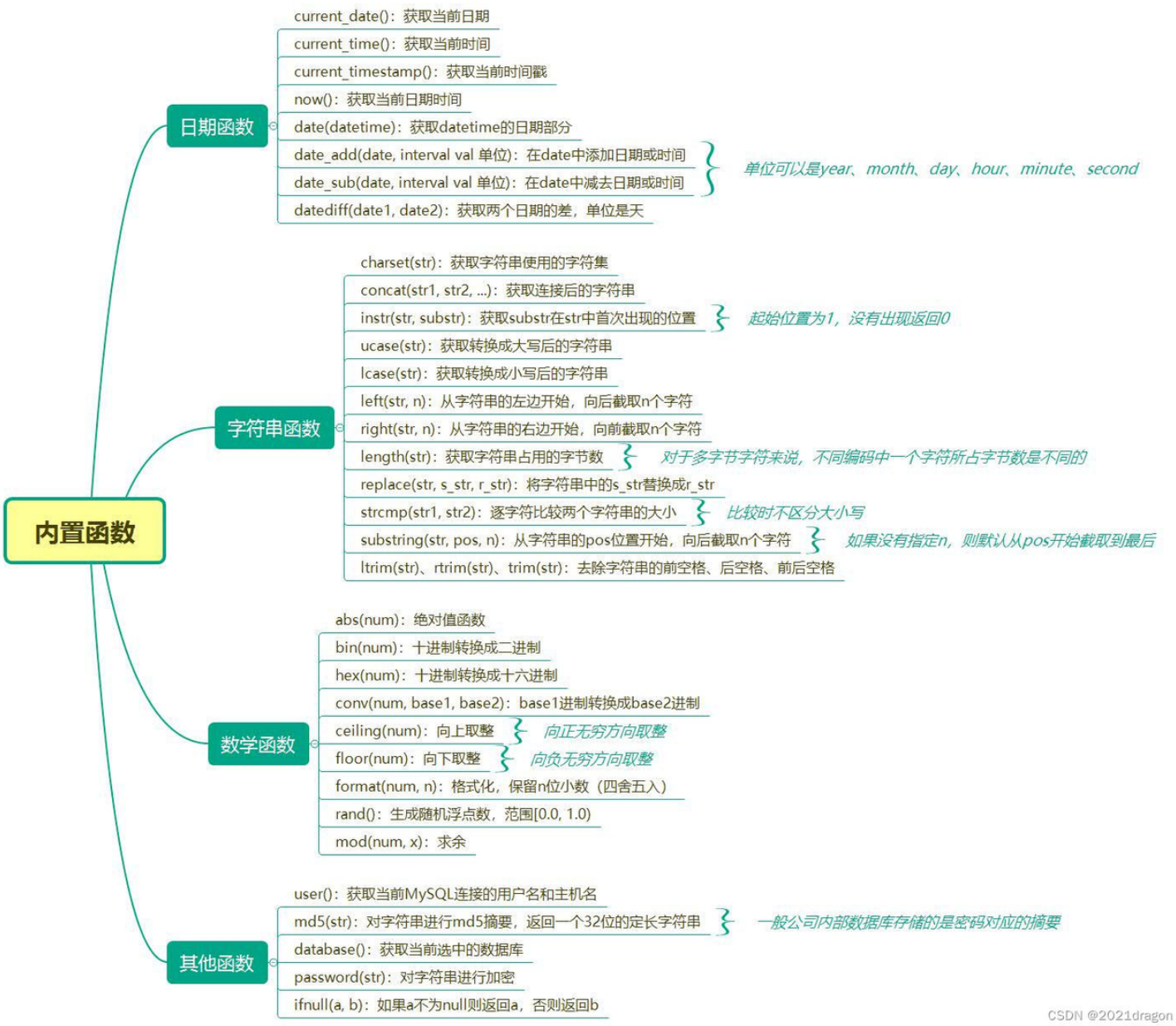


五、表的约束

六、基本查询（DML）

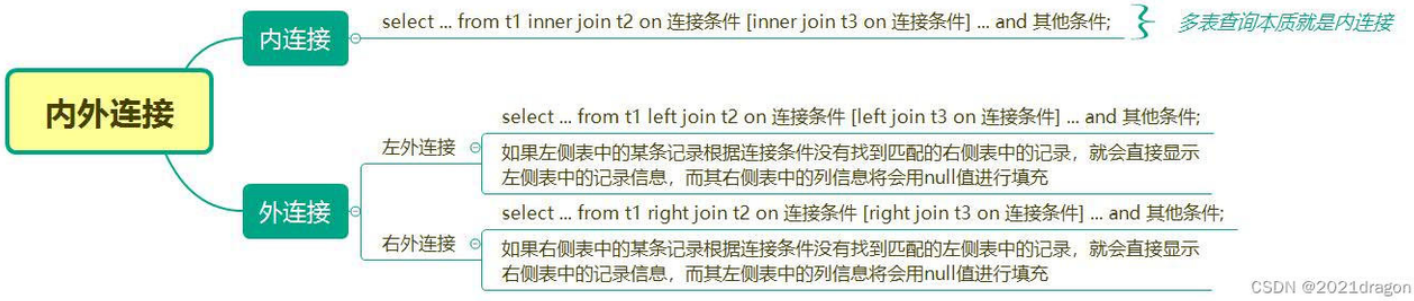


七、内置函数

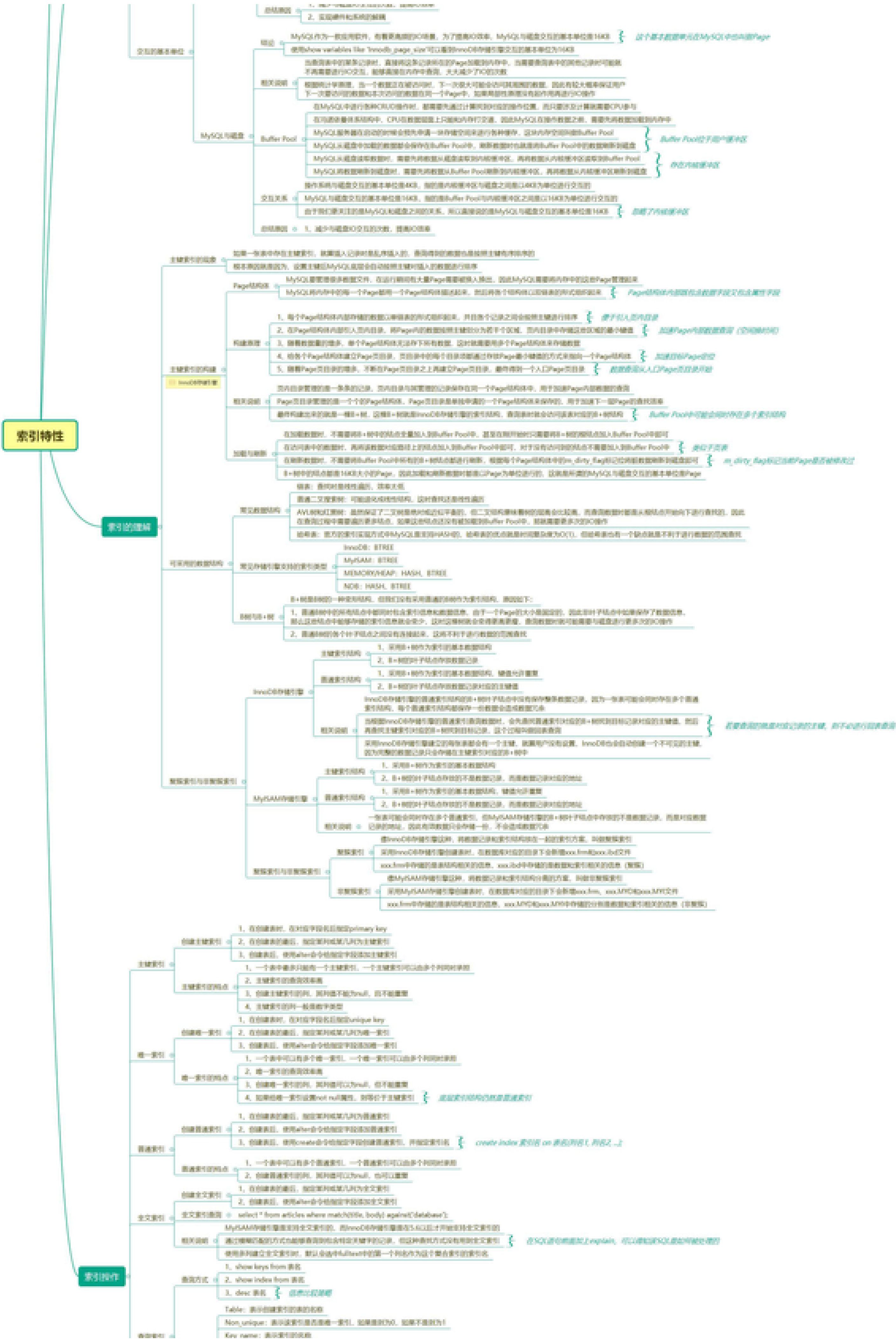


