Life on the Farm: Using SQL for Fun and Profit in Windows Live

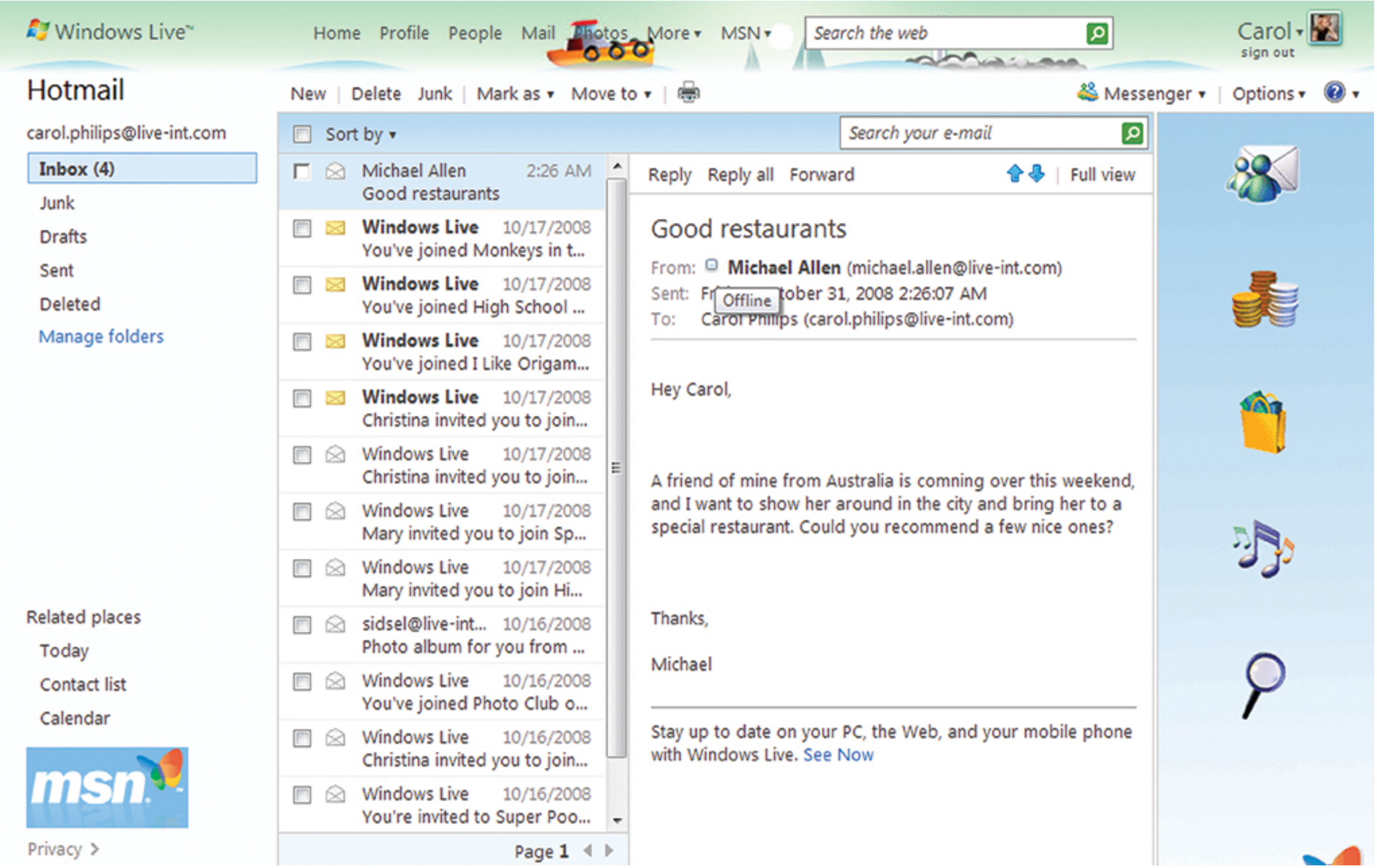
David Nichols

Microsoft Windows Live

Oct 11, 2009

What is Windows Live?

