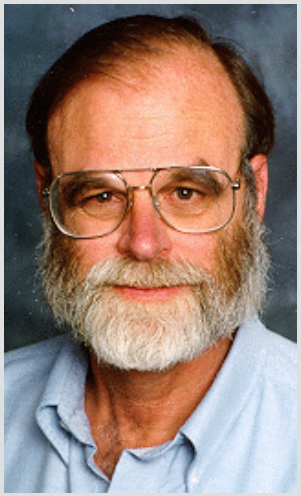
SCIENTIFIC DATA MANAGEMENT

CS561-SPRING 2012
WPI, MOHAMED ELTABAKH



While the commercial world has standardized on the relational data model and SQL, no single standard or tool has critical mass in the scientific community. There are many parallel and competing efforts to build these tool suites – at least one per discipline. Data interchange outside each group is problematic. In the next decade, as data interchange among scientific disciplines becomes increasingly important, a common HDF-like format and package for all the sciences will likely emerge.

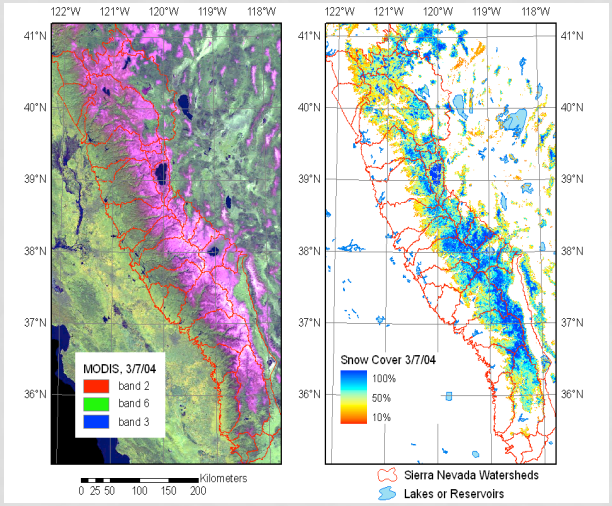
SCIENTIFIC DATA MANAGEMENT

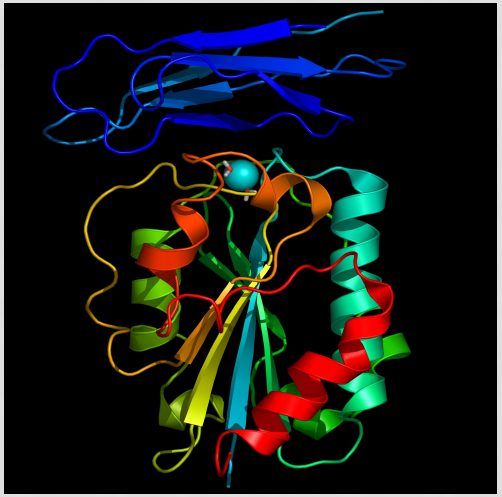
- Scientific instruments and computer simulations are creating vast amounts of scientific data

- **Scientific Domains**
  - Biology
  - Chemistry
  - Physics
  - Astronomy
  - Earth science
  - …

REQUIREMENTS I

- **Complex Data Types**
  - Arrays, sequences, images, time series, etc.
  - Efficient support for these data types
    - Storage, indexing, searching, and complex scientific op





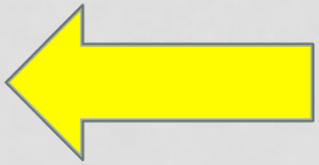
REQUIREMENTS II

- **New Data-Analysis Methods**
  - Scientific tools and algorithms are complex
    - N^2 or N^3 of the data size (N)
  - Need for faster and parallel algorithms
- **Science Centers**
  - Too hard to move data (too large)
  - Move the programs and tools to the data
  - Science centers provide access to data and tools

REQUIREMENTS III

- **Metadata Describing Data**
  - In Scientific applications, data by itself is not enough
  - Metadata can describe
    - How the data is generated/derived **(provenance)**
    - Comments about the data **(annotation)**
    - Structure of data, e.g., column names, measurement units, etc.
- **Uncertainty Processing**
  - Most scientific data have uncertain value
  - Representation of uncertain data
  - Query processing of uncertain values