八、复合查询

九、内外连接



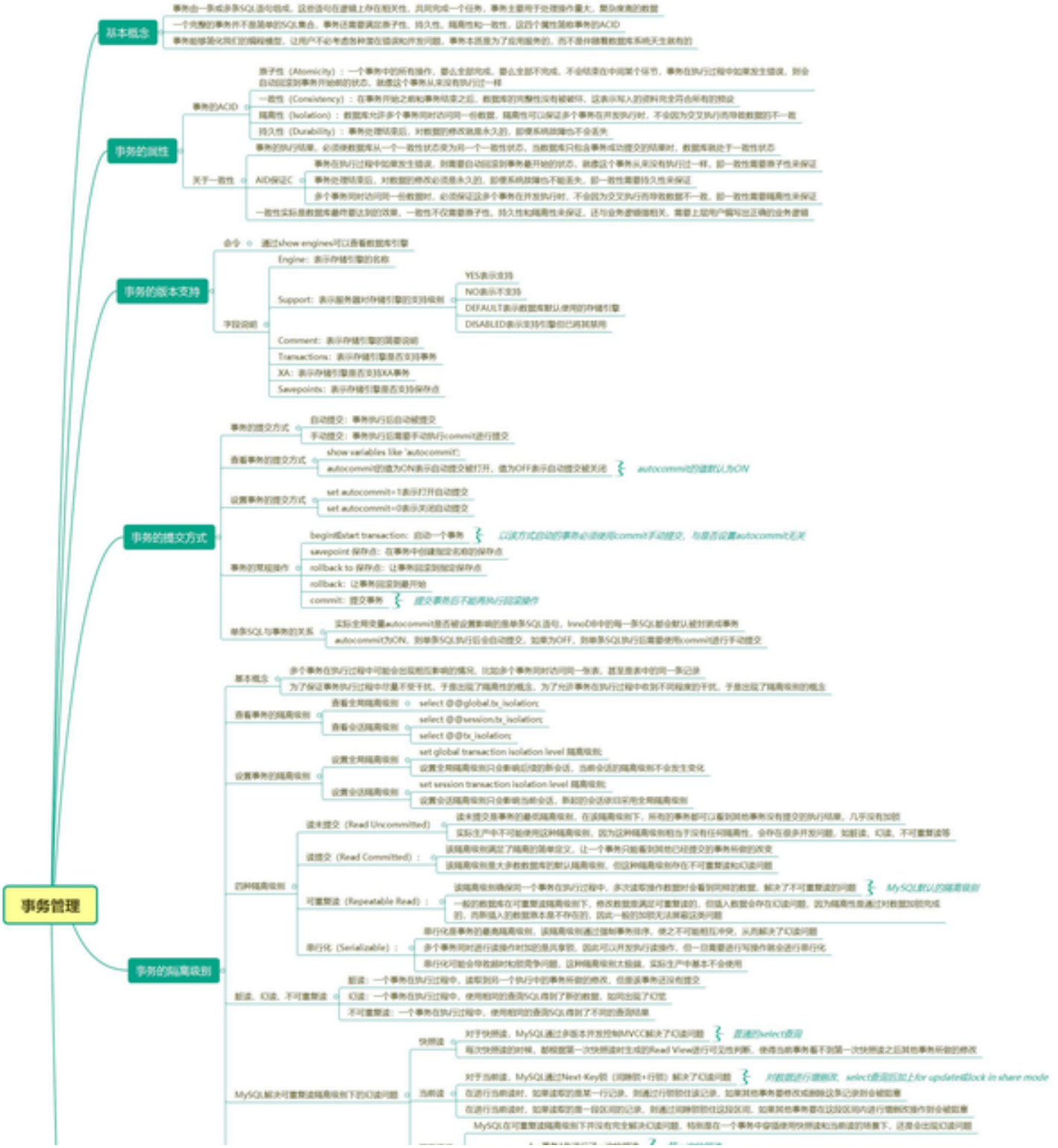
十、索引特性

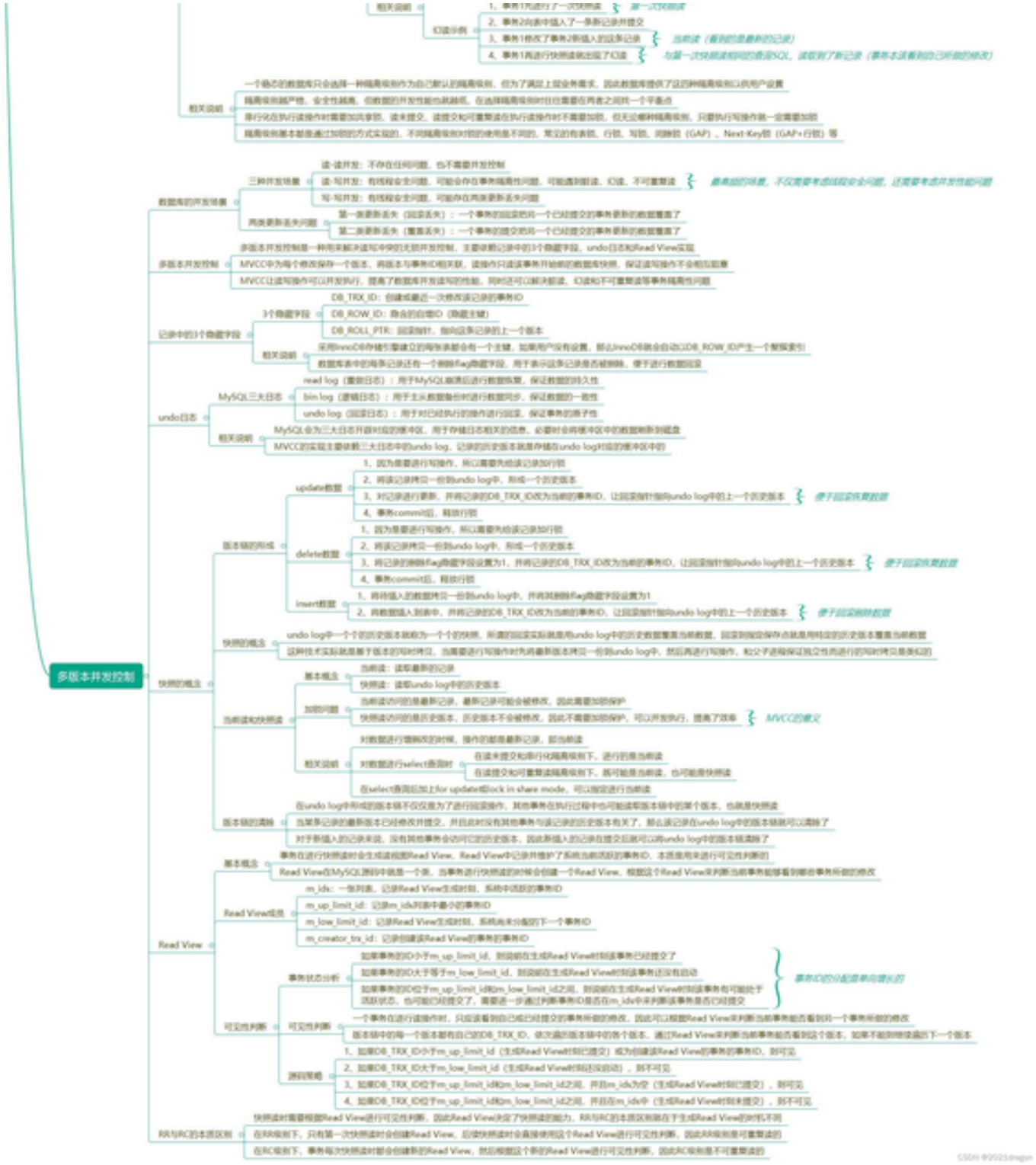


索引特性
交互的基本单位
MySQL与磁盘
Buffer Pool
索引的理解
主键索引的构建
InnoDB：BTREE
MyISAM：BTREE
MEMORY/HEAP：HASH、BTREE
NDB：HASH、BTREE
InnoDB存储引擎
MyISAM存储引擎
索引操作
主键索引
唯一索引
普通索引
全文索引
select * from articles where match(title, body) against('database');
1、show keys from 表名
2、show index from 表名
3、desc 表名
Table：表示创建索引的表的名称

