SQL Server DDL queries to create all entities for an rCAD database instance

## 1. Sequence Metadata Compartment Database Tables

```
CREATE TABLE SequenceMain (
	SeqID int NOT NULL,
	TaxID int NOT NULL,
	LocationID tinyint NOT NULL,
	SeqTypeID tinyint NOT NULL,
	SeqLength int NOT NULL,
	Source varchar(8000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	Comment varchar(8000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
CONSTRAINT PK_SequenceMain PRIMARY KEY NONCLUSTERED
(
	SeqID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE SequenceType (
	SeqTypeID tinyint NOT NULL,
	MoleculeType varchar(100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
	GeneType varchar(100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
	GeneName varchar(100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
CONSTRAINT PK_SequenceType PRIMARY KEY CLUSTERED
(
	SeqTypeID ASC
) WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE SequenceAccession (
	SeqID int NOT NULL,
	AccessionID varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
	AccessionVersion smallint NOT NULL,
CONSTRAINT PK_SeqAccession PRIMARY KEY NONCLUSTERED
(
	SeqID ASC,
	AccessionID ASC,
	AccessionVersion ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE CellLocationInfo (
	LocationID tinyint NOT NULL,
	Description varchar(100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
CONSTRAINT PK_CellLocationInfo PRIMARY KEY CLUSTERED
(
	LocationID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);
```

## 2. Sequence Alignment Compartment Database Tables and Views

```
CREATE TABLE Alignment (
	AlnID int NOT NULL,
	SeqTypeID tinyint NOT NULL,
	AlignmentName varchar(8000) NULL,
	NextColumnNumber int NOT NULL,
CONSTRAINT PK_Alignment PRIMARY KEY CLUSTERED
(
	AlnID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE AlignmentColumn (
	AlnID int NOT NULL,
	PhysicalColumnNumber int NOT NULL,
	LogicalColumnNumber int NOT NULL,
```

```
CONSTRAINT PK_AlignmentColumn PRIMARY KEY CLUSTERED
(
	AlnID ASC,
	LogicalColumnNumber ASC,
	PhysicalColumnNumber ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE AlignmentSequence (
	SeqID int NOT NULL,
	AlnID int NOT NULL,
	RowLabel varchar(8000) NOT NULL,
	FirstNTPhysicalColumnNumber int NOT NULL,
	LastNTPhysicalColumnNumber int NOT NULL,
CONSTRAINT PK_AlnSequence PRIMARY KEY CLUSTERED
(
	AlnID ASC,
	SeqID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE AlignmentData (
	SeqID int NOT NULL,
	AlnID int NOT NULL,
	PhysicalColumnNumber int NOT NULL,
	BioSymbol char(1) NOT NULL,
	SequenceIndex int NOT NULL,
CONSTRAINT PK_Sequence PRIMARY KEY CLUSTERED
(
	AlnID ASC,
	SeqID ASC,
	PhysicalColumnNumber ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE VIEW vAlignmentGrid
AS
SELECT  AlnSeq.SeqID,
	AlnSeq.AlnID,
	AlnCol.LogicalColumnNumber,
	CASE WHEN AlnDataQuery.BioSymbol IS NULL THEN
		CASE WHEN ((AlnCol.LogicalColumnNumber < AlnColFirstNt.LogicalColumnNumber)
		OR (AlnCol.LogicalColumnNumber > AlnColLastNt.LogicalColumnNumber))
		THEN '~'
		ELSE '-'
		END ELSE
		AlnDataQuery.BioSymbol
	END AS BioSymbol,
	AlnDataQuery.SequenceIndex
FROM
	AlignmentSequence AS AlnSeq INNER JOIN
	AlignmentColumn AS AlnColFirstNt
	ON AlnSeq.AlnID = AlnColFirstNt.AlnID AND AlnSeq.FirstNTPhysicalColumnNumber =
AlnColFirstNt.PhysicalColumnNumber
	INNER JOIN AlignmentColumn AS AlnColLastNt
	ON AlnSeq.AlnID = AlnColLastNt.AlnID AND AlnSeq.LastNTPhysicalColumnNumber =
AlnColLastNt.PhysicalColumnNumber
	INNER JOIN AlignmentColumn AS AlnCol
	ON AlnSeq.AlnID = AlnCol.AlnID LEFT OUTER JOIN
		(SELECT AlnData.AlnID, AlnData.SeqID, AlnData.PhysicalColumnNumber,
AlnData.BioSymbol, AlnData.SequenceIndex
		 FROM AlignmentData AS AlnData) AS AlnDataQuery
  ON AlnDataQuery.SeqID = AlnSeq.SeqID AND AlnDataQuery.AlnID = AlnSeq.AlnID AND
  AlnDataQuery.PhysicalColumnNumber = AlnCol.PhysicalColumnNumber;

CREATE VIEW vAlignmentGridUngapped
AS
SELECT  AlnSeq.SeqID,
	AlnSeq.AlnID,
	AlnCol.LogicalColumnNumber,
	AlnDataQuery.BioSymbol,
```

```
        AlnDataQuery.SequenceIndex
FROM
        AlignmentSequence AS AlnSeq INNER JOIN
        AlignmentColumn AS AlnColFirstNt
        ON AlnSeq.AlnID = AlnColFirstNt.AlnID AND AlnSeq.FirstNTPhysicalColumnNumber =
AlnColFirstNt.PhysicalColumnNumber
        INNER JOIN AlignmentColumn AS AlnColLastNt
        ON AlnSeq.AlnID = AlnColLastNt.AlnID AND AlnSeq.LastNTPhysicalColumnNumber =
AlnColLastNt.PhysicalColumnNumber
        INNER JOIN AlignmentColumn AS AlnCol
        ON AlnSeq.AlnID = AlnCol.AlnID INNER JOIN
                (SELECT AlnData.AlnID, AlnData.SeqID, AlnData.PhysicalColumnNumber,
AlnData.BioSymbol, AlnData.SequenceIndex
                 FROM AlignmentData AS AlnData) AS AlnDataQuery
  ON AlnDataQuery.AlnID = AlnSeq.AlnID AND AlnDataQuery.SeqID = AlnSeq.SeqID AND
  AlnDataQuery.PhysicalColumnNumber = AlnCol.PhysicalColumnNumber;
```

