数据库是一个以某种结构存储数据的容器，类似一个数据结构，只不过数据结构存储在内存中，而数据库最终会持久化到文件系统中。

关系型数据库都要表单Table的概念，非关系型数据库一般没有表单概念或有弱表单概念。

ORM(Object Relational Mapping)，对象关系映射：面向对象是从软件工程基本原则发展而来，关系数据库则是从数学理论发展而来，两套理论有显著的区别，为了解决不匹配显著，ORM就应运而生。

ORM：将数据库中table中的数据变成面向对象中的class，将class中的数据保存到table中，这样就不需要通过sql语句来实现了。

尽管有了ORM，但仍然需要学习sql，一方面ORM有一定的缺陷，功能不一定完善，另一方面，学习sql可以更好了解什么样的查询会产生瓶颈，什么样的查询更有效率。

表单Table：存储一类数据，不同类别的数据放在不同的表中。

Table中没有继承的概念，如果希望实现类似面向对象中继承的行为，可以在表单中增加一个外键，存储表单中的对象需要继承的表中的id。

列(column)：表中的一个字段，所有表都是有一列或多列组成的。一列包含相同类型的数据。

行(row)：表中的一条记录，是一组相关的数据。

sql(Structured Query Language)：专门用来和关系数据库沟通的语言，用于存储，查询，更新和管理数据库。

表单设计：

主键(Primary Key)：用于唯一标识一条数据的field(可以是一个field，也可以是多个，可以把好几个field联合作为主键，但不推荐这么做)

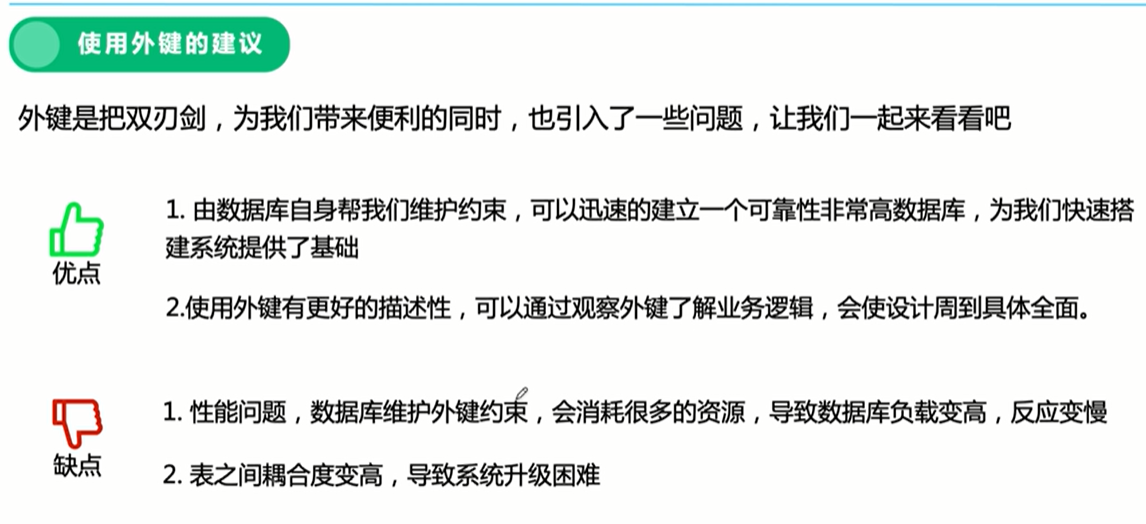
外键(Foreign Key)：存储两个表之间的关联关系。把外键作为主关键字的表称为主表，具有此外键的表称为从表。

外键插入：

插入准则：在Tweet表中插入数据的user\_id必须对应user表中id的某个值，否则会报错。

阻止执行：

1. 从表插入新行，其外键值不是主表的主键值便阻止插入；
2. 从表修改外键值，新值不是主表的主键值便阻止修改；
3. 主表删除行，其主键值在从表里存在便阻止删除（要想删除，先把从表中的相关行删除掉）；
4. 主表修改主键值，旧值在从表里存在便阻止修改（要想删除，必须先删除从表中的相关行，注意，这里是删除而不是修改，因为主键新值之前肯定不在主表中(主键唯一)，所以无法先在从表中修改，参考第2条）



SQL中的join在工作中一般不推荐，效率低。

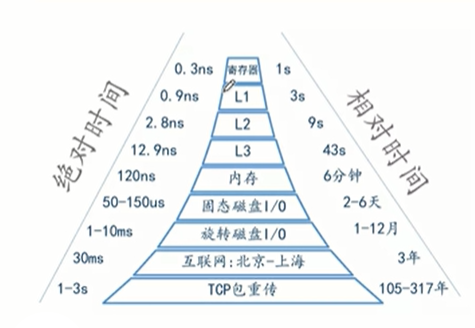
数据库中日期时间一般用UTC时间。

数据库中会有冗余存储，即多个表中都会存储某些相同的字段，这是为了保证查询的效率。

数据库的增删查改：

数据库索引：

各种介质的访问速度：



其中相对时间是把寄存器的访问时间当成1s来看。

以MySQL为例，MySQL进行数据存储，是以页为单位把数据存储在硬盘上(页大小为16KB)。查询时，把页从硬盘读取到内存中，再进行查询。如果写的sql语句进行全表扫描的话，假如表非常大，扫描速度会非常漫长，原因有两点：

