グリッド協議会 第28回ワークショップ
2009年 12月 17日（木）

NoSQL データストアのデータモデル

首藤 一幸
東京工業大学

耳にしたことありますでしょうか

- NoSQL, key-value store, document-oriented DB, graph DB, …
- memcached, Bigtable, Dynamo, Amazon SimpleDB, Cassandra, Voldemort, Ringo, VPork, MongoDB, CouchDB, Tokyo Cabinet/Tokyo Tyrant, Flare, ROMA, kumofs, Kai, Redis, HadoopのHBase, Hypertable, PNUTS, Scalaris, Dynomite, ThruDB, Neo4j, IBM ObjectGrid, Oracle Coherence, Velocity, …

NoSQL

- **NoSQL** is an umbrella term for a loosely defined class of non-relational data stores that break with a long history of relational databases and ACID guarantees. Data stores that fall under this term may not require fixed table schemas, and usually avoid join operations. The term was first popularised in early 2009. (Wikipediaより)

- RDB ではないデータストア
- 表全体に渡る ACID 性は緩和して、代わりに何かを得る。
  - ACID: DBトランザクションが満たすべき性質。atomicity/原子性, consistency/一貫性, isolation/独立性, durability/永続性。1970年代終わり頃に定義。
  - CAP則とか eventual consistency とか…略

なぜ？

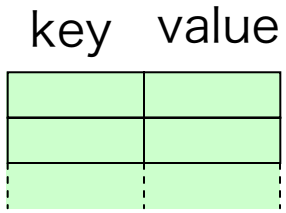
- **分散/スケールアウト, データの複製**は大前提
- 高い性能 / 少ない台数
  - GREE: アクセス履歴
    MySQL 12台/load average 2.0以上 ⇒ Flare 6台/load average 0.2〜0.3
  - mixi: 最終ログイン時刻DB, 1万 queries/sec, 1万クライアント接続
    Tokyo Tyrant/Tokyo Cabinet 2台
  - IBM: 金融方面のRFP「○ msec 以内に反応」
    RDBでは無理と判断し、ObjectGridで達成

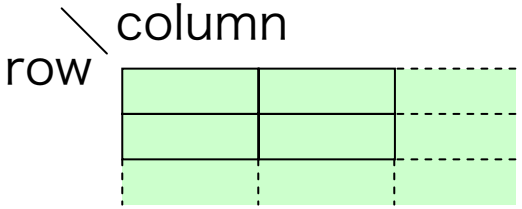
- 高いスケーラビリティ / 数百台〜 / 超大容量
  - Google: ウェブページ, 〜数百TB (2006年)
    Bigtable 数千台
- 両方？
  - Facebook: RDBの負荷軽減
    memcachedでキャッシュ, 800台, 28 TB (2008/12)
  - Amazon: ショッピングカートの中身など
    Dynamo: 〜数百台

NoSQL なデータストア

- 例

	データモデル
– OSS memcached	map (key-value)
– Amazon Dynamo	map (key-value)
– Google Bigtable	table
– Amazon SimpleDB	table
– Windows Azure Table	table



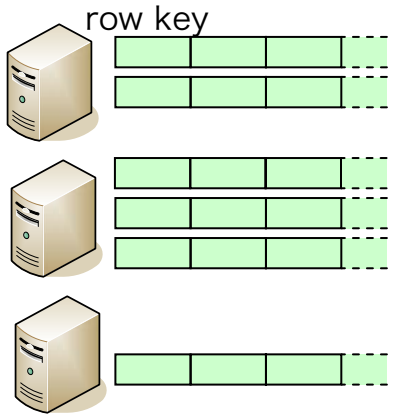


- 共通する構造を見ていく。
  - データモデルと分散の関係

分散と atomicity との関係

- 1 key, 1行が基本単位
  - key-valueペア
  - 行 (row, item, entity)

の　集合



- **atomic な更新の単位 ≦ 分散の単位**
  - key-value ペア, 行は、同一マシンに載る。
  - key, 行が異なると、同一マシンに載るとは限らない。