## 3. Structural Relationships Compartment Database Tables

```
CREATE TABLE SecondaryStructureBasePairs (
        SeqID int NOT NULL,
        AlnID int NOT NULL,
        FivePrimeElementSequenceIndex int NOT NULL,
        ThreePrimeElementSequenceIndex int NOT NULL,
CONSTRAINT PK_SecondaryStructureBasePairs PRIMARY KEY CLUSTERED
(
        SeqID ASC,
        AlnID ASC,
        FivePrimeElementSequenceIndex ASC
)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE SecondaryStructureExtents (
        SeqID int NOT NULL,
        AlnID int NOT NULL,
        ExtentID int NOT NULL,
        ExtentOrdinal int NOT NULL,
        ExtentStartIndex int NOT NULL,
        ExtentEndIndex int NOT NULL,
        ExtentTypeID tinyint NOT NULL,
CONSTRAINT PK_SecondaryStructureExtents PRIMARY KEY CLUSTERED
(
        SeqID ASC,
        AlnID ASC,
        ExtentID ASC,
        ExtentOrdinal ASC
)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE SecondaryStructureExtentTypes (
        ExtentTypeID tinyint NOT NULL,
        ExtentType varchar(100) NOT NULL,
CONSTRAINT PK_SeconaryStructureExtentTypes PRIMARY KEY CLUSTERED
(
        ExtentTypeID ASC
)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF)
);
```

