

ওরাকল ডাটাবেজ অ্যাডমিনিস্ট্রেশন

মো: ইফতেখারুল আলম

(পূর্ব প্রকাশিতের পর)

পাত অ্যালোচনার বিভিন্ন স্টেটেরে স্ট্রাকচার এবং একের মাধ্যমে পরস্পর সম্পর্ক নিজে অ্যালোচনা করা হয়েছিল। এবারকার অ্যালোচনায় অংশলগত করা হয়েছে ডাটা স্টেটেরেজের বিভিন্ন মেথড ও তাদের সাথে সম্পর্কিত বিভিন্ন গুরুত্বপূর্ণ বিষয়, ইনডেক্স, ইন্ডিক্সিট কনস্টেন্ট নিয়ে।

ডাটাবেজ টেবিল ব্যবস্থাপনা : ওরাকল ডাটাবেজে ডাটা সংরক্ষণের বিভিন্ন মেথড রয়েছে। যেমন: রেঞ্জলার টেবিল, পার্টিশন টেবিল, ইনডেক্স অর্গানাইজড টেবিল, ক্লাস্টার টেবিল।

রেঞ্জলার টেবিল : সাধারণত আমরা সেলব টেবিলের ডাটা সংরক্ষণ করি তাই হলে রেঞ্জলার টেবিল। ডাটাবেজ অ্যাডমিনিস্ট্রেশনের ওপর টেবিলের রেঞ্জলার হার্ডওয়্যারের কোথায় কোথায় বিতরণ হবে তার খুব কমই নিয়ন্ত্রণ থাকে। টেবিলের অ্যাডমিনিস্ট্রেশন ওপর নির্ভর করে রেঞ্জলার যেকোনো অর্ডারে সংরক্ষিত হয়।

পার্টিশন টেবিল : বড় টেবিলে ফেলেতে ব্যাপক পরিমাণ গ্রেনেস একই সমুদয় সংঘটিত হচ্ছে ডাটা ম্যানুয়ালিট করে সেই সব টেবিলে এখনকার টেবিল দেখতে পাওয়া যায়। পার্টিশন টেবিলের সারিকে একধিক অংশে বিভক্ত করে মাস, কলেশজিট অথবা লিন্ট পর্যায়েই মাধ্যমে সংরক্ষণ করা হয়। পার্টিশন টেবিলের বিভিন্ন অংশ আলাদাভাবে একেকটি সোস্টেমট এবং তা বিভিন্ন টেবিল স্পেসের আওতাধর থাকে, যা আমরা বড় অ্যালোচনায় উল্লেখ করেছিলাম।

ইনডেক্স অর্গানাইজড টেবিল : এখানে কী (Key) ড্যাভুর ওপরে ভিত্তি করে ডাটা সংরক্ষণ হয়ে থাকে। এর জন্য আসা করা লুকআপ টেবিলের প্রয়োজন হয় না। কারণ সব ডাটাই ইনডেক্স ট্রি থেকে সার্ভেই হয়ে থাকে।

ক্রিয়েট টেবিল : কোনো টেবিল আমাদের নিয়ন্ত্রণের কমান্ড তৈরি করতে হলে অবশ্যই CREATE TABLE সিস্টেম ব্রিভিলেজ থাকতে হবে। অন্য কোনো ইউজারের ক্ষিমাতে তৈরি করতে হলে অবশ্যই আমাদের CREATE ANY USER সিস্টেম ব্রিভিলেজ থাকতে হবে। টেবিল তৈরির সময় খোয়াল রাখা প্রয়োজন যাকে টেবিলগুলো ভিন্ন ভিন্ন টেবিল স্পেসের আওতাধর থাকে এবং ফ্লাগমেন্টেশনের হার থেকে বলা পাবার জন্য লোকালি ম্যাসেজড টেবিল স্পেসের ব্যবহার নির্দিষ্ট করতে হবে। নিচে hr ডিভিডের employee টেবিল ক্রিয়েট কিভাবে করা যাবে তা দেখা যাবে।

