CREASE hingegen den Wert 0 besäße und jeder neue Extent lediglich eine Größe von 1 MB beanspruchte, würde es zu keiner Fehlermeldung kommen, bzw. die entsprechende Warnmeldung erst zu einem wesentlich späteren Zeitpunkt erscheinen. Abbildung 9-3 verdeutlicht die Auswirkung von PCTINCREASE.

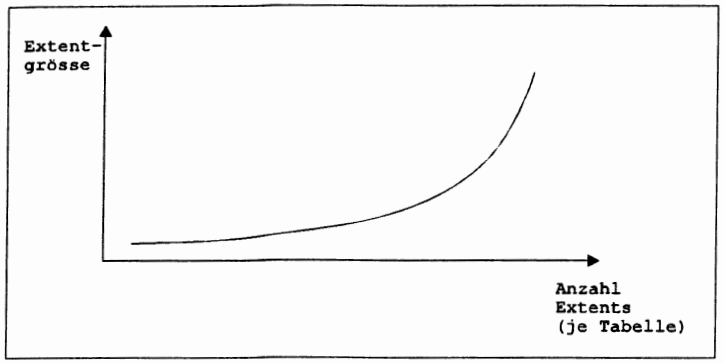

Abbildung 9-3: *Wachstum einzelner Extents mittels PCTINCREASE*

Abbildung 9-4 veranschaulicht den Zusammenhang zwischen Tabelle und Extents grafisch. Der dunkelgrau unterlegte Bereich ist ein komplett belegter Extent. Der hellgrau unterlegte Bereich ist ein Extent, der noch nicht vollständig gefüllt ist. Der Speicherplatz, den dieser Extent belegt, ist allerdings aus Sicht des Tablespace schon komplett belegt. Keine andere Tabelle kann Bytes in diesen Bereich des Tablespace schreiben.

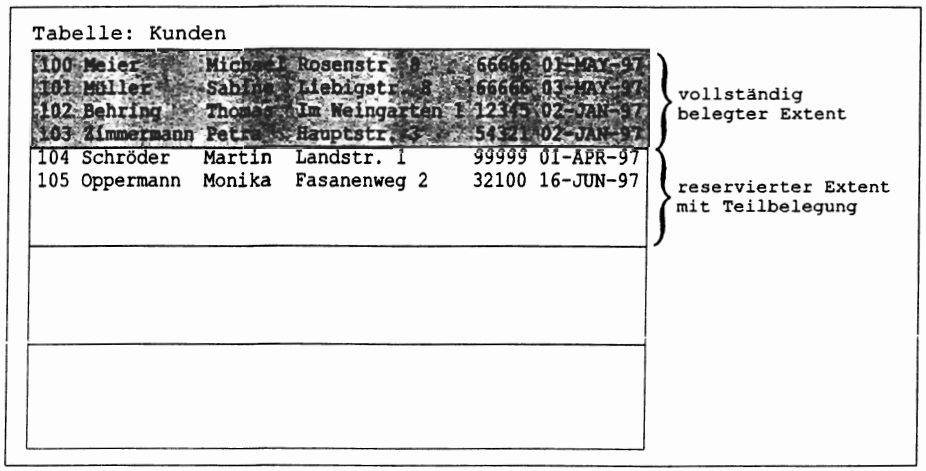

Abbildung 9-4: *Extents und Tabellendaten*

Hinweis:
Aus physikalischer Sicht stellt ein Index, der für eine Tabelle erzeugt wird, ebenfalls ein Objekt wie eine Tabelle dar. Aus diesem Grund können auch bei der Erzeugung eines Indexes die gleichen Speicherparameter angegeben werden, wie sie auch für die Erzeugung bzw. Veränderung von Tabellen gültig sind.

9.1.4 Blöcke

Blöcke bilden die kleinste Speichereinheit innerhalb einer Oracle-Datenbank. Die Größe eines Blocks ist auf 2048 Bytes voreingestellt. Sie kann jedoch in Abhängigkeit vom Betriebssystem variieren.

