Database Concepts

John “Scooter” Morris

Genentech, Inc.

Overview

Limitations

Data Modeling

Data Access Methods

Types of Databases

Uses of Databases

- **Genomic Data**
  - **Problems….**

Limitations

2 Hours is not enough!

What am I not telling you about?

- **Database normalization**
- **Object-based approaches to database design**
- **Object-relational mapping**
- **Relational calculus, relational algebra**
- **…. Too much more to mention ….**

This is an *introduction*

- **enough to get started and to know what you don't know (I hope)**

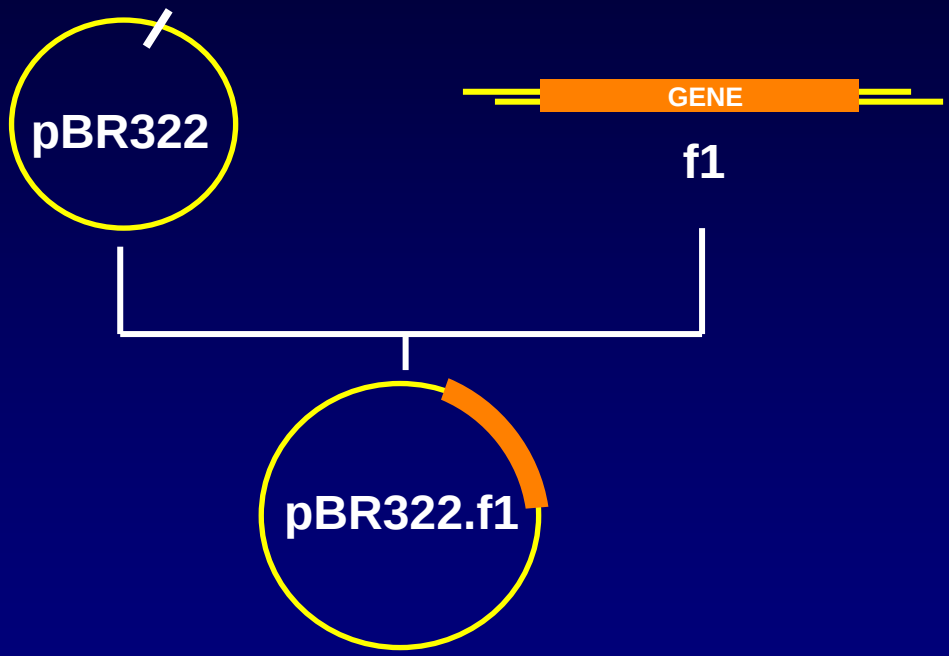
Example Problem

A system to automate the tracking and documentation of plasmid construction

Terminology:

- **fragment: a length of double-stranded DNA**
- **plasmid: a circular fragment**
- **recipe: a series of manipulations of the DNA to produce a new plasmid with cDNA of interest inserted**
- **ACL: access control list**

Example problem



Data Modeling

The FIRST Step

Structured way to understand the data semantics

Independent of underlying platform

Way to communicate with team members (including users)

Excellent (minimal?) documentation

Example: ER Diagrams

ER Diagrams

Entity (Entity Type)

- A collection of entities that share common properties
  - e.g. Fragment, Recipe, Gene

Attribute

- Property of an entity that is of interest
  - e.g. Name, File, Sequence

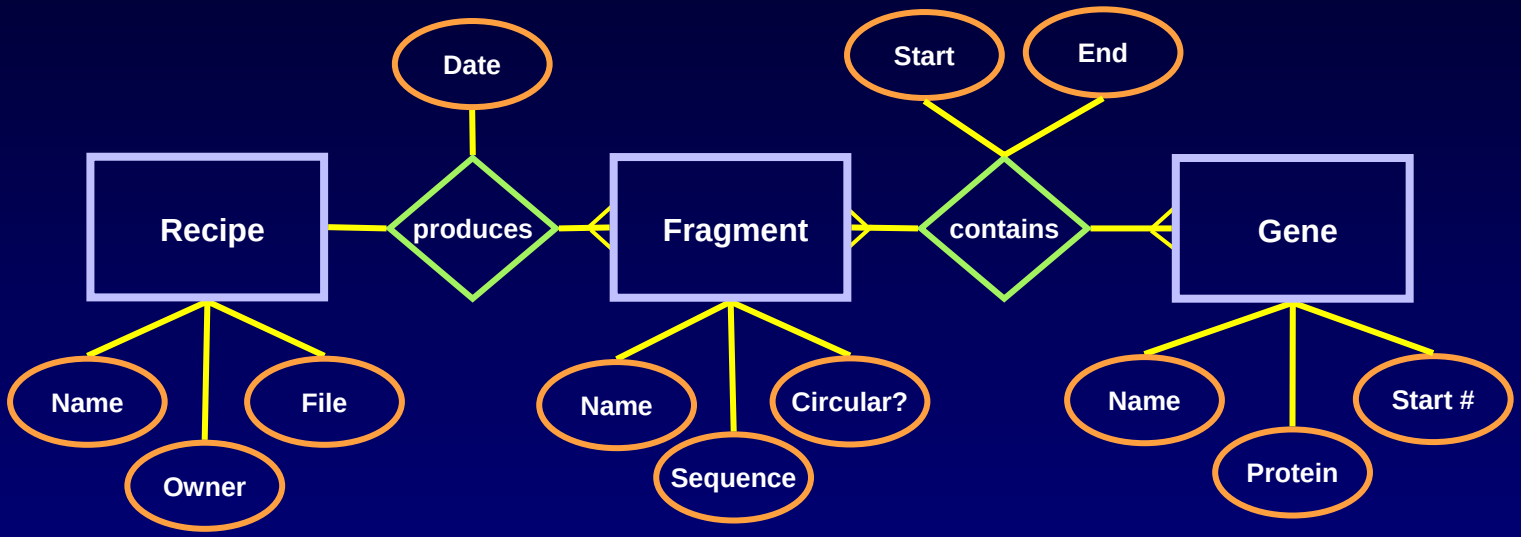
Relationship

- An association between entities
  - e.g. Produces

Degree

- Number of entities involved in the relationship
  - one-to-many, one-to-one, many-to-many

ER Diagrams


Date
Start
End
Recipe
produces
Fragment
contains
Gene
Name
File
Owner
Name
Circular?
Sequence
Name
Start #
Protein

ER Diagrams

Questions?

Recommended Reading:

- Chen, P.S. The entity-relationship model: toward a unified view of data. ACM Trans on Database Syst. pp 9-36 (March 1976)

Data Access Methods

How is the data accessed?

Why do we care?

