Metabolite Database

--A bioinformatics website

Group 23

Outline of presentation

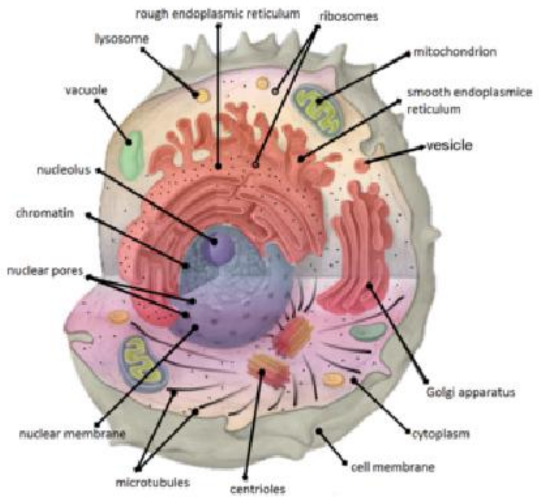
- Motivation and Application Description
- Dataset Description and Reorganization
- Demonstration
- Implementation
- Evaluation
- Conclusion

Motivation

- Bioinformatics has been an emerging and promising research field since the late 20th century.
- With the high-throughput technology, the number of known metabolites, reaction, genes are increasing exponentially.
- The types of existing database structures do not meet the diverse requirements of scientific research

A quick recap of the biological concept

Compartments and Metabolites



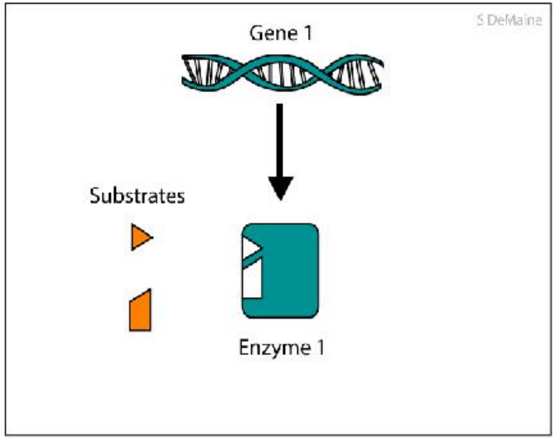
For each notation, it is a compartment.
https://en.wikipedia.org/wiki/Cellular_compartment

The metabolites model is always divided by different boundaries. For each closed part, we call it "Compartment". Usually it is surrounded by a single or double lipid layer membrane.

The same chemical(or compound) in the different compartments would be believed as a different metabolites, for example: the glucose in the cytosol, would be denoted by: glu_D_c (Bigg_id), but the glucose in the extracellular space would be denoted by another metoblite: glu_D_e. However, they are the same compound within the different compartment.

A quick recap of the biological concept

Reactions and Gene



Some genes are assoicated with the production of the enzyme.
https://droso4schools.wordpress.com/l4-enzymes/

The **reactions** are the process where one or more metabolites are converted to one or more different metabolites.

stoichometry: strch1_e + 8 h2o_e -> 8 glc__D_e + strch2_e

A **gene** is the basic physical and functional unit of heredity. Genes are made up of DNA. Some genes act as instructions to make molecules called proteins. However, many genes do not code for proteins.

Therefore, we know the reactions and the metabolites have a strong relationship in the dataset, however, the gene and reactions have a weak relationship. In our database, we only show the relationship between the reactions and the metabolites

Functionalities

Database Management :

- Querying the reactions, metabolites, genes.
- Adding/modifying/deleting the information with a Admin User account.
- Showing the relationship between the metabolites and reactions.
- Interactive plugin like leaving a comment.

Metabolic engineering :

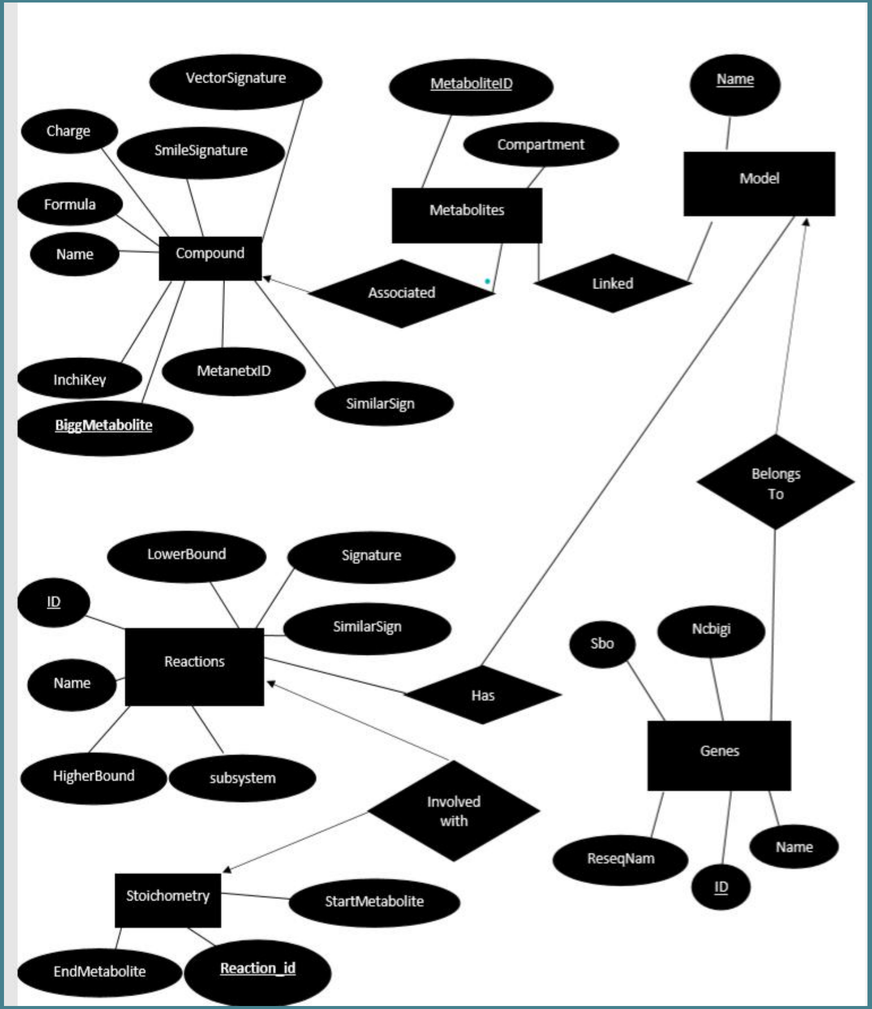
- Retrieving the metabolic pathways with certain steps reactions.
- Finding all the products that a source metabolites can produce.

