SQL Anywhere による 中国語、日本語、韓国語 データでの全文検索の実行

Hong Shi 、 Evguenia Eflov 著

2009年2月

このホワイトペーパーは、**SQL Anywhere 11**を対象に書かれました。

目次

概要

このマニュアルでは、CJK (Chinese, Japanese, and Korean ：中国語、日本語、韓国語) データが格納されているデータベースで全文検索を実行するための手順と補足情報について説明します。

SQL Anywhere データベースでは、MBCS (multi-byte character set ：マルチバイト文字セット) データをカラムに格納する際に、NCHAR、NVARCHAR、または LONG NVARCHAR のデータ型を使用できます。それ以外のデータ型 (CHAR、VARCHAR、および LONG VARCHAR など) を使用してカラムに CJK データを格納するには、データベースの作成時に以下のいずれかの照合を使用する必要があります。

言語	Windows照合	Unix照合
Japanese	932JPN	EUC_JAPAN
Korean	949KOR	EUC_KOREA
Chinese (Simplified)	936ZHO	EUC_CHINA
Chinese (Traditional - Hong Kong)	950ZHO_HK	EUC_TAIWAN
Chinese (Traditional - Taiwan)	950ZHO_TW	EUC_TAIWAN
Multiple languages	UTF8BIN	UTF8BIN
Multiple languages	UCA	UCA

default_char テキスト構成オブジェクトから設定を継承するテキスト構成オブジェクトを使用してテキスト・インデックスが作成される場合、CHAR データベース照合を使用して文字のデータ型のソートと比較が行われます。NCHAR、NVARCHAR、または LONG NVARCHAR のデータ型が格納されているカラムでは、CHAR 照合を使用するテキスト･インデックスを作成することはできません。初期化ユーティリティ（dbinit）の -z オプションを使用して、CHAR データベース照合の照合順序を指定します。

default_nchar テキスト構成オブジェクトから設定を継承するテキスト構成オブジェクトを使用してテキスト・インデックスが作成される場合、NCHAR データベース照合を使用して文字のデータ型のソートと比較が行われます。NCHAR 照合を使用するテキスト・インデックスは、どのようなデータ型のカラムでも作成することができます。ただし、NCHAR 照合を使用して作成されたテキスト・インデックスを使用する場合、データ変換により、CHAR、VARCHAR、および LONG VARCHAR のデータ型が格納されているカラムのインデックス付けのパフォーマンスに影響を及ぼす可能性があります。同じ CHAR および NCHAR 照合を使用するデータベースでは、カラムのインデックス付けのパフォーマンスが影響を受けることはありません。初期化ユーティリティ（dbinit）の -zn オプションを使用して、NCHAR データベースを指定します。

異なる CHAR 照合と NCHAR 照合を使用する 1 つのデータベースで、CHAR データベース照合を使用するテキスト・インデックスと、NCHAR データベース照合を使用するテキスト・インデックスを使用して、同じデータセットにインデックス付けを行うと、異なる用語や全文検索結果が生成される可能性があります。

CJK 文字では、大文字小文字やアクセントの有無の区別は直接的にはサポートされませんが、異なる CJK 文字間の比較にこれらの概念が使用されます。大文字小文字やアクセントの有無を区別しないように照合を調整し

た場合、SQL Anywhere は、各言語の要件に従って、それに相当するような文字の比較を言語内で行います。たとえば日本語の場合、区別しないように照合を調整すると、ひらがなとカタカナの同じ文字が一致します。ユニコード照合アルゴリズムおよび異なる文字列の比較と解析方法の詳細については、http://www.unicode.org/unicode/reports/tr10/ を参照してください。

前提条件

このマニュアルで説明されている手順を実行する前に、以下の情報をお読みになることをおすすめします。

- 「SQL Anywhere サーバ - データベース管理 ≫ データベースの設定 ≫ 国際言語と文字セット」の「照合の知識」
- 「SQL Anywhere サーバ - データベース管理 ≫ データベースの設定 ≫ 国際言語と文字セット ≫ 照合の知識」の「Unicode 照合アルゴリズム (UCA)」

上記の情報には、文字のソートおよびグループ化に使用される方法論の概要と、特定の照合を使用してデータベースを作成する場合の推奨事項が示されています。

このマニュアルの手順を実行するには、以下のソフトウェアが必要です。

•SQL Anywhere 11.0.1

CJK 文字セットが含まれている適切なユニコード・フォントをインストールして構成する必要があります。

Microsoft Windows オペレーティング・システムでユニコード・フォント・セットをインストールするには、

以下の手順を実行します。(※日本語版のWindowsでは既にインストール済みです)

1. **スタート > コントロール・パネル** をクリック
2. **地域と言語のオプション** をダブルクリック
3. **言語 タブ** をクリック
4. **東方アジア言語のファイルをインストールする(Install Files For East Asian Languages)** をクリック
5. **OK** をクリック

