اشیاء شِما در اوراکل

(Oracle Schema Objects)

نویسندگان:

زهرا فرجام فرد

ایمان لؤلؤیی

هادی جعفری منفرد

سید محمد مهدی رشتی

معرفی اشیاء شما: (Schema Object)

شما مجموعه ای از ساختار داده ها، یا اشیاء می باشد. کاربر، می تواند یک شما را شخصی کند، که در این صورت نامی مشابه کاربر دارد. هر کاربر صاحب یک شمای منحصر به فرد است. اشیاء شما می توانند بوسیله SQL ایجاد و دستکاری شوند که دارای انواع زیر می باشند:

- کلاسترها (Clusters)
- لینک های پایگاه داده (Database links)
- محرک های پایگاه داده (Database triggers)
- بعد (Dimensions)
- کتابخانه های پروسیجر خارجی (External procedure libraries)
- کلاس ها و منابع جاوا (Java classes, Java resources, and Java sources)
- دید لحظه ای(عکسبرداری) (Materialize view and materialized view logs)
- جدول، نوع و دیدگاه اشیاء (Object tables, object types, object views)
- عملگرها (Operators)
- رشته ها (Sequences)
- توابع، پروسیجرها، و پکیج های ذخیره شده (Stored function, procedures, packages)
- مترادف ها (Synonyms)
- جدول ها وجدول های با ساختار ایندکس (Tables and index-organized table)
- دیدگاه ها (Views)

انواع دیگر اشیا که در پایگاه داده ذخیره شده و می توانند بوسیله SQL ایجاد و دستکاری شوند، عبارتند از:

- متن ها (Contexts)

- دایرکتوریها (Directories)
- پروفایلها (Profiles)
- نقش ها (Roles)
- فضای جدول (Tablespaces)
- کاربران (Users)
- قسمتهای (Rollback segments)

اشیاء شما، ساختار نگهداری داده های منطقی هستند. این اشیاء با فایل های فیزیکی روی دیسک که اطلاعاتشان را ذخیره می کنند، تناظر یک به یک ندارند. هر چند، اوراکل، یک شی از شما را به طور منطقی در یک tablespace از پایگاه داده ذخیره می کند. داده هر شی به طورمنطقی ، در یک یا بیشتر از یک فایل های داده tablespace، در بر گرفته شده اند.

هیچ رابطه ای بین شما و tablespace ها وجود ندارد: یک tablespace می تواند شامل اشیایی از شماهای مختلف باشد، و اشیاء یک شما می توانند در tablespace های مختلف قرار داده شوند.

شکل زیرارتباط بین اشیاء، tablesapace ها، و فایل های داده (datafiles) را نشان می دهد.

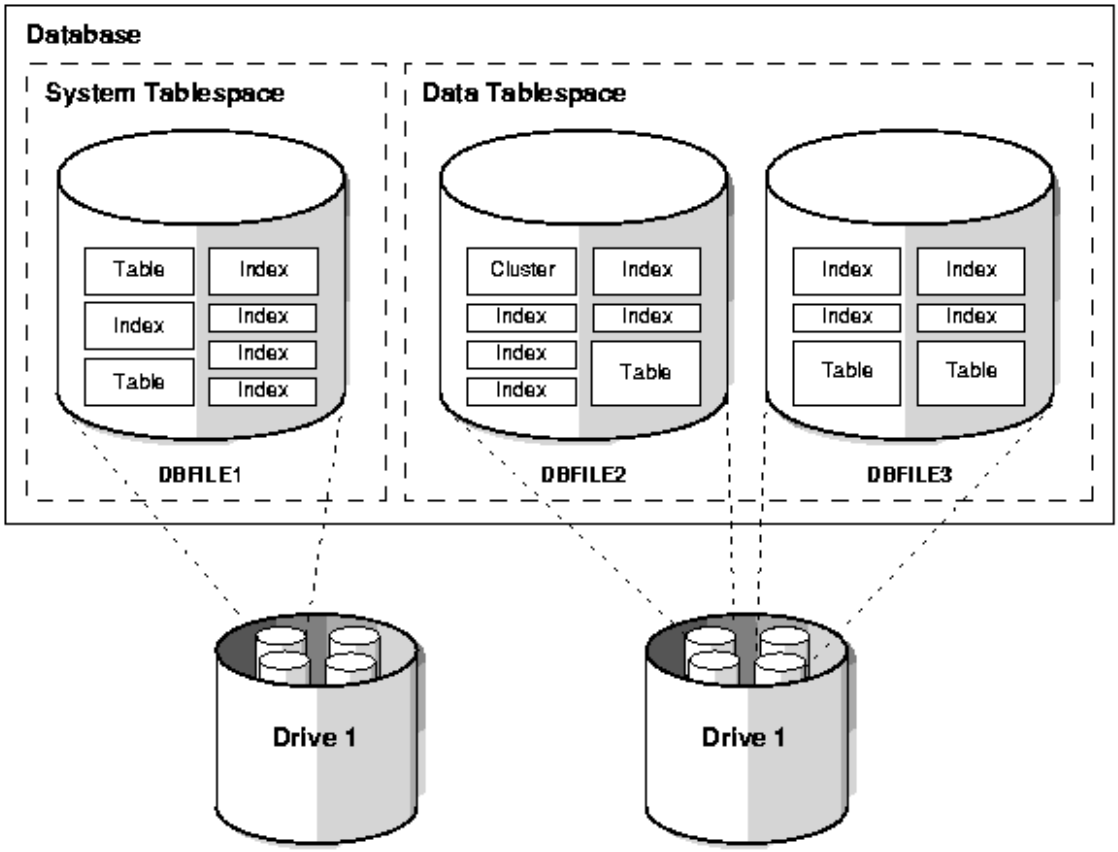

Figure ۱۰-۱ Schema Objects, Tablespaces, and Datafiles

جدول ها (Tables) :

جدول ها، واحدی اساسی از نگه داری داده در یک پایگاه داده اوراکل هستند. داده در ردیف ها و ستون ها ذخیره می شوند. جدول با یک نام (مانند employees) و مجموعه ای از ستون ها تعریف می شود. به هر ستون یک نام (مثل: employee_id, last_name, job_id)، یک نوع داده (مثل: VARCHAR۲,DATE, NUMBER)، و یک عرض (width) می دهید. عرض می تواند بوسیله یک نوع داده مثل DATE ، از قبل تعیین گردد. چنانچه ستون ها از نوع NUMBER باشند به جای عرض، دقت (precision) ومقیاس (scale) ، تعریف می شوند. ردیف مجموعه ای از اطلاعات یک ستون مطابق با یک رکورد منفرد می باشد.