  - 分散トランザクションはavailabilityを損ねるため、避ける。
    ⇒ atomic な更新は key-value ペア、行でだけ保証。

memcached

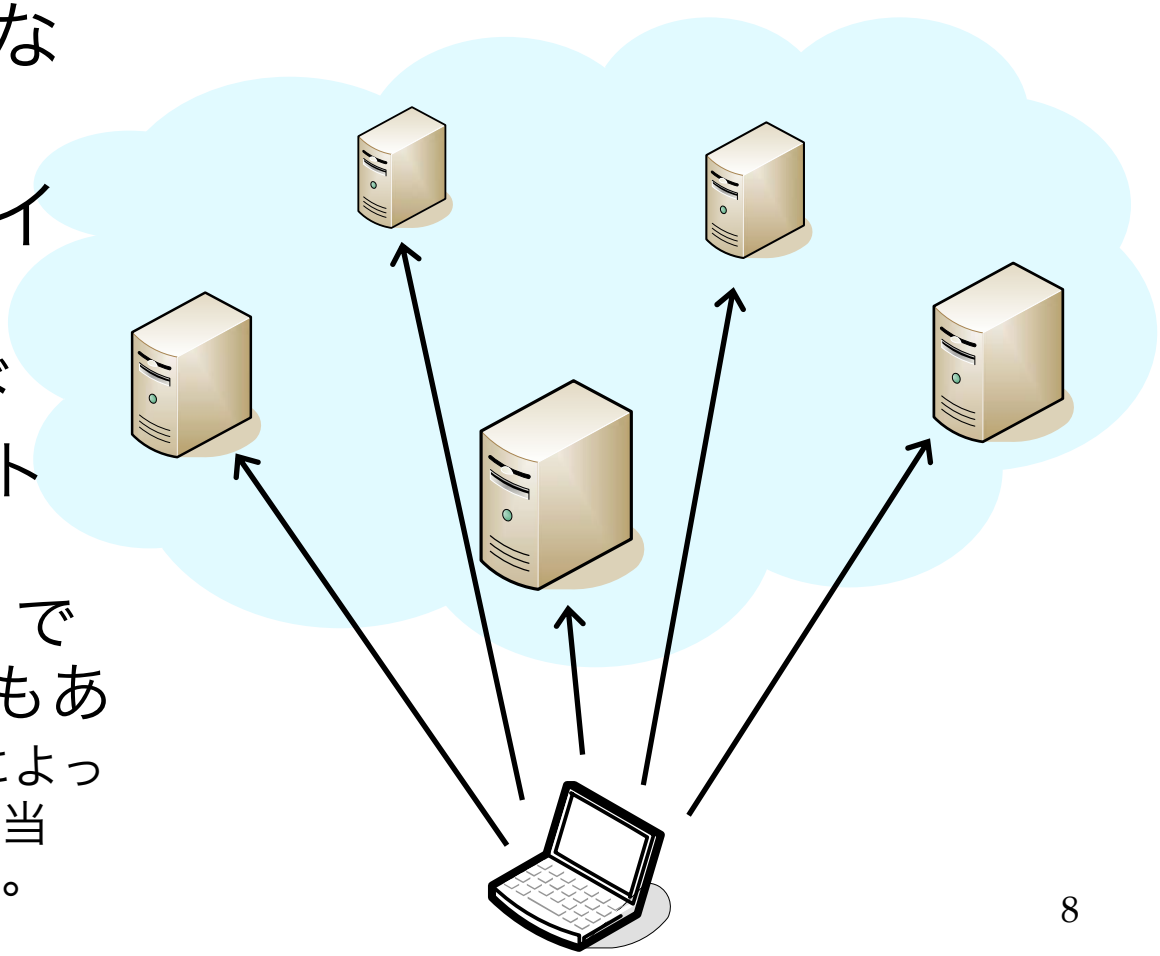
- in-memory key-value store:
  ネットワーク越しに使う map
  - ○万 queries/sec
  - テキストプロトコル (バイナリプロトコルもある)
    - set aKey 0 0 6 aValue
    - get aKey