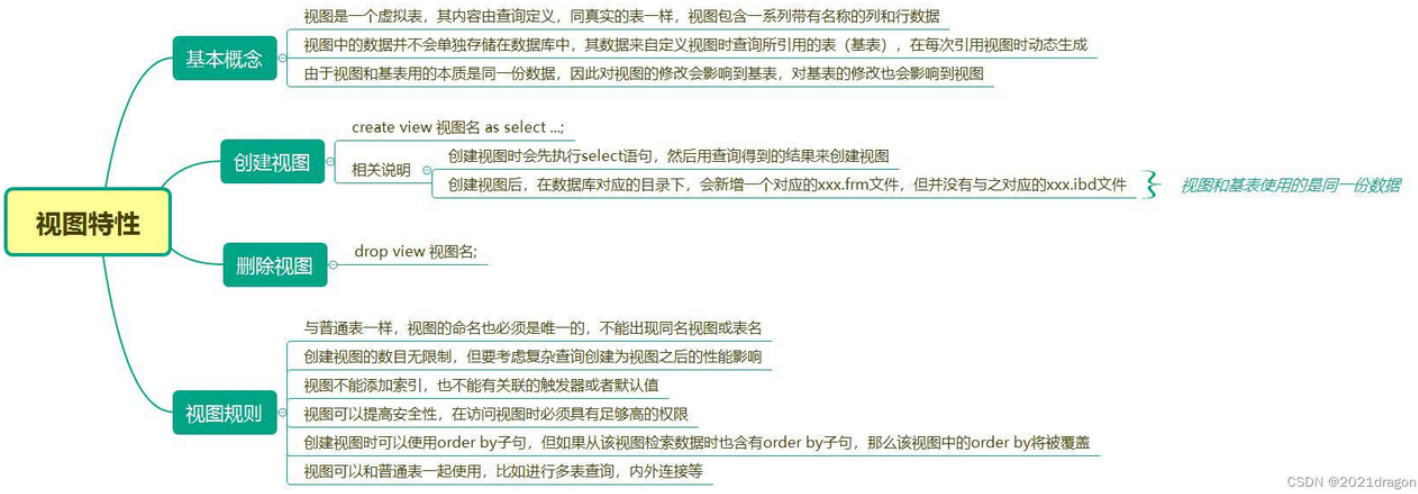

Seq_in_index：表示该列在索引中的位置
如果索引是单列的，则该列的值为1
如果索引是复合索引，则该列的值为每列在索引定义中的顺序
Column_name：表示定义索引的列字段
Collation：表示列以何种顺序存储在索引中，"A"表示升序，null表示无分类
Cardinality：索引中唯一值数目的估计值，基数越大，当进行联合时，MySQL使用该索引的机会就越大
Sub_part：表示列中被输入索引的字符的数量
若列只是部分被输入索引，则该列的值为被输入索引的字符的数目
若整列被输入索引，则该列的值为null
Packed：指示关键字如何被压缩，若没有被压缩，则值为null
Null：用于显示索引列中是否包含null值，若包含为YES，若不包含为NO
Index_type：显示索引使用的类型和方法（BTREE，FULLTEXT，HASH，RTREE）
Comment：显示评注
字段说明
删除主键索引
alter table 表名 drop primary key;
删除非主键索引
1、alter table 表名 drop index 索引名;
2、drop index 索引名 on 表名;
删除索引
一个表最多只能有一个主键索引，因此删除主键索引时不用指明索引名
一个表可能有多个非主键索引，因此删除非主键索引时需要指明索引名
相关说明
1、比较频繁作为查询条件的字段应该创建索引
2、唯一性太差的字段不适合单独创建索引，即使频繁作为查询条件
3、更新非常频繁的字段不适合创建索引
4、不会出现在where子句中的字段不应该创建索引
索引创建规则
创建索引的目的就是为了提高查询效率