Dataset

- We acquired the data from the BiGG database (A Biochemical Genetic and Genomic (BiGG) knowledge-base of large scale metabolic reconstructions URL: http://bigg.ucsd.edu/)
- We extract the data from their website in JSON files.
  - The datasets were clean to begin with which made populating our database go very smoothly
  - JSON is easy to read and parse as well.
- Size of Tables
  - 14069 Metabolites
  - 25193 Reactions
  - 10584 Genes
  - 8002 Compounds

ER Model

Checkpoint 2

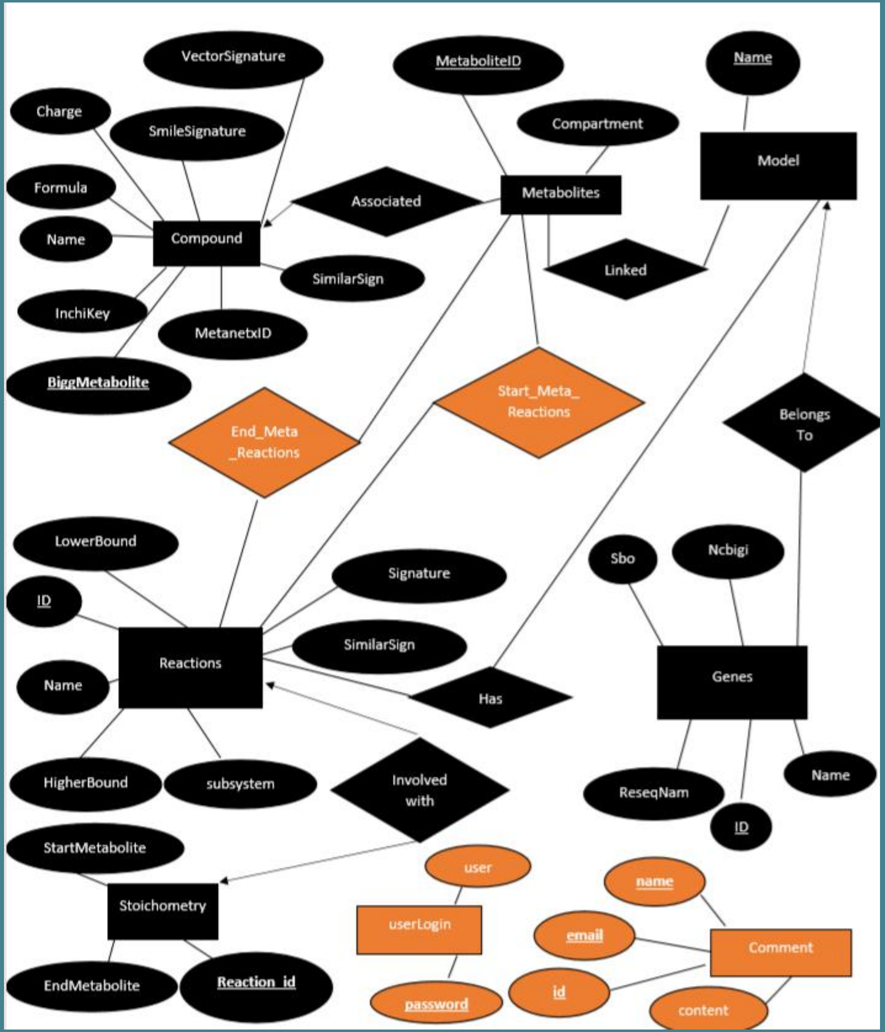


Final ER Model

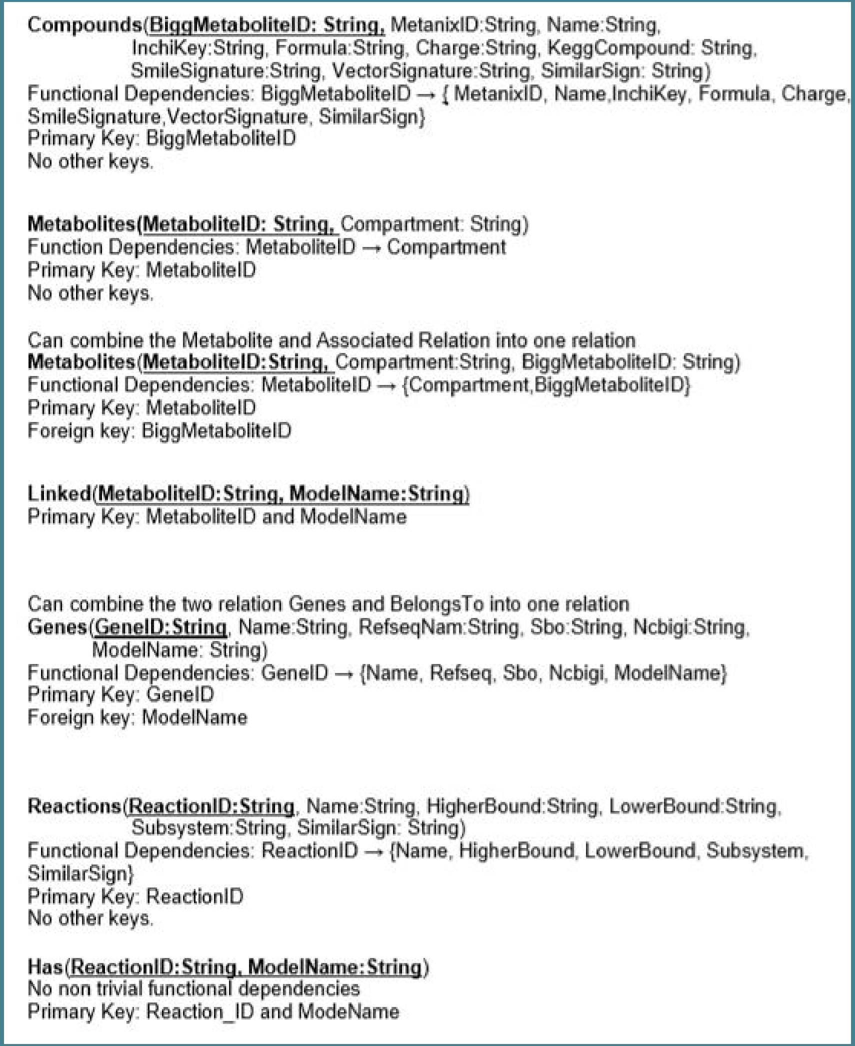
*Added Entities and Relations

Entities: 8

Relationships: 7



Relational Model



Compounds(BiggMetaboliteID: String, MetanixID:String, Name:String, InchiKey:String, Formula:String, Charge:String, KeggCompound: String, SmileSignature:String, VectorSignature:String, SimilarSign: String)