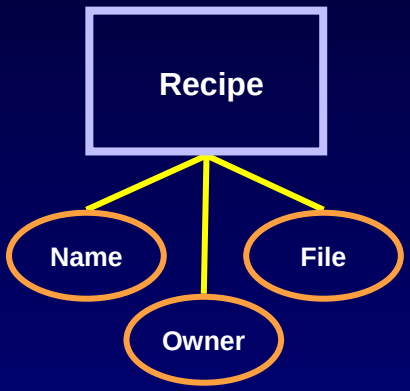
- Important for special-purpose databases
- Some systems give you choices

Terminology:

- Index: an access path into the data
- Key: a field used to access the data
- Primary key: a field (combination) whose values uniquely identify the record

Data Access Methods - Linear

Simple record-oriented view



ID	NAME	FILE	OWNER
F1	pBR322	pBR322.cr	scooter
F2	pBR322.f1	pBR322f1.cr	scooter
F3	f1	f1.cr	ckw
F4	pHR5CV	pHR5CV.cr	ckw
F5	f2	f2.cr	wiw

Access is through sequential reads

OK for small data stores -- very slow when the number of records gets large

Data Access Methods - Hash

Compute a function to access the data

- **e.g. add up the characters to produce an integer**

Usually requires a separate *index*

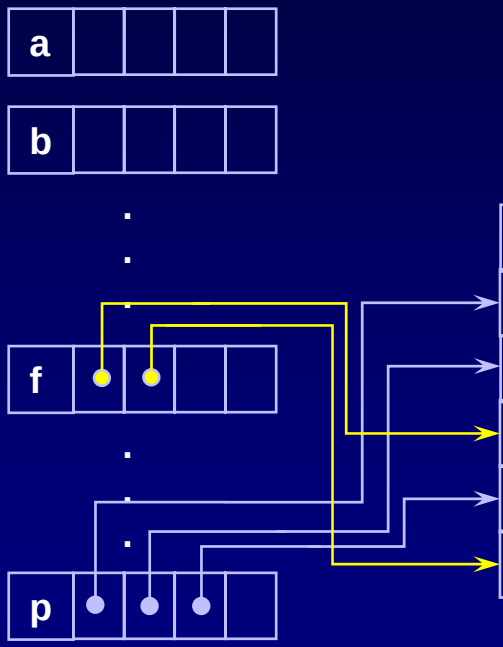
The "goodness" of the hash function is important

- **A perfect hash function would result in a direct access to the data (i.e. a one-to-one relationship)**
- **Perfect hash functions are almost never possible**
- **This results in the possibility of multiple "hits" per hash value (or bucket)**

Data Access Methods - Hash

Simple (and silly) example:

- Hash on the first letter of the recipe name



ID	NAME	FILE	OWNER
F1	pBR322	pBR322.cr	scooter
F2	pBR322.f1	pBR322f1.cr	scooter
F3	f1	f1.cr	ckw
F4	pHR5CV	pHR5CV.cr	ckw
F5	f2	f2.cr	wiw

Data Access Methods - BTree

Good for sequenced or character data

In general, the *index set* is a tree whose leaves consist of pointers into a *sequence* set

Each node in the index set points to three lower nodes

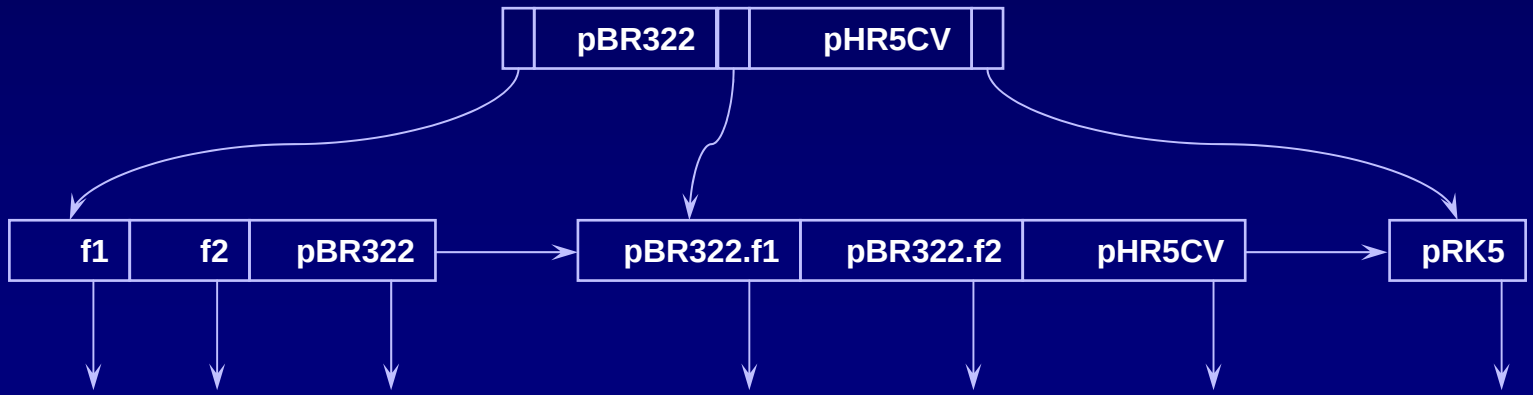
Access is by value comparison:

- **For value V:**
  - **if V <= left value, move to the left lower node**
  - **if left value < V <= right value, move to the middle lower node**
  - **if V > right value, move to the right lower node**

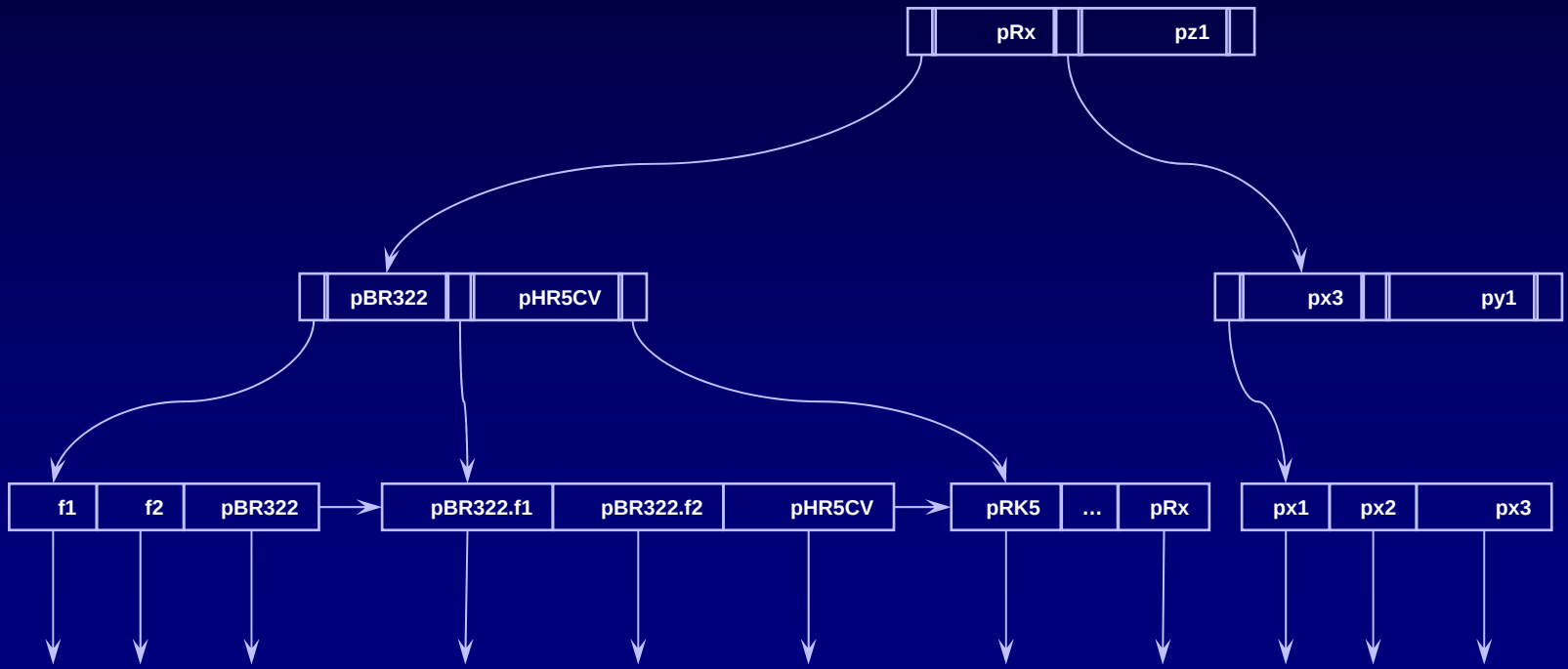
Data Access Methods - BTree

Example: find pBR322.f2 assuming a Btree index on fragment name

- **pBR322.f2 > pBR322 and <= pHR5CV**
  - **we take the middle node, which contains pBR322.f2**
- **If there are more layers, continue repeating the algorithm until you get to the sequence set**



Data Access Methods - BTree



Data Access Methods

There are many other indexing techniques

Indexing can substantially improve access times

Deciding *what* field to index on depends on usage patterns

You can have multiple indices, but that substantially increases insert time and space requirements

Data Access Methods

Questions?