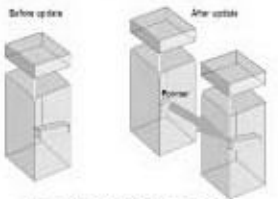
```
CREATE TABLE hr.employees (
employee_id NUMBER(5),
first_name VARCHAR2(20),
last_name VARCHAR2(25),
email VARCHAR2(25),
phone_number VARCHAR2(20)
hire_date DATE DEFAULT SYSDATE
job_id VARCHAR2(10)
```

```
salary NUMBER(8,2)
commission_pct NUMBER(2,2)
manager_id NUMBER(6)
department_id NUMBER(4)
);
```

ট্যেম্পোরারি টেবিল : যখন কোনো নির্দিষ্ট সেশন অথবা ট্রানজেকশনের জন্য বিশেষ কোনো টেবিলের ক্ষেত্রে এই টেবিলের প্রয়োজন হয় তখন একে ব্যবহার করা হয়। এর বিশেষ সুবিধা হলো একে কোনো ডিএমএল সেক বরাদ্দ থাকে না এবং এর ওপর ভিত্তি করে ইনডেক্স, ভিউ এবং ট্রিগার তৈরি করা যায়। নিচে ট্যেম্পোরারি টেবিল তৈরির কমান্ড দেয়া হলো।

```
CREATE GLOBAL TEMPORARY TABLE
hr.employees_temp
AS SELECT * FROM hr.employees;
```

রো মাইগ্রেশন এবং রো চেইনিং : রো মাইগ্রেশন এবং রো চেইনিং বেশ বড় একটি সমস্যা। যখন PCTFREE-এর জন্য ড্যাভুর কম নেয়া হয়, তখন কোনো অপডেট অপারেশন সংঘটিত হলে ডাটা ব-কে অপডেট স্পেসের সম্ভাবনা দেখা দিতে পারে। এসব ক্ষেত্রে ওরাকল সার্ভার তার স্বয়ংক্রিয় অক্টিভিটি সম্পূর্ণ রো-কে অন্য নতুন কোনো ব-কে স্থানান্তর করবে, যাতে তা নতুন স্থানকে নির্দেশ করতে পারে। একে বলা হয় রো মাইগ্রেশন। যখন কোনো রো মাইগ্রেন্ট হয় তখন ওই রো সংক্রান্ত ইনপুট/আউটপুট (আই/ও) প্যারামিটারে খারাপ হয়। কারণ এর ফলে ওরাকল সার্ভারকে অবশ্যই দুটি অপারেশন সম্পন্ন করতে হবে। রো চেইনিং তখনই সংঘটিত হবে যখন কোনো রো একটি ব-কের চেয়ে বড় হবে। সাধারণত যখন কোনো রো বেশ বড় অপারেশন কলাম দ্বারা করে তখনই রো চেইনিং বেশি দেখা যায়। এসব ক্ষেত্রে ওরাকল সার্ভার রো-কে ছোট ছোট অংশে বিভক্ত করে, যাকে বলা হয় রো পিন এবং প্রতিটি পিনকে আলাদা আলাদা ব-কে রাখা হয়। আর ওই সব ব-কে প্রয়োজনীয় পেরেন্টার স্থান করা হয়, যাতে একসাথে পুরো কলামকে স্ট্রাইইভ করা যায়। ডাটাবেজ-ক সাজিই বড় নিলে এ সমস্যা দূর করা যায়।



চিত্র-১ : রো মাইগ্রেশন এবং রো চেইনিং

ব-ক এবং স্টোরেজ ইন্ডিক্সাইজেশন প্যারামিটার পরিবর্তন : নিচের কমান্ড প্রয়োগ করে ব-ক ইন্ডিক্সাইজেশন প্যারামিটার পরিবর্তন

করা যায় :

```
ALTER TABLE hr.employees
PCTFREE 30
PCTUSED 50
STORAGE(NEXT 500K
NEXTENTS 2
MAXEXTENTS 100);
```

উপরের কমান্ড প্রয়োগ করার ফলাফল :