Unix オペレーティング・システムでユニコード・フォント・セットをインストールするには、readme_en.txt ファイル内の手順を参照してください。

CJK データが格納されているデータベースでの全文検索の設定および実行

以下の手順に従って、NGRAM テキスト・インデックスを設定し、CJK 文字の全文検索を実行します。NGRAM テキスト・インデックスを使用した全文検索の詳細については、SQL Anywhere のヘルプの「SQL Anywhere サーバ - SQL の使用法 ≫ データのクエリと変更 ≫ データのクエリ ≫ 全文検索のタイプ ≫ チュートリア

ル：NGRAM テキスト・インデックスへの全文検索の実行」を参照してください。

このチュートリアルのクエリの実行に必要な正しいアジア言語文字は、サンプル・データベースの uca.db および utf8.db に用意されています。このチュートリアル内の SQL 文は、サンプル・データベース・テーブルからデータをフェッチして使用します。したがって、SQL 文に定数文字列を含めたり、CJK 文字を入力する必要はありません。

このチュートリアルでは、uca.db と utf8.db の 2 つのサンプル・データベースを使用します。utf8.db サンプル・データベースは、CHAR データベース照合のデモンストレーションに使用します。cjkdata および queries テーブルは、どちらも同じデータベースに格納されています。cjkdata テーブルには、マルチバイトの非空白文字のみが格納されています。queries テーブルには、サンプルの CONTAINS クエリが格納されています。
UTF8BIN 照合の場合、以下のようなシングルバイトの非空白文字はインデックス付けされず、用語に現れません。

! " # $ % & ( ) * + , - . / : ;< = > ? @ [¥] ^ _ ` { | } ~

UCA 照合の場合、(空白文字に加えて) 以下のような文字は英数字とはみなされず、辞書に現れません。

- 未割り当て文字
- PUA (Private Use Area ：私用領域) 文字
- ユニコード規格でアルファベットとして定義されていない文字

サンプル・データベースの起動とサンプル・データの表示

1. サンプル・データベース・ファイルを新しいディレクトリ (c:/NGRAMsample/ など) にコピーします。
2. コマンド・プロンプトで、uca および utf8 データベースを起動するコマンドを実行します。たとえば、以下のようなコマンドを実行します。

```
dbeng11 c:¥NGRAMsample¥uca.db c:¥NGRAMsample¥utf8.db
```

3. コマンド・プロンプトで以下のコマンドを実行し、uca および utf8 データベースに接続します。

```
dbisql -c "ENG=uca;DBN=uca;UID=dba;PWD=sql"
dbisql -c "ENG=uca;DBN =utf8;UID=dba;PWD=sql"
```

4. 両方の Interactive SQL ウィンドウで以下の文を実行し、uca および utf8 データベース内のサンプル・テーブルのデータを表示します。

```
SELECT *
   FROM "DBA"."cjkdata" ORDER BY "id1";
```

5. 両方の Interactive SQL ウィンドウで以下の文を実行し、このチュートリアルで使用する CONTAINS クエリを表示します。

```
SELECT *
   FROM "DBA"."queries" ORDER BY "id";
```

テキスト構成オブジェクトとテキスト・インデックスの作成およびテスト

1. 両方の Interactive SQL ウィンドウで以下の文を実行し、uca および utf8 データベースに 2 組のテキスト構成オブジェクトを作成します。一方のテキスト構成オブジェクトは default_nchar から設定を継承し、もう一方は default_char から設定を継承します。

```
CREATE TEXT CONFIGURATION myNcharNGRAMTextConfig1 FROM default_nchar;
CREATE TEXT CONFIGURATION myNcharNGRAMTextConfig2 FROM default_nchar;
CREATE TEXT CONFIGURATION myCharNGRAMTextConfig1 FROM default_char;
CREATE TEXT CONFIGURATION myCharNGRAMTextConfig2 FROM default_char;
```

2. 両方の Interactive SQL ウィンドウで以下の文を実行し、TERM BREAKER アルゴリズムを NGRAM に、MAXIMUM TERM LENGTH (N) を適切な値に変更します。一般に、NGRAM の表意文字の長さは 2 か 3、アルファベットの長さは 4 か 5 です。したがって、中国語データの N の推奨値は 2 か 3、日本語または韓国語データの場合は 4 か 5 になります。クエリのタイプによっては、N に 1 を設定した方が良い場合もあります。1 文字よりも長い単語を使用するクエリは、N の値が 1 のテキスト・インデックスで実行されると処理が遅くなります。この例では、一方のテキスト構成オブジェクトでは N=1 を使用し、もう一方では N=2 を使用します。

```
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig1 TERM BREAKER NGRAM;
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig1 MAXIMUM TERM LENGTH 1;
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig2 TERM BREAKER NGRAM;
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig2 MAXIMUM TERM LENGTH 2;
ALTER TEXT CONFIGURATION myCharNGRAMTextConfig1 TERM BREAKER NGRAM;
ALTER TEXT CONFIGURATION myCharNGRAMTextConfig1 MAXIMUM TERM LENGTH 1;
ALTER TEXT CONFIGURATION myCharNGRAMTextConfig2 TERM BREAKER NGRAM;
ALTER TEXT CONFIGURATION myCharNGRAMTextConfig2 MAXIMUM TERM LENGTH 2;
```

