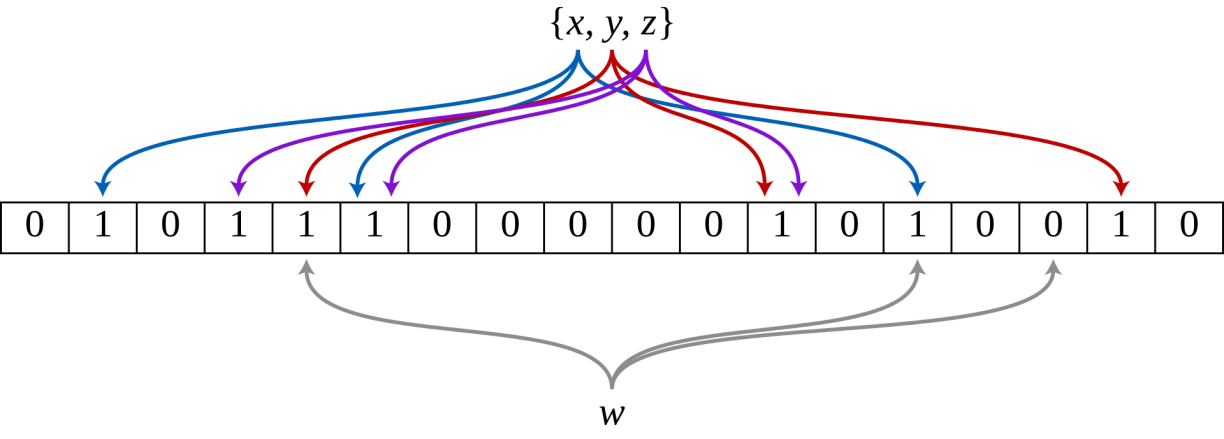
**Bloom Filtering**

Bloom Filter یک نوع ساختار داده احتمالاتی است که توسط Burton Howard Bloom در سال 1970 تصور شده و برای آزمایش اینکه آیا یک عنصر عضو یک مجموعه است ، استفاده می شود. براساس این ساختار، داده ها در لحظه ورود به پایگاه داده با استفاده از ایندکس(index) کد می شوند اما در زمان خروج یا حذف از پایگاه داده جایی ثبت نمی شوند به این دلیل با دو مفهوم "احتمالا در مجموعه وجود دارد" و "قطعا وجود ندارد" روبرو هستیم. وقتی جستجویی در این داده انجام می شود اگر گزینه "قطعا وجود ندارد" صادر شود با احتمال صد درصد می توانیم بگوییم داده در مجموعه نیست که احتمال "منفی کاذب"(False Negative) را نشان می دهد اما وقتی گزینه "احتمالا در مجموعه وجود دارد" صادر شود دو حالت وجود دارد یا داده وارد مجموعه شده و هنوز هست و یا داده وارد مجوعه شده اما سپس خارج شده است که احتمال "مثبت کاذب"(False Positive) ایجاد می شود.

بلوم فیلتر اساسا شامل یک بردار بیتی یا لیست بیتی(یک لیست شامل مقدار بیت فقط 0 یا 1) به طول m، با مقدار اولیه 0 مانند شکل زیر است:



### اندازه Bloom filter و تعداد توابع Hash

شاید به این فکر کرده باشید که اگر اندازه بلوم فیلتر کوچک در نظر گرفته شود ممکن است سریع همه خانه های آن برابر 1 شود و بعد از این هر کلمه‌ای چه در لیست باشد چه نباشد را بررسی کنیم جواب مثبت برگردانده بشود. بنابراین اندازه بلوم فیلتر خیلی مهم است. در اینجا اگر اندازه بلوم فیلتر بزرگ انتخاب شود ما جواب های مثبت کاذب کمتری داریم و اگر اندازه کوچک باشد ما جواب های مثبت کاذب بیشتری خواهیم داشت.