شما می توانید برای ستون های جدول، قوانینی مشخص نمایید. این قوانین، محدودیت های یکپارچگی نامیده می شوند. یک مثال از این محدودیت ها را می توان NULL نبودن بر شمرد. این محدودیت، ستون را مجبور می سازد که برای هر ردیف حتما مقداری را داشته باشد.

پس از اینکه جدولی را ایجاد نمودید، ردیف هایی از داده ها را با استفاده از جملات SQL وارد می کنید. جدول داده ها می توانند بوسیله SQL مورد درخواست قرار گیرند و یا پاک و یا به روز رسانی شوند.

شکل زیر مثالی از یک جدول با نام emp را نمایش می دهد.

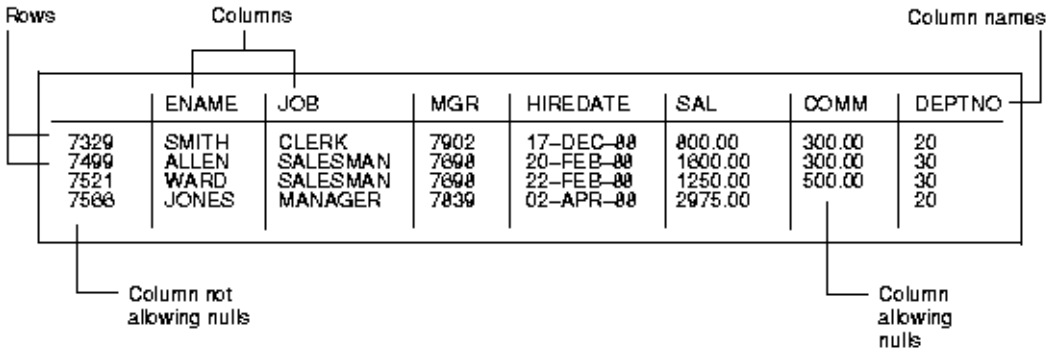

Figure ۱۰-۲ The EMP Table

چگونه جدول داده ها ذخیره می شود؟

زمانی که یک جدول ایجاد می کنید، اوراکل به طور اتوماتیک بخشی از داده را به یک tablesapace ، به منظور نگه داری داده بعدی جدول، تخصیص می دهد. شما می توانید تخصیص و استفاده از فضا، برای بخشی از داده جدول را به طروق زیر کنترل نمایید:

- می توانید قسمتی از فضای تخصیص داده شده به سگمنت داده را بوسیله ست کردن پارامتر ذخیره سازی برای سگمنت داده، کنترل نمایید.

- شما می توانید استفاده از حافظه آزاد در بلوکهای داده، که سگمنتهای داده را تشکیل می دهند را با استفاده از پارامترهای PCTFREE و PCTUSED کنترل کنید.

اوراکل به جای اینکه داده را در بخش داده در یک tablespace ذخیره نماید، در یک جدول کلاستر شده در بخش داده ایجاد شده برای کلاستر، ذخیره می کند. نگهداری (storage) پارامترها، نمی تواند زمان ایجاد کردن یا تغییر دادن یک جدول کلاستر شده را، مشخص نماید. نگه داری پارامترها، کنترل کردن، ذخیره سازی همه جدول ها را در یک کلاستر، ایجاد می کند.
Tablespace ی که شامل یک سگمنت داده جدول کلاستر نشده باشد، یک tablespace پیش فرض و یا یک tablespace ی که صریحا در جمله CREATE TABLE ، نام گذاری شده، می باشد.

فرمت و سایز ردیف:

اوراکل هر ردیف از یک جدول پایگاه داده، که شامل داده با کمتر از ۲۵۶ ستون باشد، به عنوان یک قطعه یا بیشتر از یک قطعه، ذخیره می کند. اگر کل ردیف بتواند در یک بلوک داده منفرد، قرار داده شود، آنگاه اوراکل ردیف را به عنوان یک قطعه ذخیره می کند. اما، اگر همه یک ردیف نتواند در یک بلوک داده جای گیرد و یا اینکه یک به روز رسانی بر روی ردیف باعث شود، که ردیف بیشتر از بلوک رشد نماید، آنگاه اوراکل ردیف را با استفاده از چندین قطعه ذخیره می نماید. هر بلوک معمولا، فقط شامل یک قطعه برای یک ردیف است. زمانی که اوراکل مجبور به ذخیره یک ردیف در بیش از یک قطعه می شود، بلوک ها به همدیگر به نوعی زنجیر می شوند.
هر قطعه ردیف، زنجیر شده یا زنجیر نشده، برای همه یا بعضی از ستون های ردیف، داده و سرآیند ردیف (row header) ، را شامل می شود. همچنین ستون های اختصاصی می توانند قطعات ردیف و به دنبال آن بلوک های داده، را اندازه گیری کنند. در شکل زیر فرمت یک قطعه ردیف را مشاهده می کنید.

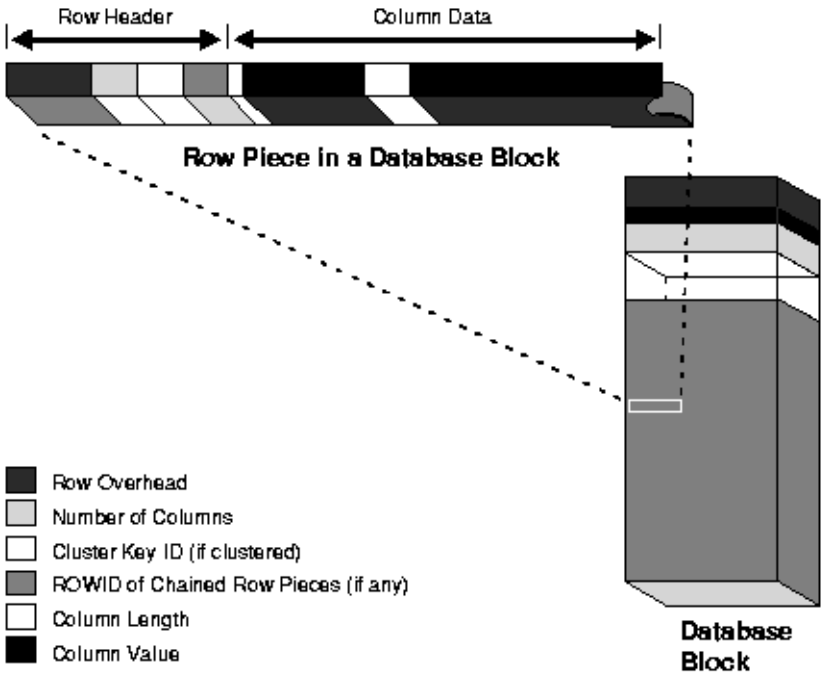

Figure 10-3 The Format of a Row Piece

سرآیند ردیف (row header) بر داده، مقدم است و شامل اطلاعات زیر می باشد:

- قطعه های ردیف (Row pieces)
- زنجیره ای (Chaining)
- ستون های یک قطعه ردیف (Columns in the row piece)
- کلیدهای کلاستر (Cluster keys)

پس از این اطلاعات، هر ردیف، طول (length) ستون و داده را شامل می شود. که طول ستون نیازمند ۱ بایت برای ستون هایی است، که ۲۵۰ بایت یا کمتر را ذخیره می کنند، یا نیازمند ۳ بایت برای ستون هایی است که بیشتر از ۲۵۰ بایت را ذخیره می کنند، و توجه داشته باشید که طول ستون به داده مقدم است. فضای مورد نیاز برای داده ستون ها، هم بستگی به نوع داده دارد.

