字节一面：20亿手机号存储选int还是string？varchar还是char？为什么？

原创 捡田螺的小男孩 捡田螺的小男孩 2025年04月28日 09:04 广东

前言

大家好，我是**田螺**。

最近一位星球粉丝说，他去面试了字节，问了这么一道题，20亿手机号存储，选int还是string？varchar还是char？为什么？

他支支吾吾回答了几句，好像看起来，**面试官面色凝重，对他不是很满意**，果然最好还是挂了。。。

本文跟**大家聊聊我的思路**。

- 20亿数据，用Int存储存在哪些问题？
- 面试官的隐藏考察点
- 日常开发避坑点

1. 20亿数据，用Int或者BigInt能有在哪些问题？

1.1 int存得下11位数字嘛？

首先，我们都知道手机号，**是11位的数字**，比如 `13728199213`.

在Java中，int是 32位，最大值为 `2^31 - 1 = 2,147,483,647` 。约等于 2×10^9。显然，如果用int，**根本存不下 11位的手机号码**。

要想存得下，**得用64位的Long类型**，也就是对应数据库的bigInt。

1.2 数据完整性

例如手机号01324567890，用Long存会变成1324567890，直接破坏数据完整性。

```
Long phoneNumber =01324567890L; //编译报错，Java不允许前导0的Long整数
```

并且，有时候，有些手机号可能包含国家代码如 `(+86)` ，或者有些时候，是有连字符的，比如 `137-2819-9213` . 这些原因都导致不能用整型类型存储。

1.3 查询麻烦

比如，你要查找，手机号是 137 开头的手机号号码，如果用BigInt(Long类型)需先转字符串再模糊匹配，效率暴跌。

2. 用String有哪些好处

- **保真**：数字、符号、前导零全能存，原样保留。
- **灵活**：支持模糊查询、国际号码，扩展无忧。
- **省心**：无需担心溢出或格式转换问题。

```
CREATE TABLE user_tab (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '用户ID',
  phone_number VARCHAR(20) NOT NULL COMMENT '手机号',
  PRIMARY KEY (id),
  UNIQUE KEY idx_phone (phone_number)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='用户表';
```

2. 面试官的隐藏考察点

面试的时候，面试官主要考察候选人的一些**业务扩展性、数据容错性、思考问题全面性**等能力。我们先通过：为什么用 VARCHAR(20) 而不是 VARCHAR(11)，**来给面试官秀一波肌肉~~**

2.1 为什么用 VARCHAR(20) 而不是 VARCHAR(11)

我们就拿手机号来说，为什么更建议用 VARCHAR(20)，而不是VARCHAR(11)呢？

因为我们都知道，手机号是 11 位的，为什么不直接用 VARCHAR(11) 呢？

如果你日常开发中，就有**思考数据容错性习惯**的话，就会想到：

- 如果遇到国际号码：+8613822223333（14位）
- 带国家码的号码：008613822223333（15位）
- 分机号：13822223333#123（超11位）

这些场景，都会导致 VARCHAR(11) 报错崩盘。

其次就是**业务扩展性思考**：VARCHAR(11)只能存纯11位数字，假设未来业务需要：

- 支持座机号（如010-62223333，含横杠）
- 支持虚拟号（如17012341234-5678）
- 支持其他登录方式（如邮箱+手机号混合存储）

因此，**字段长度和类型需提前为业务变化留余地，避免频繁改表**。这就是日常开发中的，业务扩展性思维思考。

还有**数据容错性思考**，

- **输入不可控性**：用户可能输入带空格/符号的号码（如138 2222 3333），直接存原始值更方便清洗。
- **设计妥协**：若强制用VARCHAR(11)，需在代码层严格过滤非数字字符，增加复杂度。

还有**思考问题全面性，比如存储成本思考**。

- **VARCHAR(11)**：最大占 11字节（utf8mb4下1字符占4字节，但数字和+号只占1字节）
- **VARCHAR(20)**：最大占 20字节
- 20亿数据相差仅约 18GB（和用BIGINT的16GB对比，总成本仍可接受）。