こういうアレ:
aMap.put(aKey, aValue)
value = aMap.get(aKey)

- 用途
  - RDB から取得したデータをキャッシュ、など
- ウェブ上サービスの裏側で広く使われている
  - はてな, livedoor, mixi, Facebook, YouTube, Digg, Twitter, Wikipedia, …
  - Facebook: 800台以上で 28 TB (2008/12)

memcached: ノードの分散

- memcached 自体に分散のための機能はない。
- 担当ノードはクライアントが決める。
  - キー → 担当ノード
  - 方式はクライアントライブラリ次第。
  - consistent hashing ではないライブラリもある → ノード追加/削除によって、キー全体に渡って担当ノードが変わってしまう。

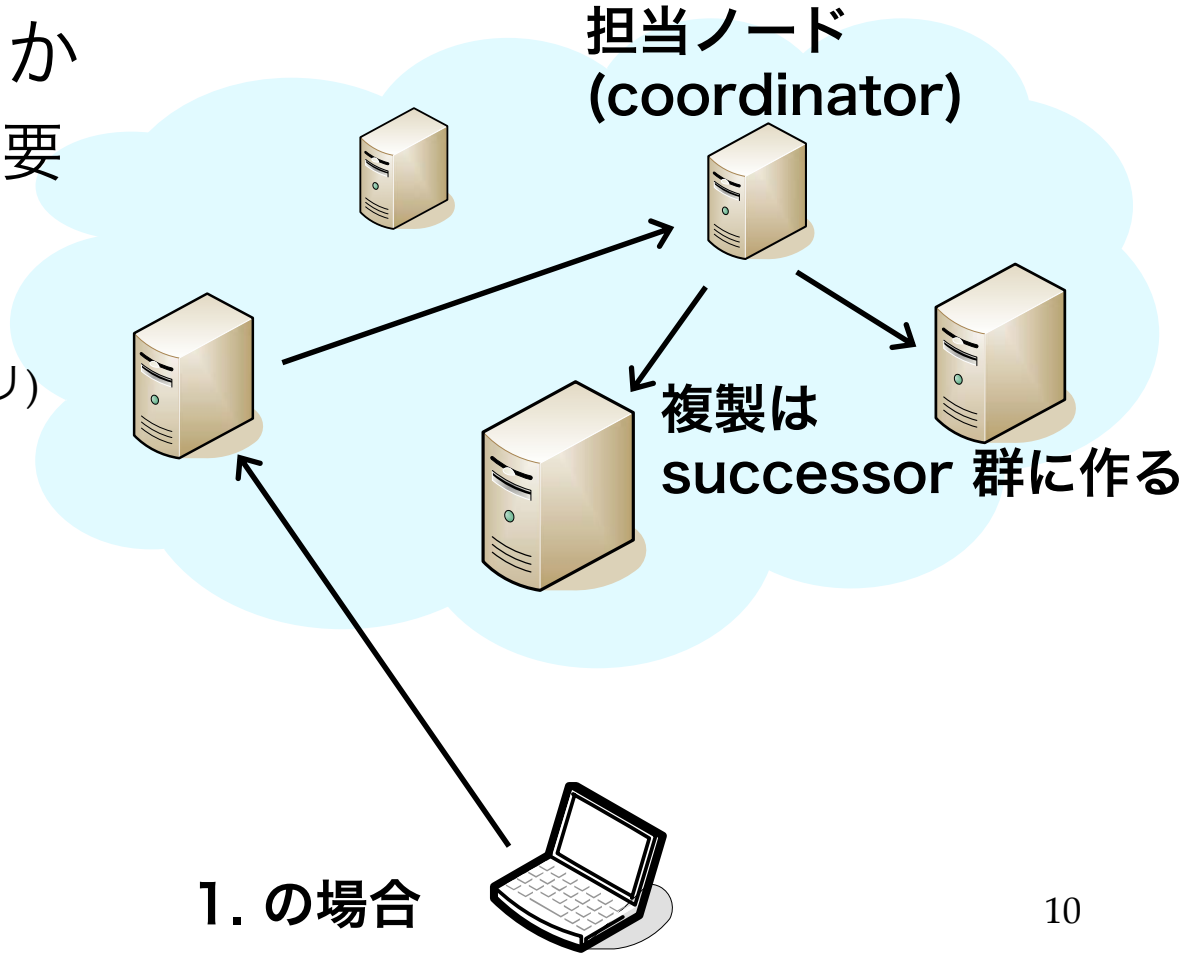


Dynamo

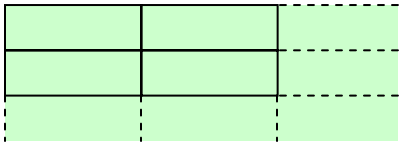
- key-value store
  - in-memory とは限らない。Berkeley DB, MySQL も使える。
- Amazonが論文を発表。SOSP'07。実装は非公開。
  - 新奇なアイディアというよりは、手堅い設計。
  - Amazon が内部で使用 とのこと。
    - ショッピングカート, …
- 狙う規模: 数百台
- 担当ノードの決め方: consistent hashing
- availability: "always-on experience"
  - 完全に非集中
- 一貫性: eventual consistency
  - 不整合が起きた場合、データに付いてるvector clocks [Lamport1978] を手がかりに、アプリ側で解決する。
- Service Level Agreements (SLA)
  - ウェブアプリのバックエンド ⇒ 低遅延 重視
  - クライアントとの SLA に基づいて、パラメータ (e.g. 複製数) を調整。
    - 例: 500 req/s までなら要求の 99.9% までは 300 msec 以内に返答を返す。

Dynamo: ノードの分散

- 担当ノードを決めるのは、次のどちらか