یک بلوم فیلتر ساده از دو عملیات test (یا بررسی) و add (یا اضافه کردن) پشتیبانی می‌کند. Test برای چک کردن اینکه آیا یک عنصر در مجموعه وجود دارد یا نه و عملیات add هم برای اضافه کردن یک عنصر به مجموعه. اما به دلیل عملیات بیتی موجود در این نوع هش (Hash) امکان حذف داده از لیست ایندکس موجود وجود ندارد. برای حل این مشکل می توانید از یک نسخه تغییر یافته بلوم فیلتر به نام Counting bloom filter استفاده کنید. به جای مرتب سازی یک بیت تکی از مقدارها، ما یک مقدار integer را ذخیره می کنیم و بردار بیتی ما بردار integer خواهد بود. قابلیت حذف کردن باعث افزایش اندازه و هزینه ما خواهد شد. در این حالت به جای علامت گذاری یک مقدار بیت با 1 هنگام ورود داده، ما مقدار integer را به مقدار 1 افزایش می دهیم و اگر داده جدیدی وارد بلوم فیلتر کردیم یک واحد دیگه افزایش می‌دهیم مثلا مقدار خانه اگر 1 بود با افزایش مجدد به 2 تغییر داده می‌شود و در هنگام حذف کلمه یک واحد از خانه‌هایی خروجی توابع هش کم می‌کنیم. برای بررسی اینکه یک عنصر موجود هست یا نه کافی هست که عنصر را هش کنیم و خانه هایی را که بررسی میکنیم مقدارشون بیشتر از 0 باشه.

## مثال های استفاده از بلوم فیلتر:

* برای هشدار به کاربران در مورد ضعیف بودن پسورد که در بالا توضیح داده شد.
* برای جلوگیری از دسترسی کاربران به سایت های مخرب
* برای بررسی اینکه چه کاربرانی قبلا از سایت شما بازدید کردند بر اساس IP هاشون.
* برای بررسی اینکه چه تعداد از کاربران محتوای سایت شما را خوانده اند

**استفاده از Bloom Filter در پایگاه داده Cassandra :**

با توجه به مدل ساختاری کاساندرا، این پایگاه داده در پروسه خواندن خود، داده های روی دیسک (در SSTables) را با داده های موجود در RAM (در MemTables) ادغام می کند. برای جلوگیری از بررسی هر پرونده داده SSTable برای پارتیشن مورد درخواست ، کاساندرا از ساختار داده ای Bloom Filter استفاده می کند. همانطور که در بالا گفته شد Bloom Filters نمی توانند تضمین کنند که داده ها در SSTable معین وجود دارد ، اما اگر به آنها اجازه دهیم رم بیشتری مصرف کنند کاربران می توانند فیلترهای Bloom دقیق تری بسازند. اپراتورها این امکان را دارند که با تنظیم bloom\_filter\_fp\_chance در یک شناور بین 0 و 1 ، این رفتار را در هر جدول تنظیم کنند. مقدار پیش فرض مربوط به bloom\_filter\_fp\_chance برای جداول با استفاده از LeveledCompactionStrategy 0.1 و برای سایر موارد 0.01 است. مقادیر معمولی برای bloom\_filter\_fp\_chance معمولاً بین 0.01 (1٪) تا 0.1 (10٪) احتمال مثبت کاذب است ، جایی که Cassandra ممکن است یک SSTable را برای یک ردیف اسکن کند ، فقط برای پیدا کردن اینکه در دیسک وجود ندارد. پارامتر باید توسط کاربر تنظیم شود، کاربرانی که رم بیشتری دارند و دیسک های کندتر ممکن است از تنظیم bloom\_filter\_fp\_chance به عددی کمتری (مانند 0.01) بهره ببرند تا از انجام بیش از حد IO جلوگیری کنند. کاربرانی که رم کمتری دارند ، گره های متراکم تری دارند ، یا دیسک های خیلی سریع ممکن است یک bloom\_filter\_fp\_chance بالاتر را تحمل کنند تا RAM را صرفه جویی در عملیات بیش از حد IO ذخیره کنند. در زمانهای کاری که به ندرت خوانده می شوند ، یا فقط با اسکن کل مجموعه داده ها (مانند بارهای کاری تحلیلی) خواندن را انجام می دهند ، تنظیم bloom\_filter\_fp\_chance روی تعداد بسیار بالاتر قابل قبول است.

<https://virgool.io/@hossein52hz/%D9%85%D9%88%D8%AC%D9%88%D8%AF%DB%8C-%D8%B9%D8%AC%DB%8C%D8%A8-%D8%A8%D9%87-%D9%86%D8%A7%D9%85-bloom-filter-%D8%AF%D8%B1-%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87-%D9%86%D9%88%DB%8C%D8%B3%DB%8C-iajb0fn0ata1>

<http://cassandra.apache.org/doc/4.0/operating/bloom_filters.html>