十一、事务管理

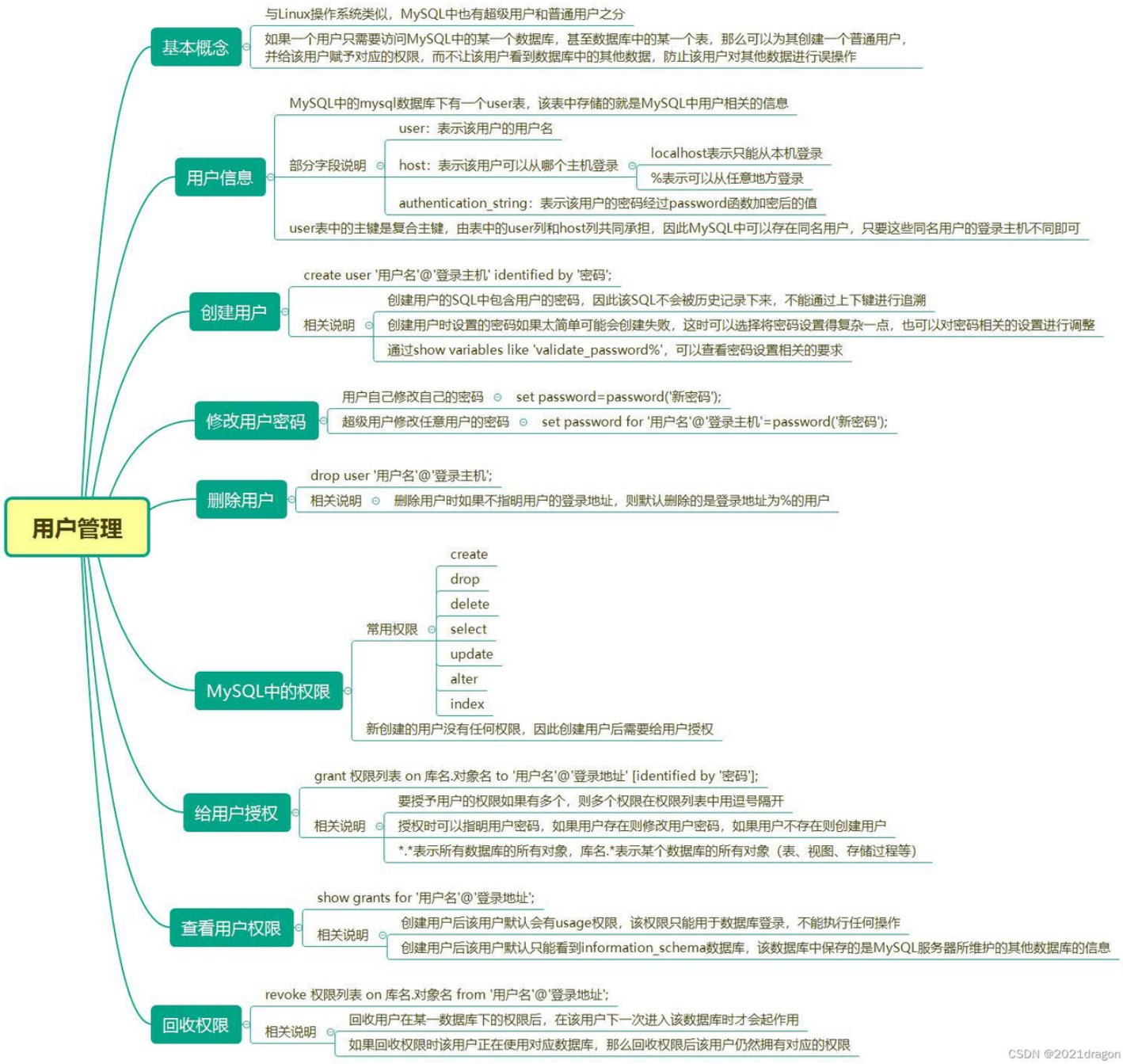

事务管理
基本概念
事务由一条或多条SQL语句组成，这些语句在逻辑上存在相关性，共同完成一个任务，事务主要用于处理操作量大，复杂度高的数据
一个完整的事务并不是简单的SQL集合，事务还需要满足原子性、持久性、隔离性和一致性，这四个属性简称事务的ACID
事务能够简化我们的编程模型，让用户不必考虑各种潜在错误和并发问题，事务本质是为了应用服务的，而不是仲裁着数据库系统天生就有的
事务的属性
事务的ACID
原子性（Atomicity）：一个事务中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节，事务在执行过程中如果发生错误，则会自动回滚到事务开始前的状态，就像这个事务从来没有执行过一样
一致性（Consistency）：在事务开始之前和事务结束之后，数据库的完整性没有被破坏，这表示写入的资料完全符合所有的预设
隔离性（Isolation）：数据库允许多个事务同时访问同一份数据，隔离性可以保证多个事务在并发执行时，不会因为交叉执行而导致数据的不一致
持久性（Durability）：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失
关于一致性
事务的执行结果，必须使数据库从一个一致性状态变为另一个一致性状态，当数据库只包含事务成功提交的结果时，数据库就处于一致性状态
ACID保证C
事务在执行过程中如果发生错误，则需要自动回滚到事务最开始的状态，就像这个事务从来没有执行过一样，即一致性需要原子性来保证
事务处理结束后，对数据的修改必须是永久的，即便系统故障也不能丢失，即一致性需要持久性来保证
多个事务同时访问同一份数据时，必须保证这多个事务在并发执行时，不会因为交叉执行而导致数据不一致，即一致性需要隔离性来保证
一致性实际是数据库最终要达到的效果，一致性不仅需要原子性、持久性和隔离性来保证，还与业务逻辑强相关，需要上层用户编写出正确的业务逻辑
事务的版本支持
命令
通过show engines可以查看数据库引擎
字段说明
Engine：表示存储引擎的名称
Support：表示服务器对存储引擎的支持级别
YES表示支持
NO表示不支持
DEFAULT表示数据库默认使用的存储引擎
DISABLED表示支持引擎但已将其禁用
Comment：表示存储引擎的简要说明
Transactions：表示存储引擎是否支持事务
XA：表示存储引擎是否支持XA事务
Savepoints：表示存储引擎是否支持保存点
事务的提交方式
事务的提交方式
自动提交：事务执行后自动被提交
手动提交：事务执行后需要手动执行commit进行提交
查看事务的提交方式
show variables like 'autocommit';
autocommit的值为ON表示自动提交被打开，值为OFF表示自动提交被关闭
autocommit的值默认为ON
设置事务的提交方式
set autocommit=1表示打开自动提交
set autocommit=0表示关闭自动提交
事务的常规操作
begin或start transaction：启动一个事务
以这方式启动的事务必须使用commit手动提交，与是否设置autocommit无关
savepoint 保存点：在事务中创建指定名称的保存点
rollback to 保存点：让事务回滚到指定保存点
rollback：让事务回滚到最开始
commit：提交事务
提交事务后不能再执行回滚操作
单条SQL与事务的关系
实际全局变量autocommit是否被设置影响的是单条SQL语句，InnoDB中的每一条SQL都会默认被封装成事务