  1. クライアントから要求を受けたノード
  2. クライアント
     (要 クライアントライブラリ)



Bigtable



- table
  - ただし、固定されたスキーマはない。
  - atomic な更新: 行単位, not 表単位
  - “sparse, distributed, persistent multi-dimensional sorted map”
- Google が論文を発表。OSDI’06。実装は非公開。
  - Google File System, Chubby などの上に実装。
  - Google が内部で使用。
  - Google App Engine (PaaS) 上のアプリから使うデータストア。
  - いくつかクローンあり: HadoopのHBase, Hypertable
- 狙う規模: 数千台でペタバイト以上
- 利用例
  - ウェブのクロール結果
    - 行: URL, 列: コンテンツ, リンク元, …
  - Google Analytics, Earth, …

Bigtable: コード例

row key	"contents:"	"anchor:cnnsi.com"	"anchor:my.look.ca"	"anchor:…
"com.cnn.www"	"<html>…	"CNN"	"CNN.com"	…
"com.example.www"	"<html>…		"Example.com"	…

2009/12/17… <html>…
2009/12/10… <html>…
2009/12/3…

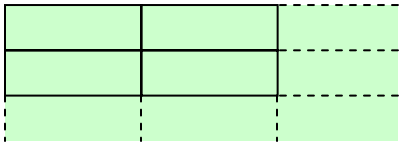
- 言語は C++。論文のFigure 2。書き込みの例。
- // Open the table
  Table *T = OpenOrDie("/bigtable/web/webtable");

```
// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");

Operation op;
Apply(&op, &r1);
```

- row key の指定が必要
  - 参考）Megastore ライブラリが補完: スキーマの宣言, 複数行にまたがるトランザクション, row key以外のインデックス, …
- **行単位**での atomic な更新

