『日本語話し言葉コーパス』
RDB講習会 初級編

主催

国語研共同研究P「会話の韻律機能に関する実証的研究」（小磯花絵）
国語研共同研究P「多様な様式を網羅した会話コーパスの共有化」（伝康晴）
国語研共同研究P「コーパス日本語学の創成」（前川喜久雄）

共催

国立国語研究所コーパス開発センター

講習会の内容

- 『日本語話し言葉コーパス』RDB（CSJ-RDB）を研究に利用できるようになるために、次のことを学ぶ

  - CSJ-RDBの概要
  - SQL（RDBを操作するための言語）の基礎
    - SELECT文
    - WHERE句
    - JOIN句（内部結合）
  - Navicat（RDBを操作するためのソフトウェア）の使い方

第1部
CSJ-RDBの概要

CSJ-RDBについて

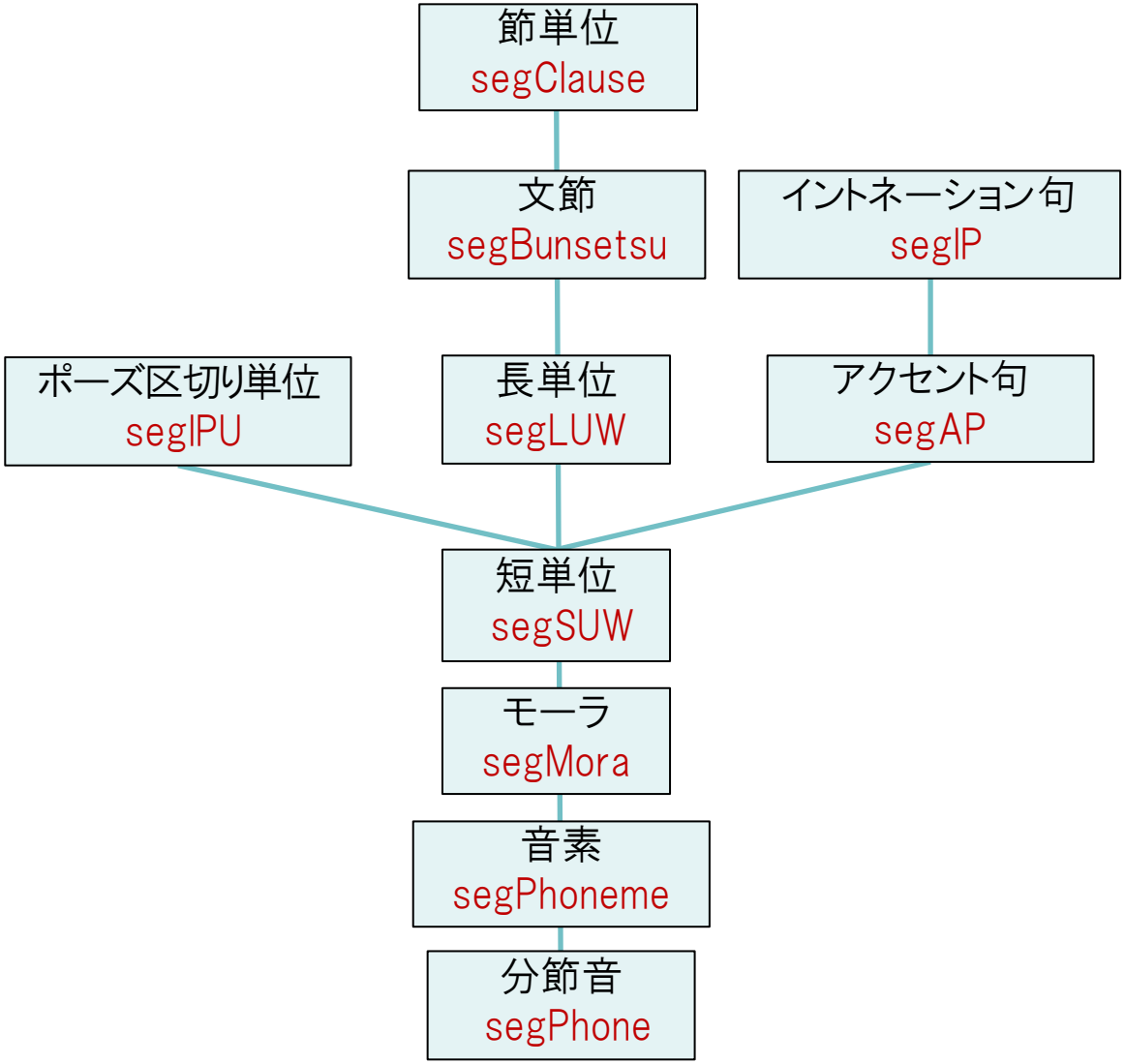
- CSJ-RDB とは
  - ✓ 『日本語話し言葉コーパス』のうちコア(201講演、約45時間)を対象
  - ✓ 第3刷のXML文書に含まれる情報をほぼ反映(若干、追加・修正)
- 参考資料
  - ✓ CSJ-RDBの概要・構成

    `http://www.ninjal.ac.jp/corpus_center/csj/data/` *
  - ✓ CSJ各種マニュアル

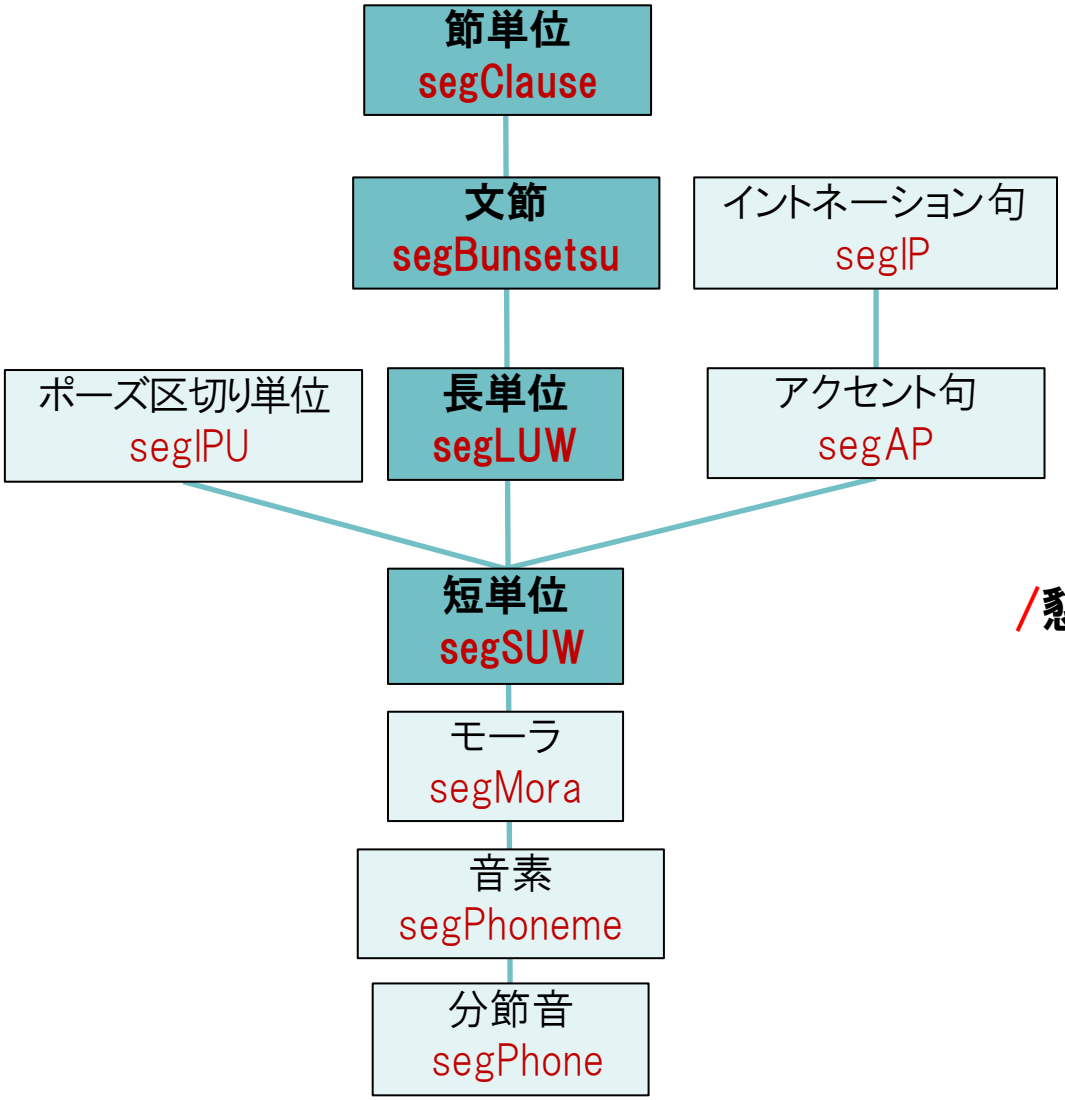
    `http://www.ninjal.ac.jp/corpus_center/csj/doc/` *

  * 近日中にCSJのURLが上記の通り変更(現在は"corpus_center/"の部分が不要)
- 今回配布した`csj_mini.db`(12講演分)についての注意事項
  - ✓ 二次配布は禁止
  - ✓ CSJ購入者以外は本データを使った研究発表は不可

