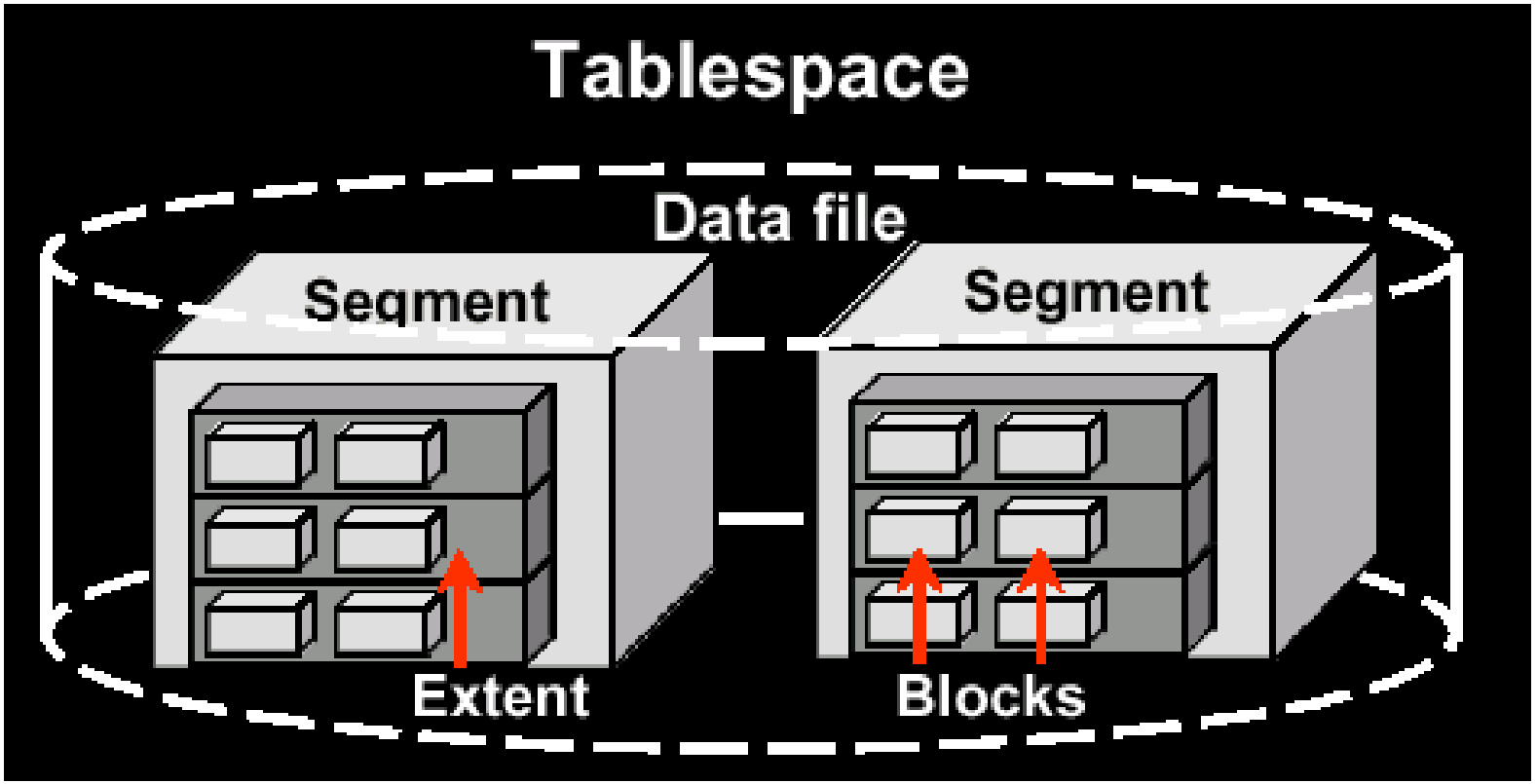
Storage Management stuff

Sukhjit Singh

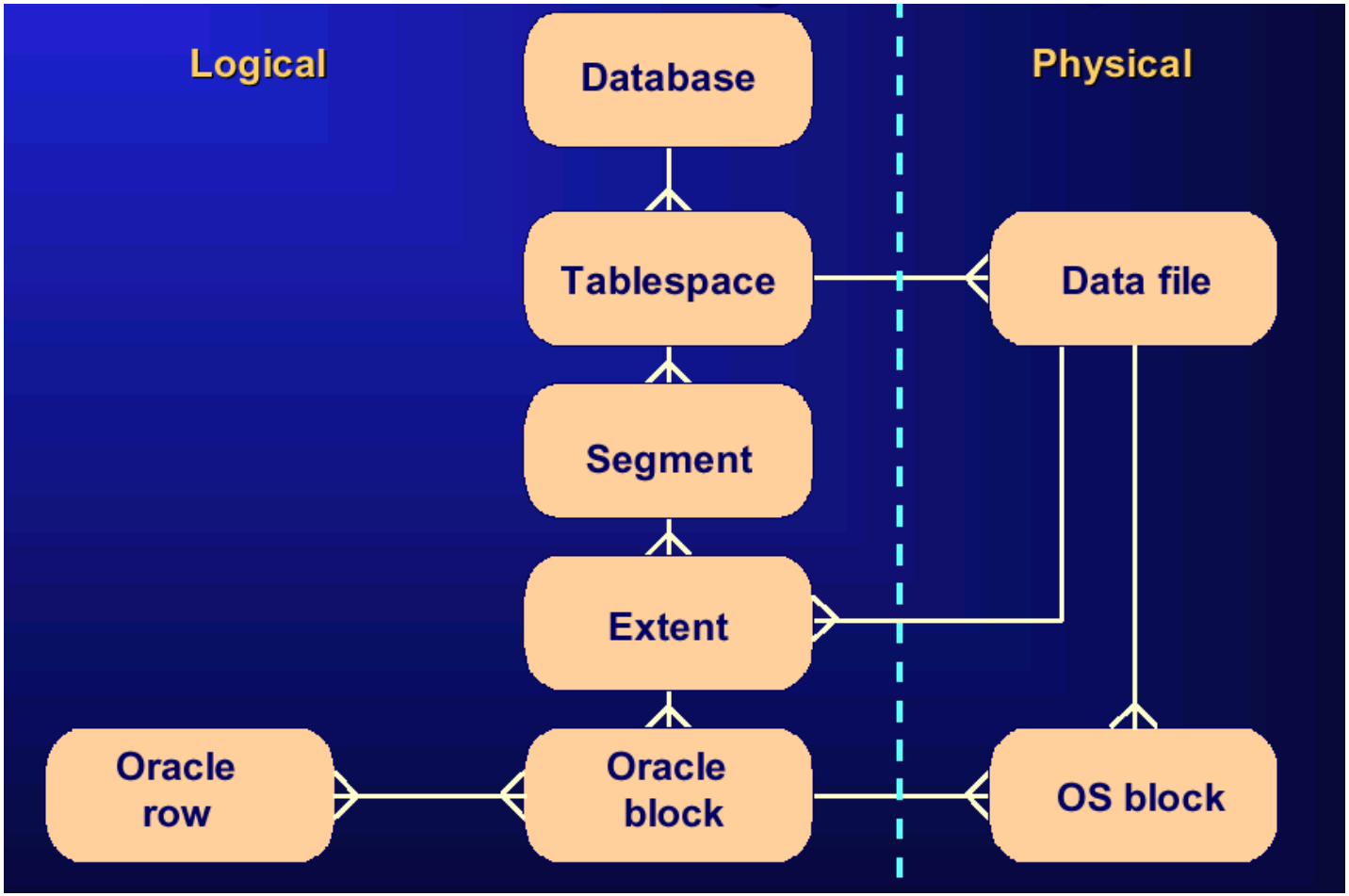
Agenda

- Data Storage Clause – including Storage Mgmt
- Sequences, Synonyms
- Indexes

Storage Management



Data Storage and Management



Segment Organization

- Application segments
  - With data: Table, cluster, table partition, nested table, LOB, IOT, IOT overflow
  - Used to access data: Index, index partition, IOT, LOB index
- System segments: Rollback, temporary, bootstrap