Functional Dependencies: BiggMetaboliteID → { MetanixID, Name,InchiKey, Formula, Charge, SmileSignature,VectorSignature, SimilarSign}
Primary Key: BiggMetaboliteID
No other keys.

Metabolites(MetaboliteID: String, Compartment: String)
Function Dependencies: MetaboliteID → Compartment
Primary Key: MetaboliteID
No other keys.

Can combine the Metabolite and Associated Relation into one relation
Metabolites(MetaboliteID:String, Compartment:String, BiggMetaboliteID: String)
Functional Dependencies: MetaboliteID → {Compartment,BiggMetaboliteID}
Primary Key: MetaboliteID
Foreign key: BiggMetaboliteID

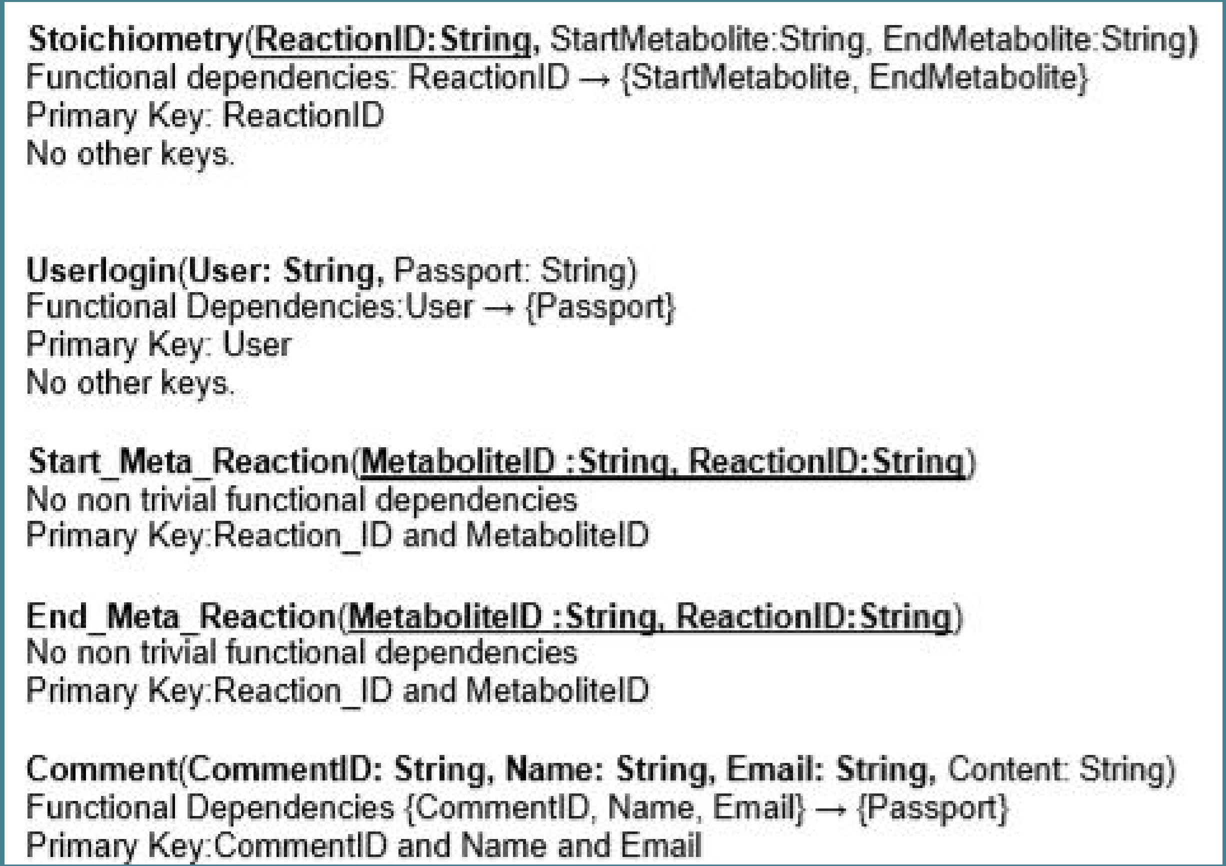
Linked(MetaboliteID:String, ModelName:String)
Primary Key: MetaboliteID and ModelName

