

Kapitel 7

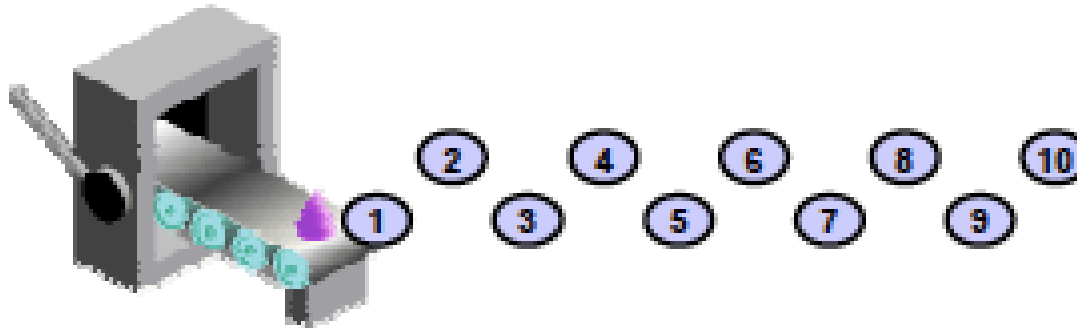
Weitere Datenbank-Objekte

Datenbankobjekte

Objekttyp	Beschreibung
Tabelle	Basiseinheit zum Speichern; besteht aus Zeilen
Sicht	Virtuelle Teilmenge der Daten von einer oder mehreren Tabellen
Prozedur Funktion Trigger	Gespeicherte ausführbare Einheiten
Package (Paket)	Zusammenfassung von gesp. Prozeduren/Funktionen und zugehörigen globalen Deklarationen
Sequenz	erzeugt eindeutige numerische Werte
Synonym	gibt Objekten alternative Namen
Index	erlaubt den direkten Zugriff von Attributwerten auf Tabellenzeilen; beschleunigt so Anfragen
u.a.	

Sequenzen

- können automatisch eindeutige Nummern erzeugen
- sind Objekte, die gemeinsam (shared) verwendet werden können
- werden vor allem zur Erzeugung von fortlaufenden Werten für Primärschlüssel genutzt
- ersetzen Anwendungs-Code
- erhöhen die Effizienz des Zugriffs auf Sequenzwerte, wenn diese im Speicher gecacht werden



Sequenzen (Forts.): Definition

Syntax:

```
create sequence Sequenz
    [increment by i]
    [start with s]
    [{maxvalue max | nomaxvalue}]
    [{minvalue min | nominvalue}]
    [{cycle | nocycle}]
    [{cache n | nocache}];
```

Beispiel: (zur Verwendung für DEPARTMENTS.department_id-Werte)

```
create sequence dept_deptid_seq
    increment by 10
    start with 120
    maxvalue 9999
    nocycle
    nocache;
```

Sequence created.

Sequenzen (Forts.): Verwendung

- Die Pseudospalte *Sequenz.nextval* gibt den nächsten verfügbaren Wert der Sequenz zurück. Dieser Wert ist immer eindeutig, auch für verschiedene Benutzer.
- Die Pseudospalte *Sequenz.currval* gibt den aktuellen Sequenzwert zurück.
- **nextval** muss für die Sequenz ausgeführt worden sein, bevor **currval** einen Wert beinhaltet.
- Es können Lücken in den Sequenzwerten entstehen, wenn:
 - ein Rollback auftritt
 - das System abstürzt
 - die Sequenz in einer anderen Tabelle mitgenutzt wird

Sequenzen: Verwendung (Forts.)

- Beispiel: Füge eine neue Abteilung namens "Support" mit der Location-ID 2500 ein:

```
insert into DEPARTMENTS  
  (department_id, department_name, location_id)  
values (dept_deptid_seq.nextval, 'Support', 2500);  
1 row created.
```

- Beispiel: Lies den aktuellen Wert der Sequenz dept_deptid_seq:

```
select dept_deptid_seq.currval  
from DUAL;
```

Sequenzen (Forts.): Modifikation

- z.B. **alter sequence dept_deptid_seq
increment BY 20
maxvalue 999999;**
Sequence altered.
- Man muss Eigentümer der Sequenz sein oder das Recht zum Ausführen des **alter**-Befehls für die Sequenz besitzen.
- Nur zukünftige Sequenznummern sind betroffen.
- Um eine Sequenz zu entfernen, nutzt man den **drop**-Befehl:
drop sequence dept_deptid_seq;
Sequence dropped.
- Die Sequenz sollte entfernt und re-definiert (**drop; create**) werden, wenn man mit einer anderen Nummer neu starten will (statt mehrmals **nextval** aufzurufen).