NEXT : যখন ওরাকল সার্ভার টেবিলের অন্য কোনো বরাদ্দ করবে তখন নতুন ড্যাভুর হবে। পরবর্তী সাজিই PCT INCREASE অনুযায়ী বাড়বে।

PCT INCREASE : এর যেকোনো পরিবর্তন ডাটা ডিকশনারিতে সন্নিবেশিত হয়। Next-এর পুনর্নির্ধারণ জন্য একে ব্যবহার করা হয়। একটি উদাহরণের মাধ্যমে আমরা ব্যাপারটা আরো সহজভাবে অনুধাবন করতে পারবো। ধরা যাক, কোনো টেবিলের দুইটি এক্সটেন্ট Next এবং PCT INCREASE যথাক্রমে 10k এবং 0k, যদি PCT INCREASE পরিবর্তন করে 100k করলে তৃতীয় এক্সটেন্ট হবে 10kx, চতুর্থটি হবে 20kx, পঞ্চমটি হবে 40kx এবং এভাবেই পরবর্তীগুলো হতে থাকবে।

MINEXTENTS : মাল বর্তমান ড্যাভুর সমান অথবা কম যেকোনো সংখ্যক মানই হতে পারবে। তবে টেবিলের ওপর এর কোনো তাৎক্ষণিক প্রতিক্রিয়া ঘটে না। তবে TRUNCATE একে ব্যবহার করা হয়।

MAXEXTENTS : বর্তমান এবং এর চেয়ে বেশি যেকোনো মাল হতে পারে। এমনকি এর মাল UNLIMITED হতে পারে।

নন-পার্টিশন টেবিল রিঅর্গানাইজেশন : যখন কোনো নন-পার্টিশন টেবিলকে রিঅর্গানাইজড করা হয়, তখন এর স্ট্রাকচার একই থাকে, তবে এর কোনো কনস্টেন্ট থাকে না। যখন কোনো টেবিলকে অন্য কোনো টেবিল স্পেসে স্থানান্তর অথবা রিঅর্গানাইজড করা হয় তখনই এর ব্যবহার হয়ে থাকে নিম্নরূপ। স্থানান্তরের পর এর কনস্টেন্ট আর দেখা যাবে না।

```
ALTER TABLE hr.employees
MOVE TABLESPACE data1;
```

ট্রানসকট টেবিল : এর ফলে টেবিলের সব রো ডিভিট হয়ে যাবে এবং ব্যবহার হওয়া স্পেস চেয়েই দ্রুত ও এর সাথে সম্পর্কিত ইনডেক্সও ট্রানসকট হয়ে যাবে।

```
TRUNCATE TABLE hr.employees;
```

ড্র্যাপ টেবিল : যখন কোনো টেবিলের প্রয়োজন আর থাকে না অথবা একে রিঅর্গানাইজড করা হয়, তখন ওই টেবিলকে ড্রপ করা হয়। কমান্ড টেবিল ড্রপের পর এর সাথে সম্পর্কিত এক্সটেন্ট উন্মুক্ত হয়ে যায়। CASCADE CONSTRAINTS অপশনটি অবশ্যই প্রয়োজন, যদি ওই টেবিলটি প্যারেন্ট টেবিল বা এবং এর সাথে অন্য টেবিলের ফরেন কী সম্পর্ক থাকে।

```
DROP TABLE hr.departments
CASCADE CONSTRAINTS;
```

ড্রপ কলাম : ড্রপ কলাম ওরাকলের একটি বিশেষ ফিচার। এর অংশে ওরাকল ৮ তারিখে এই ফিচারটি উপস্থিত ছিল না। তবে এ কথা মনে রাখা নরকার যে, ড্রপ কলাম কমান্ড চালানোর জন্য সিস্টেমে বেশ বড় আকারের

