数据库知识一般在大的互联网企业对应届生不做硬性要求，对于小的互联网企业或社会人士一般有一定的要求。其要求一般包括：

1. 熟悉基本 SQL 操作

包括增删改查（insert、delete、update、select语句），排序 order by，条件查询（where 子语句），限制查询结果数量（LIMIT语句）等

1. 稍微高级一点的 SQL 操作（如Group by，in，join，left join，多表联合查询，别名的使用，select 子语句等）
2. 索引的概念、索引的原理、索引的创建技巧
3. 数据库本身的操作，建库建表，数据的导入导出
4. 数据库用户权限控制（权限机制）
5. MySQL的两种数据库引擎的区别
6. SQL 优化技巧

创建数据库： create database basename; 删除数据库 drop database basename;

创建表：

create table tablename

{

code varchar(20) primary key, //主键:每一行都应该有可以唯一标识自己的一列；一个顾客表可以使用顾客编号列；订单表可以使用订单ID

name varchar(20) not null  
};

create table ceshi

{

ids int auto\_increment primary key,

uid varchar(20),

name varchar(20),

class varchar(20),

foreign key (class) reference class (code) //外键

}；

drop table ceshi; //删除表ceshi

导入导出数据库：

导入: create database abc; use abc; source /home/abc/abc.sql //(导入数据库名)

导出: 使用命令mysqldump

导出数据和表结构：mysqldump -u 用户名 –p 密码 databasename > database.sql

只导出表结构: mysqldump –u 用户名 –p 密码 –d databasename > database.sql

增删改查：

select 语句:检索表的一个或多个列；检索所有列(select \* from products;) 去重(select distinct vend\_id from products;) 限定数量(select prod\_name from products limit 5)前5行

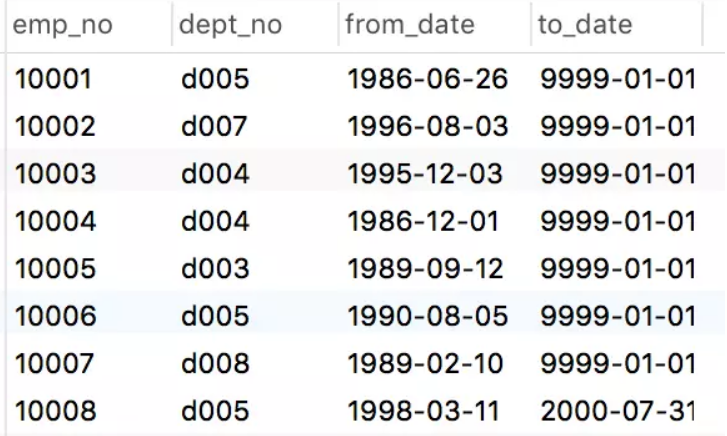
排序:select prod\_name from products order by prod\_name;(按prod\_name排序) 排序方向:order by prod\_price desc; order by prod\_price asc

数据过滤:select prod\_name, prod\_price from products where ven\_id = 1002 or vend\_id = 1003 order by prod\_name; 等价于 select prod\_name , prod\_price from products where vend\_id in (1002,1003) order by prod\_name;

select prod\_name, prod\_price from products where vend\_id not in (1002,1003) order by prod\_name;

常用聚合函数：count() ;sum(); avg();max();min();

group by 表示对by后面跟的字段进行分组，要配合聚合函数使用，分组后可以计数，求和，求平均



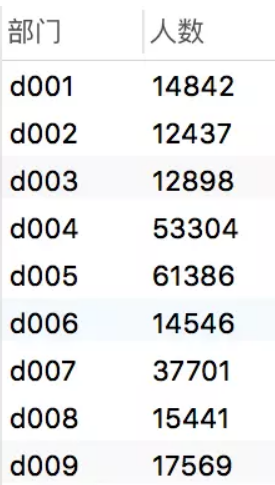
想知道每个部门有多少员工：对部门进行分组 group by dept\_no 对员工进行计数 count(emp\_no)

select dept\_no as 部门，count(emp\_no) as 人数

from dept\_emp

where to\_date = ‘9999-01-01’

group by dept\_no



group by 能替代distinct进行条件筛选

having:

having相当于条件筛选，但与where筛选不同,having是对于group by对象进行筛选

主键：某个属性组(可以是一个属性，也可以是多个属性组成一个主键)能唯一标识一条记录，不能有重复，不允许为空，这个属性称为主键；作用：用来保证数据完整性。主键只能一个

