Oracle Index types

در زیر به اختصارانواع ،جایگاه و ویژگی هر یک از index های موجود در oracle را بررسی میکنیم :

Oracle indexes	
B-tree	پیش فرض اوراکل است،درخت ایندکس متوازنی دارد؛ برای ستون هایی که درجه ی کاردینالیتی(دارای مقادیر داده ای متمایز) بالایی دارند مناسب است. معمولا در بیشتر موارد از ایندکس B-tree استفاده میشود مگر اینکه دلیل قانع کننده ای در استفاده از ایندکس های دیگر داشته باشید: **create index cust_idx2** **on cust(first_name);**
Index Organized Table(IOT)	زمانی کاراست که اغلب مقادیر ستون های جدول عضو کلید اصلی باشند،شما می توانید به این ایندکس همانند یک جدول دسترسی داشته باشید.داده ها همانند ساختار B-tree ذخیره میشوند.IOT تمامی محتوای سطر های جدول را در یک ساختار ایندکس B-tree ذخیره میکند.همچنین در مواردی که دقیقا خود کلید اصلی یا مقادیری از آن مورد جستجو قرار میگیرد بسیار مناسب است: **create table prod_sku** **(prod_sku_id number** **,sku varchar2(256),** **constraint prod_sku_pk primary key(prod_sku_id, sku)** **) organization index;**
unique	یک فرم از B-tree است،زمانی استفاده میشود که بخواهیم یک ستون را مجبور به یکتایی در مقادیر کنیم. اغلب با محدودیت[1] های کلید اصلی و کلید یونیک استفاده میشود، اما به طور مستقل نیز میتواند در باقی محدودیت ها استفاده شود.زمانی که شما یک ایندکس B-tree ایجاد می کنید میتوانید آن را به عنوان unique تعریف کنید.در این حالت همانند یک محدودیت کلید یونیک عمل میکند.زمانی که شما در چنین جدولی درج می کنید این نوع ایندکس ضمانت میکند که تمامی مقادیر غیر null به صورت مجزا و متفاوت درج شوند: **create unique index cust_uidx1** **on cust(last_name, first_name);**
Reverse-key	یک فرم از B-tree ایندکس است؛ برای متوازن کردن I/O در ستون هایی که بیشتر مقادیر آن ها به صورت متوالی و پی در پی درج می شوند،استفاده می شود. این نوع ایندکس باعث می شود تا از تمرکز I/O بر روی یک دیسک خاص جلوگیری شود : **create index cust_ridx1** **on cust(cust_id) reverse;**
Key-compressed	این نوع ایندکس باعث کاهش فضای ذخیره سازی و منابع درخواستی I/O در ایندکس های الحاقی زمانی که ستون مقدم در اغلب موارد تکرار شود، میشود: **create index cust_cidx_1** **on cust(last_name, first_name) compress 2;**
Descending	زمانی که ترتیب مقادیر ستون جدول به صورت نزولی ذخیره شده باشد(پیش فرض سعودی است) مناسب است. این نوع را نمی توان برای ستون هایی که ایندکس معکوس شده اند یا bitmap ایندکس باشند به کاربرد.به صورت پیش فرض اوراکل B-tree ایندکس را به صورت ساختار سعودی ذخیره سازی میکند، ولی شما با بکار بردن این نوع ایندکس میتوانید این ساختار ذخیره سازی را بر عکس کنید، این نوع ایندکس برای پرس و جو[2] هایی مناسب است که برخی ستون ها را بر اساس ترتیب سعودی و برخی دیگر را بر اساس ترتیب نزولی سورت میکنند : **create index cust_didx1** **on cust(cust_id desc);**

[1] constraint
[2] query

Bitmap	در محیط های data warehouse با ستون هایی که از لحاظ کاردینالیتی پایین هستند(تعداد مقادیر متمایز کمتر) و همچنین در دستورات sql از عملگر های and و or در شرط where زیاد استفاده میکنند،مناسب است. در محیط های OLTP(online transaction processing) که دائما در حال بروزرسانی رکورد ها هستند،مناسب نیستند.همچنین شما نمیتوانید ایندکسی از نوع unique bitmap بسازید. **create table f_sales(** **sales_amt number** **,d_date_id number** **,d_product_id number** **,d_customer_id number);** **create bitmap index f_sales_fk1** **on f_sales(d_date_id);**
Bitmap Join	در محیط های data warehouse زمانی که پرس و جویی از الحاق[3] جداول fact و dimension در ساختار شمای ستاره ای[4] میگیرید مناسب است .این نوع ایندکس نتیجه ی الحاق بین دو جدول را در خود ذخیره میکند. این نوع ایندکس ها سودمند هستند چرا که از الحاق دوباره ی جداول به منظور دستیابی به نتایج جلوگیری میکند.این نوع ایندکس زمانی مناسب است که الحاق بین دو جدول بر اساس کلید خارجی و کلید اصلی باشد.در محیط های OLTP مناسب نیست. **create table d_customers** **(d_customer_id number primary key** **,cust_name varchar2(30));** **create bitmap index f_sales_bmj_idx1** **on f_sales(d_customers.cust_name)** **from f_sales, d_customers** **where f_sales.d_customer_id = d_customers.d_customer_id;**
Function-based	برای ستون هایی که بر روی آنها در پرس و جو های sql ، function اعمال شده باشد مناسب است.میتواند با bitmap یا B-tree استفاده شود. وجود این نوع ایندکس ضروری است چرا که به کار بردنfunction بر روی ستون ها در شرط where باعث میشود که از ایندکس صرفه نظر شود، ولی وقتی که ما ایندکسی از نوع همان function میسازیم باعث میشود که از ایندکس استفاده شود : **create index cust_fidx1** **on cust(upper(last_name));**
Indexed virtual column	ایندکسی است که بر روی ستون های مجازی جداول تعریف میشود.برای ستون هایی که بر روی آن ها function زده شده مناسب است و جایگزینی برای ایندکس function-based است. **create table inv(** **inv_id number** **,inv_count number** **,inv_status generated always as (** **case when inv_count <= 100 then 'GETTING LOW'** **when inv_count > 100 then 'OKAY'** **end)** **);** **create index inv_idx1** **on inv(inv_status);**