ترتیب ستون ها:

ترتیب ستون ها در همه ردیف های جدول داده شده، یکسان است. ستون ها معمولا به همان ترتیبی که در عبارت CREATE TABLE ، لیست شده اند، ذخیره می شوند، البته این یک موضوع تضمین شده نمی باشد. به عنوان مثال، اگرجدولی با ستونی از نوع LONG ایجاد کنید، اوراکل این ستون را همیشه در آخر، ذخیره می کند. همچنین، اگر جدولی تغییر داده شود، ستون جدید اضافه شده به عنوان آخرین ستون ذخیره می شود.

Null ها فقدان مقدار را نشان می دهند:

یک null فقدان یک مقدار، در ستونی از یک ریف، می باشد. در واقع Null ها، داده های ناپیدا (missing) ، ناشناخته (unknown) ، یا ناجور و تطبیق نشدنی (inapplicable) را نشان می دهند. توجه داشته باشید که Null مفهوم مقادیری مانند صفر، را نمی رساند. به یک ستون اجازه Null بودن داده می شود، مگر اینکه محدودیتی مانند: NOT NULL و یا PRIMARY KEY integrity ، بر روی آن ستون تعریف شده باشد که در این صورت این ستون در همه ردیف ها باید حتما مقداری داشته باشد.

Null های متوالی در یک ردیف نیازی به انباره ذخیره سازی ندارند، چرا که سرآیند یک ردیف جدید به ستون هایی که در ردیف قبلی null بودند، اشاره می کند. به عنوان مثال، اگر سه ستون آخر یک جدول null باشند، هیچ اطلاعاتی برای آنها ذخیره نمی شود. در جدول هایی که تعداد زیادی ستون دارند، معمولا ستون هایی را که احتمال بیشتری می رود که null باشند، جزء ستون های آخر، به منظور استفاده بهینه از فضای دیسک، قرار می دهند.

مقادیر پیش فرض برای ستون ها:

شما می توانید یک مقدار پیش فرض را به یک ستون از یک جدول، انتساب دهید، بنابراین زمانی که یک ردیف جدید اضافه می شود و مقدار ستونی از قلم می افتد و یا کلمه کلیدی DEFAULT فعال می شود، مقدار پیش فرض به طور اتوماتیک برای آن ستون عرضه می شود.

نوع داده عبارات یا لیترال های پیش فرض باید با نوع داده ستون مطابقت داشته باشد و یا قابل تبدیل به نوع داده ستون باشد.

اگر برای ستونی، صریحا مقدار پیش فرض تعریف نشده باشد، آنگاه برای آن ستون مقدار null به عنوان پیش فرض در نظر گرفته می شود.

جدول های تقسیم بندی شده (Partitioned Tables) :

جدول های تقسیم بندی شده به داده شما این اجازه را می دهند که به قطعات کوچکتر قابل مدیریتی به نام پارتیشن، شکسته شوند. ایندکس ها نیز به همید منوال می توانند تقسیم بندی شوند. هر پارتیشن می تواند به طور اختصاصی مدیریت شود، و مستقل از سایر پارتیشن ها عمل نماید، بنابراین فراهم آوردن چنین ساختاری می تواند به دستیابی و کارایی بهتر کمک کند.

جدول های تو در تو (Nested Tables) :

شما می توانید یک جدول با ستونی که نوع داده اش از نوع جدولی دیگر است، ایجاد نمایید. این بدبن معناست که، جدول ها می توانند در داخل جدول های دیگر به عنوان مقدار یک ستون، قرر بگیرند، یعنی همان جدول های تو در تو. سرور اوراکل داده های جدولهای تو در تو را خارج از سطرهایی از جدول اصلی ذخیره می کنند و از مدل ذخیره سازی که ستونهایی را برای ذخیره جدول تودرتو در نظر می گیرد استفاده می کند.

ردیف جدول والد، شامل یک مجموعه واحد از مقدار شناسه ای است، که به یک نمونه از جدول داخلی اشاره می کند.

جدول های موقت (Temporary Tables) :

به منظور جدول های پایدار و دائمی (permanent) ، اوراکل می تواند جدول های موقتی (temporary) را برای نگه داری داده هایی که فقط در طول یک تراکنش یا نشست بوجود می آیند، ایجاد نماید.

عبارت CREATE GLOBAL TEMPORARY TABLE ، یک جدول موقت، که می تواند مخصوص تراکنش و یا مخصوص نشست باشد، ایجاد می نماید. جدول های موقتی که مخصوص

تراکنش هستند، داده ای را که فقط در طول تراکنش وجود دارند، نگه داری می کنند و جدول های موقت مخصوص نشست، داده ی طول نشست، را نگه داری می کنند. در یک چنین جدولی، برای هر نشست داده ی اختصاصی وجود دارد و هر نشست فقط می تواند داده مال خودش را ببیند و دستکاری نماید. قفل های DML بر روی داده جدول های موقت، تعیین نمی شوند و عبارت LOCK ، بر روی یک جدول موقت اثری ندارد، چرا که هر نشست داده مخصوص خودش را دارد.
شما می توانید شاخص هایی را بر روی جدول های موقت با استفاده از عبارت CREATE INDEX ، ایجاد کنید. توجه داشته باشید که ایندکس های ایجاد شده بر روی چنین جول هایی، موقت می باشند، و داده ایندکس شده، همان حوزه (scope) تراکنش یا نشستی را دارد که داده در جدول موقت دارد.
شما می توانید دیدگاه هایی (view) را بر روی هر دو جدول موقت و دائمی، ایجاد نمایید. همچنین می توانید تریگر هایی را بر روی جدول موقت ایجاد نمایید.

تراکنش های فرزند و والد:

جدول های موقت مخصوص تراکنش بوسیله تراکنش های کاربر (user transaction) و تراکنش های فرزندانشان (child transaction) ، قابل دسترسی هستند. اگر چه جدول موقت مخصوص تراکنش نمی تواند بوسیله دو تراکنش موجود در یک نشست مورد استفاده قرار گیرد، اما بوسیله تراکنش های موجود در نشست های مختلف می تواند استفاده شود.