3. 両方の Interactive SQL ウィンドウで以下の文を実行し、テキスト・インデックスの ngram1 を作成します。

```
CREATE TEXT INDEX ngram1 ON "DBA"."cjkdata"( "longvarchar1" )
  CONFIGURATION "DBA"."myNcharNGRAMTextConfig1";
```

このチュートリアルでは、テキスト・インデックスは IMMEDIATE REFRESH を使用します。ただし、基本となるデータがあまり変更されないかまたはわずかに変更される場合以外は、運用データベースでの IMMEDIATE REFRESH の使用はおすすめしません。

4. 両方の Interactive SQL ウィンドウで以下の文を実行し、両方のデータベースのテキスト・インデックスの用語が同じであることを確認します。

```
SELECT term, freq
  FROM sa_text_index_vocab( 'ngram1', 'cjkdata', 'DBA' )
  ORDER BY freq DESC, term ASC;
```

テキスト・インデックスの作成時に、N の値として 1 が設定されます。したがって、用語の各単語は 1 文字になります。マルチバイトの句読点は、英数字以外の文字であると解釈され、インデックス付けされません。

5. 両方の Interactive SQL ウィンドウで以下の文を実行し、queries テーブルから返されたアジア言語文字を格納するための全文検索変数を作成します。

```
CREATE VARIABLE q NVARCHAR( 100 );
```

返されたデータは、データベースの検索に使用されます。

6. 両方の Interactive SQL ウィンドウで以下の文を実行し、英数字の文字 1 文字を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 1;


SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

両方の Interactive SQL ウィンドウで、ワイド文字の B を含む 2 行が返されます。

両方の Interactive SQL ウィンドウで以下の文を実行し、英数字以外の文字 1 文字を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 2;


SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

ワイド文字の疑問符 (?) は辞書に存在しないため、どちらの Interactive SQL ウィンドウでも結果が返されません。

7. 両方の Interactive SQL ウィンドウで以下の文を実行し、ワイド文字の疑問符 (?) を含むデータが longvarchar1 カラムに存在することを確認します。

```
SELECT "q",*
  FROM "DBA"."cjkdata"
  WHERE longvarchar1 LIKE '%' || q || '%';
```

新しいテキスト・インデックスの作成およびテスト

1. utf8 の Interactive SQL ウィンドウで以下の文を実行し、既存のテキスト・インデックスを削除します。

```
DROP TEXT INDEX ngram1 ON "DBA"."cjkdata";
```

2. utf8 の Interactive SQL ウィンドウで以下の文を実行し、myCharNGRAMTextConfig1 テキスト構成オブジェクトを使用して新しいテキスト・インデックスを作成します。

```
CREATE TEXT INDEX "ngram1" ON "DBA"."cjkdata"( "longvarchar1" )
  CONFIGURATION "DBA"."myCharNGRAMTextConfig1";
```

3.両方の Interactive SQL ウィンドウで以下の文を実行し、各データベースのテキスト・インデックスの用語が異なることを確認します。

```
SELECT term, freq
  FROM sa_text_index_vocab( 'ngram1', 'cjkdata', 'DBA' )
  ORDER BY freq DESC, term ASC;
```

uca データベースの Interactive SQL ウィンドウで、19 行のデータが返されます。この結果は、NGRAM が選択されている場合に、CHAR データと NCHAR データの間に違いがないことを表しています。

utf8bin データベースの Interactive SQL ウィンドウで、23 行のデータが返されます。これは、すべての文字が有効な用語に属する文字として解釈されるためです。

4. 両方の Interactive SQL ウィンドウで以下の文を実行し、英数字の文字 1 文字を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 1;
```

```
SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

5. 両方の Interactive SQL ウィンドウで以下の文を実行し、英数字以外の文字 1 文字を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 2;

SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

utf8 データベースの検索で、ロー 1 と 4 が返されます。

6. 両方の Interactive SQL ウィンドウで以下の文を実行し、uca および utf8 データベースからテキスト・インデックスの ngram1 を削除します。

```
DROP TEXT INDEX "ngram1" ON "DBA"."cjkdata";
```