外键：表的外键是另一个表的主键，外键可以有重复，可以是空值；作用：用来和其他表建立联系。一个表可以有多个外键，保证数据一致性。

索引：该字段没有重复值，但可以有一个空值；作用：提高查询的速度。一个表可以多个唯一索引。

分解数据为多个表能够更有效的存储,更方便的处理，并且有更大的可伸缩性，但是有代价的。数据存储在多个表中，如何用单条select语句检索出数据？用联结(用一条select语句关联不同的表)

别名:除了用于列明和计算字段外，SQL还允许给表名起别名

联结(join)：mysql包括两种连接(内连接(inner join)和外连接(outer join))

什么是内联结(inner join 等值连接)，左外联结(left join)，右外联结(right join)？

内联结(inner join): 只返回两个表中联结字段相等的行。

左外联结(left outer join):除了匹配2张表中相关联的记录外，还会匹配左表中剩余的记录，左表中未能够匹配的，字段用NULL表示

右联结(right outer join): 除了匹配2张表中相关联的记录外，还会匹配右表中剩余的记录，左表中未能匹配到的，字段用NULL 表示

两张表goods表(id,goods\_name,price)和category表(id,cate\_name,goods\_id)

内联接(等值联接)

select \*

from goods inner join category

on goods.id = category.goods\_id

order by goods.id

左联接(查询结果中返回左表所有的行，右表中没有能够匹配左边的行，右表就用null表示)

select goods.\* , category.cate\_name

from goods left join category

on goods.id = category.goods\_id

order by goods.id

右联接(返回右表中所有的行，左表中没有能够匹配的行，左表就用null表示)

select a.goods\_name, a.price, b.\*

from goods as a

right join category as b

on a.id = b.goods\_id

order by b.id

自联结:需要为表定义别名;将一张表当作两张表

select b.\*

from shopping as a, shopping as b

where a.name = ”惠惠”

and a.price < b.price

order by b.id

mysql索引

索引的原理：索引是一个排序的列表，在这个列表中存储着索引的值和包含这个值的数据所在行的物理地址。索引可以加快查询速度，使用索引后可以不用扫描全表来定位某行的数据，而是通过索引项找到该行数据对应的物理地址然后访问相应的数据。

创建索引：

创建表时添加索引

create table tablename(

ID int not null,

username varchar(16) not null,

INDEX indexname (username(length))

);

创建表后添加索引：

alter table tablename add index indexname(column\_name);

create index indexname on tablename (column\_name);

创建索引时要对表加锁，因此实际操作中需要在业务空闲期进行。

根据索引查询：为column\_1建立了索引

具体查询： select \* from tablename where column\_1 = column\_2;

模糊查询： select \* from tablename where column\_1 = like ‘%aaa%’;

删除索引： drop index intexname on tablename;

mysql数据库的四类索引：

index：普通索引，数据可以重复，没有任何限制

unique：唯一索引，要求索引值必须唯一，但允许有空值；如果是组合索引，那么列值的组合必须唯一。

primary key：主键索引，一种特殊的唯一索引。一个表只能有一个主键，不允许有空值，一半是在创建表的同时创建主键索引。

组合索引：在多个字段上创建的索引，只有在查询条件中使用了创建索引时的第一个字段，索引才会被使用。

全文索引：对于大表的文本域：char,varchar,text列才能创建全文索引，主要用于查找文本中的关键字，并不是直接与索引中的值进行比较。

索引可以提高查询的速度，但是创建和维护索引需要耗费时间，同时也会影响插入的速度。如果需要插入大量的数据时，最好是先删除索引，插入数据后再建立索引。

索引生效条件：

假设index(a,b,c)

最左前缀匹配：模糊查询时，使用%匹配时：‘a%’会使用索引，’%a‘不会使用索引。

条件中有or索引不会生效。

a and c，a生效，c不生效。

b and c，都不生效。

使用索引的注意事项：

索引不会包含有NULL值的列。在设计数据库时不要让字段的默认值为NULL

使用短索引：对串进行索引，如果可能应该指定一个前缀长度。例如一个CHAR(255)的列，如果在前10个或20个字符内，多数值是唯一的，那么就不要对整个列进行索引。短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作。

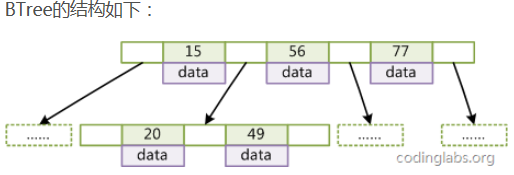
Like语句操作：一般情况下不鼓励使用like操作。如果非使用不可，如何使用也是一个问题。Like “%aaa%”不会使用索引而like”aaa%”可以使用索引。