Can combine the two relation Genes and BelongsTo into one relation
Genes(GeneID:String, Name:String, RefseqNam:String, Sbo:String, Ncbigi:String, ModelName: String)
Functional Dependencies: GeneID → {Name, Refseq, Sbo, Ncbigi, ModelName}
Primary Key: GeneID
Foreign key: ModelName

Reactions(ReactionID:String, Name:String, HigherBound:String, LowerBound:String, Subsystem:String, SimilarSign: String)
Functional Dependencies: ReactionID → {Name, HigherBound, LowerBound, Subsystem, SimilarSign}
Primary Key: ReactionID
No other keys.

Has(ReactionID:String, ModelName:String)
No non trivial functional dependencies
Primary Key: Reaction_ID and ModeName

Relational Model



Stoichiometry(ReactionID:String, StartMetabolite:String, EndMetabolite:String**)**
Functional dependencies: ReactionID → {StartMetabolite, EndMetabolite}
Primary Key: ReactionID
No other keys.

Userlogin(User: String, Passport: String)
Functional Dependencies:User → {Passport}
Primary Key: User
No other keys.

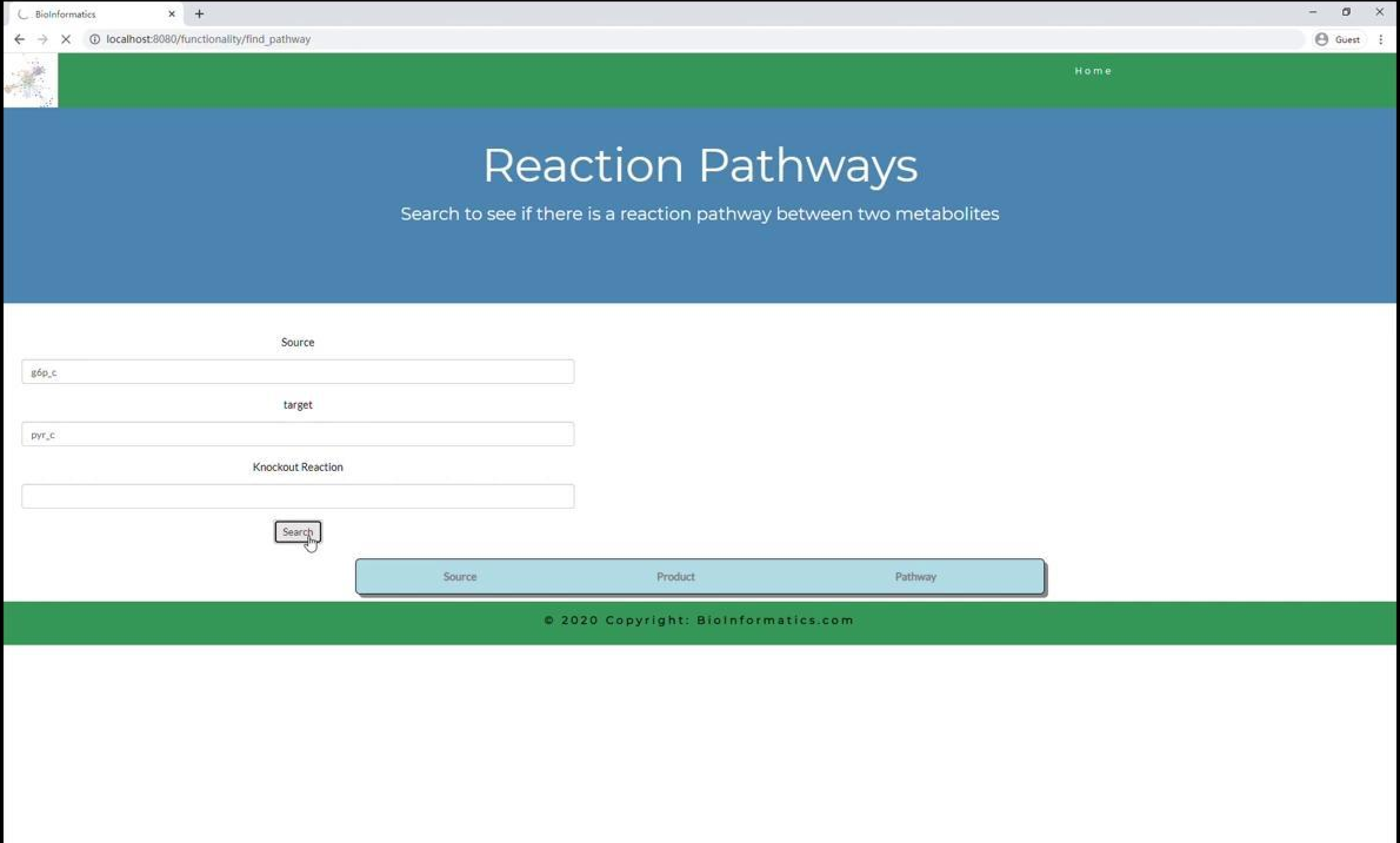
Start_Meta_Reaction(MetaboliteID :String, ReactionID:String)
No non trivial functional dependencies
Primary Key:Reaction_ID and MetaboliteID