Storage Management Parameters

The following parameters influence the segment storage allocation:

- INITIAL: Size of the first extent in bytes
- NEXT: Starting size of subsequent extents
- PCTINCREASE: Percent increase of third and subsequent extents' size
- MINEXTENTS: Extents initially allocated
- MAXEXTENTS: Maximum extents allowed for a segment

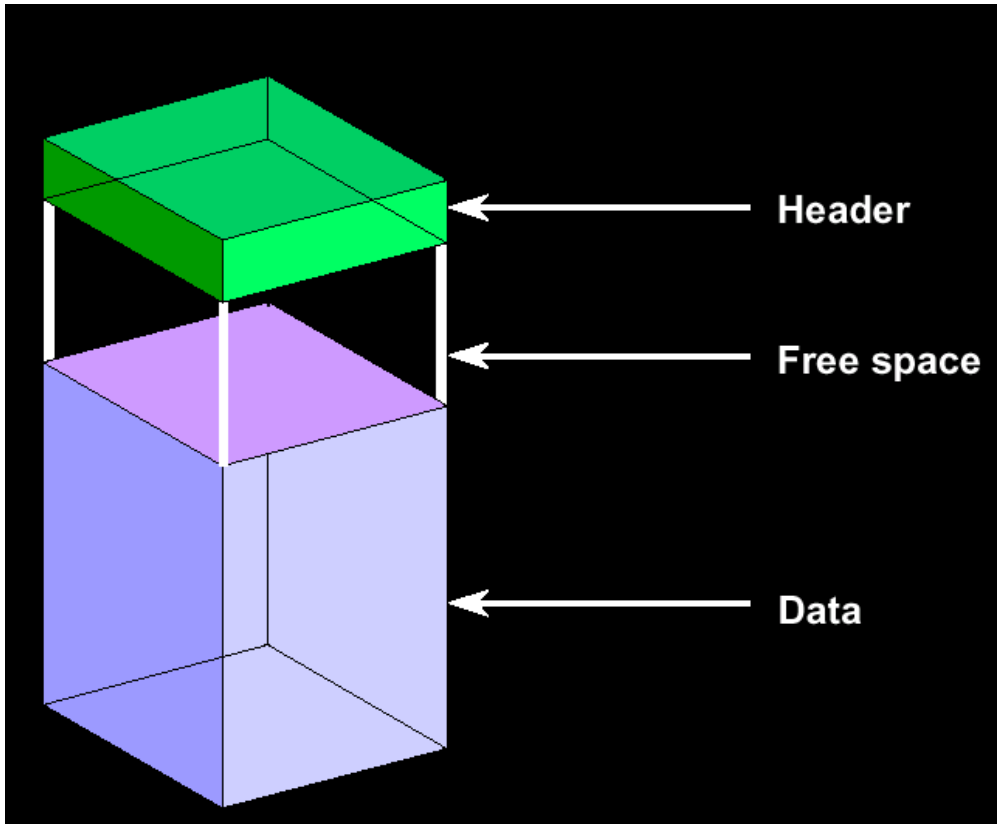
Storage Mgmt. – Example

```
CREATE TABLE emps (
     empno NUMBER(4), name VARCHAR2(30),
     deptno NUMBER(3), hire_date DATE )
  STORAGE
    ( INITIAL          2M
      NEXT             500K
      PCTINCREASE 0
      MINEXTENTS   1
      MAXEXTENTS   50 )
  TABLESPACE case_large_data;
```

