一、数据库选择

1. NoSQL——Not only sql

① 特点：

a. NoSQL 具有灵活的数据模型， 可以处理非结构化/半结构化大数据

b. 海量数据高扩展性：关系型数据库的默认扩展方式就是向上扩展。为了支持更多的并发用户以及存储更多的数据，你需要越来越好的服务器，更好的 CPU、更多的内存、更大的磁盘来维护所有表, 服务器的不足和浪费等都会导致性价比会降低。一种解决办法是采用水平扩展，需要根据功能等方面进行分片，并分好全局数据，分别存储到各个服务器上，过程复杂。而 NoSQL 数据库从一开始就是分布式、水平扩展的，且通常都支持自动分片，意味着他会自动实现负载均衡，且即使某台服务器宕机，也会被快速且透明的替换掉。

c. 敏捷开发：关系型数据库需要在添加数据前定义好模式，且模式的改变开销极大。而 NoSQL 更为灵活

d. 灾难恢复： NoSQL 数据库支持自动复制，这意味着你可以获得高可用性与灾备恢复功能。从开发者的角度来看，存储环境本质上是虚拟化的。

e. 不够成熟， 学习成本高

f. 高并发性

② 分析：

NoSQL 比较适用的场景，数据库表 schema 频繁变化；数据库表字段是复杂数据类型；高并发数据库需求；海量数据的分布式存储

以上几点其实我们的企业应用设计都不是要求很高，所以我觉得其实没有很大必要去使用 NoSQL，而且对于非结构化数据以及半结构化数据也没有很多的涉及。在资料中有提到说将 NoSQL 和传统关系型数据库结合，用 NoSQL 来弥补关系型数据库的性能上的不足。

2. PostgreSQL: 开源，至今仍在更新

① 特点：

a. 自带全文检索功能

b. 支持服务器端脚本: TCL, Python, R, Perl, Ruby, MRuby ... 自带 map-reduce 了.

c. 可以把 70 种外部数据源 (包括 Mysql, Oracle, CSV, hadoop ...) 当成自己数据库中的表来查询

d. PostgreSQL 的稳定性极强， Innodb 等引擎在崩溃、断电之类的灾难场景下抗打击能力有了长足进步

e. 有非常丰富的统计函数和统计语法支持，比如分析函数

f. 多种数据类型和索引的支持

g. 排名很高

②分析：

我觉得可以在 PostgreSQL 和 MySQL 中选择一个，百度了一下 ERP 岗位招聘的信息，几乎公司里都还用的是 Oracle, SQL Server， MySQL 这种， 不过感觉 PostgreSQL 也蛮有诱惑力的哈哈。

3. openerp: 一个可以自己通过源代码修改、裁剪的 erp 系统

4. SQLite 中小型数据库

SQLite 的兴起跟 Android 有很大的关系，安卓中用户短信、联系人数据等都是用 SQLite 存储的。甚至，我们熟悉的 QQ、迅雷等，也都是采用 SQLite 来存储数据。其性能的优势来源于缺少的功能。

5. MySQL

中型数据库, 很多企业的 ERP 系统和网上商城系统用的都是这两种。

二、 数据库设计：

1. 采用自增长的 primary key
    ① 为了性能上的优化，数字的比较远远优于字符串的比较，可大大提高查找速度
    ② 避免使用特定编号出现的冲突问题，例不同国家的身份证号可能会冲突
    ④ 避免“千年虫”软件问题，比如当初设计表的时候设计的位数太短
2. 避免使用复合主键：
    只比较一个字段比多个字段快得多，可以考虑多用 id (之前讨论的时候我说可以用多个字段来代表主键，看来还是有点问题的。)
3. 采用双主键：
    一个系统随机生成的主键 SID, 一个 login id,为了防止删除后增加同样的 login id 造成的混乱。
    具体应用场景可能还不太理解，需要再查一下？？？？？？？？？？？？？？？？？、
4. 以固定的数据库和表应付客户固定的变化需求
    如果允许用户在使用过程中自动生成新的表格将会给后期 ERP 系统升级带来很大的困难，有很多的不可掌控性。
    早期做数据库，可能存在一种情况，对于公司每年度的数据，都会新建一张表，用来提高操作速度，但是做跨年数据分析的时候就会变得很费力。现在的 Oracle 数据库等大多数数据库系统都可以做到在常数的时间内返回一定的数据，根据 primary key 在 100 万条数据中取 10 条数据和在 1 亿条数据中取 10 条数据，在时间上并没有太大差别。
5. 避免一次取数据库大量数据，在取大量数据时，一定要进行分页
    基本上是很多数据库设计的基本准则，大规模数据的一次取出会导致服务器长时间处于停滞状态，从而影响其他在线用户的响应速度。
    SQL 分页主要用于控制大数据量表在前台(报表，表格，列表框之类)的分页显示。这样做不仅能够提高查询效率，而且还能是前台看起来比较简洁。
    对于不同的数据库管理系统，有不同的实现分页查询的方法，比如说对于 MySQL，实现分页查询的语句如下：
    select * from table WHERE … LIMIT 0,10 （即返回 0-10 行数据，内部具体如何分页我们不做考虑，是底层的东西）

三、 数据库系统的 C/S 和 B/S 结构

B/S（Browser/Server）结构即浏览器和服务器结构。客户机不需要安装任何特定的软件，只需要有一个浏览器就可以就可以与服务器进行交互,。如很多采用 ASP、PHP、JSP 技术的网站就是典型的基于 B/S 体系结构的。

C/S（Client/Server）结构即客户端和服务器结构。这里的客户端从硬件上可以理解为客户机，从软件上可以理解为在客户机上安装的特定专用软件，便于理解本文以后提到的客户端均指在客户机上安装的特定专用软件。也就是说基于 C／S 体系结构的软件需在客户机上安装特定的专用软件才能访问服务器，如腾讯 QQ、网络游戏客户端等就是典型的基于 C／S 体系结构的。

个人倾向于 B/S 设计。感觉 C/S 中客户端软件破解问题，与服务器非法连接听起来还挺可怕的。而且更新成本也要高一些。

四、 参考资料

[1] MySQL, SQLServer, Oracle 分页查询语句参考博客:
https://blog.csdn.net/guyong1018/article/details/2183057
[2] 数据库系统 C/S 与 B/S 结构具体差别参考博客：
https://blog.csdn.net/jhobby/article/details/1617886
[3] NoSQL 和传统关系型数据库的对比：
https://www.cnblogs.com/wyy123/p/6053341.html
[4] PostgreSQL 和 MySQL 的对比：
https://www.zhihu.com/question/20010554