不要在列上进行计算：select \* from users where YEAR(adddate)<2007。将在每个行上进行计算，这将导致索引失效而进行全表扫描。因此改成select \* from users where adddate<2007-01-01”

不使用NOT IN 和< >操作。

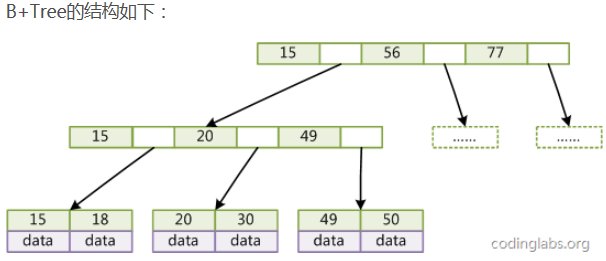
m阶B树：

每个中间节点都包含k-1个元素和k个子树，其中m/2 <=k<=m；所有叶子节点都在同一层。每个节点的元素都是索引的键和键对应表的每行所在的物理地址。



m阶B+树：

k个元素和k个子树，每个节点不保存数据，只存索引的键(B树中间节点也存键对应的表的行所在的物理地址)，所有键的对应行的物理地址都保存在叶子节点。树的所有叶节点构成一个有序链表，可以按照关键码排序的次序遍历全部记录。



B+树对比B树的优点：百度

磁盘读写代价更低：B+树的非叶子节点中不存储data，就可以存储更多的索引key。每个节点中的key越多，树的高度越小，需要的I/O次数越少。

查询速度更稳定：由于B+树非叶子节点不存储data，因此所有数据都要查询至叶子节点，而叶子节点的高度都是相同的，因此所有数据的查询速度都是一样的。

索引分为聚集(cluster)索引和非聚集(uncluster)索引两种

mysql中最常见的两种存储引擎分别是myisam(存储引擎)和innoDB，分别实现了非聚集索引和聚集索引。

索引分类中：用主键键值建立的索引称为主索引，其它称为辅助索引。因此主索引只有一个，辅助索引可以多个。

聚集索引：数据和主键索引存储在一起。

非聚集索引：数据和索引分开存储。

非聚集索引(myisam):

myisam存储引擎采用的是非聚集索引，非聚集索引的主索引和辅助索引几乎一样，只是主索引不允许重复，不允许空值。叶子节点存储指向键值对应的数据的物理地址。非聚集索引的数据表和索引表的分开存储的。

聚集索引(innoDB):

聚集索引的主索引的叶子节点存储的是键值对应的数据本身，辅助索引的叶子节点存储的是键值对应的数据的主键键值(找到主键键值后，再从主索引中查找数据本身)。

聚集索引的数据和主键索引存储在一起。因此，聚集索引的插入顺序最好按照主键的单调顺序插入，否则会引起频繁的页分裂，严重影响性能。

使用主索引时，更适合使用聚集索引，因为聚集索引只需查找一次，而非聚集索引在找到数据地址后，还要进行一次i/o查找。

聚集辅助索引存储的是主键的键值，因此不用维护辅助索引。但主索引存储的是数据本身，因此聚集索引会占用更多的空间。

聚集索引在插入新数据时比非聚集索引慢很多。因为要检查主键是否重复。同时主索引中存储数据占用空间大，因此需要更多I/O。

什么时候使用索引？

主键自动建立唯一索引

经常作为查询条件在WHERE或者ORDER BY(排序)语句中出现的列要建立索引

作为排序的列要建立索引

查询中与其他表关联的字段，外键关系建立索引

group by配合使用聚合函数的列可以建立索引，例如使用了max(col\_1)或者count(col\_1)时col\_1就要建立索引

什么时候不要使用索引？

经常增删改查的列不要建立索引

有大量重复的列不要建立索引

表记录太少不要建立索引(没意义)

mysql优化：

建立索引对查询进行优化；可以尽量避免全表扫描。

mysql的存储引擎

MyISAM不支持事务处理等高级功能。适用于高速存储和检索，以及全文搜索能力。适用于执行大量的select查询。不支持外键，表锁插入数据时，锁定整个表。查找总行数时，不需要全表扫描。

InnoDB用于需要事务处理的应用程序，包括ACID事务支持。如果应用种执行大量的insert或update操作，应该使用InnoDB。可以提高多用户并发操作的性能。支持外键，行锁。查找总行数时，全表扫描。

mysql事务：

几个SQL语句，要么全部执行成功，要么全部执行失败。数据库事务的三个常用命令:begin transaction, commit transaction, rollback transaction.