- Not:
  - Bing – search
  - Azure – cloud computing platform
  - MSN – News and entertainment portal
- Windows Live is
  - Mail (Hotmail)
  - Instant messaging
  - Photo and file sharing
  - Calendar
  - Social networking
  - Blogging
  - File sync



□	Michael Allen Good restaurants	2:26 AM
□	**Windows Live** You've joined Monkeys in t...	10/17/2008
□	**Windows Live** You've joined High School ...	10/17/2008
□	**Windows Live** You've joined I Like Origam...	10/17/2008
□	**Windows Live** Christina invited you to join...	10/17/2008
□	Windows Live Christina invited you to join...	10/17/2008
□	Windows Live Mary invited you to join Sp...	10/17/2008
□	Windows Live Mary invited you to join Hi...	10/17/2008
□	sidsel@live-int... Photo album for you from ...	10/16/2008
□	Windows Live You've joined Photo Club o...	10/16/2008
□	Windows Live Christina invited you to join...	10/16/2008
□	Windows Live You're invited to Super Poo...	10/16/2008

Good restaurants

From: **Michael Allen** (michael.allen@live-int.com)
Sent: Fr tober 31, 2008 2:26:07 AM
To: Carol Philips (carol.philips@live-int.com)

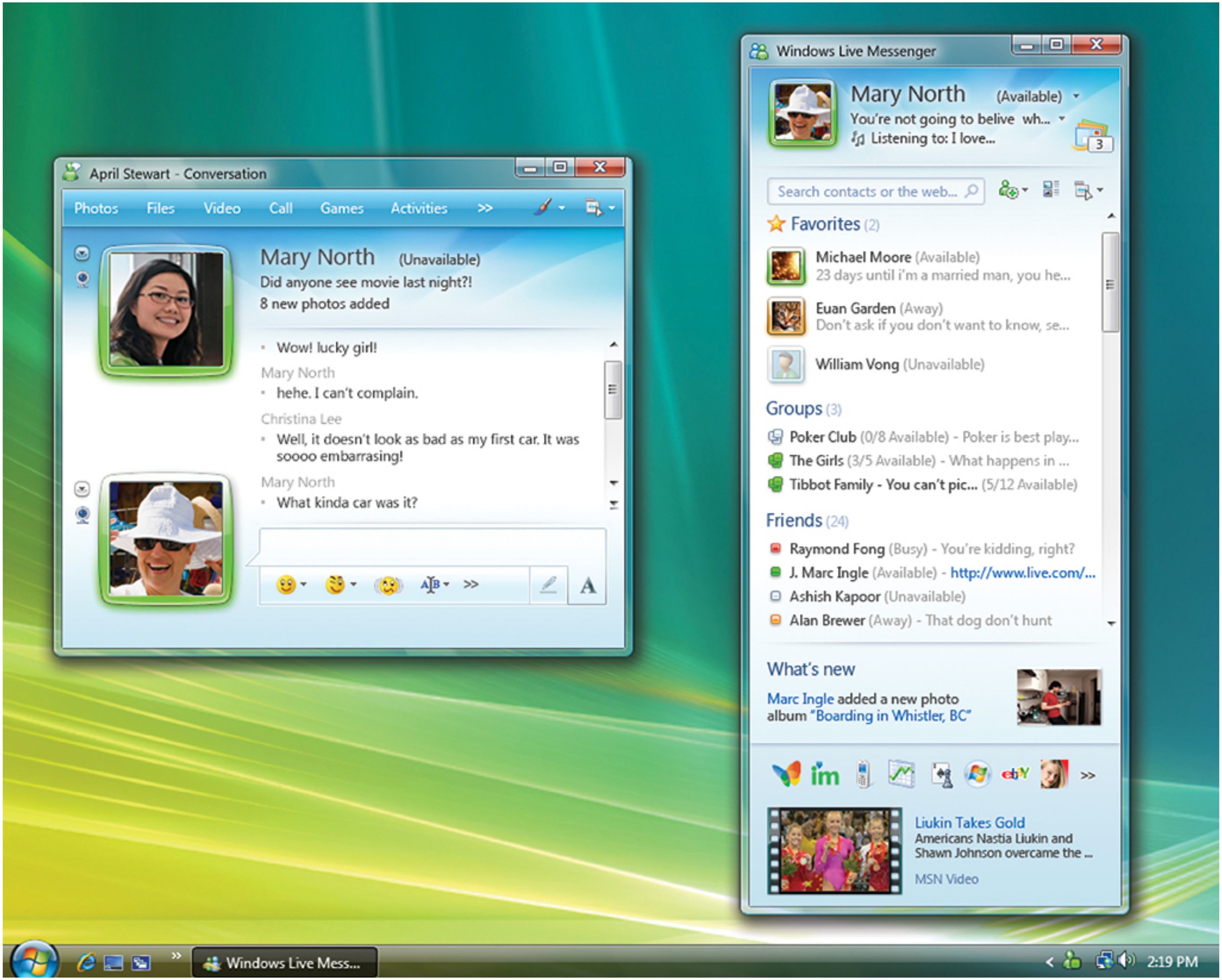
Hey Carol,

A friend of mine from Australia is comning over this weekend, and I want to show her around in the city and bring her to a special restaurant. Could you recommend a few nice ones?