[3] join
[4] Star schema

virtual	به شما اجازه میدهد که ایندکسی بدون درنظر گرفتن منابع فیزیکی نظیر سگمنت با قرار دادن شرط NOSEGMENT، تعریف کنید.برای بهینه کردن دستورات sql بدون مصرف کردن منابع فیزیکی مورد نیاز ایندکس، مناسب است.تمامی انواع ایندکس میتوانند از نوع virtual تعریف شوند.شما میتوانید بهینه ساز[5] اوراکل را مجبور به استفاده از این نوع ایندکس ها بکنید : **SQL> alter session set "_use_nosegment_indexes"=true;** یکی از مواقع استفاده از این نوع ایندکس ها زمانی است که شما ایندکس بسیار حجیمی داشته باشید و بخواهید آن را در حالت بدون اختصاص فضا بسازید، برای حصول به این نتیجه باید ابتدا این ایندکس را در حالت nosegment بسازید و از طریق بهینه ساز بررسی کنید که آیا آین ایندکس استفاده میشود یا خیر! در صورت مثبت و سودمند بودن، آن را حذف و دوباره در حالت بدون استفاده از nosegment بسازید. **create index cust_idx1** **on cust(first_name) nosegment;**
invisible	در این شرایط ایندکس برای بهینه ساز قابل رویت نیست و همچنین از آن ایندکس زمانی که از طریق پرس و جو ها داده ها را بازیابی میکند استفاده نمیکند.ولی ساختار ایندکس با تغییرات بر روی جدول مربوطه بروز رسانی میشود. برای زمانی که بخواهیم ایندکس را قبل از قابل رویت کردن برای اپلیکیشن، تست کنیم مناسب است.تمام انواع ایندکس ها میتوانند بصورت غیر قابل رویت ساخته شوند (این خصیصه در اوراکل 11 و نسخه های بالا تر قابل اجراست). **create index cust_iidx1** **on cust(last_name) invisible;**
Globally and Locally Partitioned Indexes[6]	پارتیشن ایندکس یک مفهوم فیزیکی است ولی به لحاظ فیزیکی در سگمنت های متفاوت تقسیم بندی شده است. به کارگیری از آن در مواقعی که دیتابیس های بزرگی داریم مناسب است و باعث افزایش کارایی میشود.پارتیشن ایندکس میتواند بصورت سراسری[7] و یا محلی[8] باشد.پارتیشن ایندکس سراسری از استراتژی پارتیشن بندی که به سگمنت های جدول مربوطه نگاشته نشده باشد،استفاده میکند.شما میتوانید پارتیشن ایندکس سراسری را بر روی جداول منظم[9] یا پارتیشن بندی شده بسازید. **create index f_sales_gidx1 on f_sales(sales_amt)** **global partition by range(sales_amt)** **(partition pg1 values less than (25)** **,partition pg2 values less than (50)** **,partition pg3 values less than (maxvalue));** پارتیشن ایندکس محلی، باید بر روی جداول پارتیشن بندی شده ساخته شود.این نوع ایندکس از همان استراتژی پارتیشن بندی جدول مربوطه تبعییت میکند؛ فقط شامل مقادیر پارتیشن جدول مربوطه است.بر خلاف حالت سراسری که فقط میتواند در حالت B-tree ساخته شود در این نوع در حالت bitmap نیز ساخته میشود : **create table f_sales(** **sales_amt number** **,d_date_id number** **,d_product_id number** **,d_customer_id number)** **partition by range(sales_amt)(** **partition p1 values less than (100)**

[5] optimizer

[6] قابل پیاده سازی در نسخه ی Enterprise oracle است.

[7] global

[8] local

[9] regular

,partition p2 values less than (1000) ,partition p3 values less than (maxvalue)); create index f_sales_idx2 on f_sales(d_date_id, sales_amt) local;	
Domain ایندکس بر روی اپلیکیشن های خاص پیاده سازی میشود. این نوع ، ایندکس ها را با انواع داده ها،داکیومنت ها،عکس ها، ویدئو و داده های سه بعدی تطبیق میدهند. B-tree cluster ایندکسی است که بر روی کلید جدول کلاستر شده تعریف میشود.این نوع ایندکس، کلید کلاستر را با آدرس بلاک های دیتابیس مربوط میسازد.hash cluster ایندکس نیز بر روی جداول کلاستر تعریف میشود با این تفاوت که از توابع hash به جای کلید ایندکس استفاده میکند.	**Domain , B-tree Cluster and Hash Cluster Indexes**

تهیه و تنظیم :علی امجدی محب