Recommended Reading:

- Knuth, D. E. The Art of Computer Programming, Volume III: Sorting and Searching. Reading, Mass.: Addison-Wesley (1973)

Types of Databases

Flat-file
Hierarchical
Network
Relational
Object
Object-Relational

Flat-File Databases

No database-enforced (or provided) linkage between records

Excellent for small or special-purpose databases

Might include support for single or multiple indexes

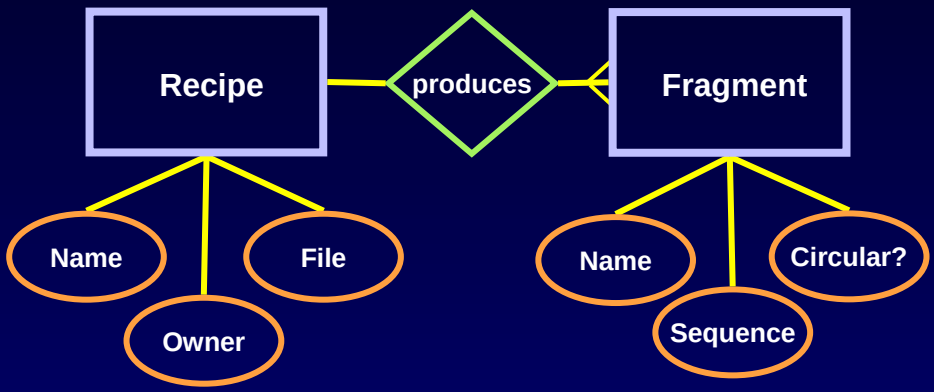
Major feature: ease of use (Filemaker, Access)

Major drawback: scalability & flexibility

e.g.:

- ndbm
- Berkeley DB (*Sleepycat DB)*
- vi, grep, sed
- FileMaker
- Access

Hierarchical Databases



- Relationship between Recipe and Fragment is one-to-many (master-detail)
- Assume two recipes: r1.cr and r2.cr
- r1.cr produces 2 plasmids and 1 fragment:
  - r1.p1, r1.p2, and r1.f1
- r2.cr produces 2 fragments:
  - r2.f1, and r2.f2

Hierarchical Databases

Master

ID	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

Detail

ID	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE
F3	r1.f1	ATCG...	FALSE

ID	NAME	SEQUENCE	CIRCULAR
F4	r2.f1	ATCG...	FALSE
F5	r2.f2	ATCG...	FALSE

Hierarchical Databases

Database provides explicit master-detail support

Ideal for many business applications

Restricted to a strict hierarchy

Queries down the hierarchy are very efficient

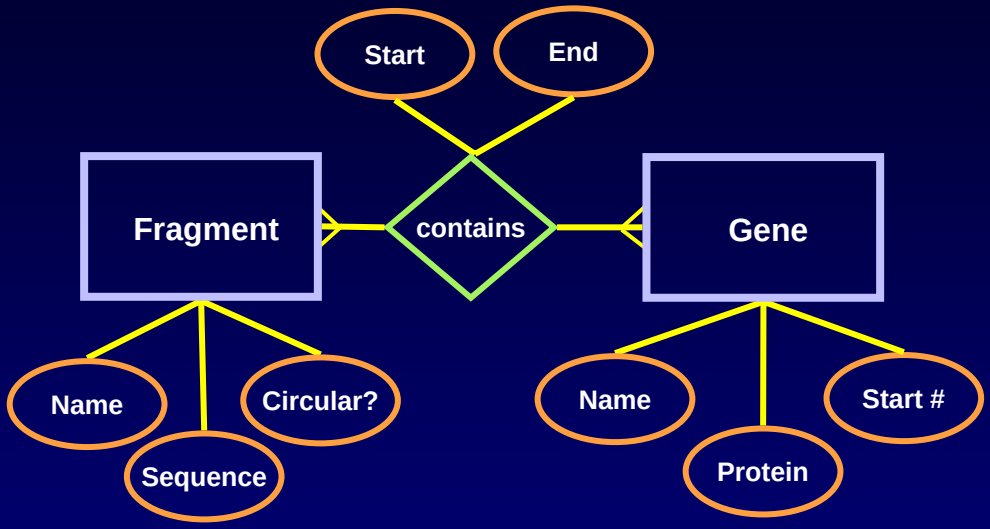
Any other queries are very expensive

e.g.

- IMS

What about many-to-many relationships?

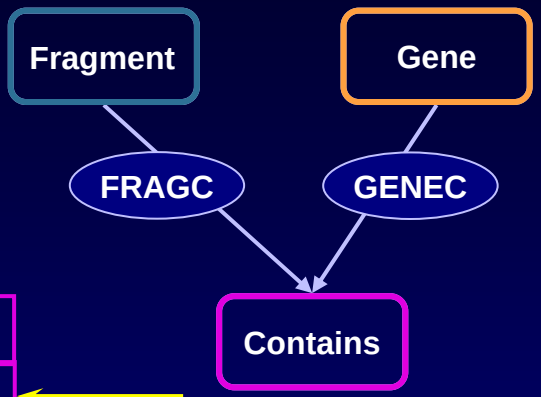
Networked Databases



- Fragment and Gene have a many-to-many relationship
- Not represented well by hierarchical databases

Networked Databases

ID	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE



ID	START	END
C1	15	250
C2	4300	3400
C3	115	350
C4	4400	3300

ID	NAME	PROTEIN	START #
G1	AMP	MAKK...	-5
G2	TET	MYAK...	-10

Networked Databases

Based on set theory

Database provides explicit linkage support

Very significant design costs

Queries along the connection path are very efficient

Any other queries are very expensive

e.g.

- CODASYL

What if I want to “discover” other relationships?

Relational Databases

Based on *relational views* (tables)

Associations are based on data values, not expressed linkages

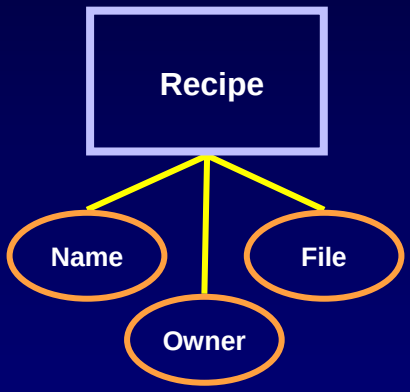
***All* data is expressed in tables**

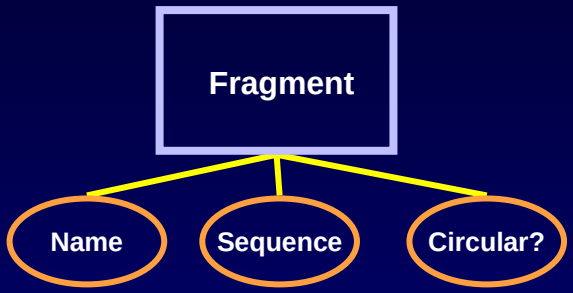
Terminology:

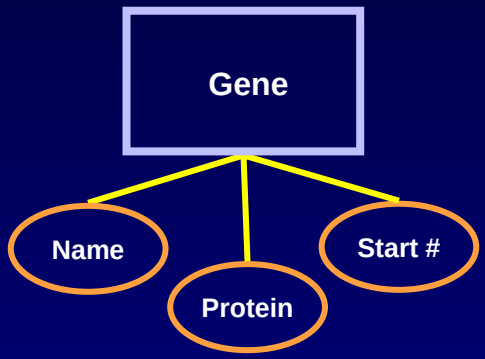
- **Rows are called *tuples***
- **Columns (*attributes*) are of a common domain (type)**

Relational Databases

Start by defining tables for our entities:







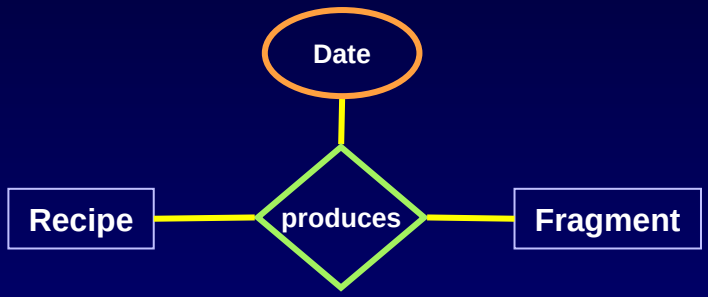
FRAG	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE
F3	r2.f1	ATCG...	FALSE

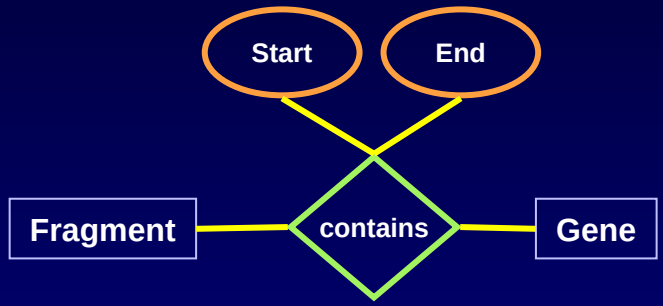
RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

ID	NAME	PROTEIN	START #
G1	AMP	MAKK...	-5
G2	TET	MYAK...	-10
G3	NGF	MYAK...	-1

Relational Databases

Now define tables for relationships, adding attributes for the associations:





RCP	FRAG	DATE
R1	F1	09091985
R1	F2	09091985
R2	F3	10051985

FRAG	GENE	START	END
F1	G1	15	250
F1	G2	4300	3400
F2	G1	115	350
F2	G2	4400	3300
F3	G3	5	500

Relational Algebra

- **Selection**
  - **Selection of tuples based on Boolean criteria**

FRAG

FRAG	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE
F3	r2.f1	ATCG...	FALSE

FRAG where CIRCULAR='TRUE'

FRAG	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE

FRAG where NAME='r2.f1'

FRAG	NAME	SEQUENCE	CIRCULAR
F3	r2.f1	ATCG...	FALSE

Relational Algebra

- **Projection**
  - **Selection of attributes**

GENE

ID	NAME	PROTEIN	START #
G1	AMP	MAKK...	-5
G2	TET	MYAK...	-10
G3	NGF	MYAK...	-1

GENE[NAME]

NAME
AMP
TET
NGF

GENE[NAME,ID]

ID	NAME
G1	AMP
G2	TET
G3	NGF

Relational Algebra

- Join (equijoin)
  - Matrix product of two relations based on equality of attributes with the same domain

Recipe

RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

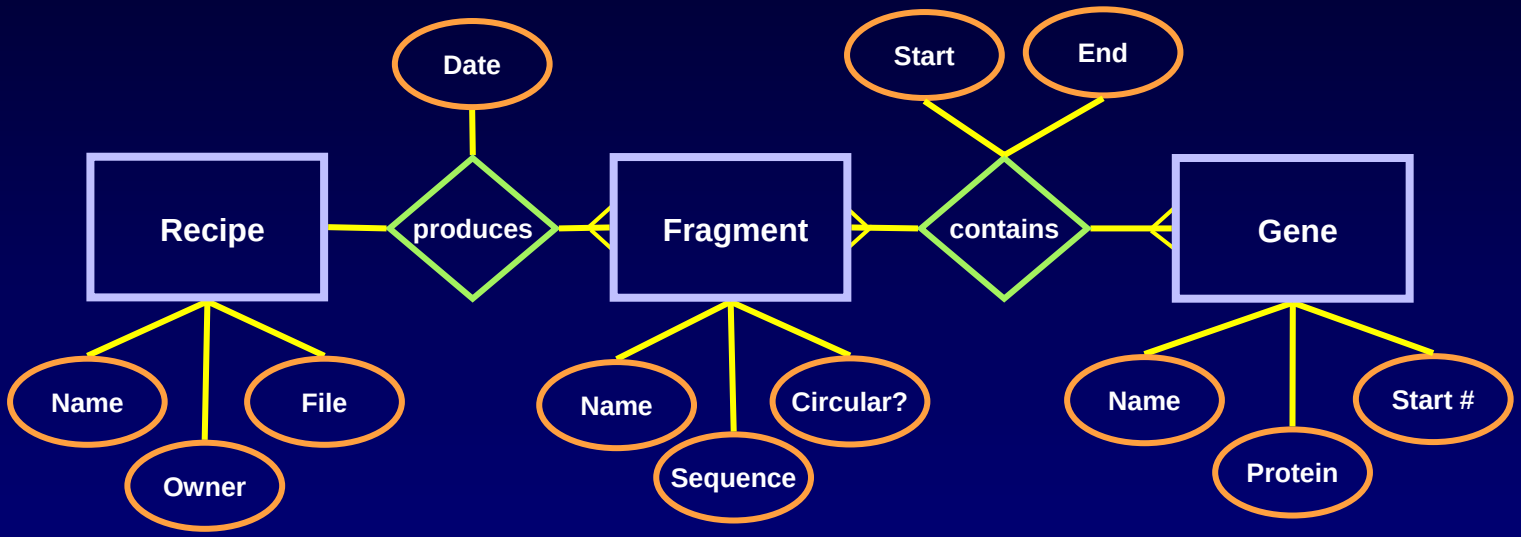
Produces

RCP	FRAG	DATE
R1	F1	09091985
R1	F2	09091985
R2	F3	10051985

(RECIPE times PRODUCES) where RECIPE.RCP = PRODUCES.RCP

RCP	NAME	FILE	OWNER	FRAG	DATE
R1	r1	r1.cr	scooter	F1	09091985
R1	r1	r1.cr	scooter	F2	09091985
R2	r2	r2.cr	ckw	F3	10051985

Example

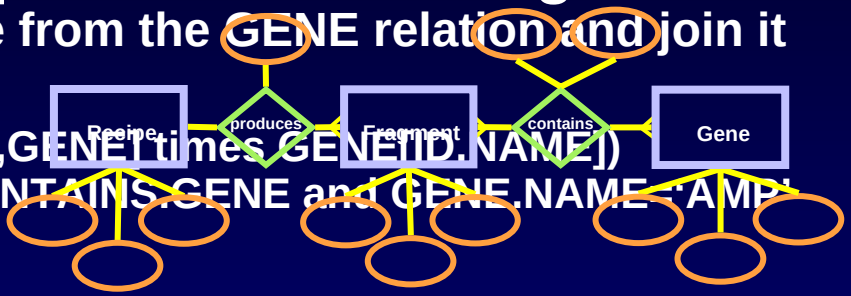

Date
Start
End
Recipe
produces
Fragment
contains
Gene
Name
File
Owner
Name
Circular?
Sequence
Name
Start #
Protein

Relational Algebra

Query: What recipes produce the AMP gene?