End_Meta_Reaction(MetaboliteID :String, ReactionID:String)
No non trivial functional dependencies
Primary Key:Reaction_ID and MetaboliteID

Comment(CommentID: String, Name: String, Email: String, Content: String)
Functional Dependencies {CommentID, Name, Email} → {Passport}
Primary Key:CommentID and Name and Email

Normalization Forms

Relations	Functional Dependencies	Best Normal Form Achieved
Compounds	FD: BiggMetaboliteID → {All Attributes}	BCNF: BiggMetaboliteID is a non-trivial candidate key
Metabolite	FD: MetaboliteID → {All Attributes}	BCNF: MetaboliteID is a non-trivial candidate key
Linked	No Functional Dependency	BCNF and 3NF: No FD relation is therefore 3NF and BCNF
Genes	FD: GeneID → {All Attributes}	BCNF: GeneID is a non-trivial candidate key
Reaction	FD: ReactionID → {All Attributes}	BCNF: ReactionID is a non-trivial candidate key
Has	No Functional Dependency	BCNF and 3NF: No FD relation is therefore 3NF and BCNF
Stoichiometry	FD: ReactionID → {All Attributes}	BCNF: ReactionID is a non-trivial candidate key
UserLogin	FD: User→ {All Attributes}	BCNF: User is a non-trivial candidate key
Start_Meta_Reaction	No Functional Dependency	BCNF and 3NF: No FD relation is therefore 3NF and BCNF:
End_Meta_Reaction	No Functional Dependency	BCNF and 3NF: No FD relation is therefore 3NF and BCNF
Comment	FD: Comment, Name, and Email → {All Attributes}	BCNF: Comment, Name, and Email are non-trivial candidate key



BioInformatics

localhost:8080/functionality/find_pathway

Guest

Home

Reaction Pathways

Search to see if there is a reaction pathway between two metabolites

Source

g6p_c

target

pyr_c

Knockout Reaction

Search

Source	Product	Pathway

© 2020 Copyright: BioInformatics.com

Tech Stack

Backend:

- Java
- JPA and Hibernate: Java API for interacting with Relational Databases
- Spring Boot - Framework for creating Java based applications
- MySQL

Frontend:

- Thymeleaf - Java Template Engine
- HTML
- CSS
- JavaScript

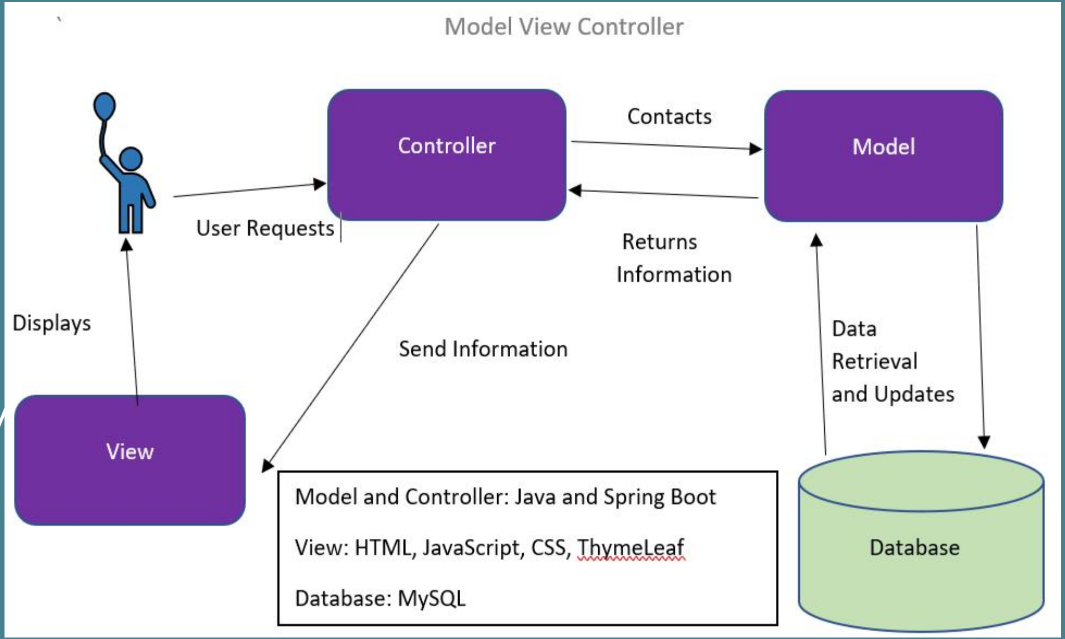
System Architecture

Software Architecture:

Restful Web Service

Software design pattern:

Model View Controller (MV

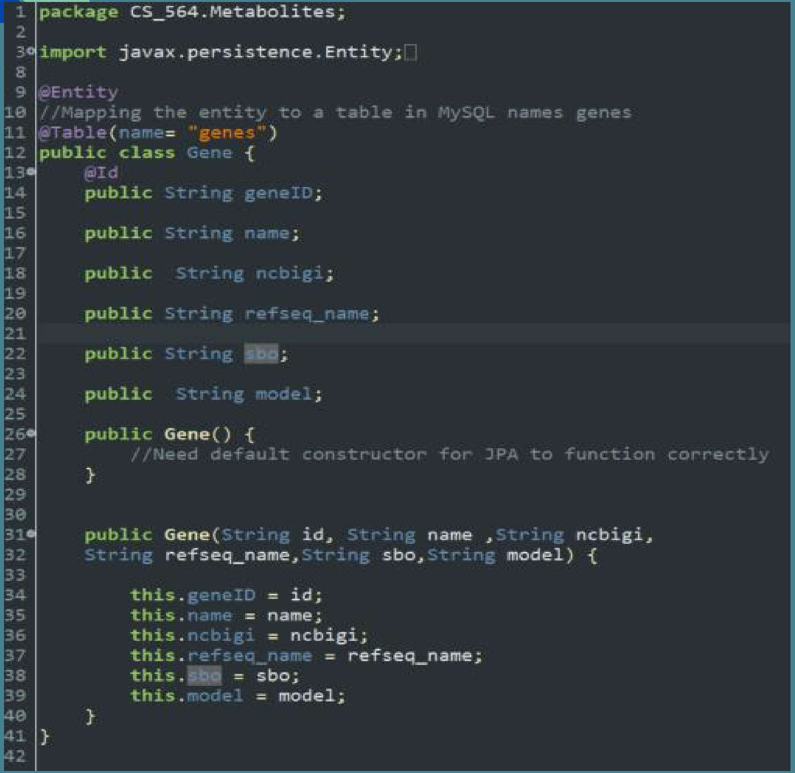


Model and Controller: Java and Spring Boot

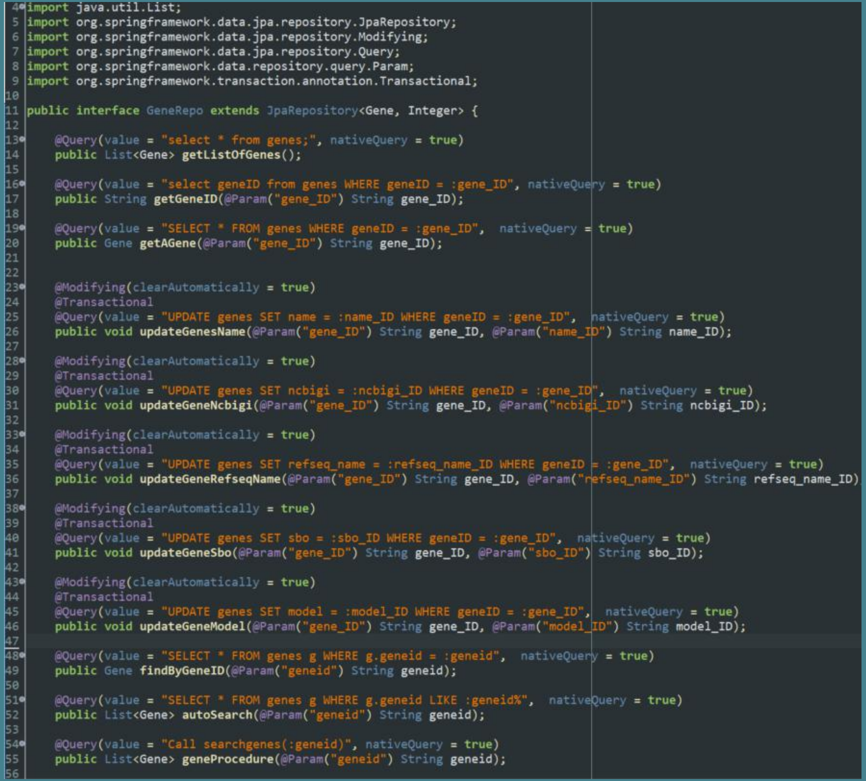
View: HTML, JavaScript, CSS, ThymeLeaf

Database: MySQL

Java Persistence API and Hibernate: Entity Class And Queries (an instance of genes table)