Thanks,

Michael

Stay up to date on your PC, the Web, and your mobile phone with Windows Live. See Now



Mary North (Unavailable)
Did anyone see movie last night?!
8 new photos added

- Wow! lucky girl!

Mary North
- hehe. I can't complain.

Christina Lee
- Well, it doesn't look as bad as my first car. It was soooo embarrasing!

Mary North
- What kinda car was it?

Mary North (Available)
You're not going to belive wh...
Listening to: I love...

Favorites (2)

Michael Moore (Available)
23 days until i'm a married man, you he...

Euan Garden (Away)
Don't ask if you don't want to know, se...

William Vong (Unavailable)

Groups (3)

Poker Club (0/8 Available) - Poker is best play...
The Girls (3/5 Available) - What happens in ...
Tibbot Family - You can't pic... (5/12 Available)

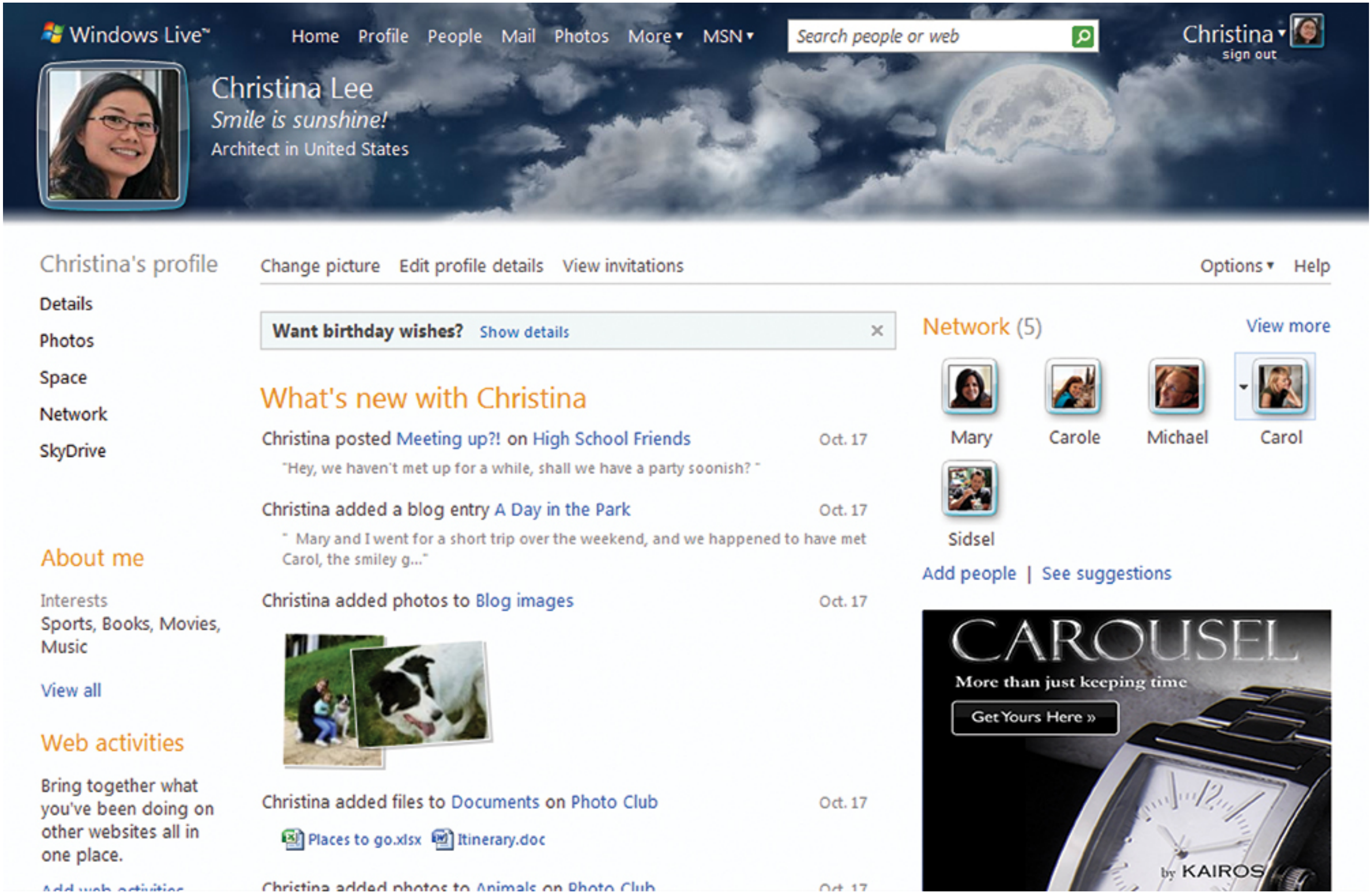
Friends (24)