আনুষ্ট্র স্পেস প্রয়োজন। যেহেতু এর ফলে টেবিলের নির্দিষ্ট একটি কলামের সব ডাটা ডিলিট করতে হয়, তাই এটা সম্ভবসময়ক-ও পটে। CHECKPOINT বড় টেবিলের কলাম ড্রপের সময় আনুষ্ট্র স্পেসের ব্যবহার মিত্রিত করে। CHECKPOINT প্রতি ১০০০ রো পর পর সংঘটিত হবে। ওই অপারেশন সংঘটিতের সময় পর্যন্ত টেবিল INVALID স্ট্যাটাস থাকে। যদি কোনো কারণে এই অপারেশন চলার সময় ইনস্ট্যান্স ফেল করে টেবিল কিন্তু পরবর্তী স্ট্যাটাসের সময়ও INVALID থাকবে এবং ওই অপারেশন সম্পূর্ণ করবে।

```
ALTER TABLE hr.employees
DROP COLUMN comments
CASCADE CONSTRAINTS CHECKPOINT
1000;
```

টেবিল সংক্রান্ত তথ্য : নিম্নের স্ক্রী ডিউ থেকে টেবিল সংক্রান্ত তথ্য জানা যাবে।

```
01. DBA-TABLE
SELECT table_name FROM dba_tables WHERE
owner = 'HR';
TABLE_NAME
-----
COUNTRIES
DEPARTMENTS
DEPARTMENTS_HIST
EMPLOYEES
EMPLOYEES_HIST
JOBS
JOB_HISTORY
LOCATIONS
REGIONS
02. DBA-Objects :
SQL> select object_name, created
2 from DBA_OBJECTS
3 where object_name like 'EMPLOYEES'
4 and owner = 'HR';
OBJECT_NAME CREATED
-----
EMPLOYEES 16-APR-01
```

ইনডেক্স ব্যবস্থাপনা : এটি ব্যাপকভাবে ব্যবহার হওয়া ডাটাবেজ যেখানে প্রতিমিত্র ডাটা স্ট্রাইভ করতে হয়। সেখানে ইনডেক্স ব্যবস্থাপনা সূত্রেভাবে সম্পূর্ণ করা একজন ডিবিএর জন্য বেশ চ্যালেঞ্জিং কাজ। ইনডেক্স মূলত একটি ট্রি স্ট্রাকচার, যা কোনো টেবিলের নির্দিষ্ট রো-কে সরাসরি অ্যাক্সেস অনুমোদন করে। লজিক্যাল ডিজাইন অথবা ফিজিক্যাল ইমনি-ফরমেশনের ওপর ডিউ করে ইনডেক্সকে কয়েক ভাগে বিভক্ত করা হয়েছে।

ফিজিক্যাল : ০১. সিঙ্গেল কলাম, ০২. ইউনিক/নন-ইউনিক, ০৩. ফসহানভিত্তিক, ০৪. ভোয়েমইন।

ফিজিক্যাল : ০১. পার্টিশন/নন-পার্টিশন বি-ট্রি, ০২. পার্টিশন/নন-বিটম্যাপ।

লজিক্যাল ইনডেক্স অ্যান্ড-কেশন পারস্পেকটিভ থেকে। অপরদিকে ফিজিক্যাল ইনডেক্স কোন প্রক্রিয়ায় ইনডেক্স সংরক্ষিত হবে তা নির্দেশ করে। এ অ্যালোচনায় শুধু নন-পার্টিশন বি-ট্রি এবং ইনডেক্সের মধ্যে সীমাবদ্ধ রাখা হয়েছে।

বি-ট্রি ইনডেক্স : মূলত সব ইনডেক্স বি-ট্রি স্টার ব্যবহার করে তবে বি-ট্রি ইনডেক্স টার্মিট একটি ইনডেক্স যা কিনা প্রতিটি ক্রয়ের ROWID (রোআইডি) ধারণ করে তার সাথে সম্পর্কিত।

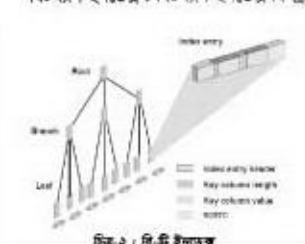
বি-ট্রি ইনডেক্সের গঠন : বি-ট্রি ইনডেক্স (চিত্র-১) থেকে দেখতে পাই এর ডিউটি অংশ, যার সর্বশেষে রয়েছে রুট। মূলত রুটে থাকে তার পরবর্তী ধাপের পরবর্তী লেভেলের ব-ক এন্ট্রি অর্থাৎ লিফ এন্ট্রির নির্দেশিত পয়েন্ট। লিভ নোড ধারণ করে ইনডেক্স এন্ট্রি যা কোনো টেবিলের রো-কে নির্দেশ করে।