## 4. Evolutionary Relationships Compartment Database Tables

```
CREATE TABLE Taxonomy (
        TaxID int NOT NULL,
        ParentTaxID int NOT NULL,
CONSTRAINT PK_Taxonomy PRIMARY KEY CLUSTERED
(
        TaxID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);
```

```
CREATE TABLE TaxonomyNames (
	TaxID int NOT NULL,
	ScientificName varchar(8000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
CONSTRAINT PK_TaxonomyNames PRIMARY KEY CLUSTERED
(
	TaxID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);

CREATE TABLE AlternateNames (
	TaxID int NOT NULL,
	NameClassID tinyint NOT NULL,
	Name varchar(8000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
);

CREATE TABLE TaxonomyNamesOrdered (
	TaxID int NOT NULL,
	ScientificName varchar(8000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
	LineageName varchar(8000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
	Level int NOT NULL,
	SortOrder int NOT NULL,
CONSTRAINT PK_TaxonomyNamesOrdered PRIMARY KEY CLUSTERED
(
	TaxID ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF)
);
```

## 5. rCAD Indices

```
CREATE NONCLUSTERED INDEX IX_SequenceMain_TaxID_SeqTypeID_LocationID
ON SequenceMain (
	TaxID ASC,
	SeqTypeID ASC,
	LocationID ASC
)
INCLUDE ( SeqID );

CREATE NONCLUSTERED INDEX IX_Taxonomy_ParentTaxID
ON Taxonomy (
	ParentTaxID ASC
);

CREATE NONCLUSTERED INDEX IX_AlignmentData_AlnID_SeqID_PhysicalColumnNumber
ON AlignmentData (
	AlnID ASC,
	SeqID ASC,
	PhysicalColumnNumber ASC
)
INCLUDE ( BioSymbol, SequenceIndex );
```

## 6. rCAD User-Defined Functions

```
CREATE FUNCTION fnSelectAllSequencesForTaxonomyByScientificName(@scientificName varchar(8000))
RETURNS @seqSet TABLE
(
	SeqID int
)
AS
BEGIN
	WITH TaxIDSet (TaxID) AS
	(
		SELECT T1.TaxID FROM Taxonomy AS T1
		WHERE T1.TaxID = (SELECT T.TaxID FROM Taxonomy AS T
				  INNER JOIN TaxonomyNames AS TN
				   ON T.TaxID = TN.TaxID
				   WHERE TN.ScientificName=@scientificName)
		UNION ALL
```

```
                SELECT T2.TaxID FROM Taxonomy AS T2
                INNER JOIN TaxIDSet ON T2.ParentTaxID = TaxIDSet.TaxID
        )
        INSERT INTO @seqSet (SeqID)
        SELECT S.SeqID FROM SequenceMain AS S
        INNER JOIN TaxIDSet AS T
        ON S.TaxID = T.TaxID;
        RETURN
END
```

This function uses a SQL Server recursive query (common table expression) to identify all sequences in the rCAD that are part of a specific branch of the phylogenetic tree. The single input parameter is the Scientific Name of the node at the top of the branch of interest. For example, to query all bacterial sequences, use "*Bacteria*".

```
CREATE FUNCTION fnSelectAllSequencesForTaxonomyByTaxID(@parentTaxID int)
RETURNS @seqSet TABLE
(
        SeqID int
)
AS
BEGIN
        WITH TaxIDSet (TaxID) AS
        (
                SELECT T1.TaxID FROM Taxonomy AS T1
                WHERE T1.TaxID = @parentTaxID
                UNION ALL
                SELECT T2.TaxID FROM Taxonomy AS T2
                INNER JOIN TaxIDSet ON T2.ParentTaxID = TaxIDSet.TaxID
        )
        INSERT INTO @seqSet (SeqID)
        SELECT S.SeqID FROM SequenceMain AS S
        INNER JOIN TaxIDSet AS T
        ON S.TaxID = T.TaxID;
        RETURN
END
```

This function uses a SQL Server recursive query (common table expression) to identify all sequences in the rCAD that are part of a specific branch of the phylogenetic tree. The single input parameter is the NCBI Taxonomy ID (TaxID) of the node at the top of the branch of interest. For example, to query all bacterial sequences, use TaxID 2.