CSJ-RDBの基本構成



CSJ-RDBの基本構成 ～統語・形態論の階層～



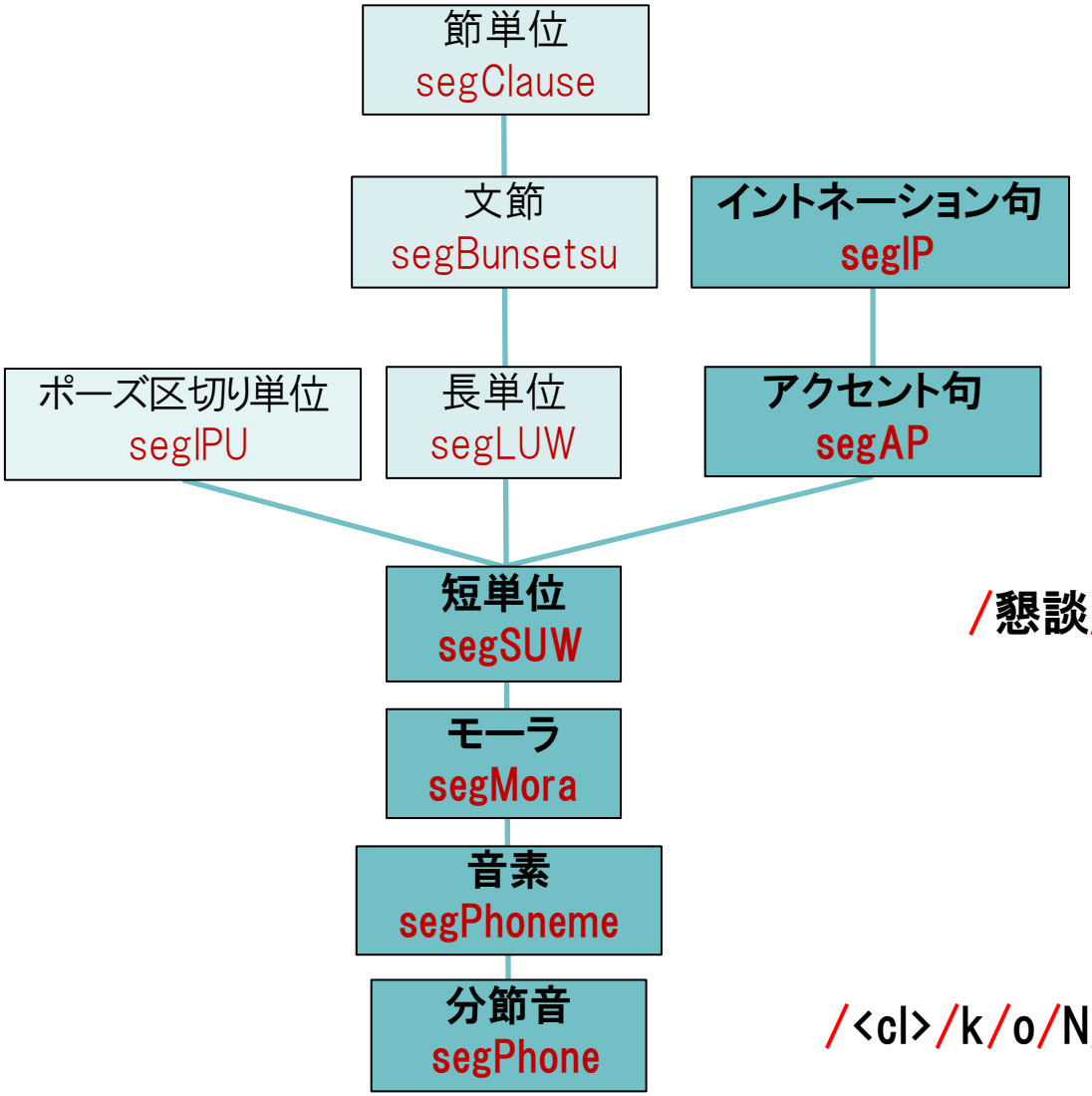
/懇談会に行ってもいいけど/

/懇談会に/行ってもいいけど/

/懇談会/に/行っ/てもいい/けど/

/懇談/会/に/行っ/て/も/いい/けど/

CSJ-RDBの基本構成 ～韻律・音韻の階層～



/懇談会に行ってもいいけど/

/懇談会に/行っても/いいけど/

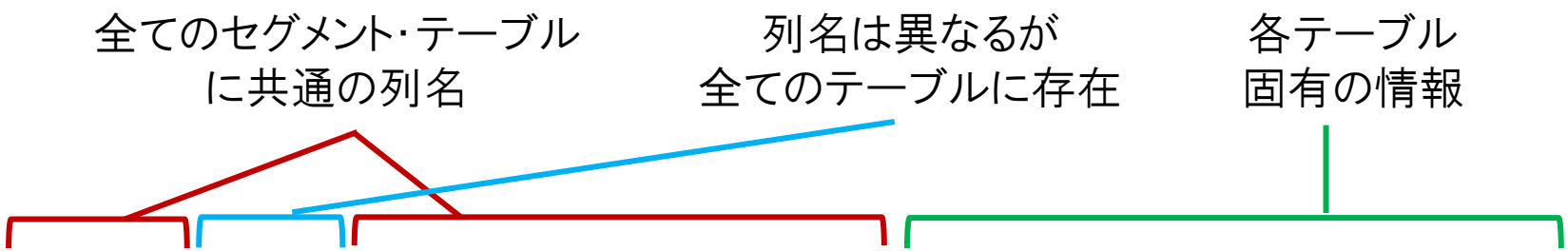
/懇談/会/に/行っ/て/も/いい/けど/

/コ/ン/ダ/ン/カ/イ/…/

/k/o/N/d/a/N/k/a/i/…/

/<cl>/k/o/N/<cl>/d/a/N/<cl>/k/a/i/…/

テーブルのサンプル ―segPhone―

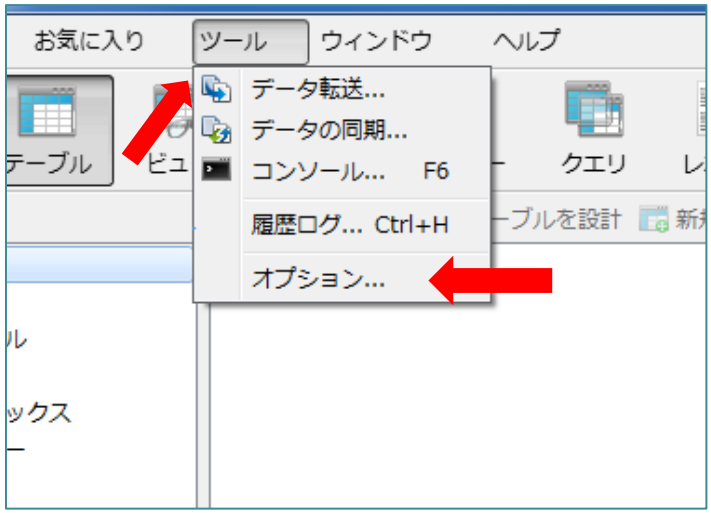


TalkID	PhoneID	StartTime	EndTime	Channel	PhoneEntity	PhoneClass	Devoiced	…
A01F0055	05551385L	5.551385	5.632454	L	h	consonant	0	
A01F0055	05632454L	5.632454	5.698874	L	a	vowel	0	
A01F0055	05698874L	5.698874	5.760029	L	Q	special	0	
A01F0055	05760029L	5.760029	5.821184	L	SclS	others	0	
A01F0055	05821184L	5.821184	5.837566	L	py	consonant	0	
A01F0055	05837566L	5.837566	5.907904	L	o	vowel	0	
A01F0055	05907903L	5.907903	5.978241	L	H	special	0	
A01F0055	05978241L	5.978241	6.028152	L	sj	consonant	0	
A01F0055	06028153L	6.028153	6.078064	L	i	vowel	1	
A01F0055	06078064L	6.078064	6.16653	L	m	consonant	0	
A01F0055	06166530L	6.16653	6.277931	L	a	vowel	0	
A01F0055	06277931L	6.277931	6.382126	L	s	consonant	0	
A01F0055	06382126L	6.382126	6.532801	L	u	vowel	0	

⇒ Navicat を使ってセグメント･テーブルを見てみよう

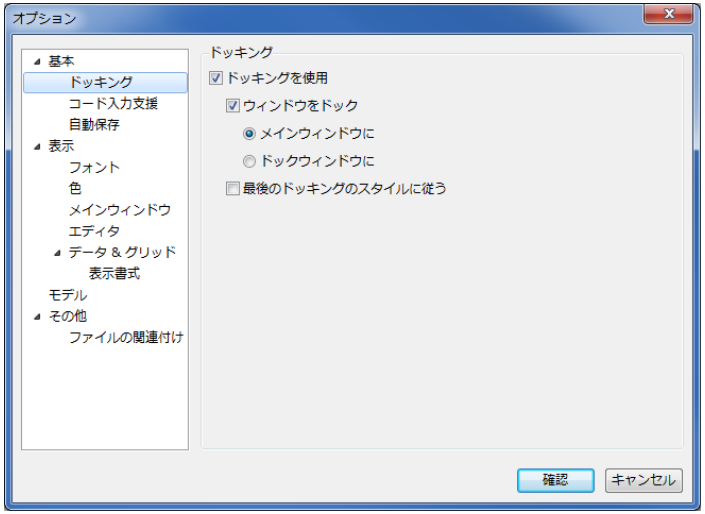
◇Navicat　初期設定

メインウィンドウ



①ツール-オプションで
オプションウィンドウを開く

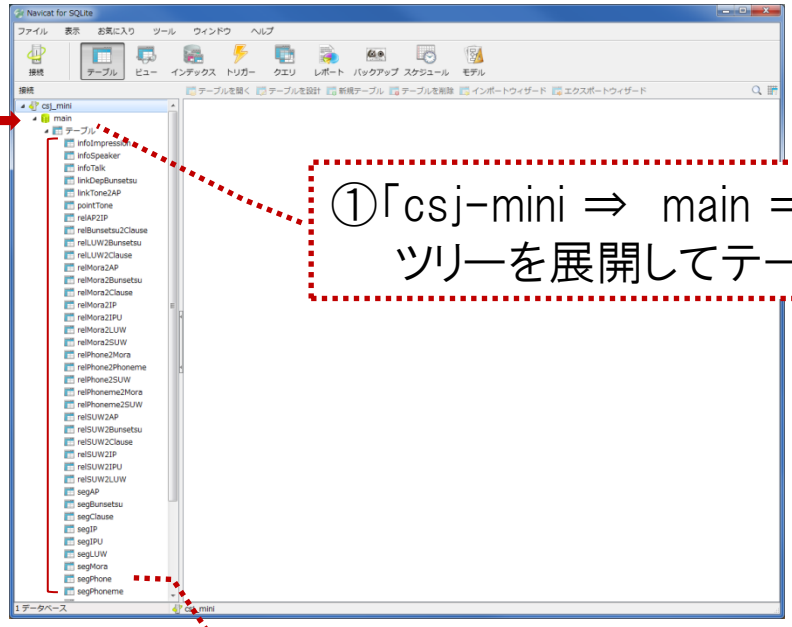
オプションウィンドウ



②「ドッキングを使用」「ウィンドウをドック」
「メインウィンドウに」をチェック

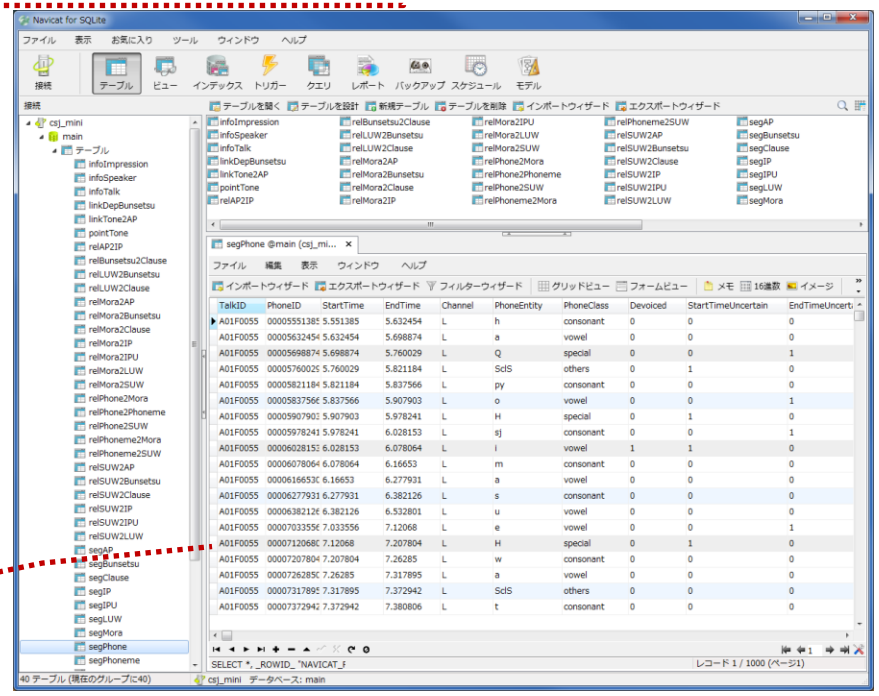
◇Navicat　テーブル操作

■　テーブルを開く



①「csj-mini ⇒　main ⇒ テーブル」の順に
　ツリーを展開してテーブルの一覧を表示

② テーブル名（例えばsegPhone ）
　をダブルクリック



③ テーブルが表示

第2部
SQLクエリの基本

RDBとは

- 相互に関係づけ可能な複数のテーブルの集合

table1　談話情報

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
D01F0023	dialog	514

table2　話者情報

SpeakerID	性別	出身地
459	男	東京都
514	男	神奈川県

table3　単語情報

TalkID	単語	品詞
A01F0055	発表	名詞
A01F0055	し	動詞
A01F0055	まし	助動詞
A01F0055	た	助動詞
D01F0023	報告	名
D01F0023	し	動
D01F0023	ます	助

三つのテーブルを共通キーで
関係づけて結合すると…

TalkID	単語	品詞	TalkType	SpeakerID	性別	出身地
A01F0055	発表	名詞	monolog	459	男	東京都
A01F0055	し	動詞	monolog	459	男	東京都
A01F0055	まし	助動詞	monolog	459	男	東京都
A01F0055	た	助動詞	monolog	459	男	東京都
D01F0023	報告	名詞	dialog	514	男	神奈川県
D01F0023	し	動詞	dialog	514	男	神奈川県
D01F0023	ます	助動詞	dialog	514	男	神奈川県

RDBの基本操作

SELECT文　列の選択

テーブルから指定した列を選択して新しいテーブルを作成

WHERE句　行の抽出

テーブルから条件に合致した行を抽出して新しいテーブルを作成

JOIN句　結合

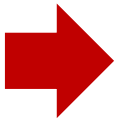
複数のテーブルを共通するキーにより関係づけて結合し，
新しい一つのテーブルを作成

基本操作① SELECT文～列の選択～

- テーブルから指定した**列**を選択

テーブル名：infoTalk

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471



新しいテーブル

TalkID	TalkType
A01F0055	monolog
A02M0098	monolog
D01F0023	dialog
D01M0009	dialog

テーブルから“TalkID”と“TalkType”の列を選択

基本操作② WHERE句～行の抽出～

- テーブルから条件に合致した行を抽出

テーブル名：infoTalk

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471

新しいテーブル

TalkID	TalkType	SpeakerID
D01F0023	dialog	514
D01M0009	dialog	471

テーブルから、列“TalkType”が“dialog”である、という条件に合致した行を抽出

基本操作③ JOIN句～結合～

- 複数のテーブルを共通するキーにより関係づけて結合し，新しい一つのテーブルを作成

共通するキー

infoTalk

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471

infoSpeaker

SpeakerID	SpeakerSex	SpeakerBirthPlace
459	男	東京都
130	女	東京都
514	男	神奈川県
471	女	神奈川県

新しいテーブル

TalkID	TalkType	SpeakerID	SpeakerSex	SpeakerBirthPlace
A01F0055	monolog	459	男	東京都
A02M0098	monolog	130	女	東京都
D01F0023	dialog	514	男	神奈川県
D01M0009	dialog	471	女	神奈川県

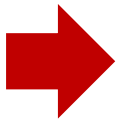
第2部　SQLクエリの基本
① SELECT文

SELECT文 ①　～列の選択～

- テーブルから指定した**列**を選択

テーブル名：infoTalk

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471



新しいテーブル

TalkID	TalkType
A01F0055	monolog
A02M0098	monolog
D01F0023	dialog
D01M0009	dialog

