# Using Indexes in PROC SQL

# Creating and Managing Indexes in PROC SQL

- An *index* is an auxiliary file that stores the physical location of values for one or more specified columns (*key columns*) in a table. In an index, each unique value of the key column(s) is paired with a location identifier for the row that contains that value.
- An index can reduce the time to locate a set of rows, especially for a large table.
- PROC SQL uses the system of directions in an index to access specific rows in the table directly, by index value.
- You can create more than one index for a single table.
- All indexes for a SAS table are stored in one index file.

# Simple and composite indexes

- Simple index: an index that is based on one column. In PROC SQL, you must specify the name of a simple index the **same** as the name of the indexed column.
- Composite index: an index that is based on two or more columns. In the index, the values of the key columns are concatenated to form a single value. In PROC SQL, you must specify a unique name for the composite index that is **not the name** of any existing column or index in the table.

# Queries that can be optimized by using an index

| Query performance is optimized when the key column occurs in ... | Example |
|---|---|
| a *WHERE clause expression* that contains<br>☐ a *comparison operator*<br>☐ the *TRIM* or *SUBSTR function*<br>☐ the *CONTAINS operator*<br>☐ the *LIKE operator*. | `proc sql;`<br>`  select empid, jobcode, salary`<br>`  from sasuser.payrollmaster`<br>`  where jobcode='FA3'`<br>`  order by empid;`<br><br>*Key Column(s):* `JobCode` |
| an *IN subquery* | `proc sql;`<br>`  select empid, lastname, firstname,`<br>`  city, state`<br>`  from sasuser.staffmaster`<br>`  where empid in`<br>`  (select empid`<br>`  from sasuser.payrollmaster`<br>`  where salary>40000);`<br>*Key Column(s):* `EmpID` |
| a *correlated subquery*, in which the column being compared with the correlated reference is indexed | `proc sql;`<br>`  select lastname, firstname`<br>`  from sasuser.staffmaster`<br>`  where 'NA'=`<br>`  (select jobcategory`<br>`  from sasuser.supervisors`<br>`  where staffmaster.empid =`<br>`  supervisors.empid);`<br><br>*Key Column(s):* `Supervisors.EmpID` |
| a *join* in which<br>☐ the join expression contains the *equals (=) operator* (an *equijoin*)<br>☐ all the columns in the join expression are indexed in one of the tables being joined. | `proc sql;`<br>`  select *`<br>`  from sasuser.payrollmaster as p,`<br>`  sasuser.staffmaster as s`<br>`  where p.empid =`<br>`  s.empid`<br>`  order by jobcode;`<br><br>*Key Column(s):* `Payrollmaster.EmpID` or `Staffmaster.EmpID` |

# Create index syntax

**CREATE** **<UNIQUE>** **INDEX** *index-name*
**ON** *table-name* (*column-name-1<, ...column-name-n>*);

where

UNIQUE

is a keyword that specifies that all values of the column(s) specified in the statement must be unique.

*index-name*

specifies the name of the index to be created. If you are creating an index on one column only, then *index-name* must be the same as *column-name-1*. If you are creating an index on more than one column, then *index-name* cannot be the same as the name of any existing column or index in the table.

*table-name*

specifies the name of the table on which the index will be created.

*column-name*

specifies a column to be indexed. Columns can be specified in any order; however, column order is important for data retrieval. The first-named column is the primary key, the second-named column is the secondary key, and so on.

```
proc sql;
   create unique index EmpID
   on work.payrollmaster(empid);
quit;
```

(Simple)

```
proc sql;
   create index daily
   on work.marchflights(flightnumber, date);
quit;
```

(Composite)

# Managing index usage

- How SAS decides whether to use an index?

  Each time you submit a query that contains a **WHERE** class, SAS decides whether to use an index or to read all the observations in the data file sequentially

  - Identifies an available index or indexes.
  - Estimates the number of rows that would be qualified. If multiple indexes are available, SAS selects the index that it estimates will return the smallest subset of rows.
  - Compares resource usage to decide whether it is more efficient to satisfy the WHERE clause by using the index or by reading all the observations sequentially.

# Determine whether SAS is using an index

- To display information about indexes that have been defined and that have been used in processing the program, specify the SAS system option *MSGLEVEL=I*.

**OPTIONS MSGLEVEL=N | I;**

where

N
displays notes, warnings, and error messages only. This is the default.

I
displays additional notes pertaining to index usage, merge processing, and sort utilities along with standard notes, warnings, and error messages.

- Example:

```
options msglevel=i;
proc sql;
select *
      from marchflights
      where flightnumber='182';
quit;
```

```
42   options msglevel=i;
43   proc sql;
44   select *
45   from marchfilghts
46   where flightnumber='182';
INFO: Index daily selected for WHERE clause
optimization.
47   quit;
```

# Controlling index usage

- Using IDXWHERE=YES|NO
  - YES: tells SAS to choose the best index to optimize a WHERE clause, and to disregard the possibility that a sequential search of the table might be more resource-efficient.
  - NO: tells SAS to ignore all indexes and satisfy the conditions of a WHERE clause with a sequential search of the table.

```
proc sql;
  select *
  from marchflights (idxwhere=no)
  where flightnumber='182';
quit;
```

# Controlling index usage

- Using IDXNAME= to direct SAS to use a specified index.

```
proc sql;
    select *
    from marchflights (idxname=daily)
    where flightnumber='182';
quit;
```

# Drop indexes

**DROP INDEX** *index-name-1 <, ...index-name-2>*
**FROM** *table-name*;

where

*index-name*
specifies an index that exists.

*table-name*
specifies a table that contains the specified index(es). The *table-name* can be one of the following:

- a one-level name
- a two-level libref.table name
- a physical pathname that is enclosed in single quotation marks.

```
proc sql;
  drop index daily
  from work.marchflights;
quit;
```

# Practice

- Create a unique simple index on the Employee_ID column of the Employee_addresses table.
- Create a composite index on city, state, and country columns of the Employee_addresses table.
- Find all the information of the employees who live in Philadelphia or San Diego. Use the MSGLEVEL=I system option to determine if any index was used in the query.

# Solution

Simple index

Composite index

Which index is being used?