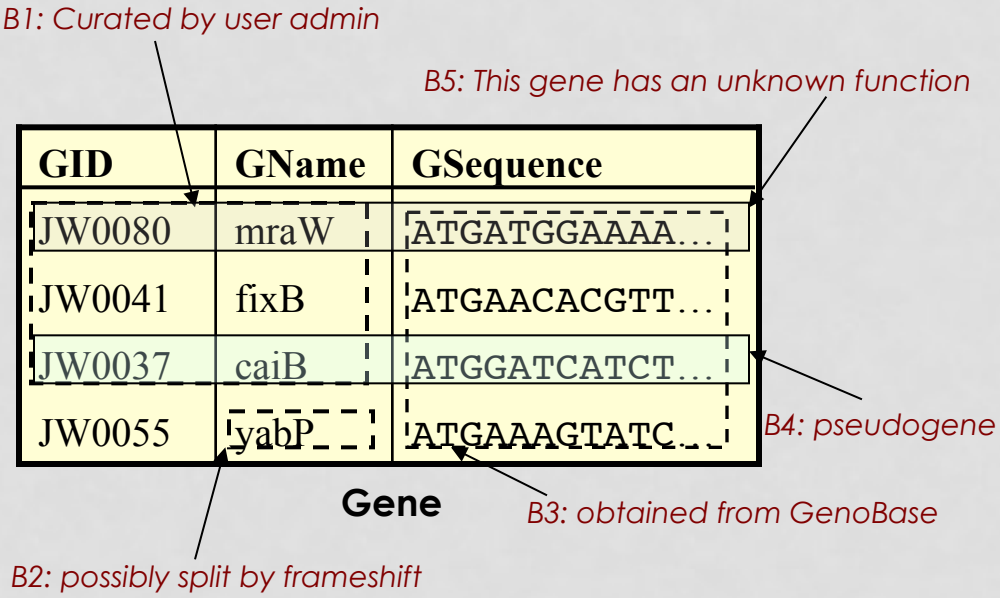
We Will Touch The Following…

- **Annotation Management**

- **Complex Dependencies**
- **SciDB– Array Databases**

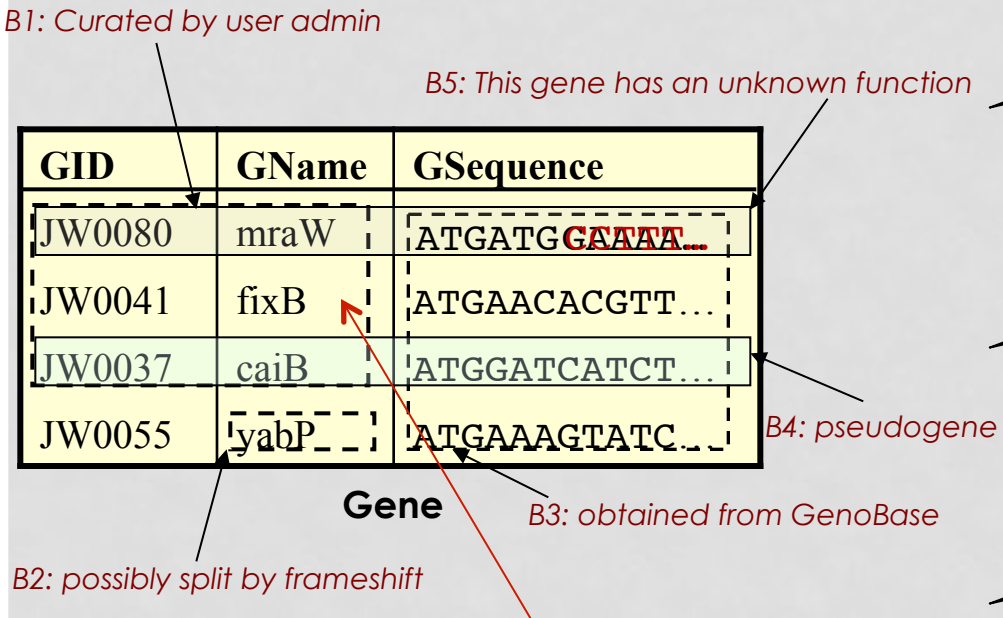
WHY ANNOTATIONS?

- Vital mechanism for sharing knowledge and building an interactive and collaborative environment among database users

- Annotations may represent:
  - Comments, feedbacks, users experiences, Lineage information, etc.



CHALLENGES IN ANNOTATION MANAGEMENT



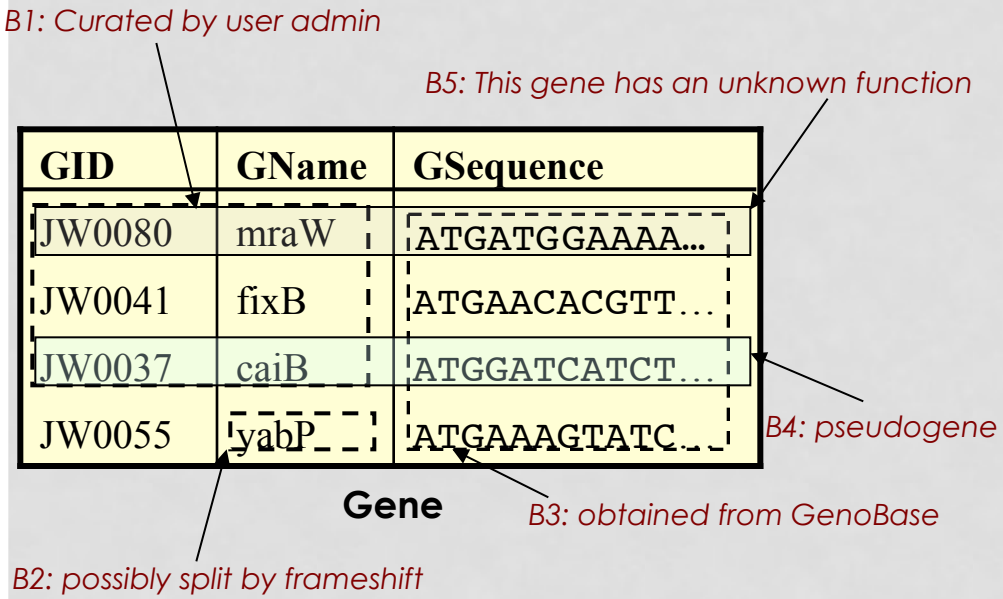
B6: under verification **[Genes with GName like 'fix%']**

Combinatorial nature at various granularities

Different behaviors under DB operations

Query evaluation and annotation propagation

MAIN FEATURES



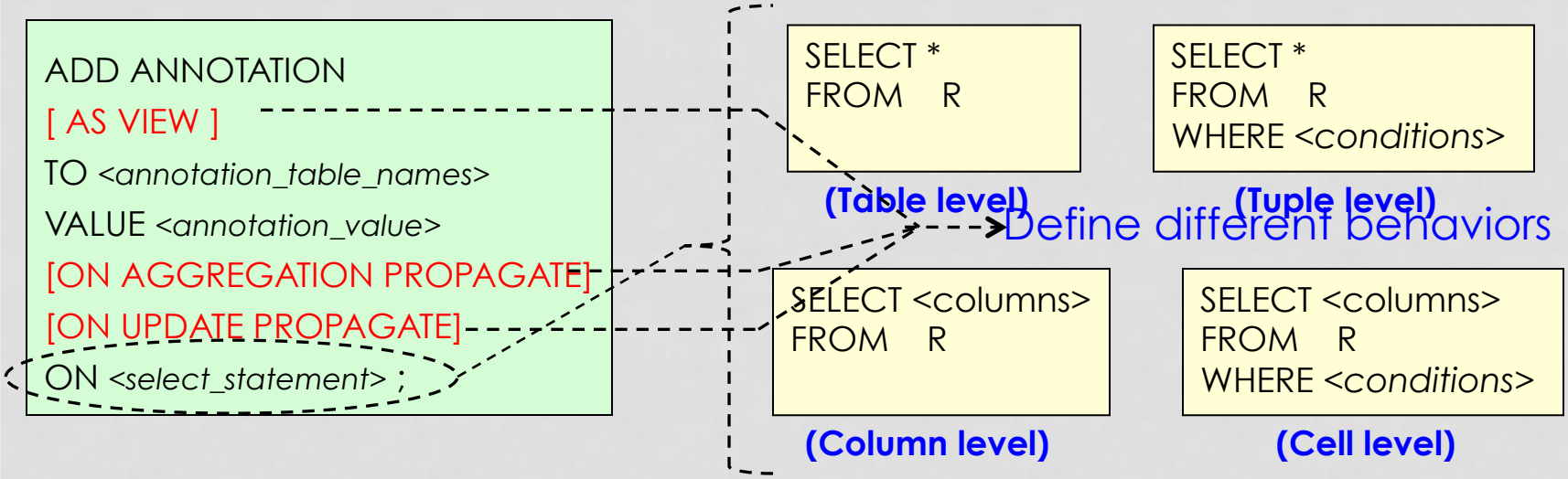
1- Adding Annotations & Defining Behaviors