在mysql中只有使用了innodb数据库引擎的数据库或表才支持事务

事务处理可以用来维护数据库的完整性，保证成批的SQL语句要么全部执行，要么全部不执行。 事务用来管理insert,update,delete语句

事务的ACID: 原子性(atomic),一致性(consistency) 隔离性(isolation) 持久性(durability)

原子性: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。事务在执行过程中发生错误,会被回滚(rollback)到事务开始前的状态,就像这个事务从没执行过一样。

一致性: 指数据处于一种语义上的有意义且正确的状态。一致性是对数据可见性的约束，保证在一个事务中的多次操作的数据中间状态对其他事务不可见。因为这些中间状态，是一个过渡状态。与事务的开始状态和结束状态不一致。 (一致性保证要么事务前状态，要么事务后状态；原子性和一致性的侧重点不同:原子性关注状态，要么全部成功，要么全部失败，不存在部分成功状态。而一致性关注数据的可见性，中间状态的数据对外部不可见，只有最初状态和最终状态的数据对外可见)

隔离性:数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离型可以防止多个事务并发时由于交叉执行而导致数据的不一致 。(多个事务并发，不能干扰)

持久性: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

mysql事务隔离级别：

未提交读：允许脏读，其它事务只要修改了数据，即使未提交，本事务也能看到修改后的数据值。也就是可能读到其它会话中未提交事务修改的数据。

提交读：只能读到已经提交的数据。(解决脏读)

可重复读：无论其它事务是否修改并提交了数据，在这个事务中看到的数据值始终不受其它事务影响。(解决不可重复度)

串行读：每次读都需要获得表级共享锁，读写相互都会阻塞。(解决幻读)

mysql默认使用innodb引擎，默认使用可重复读

悲观锁和乐观锁：

悲观锁：每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据时就会block直到它拿到锁。传统关系型数据库用到了这种机制(行锁，表锁，读锁，写锁都是在做操作前先上锁)。它指的是对数据被外界修改持保守态度。因此，在整个数据处理过程中，将数据处于锁定状态。适用于多写的应用。

乐观锁：每次拿数据时认为别人不会修改，所以不上锁但是在更新时会判断一下在此期间别人有没有去更新这个数据，可以使用版本号(一般在数据表中加上一个数据版本号version字段，表示数据被修改的次数。当数据被修改时，version值加一。当线程更新数据值时，读取到数据的同时也会读取version值，在提交更新时)等机制。乐观锁适用于多读的应用类型，可以提高吞吐量。

数据库三大范式：作用：解决数据冗余，数据有效性检查，提高存储效率

第一范式： 表中的每个字段都不能再拆分(属性不可分)；

第二范式:满足第一范式并且表中的非主键字段都依赖于主键字段(完全依赖于主键，而不是部分依赖主键属性)

第三范式:满足第二范式并且表中的非主键字段必须不传递依赖于主键字段 (消除传递依赖)

脏读，不可重复读，幻读

脏读：事务T1将某一值修改，然后事务T2读取该值，此后T1因为某种原因撤销对该值修改，这就导致T2所读到的数据是无效的。(事务B读取了事务A还没提交的数据)

不可重复读：指在数据库访问时，一个事务范围内的两次相同查询却返回不同数据。在一个事务内多次读同一数据，在这个事务还没结束时，另外一个事务也访问同一数据，那么在第一个事务中的两次读数据之间，由于第二个事务的修改，第一个事务两次读到的数据可能是不一样的。这样在一个事务内两次读到的数据是不一样的，因此称为是不可重复读。(不可重复读重点在于update)(两次事务读取到的数据不一致)

幻读：指当事务不是独立执行时发生的一种现象。比如第一个事务对一个表中的数据进行了修改，这种修改涉及到表中的全部数据行。同时，第二个事务也修改这个表中的数据，这种修改是向表中插入一行数据。那么就会发生，操作第一个事务的用户发现表中还有没有修改的数据行，就好像发生了幻觉一样。(幻读的重点在于insert) (事务A修改了数据，事务B也修改了数据。这时在事务A看来，明明修改了数据，为什么不一样)