ইনডেক্স লিভ এন্ট্রির ফরম্যাট : নিম্নলিখিত উপাদানের সমন্বয়ে ইনডেক্স লিভ এন্ট্রি গঠিত :

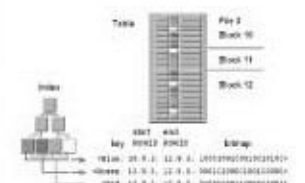
- ০১. এন্ট্রি হেডার : এর অভ্যন্তরে থাকে কলামের সংখ্যা এবং লকিংয়ের তথ্য।
- ০২. কী কলাম কেম্প : এটি ধারণ করে কী কলামের সাইজের তথ্য।
- ০৩. কী কলাম ডায়া : কী কলাম ডায়া ধারণ করে।

০৪. রোআইডি (ROWID) : রোয়ের আইডি ধারণ করে।

বিটম্যাপ ইনডেক্স : বিটম্যাপ ইনডেক্স বি-ট্রি



চিত্র-১ : বি-ট্রি ইনডেক্স



চিত্র-২ : বিটম্যাপ ইনডেক্স

ইনডেক্সের মতোই গঠিত। কেবল এর লিফ নোড একটি বিটম্যাপ ধারণ করে রোআইডির পরিবর্তে। বিটম্যাপ প্রতিটি বিট প্রতিমিত্র করে সম্ভাব্য রোআইডি-কে যদি বিটিটি সেট হয় তবে ওই রোতে সম্পর্কিত রোআইডি কী ডায়া ধারণ করছে। চিত্রের বর্ণনা অনুযায়ী লিভ নোড নিম্নলিখিত জিনিস ধারণ করে।

- ০১. এন্ট্রি হেডার : কলাম সংখ্যা এবং লক সম্পর্কিত তথ্য ধারণ করে।
- ০২. কী (Key) : রাতোক কী কলামের আকার এবং মান ধারণ করে। উদাহরণস্বরূপ বলা যায়, কী ধারণ করে শুধু একটি কলামের এবং প্রথম এন্ট্রি হচ্ছে ব-।

বিটম্যাপ ইনডেক্সের গঠন : স্টার্ট রোআইডির মধ্যে আছে ফাইল সংখ্যা জিন, ব-ক সংখ্যা দশ এবং রো সংখ্যা শূন্য। এপ্র রোআইডির মধ্যে আছে ব-ক সংখ্যা ধারণ এবং রো সংখ্যা অর্ট।

বিটম্যাপ সেগমেন্ট : এটি একটি বিটের স্ট্রিং ধারণ করে। যখন এই স্ট্রিংয়ের কোনো বিট সেট হয় তখনই ওই রো কী ডায়া ধারণ করবে আর আনসেট হবে যখন ধারণ করবে না।

বিটম্যাপ ও বি-ট্রি ইনডেক্স ব্যবহারের তুলনা : যেসব ক্ষেত্রে ডাটাবেজ উন্নয়নের ব্যবহার করা প্রয়োজন যেমন- OLTP (অনলাইন ট্রানজেকশন প্রসেস) সমন্বিত সিস্টেমে বিটম্যাপ ইনডেক্স ব্যবহার করা হয়। অপরদিকে ডাটা ওয়ারহাউজ যেখানে জটিল সব কোয়ারি বড় এবং স্ট্যাটিক টেবিলে হয় করে সেখানে বি-ট্রি ইনডেক্স ব্যবহার করা হয়।

বি-ট্রি ইনডেক্স নিম্নলিখিত কমান্ড রান করে করা যাবে :

```
CREATE INDEX hr.employees_last_name_idx
ON hr.employees(last_name)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE INDX;
```