autocommit为ON，则单条SQL执行后会自动提交，如果为OFF，则单条SQL执行后需要使用commit进行手动提交
事务的隔离级别
基本概念
多个事务在执行过程中可能会出现相互影响的情况，比如多个事务同时访问同一张表，甚至是表中的同一条记录
为了保证事务执行过程中尽量不受干扰，于是出现了隔离性的概念，为了允许事务在执行过程中收到不同程度的干扰，于是出现了隔离级别的概念
查看事务的隔离级别
查看全局隔离级别
select @@global.tx_isolation;
查看会话隔离级别
select @@session.tx_isolation;
select @@tx_isolation;
设置事务的隔离级别
设置全局隔离级别
set global transaction isolation level 隔离级别;
设置全局隔离级别只会影响后续的新会话，当前会话的隔离级别不会发生变化
设置会话隔离级别
set session transaction isolation level 隔离级别;
设置会话隔离级别只会影响当前会话，新起的会话依旧采用全局隔离级别
四种隔离级别
读未提交（Read Uncommitted）
读未提交是事务的最低隔离级别，在该隔离级别下，所有的事务都可以看到其他事务没有提交的执行结果，几乎没有加锁
实际生产中不可能使用这种隔离级别，因为这种隔离级别相当于没有任何隔离性，会存在很多并发问题，如脏读、幻读、不可重复读等
读提交（Read Committed）：
该隔离级别满足了隔离的简单定义，让一个事务只能看到其他已经提交的事务所做的改变
该隔离级别是大多数数据库的默认隔离级别，但这种隔离级别存在不可重复读和幻读问题
可重复读（Repeatable Read）：
该隔离级别确保同一个事务在执行过程中，多次读取操作数据时会看到同样的数据，解决了不可重复读的问题
MySQL默认的隔离级别
一般的数据库在可重复读隔离级别下，修改数据库满足可重复读的，但插入数据会存在幻读问题，因为隔离性是通过对数据加锁完成的，而新插入的数据原本是不存在的，因此一般的加锁无法满足这类问题
串行化（Serializable）：
串行化是事务的最高隔离级别，该隔离级别通过强制事务排序，使之不可能相互冲突，从而解决了幻读问题
多个事务同时进行读操作时加的是共享锁，因此可以并发执行读操作，但一旦需要进行写操作就会进行串行化
串行化可能会导致超时和资源争用问题，这种隔离级别太极端，实际生产中基本不会使用
事务的隔离级别
脏读、幻读、不可重复读
脏读：一个事务在执行过程中，读取到另一个执行中的事务所做的修改，但是该事务还没有提交
幻读：一个事务在执行过程中，使用相同的查询SQL得到了新的数据，如同出现了幻觉
不可重复读：一个事务在执行过程中，使用相同的查询SQL得到了不同的查询结果
MySQL解决可重复读隔离级别下的幻读问题
快照读
对于快照读，MySQL通过多版本并发控制MVCC解决了幻读问题
普通的select查询
每次快照读的时候，都根据第一次快照读时生成的Read View进行可见性判断，使得当前事务看不到第一次快照读之后其他事务所做的修改
当前读
对于当前读，MySQL通过Next-Key锁（间隙锁+行锁）解决了幻读问题
对锁索引行间隙区，select查询后加上for update或lock in share mode
在进行当前读时，如果读取的是某一行记录，则通过行锁锁住该记录，如果其他事务要修改或删除这条记录则会被阻塞
在进行当前读时，如果读取的是一段区间的记录，则通过间隙锁锁住这段区间，如果其他事务要在这段区间内进行增删改操作则会被阻塞
MySQL在可重复读隔离级别下并没有完全解决幻读问题，特别是在一个事务中曾经使用快照读和当前读的结果下，还是会出现幻读问题