- First, select the AMP gene from the GENE relation and join it to CONTAINS

TEMP1 = (CONTAINS[FRAG,GENE] times GENE[ID,NAME])
where GENE.ID=CONTAINS.GENE and GENE.NAME='AMP'



CONTAINS

FRAG	GENE
F1	G1
F1	G2
F2	G1
F2	G2
F3	G3

times

GENE

ID	NAME
G1	AMP
G2	TET
G3	NGF

=

TEMP1

FRAG	GENE	NAME
F1	G1	AMP
F2	G1	AMP

Relational Algebra

Query: What recipes produce the AMP gene?

- **Second, join the result to the PRODUCES relation and select the RCP attribute**

TEMP2 = (TEMP1 join Produces)[RCP]*

TEMP1

FRAG	GENE	NAME
F1	G1	AMP
F2	G1	AMP

join

Produces

RCP	FRAG	DATE
R1	F1	09091985
R1	F2	09091985
R2	F3	10051985

=

TEMP2

RCP
R1
R1

***NOTE: we can use the join shorthand notation because the join field has the same unqualified name: FRAG**

Relational Algebra

Query: What recipes produce the AMP gene?

- **Finally, join the result to the RECIPES relation**

Answer =
(TEMP2 join RECIPES) where TEMP2.RCP=RECIPES.RCP

TEMP2

RCP
R1

join

Recipe

RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

=

Answer

RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter

Structured Query Language (SQL)

ANSI standard syntax for relational algebra

Supported by all major commercial relational databases

Also supported by many open-source efforts

- **e.g. mysql, perl's DBI/DBD**

Will only cover:

- **INSERT**
- **SELECT**
- **UPDATE**

SQL - INSERT

Inserts data into a table row

SYNOPSIS:

- **insert into "tablename“ (first_column,...last_column) values (first_value,...last_value);**

Example:

- ***insert into* GENE (ID, NAME, PROTEIN, START#) *values* (‘G1’, ‘AMP’, ‘MAKK…’, -5);**

SQL - SELECT

Selects data from relational tables

Key syntax for expressing relational algebra

SYNOPSIS

- **select *column1*[,*column2*] from *table1*[,*table2*]**
  [where "conditions"]
  [group by "column-list"]
  [having "conditions]
  [order by "column-list" [ASC | DESC]]

SELECT - Selection

- **Selection**
  - **Selection of tuples based on Boolean criteria**

FRAG

FRAG	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE
F3	r2.f1	ATCG...	FALSE

select * from FRAG where CIRCULAR='TRUE';

FRAG	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE

select * from FRAG where NAME='r2.f1';

FRAG	NAME	SEQUENCE	CIRCULAR
F3	r2.f1	ATCG...	FALSE

SELECT - Projection

- **Projection**
  - **Selection of attributes**

GENE

ID	NAME	PROTEIN	START #
G1	AMP	MAKK...	-5
G2	TET	MYAK...	-10
G3	NGF	MYAK...	-1

select NAME from GENE;

NAME
AMP
TET
NGF

select ID,NAME from GENE;

ID	NAME
G1	AMP
G2	TET
G3	NGF

SELECT - Equijoin

- Join (equijoin)
  - Matrix product of two relations based on equality of attributes with the same domain

Recipe

RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

Produces

RCP	FRAG	DATE
R1	F1	09091985
R1	F2	09091985
R2	F3	10051985

```
select Recipe.RCP, Recipe.Name, Recipe.FILE, Recipe.OWNER, Produces.FRAG,
    Produces.DATE  from Recipe, Produces where RECIPE.RCP = PRODUCES.RCP;
```

RCP	NAME	FILE	OWNER	FRAG	DATE
R1	r1	r1.cr	scooter	F1	09091985
R1	r1	r1.cr	scooter	F2	09091985
R2	r2	r2.cr	ckw	F3	10051985

SELECT – Query Example

Query: What recipes produce the AMP gene?

- First, select the AMP gene from the GENE relation and join it to CONTAINS
  - create table TEMP1 as select CONTAINS.FRAG,CONTAINS.GENE,CONTAINS.NAME from CONTAINS, GENE where GENE.ID=CONTAINS.GENE AND GENE.NAME='AMP';

CONTAINS

FRAG	GENE
F1	G1
F1	G2
F2	G1
F2	G2
F3	G3

times

GENE

ID	NAME
G1	AMP
G2	TET
G3	NGF

=

TEMP1

FRAG	GENE	NAME
F1	G1	AMP
F2	G1	AMP

SELECT – Query Example 2

Query: What recipes produce the AMP gene?

- **Second, join the result to the PRODUCES relation and select the RCP attribute**
  - **create table TEMP2 as select Produces.RCP from TEMP1,Produces where TEMP1.FRAG=Produces.FRAG;**

TEMP1

FRAG	GENE	NAME
F1	G1	AMP
F2	G1	AMP

join

Produces

RCP	FRAG	DATE
R1	F1	09091985
R1	F2	09091985
R2	F3	10051985

=

TEMP2

RCP
R1
R1

SELECT – Query Example 3

Query: What recipes produce the AMP gene?

- Finally, join the result to the RECIPES relation
  - select Recipe.RCP, Recipe.NAME, Recipe.FILE, Recipe.OWNER from TEMP2, Recipe where TEMP2.RCP = Recipe.RCP;

TEMP2

RCP
R1

join

Recipe

RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

=

Answer

RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter

SQL - UPDATE

Updates data in a database

SYNOPSIS:

- **update "tablename"**
  set "columnname" = "newvalue"[,"nextcolumn" = "newvalue2"...]
  where "columnname" OPERATOR "value" [and|or "column" OPERATOR "value"];

Example:

- ***update* GENE *set* NAME='AMP' *where* ID='G1';**

SQL - References

Good intro tutorial

- On-line SQL Tutorial
  - http://www.sqlcourse.com/
- On-line SQL Tutorial 2
  - http://www.sqlcourse2.com/

Another good intro

- A Gentle Introduction to SQL
  - http://www.dcs.napier.ac.uk/~andrew/sql/

Relational Databases

Foundation of most production databases

Based on relational calculus and relational algebra

Allows ad-hoc query capability across record types

Supports a standard query language (SQL)

Can support either hierarchical or network models

Attributes are limited to basic types

e.g.

- Oracle
- Informix

What if I have more complicated data types?

Object-Relational Databases

Essentially an extension of the relational database model

Preserves the tabular (relational) organization of the data

Allows developers to define more complex data types (User Defined Types, UDTs)

No support for encapsulation or inheritance

Some support for methods is provided (User Defined Functions, UDFs)

SQL object extensions already standardized (SQL3)

e.g.

- Informix Universal Server
- Oracle 8/9

Object Databases

Provides persistent storage of objects

Most useful in conjunction with object-based applications

Primarily a programmer's tool, although vendors are providing SQL3 and ODBC interfaces

e.g.

- Objectivity

Types of Databases

Questions?

Recommended Reading:

- **Date, C.J. An Introduction to Database Systems. Reading, Mass.: Addison-Wesley (1981)**
- **Codd, E.F. A Relational Model of Data for Large Shared Data Banks. *CACM* 13, No. 6 (June 1970)**

Uses of Databases

…or why do I care about this stuff?