Raymond Fong (Busy) - You're kidding, right?
J. Marc Ingle (Available) - http://www.live.com/...
Ashish Kapoor (Unavailable)
Alan Brewer (Away) - That dog don't hunt

What's new

Marc Ingle added a new photo album "Boarding in Whistler, BC"

Liukin Takes Gold
Americans Nastia Liukin and Shawn Johnson overcame the ...
MSN Video



Windows Live™ Home Profile People Mail Photos More ▾ MSN ▾ Search people or web

Christina ▾
sign out

Christina Lee
Smile is sunshine!
Architect in United States

Christina's profile

Details
Photos
Space
Network
SkyDrive

About me

Interests
Sports, Books, Movies, Music

View all

Web activities

Bring together what you've been doing on other websites all in one place.

Add web activities

Change picture Edit profile details View invitations Options ▾ Help

Want birthday wishes? Show details ×

What's new with Christina

Christina posted Meeting up?! on High School Friends Oct. 17

"Hey, we haven't met up for a while, shall we have a party soonish? "

Christina added a blog entry A Day in the Park Oct. 17

" Mary and I went for a short trip over the weekend, and we happened to have met Carol, the smiley g..."

Christina added photos to Blog images Oct. 17

Christina added files to Documents on Photo Club Oct. 17

Places to go.xlsx Itinerary.doc

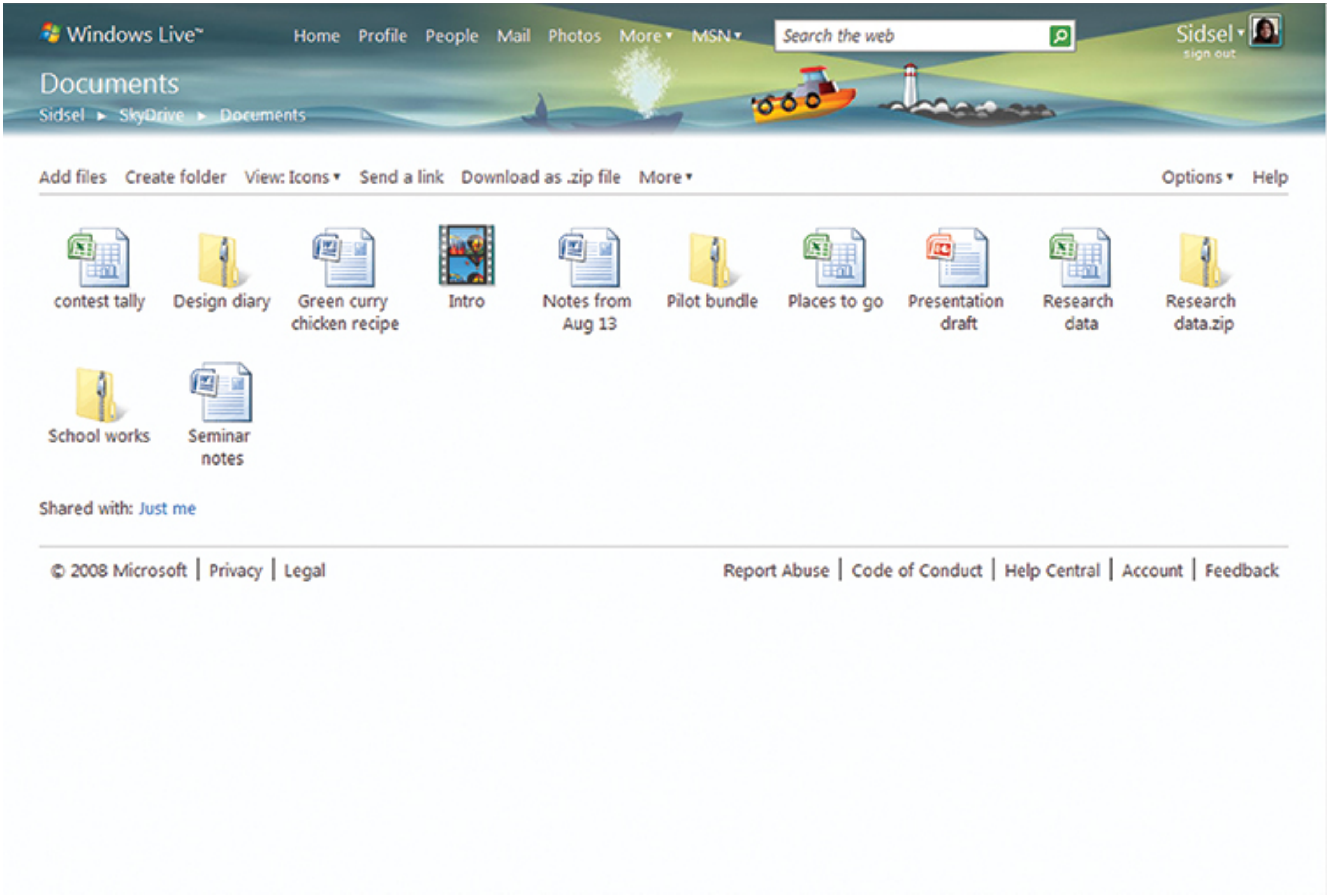
Christina added photos to Animals on Photo Club Oct. 17