Amazon SimpleDB



- table ととらえることができる
  - ただし、固定されたスキーマはない。
  - atomic な更新: 行単位, not 表単位

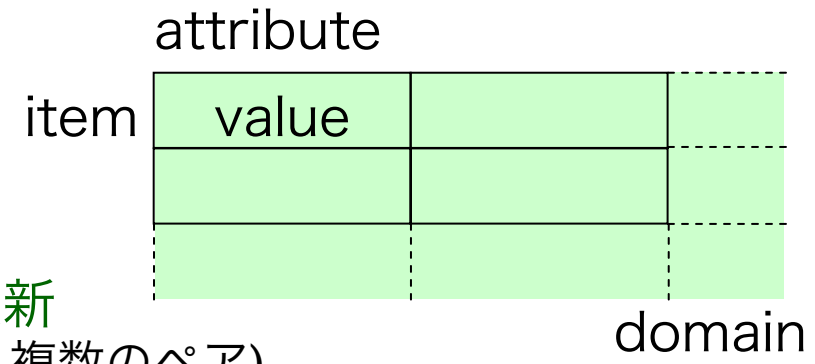
- Amazon Web Services (AWS) の１つ。
  - クラウド: 初期費用なし, 従量課金
  - REST または SOAP over HTTP でネット越しに使う。
    - 実装は非公開。Dynamo とは別のソフトウェア。
  - 自動的にスケールアウト/イン
    - “Amazon SimpleDB automatically indexes your data, creates geo-redundant replicas of the data to ensure high availability, and performs database tuning on customers’ behalf. Amazon SimpleDB also provides no-touch scaling. There is no need to anticipate and respond to changes in request load or database utilization; the service simply responds to traffic as it comes and goes …“

amazon web services™

Amazon EC2
Amazon S3
Amazon SimpleDB
Amazon SQS
Amazon CloudFront
Amazon VPC

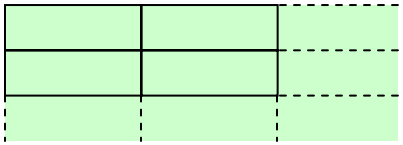
Amazon SimpleDB: API

- 表としてとらえると
  - domain:　表/table
  - item:　行/row
  - attribute:　列/column



- itemに、attributeと値のペアを追加・更新
  - void PutAttributes(domain名, item名, 複数のペア)
    ↑ 擬似コード。本当は REST, SOAP のリクエスト。
- 値を取得
  - 複数のペア GetAttributes(domain名, item名, 複数のattribute名)
- 比較的リッチな問い合わせが可能
  - 例: select * from mydomain where Year > '1975 and Year < '2008'
  - item 名が返される。
  - SQL と比べると制限はある。
  - インデックスを、いつ、どの attribute に対して作成しているのか？？？
- **行単位**での atomic な更新

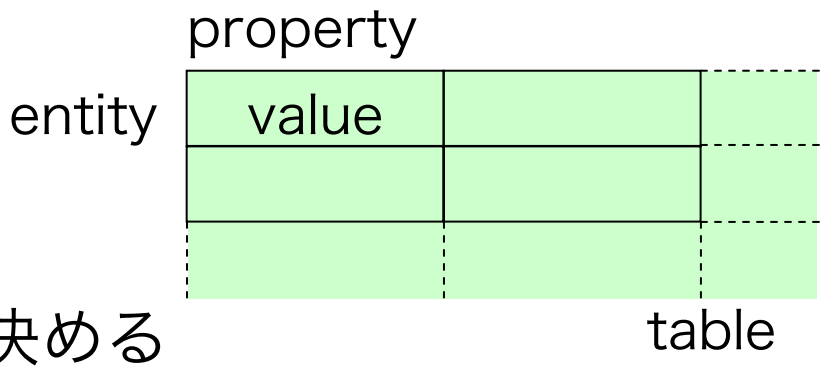
Windows Azure Table



- table
  - ただし、固定されたスキーマはない。
  - atomic な更新: 行単位, not 表単位
- 狙う規模: 数千台, 数十億行, テラバイト
  - トラフィックに応じて。
- Windows Azure Storage の一部
  - Windows Azure Table, Blob, Queue
  - .NETのAPI, LINQ または REST でネット越しに使う。
- 参考: SQL Azure: RDB提供サービス とは別物
  - SQL Azure: SQL Server (RDB) のホスティング
  - Amazon RDS: MySQL のホスティング

Windows Azure Table

- 表としてとらえると
  - table
  - entity: 行/row
  - property: 列/column



- 必須 property ― プログラマが決める
  - “PartitionKey” 同一なら、同じマシンに載る
    指定がないと、全 partition → 全マシン 走査！
  - “RowKey” partition 中で、そのentityを一意に識別する ID