Three major computing issues in bioinformatics:

- **Computation**
  - **algorithms, analysis, simulation, etc.**
- **Visualization**
  - **seeing all of the data**
- **Data management**
  - **storing and manipulating all of the data**

Databases at Genentech (today)

Relational databases used for:

- Tracking clones
- Assay results
- Preclinical data
- Clinical data
- Genomic data
- Manufacturing data

Flat-file databases used for:

- Workgroup information (Filemaker)
- Similarity searching (inverted index)
- Genomic analysis

Three-tier (actually, *n*-tier)

- UI
- “Business” Logic
- Data store

Databases at Genentech

Heavy use of the Web as UI

- Browser as the universal client
- Dynamic web pages using JSP/Servlets

Distributed Objects

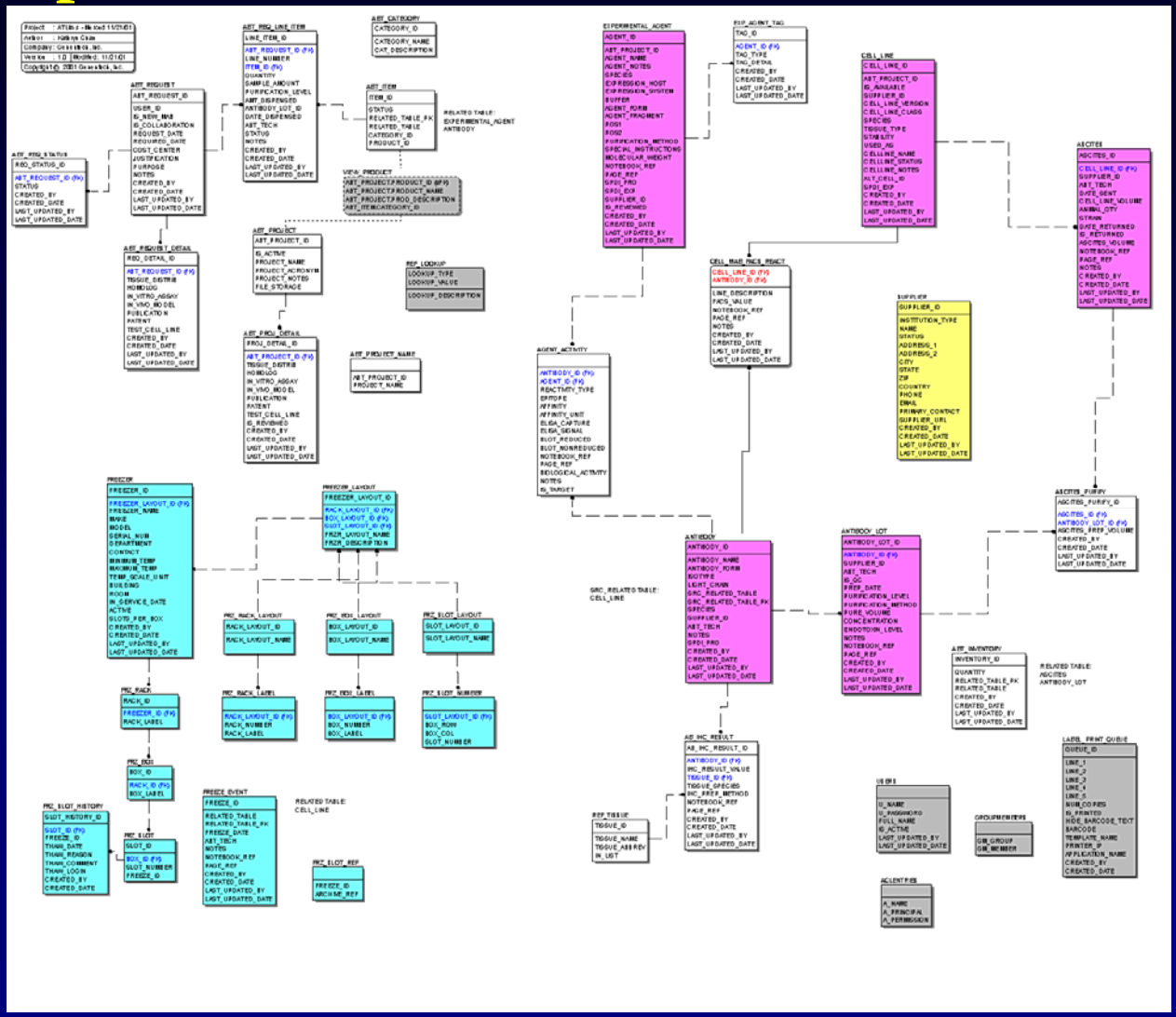
- Enterprise Java Beans

Continued use of special-purpose databases

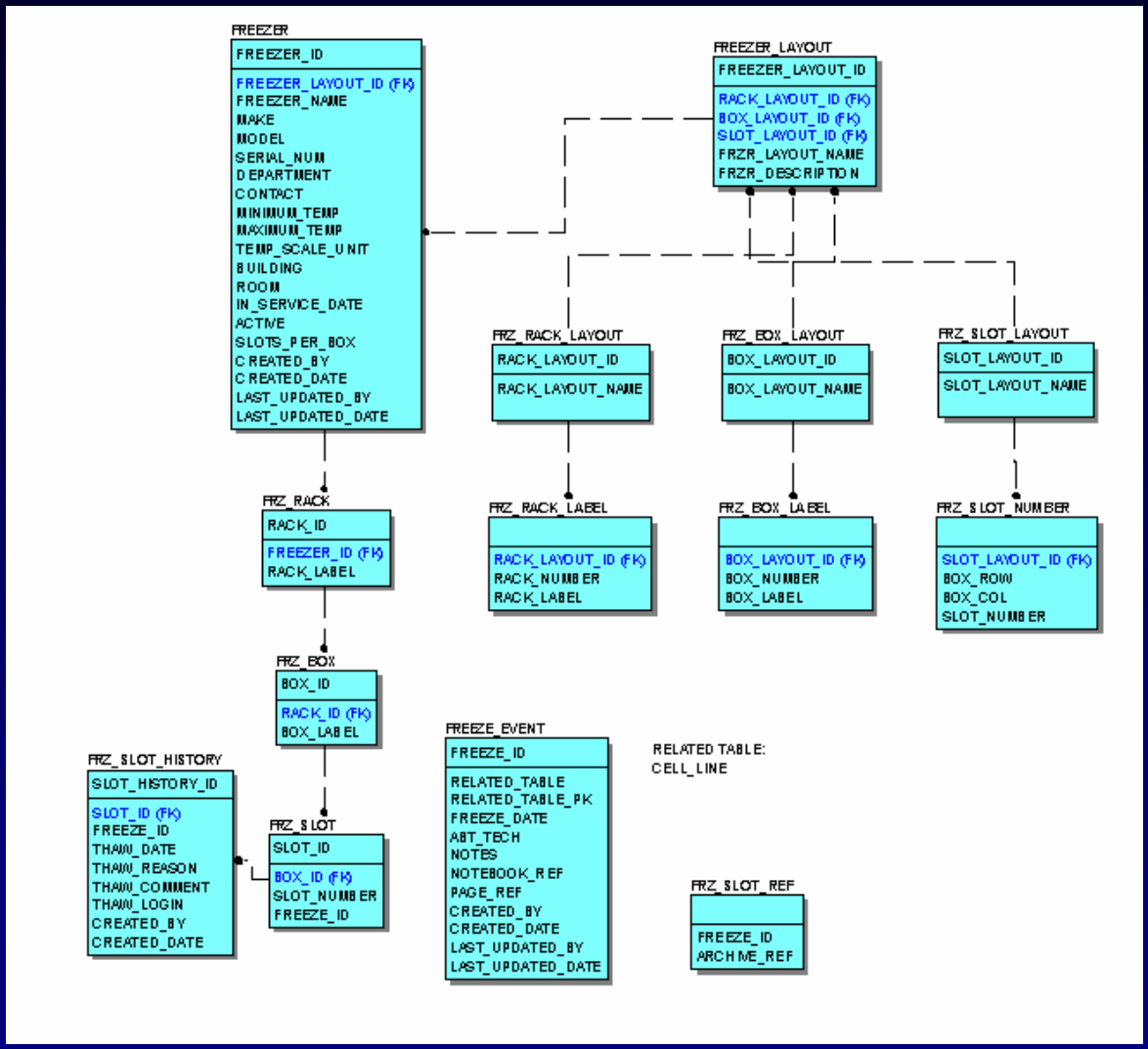
Looking closely at object-relational databases