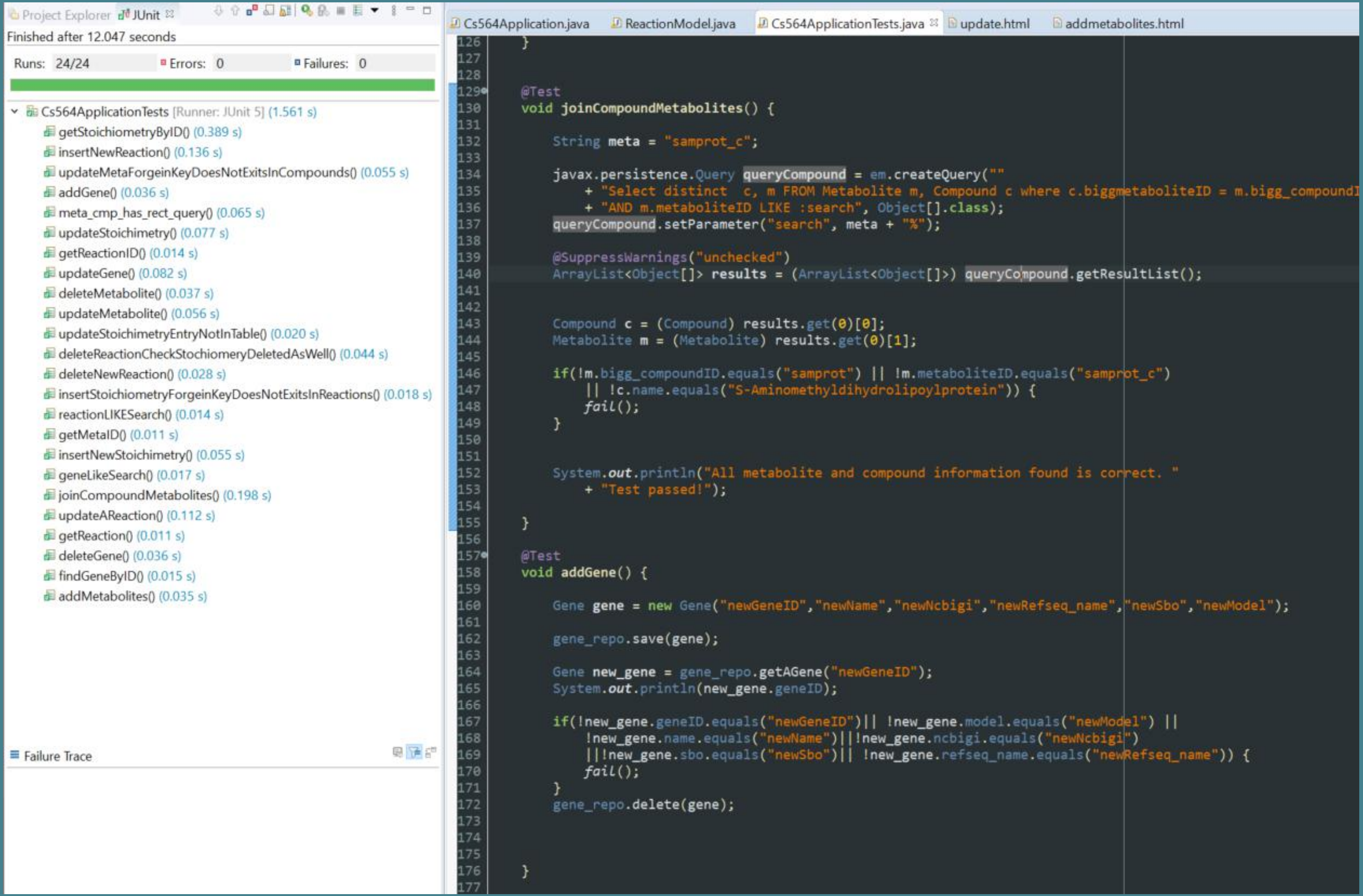


```java
package CS_564.Metabolites;

import javax.persistence.Entity;

@Entity
//Mapping the entity to a table in MySQL names genes
@Table(name= "genes")
public class Gene {
    @Id
    public String geneID;

    public String name;

    public  String ncbigi;

    public String refseq_name;

    public String sbo;

    public  String model;

    public Gene() {
        //Need default constructor for JPA to function correctly
    }


    public Gene(String id, String name ,String ncbigi,
    String refseq_name,String sbo,String model) {

        this.geneID = id;
        this.name = name;
        this.ncbigi = ncbigi;
        this.refseq_name = refseq_name;
        this.sbo = sbo;
        this.model = model;
    }
}