2- Storage Optimization Techniques

3- Propagating/Querying Annotations

1- ADDING ANNOTATIONS & DEFINING BEHAVIORS

- Declarative and simple mechanisms to annotate the data
- Specify behaviors of annotations under different DB operations



ANNOTATION BEHAVIOR: *AS VIEW*

- ***Snapshot*** vs. ***View*** annotations
  - ***Snapshot*** annotations are evaluated once
  - ***View*** annotations are continuously evaluated

ADD ANNOTATION

VALUE *'under verification'*

ON ***Select Gname, GSequence***
From GENE
Where Gname Like 'fix%' ;

GID	GName	GSequence
JW0080	mraW	ATGATGGAAAA...
JW0041	fixB	ATGAACACGTT…
JW0037	caiB	ATGGATCATCT…
JW0055	yabP	ATGAAAGTATC…
JW0070	fixR	TTTAAAGTAA...

under verification

How to maintain ***view*** annotations Up-to-date?

Adopt several optimization techniques from "*Maintenance of Materialized Views*"

ANNOTATION BEHAVIOR: *ON UPDATE PROPAGATE*

- Annotation behavior under ***update*** operations

ADD ANNOTATION
[AS VIEW]
TO *<annotation_table_names>*
VALUE *<annotation_value>*
[ON AGGREGATION PROPAGATE]
[ON UPDATE PROPAGATE]
ON *<select_statement>* ;

B5: This gene has an unknown function

GID	GName	GSequence
JW0080	mraW	ATGATGGAAAA…
JW0041	fixB	ATGAACACGTT…
JW0037	caiB	ATGGATCATCT…
JW0055	yabP	ATGAAAGTATC…

Gene

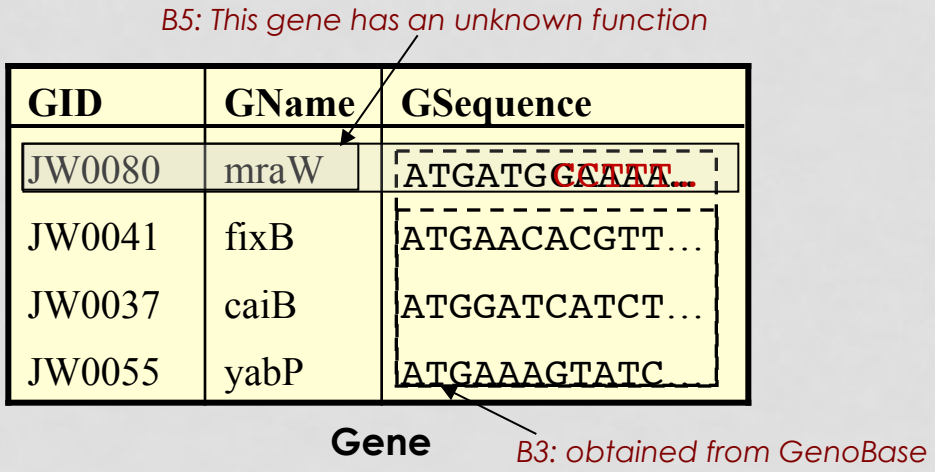
B3: obtained from GenoBase

ANNOTATION BEHAVIOR: *ON UPDATE PROPAGATE*

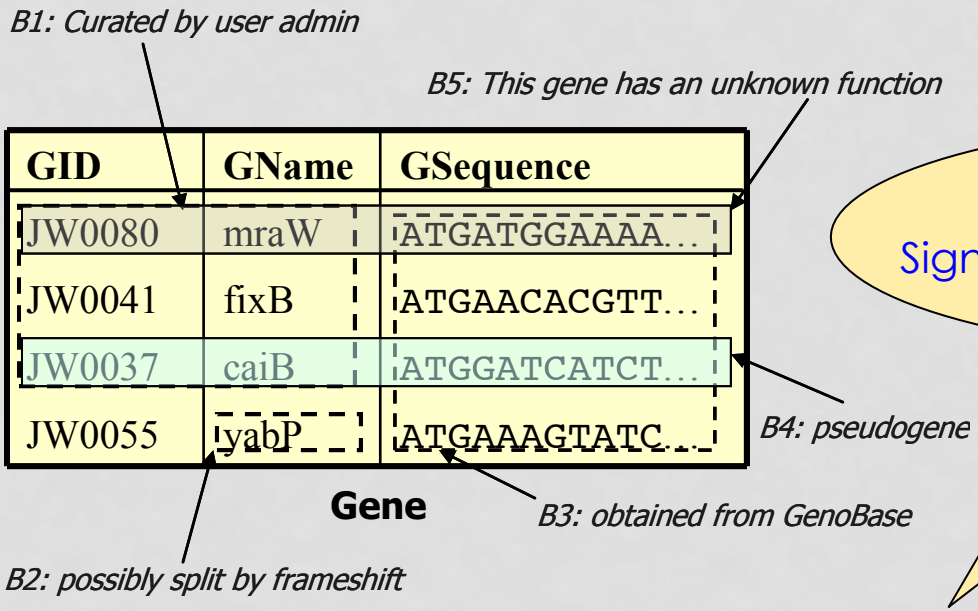
- Annotation behavior under ***update*** operations

```
ADD ANNOTATION
[ AS VIEW ]
TO <annotation_table_names>
VALUE <annotation_value>
[ON AGGREGATION PROPAGATE]

ON <select_statement> ;
```