- Abstract data types
- Java-based “methods”

Example -- ATLims



Example -- ATLims



Example -- ATLims

```
-- TABLE: FRZ_BOX
CREATE TABLE FRZ_BOX(
    BOX_ID        NUMBER(10, 0)    NOT NULL,
    RACK_ID       NUMBER(10, 0)    NOT NULL,
    BOX_LABEL     VARCHAR2(10)     NOT NULL,
    CONSTRAINT FRZ_BOX_PK PRIMARY KEY (BOX_ID)
)
;
-- TABLE: FRZ_BOX_LABEL
CREATE TABLE FRZ_BOX_LABEL(
    BOX_LAYOUT_ID    NUMBER(10, 0)    NOT NULL,
    BOX_NUMBER       NUMBER(5, 0)     NOT NULL,
    BOX_LABEL        VARCHAR2(10)     NOT NULL
)
;
-- TABLE: FRZ_BOX_LAYOUT
CREATE TABLE FRZ_BOX_LAYOUT(
    BOX_LAYOUT_ID      NUMBER(10, 0)    NOT NULL,
    BOX_LAYOUT_NAME    VARCHAR2(30)     NOT NULL,
    CONSTRAINT FRZ_BOX_LAYOUT_PK PRIMARY KEY (BOX_LAYOUT_ID)
)
;
-- TABLE: FREEZE_EVENT
CREATE TABLE FREEZE_EVENT(

    FREEZE_ID           NUMBER(10, 0)    NOT NULL,
    RELATED_TABLE       VARCHAR2(30),
    RELATED_TABLE_PK    NUMBER(10, 0),
    FREEZE_DATE         DATE,
    ABT_TECH            VARCHAR2(32),
    NOTES               VARCHAR2(255),
    NOTEBOOK_REF        NUMBER(10, 0),
    PAGE_REF            VARCHAR2(10),
    CREATED_BY          VARCHAR2(45)    DEFAULT 'admin' NOT NULL,
    CREATED_DATE        DATE            DEFAULT SYSDATE NOT NULL,
    LAST_UPDATED_BY     VARCHAR2(45)    DEFAULT 'admin' NOT NULL,
    LAST_UPDATED_DATE   DATE            DEFAULT SYSDATE NOT NULL,
    CONSTRAINT FREEZE_EVENT_PK PRIMARY KEY (FREEZE_ID)
)
;
```

```
-- TABLE: FREEZER
CREATE TABLE FREEZER(
    FREEZER_ID          NUMBER(10, 0)    NOT NULL,
    FREEZER_LAYOUT_ID   NUMBER(10, 0)    NOT NULL,
    FREEZER_NAME        VARCHAR2(30)     NOT NULL,
    MAKE                VARCHAR2(30),
    MODEL               VARCHAR2(30),
    SERIAL_NUM          VARCHAR2(30),
    DEPARTMENT          VARCHAR2(50),
    CONTACT             VARCHAR2(30),
    MINIMUM_TEMP        VARCHAR2(10),
    MAXIMUM_TEMP        VARCHAR2(10),
    TEMP_SCALE_UNIT     VARCHAR2(1),
    BUILDING            VARCHAR2(20),
    ROOM                VARCHAR2(10),
    IN_SERVICE_DATE     DATE,
    ACTIVE              VARCHAR2(1)     NOT NULL,
    SLOTS_PER_BOX       NUMBER(10,0),
    CREATED_BY          VARCHAR2(45)    NOT NULL,
    CREATED_DATE        DATE            NOT NULL,
    LAST_UPDATED_BY     VARCHAR2(45)    NOT NULL,
    LAST_UPDATED_DATE   DATE            NOT NULL,
    CONSTRAINT FREEZER_PK PRIMARY KEY (FREEZER_ID)
)
;
-- TABLE: FREEZER_LAYOUT
CREATE TABLE FREEZER_LAYOUT(
    FREEZER_LAYOUT_ID   NUMBER(10, 0)    NOT NULL,
    RACK_LAYOUT_ID      NUMBER(10, 0)    NOT NULL,
    BOX_LAYOUT_ID       NUMBER(10, 0)    NOT NULL,
    SLOT_LAYOUT_ID      NUMBER(10, 0)    NOT NULL,
    FRZR_LAYOUT_NAME    VARCHAR2(30)     NOT NULL,
    FRZR_DESCRIPTION    VARCHAR2(255),
    CONSTRAINT FREEZER_LAYOUT_PK PRIMARY KEY (FREEZER_LAYOUT_ID)
)
;
-- TABLE: FRZ_RACK
CREATE TABLE FRZ_RACK(
    RACK_ID       NUMBER(10, 0)    NOT NULL,
    FREEZER_ID    NUMBER(10, 0)    NOT NULL,
    RACK_LABEL    VARCHAR2(10)     NOT NULL,
    CONSTRAINT FRZ_RACK_PK PRIMARY KEY (RACK_ID)
)
;
```