9.2 Interne Zeilennumerierung

Im Kapitel 6 haben Sie schon kurz die Verwendung des (internen) Typs *rowid* kennengelernt. Jeder Satz einer beliebigen Tabelle auf der Oracle-Datenbank kann über seine RowId angesprochen werden, der Wert ist immer ein eindeutiger Schlüssel für einen Satz. Die *rowid* jeder Zeile, also jeden Datensatzes, wird für den Aufbau von Indizes verwendet. Die Darstellung einer Rowid folgt einer festen Nomenklatur (Tabelle 9-2):

```
Block.Zeile.Datei
```

Element	Beschreibung
Block	Definiert den Block, in dem die jeweilige Zeile der Tabelle abgelegt ist.
Zeile	Definiert die Zeile innerhalb des Blockes. Die Zeilen werden dabei fortlaufend, beginnend bei 0 durchnummeriert.
Datei	Über *Datei* wird jede Datendatei innerhalb einer Oracle-Datenbank eindeutig gekennzeichnet.

Tabelle 9-2: *Rowid-Elemente*

Das folgende Listing zeigt die Verwendung von *rowid* am Beispiel der Kundentabelle:

```
SQL> SELECT rowid, name FROM kunden;

ROWID              NAME
------------------ --------------------
000000C6.0000.0002 Meier
000000C6.0001.0002 Müller
000000C6.0002.0002 Behring
000000C6.0003.0002 Zimmermann
000000C6.0004.0002 Schröder
000000C6.0005.0002 Oppermann
000000C6.0006.0002 Testkandidat

7 rows SELECTed.

SQL>
```

Hinweis:
Die Spalte RowId ist eine Tabellenspalte, die Oracle für interne Prozesse verwendet. Sie kann nicht über SQL-Anweisungen beschrieben werden! Eine einmal vergebene RowId für eine Tabellenspalte bleibt immer gleich, sofern die Tabelle nicht exportiert und in eine andere Datenbank importiert wird. Die Pseudospalte *rowid* kann aber durchaus als Abfragebedingung in WHERE-Klauseln verwendet werden.

Eine andere Art der Nummerierung einzelner Zeilen erreicht man durch die Verwendung von *rownum*. Über diese Pseudospalte erhalten Sie eine fortlaufende Nummerierung der Ergebnismenge einer Abfrage:

```
SQL> SELECT rownum, kundennr, name FROM kunden;

   ROWNUM  KUNDENNR NAME
--------- --------- --------------------
        1       100 Meier
        2       101 Müller
        3       102 Behring
        4       103 Zimmermann
        5       104 Schröder
        6       105 Oppermann
        7       106 Testkandidat

7 rows SELECTed.

SQL>
```


Bei veränderter Abfragebedingung verändert sich allerdings auch die Zeilennumerierung der Ergebnismenge. Im Vergleich zu der obigen Abfrage folgt jetzt eine Abfrage mit einer WHERE-Klausel:

```
SQL> SELECT rownum, kundennr, name FROM kunden
  2  WHERE kundennr >103;

   ROWNUM  KUNDENNR NAME
--------- --------- --------------------
        1       104 Schröder
        2       105 Oppermann
        3       106 Testkandidat

SQL>
```

Mit Hilfe dieser Pseudospalte können Sie bei bestimmten Abfragen die mengenorientierte Denkweise durch eine satzorientierte ersetzen. Ich stand vor einiger Zeit vor dem Problem, dass eine bestimmte Abfrage, in der zwei Tabellen miteinander verknüpft wurden, zwei gleiche bzw. ähnliche Datensätze als Ergebnis zurücklieferten. Da die selektierten Tabellenspalten sich in einer Spalte unterschieden, konnte ich nicht mit dem DISTINCT-Operator arbeiten. Für die weitere Verarbeitung benötigte ich jedoch nur den ersten Satz. Die folgende Selektion zeigt ein ähnliches Beispiel:

```
SQL> SELECT a.kundennr, a.name, b.fon
  2  FROM   kunden a,
  3         telefon b
  4  WHERE  a.kundennr=100
  5  AND    a.kundennr=b.kundennr;

 KUNDENNR NAME                                 FON
--------- ---------------------------------------- --------
      100 Meier                                 7777
      100 Meier                                 12899
```

Da in der Telefontabelle zwei Einträge vorhanden sind, erscheinen auch hier im Ergebnis zwei Zeilen. Die Verwendung der Pseudospalte ROWNUM ermöglicht es jetzt jedoch, eine eindeutige Identifizierung der einzelnen Zeilen vorzunehmen:

```
SQL> SELECT rownum, a.kundennr, a.name, b.fon
  2  FROM   kunden a,
  3         telefon b
  4  WHERE  a.kundennr=100
  5  AND    a.kundennr=b.kundennr;

   ROWNUM  KUNDENNR NAME                                 FON
--------- --------- ---------------------------------- --------
        1       100 Meier                              7777
        2       100 Meier                              12899
```