2- STORAGE OF ANNOTATIONS



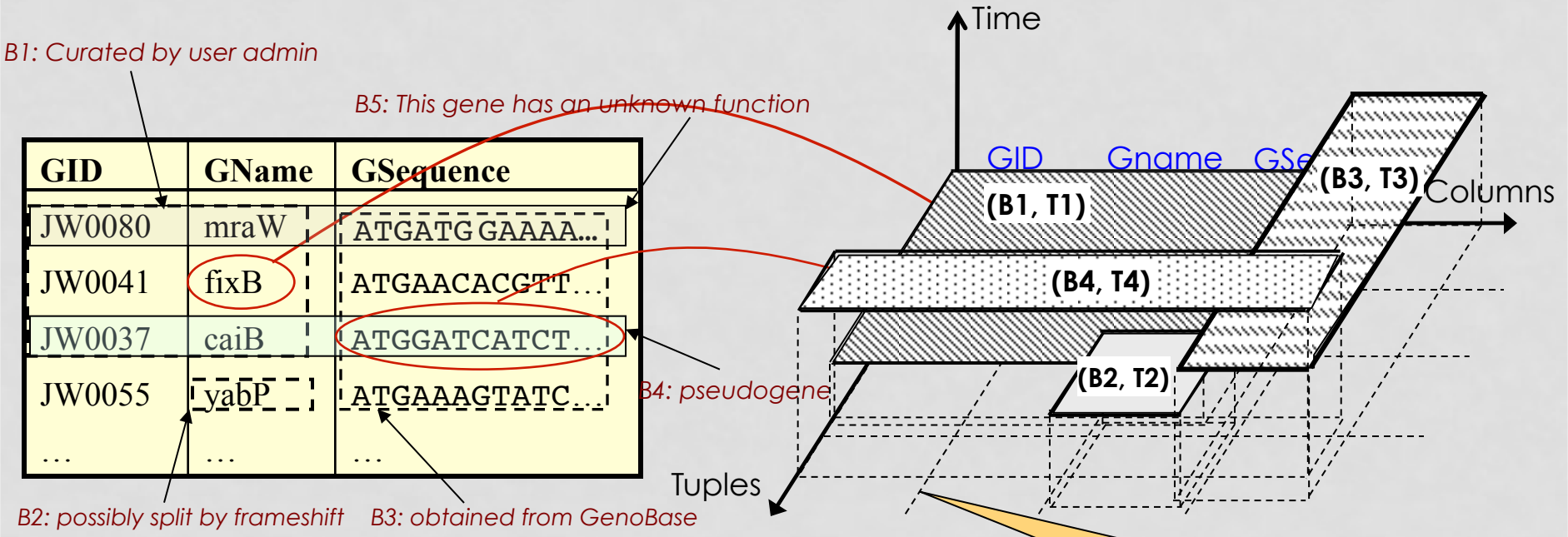
Straightforward Approach

Significant storage and I/O overhead because of the replication

-- N table values
-- 2^N Possible subsets to be annotated!

GID	GID_Ann	GName	GName_Ann	GSequence	GSequence_Ann
JW0080	B1, B5, …	mraW	B1, B5, …	ATGATGGAAAA…	B5, B3, …
JW0041	B1, …	fixB	B1,…	ATGAACACGTT…	B3, …
JW0037	B1, B4, …	caiB	B1, B4, …	ATGGATCATCT…	B3, B4
JW0055		yabP	B2, …	ATGAAAGTATC…	B3, …

COMPRESSED REPRESENTATION OF ANNOTATIONS



- Logical database tables → Two-dimensional space
- Table cells → Points in two-dimensional space
- Annotated cells → Maximum bounded rectangle(s) in three-dimensional space

>> Less storage overhead

>> Less I/O operations at insertion & query time

3- PROPAGATING/QUERYING ANNOTATIONS

- Extended ***Select*** statement to support annotation propagation and querying

Propagates annotations from non-projected columns

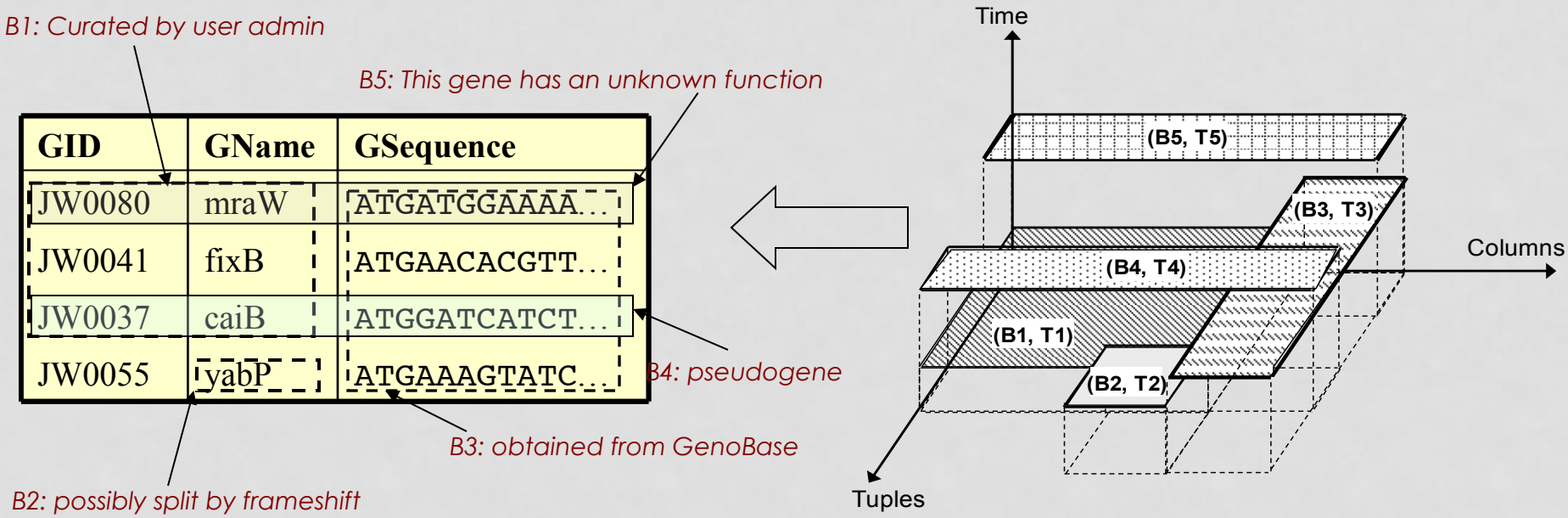
Specifies which annotations to include in the query

```
SELECT [DISTINCT] Ci , Cj , ..., [PROMOTE (Ck, Cm, ...)]
FROM   Relation_name [ANNOTATION(S1, S2, ...)], ...
[WHERE <data_annotation_conditions>]
[GROUP BY <data_columns>
   [HAVING  <data_annotation_condition>]
[ORDER BY <data_columns>]
```