Example -- ATLims

```
--- TABLE: FRZ_RACK_LABEL
CREATE TABLE FRZ_RACK_LABEL(
    RACK_LAYOUT_ID    NUMBER(10, 0)    NOT NULL,
    RACK_NUMBER       NUMBER(5, 0)     NOT NULL,
    RACK_LABEL        VARCHAR2(10)     NOT NULL
)
;
-- TABLE: FRZ_RACK_LAYOUT
CREATE TABLE FRZ_RACK_LAYOUT(
    RACK_LAYOUT_ID      NUMBER(10, 0)    NOT NULL,
    RACK_LAYOUT_NAME    VARCHAR2(30)     NOT NULL,
    CONSTRAINT FRZ_RACK_LAYOUT_PK PRIMARY KEY (RACK_LAYOUT_ID)
)
;
-- TABLE: FRZ_SLOT
CREATE TABLE FRZ_SLOT(
    SLOT_ID       NUMBER(10, 0)    NOT NULL,
    BOX_ID        NUMBER(10, 0)    NOT NULL,
    SLOT_NUMBER   NUMBER(5, 0)     NOT NULL,
    FREEZE_ID     NUMBER(10, 0)    ,
    CONSTRAINT FRZ_SLOT_PK PRIMARY KEY (SLOT_ID)
)
;
-- TABLE: FRZ_SLOT_HISTORY
CREATE TABLE FRZ_SLOT_HISTORY(
    SLOT_HISTORY_ID    NUMBER(10, 0)    NOT NULL,
    SLOT_ID            NUMBER(10, 0)    NOT NULL,
    FREEZE_ID          NUMBER(10, 0)    NOT NULL,
    THAW_DATE          DATE,
    THAW_REASON        VARCHAR2(50),
    THAW_COMMENT       VARCHAR2(250),
    THAW_LOGIN         VARCHAR2(20),
    CREATED_BY         VARCHAR2(45)     NOT NULL,
    CREATED_DATE       DATE             NOT NULL,
    CONSTRAINT FRZ_SLOT_HISTORY_PK PRIMARY KEY (SLOT_HISTORY_ID)
)
;
-- TABLE: FRZ_SLOT_LAYOUT
CREATE TABLE FRZ_SLOT_LAYOUT(
    SLOT_LAYOUT_ID      NUMBER(10, 0)    NOT NULL,
    SLOT_LAYOUT_NAME    VARCHAR2(30)     NOT NULL,
    CONSTRAINT FRZ_SLOT_LAYOUT_PK PRIMARY KEY (SLOT_LAYOUT_ID)
)
;
```

```
--- TABLE: FRZ_SLOT
CREATE TABLE FRZ_SLOT(
    SLOT_ID       NUMBER(10, 0)    NOT NULL,
    BOX_ID        NUMBER(10, 0)    NOT NULL,
    SLOT_NUMBER   NUMBER(5, 0)     NOT NULL,
    FREEZE_ID     NUMBER(10, 0)    ,
    CONSTRAINT FRZ_SLOT_PK PRIMARY KEY (SLOT_ID)
)
;
-- TABLE: FRZ_SLOT_HISTORY
CREATE TABLE FRZ_SLOT_HISTORY(
    SLOT_HISTORY_ID    NUMBER(10, 0)    NOT NULL,
    SLOT_ID            NUMBER(10, 0)    NOT NULL,
    FREEZE_ID          NUMBER(10, 0)    NOT NULL,
    THAW_DATE          DATE,
    THAW_REASON        VARCHAR2(50),
    THAW_COMMENT       VARCHAR2(250),
    THAW_LOGIN         VARCHAR2(20),
    CREATED_BY         VARCHAR2(45)     NOT NULL,
    CREATED_DATE       DATE             NOT NULL,
    CONSTRAINT FRZ_SLOT_HISTORY_PK PRIMARY KEY (SLOT_HISTORY_ID)
)
;
-- TABLE: FRZ_SLOT_LAYOUT
CREATE TABLE FRZ_SLOT_LAYOUT(
    SLOT_LAYOUT_ID      NUMBER(10, 0)    NOT NULL,
    SLOT_LAYOUT_NAME    VARCHAR2(30)     NOT NULL,
    CONSTRAINT FRZ_SLOT_LAYOUT_PK PRIMARY KEY (SLOT_LAYOUT_ID)
)
;
-- TABLE: FRZ_SLOT_NUMBER
CREATE TABLE FRZ_SLOT_NUMBER(
    SLOT_LAYOUT_ID    NUMBER(10, 0)    NOT NULL,
    BOX_ROW           NUMBER(5, 0)     NOT NULL,
    BOX_COL           NUMBER(5, 0)     NOT NULL,
    SLOT_NUMBER       NUMBER(5, 0)     NOT NULL
)
;
-- TABLE: FRZ_SLOT_REF
CREATE TABLE FRZ_SLOT_REF(
    FREEZE_ID         NUMBER(10, 0)    NOT NULL,
    ARCHIVE_REF       VARCHAR2(25)
)
;
```

Example -- ATLims

```
--- TABLE: FRZ_BOX
ALTER TABLE FRZ_BOX ADD CONSTRAINT FRZ_BOX_C1
   FOREIGN KEY (RACK_ID)
   REFERENCES FRZ_RACK(RACK_ID)
;
-- TABLE: FRZ_BOX_LABEL
ALTER TABLE FRZ_BOX_LABEL ADD CONSTRAINT FRZ_BOX_LABEL_C1
   FOREIGN KEY (BOX_LAYOUT_ID)
   REFERENCES FRZ_BOX_LAYOUT(BOX_LAYOUT_ID)
;
-- TABLE: FRZ_BOX_LAYOUT (NEW)
ALTER TABLE FRZ_BOX_LAYOUT ADD CONSTRAINT FRZ_BOX_LAYOUT_CUK1
   UNIQUE (BOX_LAYOUT_NAME)
   ;
-- TABLE: FREEZER
ALTER TABLE FREEZER ADD CONSTRAINT FREEZER_C1
   FOREIGN KEY (FREEZER_LAYOUT_ID)
   REFERENCES FREEZER_LAYOUT(FREEZER_LAYOUT_ID)
;
ALTER TABLE FREEZER ADD CONSTRAINT FREEZER_CUK1
   UNIQUE (FREEZER_NAME)
;
-- TABLE: FREEZER_LAYOUT
ALTER TABLE FREEZER_LAYOUT ADD CONSTRAINT FREEZER_LAYOUT_C1
   FOREIGN KEY (SLOT_LAYOUT_ID)
   REFERENCES FRZ_SLOT_LAYOUT(SLOT_LAYOUT_ID)
   ;
ALTER TABLE FREEZER_LAYOUT ADD CONSTRAINT FREEZER_LAYOUT_C2
   FOREIGN KEY (RACK_LAYOUT_ID)
   REFERENCES FRZ_RACK_LAYOUT(RACK_LAYOUT_ID)
;
ALTER TABLE FREEZER_LAYOUT ADD CONSTRAINT FREEZER_LAYOUT_C3
   FOREIGN KEY (BOX_LAYOUT_ID)
   REFERENCES FRZ_BOX_LAYOUT(BOX_LAYOUT_ID)
;
ALTER TABLE FREEZER_LAYOUT ADD CONSTRAINT FREEZER_LAYOUT_CUK1
   UNIQUE (FRZR_LAYOUT_NAME)
;
-- TABLE: FRZ_RACK
ALTER TABLE FRZ_RACK ADD CONSTRAINT FRZ_RACK_C1
   FOREIGN KEY (FREEZER_ID)
   REFERENCES FREEZER(FREEZER_ID)
;
-- TABLE: FRZ_RACK_LABEL
ALTER TABLE FRZ_RACK_LABEL ADD CONSTRAINT FRZ_RACK_LABEL_C1
   FOREIGN KEY (RACK_LAYOUT_ID)
   REFERENCES FRZ_RACK_LAYOUT(RACK_LAYOUT_ID)
;
-- TABLE: FRZ_SLOT
ALTER TABLE FRZ_SLOT ADD CONSTRAINT FRZ_SLOT_C1
   FOREIGN KEY (BOX_ID)
   REFERENCES FRZ_BOX(BOX_ID)
;
-- TABLE: FRZ_SLOT_HISTORY
ALTER TABLE FRZ_SLOT_HISTORY ADD CONSTRAINT FRZ_SLOT_HISTORY_C1
   FOREIGN KEY (SLOT_ID)
   REFERENCES FRZ_SLOT(SLOT_ID)
;
-- TABLE: FRZ_SLOT_NUMBER
ALTER TABLE FRZ_SLOT_NUMBER ADD CONSTRAINT FRZ_SLOT_NUMBER_C1
   FOREIGN KEY (SLOT_LAYOUT_ID)
   REFERENCES FRZ_SLOT_LAYOUT(SLOT_LAYOUT_ID)
;
-- TABLE: FRZ_SLOT_LAYOUT

ALTER TABLE FRZ_SLOT_LAYOUT ADD CONSTRAINT SLOT_LAYOUT_UNQ1
   UNIQUE (SLOT_LAYOUT_NAME)
;
exit;
```

Assignment

Complete the ER diagram given as an example. Include *at least* the following entities:

- **Recipe**
- **Fragment**
- **Element (includes genes, promoters, ribosome binding sites, etc.)**
- **Protein**
- **Owner (i.e. scientist)**
- **Restriction site**

Assume that fragments can be used in more than one recipe and can be produced by more than one recipe.

Also assume that elements can be contained in more than one fragment

Questions?

Contact information:

Scooter Morris
Genentech, Inc.
1 DNA Way
South San Francisco, CA 94080

scooter@gene.com