توجه داشته باشید که اگر تراکنش کاربر، یک INSERT در داخل جدول موقت داشته باشد، پس از آن، هیچ تراکنش فرزندی نمی تواند از جدول موقت استفاده کند.
اگر تراکنش فرزند، یک INSERT در جدول موقت انجام دهد، سپس در پایان تراکنش فرزند، داده انتساب داده شده به جدول موقت از بین می رود . پس از آن هر تراکنش کاربر و یا تراکنش فرزند دیگری، می تواند به جدول موقت دسترسی داشته باشد.

جدول های خارجی (External Tables) :

شما می توانید به داده ی منابع خارجی، چنانچه در یک جدول در داخل پایگاه داده بود، دسترسی داشته باشید. در واقع می توانید به پایگاه داده متصل شودید و متادیتایی برای جدول خارجی با استفاده از DDL ، ایجاد کنید. DDL برای یک جدول خارجی شامل دو بخش است: یک بخش که نوع هایی را که یک ستون می تواند در اوراکل داشته باشد، توضیح می دهد، و بخش دیگر، نگاشت داده خارجی به ستون داده اوراکل، را شرح می دهد.

جدول خارجی در باره هیچ کدام از داده های ذخیره شده در پایگاه داده، توضیحی نمی دهد. همچنین درباره اینکه چگونه داده در منابع خارجی ذخیره شده، شرحی نمی دهد. در عوض، توضیح می دهد که لایه جدول خارجی چگونه به ارائه دادن داده به سرور، نیاز دارد.

جدول های خارجی، فقط خواندنی هستند، بنابرای استفاده از هیچ کدام ازعملگرهای DML ممکن نیست، و بر روی آنها ایندکس هم نمی توان کرد.

دیدگاه ها (Views) :

دیدگاه نمایش مناسبی از داده ای است که توسط یک یا بیشتر از یک جدول و یا سایر دیدگاه ها، در بر گرفته شده است. در واقع دیدگاه خروجی یک درخواست (query) را می گیرد و با آن به عنوان یک جدول رفتار می کند. بنابراین، دیدگاه می تواند تصویری از یک درخواست ذخیره شده و یا یک جدول مجازی باشد. شما می تواند از یک دیدگاه در بیشتر مکان هایی که از جدول استفاده می کردید، استفاده کنید.

به عنوان مثال، جدول employees ، چندین ستون و بیشمار ردیف حاوی اطلاعات دارد. چنانجه شما بخواهید کاربران فقط پنج تا از این ستون ها و یا فقط تعداد مشخصی از ردیف ها را ببینند، کافیست، دیدگاهی از آن جدول برای کاربرانی که به آن دسترسی دارند، ایجاد نمایید.

شکل زیر مثالی از یک دیدگاه به نام STAFF را نشان می دهد که از جدول پایه employees ، مشتق شده است.

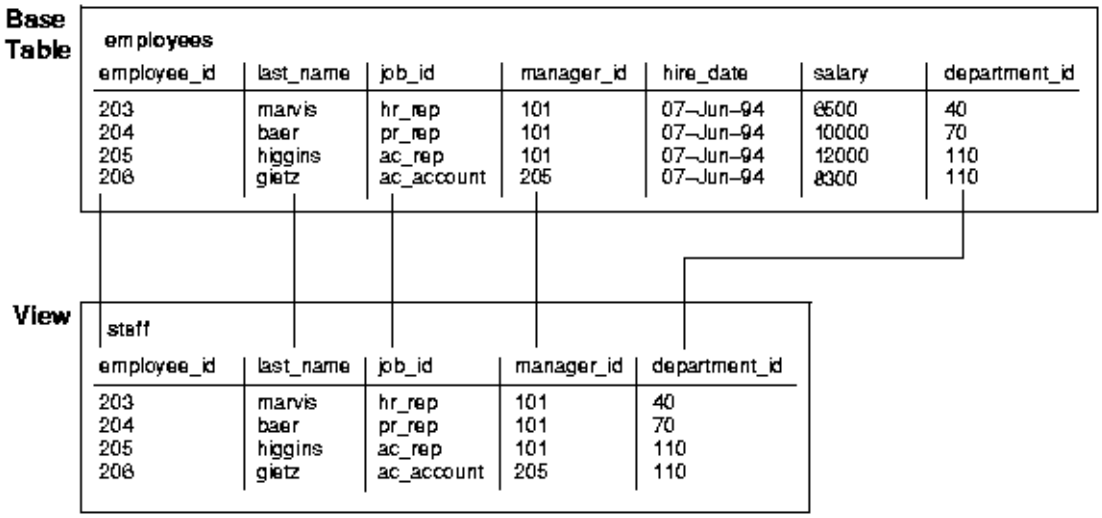

Figure ۱۰-۵ An Example of a View

از آنجایی که دیدگاه ها از جداول مشتق می شوند، شباهت های زیادی با یکدیگر دارند. به عنوان مثال، شما می توانید دیدگاهی با ۱۰۰۰ ستون داشته باشید، درست شبیه به یک جدول. همچنین می توانید به یک دیدگاه درخواست بدهید و یا با اعمال بعضی از محدودیت ها، آن را به روز رسانی، حذف و یا اضافه نمایید. تمام اعمالی که بر روی یک دیدگاه انجام می دهید بر روی جدول های پایه ای که دیدگاه از آنها مشتق شده، اثر می گذارد.

توجه:

شما نمی توانید به طور صریح تریگرهایی (triggers) را بر روی دیدگاه ها تعریف نمایید، بلکه می توانید آنها را بر روی جدول های پایه ای که توسط دیدگاه به آن اشاره می شود، تعریف کنید. اوراکل از تعریف محدودیت های منطقی بر روی دیدگاه ها را، پشتیبانی می کند.

دیدگاه ها چگونه ذخیره می شوند:

بر خلاف جداول، به دیدگاه ها هیچ فضای ذخیره ای، اختصاص داده نمی شود، چرا که واقعا شامل داده نیست. بلکه، توسط درخواستی که داده را از جدول هایی که دیدگاه به آنها اشاره می کند، استخراج و یا استنتاج می کند، تعریف می شوند. این جداول، جدول های پایه (base table) نامیده می شوند. جدول های پایه می توانند جدول های واقعی باشند و یا اینکه خودشان هم دیدگاه باشند. از

آنجا که یک دیدگاه بر اساس سایر اشیاء است، به انباری به جزء انباری برای تعریف دیدگاه در فرهنگ داده، نیاز ندارد.