所以**面试官期待的答案公式**

```
合理长度 = 基础需求 + 国际扩展 + 容错缓冲
```

当然，**这个不是固定答案，主要还是面试的时候，你回答面试官的思路和表达**，最好体现你有这几个方面的思考：**业务扩展性、数据容错性、思考问题全面性。**

2.2 极端场景

如果手机号是纯数字，并且第一位不是0的话，可以用BIGINT的，但是永远不要使用INT。通过这些极端场景的举例，**也体现你思考问题全面性的一个能力**。

3. 日常开发避坑点

设计手机号存储的时候，有哪些需要避的坑的。

主要有这几个吧：

3.1 字段长度设计过小

用 VARCHAR(11) 只存纯数字，遇到 +8613822223333（14位）直接截断。

> 用 VARCHAR(20) 兼容国际号、分机号（如 13822223333#123）。'

3.2 字符集和排序规则

使用 utf8 字符集，无法存储 emoji 或特殊符号

> 用 utf8mb4 + utf8mb4_unicode_ci，兼容所有 Unicode 字符（如 + * #）。

3.3 索引设计不当

未对手机号加唯一索引，导致重复数据。

```
添加 UNIQUE 约束：ALTER TABLE user ADD UNIQUE INDEX idx_phone (phone);
```

3.4 数据清洗与校验缺失

用户输入 138-2222-3333 或 138 222 23333，直接存储导致格式混乱。

> - 入库前统一清洗：移除空格、横杠等符号，只保留 + 和数字。
> - 正则校验：例如 ^+?\d{8,20}$（允许带 + 号的 8~20 位数字）。

3.5 忽视隐私与安全

明文存储手机号，泄露用户隐私。

> - 加密存储：使用 AES 加密或数据库内置加密函数。
> - 脱敏显示：查询结果返回 138****3333。

3.5 风控校验

```
// 严格校验（11位纯数字，无国际码）
String regex = "^1(3[0-9]|4[579]|5[0-35-9]|6[2567]|7[0-8]|8[0-9]|9[0-35-9])\\d{8}$";

// 宽松校验（允许带国际码，如+86 13812345678）
String looseRegex = "^(\\+\\d{1,3})?1(3\\d|4[579]|5[0-35-9]|6[2567]|7[0-8]|8\\d|9[0-35-9]
```

捡田螺的小男孩
专注后端技术栈
266篇原创内容

公众号

4. 最后

> 坚持原创不容易，这篇是昨晚下班后写的，写了两个小时吧。大家有兴趣，可以支持一下我的付费内容哈，很多干货~（面试技巧专栏、踩坑专栏、知识星球辅导面试）

干货：如何准备面试

> 我的八股文面试技巧专栏已经更新38篇啦，篇篇经典，主要是教大家如何回答更全面，面试找工作的小伙伴可以购买，29.9永久买断（扫描下图二维码购买，后面在新语小程序就可以看了哈）。

八股文面试技巧专栏

捡田螺的小男孩

33 订阅 | 38 内容

作者：捡田螺的小男孩，知名公众号号主.七年大厂后端程序员，掘金优秀作者,坚持写了五年多的博客。

全网有十几万粉丝，文章阅读量累计上千万。
本专栏主要写一些通用大厂八股文的回答思路
包括Java基础、mysql、redis、消息队列等方向。
每一篇都很经典，都是大厂真题.

非常值得一看,你掌握后，面试通过率大大提高

本册原价99元，现价29.9元，永久买断，目前还在持续更新。订阅量每涨100人，会涨一次价。

微信扫码查看完整内容

现价: ¥ 29.9

长按扫码购买

> 我的代码**踩坑专栏挺不错的**。平时我晚上下班和周末都在总结代码踩坑点，花了很多心血，都是我工作遇到或者从一些前辈那里请教来的踩坑点。很实用的。目前已经更新到**92篇**啦，今天刚更新一篇添加数据库表字段可能踩的坑，很多伙伴可能都踩过的坑~~

Java程序员实战踩坑专栏

捡田螺的小男孩

5 订阅 | 92 内容

作者：捡田螺的小男孩，知名公众号号主.七年大厂后端程序员，掘金优秀作者,坚持写了五年多的博客。

全网有十几万粉丝，文章阅读量累计上千万。
本专栏主要记录Java后端程序员容易踩的代码坑。
包括Java代码基础、mysql、redis等方向。每一篇都很经典，都是日常工作经常遇到的.

非常值得一看,看完后,你的日常开发工作,很可能会少几个bug.

本册原价99元，现价49.9元，永久买断，目前还在持续更新。

微信扫码查看完整内容

现价: ￥49.9

长按扫码购买