Network (5) View more

Mary Carole Michael Carol
Sidsel

Add people | See suggestions

CAROUSEL
More than just keeping time
Get Yours Here »
by KAIROS



Shared with: Just me

Why SQL?

- Familiar, tested programming model
  - Real queries, real transactions
  - Good data modeling
  - Excellent at OLTP
  - Easy to find developers that know it
- Solid systems software
  - Zillions of miles of usage in many real-life settings
  - Active product group upgrading it regularly
  - Product team tunes for modern system configurations

Challenges with using SQL

- Living without single-image database model
  - No global transactions
  - No global indexes
- Administration/maintenance
- Breaking things at scale

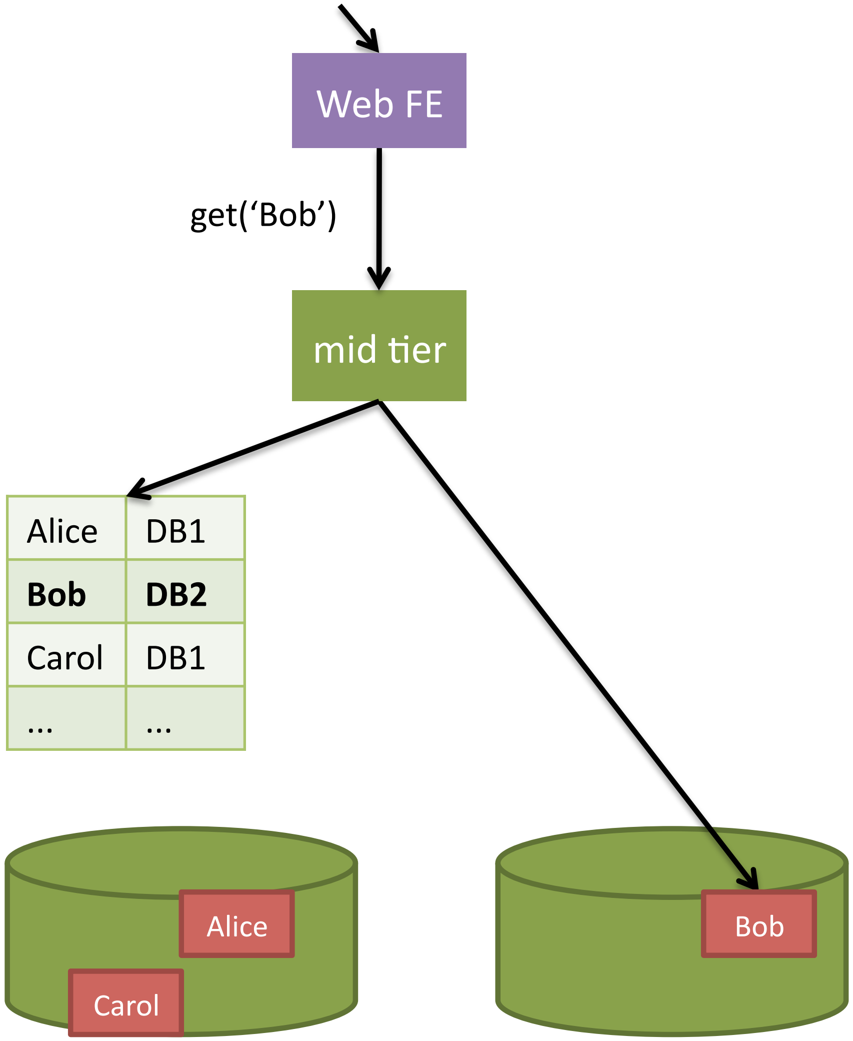