دیدگاه ها چگونه مورد استفاده قرار می گیرند؟

دیدگاه ها امکاناتی را برای نمایش داده هایی که در جداول پایه مستقر هستند، فراهم می کنند. دیدگاه ها بسیار قدرتمند هستند، چرا که به شما اجازه می دهند تا ارائه مناسبی از داده ها با انواع متفاوت، برای کاربران، داشته باشید. دیدگاه ها قالبا مورد استفاده قرار می گیرند، برای:

- فراهم آوردن یک سطح اضافی از امنیت، بوسیله محدود کردن دسترسی به یک مجموعه معین شده از ریف ها و ستون های یک جدول.
- پنهان کردن پیچیدگی داده.
- ساده کرند عبارات برای کاربر. به عنوان مثال، دیدگاه ها به کاربران اجازه می دهند، اطلاعاتی را از چندین جدول بدون اینکه به چگونگی کار آگاهی داشته باشند، انتخاب کنند.
- نمایش داده از جنبه ای متفاوت از آنچه در جدول پایه است. به عنوان مثال، یک ستون از دیدگاه می تواند تغییر نام پیدا کند، بدون اینکه اثری بر روی جدول های پایه ای که دیدگاه از آنها مشتق شده، بگذارد.
- در امان نگه داشتن برنامه کاربردی از تغییراتی که در جدول پایه رخ می دهد. به عنوان مثال، اگر درخواستی (query) که یک دیدگاه را تعریف می کند، به سه ستون از چهار ستون یک جدول پایه اشاره کند، در این صورت چنانچه به تعداد ستون های جدول پایه اضافه هم شود، باز تاثیری در تعریف دیدگاه ندارد، و برنامه هایی که از دیدگاه استفاده می کنند، بدون تغییر باقی می مانند.
- ذخیره کردن درخواست های پیچیده.

طرز کار دیدگاه ها:

اوراکل یک دیدگاه را به عنوان متن درخواستی که دیدگاه را تعریف کرده، در فرهنگ داده، ذخیره می کند. زمانیکه شما به یک دیدگاه با استفاده از عبارت SQL اشاره می کنید، اوراکل:

۱. عبارتی را که به دیدگاه اشاره می کند با درخواستی که دیدگاه را تعریف می کند، ادغام می نماید.
۲. عبارت ادغام شده را در یک ناحیه SQL مشترک (shared)، پارس می کند.
۳. عبارت را اجرا می نماید.

اوراکل عبارتی را که به یک دیدگاه در یک ناحیه SQL مشترک اشاره می کند، تنها به شرطی پارس می کند که، آن ناحیه شامل عبارت مشابه دیگری نباشد. بنابراین، شما با استفاده کردن از دیدگاه ها، بهترین استفاده را از حافظه می کنید.

استفاده از ایندکس ها در برابر دیدگاه ها:

اوراکل تعیین می کند که آیا از ایندکس ها برای یک درخواست در برابر دیدگاه، بوسیله تبدیل کردن درخواست اصلی زمانی که با درخواست تعریف دیدگاه، ادغام می شود، استفاده شود یا نه؟
دیدگاه زیر را بررسی کنید:

```
CREATE VIEW employees_view AS
  SELECT employee_id, last_name, salary, location_id
    FROM employees, departments
    WHERE employees.department_id = departments.department_id AND
                        departments.department_id = 10;
```

حالا درخواستی که کاربر داده است، بررسی کنید:

```
SELECT last_name
  FROM employees_view
```

```
  WHERE employee_id = 9876;
```

و اما درخواست پایانی که توسط اوراکل ساخته می شود:

```
SELECT last_name
  FROM employees, departments
  WHERE employees.department_id = departments.department_id AND
        departments.department_id = 10 AND
        employees.employee_id = 9876;
```

در هر صورت اوراکل اوراکل query کاربر را با query مربوط به تعریف view که در query به کار رفته است ادغام می کند. در واقع اوراکل درخواست ادغام شده را بهینه می کند، به این صورت که به درخواستی که کاربر داده به عنوان درخواستی نگاه می کند که به هیچ دیدگاهی اشاره نمی کند. بنابراین، اوراکل می تواند از ایندکس ها بر روی ستون های هر جدول پایه اشاره شده، استفاده کند، خواه ستون ها در تعریف دیدگاه اشاره شده باشند و یا در درخواست کاربر.

وابستگی ها و دیدگاه ها:

از آنجا که یک دیدگاه بوسیله درخواستی که به سایر اشیاء (جدول ها و یا سایر دیدگاه ها) اشاره می کند، تعریف می شود، بنابراین به اشیاء ارجاع داده شده وابسته می باشد. اوراکل به طور اتوماتیک این وابستگی ها را برای دیدگاه ها، هندل می کند. به عنوان مثال، اگر شما جدول پایه ای را حذف و سپس ایجاد کنید، اوراکل تعیین می کند که آیا جدل پایه جدید برای دیدگاه موجود قابل قبول و پذیرفتنی هست یا نه؟

دیدگاه های الحاقی قابل به روز رسانی (Updatable join views) :

یک دیدگاه الحاق شده (join view) به عنوان دیدگاهی که بیشتر از یک جدول دارد، تعریف می شود، و یک چنین دیدگاهی از هیچ کدام از این شرایط : DISTINC, AGGREGATION, GROUP BY, START WITH, CONNECT BY, ROWNUM و نیز مجموعه عملیات (UNION ALL, INTERSECT, ..) ، استفاده نمی کند.

دیدگاه الحاقی قابل به روز رسانی (updatable join views) ، یک دیدگاه الحاقی است که شامل دو و یا بیشتر از دو جدول پایه و یا دیدگاه می باشد، با این تفاوت که عملیات حذف، اضافه و به روز رسانی بر روی آن، اجازه داده شده است. فرهنگ داده دیدگاه ها :

ALL_UPDATABLE_COLUMNS
DBA_UPDATABLE_COLUMNS
USER_UPDATABLE_COLUMNS

شامل اطلاعاتی است که نشان می دهد ستون های دیدگاه قابل به روز رسانی هستند. به منظور ماندگاری یک به روز رسانی، دیدگاه نباید هیچ کدام از محدودیت های زیر را شامل شود:

- عملگر مجموعه ای (set operator)
- عملگر محدودیت (DISTINC operator)
- توابع اجتماع یا تجزیه (aggregate or analytic function)
- شرایط GROUP BY,ORDER BY, CONNECT BY, START WITH
- مجموعه عبارات در یک لیست انتخابی
- زیر درخواست (sub query) در یک لیست انتخابی
- الحاق کردن با بعضی از عبارات.

دیدگاه های شیئی (Object Views) :

در پایگاه داده رابطه ای شئ گرای اوراکل (Oracle object-relational database) ، یک دیدگاه شیئی (object view) به شما اجازه بازیابی، به روز رسانی، حذف و اضافه داده ای را می دهد که به عنوان یک نوع شئ (object type) ذخیره شده باشد. همچنین شما می توانید دیدگاه هایی با ستون هایی از نوع شئ، مانند REFs و مجموعه ها (جدول های تو در تو)، تعریف کنید.