1. 硬盘和内存之间的速度鸿沟
2. 页的非顺序存储

硬盘和内存之间的速度鸿沟没法避免，所以要减少读取时间，就要减少读取页的数量，避免读入一些无用的页。数据库中的索引就是用来减少页的读取量的。

以下面的sql语句为例：



可以给created\_at字段创建一个索引，所谓的索引就是在原来表的基础上比如tweet表，创建了一个新的表(索引其实就是一个表，但看不到，是数据库为你创建的)。这种索引表会按照created\_at作为第一关键字进行排序，这样查询效率会大大提高，不会再进行全表扫描了。

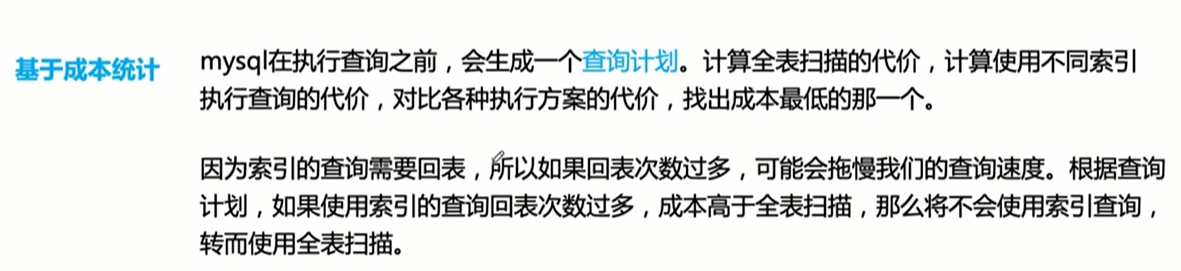
如果只创建created\_at的索引，假如sql语句中添加了user\_id=1，之前的索引表就会失效，因为user\_id=1的created\_at有可能是不连续的。

这时候就可以创建一个联合索引，先根据user\_id排序，再根据created\_at进行排序。

面试之前可以重点看看B+树的性质。

explain：explain可以查看执行计划，可以对比有联合索引和没有联合索引，执行sql语句的区别。

索引的创建根用户的查询需求有关，用户最常用的查询来建立索引。



索引的失效：

1. like通配符，将%放大左侧，会导致索引失效。



1. 左前缀原理，联合索引是根据字段顺序进行排序的，当我们使用联合索引时，必须根据联合索引的字段顺序设置where查询子句。

例如，为(user\_id, created\_at)创建一个联合索引，下面这条语句，索引是有效的：



下面这条语句，索引是无效的：只包含了created\_at字段



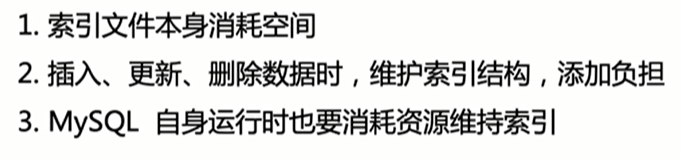
下面这条语句也会导致索引部分失效：

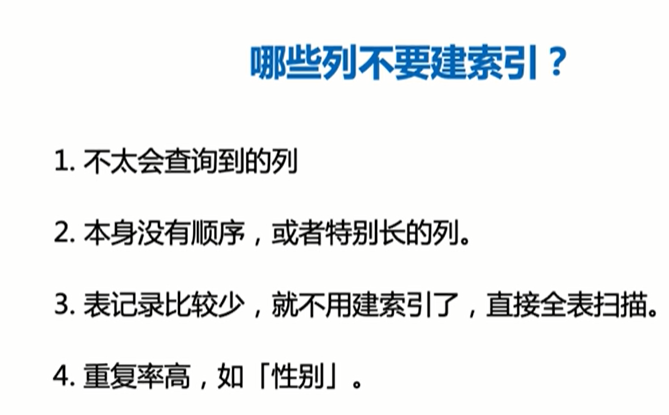


它只会在左侧的user\_id用到范围索引的情况，右侧的字段无法使用索引。因为联合索引先根据user\_id排序，再根据created\_at进行排序，查询时user\_id < 10的所有记录的created\_at并不一定是有序的(比如user\_id为1的用户在2021年中发了feed，而user\_id为2的用户在年初发了feed，显然这时的created\_at不是有序的)，所以导致右侧字段无法使用索引。

如果真的有需求对两个字段都进行范围查询的话，可以考虑把两个字段想办法合成一个字段。

索引的缺点：





尤其是重复率高的列，建了索引对效率也没什么提升。