A Database Armada
Data explosion

- World Wide Web
- Digital Libraries
- SkyServer

✓ Computation and storage exhaustive
Current line of defence

- Huge monolithic clusters
- GRIDs
- P2P overlay networks
Why it isn’t sufficient

- Need for autonomous servers (co-operation)
- Need for divergent query answering (when is it done?)

✓ Current solutions don’t cover both
The idea of Armada

- Don’t use a central server
- Track lineage information to give directions
  ✓ We call it Armada
• A reference model for an evolving database system

• Transparent distribution in SQL
Meta data

- Distribute all to all
- Keep it central
- Armada is somewhere in the middle
Armada: chunking

- Chunking is the process of fragmenting a box with data.

- The *chunk operation* takes one box, and produces new ones:

  \[
  T_0 = \text{[ } \% , S_1\text{]} : B_0 ; \left\{ \begin{array}{l}
  \text{[ } f' , S_1\text{]} : B_1 \\
  \text{[ } f' , S_2\text{]} : B_2
  \end{array} \right. \\
  T_1 = \text{[ } \% , S_1\text{]} : B_0 . \text{[ } f , S_1\text{]} : B_1 ; \\
  T_2 = \text{[ } \% , S_1\text{]} : B_0 . \text{[ } f' , S_2\text{]} : B_2 ;
  \]
Armada: evolving

- Chunking upon need
- New sites can host boxes to help out
- Old (empty) boxes stay around for routing

Boxes keep meta-data about their predecessor and successors. This allows to redirect an arbitrary client with an arbitrary query in the right direction.
Armada properties

• Sites or ships, are considered autonomous
• No full replication, no full meta-data replication, no central server
✓ Meta-data (= lineage) distribution
Armada in a database

- Active boxes hold data
- All chunks together form a full table
- Take the union of all chunks
Each Armada starts at a single site, as a single table:

```sql
CREATE TABLE treasures (  
    bag int,  
    coins bigint,  
    origin varchar(64),  
    CONSTRAINT t_b_pkey PRIMARY KEY (bag)  
);
```
Chunking

- Creation of two new tables, representing the boxes
- Inclusion of a check for the chunk function

```sql
CREATE TABLE treasures_Bx (  
    bag    int,  
    coins  bigint,  
    origin varchar(64),  
    CONSTRAINT Bx_b_pkey PRIMARY KEY (bag),  
    CONSTRAINT Bx_b_check CHECK (treasures_F1(bag))  
);
```
Moving

• “move” the data from the old to the new tables

\[
\text{INSERT INTO treasures\_Bx} \\
\text{SELECT bag, coins, origin} \\
\text{FROM treasures} \\
\text{WHERE treasures\_F1(bag);} \\
\]

\[
\text{DROP TABLE treasures;} \\
\]
Recording lineage

- Encode the lineage information into SQL views

```sql
CREATE VIEW treasures AS
    SELECT bag, coins, origin
    FROM (    
        SELECT bag, coins, origin
        FROM treasures_B1
        UNION
        SELECT bag, coins, origin
        FROM treasures_B2
    ) AS treasures;
```
Getting answers

- Clients not aware of Armada
- “Just a query” on the original table
- Execution through views

```sql
SELECT *
FROM (  
    SELECT *  
    FROM treasures;

UNION

SELECT bag, coins, origin  
FROM treasures_B1
UNION
SELECT bag, coins, origin  
FROM treasures_B2
) AS treasures;
```
Client particulars

- Centre of each operation
- Communication flow between client and servers
- No “chaining” or “recursion” at the server side
- Servers don’t play a client role
Execution revisited

• Servers won’t “execute” the views
• Clients get “query plan” instead
• Query plan consists of sub queries and their relations
Query plans

- Look like the views
- Contain sub queries
- Explain how the sub queries must be glued together
Total execution

- Client executes all sub queries from plan
- Client glues together (trivial with UNIONS)
- All boxes are consulted :(
**Chunk functions help**

- Query selection predicate
- Chunk function coverage

```sql
SELECT *
FROM (  
    SELECT bag, coins, origin  
    FROM treasures_B1  
    UNION  
    SELECT bag, coins, origin  
    FROM treasures_B2  
) AS treasures  
WHERE bag < 12;
```

```
SELECT *
FROM treasures  
WHERE bag < 12;
```
Armada is

- Lineage based de-centralised system
- Autonomous servers
- Client centric
Questions?