Storage Clause

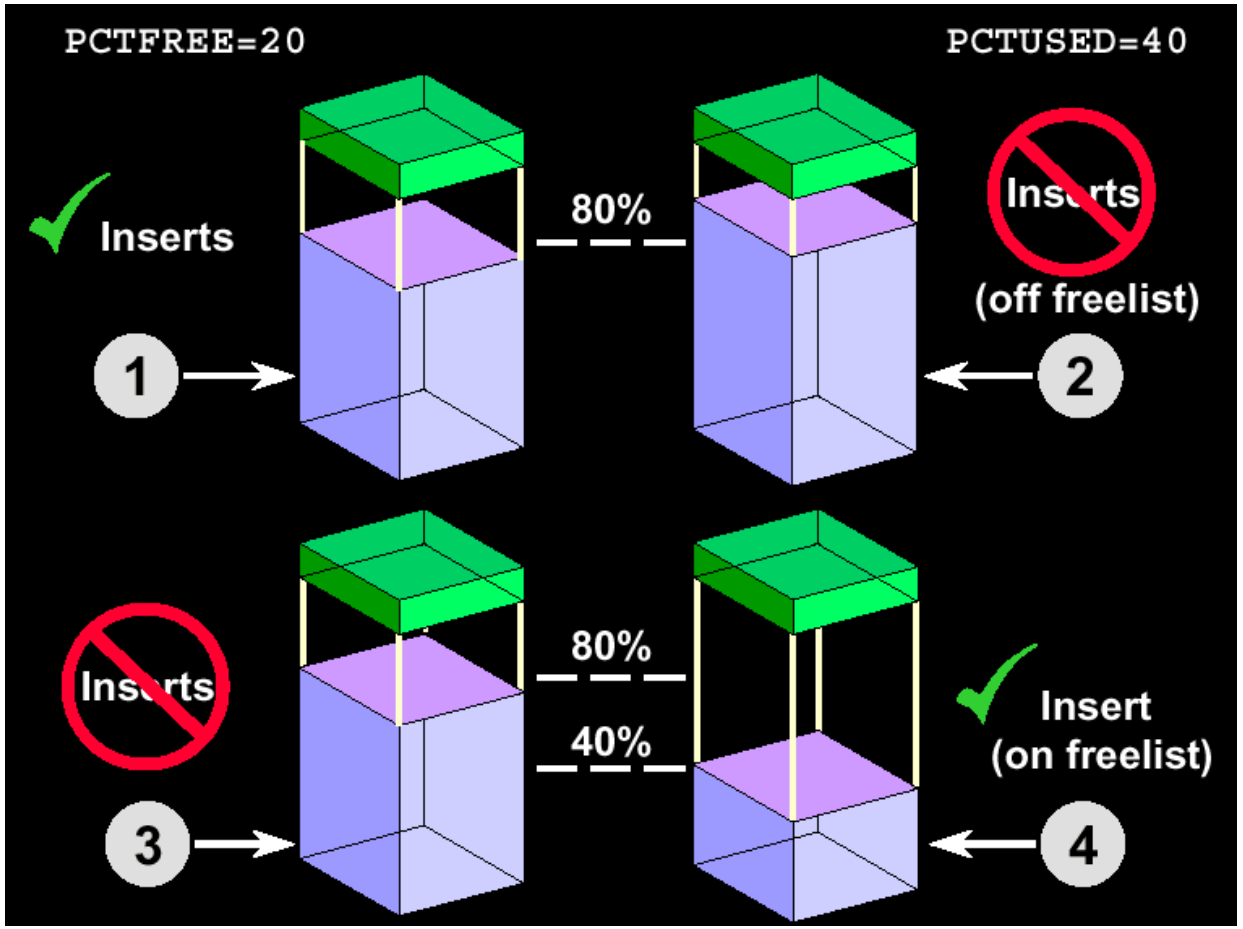
- Can be specified at Tablespace level and Table Level
- Default Storage Clause in “Create TableSpace” command, sets the default properties for Tables
- Blocks, Extents are allocated when a Segment is allocated.

DataBase Blocks Organization



- Consists of one or more data blocks organized as follows.