Sequenzen (Forts.): Informationen im Data Dictionary

describe USER_SEQUENCES

Name	Null?	Type
SEQUENCE_NAME	NOT NULL	VARCHAR2(30)
MIN_VALUE		NUMBER
MAX_VALUE		NUMBER
INCREMENT_BY	NOT NULL	NUMBER
CYCLE_FLAG		VARCHAR2(1)
ORDER_FLAG		VARCHAR2(1)
CACHE_SIZE	NOT NULL	NUMBER
LAST_NUMBER	NOT NULL	NUMBER

select sequence_name,min_value,max_value,increment_by,last_number¹
from USER_SEQUENCES;

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
LOCATIONS_SEQ	1	9900	100	3300
DEPARTMENTS_SEQ	1	9990	10	280
EMPLOYEES_SEQ	1	1.0000E+27	1	207

¹Die Spalte last_number beinhaltet die letzte im Cache vorab verfügbare Sequenznummer (bei **nocache** die nächste verfügbare).

Synonyme

- sind alternative Namen für Datenbank-Objekte
- können Namen verkürzen
- erlauben im Fall öffentlicher Synonyme (**public**) allen Benutzern, das Synonym zu nutzen anstatt “*Eigentümer.Objekt*” schreiben zu müssen, z.B. EMPLOYEES statt HRDB.EMPLOYEES

- Syntax:

```
create [public] synonym Synonym  
for      Objekt;
```

- Beispiel:

```
create synonym D_SUM  
for      DEPARTMENTAL_SALARY_SUM_VIEW;  
Synonym created.
```

- Entfernen eines Synonyms:

```
drop synonym d_sum;  
Synonym dropped.
```

Synonyme (Forts.): Informationen im Data Dictionary

describe USER_SYNONYMS ²

Name	Null?	Type
SYNONYM_NAME	NOT NULL	VARCHAR2(30)
TABLE_OWNER		VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
DB_LINK		VARCHAR2(128)

select *
from USER_SYNONYMS;

SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
EMP	ORA1	EMPLOYEES	

²Trotz der Spaltennamen TABLE ... dürfen Synonyme für beliebige Datenbank-Objekte vergeben werden.

Indexe

- werden auf einzelnen Spalten (oder Spaltenkombinationen) von Tabellen angelegt
- ermöglichen der Abfrageausführung und -optimierung, **direkt** von einem Attributwert (oder einer Wertkombination) auf die zugehörigen Tabellenzeilen **zuzugreifen**,
d.h. auf die Zeilen, in denen dieser Wert (oder diese Wertkombination) in der Index-Spalte (oder -Spaltenkombination) steht
- beschleunigen also Anfragen mit Bedingungen wie $A=x$ oder $A=B$ (Equijoins), teilweise auch Anfragen mit Vergleichen wie $A>x$
- können auch **sortierte** Durchläufe unterstützen (gemäß Index-Spalte)
- werden typischerweise durch Suchbäume implementiert, die auf externe Speicherung ausgelegt sind, insbes. durch **B*-Bäume**
- werden vom DBMS-Server automatisch genutzt und gewartet



Indexe (Forts.): Erzeugen von Indexen

- automatisch: Wenn man in einer Tabellendefinition ein **primary key**- oder ein **unique**-Constraint-angibt, wird automatisch ein “unique index” auf der zugehörigen Spalte(nkombination) erzeugt.

Ein solcher Index garantiert, dass keine Spaltenwerte doppelt vorkommen.

- manuell: Benutzer können allgemeine oder “unique” Indexe explizit definieren und damit sofort erzeugen, und diese wieder entfernen.

```
create [unique] index Indexname  
on Tabelle (Spalte[, Spalte ...]);          drop index Indexname;
```

- z.B.:

```
create index idx_emp_name  
on EMPLOYEES(last_name);  
Index created.
```

```
create index idx_best_wl3  
on BESTELLUNG(Ware,LName);  
Index created.
```

³unterstützt Zugriffe nach Ware oder nach {Ware ... **and** LName ...}; ein Index auf dem Primärschlüssel (KNr,LName,Ware) existiert bereits

Erzeugen von Indexen (Forts.)