テーブルから“TalkID”と“TalkType”の列を選択

※SQLで書くと

```
SELECT  infoTalk.TalkID,  infoTalk.TalkType   FROM   infoTalk
```

選択する列名（テーブル名.列名の形式）　　テーブル名

SELECT文 ①　～列の選択～

- 基本操作

```
SELECT テーブル名.列名1, テーブル名.列名2, …
FROM テーブル名
```

- テーブルが一つの場合は列名の前のテーブル名は省略可

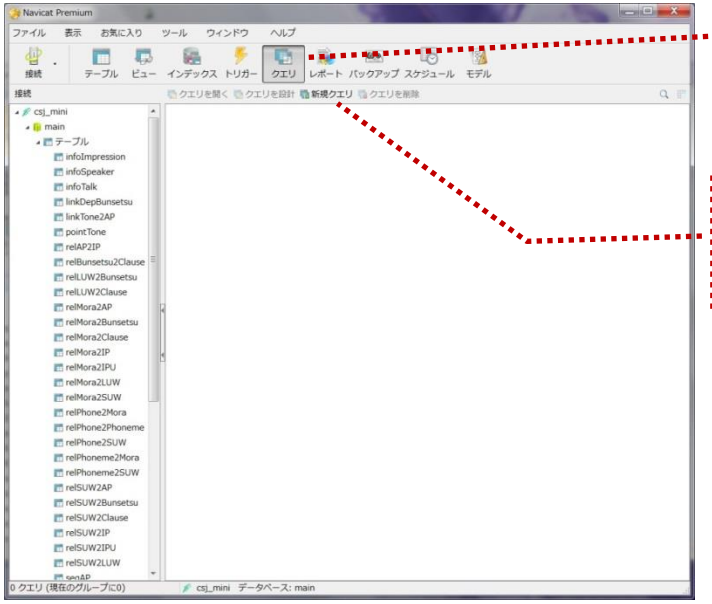
```
SELECT 列名1, 列名2, …
FROM テーブル名
```

- *(半角アスタリスク)ですべての列を選択

```
SELECT　*
FROM テーブル名
```

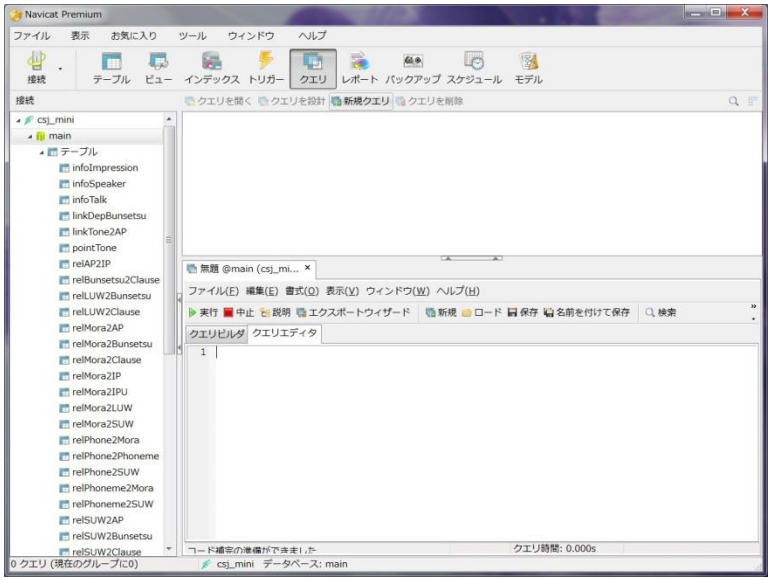
◇ Navicat　クエリ操作一般

■　クエリエディタを開く



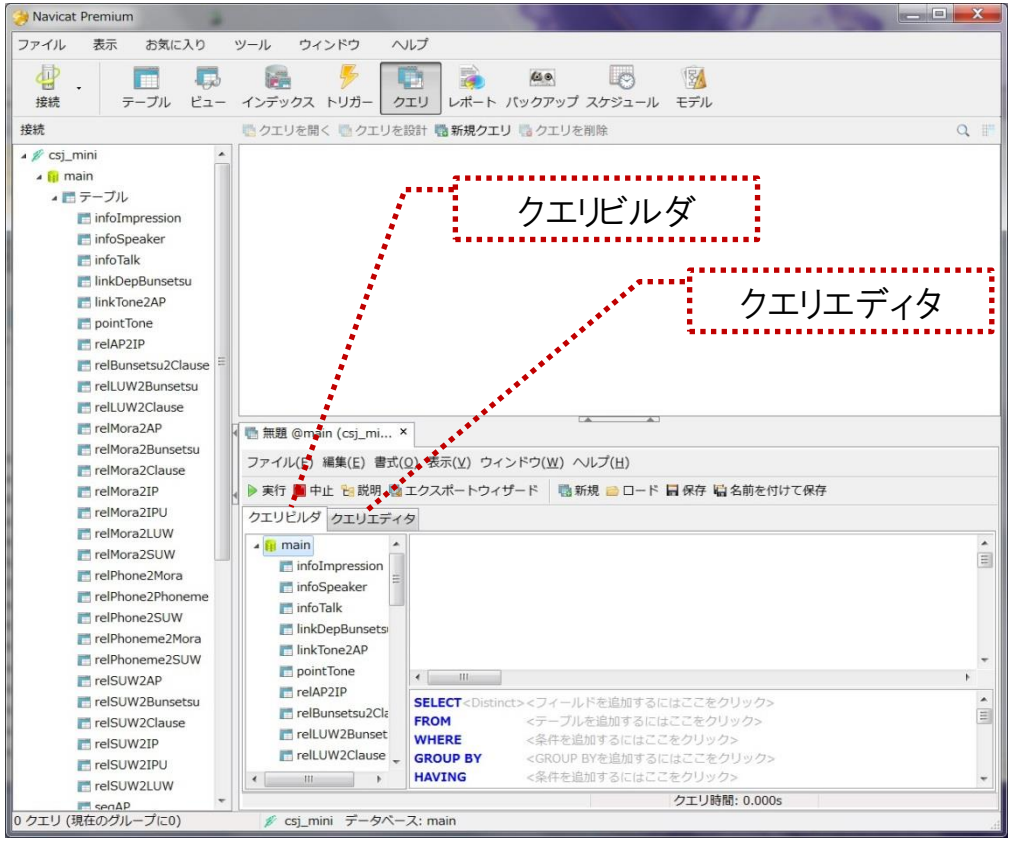
①[クエリ]アイコンをクリック

②[新規クエリ]を選択
⇒クエリタブ「無題@main･･･」が開く。



◇ Navicat　クエリ操作一般

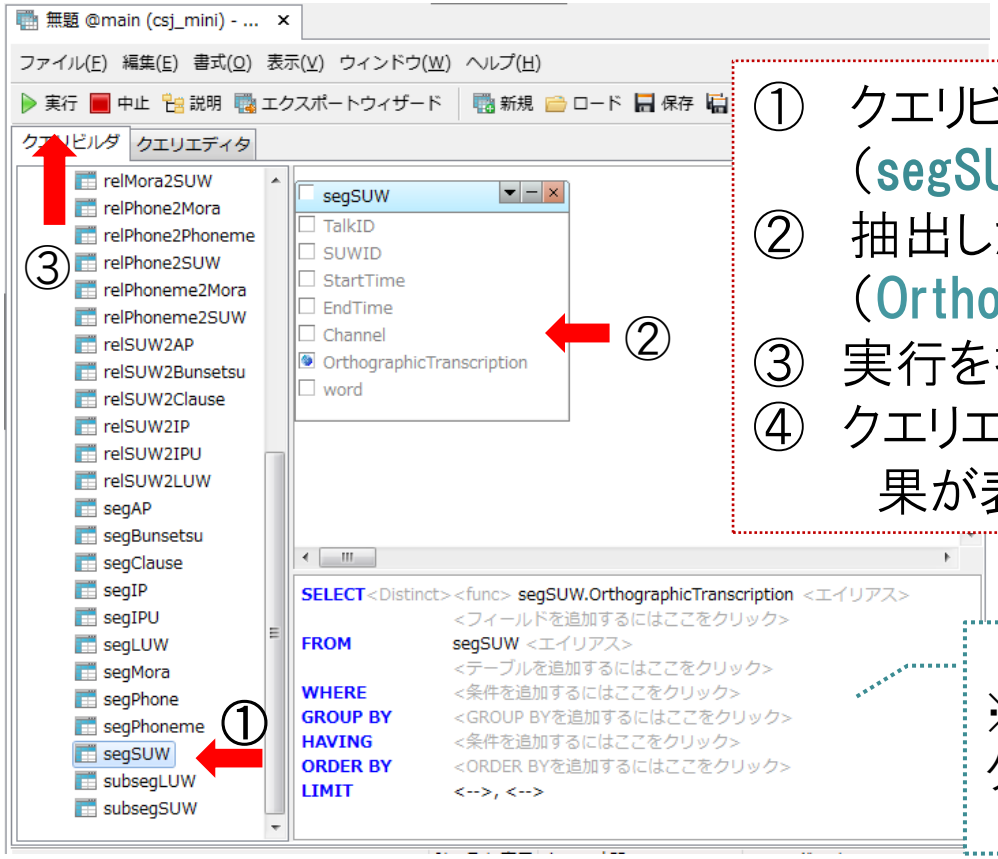
クエリビルダ　…クエリを視覚的に作成
クエリエディタ…SQL文を直接編集



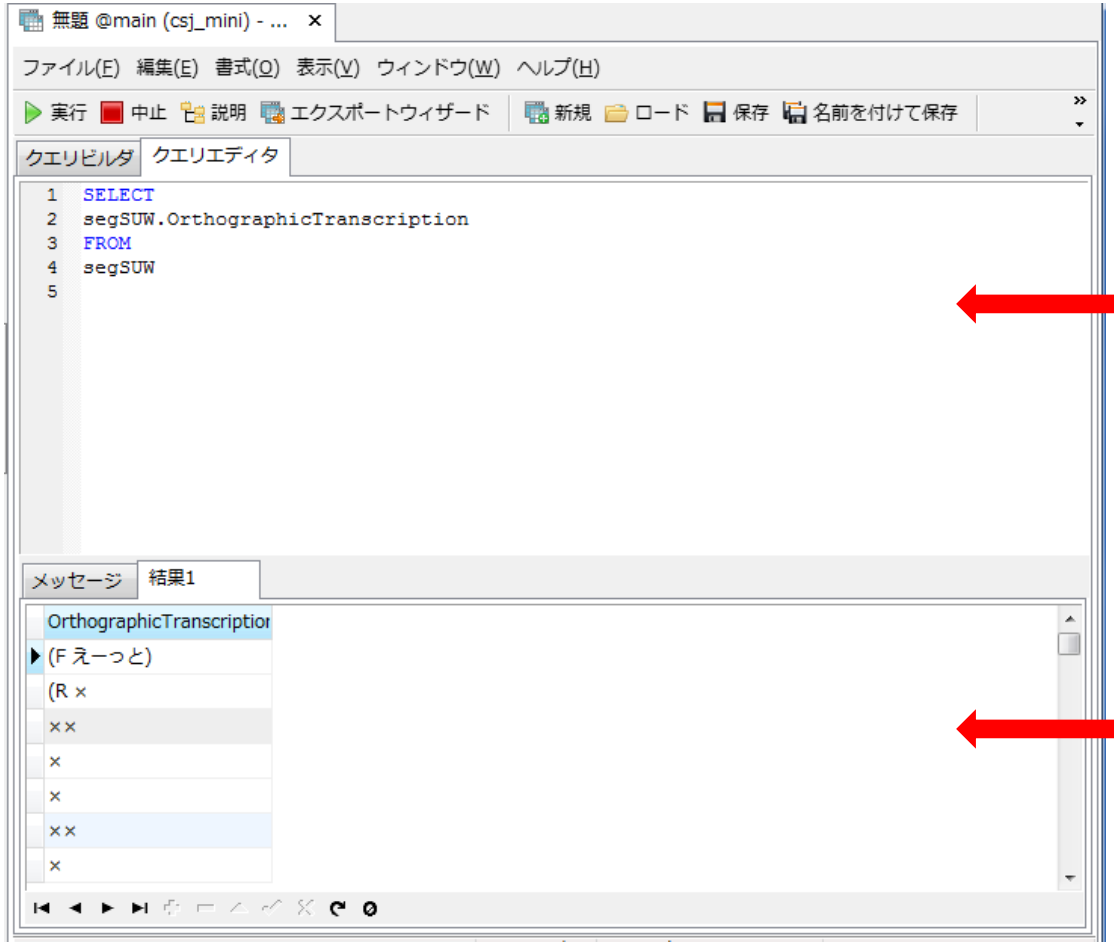
- ✓ クエリビルダで作成したSQL文はクエリエディタに即時反映される
- ✓ クエリビルダ･クエリエディタを適宜使いながら、SQL文を作成する

◇ Navicat　SELECT

例）　クエリビルダを使い、短単位テーブル（segSUW）から
　　基本形（OrthographicTranscription）の列を選択する



◇ Navicat　SELECT



④　実行されたSQL文
（直接ここでクエリ文を書いたり編集することも可能）

④ 実行結果

例題1～2

- **例題1**　テーブルから一つの列を選択

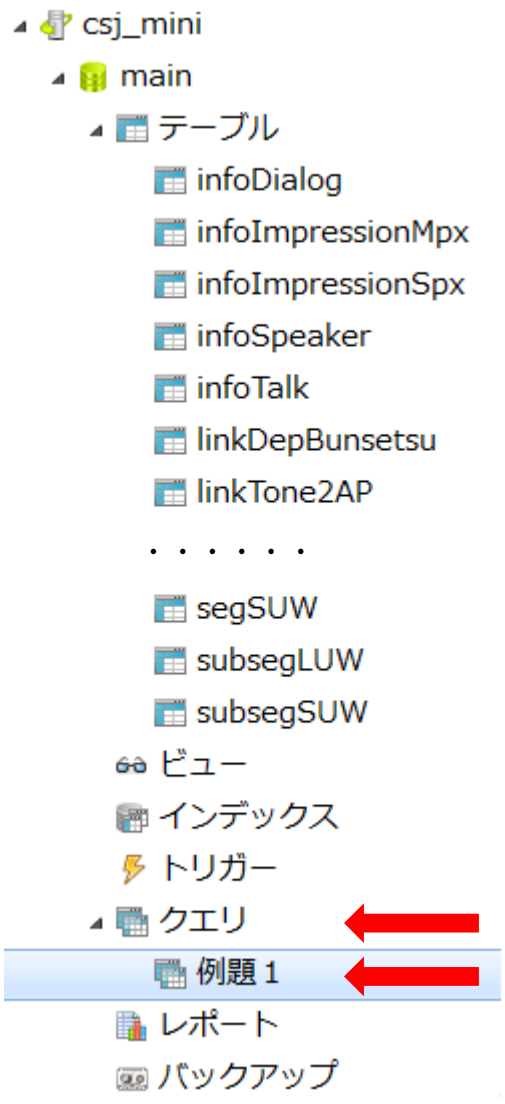
  短単位テーブル(segSUW)から開始時間(StartTime)の列を選択

- **例題2**　テーブルから複数の列を選択

  短単位テーブルから、開始時間、終了時間(EndTime)、
  基本形(OrthographicTranscription)の列を選択

◇Navicat　クエリの保存

- 作成したクエリは名前を付けて保存可能
  （ツールバーから「保存」を選択）
- メインウィンドウ左側の「クエリ」に保存される
- ファイルをダブルクリックすることで、保存したクエリ文を実行･編集することができる



SELECT文 ②　～列の演算～

- 選択する列同士を四則演算(+ － ＊ /)で計算することができる

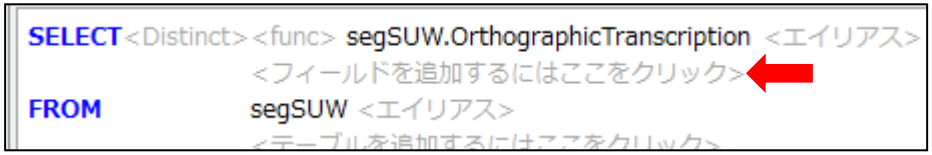
```
SELECT　列名1 - 列名2
FROM　テーブル名
```

```
SELECT　( 列名1 + 列名2 ) / 列名3
FROM　テーブル名
```