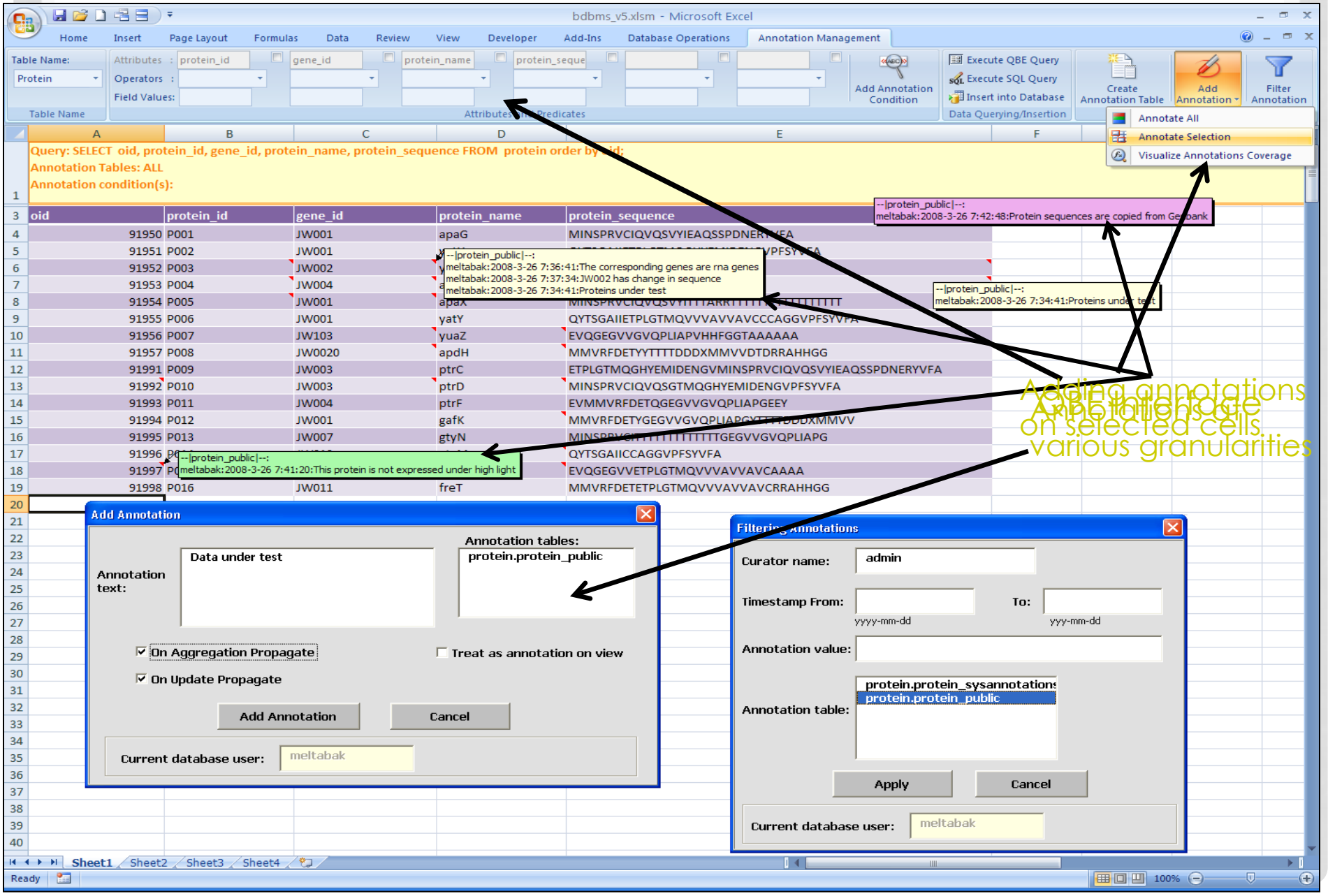
```
SELECT GID, GName, Promote(GSeq)
FROM Gene [Annotation(lab_comments)]
WHERE GName like 'Pho%'
 AND lab_comments.curator = 'Admin';
```

QUERY EXECUTION

- Reverse the mapping from 3D space to logical representation

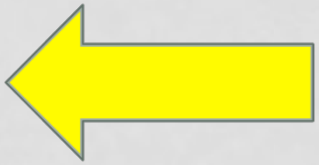


MANAGING ANNOTATIONS VIA GUI



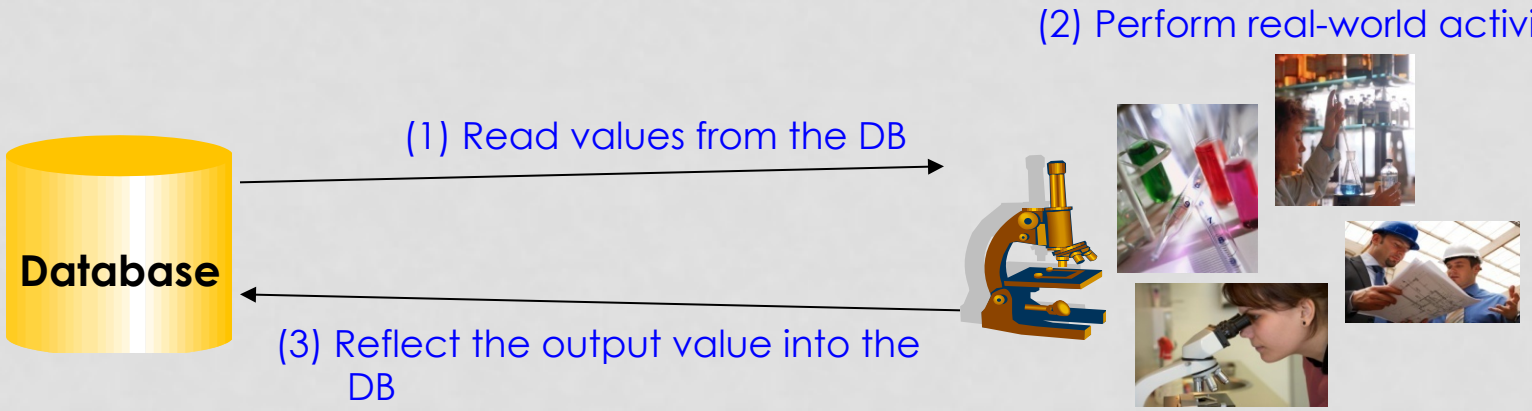
We Will Touch The Following…

- **Annotation Management**
- **Complex Dependencies**
- **SciDB– Array Databases**