Outline

- Scaling up
- Data reliability
- Tools for partitioned stores
- Operational issues
- Conclusions

SCALING UP

Basic partitioning

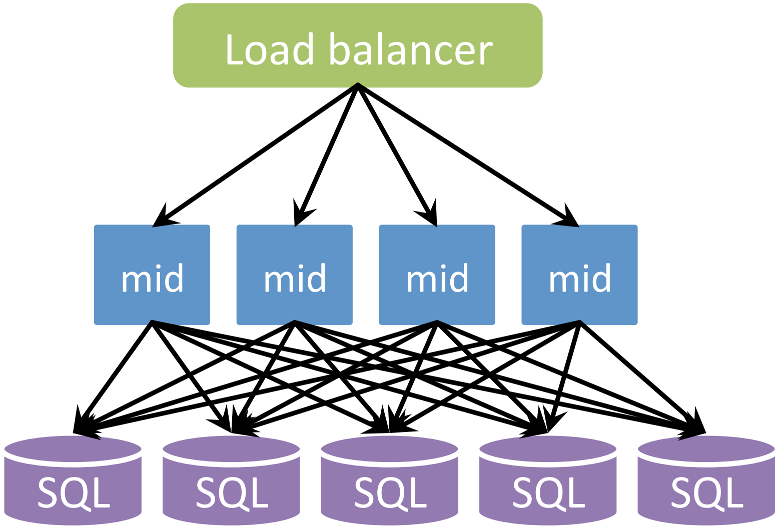
- Database partitioned by user; many users per instance DB.
- Front ends send requests to proper database.
- Location is determined by lookup
  - Lookup Partition Service (LPS) DBs map users to partitions
  - Partition table is partitioned by hash
- Explicit placement helps manage system
  - Incremental rebalancing
  - Balance load



Alice	DB1
Bob	**DB2**
Carol	DB1
…	…

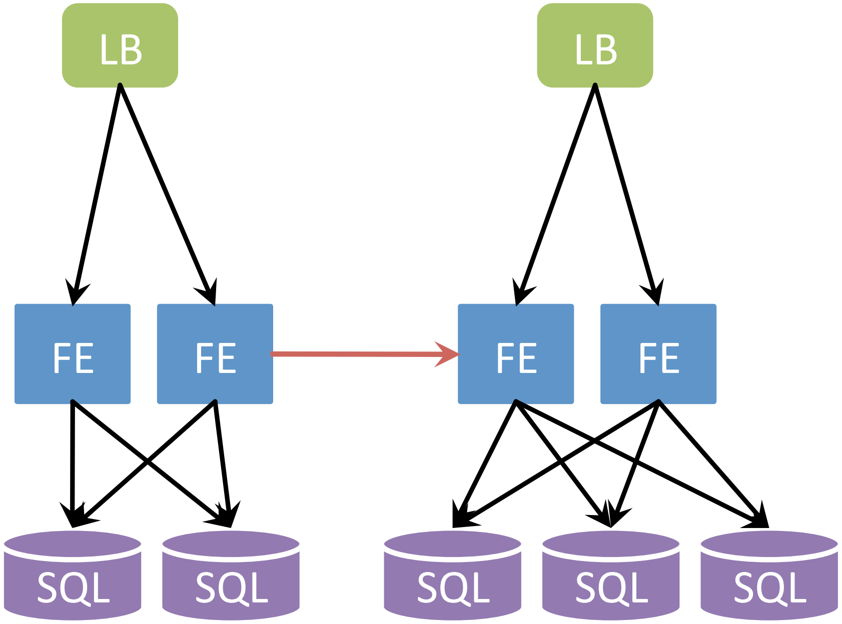
Connection scaling

- Assumes no component sees load proportional to total system load.
- But each SQL server sees connection from each FE