Storage Clause Usage



Storage Clause – Dictionaries

- **Data Dictionary Views**
  - DBA_TABLESPACES
  - DBA_DATA_FILES
  - DBA_SEGMENTS
  - DBA_EXTENTS
  - DBA_FREE_SPACE

SEQUENCES

SEQUENCES

- Automatically generates unique numbers
- Is a sharable object
- Is typically used to create a primary key value
- Replaces application code
- Speeds up the efficiency of accessing sequence values when cached in memory

Create Sequence

- Define a sequence to generate sequential numbers automatically.

```
CREATE SEQUENCE sequence
          [INCREMENT BY n]
          [START WITH n]
          [{MAXVALUE n | NOMAXVALUE}]
          [{MINVALUE n | NOMINVALUE}]
          [{CYCLE | NOCYCLE}]
          [{CACHE n | NOCACHE}];
```

Create Sequence - Example

- Create a sequence named DEPT_DEPTNO to be used for the primary key of the DEPT table.
- Do not use the CYCLE option.

```
SQL> CREATE SEQUENCE dept_deptno
  2          INCREMENT BY 1
  3          START WITH 91
  4          MAXVALUE 100
  5          NOCACHE
  6          NOCYCLE;
Sequence created.
```

Sequences – Dictionary View

- Verify your sequence values in the USER_SEQUENCES data dictionary table.

```
SQL> SELECT   sequence_name, min_value, max_value,
  2           increment_by, last_number
  3  FROM     user_sequences;
```

- The LAST_NUMBER column displays the next available sequence number.

NEXTVAL OR CURRVAL

- NEXTVAL returns the next available sequence value.

  It returns a unique value every time it is referenced, even for different users.

- CURRVAL obtains the current sequence value.

  NEXTVAL must be issued for that sequence before CURRVAL contains a value.

Using a Sequence

- Insert a new department named “MARKETING” in San Diego.

```
SQL> INSERT INTO     dept(deptno, dname, loc)
  2  VALUES          (dept_deptno.NEXTVAL,
  3                   'MARKETING', 'SAN DIEGO');
1 row created.
```

- View the current value for the DEPT_DEPTNO sequence.

```
SQL> SELECT   dept_deptno.CURRVAL
  2  FROM     dual;
```

Modify Sequence

```
SQL> ALTER SEQUENCE dept_deptno
  2          INCREMENT BY 1
  3          MAXVALUE 999999
  4          NOCACHE
  5          NOCYCLE;
Sequence altered.
```

Using a Sequence

- Caching sequence values in memory allows faster access to those values.
- Gaps in sequence values can occur when:
  - A rollback occurs
  - The system crashes
  - A sequence is used in another table
- View the next available sequence, if it was created with NOCACHE, by querying the USER_SEQUENCES table.
- Change the increment value, maximum value, minimum value, cycle option, or cache option.

Modify Sequence

- Must be the owner or have the ALTER privilege for the sequence.
- Only future sequence numbers are affected.
- The sequence must be dropped and re-created to restart the sequence at a different number.
- Some validation is performed.

Dropping Sequence

- Remove a sequence from the data dictionary by using the DROP SEQUENCE statement.
- Once removed, the sequence can no longer be referenced.

```
SQL> DROP SEQUENCE dept_deptno;
Sequence dropped.
```

Synonyms

Synonyms

- Simplify access to objects by creating a synonym (another name for an object).
  - Refer to a table owned by another user.
  - Shorten lengthy object names.

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

Synonyms

- Create a shortened name for the DEPT_SUM_VU view.

```
SQL> CREATE SYNONYM d_sum
  2  FOR    dept_sum_vu;
Synonym Created.
```

- Drop a Synonyms

```
SQL> DROP SYNONYM d_sum;
Synonym dropped.
```

Indexes

Index

- Is a schema object
- Is used by the Oracle Server to speed up the retrieval of rows by using a pointer
- Can reduce disk I/O by using rapid path access method to locate the data quickly
- Is independent of the table it indexes
- Is used and maintained automatically by the Oracle Server

Index

- Created automatically, when you define a PRIMARY KEY or UNIQUE constraint in a table definition.
- Created manually: Users can create nonunique indexes on columns to speed up access time to the rows.

Index

- An Index can be created one or more columns – using a composite key.

```
CREATE INDEX index
ON table (column[, column]...);
```

- Improve the speed of query access on the ENAME column in the EMP table.

```
SQL> CREATE INDEX    emp_ename_idx
  2  ON              emp(ename);
Index created.
```

When to create Indexes?

- The column is used frequently in the WHERE clause or in a join condition.
- The column contains a wide range of values.
- The column contains a large number of null values.
- Two or more columns are frequently used together in a WHERE clause or a join condition.
- The table is large and most queries are expected to retrieve less than 2–4% of the rows.