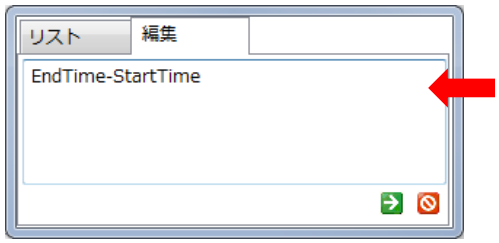
◇Navicat　ファンクション操作

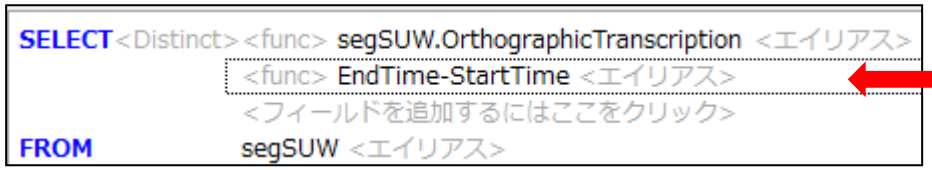
例）　短単位テーブルから開始時間と終了時間の差分（短単位の継続長）を計算し、基本形とあわせて出力

①計算したい場合、
〈フィールドを追加するにはここをクリック〉部分をクリック



②立ち上がったウィンドウの[編集]タブを選択し、計算式を入力する
ここでは　EndTime - StartTime　と入力し を押す





※記入した計算式が反映

SELECT文 ③ ～列に別名定義～

- ASを用いて列や計算結果に別名（エイリアス）をつけることができる

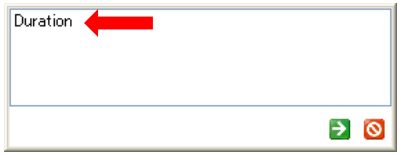
```
SELECT    列名1 - 列名2  AS  名前
FROM   テーブル名
```

◇Navicat　エイリアス操作

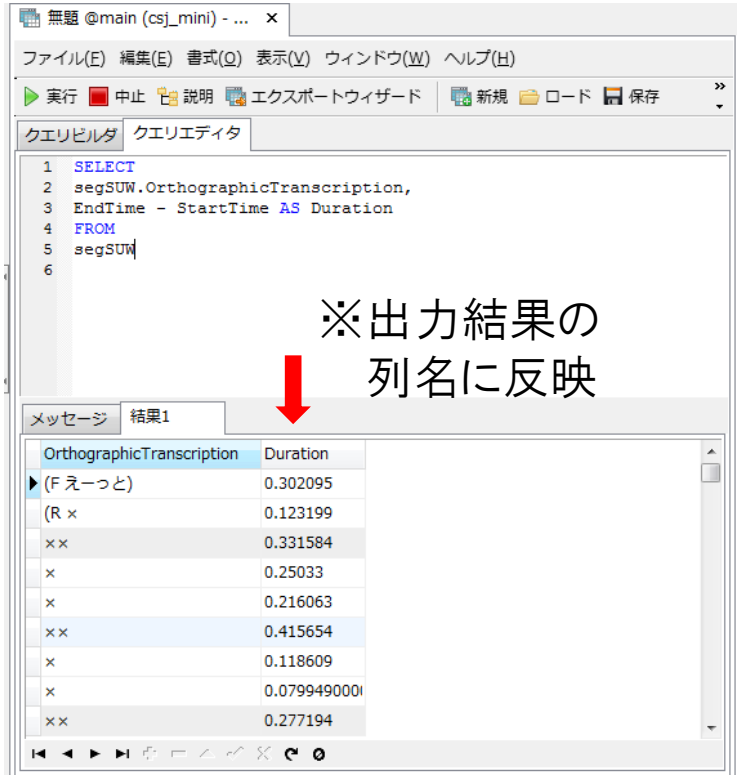
例）短単位テーブルから短単位の継続長を計算し、基本形とあわせて出力。計算した継続長に別名 “Duration” を付ける。

①別名を付けたい場合、
〈エイリアス〉部分をクリック

②立ち上がったウィンドウ内に別名を
入力し を押す



※記入した別名が「AS 別名」として反映



第2部 SQLクエリの基本
② WHERE句

WHERE 句 ①　～行の抽出～

- テーブルから条件に合致した行を抽出

テーブル名：infoTalk

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471

新しいテーブル

TalkID	TalkType	SpeakerID
D01F0023	dialog	514
D01M0009	dialog	471

テーブルから、列“TalkType”が”dialog”である、という条件に合致した行を抽出

※SQLで書くと

```
SELECT infoTalk.*  FROM  infoTalk  WHERE  infoTalk.TalkType = “dialog”
```

抽出する列名（「*」は全ての列）　テーブル名　抽出条件（列“TalkType”が“dialog”である行）

WHERE 句 ② ～比較演算子～

- **基本操作**

```
SELECT 列名1,列名2 …
FROM   テーブル名
WHERE  条件式
```

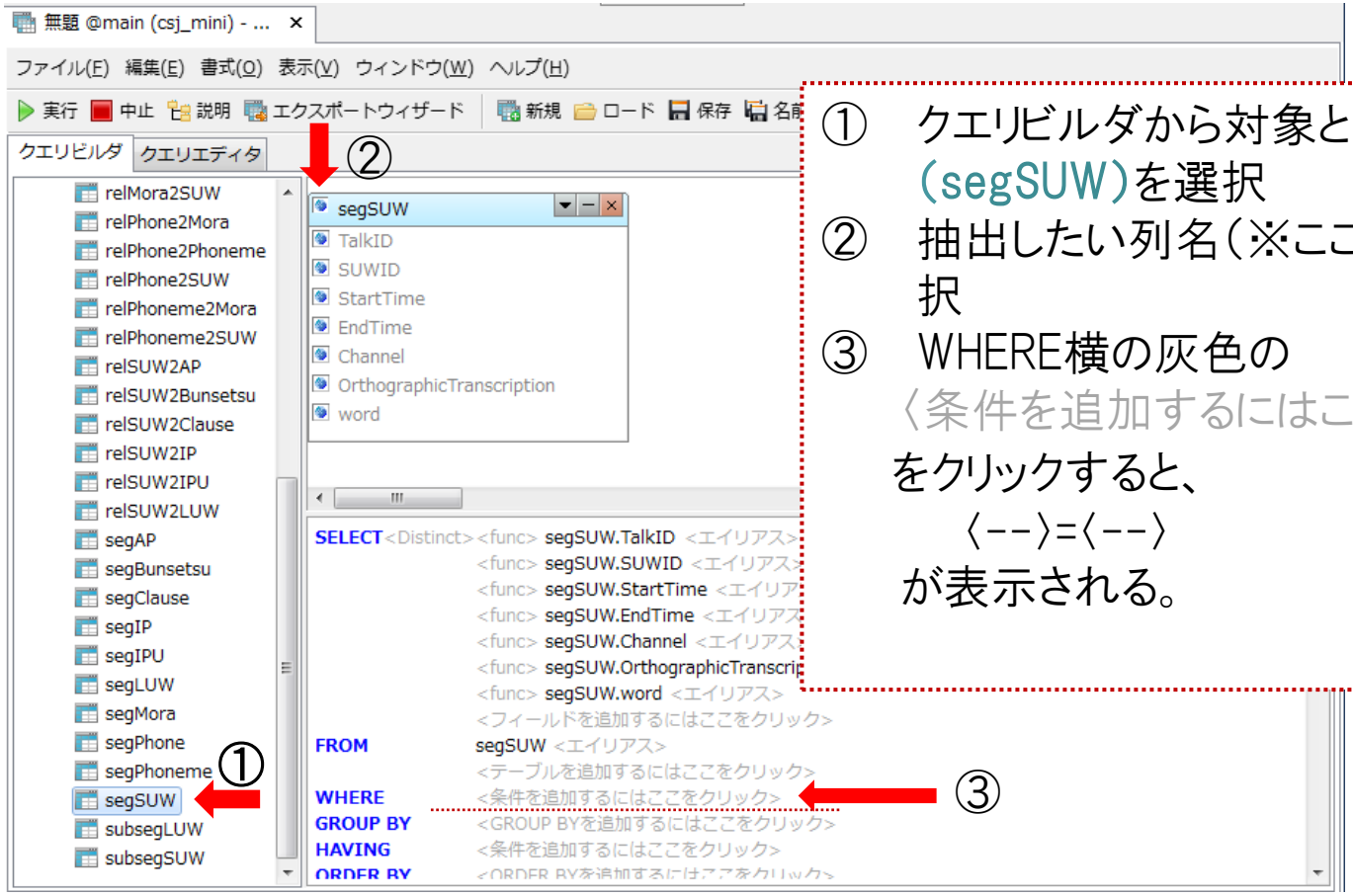
- **条件式で用いる演算子1**　比較演算子

演算子	使用例	説明	補足
=	a = b	a と b は等しい	「==」でも可
<>	a <> b	a と b は等しくない	「!=」でも可
>	a > b	a は b より大きい	
>=	a >= b	a は b 以上	
<	a < b	a は b より小さい	
<=	a <= b	a は b 以下	

例）
```
WHERE   StartTime  =  60          # 開始時間が60秒に一致
WHERE   TalkID  =  “D01F0023”     # 談話IDが“D01F0023”に一致
WHERE   StartTime  >=  400        # 開始時間が400秒以降
```

◇Navicat　WHERE

例）短単位テーブルから開始時間が400以上の行を抽出し、
全ての列を選択



① クエリビルダから対象とするテーブル（segSUW）を選択

② 抽出したい列名（※ここでは全ての行）を選択

③ WHERE横の灰色の
〈条件を追加するにはここをクリック〉
をクリックすると、
〈--〉=〈--〉
が表示される。

（次ページ続く）

◇Navicat　WHERE

- ✓ = をクリックすると演算子リスト(図1)が出現
- ✓ そこから適当な演算子を選択する
  （今回は「>=」を選択）

```
SELECT <Distinct> <func> segSUW.TalkID <エイリアス> ,
               <func> segSUW.SUWID <エイリアス> ,
               <func> segSUW.StartTime <エイリアス> ,
               <func> segSUW.EndTime <エイリアス> ,
               <func> segSUW.Channel <エイリアス> ,
               <func> segSUW.OrthographicTranscription <エイリアス> ,
               <func> segSUW.word <エイリアス>
               <フィールドを追加するにはここをクリック>
FROM           segSUW <エイリアス>
               <テーブルを追加するにはここをクリック>
WHERE          <--> = <-->
               <条件を追加するにはここをクリック>
```

- ✓ <--> をクリックすると、演算子の左右に来る要素の選択・入力を補助するボックス(図2・3)が出現。
- ✓ 列名を指定する場合は「リスト」から選択(図2)。
- ✓ 数字・文字を指定する場合は「編集」で記入(図3)。
- ✓ 左側は「リスト」から“segSUW.StartTime”を選択。
- ✓ 右側は「編集」で“400”と記入。

図1

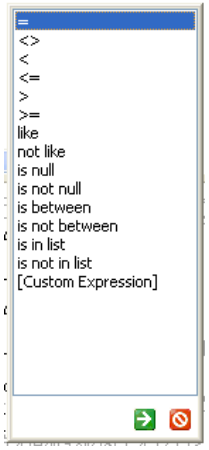


図2

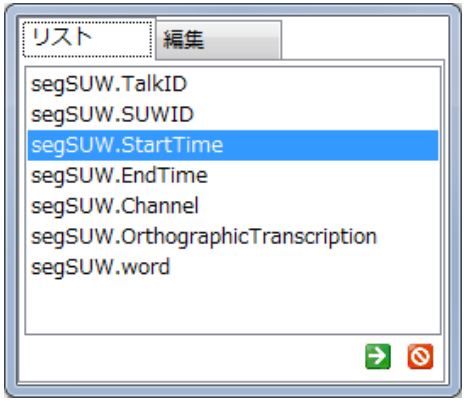
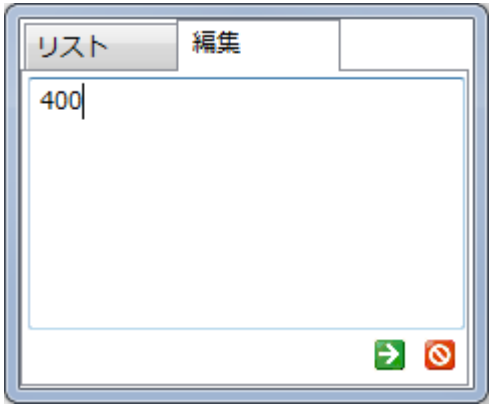
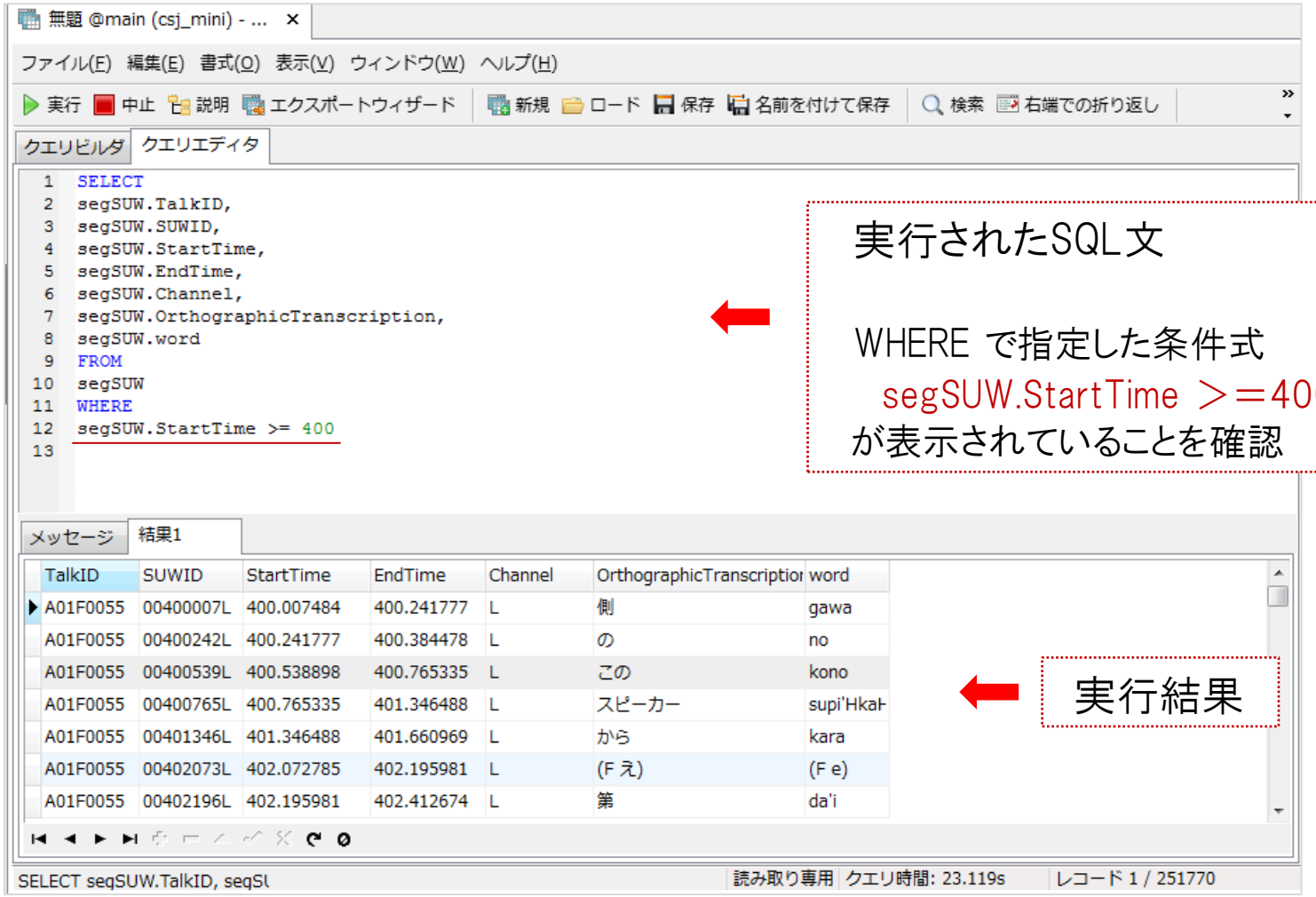