COMPLEX DEPENDENCIES

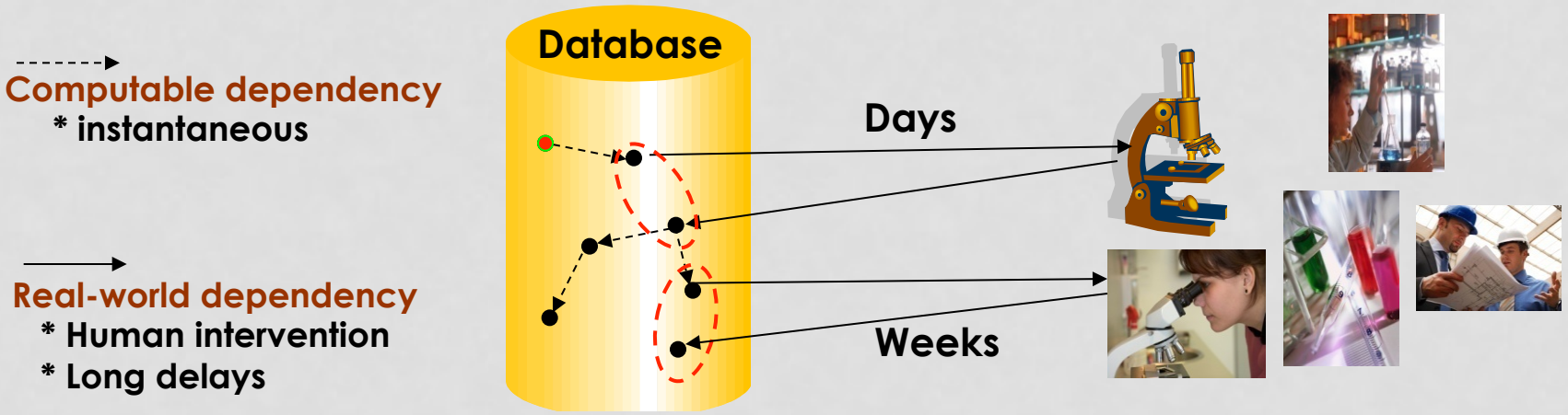
- **The cycle of processing the data is complex**
  - May involve ***Real-world Activities***, e.g. wet-lab experiments, instruments readings, manual measurements, etc.



- Updating an input value to a real-world activity may render the output value ***invalid*** until the activity is re-executed. ***But…***
  - The database system is not aware of the dependency

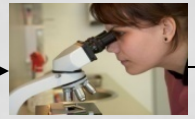
CHALLENGES

- **In typical scenarios, dependencies cascade**



- Between the updates, parts of the database are ***temporarily inconsistent*** **(invalid/outdated)**
  - Data still needs to be available for querying

EXAMPLE QUERY



GID	StartPos	Binding site	GSeq	GDirection	GFunction
JW0013	5130	AATG…	TGCT…		
JW0014	**10916**	TCCA…	GGTT…		
JW0015	21112	GTAA…	**GGCT…**		
JW0018	**31166**	CCGG…	CGTT…		
JW0019	1905	AAAT…	**TGTG…**	+	
JW0012	17404	GTAA…	TTCG…	-	

GENE table

Crucial information is missing that affect:
>> The quality of the answer
>> Any decisions based on the results

Select GID, BindingSite
From Gene
Where GFunction = “F2”;

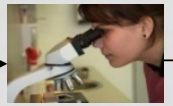
Up-to-date ***Outdated***

GID	BindingSite
JW0014	TCCA…
JW0015	GTAA…

Evaluated based on Up-to-date values

Evaluated based on Outdated values

OBJECTIVE



GID	StartPos	Binding site	GSeq	GDirection	GFunction
JW0013	5130	AATG…	TGCT…	+	F1
JW0014	**10916**	TCCA…	GGTT…	+	F2
JW0015	21112	GTAA…	**GGCT…**	+	F2
JW0018	**31166**	CCGG…	CGTT…	-	F4
JW0019	1905	AAAT…	**TGTG…**	+	F5
JW0012	17404	GTAA…	TTCG…	-	F7

- DBMS needs to be aware of the real-world dependencies
  - Keeps track and enforces the dependencies
  - Maintains the consistency of the data ***(Best effort)***

MAIN FEATURES

1. **Registering Activities and Expressing Dependencies**
2. **Extended Querying Mechanisms**
3. **Systematic Tracking of Outdated Data**

1- REGISTERING ACTIVITIES AND EXPRESSING DEPENDENCIES

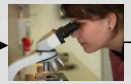
- Registering real-world activities into the database
- Expressing dependencies among the data items on these activities

Create Function <*activity-name*> (<*input-types*>)
Returns <*output-type*> As real-world activity;

Create Table <*R*>
(
<*columns_definitions* >
….
Add Dependency Using <*func_name*>
Source <$T_1.c_1$[, $T_2.c_2$, ...] >
Destination <$R.c_0$>
[Where <*predicates*>]);

Alter Table <*R*>
Add Dependency Using <*func_name*>
Source <$T_1.c_1$[, $T_2.c_2$, ...] >
Destination <$R.c_0$>
[Where <*predicates*>]
[Invalidate Destination] ;

EXAMPLE 1: SINGLE-TABLE DEPENDENCY

Exp1  Exp2 

GID	StartPos	Binding site	GSeq	GDirection	GFunction
JW0013	5130	AATG…	TGCT…	+	F1
JW0014	10910	TCCA…	GGTT…	+	F2
JW0015	21112	GTAA…	GTCT…	+	F2
JW0018	31161	CCGG…	CGTT…	-	F4
JW0019	1905	AAAT…	TGCC…	+	F5
JW0012	17404	GTAA…	TTCG	-	F7

GENE table

> Create Function ***Exp1*** (*int*)
Returns *text* ***As real-world activity***;