7. 両方の Interactive SQL ウィンドウで以下の文を実行し、myNcharNGRAMTextConfig2 テキスト構成オブジェクトを使用して新しいテキスト・インデックスを作成します。

```
CREATE TEXT INDEX "ngram2" ON "DBA"."cjkdata"( "longvarchar1" )
  CONFIGURATION "DBA"."myNcharNGRAMTextConfig2";
```

8. 両方の Interactive SQL ウィンドウで以下の文を実行し、uca および utf8 データベースの新しいテキスト・インデックスの用語が同じであることと、単語の長さが 2-grams であることを確認します。

```
SELECT term, freq
  FROM sa_text_index_vocab( 'ngram2', 'cjkdata', 'DBA' )
  ORDER BY freq DESC, term ASC;
```

両方の Interactive SQL ウィンドウで、15 行のデータが返されます。

9. 両方の Interactive SQL ウィンドウで以下の文を実行し、文字グループの最後の位置にある文字を

検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 3;

SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

指定された単語の長さが 2 であり、最後の文字がインデックス付けされないため、データが返されません。

10. 両方の Interactive SQL ウィンドウで以下の文を実行し、文字グループの最後の位置にある文字を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 3;

SELECT LIST( term, ' OR ' ) INTO q
  FROM dbo.sa_text_index_vocab( 'ngram2', 'cjkdata', 'DBA' )
  WHERE term LIKE '%' || q || '%';

SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

これらの文により検索用語が改良され、データベースごとに 2 行のデータが返されます。

11. 両方の Interactive SQL ウィンドウで以下の文を実行し、1 文字を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 1;

SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

検索パラメータが辞書と一致しないため、データが返されません。

12. 両方の Interactive SQL ウィンドウで以下の文を実行し、1 文字を検索します。

```
SELECT "contains_query" || '*' INTO q
  FROM "DBA"."queries"
  WHERE "id" = 1;

SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

検索パラメータが辞書と一致するため、両方の Interactive SQL ウィンドウで 2 行のデータが返されます。

13. 両方の Interactive SQL ウィンドウで以下の文を実行し、長さが 2 の単語を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 4;

SELECT "id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

14. 両方の Interactive SQL ウィンドウで以下の文を実行し、長さが 5 の単語を検索します。

```
SELECT "contains_query" INTO q
  FROM "DBA"."queries"
  WHERE "id" = 5;

SELECT "q","id1", "longvarchar1"
  FROM "DBA"."cjkdata"
  WHERE CONTAINS( "longvarchar1", q );
```

データベースごとに 2 行のデータが返されます。

15. utf8 の Interactive SQL ウィンドウで以下の文を実行し、既存のテキスト・インデックスを削除します。

```
DROP TEXT INDEX ngram2 ON "DBA"."cjkdata";
```

16. utf8 の Interactive SQL ウィンドウで以下の文を実行し、myCharNGRAMTextConfig1 テキスト構成オブジェクトを使用して新しいテキスト・インデックスを作成します。

```
CREATE TEXT INDEX "ngram2" ON "DBA"."cjkdata"( "longvarchar1" )
  CONFIGURATION "DBA"."myCharNGRAMTextConfig2";
```

この文では NGRAM が CHAR とともに使用され、その単語の最大長は 2 です。

17. 両方の Interactive SQL ウィンドウで以下の文を実行し、uca および utf8 データベースの新しいテキスト・インデックスの用語が異なることを確認します。

```
SELECT term, freq
  FROM sa_text_index_vocab( 'ngram2', 'cjkdata', 'DBA' )
  ORDER BY freq DESC, term ASC;
```

今回は UTF8BIN 照合を使用してテキスト・インデックスが作成されるため、utf8 データベースの用語にはマルチバイトの句読点が現れます。uca データベースは 15 行のデータを返し、utf8bin データベースは 22 行のデータを返します。

テキスト・インデックスおよびテキスト構成オブジェクトの削除

このマニュアルの手順を再度実行するには、テキスト・インデックスとテキスト構成オブジェクトを削除する必要があります。サンプル・データベースからテキスト・インデックスとテキスト構成オブジェクトを削除するには、両方の Interactive SQL ウィンドウで以下の文を実行します。

```
DROP TEXT INDEX ngram2 ON "DBA"."cjkdata";
DROP TEXT CONFIGURATION "DBA"."myCharNGRAMTextConfig1";
DROP TEXT CONFIGURATION "DBA"."myCharNGRAMTextConfig2";
DROP TEXT CONFIGURATION "DBA"."myNcharNGRAMTextConfig1";
DROP TEXT CONFIGURATION "DBA"."myNcharNGRAMTextConfig2";
```

まとめ

このマニュアルの手順を実行することにより、複数のテキスト・インデックスを作成およびテストし、CJK データが格納されているデータベースでの全文検索を実行しました。

法的注意