Empfehlungen: Erzeuge einen Index, wenn:

+++ ++

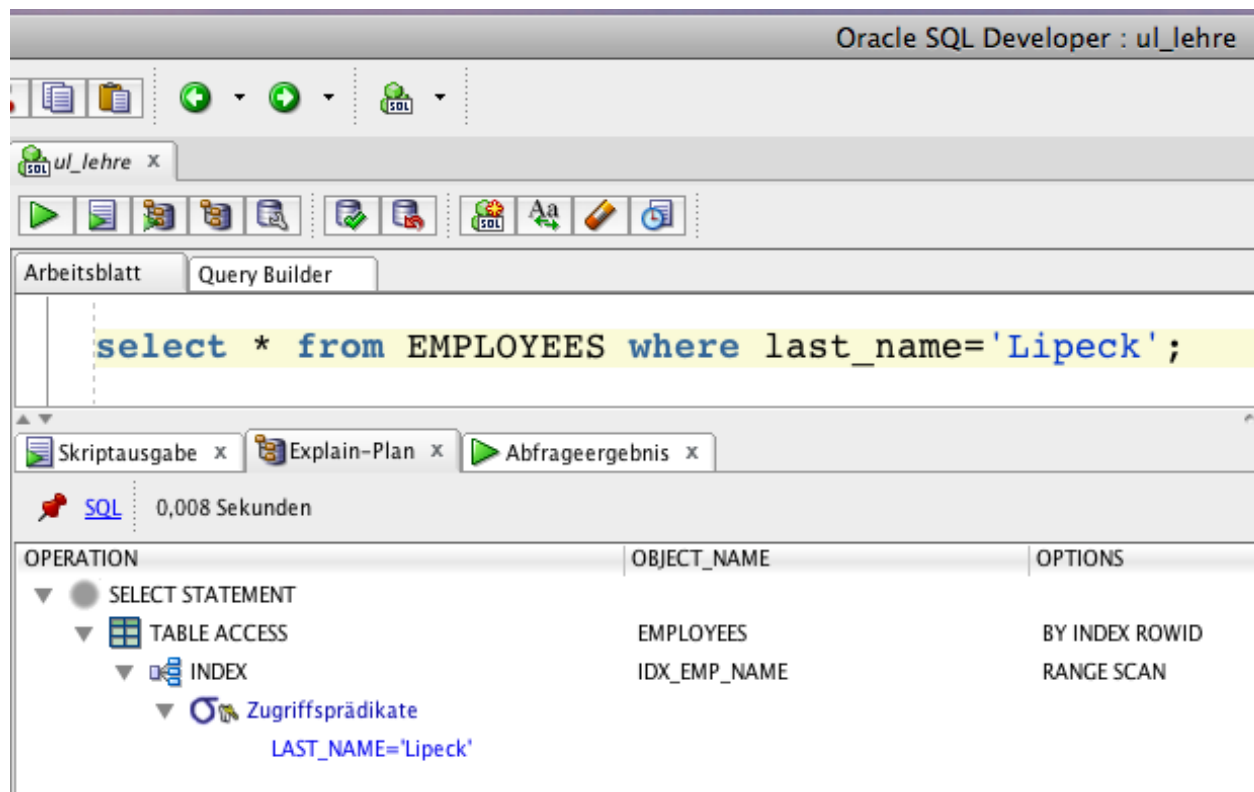
- eine Spalte eine Vielzahl von Werten beinhaltet
- eine Spalte viele **null**-Werte beinhaltet
- eine oder mehrere Spalten häufig zusammen (in Vergleichen wie $A=|>...$) in **where**-Klauseln oder Join-Bedingungen verwendet werden
- die Tabelle groß ist und die meisten Anfragen weniger als 2% bis 4% der Tabellenzeilen liefern.

Empfehlungen: Erzeuge keinen Index, wenn:

- die Spalten selten oder nur in Ausdrücken versteckt in Anfragebedingungen verwendet werden
- die Tabelle klein ist oder die meisten Anfragen mehr als 2% bis 4% der Tabellenzeilen liefern
- die Tabelle häufig aktualisiert wird.

Ausführungspläne

Man kann sich den vom Oracle-Optimierer ausgewählten Ausführungsplan zu einer Anfrage anzeigen lassen, und so u.a. feststellen, ob ein erzeugter Index wirklich vom DBMS benutzt wird; z.B. im SQL Developer:



The screenshot shows the Oracle SQL Developer interface with the title bar 'Oracle SQL Developer : ul_lehre'. The main window displays a query in the 'Arbeitsblatt' (Worksheet) tab: `select * from EMPLOYEES where last_name='Lipeck';`. Below the query, the 'Explain-Plan' window is open, showing the execution plan for the query. The plan indicates that the table 'EMPLOYEES' is accessed using the index 'IDX_EMP_NAME' via a 'RANGE SCAN' operation. The execution time is 0,008 Sekunden.

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	EMPLOYEES	BY INDEX ROWID
INDEX	IDX_EMP_NAME	RANGE SCAN
Zugriffsprädikate		
LAST_NAME='Lipeck'		