図3



◇Navicat　WHERE



例題3～5

- **例題3**　テーブルから条件に合う行を抽出（= 演算子、右辺が数値）

  モーラテーブル（segMora）からアクセント核の有無（PerceivedAcc）が 1 である行を抽出し、全ての列を選択

- **例題4**　テーブルから条件に合う行を抽出（= 演算子、右辺が文字列）

  モーラテーブルのモーラ記号（moraEntity）が「ア」である行を抽出し、全ての列を選択

- **例題5**　テーブルから条件に合う行を抽出（> 演算子）

  モーラテーブルから継続長が 0.1（秒）より大きい行を抽出し、全ての列を選択

ヒント：継続長はモーラの終了時間から開始時間を引くことで計算できます

WHERE 句 ③ ～パターンマッチング～

条件式で用いる演算子2　パターンマッチング

［形式］ 列名　LIKE　“パターン”　…列名がパターンにマッチする場合
列名　NOT LIKE　“パターン”　…列名がパターンにマッチしない場合

使用できるワイルドカード

ワイルドカード	意味
%	任意の0文字以上の文字列
_	任意の1文字

例）

WHERE OrthographicTranscription　LIKE　“あ%”　…「あ＋ 0文字以上の任意の文字列」の形式の基本形
（「あ」「あか」「あかい」など）

WHERE OrthographicTranscription　LIKE　“%い”　…「0文字以上の任意の文字列＋い」の形式の基本形
（「い」「はい」「美しい」「大きい」「よろしい」など）

WHERE OrthographicTranscription　LIKE　“あ_”　…「あ＋ 任意の1文字」の形式の基本形
（「ああ」「あか」「あさ」「あほ」など）

例題6～7

- **例題6**　テーブルから条件に合う行を抽出（LIKE 演算子）

  文節テーブル（segBunsetsu）から、基本形（OrthographicTranscription）が「は」で終了する行を抽出し、全ての列を選択

- **例題7**　テーブルから条件に合う行を抽出（LIKE 演算子）

  文節テーブルから、基本形がフィラーである行を抽出し、全ての列を選択

  ヒント：フィラーは「(F あのー)」や「(F えー)」のように、記号 ”(F)” で囲まれています

WHERE 句 ④ ～論理演算子～

- **条件式で用いる演算子3** 論理演算子

演算子	使用例	説明
AND	p AND q	p と q が共に真の場合
OR	p OR q	p か q の少なくとも一つが真の場合
NOT	NOT p	p が真ではない（偽である）場合

pとqは条件式

例）WHERE TalkID = ″D01M0009″ AND OrthographicTranscription LIKE ″%です”

WHERE MoraEntity = “ア” OR MoraEntity = “ウ″

WHERE MoraEntity = “ア” OR MoraEntity = “ウ” OR MoraEntity = “オ″

◇Navicat　WHERE

例）モーラテーブルから、モーラ記号が“ア”であり、かつ、
継続長が 0.1 秒以上の行を抽出し、全ての列を選択

例題8～9

- **例題8**　テーブルから条件に合う行を抽出（OR 演算子）

節単位テーブル（segClause）から、
節境界ラベル（ClauseBoundaryLabel）が “/並列節シ/” である、**あるいは**、
節境界ラベルが “/並列節デ/” である行を抽出し、全ての列を選択

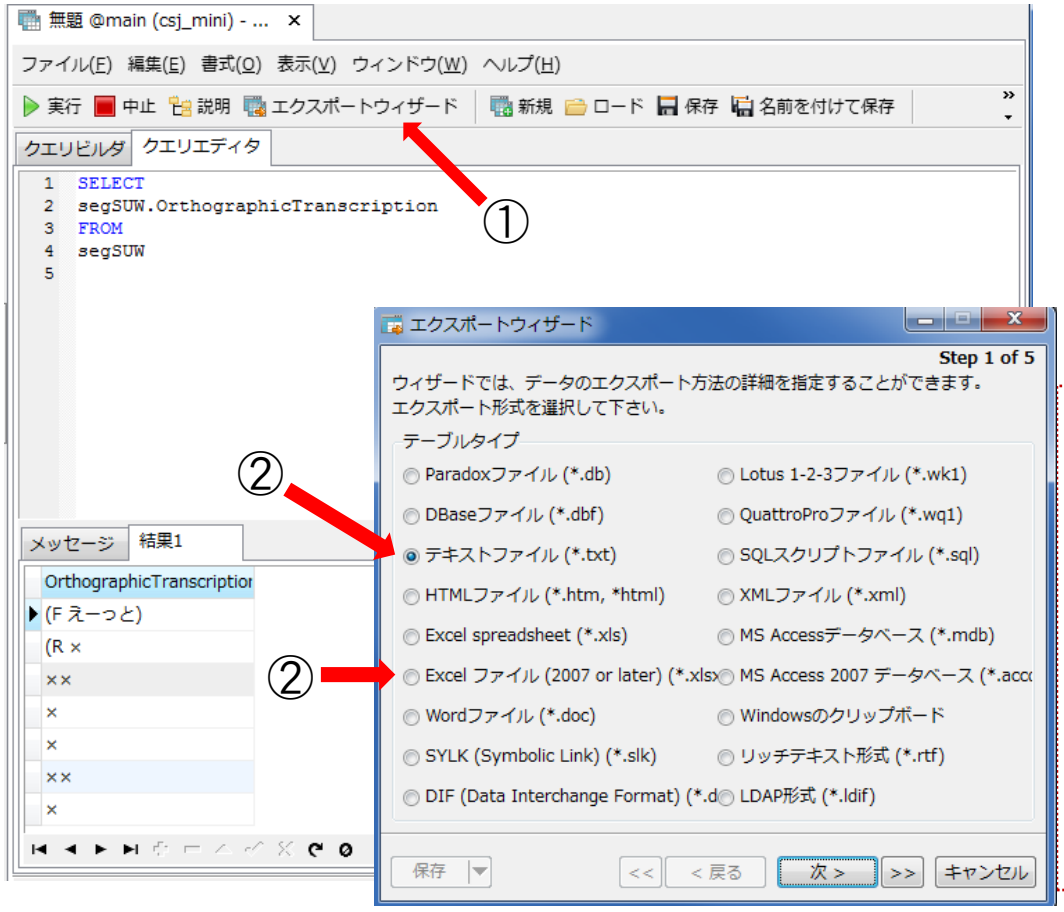
- **例題9**　テーブルから条件に合う行を抽出（AND 演算子）

節単位テーブルから、
節境界ラベルが強境界の並列節に該当する行で、**かつ**、
節境界ラベルが “/並列節シ/” でも “/並列節デ/” でもない行を抽出し、全ての列を選択

ヒント：強境界は “/並列節ケド/” や “/並列節ガ/” のように両端が “/” で囲まれています

◇Navicat　出力結果の保存

- エクスポートウィザード



① ツールバーのエクスポートウィザードを押すと、エクスポートウィザードが新しいウィンドウで開く

② 形式を選択する
（この例ではテキストファイルを選択）

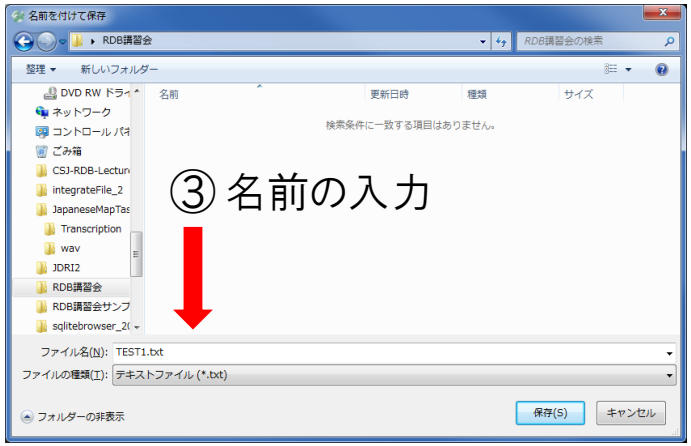
※ Excel ファイルとして保存する場合「Excel ファイル(2007 or later)」を選択

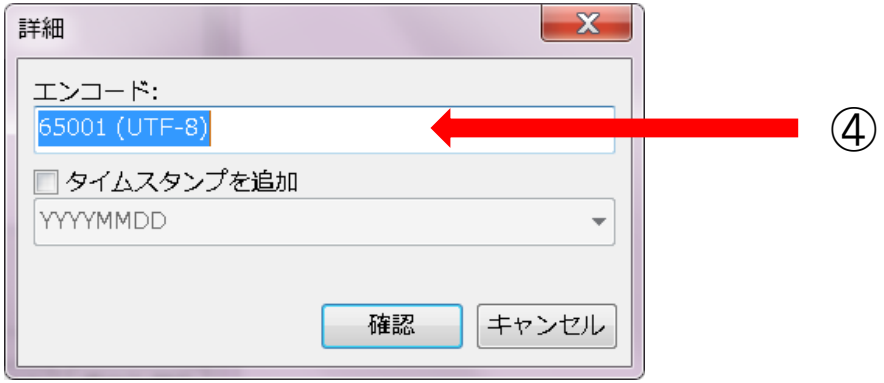
◇Navicat　出力結果の保存

③ … を押し、別ウィンドウでエクスポート先を指定し、名前を付ける(ここでは　TEST1.txt)

④「詳細」を押し、エンコードを適宜選択し「確認」を押す
(Windows であれば “Current Windows Codepage”,
Mac であれば初期値の “65001 (UTF-8)”を選択)

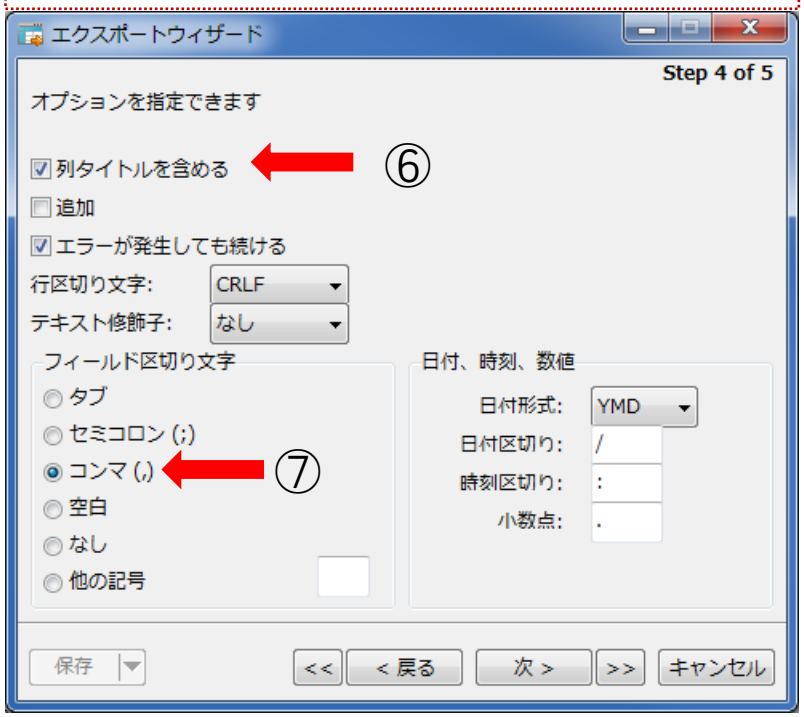
⑤「次」を押す



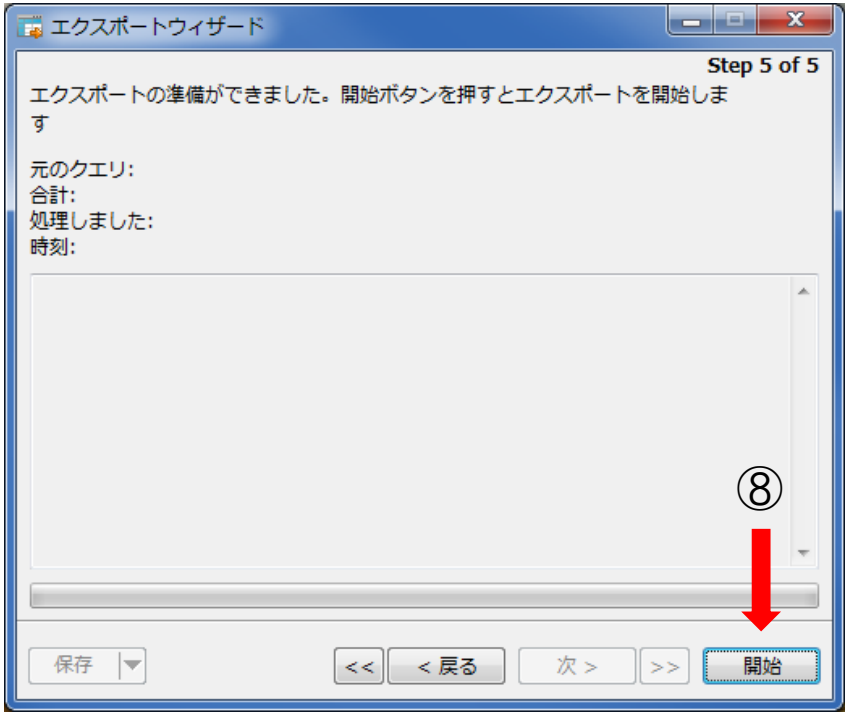


◇Navicat　出力結果の保存

⑥「列タイトルを含める」を選択
⑦ 区切り文字などのオプションを選択
（ Excel ファイルとして保存する場合は
　区切り文字の指定はなし）



⑧ 開始ボタンを押すとエクスポート開始



第2部 SQLクエリの基本
③ JOIN句

JOIN 句

- 内部結合(INNER JOIN句)

```
SELECT   table1.列A, table2.列B
FROM     table1
INNER JOIN  table2  ON …
```