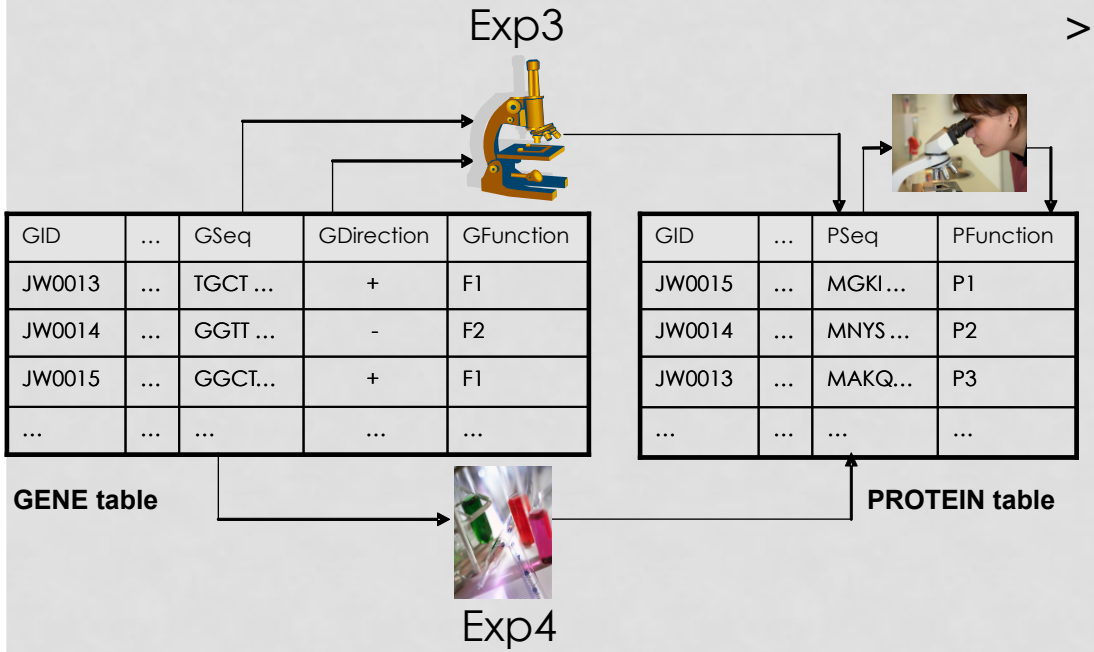
> Create Function ***Exp2*** (*text, char*)
Returns *text* ***As real-world activity***;

> Create Table GENE(
GID text,
StartPos int,
BindingSite text,
GSeq text,
GDirction char,
GFunction text,

ADD Dependency Using *Exp1*
Source *StartPos*
Destination *BindingSite*,

ADD Dependency Using *Exp2*
Source *GSeq, GDirection*
Destination *GFucntion*);

EXAMPLE 2: CROSS-TABLES DEPENDENCY



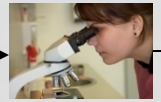

```
> Create Table Protein(
    GID text,
    PSeq text,
    PFunction text,
```

```
ADD Dependency Using Exp3
Source Gene.GSeq, Gene.GDirection
Destination Protein.PSeq
Where Protein.GID = Gene.GID
And Gene.GFunction = 'F1',
```

```
ADD Dependency Using Exp4
Source Gene.GSeq
Destination Protein.PSeq
Where Protein.GID = Gene.GID
And Gene.GDirection = '+');
```

2- EXTENDED QUERYING MECHANISMS

- Reflecting the status of the values in the query results as ***up-to-date*** or ***outdated***
- Evaluating queries on up-to-date data only ***(no false-positive results)***
- Evaluating queries on both up-to-date and outdated data ***(include false-positive results)***



GID	StartPos	Binding site	GSeq	GDirection	GFunction
JW0013	5130	AATG…	TGCT…	+	F1
JW0014	**10916**	TCCA…	GGTT…	+	F2
JW0015	21112	GTAA…	**GGCT…**	+	F2
JW0018	**31166**	CCGG…	CGTT…	-	F4
JW0019	1905	AAAT…	**TGTG…**	+	F5
JW0012	17404	GTAA…	TTCG…	-	F7
JW0120	19803	GTAA…	AATT…	+	F2

EXTENDED QUERY OPERATORS: EXAMPLE

Extended semantics of predicate evaluation

GID	StartPos	Binding site	GSeq	GDirection	GFunction	GFunction = "F2" AND BindingSite= "GTAA..."
JW0013	5130	AATG...	TGCT...	+	F1	F
JW0014	10916	TCCA...	GGTT...	+	F2	-ve
JW0015	21112	GTAA...	GGCT...	+	F2	+ve
JW0018	31166	CCGG...	CGTT...	-	F4	F
JW0019	1905	AAAT...	TGTG...	+	F5	F
JW0012	17404	GTAA...	TTCG...	-	F7	F
JW0120	19803	GTAA...	AATT...	+	F2	T

Potentially false-negative

Potentially false-positive

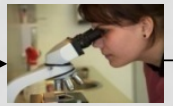
False

True

GENE table

σ BinidingSite='GTAA...' And GFunction='F2' (GENE)

SUMMARY

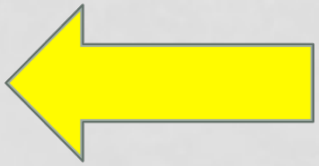


GID	StartPos	Binding site	GSeq	GDirection	GFunction
JW0013	5130	AATG…	TGCT…	+	F1
JW0014	**10916**	TCCA…	GGTT…	+	F2
JW0015	21112	GTAA…	**GGCT…**	+	F2
JW0018	**31166**	CCGG…	CGTT…	-	F4
JW0019	1905	AAAT…	**TGTG…**	+	F5
JW0012	17404	GTAA…	TTCG…	-	F7

Make data updates ***instantly*** available for querying while maintaining the ***consistency*** of the derived data under the presence of ***real-world dependencies***

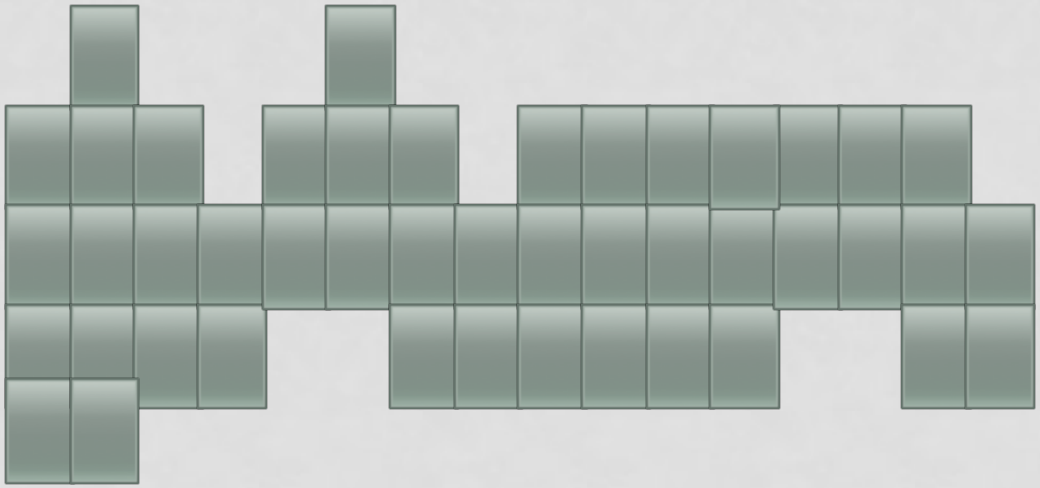
We Will Touch The Following…

- **Annotation Management**
- **Complex Dependencies**
- **SciDB– Array Databases**