```



```java
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

public interface GeneRepo extends JpaRepository<Gene, Integer> {

    @Query(value = "select * from genes;", nativeQuery = true)
    public List<Gene> getListOfGenes();

    @Query(value = "select geneID from genes WHERE geneID = :gene_ID", nativeQuery = true)
    public String getGeneID(@Param("gene_ID") String gene_ID);

    @Query(value = "SELECT * FROM genes WHERE geneID = :gene_ID",  nativeQuery = true)
    public Gene getAGene(@Param("gene_ID") String gene_ID);


    @Modifying(clearAutomatically = true)
    @Transactional
    @Query(value = "UPDATE genes SET name = :name_ID WHERE geneID = :gene_ID",  nativeQuery = true)
    public void updateGenesName(@Param("gene_ID") String gene_ID, @Param("name_ID") String name_ID);

    @Modifying(clearAutomatically = true)
    @Transactional
    @Query(value = "UPDATE genes SET ncbigi = :ncbigi_ID WHERE geneID = :gene_ID",  nativeQuery = true)
    public void updateGeneNcbigi(@Param("gene_ID") String gene_ID, @Param("ncbigi_ID") String ncbigi_ID);

    @Modifying(clearAutomatically = true)
    @Transactional
    @Query(value = "UPDATE genes SET refseq_name = :refseq_name_ID WHERE geneID = :gene_ID",  nativeQuery = true)
    public void updateGeneRefseqName(@Param("gene_ID") String gene_ID, @Param("refseq_name_ID") String refseq_name_ID)

    @Modifying(clearAutomatically = true)
    @Transactional
    @Query(value = "UPDATE genes SET sbo = :sbo_ID WHERE geneID = :gene_ID",  nativeQuery = true)
    public void updateGeneSbo(@Param("gene_ID") String gene_ID, @Param("sbo_ID") String sbo_ID);

    @Modifying(clearAutomatically = true)
    @Transactional
    @Query(value = "UPDATE genes SET model = :model_ID WHERE geneID = :gene_ID",  nativeQuery = true)
    public void updateGeneModel(@Param("gene_ID") String gene_ID, @Param("model_ID") String model_ID);

    @Query(value = "SELECT * FROM genes g WHERE g.geneid = :geneid",  nativeQuery = true)
    public Gene findByGeneID(@Param("geneid") String geneid);

    @Query(value = "SELECT * FROM genes g WHERE g.geneid LIKE :geneid%",  nativeQuery = true)
    public List<Gene> autoSearch(@Param("geneid") String geneid);

    @Query(value = "Call searchgenes(:geneid)", nativeQuery = true)
    public List<Gene> geneProcedure(@Param("geneid") String geneid);

```

Evaluation : JUnit Tests



Project Explorer | JUnit

Finished after 12.047 seconds

Runs: 24/24 | Errors: 0 | Failures: 0

```
Cs564ApplicationTests [Runner: JUnit 5] (1.561 s)
    getStoichiometryByID() (0.389 s)
    insertNewReaction() (0.136 s)
    updateMetaForgeinKeyDoesNotExitsInCompounds() (0.055 s)
    addGene() (0.036 s)
    meta_cmp_has_rect_query() (0.065 s)
    updateStoichimetry() (0.077 s)
    getReactionID() (0.014 s)
    updateGene() (0.082 s)
    deleteMetabolite() (0.037 s)
    updateMetabolite() (0.056 s)
    updateStoichimetryEntryNotInTable() (0.020 s)
    deleteReactionCheckStochiomeryDeletedAsWell() (0.044 s)
    deleteNewReaction() (0.028 s)
    insertStoichimetryForgeinKeyDoesNotExitsInReactions() (0.018 s)
    reactionLIKESearch() (0.014 s)
    getMetaID() (0.011 s)
    insertNewStoichimetry() (0.055 s)
    geneLikeSearch() (0.017 s)
    joinCompoundMetabolites() (0.198 s)
    updateAReaction() (0.112 s)
    getReaction() (0.011 s)
    deleteGene() (0.036 s)
    findGeneByID() (0.015 s)
    addMetabolites() (0.035 s)
```

Failure Trace

Cs564Application.java | ReactionModel.java | Cs564ApplicationTests.java | update.html | addmetabolites.html

```java
    }


    @Test
    void joinCompoundMetabolites() {

        String meta = "samprot_c";

        javax.persistence.Query queryCompound = em.createQuery(""
            + "Select distinct  c, m FROM Metabolite m, Compound c where c.biggmetaboliteID = m.bigg_compoundI
            + "AND m.metaboliteID LIKE :search", Object[].class);
        queryCompound.setParameter("search", meta + "%");

        @SuppressWarnings("unchecked")
        ArrayList<Object[]> results = (ArrayList<Object[]>) queryCompound.getResultList();


        Compound c = (Compound) results.get(0)[0];
        Metabolite m = (Metabolite) results.get(0)[1];

        if(!m.bigg_compoundID.equals("samprot") || !m.metaboliteID.equals("samprot_c")
            || !c.name.equals("S-Aminomethyldihydrolipoylprotein")) {
            fail();
        }


        System.out.println("All metabolite and compound information found is correct. "
            + "Test passed!");

    }

    @Test
    void addGene() {

        Gene gene = new Gene("newGeneID","newName","newNcbigi","newRefseq_name","newSbo","newModel");

        gene_repo.save(gene);

        Gene new_gene = gene_repo.getAGene("newGeneID");
        System.out.println(new_gene.geneID);

        if(!new_gene.geneID.equals("newGeneID")|| !new_gene.model.equals("newModel") ||
            !new_gene.name.equals("newName")||!new_gene.ncbigi.equals("newNcbigi")
            ||!new_gene.sbo.equals("newSbo")|| !new_gene.refseq_name.equals("newRefseq_name")) {
            fail();
        }
        gene_repo.delete(gene);



    }
```

Conclusion/Lessons Learned

- Have a chance to practice what we learned about the DBMS in the lecture, like how to organize the dataset, how to optimize the query, etc.
- Created a multidisciplinary project that converted biological concept into computer science project.
- First time making an web application with an embedded Relational Database. It was a really a good experience.
- Learned a lot of new frameworks, languages and API's,
  - Spring Boot, Thymeleaf, JavaScripts, Java persistence ...

Thanks for listening !

Any questions?