```typescript
/* eslint-disable no-use-before-define */
import {
  Association,
  BelongsToManyAddAssociationMixin,
  BelongsToManyCountAssociationsMixin,
  BelongsToManyCreateAssociationMixin,
  BelongsToManyGetAssociationsMixin,
  BelongsToManyHasAssociationMixin,
  DataTypes,
  HasManyAddAssociationMixin,
  HasManyCountAssociationsMixin,
  HasManyCreateAssociationMixin,
  HasManyGetAssociationsMixin,
  HasManyHasAssociationMixin,
  HasOneCreateAssociationMixin,
  HasOneGetAssociationMixin,
  HasOneSetAssociationMixin,
  Model,
} from 'sequelize';

import {FriendlyId, UUID} from '@common/domain/genericTypes';
import {JobRun} from '@common/domain/job/jobRun';
import {RUN_STATUS_NOT_STARTED} from '@common/domain/job/runStatus';
import {StudyPermissions} from '@common/domain/permissions/study';
import {StudyAnalysisPermissions} from '@common/domain/permissions/studyAnalysis';
import {Study, StudyAnalysis, StudyAnalysisAttributes, StudyAttributes} from '@common/domain/study';
import {ComputationType, StudyRun} from '@common/domain/studyRun';
import {UserSummary} from '@common/domain/user';
import {CounterScreenModel} from '@server/infra/postgres/models/counterScreenModel';
import {ReadsetModel} from '@server/infra/postgres/models/readsetModel';
import {ReferenceDbModel} from '@server/infra/postgres/models/referenceDbModel';
import {StudyRunModel} from '@server/infra/postgres/
```

```typescript
models/studyRunModel';
import {sequelizeOptions} from '@server/infra/postgres/sequelizeInstance';
import {
  comment,
  createdAt,
  createdBy,
  description,
  friendlyId,
  id,
  lastModifiedBy,
  name,
  permissions,
  updatedAt,
} from '@server/infra/postgres/sharedAttributes';

export class StudyAnalysisModel extends Model<StudyAnalysis, StudyAnalysisAttributes> implements StudyAnalysis {
  declare static associations: {
    studyRuns: Association<StudyAnalysisModel, StudyRunModel>;
  };
  declare id: UUID;
  declare name: string;
  declare computationParameters: Record<string, unknown>;
  declare computationType: ComputationType;
  declare description?: string | null;
  declare friendlyId: FriendlyId;
  /**
   * associations
   */
  declare studyId: string;
  /**
   * Audit
   */
  declare createdBy: UserSummary;
  declare lastModifiedBy: UserSummary;
  /**
   * Automatically added by Sequelize. ISO timestamp.
   *
   * **WARNING**: Not really a string! Actually a Date.
   * This only becomes a string after it's stringified when sent from the server to the client.
   * Which means:
   * * while on server, the real type is Date
   * * on client, the real type is String
   */
  declare readonly createdAt: string;
  /**
   * Automatically added by Sequelize. ISO timestamp.
   *
   * **WARNING**: Not really a string! Actually a Date.
   * This only becomes a string after it's stringified when sent from the server to the client.
   * Which means:
   * * while on server, the real type is Date
   * * on client, the real type is String
   */
  declare readonly updatedAt: string;
  declare permissions: StudyAnalysisPermissions;
  /**
   * StudyRuns
   */
  declare getStudyRuns: HasManyGetAssociationsMixin<StudyRunModel>;
  declare addStudyRun: HasManyAddAssociationMixin<StudyRunModel, string>;
  declare hasStudyRun: HasManyHasAssociationMixin<StudyRunModel, string>;
  declare countStudyRuns: HasManyCountAssociationsMixin;
  declare createStudyRun: HasManyCreateAssociationMixin<StudyRunModel>;
}

StudyAnalysisModel.init(
  {
    id,
    friendlyId,
    createdAt,
    updatedAt,
    createdBy,
    computationParameters: {
      type: DataTypes.JSON,
    },
    computationType: {
      type: DataTypes.STRING,
    },
    lastModifiedBy,
    name,
    description,
    permissions,
    studyId: {
      type: DataTypes.UUID,
      allowNull: false,
    },
  },
  {
    ...sequelizeOptions,
    tableName: 'analysis',
  },
);
export class StudyModel extends Model<Study, StudyAttributes> implements Study {
  declare id: UUID;
  declare studyNumber: string;
  declare subject: string;
  declare description?: string | null;
  declare tags: string[];
  declare friendlyId: FriendlyId;
  declare name: string;

  /**
   * Audit
   */
  declare createdBy: UserSummary;
  declare lastModifiedBy: UserSummary;

  /**
   * Automatically added by Sequelize. ISO timestamp.
   *
   * **WARNING**: Not really a string! Actually a Date.
   * This only becomes a string after it's stringified when sent from the server to the client.
   * Which means:
   * * while on server, the real type is Date
   * * on client, the real type is String
   */
  declare readonly createdAt: string;
  /**
   * Automatically added by Sequelize. ISO timestamp.