বিটম্যাপ ইনডেক্স নিম্নলিখিত কমান্ড রান করে করা যাবে :

```
CREATE BITMAP INDEX orders_region_idx
ON orders(region_id)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE INDX
```

ইনডেক্সের স্টেটাস প্যারামিটার পরিবর্তন

```
(কমান্ড) :
ALTER INDEX employees_last_name_idx
STORAGE(NEXT 400K
MAXEXTENTS 100);
```

```
ইনডেক্সের স্পেস অ্যালোকেশন (কমান্ড) :
ALTER INDEX orders_region_idx
ALLOCATE EXTENT (SIZE 200K
DATAFILE 'YDISK6/idx01.dbf');
```

ইনডেক্সের স্পেস ডি অ্যালোকেশন

```
(কমান্ড) :
ALTER INDEX orders_idx
DEALLOCATE UNUSED;
```

ইনডেক্সের মূল গঠন (কমান্ড) :

```
ALTER INDEX orders_region_idx REBUILD
TABLESPACE INDX02;
```

ইনডেক্সের ড্রপ করা (কমান্ড) :

```
DROP INDEX hr.departments_name_idx;
```

ইনডেক্স সংক্রান্ত তথ্য : নিম্নলিখিত ডিউউপা পর্যালোচনা করলে ইনডেক্সের বিভিন্ন তথ্য উদগাটন করা যাবে।

DBA-INDEXES : ইনডেক্সের তথ্য জানা যাবে।

DBA-IND-COLUM : ইনডেক্স কলামের তথ্য জানা যাবে।

VOBJOCT-USA GC : ইনডেক্সের ব্যবহার জানা যাবে।

ডাটা ইন্টিগ্রিটি ব্যবস্থাপনা : যেকোনো ডাটাবেজের ডাটা এর বিজনেস নিয়মে দৃঢ়ভাবে আটকে রাখাই হলো ইন্টিগ্রিটি। প্রাথমিক তিন প্রক্রিয়ায় ডাটা ইন্টিগ্রিটি নিশ্চিত করা হয়। ০১. application code, ০২. Database trigger, ০৩. Declarative integrity constraint। সিস্টেমে ডিজাইনের ওপর ডিউ করে বিজনেস রুলের মার্মিৎ করা হয় এবং নির্মিত করা হয় যেকোনো একটি পপ। এখানে ডিবিএ-কে শুধু ডিজাইনের মাধ্যমে পর্যবেক্ষণ করা মেমডকে বাস্তবায়নে নিয়ন্ত্রিত থাকতে এবং ইন্টিগ্রিটি চাহিদার বিপরীতে সুস্থ পারফরমেন্স নিশ্চিত করতে হয়।

অ্যাপ্লিকেশন কোড হয় ডাটাবেজের অভ্যন্তরে স্টোর প্রসিডিউরে অথবা ট্রাigger এনে রানিং অ্যাপ্লিকেশনের মাধ্যমে প্রয়োগ করা হয়। অপরদিকে ডাটাবেজ ট্রিগার PL/SQL প্রোগ্রাম যা কিনা কোনো ইভেন্ট যেমন- ইনসার্ট, আপডেট কোন টেবিলে সংঘটিত হয় তখন ঘটে। তবে আগলেনায় ইন্টিগ্রিটি কনস্ট্র্যান্টের ওপর সীমাবদ্ধ রাখা হয়েছে।

ইন্টিগ্রিটি কনস্ট্র্যান্ট : প্রধানত পাঁচ ধরনের কনস্ট্র্যান্ট আছে। যেমন- Not Null, Unique, primary check ইত্যাদি।

Not Null : এর প্রয়োগের মাধ্যমে কোনো টেবিলের সব কলামে কোনো না কোনো মান নিশ্চিত করে। বাই ডিফল্ট সব কলামই Null ভ্যালুর উপস্থিতি অনুমোদন করে।

Unique : এর মাধ্যমে একটি নির্দিষ্ট কলামের প্রতিটি এন্ট্রিকে স্বাতন্ত্র্য করা হয়। এর ফলে কোনো কলামের দুইটি রো-তে ডুপি-কেট ভ্যালু থাকতে পারে না।