例

	PartitionKey	RowKey	Date		Description
Partition 1	Example Doc	V1.0	2007/8/2	…	Committed version
	Example Doc	V2.0.1	2007/9/28	…	Alice’s version
Partition 2	FAQ Doc	V1.0	2007/5/2	…	Committed version
	FAQ Doc	V1.0.1	2007/7/6	…	Alice’s version
	FAQ Doc	V1.0.2	2007/8/1	…	Sally’s version

データモデル: Bigtable

データモデル

- table
- multi-dimensional sorted map

	contents:	anchor:cnnsi.com	anchor:my.look.ca	anchor:…
com.cnn.www	<html>…	CNN	CNN.com	…
com.example.www	<html>…		Example.com	…

2009/12/17… <html>…
2009/12/10… <html>…
2009/12/3…

物理データ構造

- sorted map

com.cnn.www+contents:+2009/12/17…	<html>…
com.cnn.www+anchor:cnnsi.com+2009/…	CNN
com.cnn.www+anchor:my.look.ca+2009/…	CNN.com
com.example.www+contents:+2009/12/17…	<html>…
com.example.www+contents:+2009/12/10…	<…
com.example.www+contents:+2009/12/13…	<…
…	…

マシン群で分担

“Row key + column key + timestamp” をキーとする map

これが、稀に key-value store と呼ばれることがある理由？

データモデル: Azure Table

Bigtable と同じかもしれないし、こうかもしれない

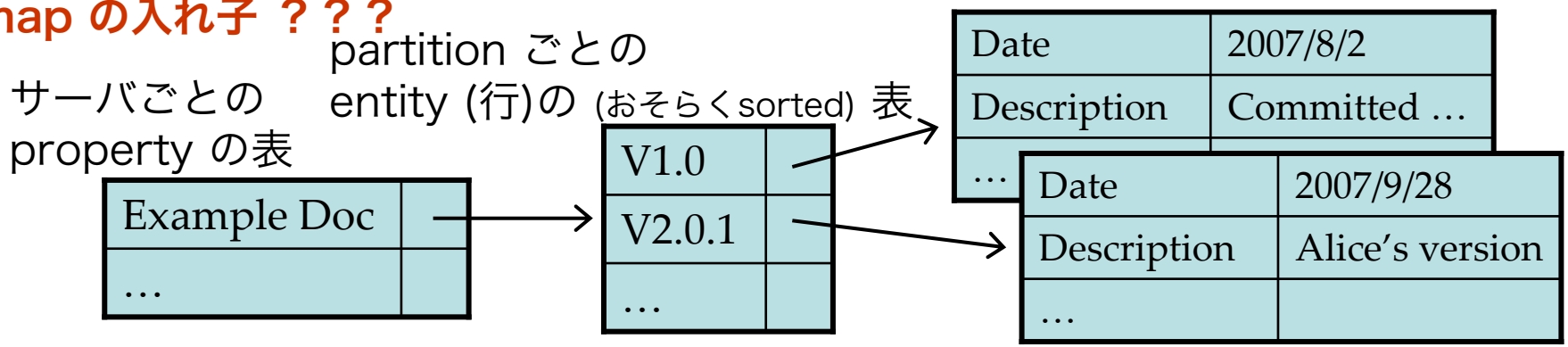
データモデル

- table

	PartitionKey	RowKey	Date		Description
Partition 1	Example Doc	V1.0	2007/8/2	…	Committed version
	Example Doc	V2.0.1	2007/9/28	…	Alice's version
Partition 2	FAQ Doc	V1.0	2007/5/2	…	Committed version
	FAQ Doc	V1.0.1	2007/7/6	…	Alice's version
	FAQ Doc	V1.0.2	2007/8/1	…	Sally's version

物理データ構造 (直感での想像)