   *
   * **WARNING**: Not really a string! Actually a Date.
   * This only becomes a string after it's stringified when sent from the server to the client.
   * Which means:
   * * while on server, the real type is Date
   * * on client, the real type is String
   */
  declare readonly updatedAt: string;

  declare permissions: StudyPermissions;

  /**
   * StudyRuns
   */
  declare getStudyRuns: HasManyGetAssociationsMixin<StudyRunModel>;
  declare addStudyRun: HasManyAddAssociationMixin<StudyRunModel, string>;
  declare hasStudyRun: HasManyHasAssociationMixin<StudyRunModel, string>;
  declare countStudyRuns: HasManyCountAssociationsMixin;
  declare createStudyRun:
```

```
HasManyCreateAssociationMixin<StudyRunModel>;

  /**
   * Readsets
   */
  declare getReadsets: BelongsToManyGetAssociationsMixin<ReadsetModel>;
  declare addReadset: BelongsToManyAddAssociationMixin<ReadsetModel, string>;
  declare removeReadset: BelongsToManyAddAssociationMixin<ReadsetModel, string>;
  declare hasReadset: BelongsToManyHasAssociationMixin<ReadsetModel, string>;
  declare countReadsets: BelongsToManyCountAssociationsMixin;
  declare createReadset: BelongsToManyCreateAssociationMixin<ReadsetModel>;

  /**
   * inclusions are optional, populated when explicitly include a relation
   */
  declare readonly readsets?: ReadsetModel[];

  declare static associations: {
    studyRuns: Association<StudyModel, StudyRunModel>;
    readsets: Association<StudyModel, ReadsetModel>;
    analysis: Association<StudyModel, StudyAnalysisModel>;
  };
}

export class StudyRunModel extends Model<StudyRun, StudyRunCreationAttributes> implements StudyRun {
  declare id: UUID;
  declare friendlyId: FriendlyId;
  declare name: string;
  declare description: string;
  declare computationParameters: Record<string, unknown>;
  declare computationType: ComputationType;
  declare pipeline: JobRun;
  declare comment: string;

  /**
   * Audit
   */
  declare createdBy: UserSummary;
  declare lastModifiedBy: UserSummary;

  /**
   * Automatically added by Sequelize. ISO timestamp.
   *
   * **WARNING**: Not really a string! Actually a Date.
   * This only becomes a string after it's stringified when sent from the server to the client.
   * Which means:
   * * while on server, the real type is Date
   * * on client, the real type is String
   */
  declare readonly createdAt: string;
  /**
   * Automatically added by Sequelize. ISO timestamp.
   *
   * **WARNING**: Not really a string! Actually a Date.
   * This only becomes a string after it's stringified when sent from the server to the client.
   * Which means:
   * * while on server, the real type is Date
   * * on client, the real type is String
   */
  declare readonly updatedAt: string;

  /**
```

```
   * associations
   */
  declare studyId: string;
  declare readsetId: string;
  declare referenceDbId: string;
  declare analysisId: string;

  /**
   * Reference DBs
   */
  declare getReferenceDb: HasOneGetAssociationMixin<ReferenceDbModel>;
  declare setReferenceDb: HasOneSetAssociationMixin<ReferenceDbModel, string>;
  declare createReferenceDb: HasOneCreateAssociationMixin<ReferenceDbModel>;

  /**
   * Readset
   */
  declare getReadset: HasOneGetAssociationMixin<ReadsetModel>;
  declare setReadset: HasOneSetAssociationMixin<ReadsetModel, string>;
  declare createReadset: HasOneCreateAssociationMixin<ReadsetModel>;

  /**
   * Counter Screen
   */
  declare getCounterScreens: HasManyGetAssociationsMixin<CounterScreenModel>;
  declare createCounterScreen: HasManyCreateAssociationMixin<CounterScreenModel>;

  declare static associations: {
    study: Association<StudyRunModel, StudyModel>;
    readset: Association<StudyRunModel, ReadsetModel>;
    refDb: Association<StudyRunModel, ReferenceDbModel>;
    counterScreens: Association<StudyRunModel, CounterScreenModel>;
  };
}

StudyRunModel.init(
  {
    id,
    friendlyId,
    name,
    description,
    comment,
    computationParameters: {
      type: DataTypes.JSON,
    },
    computationType: {
      type: DataTypes.STRING,
    },
    createdBy,
    lastModifiedBy,
    createdAt,
    updatedAt,
    studyId: {
      type: DataTypes.UUID,
      allowNull: false,
    },
    analysisId: {
      type: DataTypes.UUID,
      allowNull: false,
    },
    readsetId: {
      type: DataTypes.UUID,
    },
    referenceDbId: {
      type: DataTypes.UUID,
    },
    pipeline: {
      type: DataTypes.JSON,
      allowNull: false,
      defaultValue: {
        status: RUN_STATUS_NOT_STARTED,
        hpcJob: null,
      },
    },
  },
  {
    ...sequelizeOptions,
    tableName: 'studyRuns',
  },
);

```