SciDB: A Database System for Scientific Data

- **Open-source data base system started in 2008**
- **Team from MIT, Wisconsin, Stanford, and others**
- **Focus on arrays → Array data model**
  - **Makes scientists in many fields (not all) happy**



SciDB Data Model

- **Arrays**
  - Superset of tables (tables with a primary key are a 1-D array)
- **Nested multidimensional arrays**
- **Every cell is a tuple of values**

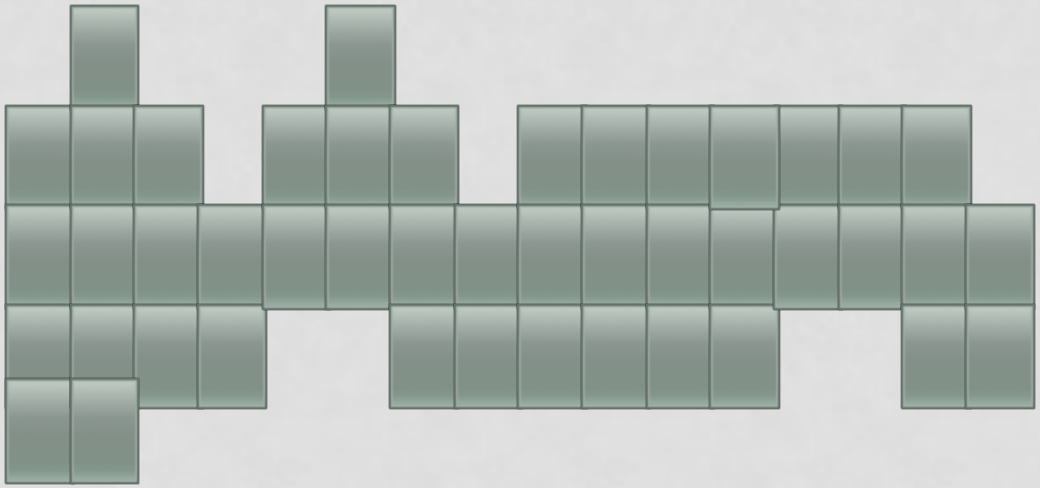
2D Array. Every tuple has two attributes

J \ I	[0]	[1]	[2]	[3]	[4]
[0]	(2, 0.7)	(5, 0.5)	(4, 0.9)	(2, 0.8)	(1, 0.2)
[1]	(5, 0.5)	(3, 0.5)	(5, 0.9)	(5, 0.5)	(5, 0.5)
[2]	(4, 0.3)	(6, 0.1)	(6, 0.5)	(2, 0.1)	(7, 0.4)
[3]	(4, 0.25)	(6, 0.45)	(6, 0.3)	(1, 0.1)	(0, 0.3)
[4]	(6, 0.5)	(1, 0.6)	(5, 0.5)	(2, 0.15)	(2, 0.4)

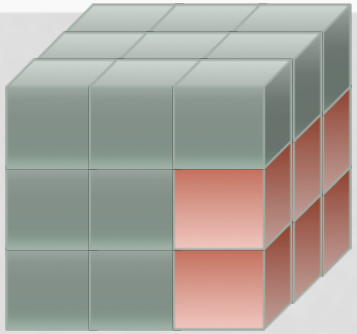
Figure 1: Simple Two Dimensional SciDB Array.

ENHANCED ARRAYS

- Supports irregular shapes
- New operators to change the shape of a given array



DATA STORAGE



- Optimized for both dense and sparse array data
  - Different data storage, compression, and access
- Arrays are “chunked” (in multiple dimensions)
- Chunks are partitioned across a collection of nodes
- Chunks have ‘overlap’ to support neighborhood operations
- Replication provides efficiency and back-up
- Fast access to data sliced along any dimension
  - Without materialized views

EXAMPLE OF STORAGE

J \ I	[0]	[1]	[2]	[3]	[4]
[0]	(2, 0.7)	(5, 0.5)	(4, 0.9)	(2, 0.8)	(1, 0.2)
[1]	(5, 0.5)	(3, 0.5)	(5, 0.9)	(5, 0.5)	(5, 0.5)
[2]	(4, 0.3)	(6, 0.1)	(6, 0.5)	(2, 0.1)	(7, 0.4)
[3]	(4, 0.25)	(6, 0.45)	(6, 0.3)	(1, 0.1)	(0, 0.3)
[4]	(6, 0.5)	(1, 0.6)	(5, 0.5)	(2, 0.15)	(2, 0.4)

Step 1: Vertically partition *attributes* in the *logical array*.

J \ I {A}

2	5	4	2	1
5	3	5	5	5
4	6	6	2	7
4	6	6	1	0
6	1	5	2	2

J \ I {B}

0.7	0.5	0.9	0.8	0.2
0.5	0.5	0.9	0.5	0.5
0.3	0.1	0.5	0.1	0.4
0.25	0.45	0.3	0.1	0.3
0.5	0.6	0.5	0.15	0.4

Step 2: Decompose each attribute array into equal sized, and potentially overlapping, *chunks*.

J \ I $\{A_1\}$

2	5	4
5	3	5
4	6	6

J \ I $\{A_2\}$

4	2	1
5	5	5
6	2	7

J \ I $\{A_3\}$

4	6	6
4	6	6
6	1	5

J \ I $\{A_4\}$

6	2	7
6	1	0
5	2	2

Figure 5: SciDB Storage Manager

Separate attributes first

Divide into overlapping chunks

Distribute these chunks over the cluster nodes

SciDB DDL

```
CREATE ARRAY Test_Array
     < A: integer NULLS,
       B: double,
       C: USER_DEFINED_TYPE >
      [I=0:99999,1000, 10, J=0:99999,1000, 10 ]
      PARTITION OVER ( Node1, Node2, Node3 )
      USING block_cyclic();
```

attribute names

A, B, C

index names

I, J

chunk size

1000

overlap

10

Array Query Language (AQL)

```
SELECT Geo-Mean ( T.B )
FROM Test_Array T
WHERE
     T.I BETWEEN :C1 AND :C2
AND T.J BETWEEN :C3 AND :C4
AND T.A = 10
GROUP BY T.I;
```

User-defined aggregate on an attribute B in array T

Subsample

Filter

Group-by

>> Many new operators for array processing

E.g., Slicing, multiplication, sampling, etc.

>> Many new optimization and indexing techniques