- 外部結合(初級編では対象外)

INNER JOIN句

- 対応する値がある行のみ結合

■ infoTalk

対応させるキー

TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471
R00M0187	monolog	423

対応

■ infoSpeaker

SpeakerID	SpeakerSex	SpeakerBirthPlace
459	男	東京都
130	女	東京都
514	男	神奈川県
471	女	神奈川県
131	男	群馬県

■ 新しいテーブル

TalkID	TalkType	SpeakerID	SpeakerSex	SpeakerBirthPlace
A01F0055	monolog	459	男	東京都
A02M0098	monolog	130	女	東京都
D01F0023	dialog	514	男	神奈川県
D01M0009	dialog	471	女	神奈川県

INNER JOIN句

```
SELECT  infoTalk.*,  infoSpeaker.SpeakerSex,  infoSpeaker.SpeakerBirthPlace
FROM  infoTalk
INNER JOIN  infoSpeaker  ON  InfoTalk.SpeakerID  =  infoSpeaker.SpeakerID
```

結合元のテーブル名

結合先のテーブル名

対応させるキー

結合元

■ infoTalk

対応させるキー

結合先

■ infoSpeaker

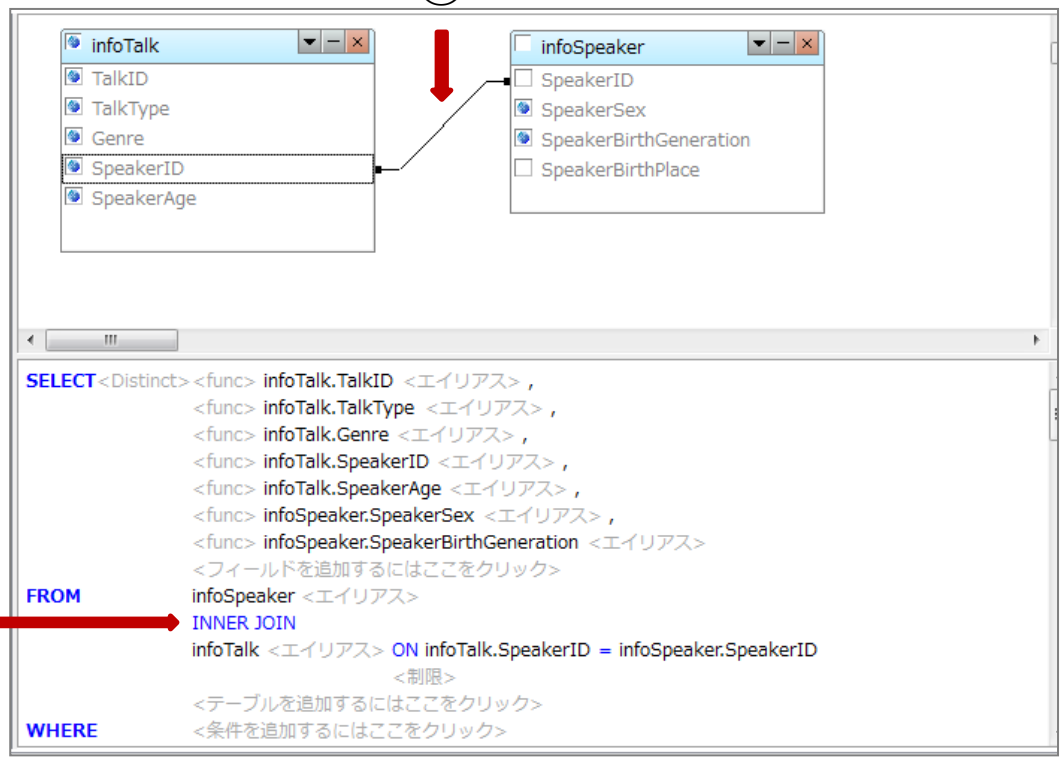
TalkID	TalkType	SpeakerID
A01F0055	monolog	459
A02M0098	monolog	130
D01F0023	dialog	514
D01M0009	dialog	471
R00M0187	monolog	423

SpeakerID	SpeakerSex	SpeakerBirthPlace
459	男	東京都
130	女	東京都
514	男	神奈川県
471	女	神奈川県
131	男	群馬県

◇Navicat　JOIN

例）談話基本情報テーブル(infoTalk)と話者基本情報テーブル(infoSpeaker)を、
SpeakerID をキーに内部結合し、infoTalkの全ての列と、infoSpeaker の
話者性別(SpeakerSex)と話者生年代(SpeakerBirthGeneration)の列を選択

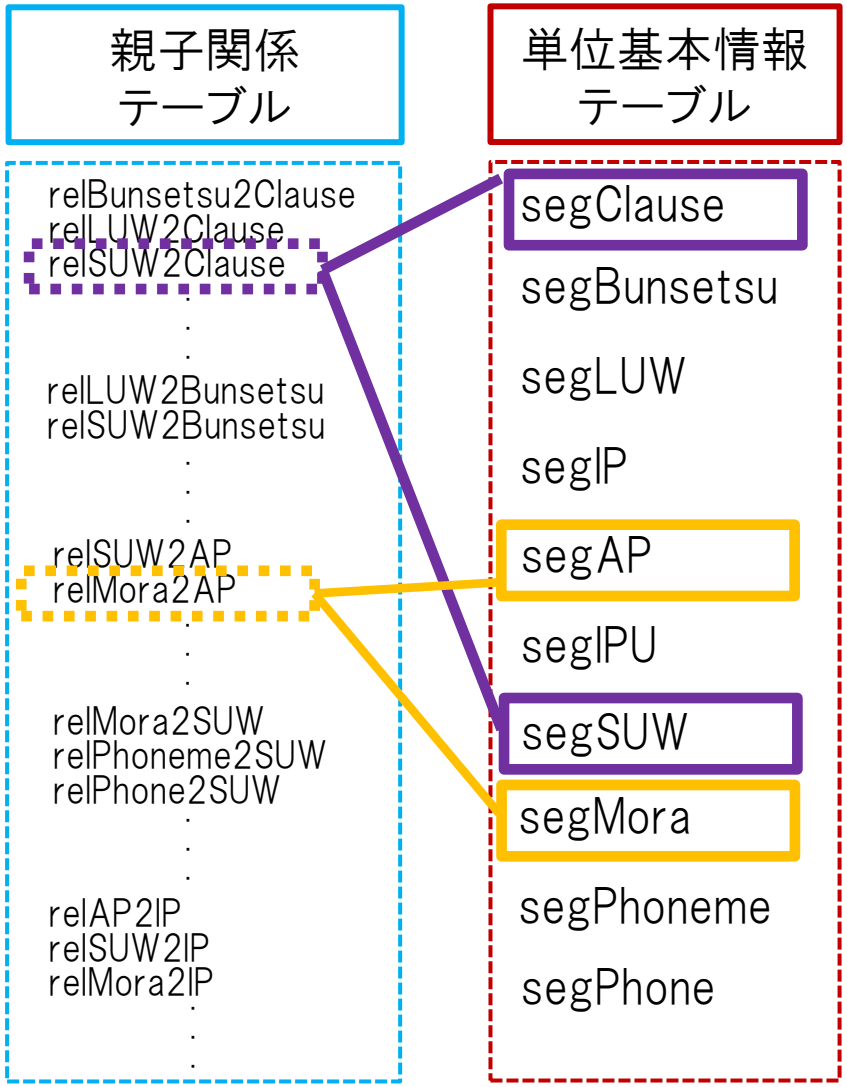
① クエリビルダで結合させるテーブルを選択し表示
② テーブル間で対応させるキーを線で結ぶ
③ 適宜JOINの種類を変更

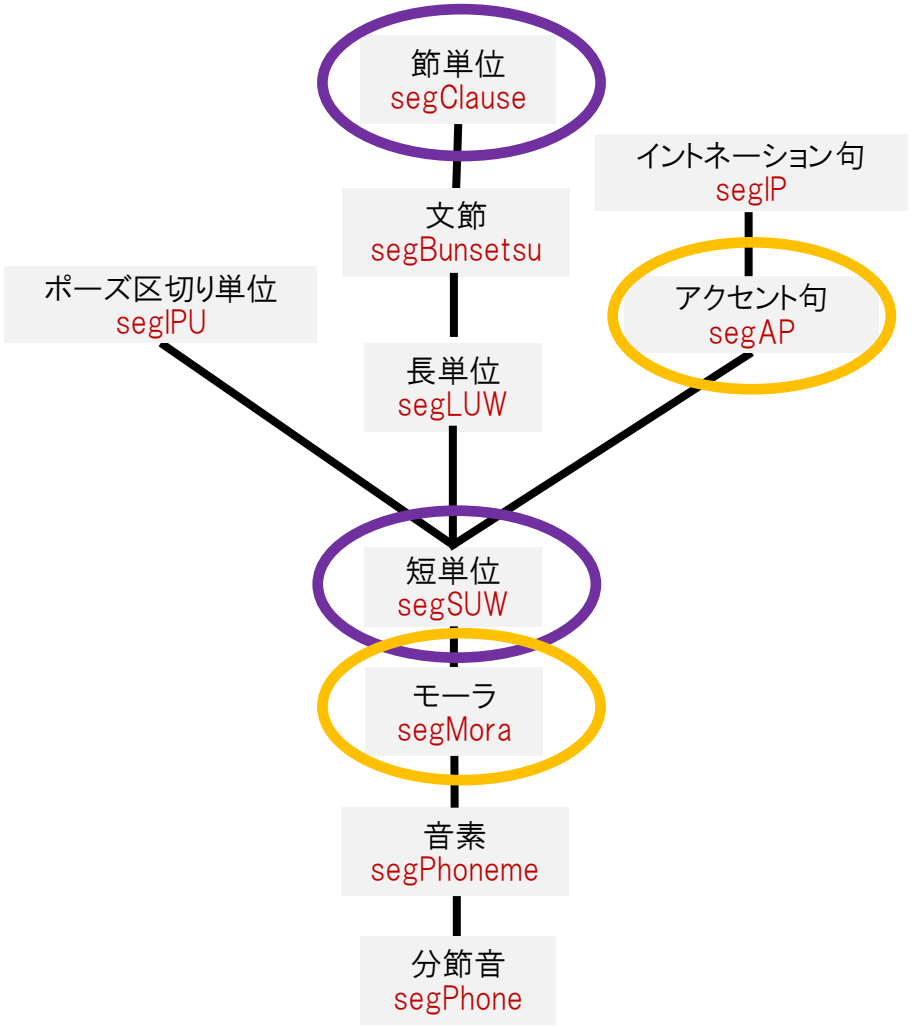


初期値INNER JOIN
この部分をクリックすることで
JOIN の種類を変更できる

③

CSJ-RDBの構成 ―親子関係テーブル―





親子関係テーブルの具体例

■ segLUW （子供のテーブル）

TalkID	LUWID	OrthographicTranscription
A01F0055	00053773L	いつ頃
A01F0055	00054186L	から
A01F0055	00054505L	可能
A01F0055	00054862L	な
A01F0055	00054980L	のでしょ
A01F0055	00055415L	う
A01F0055	00055478L	か

子供の単位ID

■ relLUW2Bunsetsu （親子関係テーブル）

親単位中の子単位の数と位置（何番目か）

TalkID	LUWID	BunsetsuID	nth	len
A01F0055	00053773L	00053773L	1	2
A01F0055	00054186L	00053773L	2	2
A01F0055	00054505L	00054505L	1	5
A01F0055	00054862L	00054505L	2	5
A01F0055	00054980L	00054505L	3	5
A01F0055	00055415L	00054505L	4	5
A01F0055	00055478L	00054505L	5	5