- map の入れ子 ？？？



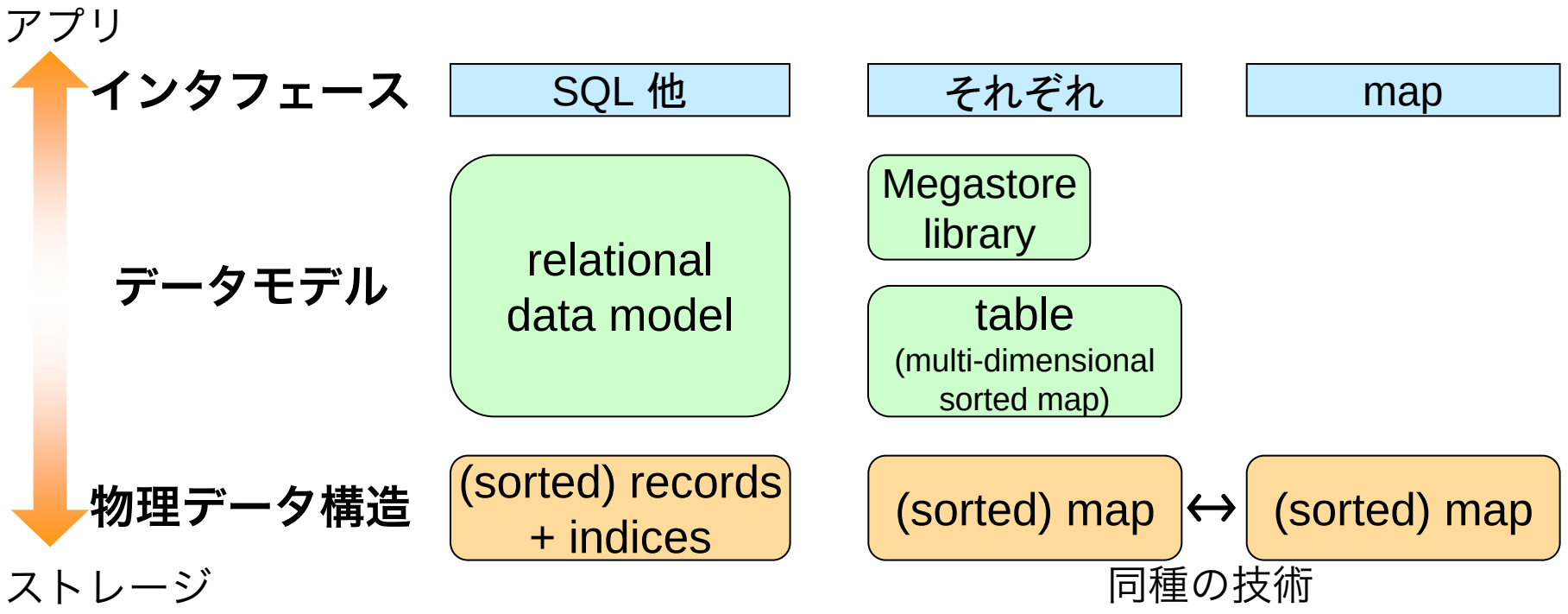
データモデル

- 実装（物理データ構造）は非公開
  - Amazon SimpleDB
    - GetAttributes(…) は要 item 名 → 行を特定, 担当マシンを特定
    - select は… どう実装してるのか？
  - Windows Azure Table
    - 問い合わせ時、PartitionKey で担当マシンを特定。
      指定しないと、全マシンで走査。

- 要考察: Bigtable と同じか？異なるか？
  - 異なるとしたら、どう？

データモデル

- RDB, NoSQLなデータストアの構造を整理・比較（仮説）



まとめ

- NoSQL データストアの中身、特に、スケールアウトを可能とするデータモデルを概観した。

- 今後
  - NoSQL データストアの活かしどころ、使い方がこなれて、広く知られていく。
  - クラウド上の設計・開発方法論が (同上)
    - 例: message queue を活用して、行や表をまたがって整合性を保つ。
    - これを早く獲得することが競争力に。

謝辞

- 古橋貞之 様
- 萩原正義 様
- 佐藤一憲 様
- 平林幹雄 様
- 吉田幹 様