多版本并发控制
数据库的并发场景
三种并发场景
读-读并发：不存在任何问题，也不需要并发控制
读-写并发：有线程安全问题，可能会存在事务隔离性问题，可能遇到脏读、幻读、不可重复读
写-写并发：有线程安全问题，可能存在两类更新丢失问题
最难的场景，不仅需要考虑线程安全问题，还需要考虑并发性能问题
两类更新丢失问题
第一类更新丢失（回滚丢失）：一个事务的回滚把另一个已经提交的事务更新的数据覆盖了
第二类更新丢失（覆盖丢失）：一个事务的提交把另一个已经提交的事务更新的数据覆盖了
多版本并发控制
多版本并发控制是一种用来解决读写冲突的无锁并发控制，主要依赖记录中的3个隐藏字段、undo日志和Read View实现
MVCC中为每个修改保存一个版本，将版本与事务ID相关联，读操作只读该事务开始前的数据库快照，保证读写操作不会相互阻塞
MVCC让读写操作可以并发执行，提高了数据库并发读写的性能，同时还可以解决脏读、幻读和不可重复读等事务隔离性问题
记录中的3个隐藏字段
3个隐藏字段
DB_TRX_ID：创建或最近一次修改该记录的事务ID
DB_ROW_ID：隐含的自增ID（隐藏主键）
DB_ROLL_PTR：回滚指针，指向该记录的上一个版本
相关说明
采用InnoDB存储引擎建立的每张表都会有一个主键，如果用户没有设置，那么InnoDB就会自动以DB_ROW_ID产生一个聚簇索引
数据库表中的每条记录还有一个删除flag隐藏字段，用于表示该条记录是否被删除，便于进行数据回滚
undo日志
redo log（重做日志）：用于MySQL崩溃后进行数据恢复，保证数据的持久性
MySQL三大日志
bin log（逻辑日志）：用于主从数据备份时进行数据同步，保证数据的一致性
undo log（回滚日志）：用于对已经执行的操作进行回滚，保证事务的原子性
相关说明
MySQL会为三大日志开辟对应的缓冲区，用于存储日志相关的信息，必要时会将缓冲区中的数据刷新到磁盘
MVCC的实现主要依赖三大日志中的undo log，记录的历史版本就是存储在undo log对应的缓冲区中的
版本链的形成
update数据
1、因为需要进行写操作，所以需要先给该记录加行锁
2、将该记录拷贝一份到undo log中，形成一个历史版本
3、对记录进行更新，并将记录的DB_TRX_ID改为当前的事务ID，让回滚指针指向undo log中的上一个历史版本
4、事务commit后，释放行锁
便于回滚恢复数据
delete数据
1、因为是要进行写操作，所以需要先给该记录加行锁
2、将该记录拷贝一份到undo log中，形成一个历史版本
3、将记录的删除flag隐藏字段设置为1，并将记录的DB_TRX_ID改为当前的事务ID，让回滚指针指向undo log中的上一个历史版本
4、事务commit后，释放行锁
便于回滚恢复数据
insert数据
1、将待插入的数据拷贝一份到undo log中，并将其删除flag隐藏字段设置为1
2、将数据插入到表中，并将记录的DB_TRX_ID改为当前的事务ID，让回滚指针指向undo log中的上一个历史版本
便于回滚删除数据
快照的概念
undo log中一个个的历史版本就称为一个个的快照，所谓的回滚实际就是用undo log中的历史数据覆盖当前数据，回滚到指定保存点就是用特定的历史版本覆盖当前数据
这种技术实际就是基于版本的写时拷贝，当需要进行写操作时先将最新版本拷贝一份到undo log中，然后再进行写操作，和父子进程保证独立性而进行的写时拷贝是类似的
快照的概念
当前读和快照读
基本概念
当前读：读取最新的记录
快照读：读取undo log中的历史版本
加锁问题
当前读访问的是最新记录，最新记录可能会被修改，因此需要加锁保护
快照读访问的是历史版本，历史版本不会被修改，因此不需要加锁保护，可以并发执行，提高了效率
MVCC的意义
相关说明
对数据进行增删改的时候，操作的都是最新记录，即当前读
对数据进行select查询时
在读未提交和串行化隔离级别下，进行的是当前读
在读提交和可重复读隔离级别下，既可能是当前读，也可能是快照读
在select查询后加上for update或lock in share mode，可以指定进行当前读
版本链的清除
在undo log中形成的版本链不仅仅是为了进行回滚操作，其他事务在执行过程中也可能读取版本链中的某个版本，也就是快照读
当某条记录的最新版本已经修改并提交，并且此时没有其他事务与该记录的历史版本有关了，那么该记录在undo log中的版本链就可以清除了
对于新插入的记录来说，没有其他事务会访问它的历史版本，因此新插入的记录在提交后就可以将undo log中的版本链清除了
Read View
基本概念
事务在进行快照读时会生成读视图Read View，Read View中记录并维护了系统当前活跃的事务ID，本质是用来进行可见性判断的
Read View在MySQL源码中就是一个类，当事务进行快照读的时候会创建一个Read View，根据这个Read View来判断当前事务能够看到哪些事务所做的修改
Read View成员
m_ids：一张列表，记录Read View生成时刻，系统中活跃的事务ID
m_up_limit_id：记录m_ids列表中最小的事务ID
m_low_limit_id：记录Read View生成时刻，系统尚未分配的下一个事务ID
m_creator_trx_id：记录创建该Read View的事务的事务ID
可见性判断
事务状态分析
如果事务的ID小于m_up_limit_id，则说明在生成Read View时刻该事务已经提交了
如果事务的ID大于等于m_low_limit_id，则说明在生成Read View时刻该事务还没有启动
如果事务的ID位于m_up_limit_id和m_low_limit_id之间，则说明在生成Read View时刻该事务有可能处于活跃状态，也可能已经提交了，需要进一步通过判断事务ID是否在m_ids中来判断该事务是否已经提交
事务ID的小规模单向增长的
可见性判断
一个事务在进行读操作时，只应该看到自己或已经提交的事务所做的修改，因此可以根据Read View来判断当前事务能否看到另一个事务所做的修改
版本链中的每一个版本都有自己的DB_TRX_ID，依次遍历版本链中的各个版本，通过Read View来判断当前事务能否看到这个版本，如果不能则继续遍历下一个版本
源码策略
1、如果DB_TRX_ID小于m_up_limit_id（生成Read View时已提交）或为创建该Read View的事务的事务ID，则可见
2、如果DB_TRX_ID大于m_low_limit_id（生成Read View时刻还没启动），则不可见
3、如果DB_TRX_ID位于m_up_limit_id和m_low_limit_id之间，并且m_ids为空（生成Read View时刻已提交），则可见
4、如果DB_TRX_ID位于m_up_limit_id和m_low_limit_id之间，并且在m_ids中（生成Read View时刻未提交），则不可见
RR与RC的本质区别
快照读时需要根据Read View进行可见性判断，因此Read View决定了快照读的能力，RR与RC的本质区别就在于生成Read View的时机不同
在RR级别下，只有第一次快照读时会创建Read View，后续快照读时会直接使用这个Read View进行可见性判断，因此RR级别是可重复读的
在RC级别下，事务每次快照读时都会创建新的Read View，然后根据这个新的Read View进行可见性判断，因此RC级别是不可重复读的

十二、视图特性



十三、用户管理



十四、C语言连接数据库