親の単位ID

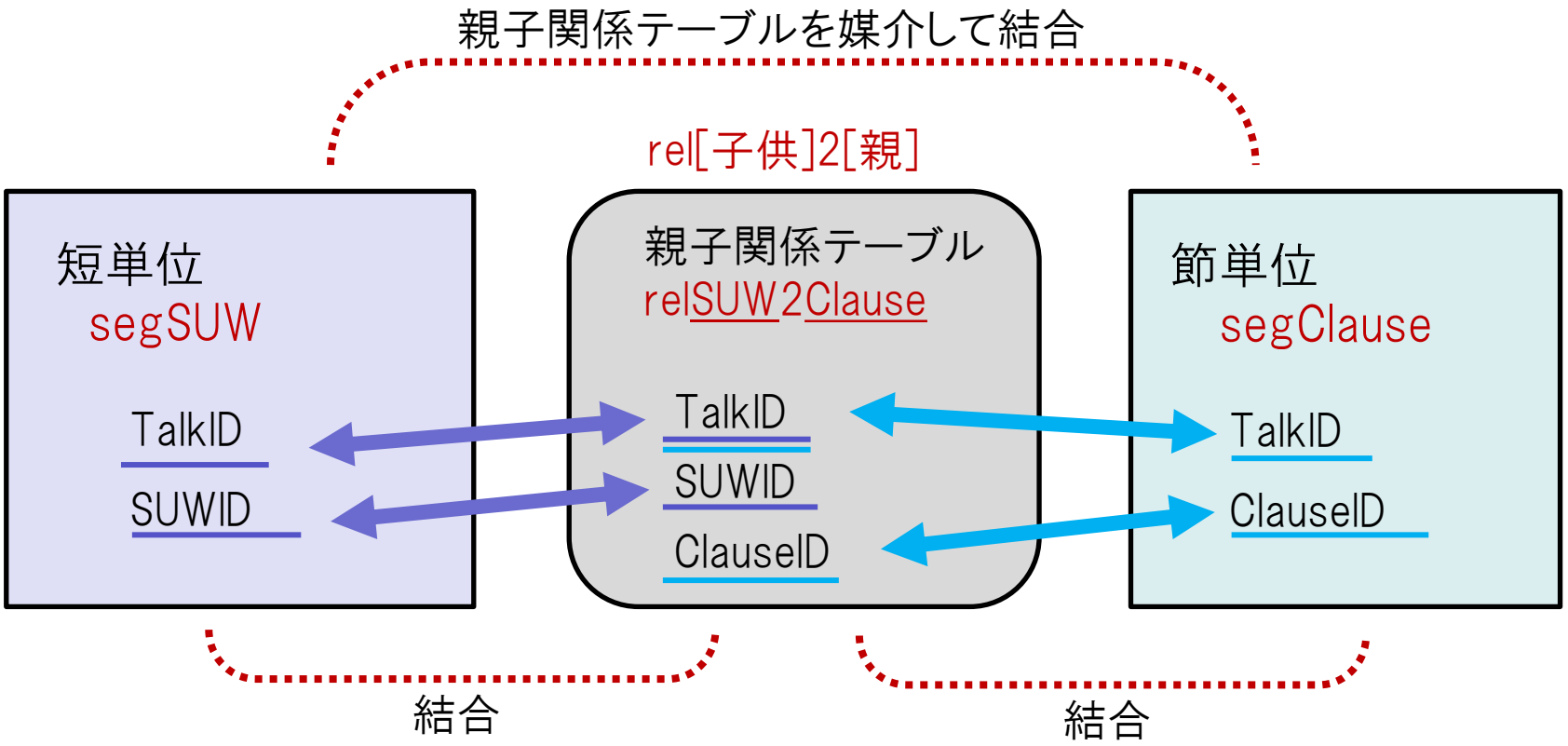
■ segBunsetsu （親のテーブル）

TalkID	BunsetsuID	OrthographicTranscription
A01F0055	00053773L	いつ頃から
A01F0055	00054505L	可能なのでしょうか

■ 内部結合後

TalkID	nth	len	LUWID	OrthographicTranscription	BunsetsuID	OrthographicTranscription
A01F0055	1	2	00053773L	いつ頃	00053773L	いつ頃から
A01F0055	2	2	00054186L	から	00053773L	いつ頃から
A01F0055	1	5	00054505L	可能	00054505L	可能なのでしょうか
A01F0055	2	5	00054862L	な	00054505L	可能なのでしょうか
A01F0055	3	5	00054980L	のでしょ	00054505L	可能なのでしょうか
A01F0055	4	5	00055415L	う	00054505L	可能なのでしょうか
A01F0055	5	5	00055478L	か	00054505L	可能なのでしょうか

短単位テーブルと節単位テーブルの結合

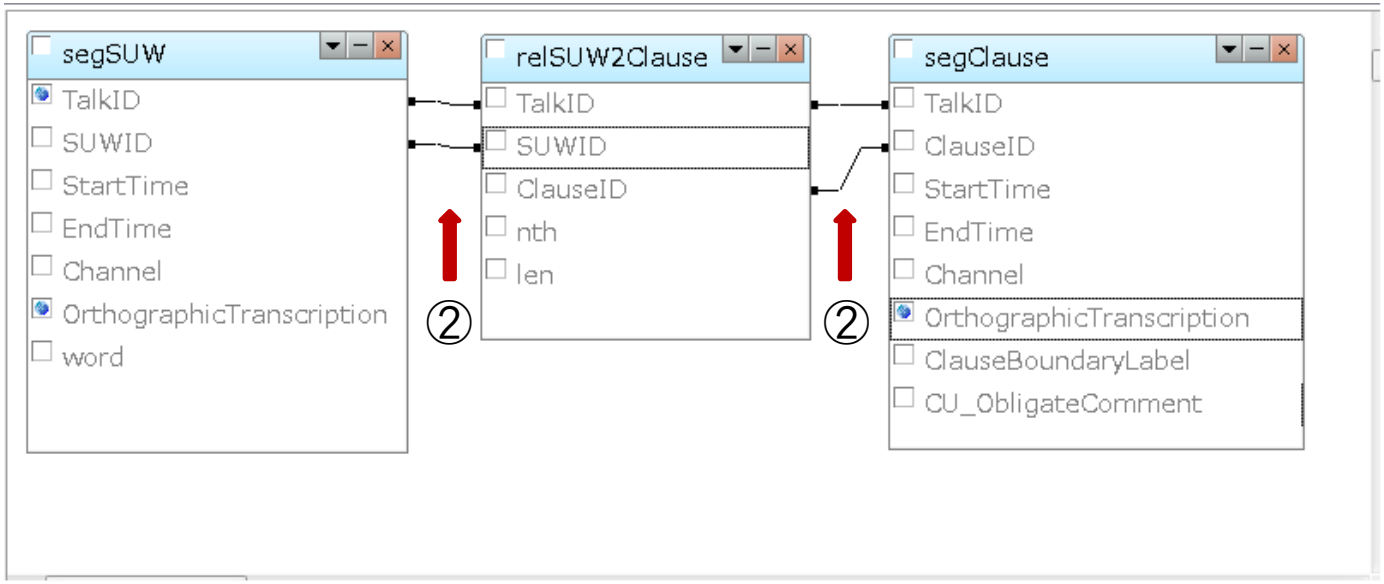


※ Navicat で segSUW と segClause を結合してみよう

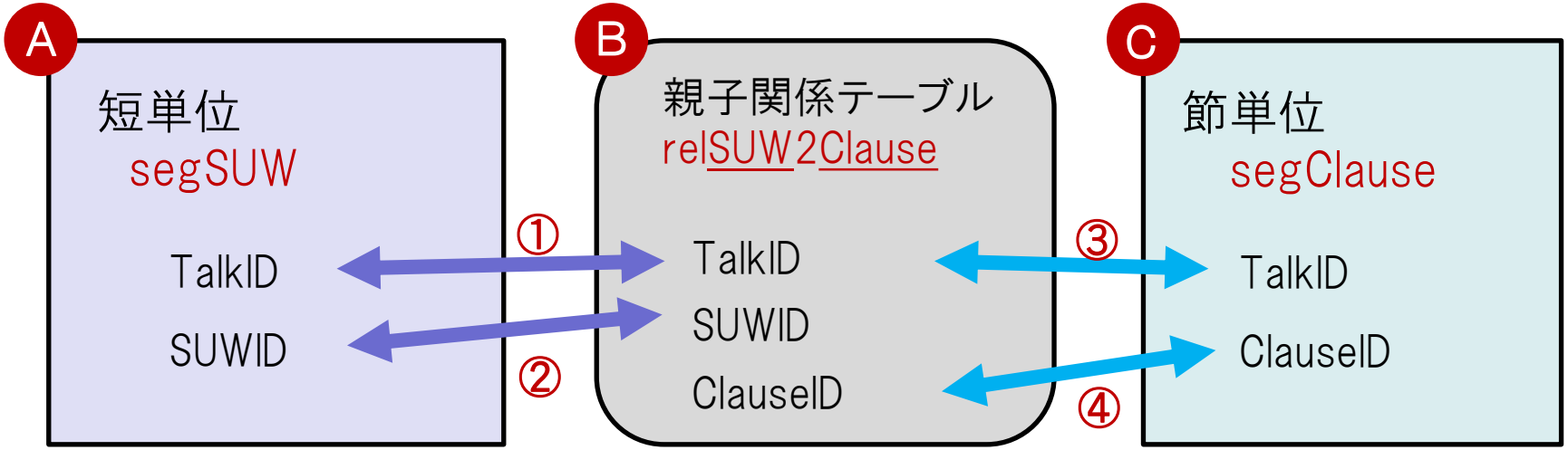
◇Navicat　JOIN

例）短単位テーブルと節単位テーブルを、両者の親子関係テーブル（relSUW2Clause）を利用して結合し、短単位テーブルの談話IDと基本形、及び、節単位テーブルの基本形を選択

① クエリビルダで結合させるテーブル（三つ）を選択
② テーブル間で対応させるキーを線で結ぶ



SQL文　～短単位と節単位の結合の場合～




```
SELECT  segSUW.*,  segClause.*
FROM  segSUW  A
INNER JOIN  relSUW2Clause  B
  ON  segSUW.TalkID = relSUW2Clause.TalkID  AND        ①
      segSUW.SUWID = relSUW2Clause.SUWID               ②
INNER JOIN  segClause  C
  ON  relSUW2Clause.TalkID  =  segClause.TalkID  AND   ③
      relSUW2Clause.ClauseID  =  segClause.ClauseID    ④
```

例題10～11

- **例題10**　二つのセグメント・テーブルの結合

文節テーブルと節単位テーブルを結合し、TalkID、文節の基本形、節単位の基本形の列を選択

ヒント：結合のために、両者の親子関係テーブル relBunsetsu2Clause　を利用

- **例題11**　二つのセグメント・テーブルの結合

文節テーブルと節単位テーブルを結合し、基本形が「は」で終了する文節のみ抽出し、TalkID、文節の基本形、節単位の基本形の列を選択

ヒント：例題6を参照

例題12

- **例題12**　二つのセグメント･テーブルの結合 + 親子関係テーブルを用いた条件

節単位の先頭の文節を抽出し、TalkID、文節の基本形、節単位の基本形を選択

ヒント：節単位の先頭の文節を指定するにはどのような条件を付ければよいか、
以下のテーブルを参考に考えてみましょう。

clauseID=00176804L中の
文節の位置（何番目か）

clauseID=00176804Lに
含まれる文節の数

TalkID	BunsetsuID	ClauseID	nth	len
A01F0055	00176804L	00176804L	1	6
A01F0055	00176882L	00176804L	2	6
A01F0055	00177374L	00176804L	3	6
A01F0055	00177569L	00176804L	4	6
A01F0055	00178510L	00176804L	5	6
A01F0055	00179521L	00176804L	6	6

例題13

- **例題13**　二つのセグメント･テーブルの結合 + 親子関係テーブルを用いた条件

節単位の末尾の文節を抽出し、TalkID、文節の基本形、節単位の基本形を選択

ヒント：節単位の末尾の文節を指定するにはどのような条件を付ければよいか、
以下のテーブルを参考に考えてみましょう。nth と len の両方を使います。

clauseID=00176804L中の
文節の位置（何番目か）

clauseID=00176804Lに
含まれる文節の数

TalkID	BunsetsuID	ClauseID	nth	len
A01F0055	00176804L	00176804L	1	6
A01F0055	00176882L	00176804L	2	6
A01F0055	00177374L	00176804L	3	6
A01F0055	00177569L	00176804L	4	6
A01F0055	00178510L	00176804L	5	6
A01F0055	00179521L	00176804L	6	6

例題14

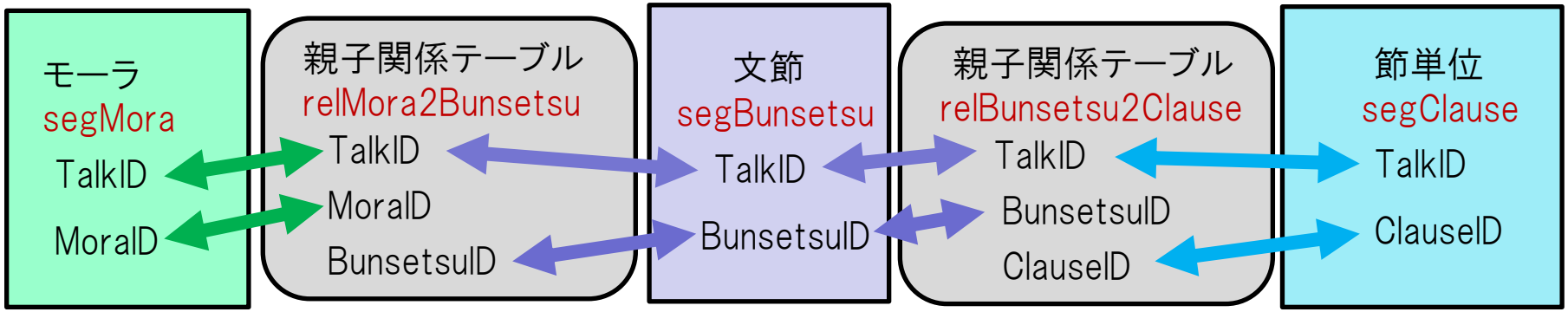
- **例題14**　二つのセグメント･テーブルの結合 + 親子関係テーブルを用いた条件

節単位の先頭の文節であり、かつ、基本形が「は」で終了する文節を抽出し、
各文節の継続長を計算し、“Duration” という別名を付けた上で、
TalkID、文節の基本形と合わせて表示。