不可重复读&幻读区别：

如果使用锁机制来实现这两种隔离级别，在可重复读中，该sql第一次读取到数据后，就将这些数据加锁，其它事务无法修改这些数据，就可以实现可重复读了。但这种方法却无法锁住insert的数据，所以当事务A先前读取了数据，或者修改了全部数据，事务B还是可以insert数据提交，这时事务A就会发现莫名其妙多了一条之前没有的数据，这就是幻读，不能通过行锁来避免。需要串行化(表锁)隔离级别，读用读锁，写用写锁，读锁和写锁互斥，这样做可以有效的避免幻读，不可重复读，脏读等问题，但会极大降低数据库的并发能力。

关系模式R=(A,B,C,D,E)中的函数依赖集F= {A->B,B->C,CE->D},这是第几范式？

首先要确认主键：A->B,且B->C，故A->C ; A->C且CE->D，故AE->D ;故AE->ABCD即AE联合起来能够确定一条记录，故AE是联合主键

A->B且B->C，存在传递，排除3NF；不存在E->B，但有A->B，而主键是AE，故存在非主属性部分依赖于主属性，排除1NF

数据库约束(用于保证数据库表数据的完整性，正确性和一致性)：

约束可以分为：主键约束(primary key) 外键约束(foreign key) 唯一约束(unique) 检查约束(check) 空值约束(not null) 默认值约束(default)

主键：能唯一确定一条记录的唯一标识（字段不能重复，不能为空）；用来保证数据完整性

外键: 用于与另一张表关联。能确定另一张表记录的字段，用于保持数据的一致性(A表中的一个字段，是B表的主键，那么A表中的这个字段可以是A表的外键)；即表的外键是另一个表的主键，外键可以有重复，可以是空值； 用来和其他表建立联系

如何在删除主表记录时，一并删除从表相关联的记录？

如果两张表存在主外键关系，那么在删除主键表的记录时，如果从表有相关联的记录，那么将导致删除失败。所以在定义外键约束时，可以同时指定3种删除策略：1)将从表记录一并删除(级联删除)。2)将从表记录外键字段设置为NULL 。3)将从表记录外键字段设置为默认值

常用的聚合函数：

sum(求和) avg(平均数) max(最大值) min(最小值) count(记录数)

什么是视图？

char和varchar区别:

char长度固定为创建表时声明的长度，长度值范围在1到255。varchar长度不固定

当char值被存储时，它们被用空格填充到特定长度。检索char值时需删除末尾空格

mysql哪几种锁？

MyISAM支持表锁，InnoDB支持表锁和行锁，默认为行锁

表级锁：开销小，加锁快，不会出现死锁。锁粒度大，发生锁冲突概率高，并发量低

行级锁：开销大，加锁慢，会出现死锁。锁粒度小，发生锁冲突概率小，并发度高。

存储过程：

我们常用的SQL语句在执行时需要先编译，然后执行。而存储过程是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数(如果该存储过程带有参数)来调用执行它。

一个存储过程就是一个可编程的函数，它在数据库中创建并保存。它可以有SQL语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的函数，或者封装特定功能时，存储过程是非常有用的。

优点：

存储过程被创建后，可以在程序中被多次调用。

存储过程能实现较快的执行速度。因为存储过程是预编译的，优化器对其进行分析优化，并且给出最终被存储在系统表中的执行计划。而批处理的SQL语句每次运行都需要进行编译和优化，速度相对要慢一些。

存储过程能减少网络流量：当客户计算机调用存储过程时，网络中传输的只是调用语句。

触发器：一张表发生了某件事(插入，删除，更新操作)，然后自动触发了预先编写好的若干条SQL语句的执行。

特点：触发时间的操作和触发器里的SQL语句是一个事务操作，具有原子性，要么全部执行，要么不执行。

delete,drop,truncate区别:

truncate和delete只删除数据，不删除表结构，drop删除表结构，并且释放所占用的空间

delete需要事务管理,commit之后才能生效。drop和truncate操作立即生效，不可回滚

使用场合：

当你不再需要该表时，用drop

当你仍要保留该表，但要删除所有记录时，用truncate

当你需要删除部分记录时，用delete(always with a where clause)

数据库查询优化：

对查询优化，尽量避免全表扫描。首先应考虑在where及order by涉及的列上建立索引

尽量避免在where子句中使用!=或<>操作符，否则索引失效而进行全表扫描。

尽量避免在where子句中对字段进行null值判断。否则索引失效进行全表扫描。

尽量避免在where子句中使用or来连接条件，否则索引失效进行全表扫描。

最左匹配规则： select id from t where name like ‘aaa%’。先字符串后接%,否则索引失效

尽量避免在where子句中对字段进行函数操作，否则索引失效，全表扫描。

mysql一条语句的执行过程：

两个字段(name,age)建立联合索引