

Database Design II

Author: Joshua Thompson
Created: April 21, 2009
The LINGUIST List

Overview

- An index is a shortcut way for the database to find rows without having to read the table
- Indexes are created on columns or combinations of columns
- Know your data and how it will be accessed
- Proper indexing is vital for database performance
- Over-indexing can be worse than under-indexing

Creating Indexes

- Indexes are created using CREATE INDEX:

```
CREATE INDEX idx_employees1 ON employees (LastName,FirstName);  
CREATE UNIQUE INDEX idx_employees2 ON employees (EmployeeNumber);
```

- If you index on more than one column this is called a *composite index*
- The DBMS will not use a composite index unless you reference either the entire index or a leading prefix. In the above example, LastName would be a leading prefix.
- If you create a UNIQUE index then duplicate values for that column will not be permitted.

Primary Keys

- A primary key is an index that uniquely identifies the rows in a table
- A table can have only one primary key
- Primary keys are the target of foreign key relations
- A primary key is either a *natural key* or a *surrogate key*.
- Natural keys are part of the actual data, for example social security numbers or employee ID numbers.
- Surrogate keys are “made up” data, often in the form of a non-repeating sequence number or UUID added at the time the row was inserted.

Primary Keys

- The columns in a primary key cannot contain NULL values
- Primary keys can be created at table creation time or added later:

```
CREATE TABLE employees (  
    EmployeeID int unsigned,  
    FirstName char(50),  
    LastName char(50),  
    CONSTRAINT pk_employees PRIMARY KEY (EmployeeID)  
);
```

```
ALTER TABLE invoices  
ADD CONSTRAINT pk_invoices PRIMARY KEY (InvoiceID);
```

Natural vs. Surrogate Keys

- Like many database design issues there are pros and cons to each approach. It depends a lot on your data and how it is used (which is why you should know those things!)
- Surrogate keys are useful when the natural key is cumbersome to use; such as when the natural key consist of multiple columns.
- Surrogate keys are immutable; the same row will always have the same ID even if the data changes.
- Surrogate keys may be faster in some cases as they are usually integers and faster to index than strings.

Natural vs. Surrogate Keys

- Surrogate keys hold no meaning in and of themselves, which makes it more difficult for humans working with data directly.
- Surrogate keys may mean extra indexing is needed if the natural key is queried frequently.
- Surrogate keys can cause difficulties because the real world data is based on natural keys
- Surrogate keys that are sequential numbers can leak information about your data that you don't want people to know

Foreign Keys

- Foreign keys declare one or more columns in a table as containing the primary key of a row in another table
- The columns in the FK must exactly match those in the primary key of the foreign table in data type and order (though not in name.)
- Foreign keys help the DBMS enforce data integrity by insuring that the foreign key columns contain only values pointing to existing rows in the foreign table, and keep people from deleting rows referenced in other tables

Foreign Keys

- As with primary keys you can create foreign keys either at the time you create the table or at a later time via ALTER TABLE:

```
CREATE TABLE projects (  
    ProjectID int unsigned,  
    ProjectLeaderID char(50),  
    ProjectName char(50),  
    CONSTRAINT pk_projects PRIMARY KEY (ProjectID),  
    CONSTRAINT fk_projects1 FOREIGN KEY (ProjectLeaderID)  
        REFERENCES employees (EmployeeID);  
);
```

```
ALTER TABLE teams  
ADD CONSTRAINT fk_teams1 FOREIGN KEY (TeamLeaderID)  
REFERENCES employees (EmployeeID);
```

Choosing Good Indexes

- It is important to know the nature of your data and how it will be accessed
- Indexes speed up SELECT but slow down INSERT, UPDATE and DELETE
- Too much indexing can slow your database to a crawl
- For small tables (such as reference tables) an index may actually be slower than just reading the whole table
- If all the columns you are SELECTing are contained in the index then the DBMS can usually perform your query without ever reading the actual table

Ask your DBMS If Indexing is Right for YOU

- Most databases have a way to query what indexing will be used for a given SELECT statement. For example in MySQL it is “EXPLAIN SELECT ...” and in Oracle it is “EXPLAIN PLAN FOR SELECT ...”
- Links for further information:

<http://dev.mysql.com/doc/refman/5.0/en/explain.html>

<http://www.oracle-base.com/articles/8i/ExplainPlanUsage.php>