例題15～16

- **例題15**　三つのセグメント･テーブルの結合

「モーラテーブル」「文節テーブル」「節単位テーブル」を結合し、TalkID、モーラ記号、文節の基本形、節単位の基本形の列を選択

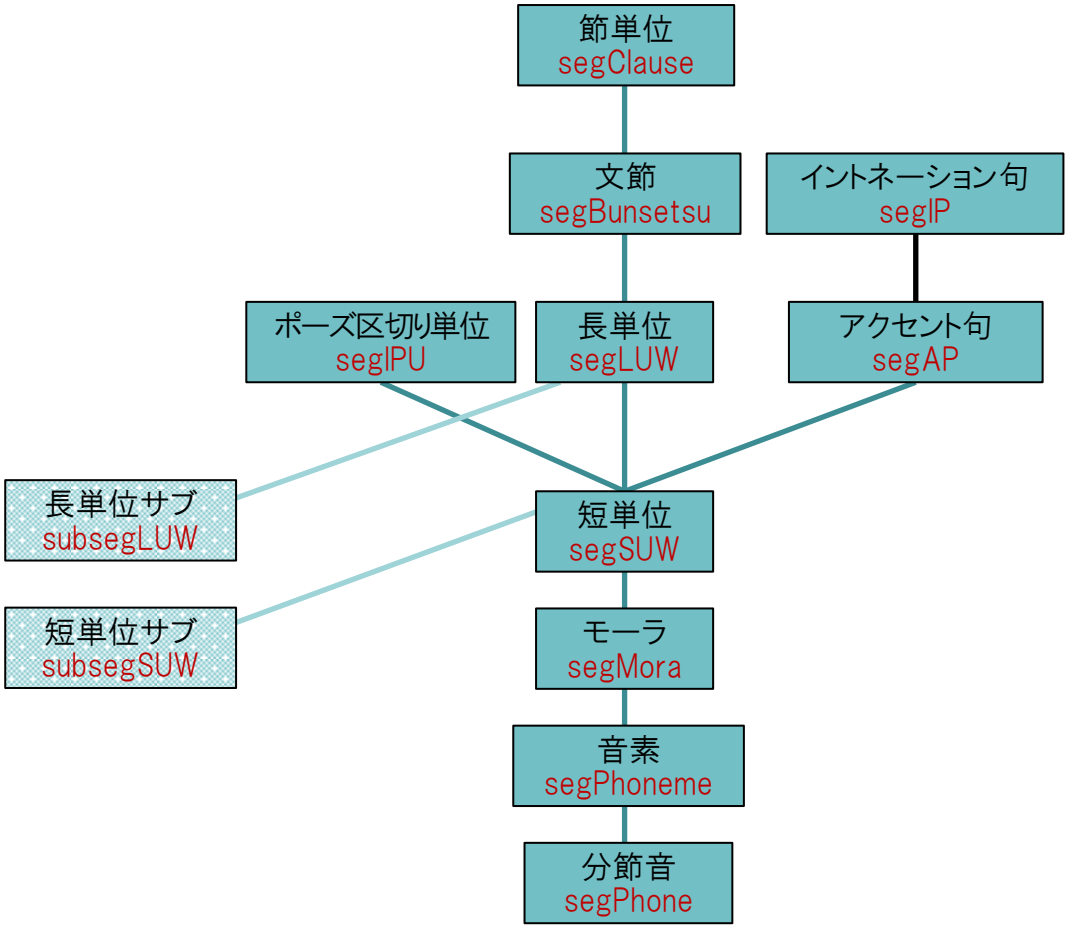


- **例題16**　三つのセグメント･テーブルの結合

節単位の先頭の文節で、かつ、基本形が「は」で終了する文節の末尾のモーラ（つまり「ワ」）を抽出し、各モーラの継続長を計算した上で、“Duration”という別名を付け、TalkID、文節の基本形、節単位の基本形と合わせて表示。

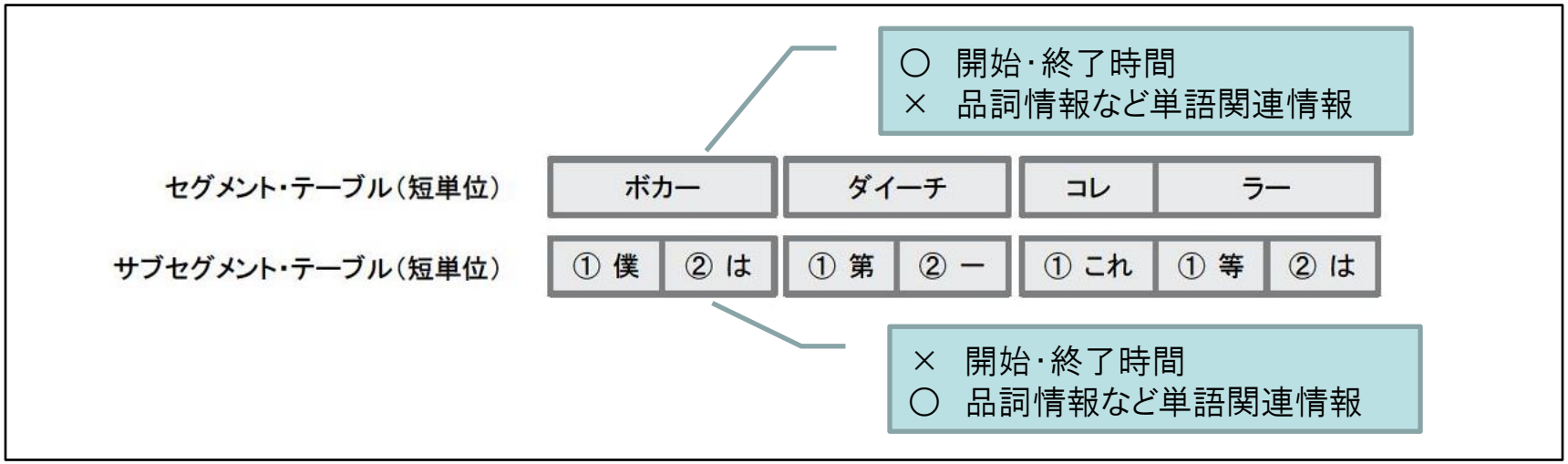
第3部
CSJ-RDBの構成 応用編

CSJ-RDBの構成 ーサブセグメント・テーブルー



サブセグメント・テーブル（短単位・長単位）

複数の語が融合して、分割できない一つの要素を形成する場合



■ セグメント・テーブル segSUW

TalkID	SUWID	StartTime	EndTime	OrthographicTranscription	word
A01F0122	00513758L	513.757614	514.114756	第ー	(W daici)
A01F0122	00514115L	514.114756	514.483693	母音	bo'iN

■ サブセグメント・テーブル subsegSUW （一部抜粋）

TalkID	SUWID	nth	len	PlainOrthographicTranscription	SUWLemma	SUWPOS
A01F0122	00513758L	1	2	第	第	接頭辞
A01F0122	00513758L	2	2	ー	ー	名詞
A01F0122	00514115L	1	1	母音	母音	名詞

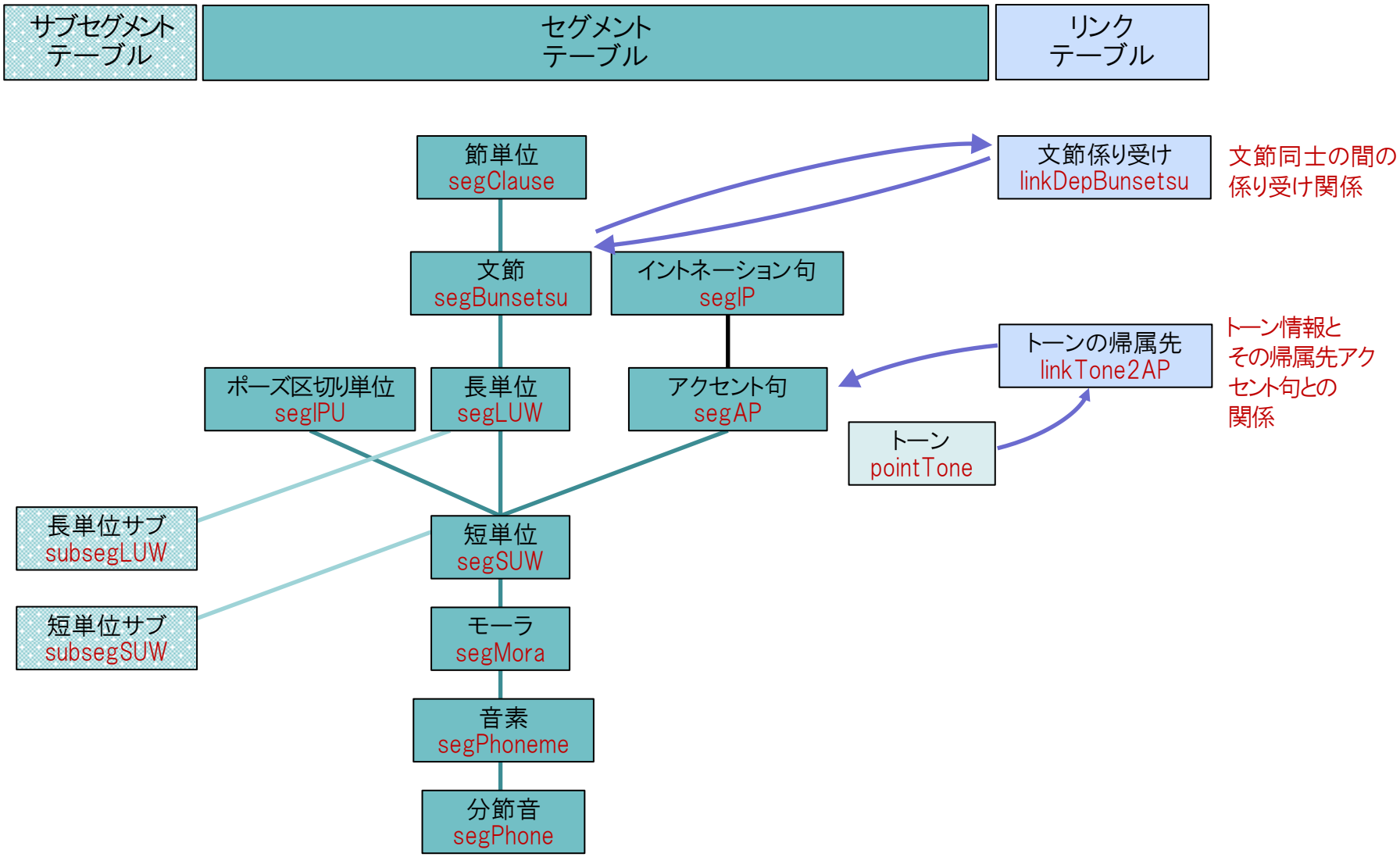
例題17

- **例題17**　サブセグメント・テーブルとセグメント・テーブルを結合

サブ短単位テーブル（subsegSUW）と 短単位テーブルを結合し、
サブ短単位が2短単位以上から成る行を抽出した上で、
TalkID、短単位ID、短単位の基本形（segSUW.OrthographicTranscription）、
サブ短単位の発音形（subsegSUW.PhoneticTranscription）、
品詞（subseSUW.SUWPOS）を選択

ヒント：subsegSUW テーブルの len を用います

CSJ-RDBの構成 ーリンク・テーブルー



リンク・テーブル　linkDepBunsetsu

■ linkDepBunsetsu テーブル

TalkID	BunsetsuID	ModifieeBunsetsuID
A01F0055	00395174L	00395851L

係り元の文節　　係り先の文節

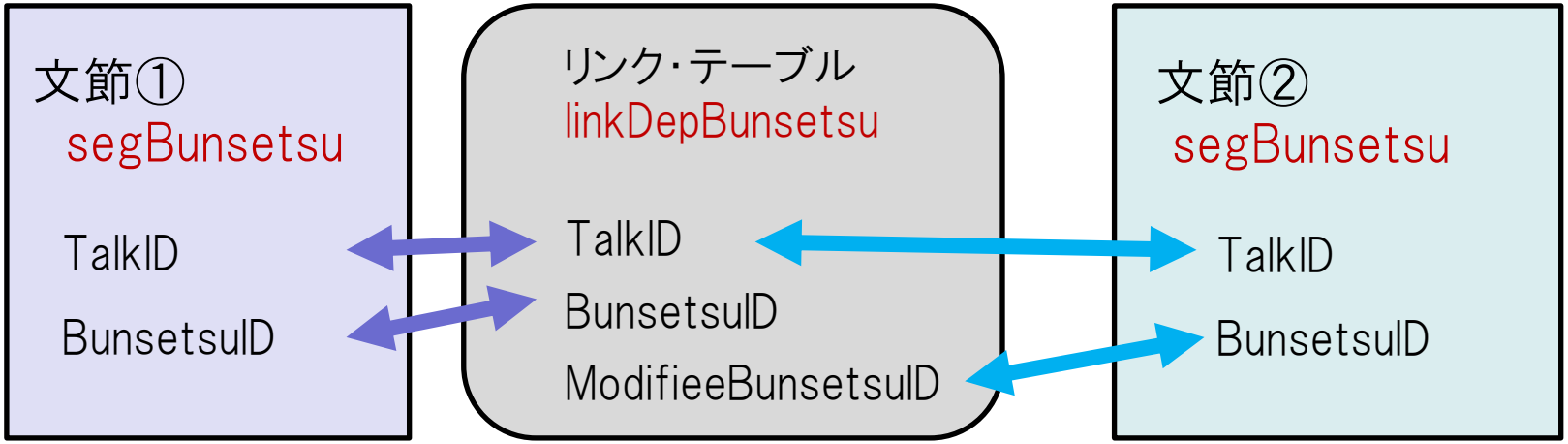
■ segBunsetsu テーブル

TalkID	BunsetsuID	StartTime	EndTime	OrthographicTranscription
A01F0055	00395174L	395.173616	395.850548	こちらだけが
A01F0055	00395851L	395.850548	396.642817	点滅します

例題18

- **例題18**　係り元と係り先の文節を結合

linkDepBunsetsu を利用して係り元と係り先の文節テーブル（いずれもsegBunsetsu）を結合した上で、係り元の文節の基本形が「が」で終わるものを抽出し、TalkID、係り元と係り先の文節の基本形を選択



注意：文節①と文節②のテーブルの名前が同じなので問題が生じます。
　　各テーブルに別名（例えば “segB1”,“segB2”）を付ける必要があります。

リンク・テーブル　linkTone2AP

■ pointTone テーブル

TalkID	ToneID	Time	tone
A01F0055	28	11.77875	%L
A01F0055	29	11.89875	A
A01F0055	30	12.13375	L%
A01F0055	31	12.25875	H%

■ linkTone2AP テーブル

TalkID	ToneID	APID
A01F0055	28	00011679L
A01F0055	29	00011679L
A01F0055	30	00011679L
A01F0055	31	00011679L

帰属元のトーン　　帰属先のAP

■ segAP テーブル

TalkID	APID	StartTime	EndTime	OrthographicTranscription
A01F0055	00011679L	11.67941	12.362888	聴取に

例題19

- **例題19**　帰属元のトーンと帰属先のアクセント句を結合

linkTone2AP を利用して帰属元のトーン(pointTone)と帰属先のアクセント句（segAP）を結合した上で、トーン(tone)がアクセント核“A”である行を抽出し、

TalkID、APの基本形、トーンを選択