When not to create Indexes?

- The table is small.
- The columns are not often used as a condition in the query.
- Most queries are expected to retrieve more than 2–4% of the rows.
- The table is updated frequently.

Index Dictionaries

- The USER_INDEXES data dictionary view contains the name of the index and its uniqueness.
- The USER_IND_COLUMNS view contains the index name, the table name, and the column name.

```
SQL> SELECT   ic.index_name, ic.column_name,
  2           ic.column_position col_pos,ix.uniqueness
  3  FROM     user_indexes ix, user_ind_columns ic
  4  WHERE    ic.index_name = ix.index_name
  5  AND      ic.table_name = 'EMP';
```

Function Based Indexes

- A function-based index is an index based on expressions.
- The index expression is built from table columns, constants, SQL functions, and user-defined functions.

```
SQL> CREATE TABLE test (col1 NUMBER);

SQL> CREATE INDEX test_index on test(col1,col1+10);

SQL> SELECT col1+10 FROM test;
```

Remove Index

- Remove an index from the data dictionary.

```
SQL> DROP INDEX index;
```

- Remove the EMP_ENAME_IDX index from the data dictionary.

```
SQL> DROP INDEX emp_ename_idx;
Index dropped.
```

- To drop an index, you must be the owner of the index or have the DROP ANY INDEX privilege.

Unique vs Non-Unique Indexes

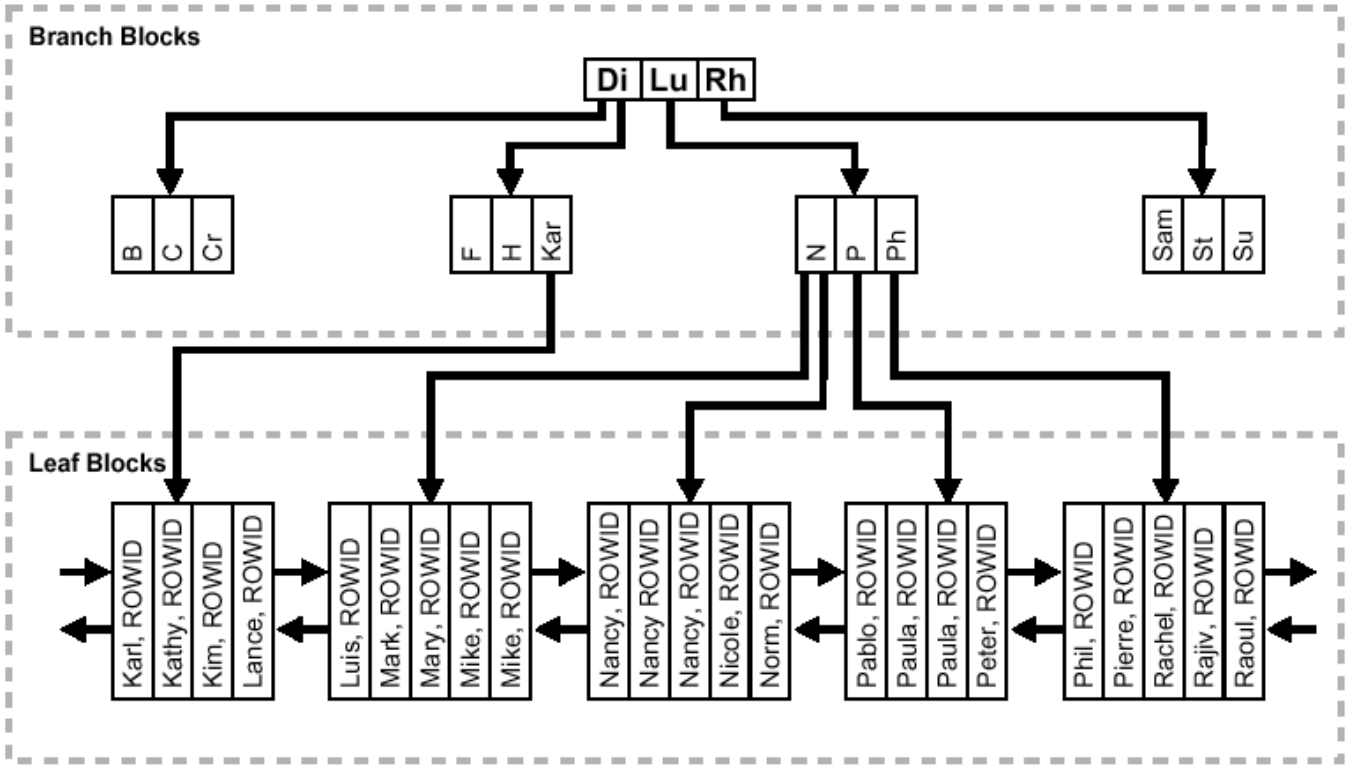
- Unique Indexes are created with Primary Keys. Assure no duplicates records are stored in table.
- Non-Unique Indexes impose no restriction on data columns.
- Unique Indexes can also be created explicitly on columns that unique constraint on them.
- NULL Values with Unique Indexes are considered to be distinct.