Primary : কোনো নির্দিষ্ট ডাটাবেজের একটি কলামে সর্বোচ্চ একটি Primary Key থাকতে পারে। এর ফলে নির্দিষ্ট কলামে দুইটি ডুপি-কেট ভ্যালু তিরোহিত হয় এবং কলামে কোনোভাবেই Null ভ্যালু ধারণ করবে না।

CHECK : এর মাধ্যমে কলামের সব ভ্যালুকে একটি শর্তের আওতাধীন থাকতে হয়।

FOREIGN KEY : একাধিক টেবিলের

মধ্যে সম্পর্কের মাধ্যমে স্থাপন করা হয়। ফরেন কী অন্য কোনো টেবিলের প্রাইমারি কীর ওপর ভিত্তি করে সম্পর্ক তৈরি করে। ফরেন কী যে টেবিলে তৈরি হয় তাকে বলা হয় চাইল্ড টেবিল আর যে টেবিল প্রাইমারি কী ধারণ করে তাকে বলা হয় প্যারেন্ট টেবিল।

কনস্ট্রইন্টের স্টেটসমূহ : এর নিচের উল্লি-খিত যেকোনো একটি অবস্থায় থাকতে পারে।

DISABLE NONVALIDATE : এ অবস্থায় সার্ভার কনস্ট্রইন্টকে চেক করবে না। এমনকি যদি কোনো নতুন ডাটা টেবিলে প্রবেশ করে অথবা আপডেট হয় তবেও কনস্ট্রইন্ট এর রুল মানবে না।

DISABLE VALIDATE : এ অবস্থায় কনস্ট্রইন্টের কলামে কোনো মেজিফিকেশন সম্ভব হয় না। সর্বোপরি ওই কলামের ওপর ভিত্তি করা ইনডেক্স ড্রপ হয়ে যায় এবং কনস্ট্রইন্ট ডিক্র্যাবল হয়।

ENABLE NONVALIDATE : এ অবস্থায় কোনো ডাটা কনস্ট্রইন্ট রুল ভঙ্গ করে ডাটাবেজে প্রবেশ করতে পারবে না।

নিচের কমান্ড দিয়ে দেখানো হলো কিভাবে কনস্ট্রইন্ট টেবিল ক্রিয়াকর্মের সময় ডিফাইন করা যায়।

```
CREATE TABLE hr.employee(
id NUMBER(7)
CONSTRAINT employee_id_pk PRIMARY KEY
```

```
DEFERRABLE
USING INDEX
STORAGE(INITIAL 100K NEXT 100K)
TABLESPACE indx,
last_name VARCHAR2(25)
CONSTRAINT employee_last_name_nn NOT
NULL,
dept_id NUMBER(7))
TABLESPACE users;
```

নিচের কমান্ডগুলো দিয়ে দেখানো হলো কিভাবে কনস্ট্রইন্টকে বিভিন্ন স্টেটে এনাল করা যায়।

```
ALTER TABLE hr.departments
ENABLE [NOVALIDATE/ VALIDATE] CON-
STRAINT dept_pk;
```

নিচের ডিউগুলো দিয়ে জালা যায় বিভিন্ন তথ্য।

- DBA_CONSTRAINTS
- DBA_CONS_COLUMNS

নিচের কমান্ড দিয়ে নাম টাইপ এবং স্ট্যাটাস জালা যায় (HR স্কিমার Employee টেবিলের)।

```
SQL> SELECT constraint_name,
constraint_type, deferrable,
2 deferred, validated
3 FROM dba_constraints
4 WHERE owner='HR'
5 AND table_name='EMPLOYEE';
CONSTRAINT_NAME C DEFERRABLE
DEFERRED VALIDATED
```

```
EMPLOYEE_DEPT.. R DEFERRABLE DEFERRED
VALIDATED
EMPLOYEE_ID_PK P DEFERRABLE IMMEDIATE
VALIDATED
SYS_C00565 C NOT DEFERRABLE IMMEDIATE
VALIDATED
3 rows selected.
```

ফিডব্যাক : Iftekhar@infobizsol.com