دیدگاه های مادی (Materialized Views) :

این نوع دیدگاه ها، اشیایی از شما هستند که می توانند برای خلاصه سازی (summarize) ، محاسبه کردن (compute) ، تکرار کردن (replicate) ، و توزیع کردن (distribute) داده مورد استفاده

قرار گیرند. آنها برای محیط های محاسبه ای گوناگون مثل، پشتیبانی کردن از تصمیم گیری ها (decision support) ، انبار کردن داده (data warehousing) و محاسبات توزیع شده و یا متحرک (distributed or mobile computing) ، مناسب می باشند.

- در data warehouses ، این نوع دیدگاه ها، برای محاسبه کردن و ذخیره نمودن داده اجتماع شده، مانند جمع ها و میانگین ها، مورد استفاده قرار می گیرند. در این گونه محیط ها معمولا این دیدگاه ها به عنوان summaries ، ارجاع داده می شوند، چرا که آنها داده خلاصه شده را ذخیره می کنند. همچنین از این دیدگاه ها برای محاسبه کردن join ها با یا بدون تجمع، استفاده می شود.
- در محیط های توزیع شده، از این دیدگاه ها برای تکرارداده در سایت های توزیع شده و انجام به روز رسانی های همزمان در چندین سایت، با متدهای حل تضادها، استفاده می شود.
- در محیط های محاسبه متحرک، از این دیدگاه ها برای بارگذاری زیر مجموعه ای از داده ی سرور مرکزی به مشتری متحرک استفاده می شود، البته بوسیله تازه سازیهای (refreshes) دوره ای از طرف سرور مرکزی به مشتری و نیز انتشار به روز رسانی های انجام شده توسط مشتری، به سمت سرور مرکزی.

این دیدگاه ها شبیه به ایندکس ها در چندین مورد می باشند، از جمله:

- برای انبار کردن، فضایی را مصرف می کنند.
- چنانچه داده در جدول اصلی تغییر پیدا کند، آنها هم باید تجدید شوند (refresh).
- آنها کارایی اجرایی SQL را، زمانیکه برای باز نویسی بکار می روند، بهتر می کنند.
- موجودیت آنها برای برنامه های کاربردی و کاربران، ناپیدا (transparent) است.

بر خلاف ایندکس ها، این نوع دیدگاه ها می توانند به طور مستقیم با استفاده از عبارت SELECT، مورد دسترسی قرار بگیرند. همچنین این دیدگاه ها، وابسته به نوع تازه سازی که مورد نیاز است، می توانند مستقیما در عبارات INSERT, UPDATE, DELETE مورد دسترسی قرار گیرند.

ابعاد (Dimensions) :

یک بعد، ارتباط های مرتبه ای (فرزند/والد) بین جفت ستون ها و یا مجموعه ای از ستون ها را، بیان می کند. هر مقداری در سطح فرزند با یک و فقط یک مقدار در سطح والد، ارتباط داده می شود(associate). یک ارتباط مرتبه ای، وابستگی تابعی از یک سطح یک سلسله مراتب به سطح بعدی در آن سلسله مراتب، است. یک بعد، کانتینری از ارتباطات منطقی بین ستون هاست، و هیچ فضای داده ای به آن انتساب داده نمی شود.
عبارت CREATE DIMENSION ، ذکر می کند که:

- شروط سطح چندگانه (Multiple level clauses) ، که هر کدام یک ستون و یا مجموعه ای از ستون ها را در بعد مشخص می کنند.
- یک و یا بیشتر از یک، شروط سلسله مراتبی (one or more hierarchy clauses) ، که ارتباط های والد/فرزندی را بین سطوح مجاور مشخص می کنند.
- شروط صفات اختیاری (optional attribute clauses) ، که هر کدام یک ستون اضافی و یا مجموعه ای از ستون ها را که وابسته به یک سطح خاص هستند، را مشخص می کنند.

ستون ها در یک بعد می توانند هم از همان جدول (denormalized) و هم از چندین جدول (fully or partially normalized) ، باشند. برای تعریف یک بعد بر روی ستون هایی از چندین جدول، با استفاده از شرط JOIN ، جدول ها را به هم متصل می کنیم.
به عنوان مثال، یک بعد زمانی نرمال شده می تواند شامل یک جدول تاریخ، یک جدول ماه، و یک جدول سال، با الحاق شروطی که هر ردیف تاریخ را یه یک ردیف ماه و هر ردیف ماه را به یک ردیف سال متصل می کند، باشد. در یک بعد زمانی نرمال نشده کامل، تاریخ، ماه، و سال همه در یک جدول

هستند. در هر صورت، چه نرمال شده و چه نرمال نشده، ارتباطات سلسله مراتبی بین ستون ها، لازم است که در عبارت CREATE DIMENTION ،مشخص شود.

تولید کننده ی متوالی (The Sequence Generator) :

این تولید کننده، سری متوالی از اعداد را فراهم می کند. از این تولید کننده، مخصوصا در محیط های چند کاربره برای تولید کردن اعداد متوالی یکتا و منحصر به فرد، بدون سر بار اضافی (overhead) دیسک ورودی/خروجی و یا قفل کردن تراکنش ها، استفاده می شود. به عنوان مثال، فرض کنید که دو کاربر به طور همزمان بخوانند ردیف های جدیدی را به جدول employees اضافه کنند. با استفاده از یک دنباله برای تولید کردن اعدادی منحصر به فرد برای ستون employees_id ، هر کاربر باید منتظر کاربر دیگر، برای وارد کردن عدد employee موجود بعدی، بماند. دنباله به طور اتوماتیک مقادیر درستی برای هر کاربر، تولید می کند.

بنابراین، تولید کننده دنباله، تسلسل را در جایی که عباراتی از دو تراکنش باید اعدادی متوالی در یک زمان تولید کنند، کاهش می دهد. در نتیجه، استفاده از تولید کننده توان عملیاتی (throughput) تراکنش ها را بهبود می بخشد، و زمان انتظار کاربران را به طور قابل توجهی کوتاه می کند.

اعداد دنباله از نوع اعداد صحیح در اوراکل هستند که حد اکثر ۳۸ رقم دارند. یک تعریف دنباله، اطلاعات عمومی از جمله موارد زیر را نشان می دهد:

- نام دنباله.
- آیا دنباله صعودی است، یا نزولی.
- فاصله بین اعداد.
- آیا اوراکل مجموعه ای از دنباله اعداد تولید شده را در حافظه نگه داری کند، یا نه.

اوراکل تعریف های همه دنباله ها را برای یک پایگاه داده خاص، به عنوان یک جدول فرهنگ داده منفرد، در tablespace سیستم، ذخیره می کند. بنابراین، تعریف همه دنباله ها، همیشه در دسترس هستند، چرا که tablespace سیستم، همیشه فعال (online) است.