Noch deutlicher wird es anhand der folgenden Selektion:

```
SQL> SELECT rownum, a.kundennr, a.name, b.fon
  2  FROM   kunden a,
  3         telefon b
  4  WHERE  a.kundennr=100
  5  AND    a.kundennr=b.kundennr
  6  AND    rownum=1;

   ROWNUM  KUNDENNR NAME                              FON
--------- --------- --------------------------------- --------
        1       100 Meier                             7777
```

Es ist allerdings nicht möglich, mit Hilfe der Spalte ROWNUM Vergleiche durchzuführen, da sie *vor* der Bearbeitung von WHERE-Klauseln erzeugt wird. Oftmals steht man allerdings vor dem Problem, den folgenden Pseudocode in SQL zu übersetzen:

„Zeige mir die ersten 10 Sätze der Ergebnismenge" oder
„Zeige mir die nächsten 10 Sätze der Ergebnismenge".

Solche Art der Abfragen sind mit Hilfe der ROWNUM-Spalte nicht ohne weiteres zu realisieren, da in der WHERE-Klausel als Vergleichsoperator lediglich ein „<" stehen darf. Dieses Problem lässt sich allerdings durch Umformulierung und Verwendung des MINUS-Operators umgehen:

```
SQL> SELECT kundennr, name FROM kunden WHERE rownum < 5
  2  MINUS
  3  SELECT kundennr, name FROM kunden WHERE rownum < 2;

 KUNDENNR NAME
---------- ---------------
       101 Müller
       102 Behring
       103 Zimmermann

SQL>
```


9.3 Import / Export von Daten

In den vorherigen Abschnitten haben Sie die grundlegende Speicherstruktur einer Oracle-Datenbank kennengelernt. Im Vergleich zu einem dBase-System mit dem bekannten Dateiformat weist Oracle mit den .ORA-Dateien eine wesentlich komplexere Struktur auf. Man kann auf solche Dateien nicht mehr mit Betriebssystemroutinen zugreifen, um die Tabelleninhalte auszulesen. Das DBMS und die Datendateien bilden bei einer solchen Datenbank eine untrennbare Einheit. Vorteil dieses Prinzips ist die Wahrung von Sicherheitsmechanismen, da Anwender ausschließlich über das DBMS auf Dateninhalte zugreifen können. Demgegenüber steht ein erhöhter Arbeitsaufwand, wenn Daten in Form von Tabellen von einem System auf ein anderes portiert werden sollen. Während man in der dBase-Welt noch relativ einfach die entsprechenden .DBF-Dateien mit Betriebssystemroutinen kopieren konnte, bedient man sich bei Oracle hauseigener Werkzeuge. Es funktioniert nicht, einfach eine entsprechende .ORA-Datei auf ein anderes System zu kopieren! Um Informationen aus einer Datenbank in eine andere Datenbank zu portieren, müssen die entsprechenden Informationen aus der Quelldatenbank exportiert und in die Zieldatenbank wieder importiert werden. Für diese Arbeit gibt es in jeder Oracle-Version entsprechende Werkzeuge. Die Programme erfordern eine Anmeldung an der Datenbank, so dass das System überprüfen kann, ob der Anwender überhaupt eine entsprechende Berechtigung besitzt.

Bei den Im- und Export-Tools von Oracle 8 handelt es sich um die Programme IMP80.EXE und EXP80.EXE. Beide befinden sich im BIN-Verzeichnis der Oracle-Installation.

9.3.1 Benutzer und Tabellen exportieren

Im Verlauf dieses Buches haben Sie eine Reihe von Tabellen und weiterer Objekte angelegt. Sämtliche Objekte eines Benutzers sollen jetzt exportiert werden. Alle Tabelleninformationen (Struktur und Inhalt), sowie angelegte Indizes o.ä. werden also vom DBMS in eine Datei kopiert, dessen Format das Import-Tool von Oracle lesen kann. Starten Sie jetzt

exp chef/chef@orcl. kunde

chef0 export Tabellen
chef0 import Tabelle vom chef0

expdat.dmp umbenennen; dann fragt import-Befehl nach dem Dateinamen

man kann Tabellen in andere Datenbanken und andere Schemas auf anderen Rechnern importieren / exportieren