Index Storage

- B-tree indexes
- Bitmap indexes
- Bitmap Join indexes
- Indexes useful in clustering (Covered in future classes.)
  - B-tree cluster indexes
  - Hash cluster indexes
  - Reverse key indexes

Index Storage

- Balanced B-Tree



Index Storage

- Bitmap Index –
  - Columns with low cardinality are good as Bitmap Indexes.
  - Provide Good performance with And or OR Operations.
  - Columns with high cardinality are not good candidates for Bitmap indexes.
  - Bitmap Indexes include columns that have NULLS.

Index Storage

- Bitmap Index Example

CUSTOMER #	MARITAL_ STATUS	REGION	GENDER	INCOME_ LEVEL
101	single	east	male	bracket_1
102	married	central	female	bracket_4
103	married	west	female	bracket_2
104	divorced	west	male	bracket_4
105	s			
106	r			

REGION='east'	REGION='central'	REGION='west'
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0
0	1	0

Index Storage

"How many of our married customers live in the central or west regions?" '

```
SELECT COUNT(*) FROM CUSTOMER
    WHERE MARITAL_STATUS = 'married' AND REGION IN ('central','west');
```

status = 'married'		region = 'central'		region = 'west'						
0		0		0		0		0		0
1		1		0		1		1		1
1	AND	0	OR	1	=	1	AND	1	=	1
0		0		1		0		1		0
0		1		0		0		1		0
1		1		0		1		1		1

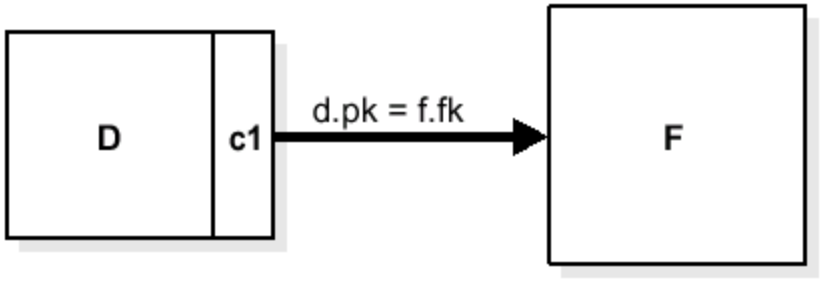
Index Storage

- Bitmap Join Index
  - A bitmap index created using two or more tables with a join.
  - Used Traditionally in Data-Warehousing.
  - Efficient in Storage given the amount of space taken is less.
  - Provides High performance since the indexes are pre-calculated.

Index Storage

- How to Bitmap Indexes work?

`Fi` -- Fact table *i*
`Di` -- Dimension table *i*
`pk` -- The primary key column on the dimension table
`fk` -- The fact table column participating in the join with the dimension tables




```
CREATE BITMAP INDEX bji ON f (d.c1) FROM f, d WHERE d.pk = f.fk

SELECT SUM(f.sales)
FROM d, f
WHERE d.pk = f.fk and d.c1 = 2
```