دنباله اعداد، بوسیله عبارات SQL که به دنباله اشاره می کنند، قابل استفاده هستند. شما می توانید عبارتی را برای تولید کردن یک دنباله عدد جدید و یا استفاده کردن از دنباله موجود، صادر کنید (issue) . بعد از اینکه عبارتی در یک نشست کاربر، دنباله ای را تولید کرد، آن دنباله مخصوص فقط در همان نشست قابل دسترسی است. هر کاربری که به یک دنباله اشاره می کند، به دنباله عدد جاری هم دسترسی دارد.

دنباله اعداد، مستقل از جداول، تولید می شوند. بنابراین، یک تولید کننده دنباله، می تواند برای بیشتر از یک جدول مورد استفاده قرار گیرد. تولید دنباله عدد، برای تولید کردن اتوماتیک کلید های اصلی (primary keys) برای داده هایتان و نیز هماهنگ کردن کلید ها بین چندین ردیف و یا چندین جدول، مفید است.

کلمه مترادف (Synonyms) :

Synonym نامی مستعار برای هر جدول، دیدگاه، دیدگاه مادی، دنباله، پروسیجر، تابع، و یا پکیج است. از آنجا که synonym واقعا یک نام مستعار است، بنابراین به هیچ انباری به جز مکانی برای تعریفش در فرهنگ داده، نیاز ندارد.

Synonym ها اغلب برای امنیت و سادگی، به کار می روند. به عنوان مثال، آنها می توانند اعمال زیر را انجام دهند:

- نام و مالکیتی برای یک شئ ایجاد کنند.
- مکانی شفاف برای اشیاء راه دور از یک پایگاه داده توزیع شده، فراهم می کند.
- عبارات SQL را برای کاربران پایگاه داده، مشخص می کند.
- ایجاد محدودیتهای دستیابی مشابه روش ایجاد دیدگاه و کنترل دستیابی به صورت دقیق.

شما می توانید هم synonymهای عمومی (public) و هم اختصاصی(private) ، ایجاد کنید. یک synonym عمومی بوسیله گروهی از کاربران که PUBLIC نامیده می شود، و هر کاربر دیگری در پایگاه داده، قابل دسترسی است. اما یک synonym خصوصی فقط در شمای کاربر خاصی قرار دارد، که این کاربر قابلیت استفاده آن را برای دیگران کنترل می کند.

Synonym ها در هر دومحیط سیستم های توزیع شده و توزیع نشده، بسیار مفید هستند، چرا که آنها مشخصات اشیاء اساسی (underlying) ، که شامل مکانشان در یک سیستم توزیع شده است، پنهان می کنند. این امر دارای مزیت است، چرا که اگر قرار باشد شیئی مجددا نام گذاری شود و یا انتقال داده شود، در این صورت فقط کافیست synonym دوباره تعریف شود.

Synonym ها همچنین می توانند عبارات SQL را برای کاربران سیستم پایگاه داده توزیع شده، مشخص کنند. مثال زیر نشان می دهد که چگونه و چرا synonym های عمومی، اغلب بوسیله مدیران پایگاه داده برای مخفی کردن مشخصات یک جدول پایه و کاهش پیچیدگی عبارات SQL ، ایجاد می شوند. فرض کنید که:

- جدولی با نام SALES_DATA در شمای خصوصی کاربری، با نام JWARD ، داریم.
- برای این جدول امتیاز SELECT ، به صورت PUBLIC عرضه شده است.

با توجه به این موارد شما می توانید درخواستی به این جدول بوسیله عبارات SQL ، شبیه به درخواست زیر، داشته باشید.

```
SELECT * FROM jward.sales_data;
```

توجه داشته باشید که در مثال بالا jward ، نام شمایی است که جدول sales_data را شامل می شود.

فرض کنید که مدیر پایگاه داده، یک synonym عمومی، با استفاده از عبارت زیر ایجاد می کند:

```
CREATE PUBLIC SYNONYM sales FOR jward.sales_data;
```

بعد از اینکه synonym عمومی ایجاد شد، شما می توانید در خواستی که به جدول SALE_DATA می دهید، با استفاده از عبارت SQL ای، مانند عبارت زیر، باشد:

SELECT * FROM sales;

توجه کنید که synonym عمومی sales ، نام جدول و همچنین نام شمایی را که شامل جدول می شود، از دید پنهان می کند.

خوشه بندی(Clusters):

خوشه بندی روشی برای ذخیره سازی داده جدول است. یک خوشه، گروهی از جدولها را شامل می شود که به خاطر اینکه دارای ستونهای مشترکی هستند که عمدتا با هم مورد فراخوانی قرار می گیرند، از بلوک داده مشترکی استفاده می کنند. برای مثال جداول کارمند و دپارتمان ستون اشتراکی department_id دارند. زمانی که شما جداول کارمند و دپارتمان را خوشه بندی می کنید، اوراکل بطور فیزیکی تمام سطر های این جداول را در بلوک داده یکسان ذخیره می کند. شکل ۱۷-۱۰ جریان خوشه بندی این دو جدول را نمایش می دهد:

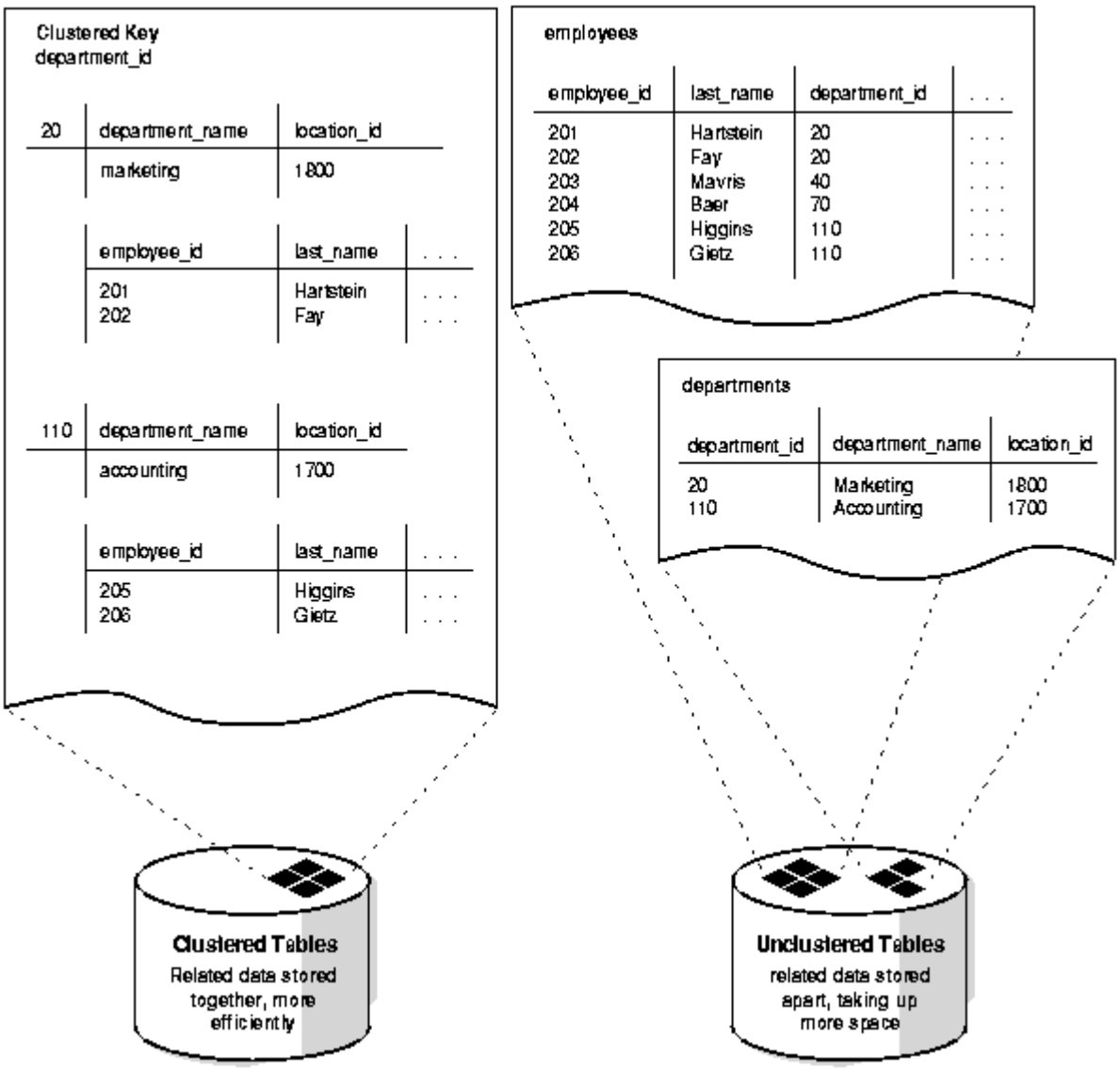

Figure 10-17 Clustered Table Data

چون خوشه ها داده های مربوط به هم در جداول مختلف را در یک بلوک داده ذخیره می کنند، استفاده از خوشه بندی مناسب دارای مزایای زیر می باشد:

- I/O دیسک برای الحاق جداول خوشه بندی شده پایین می آید.
- زمان دستیابی برای الحاق جدول های خوشه بندی شده پایین می آید.

- در یک خوشه، مقدار کلید خوشه، برابر با مقدار کلید در ستونها برای یک سطر مشخص است. هر مقدار کلید خوشه تنها یکبار در هر یک از خوشه و ایندکس خوشه ذخیره می شود. و مهم نیست که این کلید در چند سطر از جدول تکرار شده باشد. در نتیجه مقدار حافظه کمتری برای ذخیره جداول مربوط به هم نسبت به روش غیر خوشه بندی اشغال می شود. برای مثال در شکل ۱۷-۱۰ توجه کنید که چگونه هر کلید خوشه (department_id) برای سطرهای مختلف که در جداول کارمند و دپارتمان دارای این مقدار هستند، تنها یکبار ذخیره می شود.

خوشه های در هم سازی (Hash Clusters):

خوشه های در هم سازی، داده های جدول را شبیه روش ایندکس خوشه بندی معمولی (خوشه ها به وسیله ایندکس کلید دهی می شوند نه ب وسیله تابع در هم سازی) گروه بندی می کنند. به هر حال یک سطر در یک خوشه در هم سازی برمبنای نتیجه تابع درهم سازی بر کلید خوشه سطر ذخیره می شوند. تمام سطرهای با مقدار کلید یکسان بر روی دیسک در کنار یکدیگر ذخیره می شوند.

خوشه های در هم سازی گزینه بهتری نسبت به استفاده از جدول ایندکس شده یا جدول خوشه بندی شده با ایندکس هستند. به خصوص در زمانیکه یک جدول متناوبا مورد درخواستهای یکسان قرار می گیرد (برای مثال باز گرداندن تمام سطرهای دپارتمان شماره ۱۰). برای این درخواستها، کلید خوشه بندی مورد نظر در هم سازی می شود. نتیجه تابع در هم سازی به صورت مستقیم محل ذخیره سازی این سطور را بر روی دیسک نشان می دهد.

درهم سازی یک روش اختیاری برای ذخیره داده ها است که کارآیی بازیابی اطلاعات را افزایش می دهد. برای استفاده از در هم سازی کافیست یک خوشه درهم سازی ایجاد کنید و جداول را در آن بار گذاری کنید. اوراکل به طور فیزیکی سطرهای یک جدول را در خوشه در هم سازی ذخیره می کند و نتایج را نیز به وسیله حاصل تابع درهم سازی بازیابی می کند.

اوراکل از یک تابع درهم سازی برای تولید مقادیر عددی که مقدار درهم سازی نامیده می شوند استفاده می کند که این مقادیر را بر اساس مقادیر کلید خوشه تولید می کند. کلید یک خوشه در هم سازی، مانند کلید ایندکس خوشه، ممکن است مربوط به یک یا ترکیبی از چند ستون باشد. اوراکل برای یافتن و یا ذخیره سازی یک سطر در یک بلوک داده، مقدار حاصل از تابع در هم سازی بر روی

کلید خوشه محاسبه می کند. مقدار در هم سازی بدست آمده به یک بلوک داده در خوشه اشاره می کند که اوراکل عمل خواندن یا نوشتن را بر روی آن بلوک انجام می دهد.

یک خوشه در هم سازی، گزینه ای جایگزین برای جداول خوشه بندی نشده با ایندکس و یا با ایندکس خوشه است. بوسیله یک جدول ایندکس شده یا ایندکس خوشه، اوراکل محل سطرها را در یک جدول به وسیله مقادیری که اوراکل در ایندکس های جداگانه ذخیره کرده است بدست می آورد. برای یافتن و یا نوشتن یک سطر در یک جدول ایندکس شده و یا خوشه بندی شده با ایندکس، حداقل به دو عمل I/O نیاز است:

- یک و یا بیشتر عمل I/O برای یافتن و یا ذخیره کردن مقدار کلید از ایندکس
- یک عمل I/O دیگر برای خواندن و یا نوشتن سطر در جدول و یا خوشه

منابع:

Oracle9i Database Concepts (Oracle Corporation)
Oracle ۱۰g : Database Administration آموزش
www.oracle.com