Cluster affinity

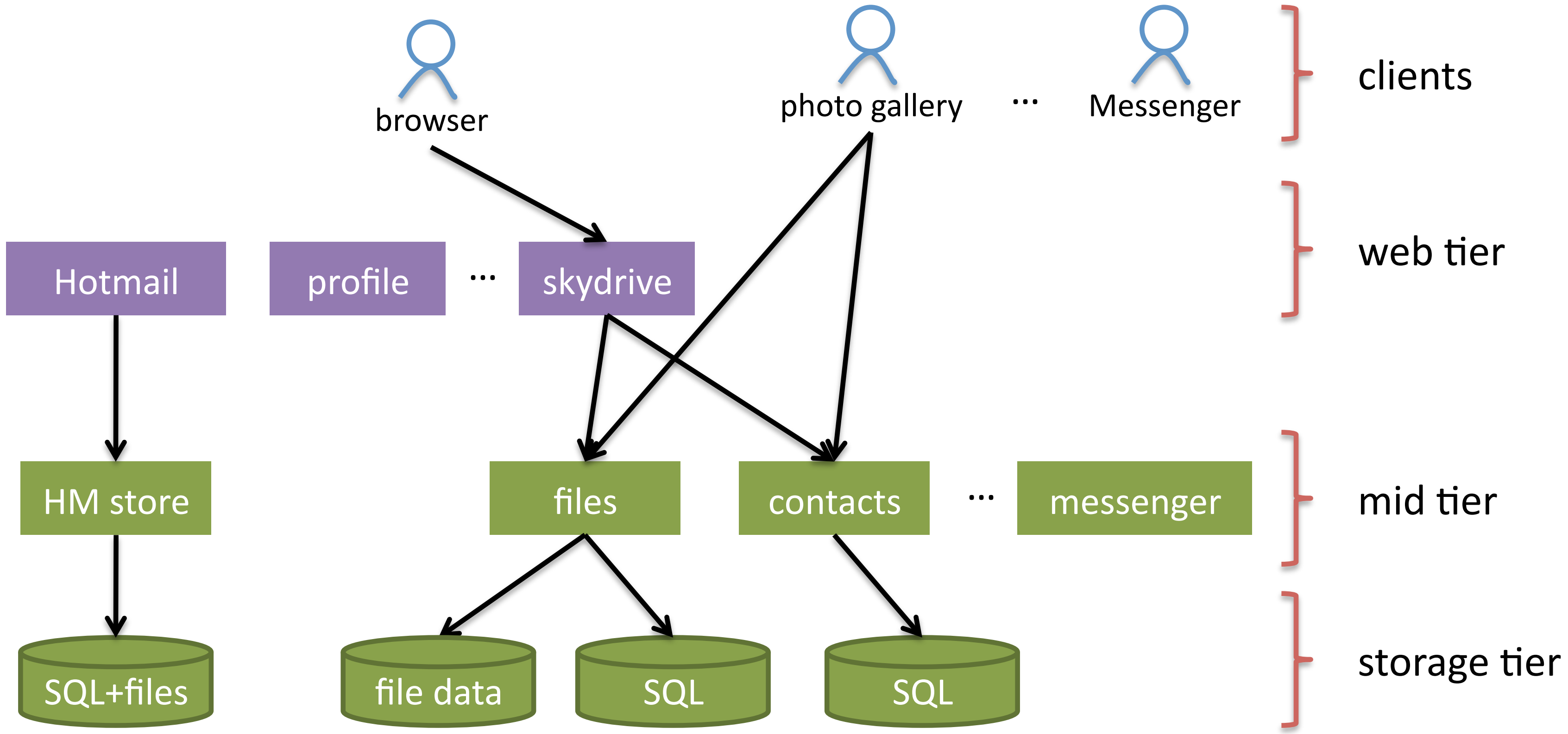
- FE's are in pools, each serving subset of DBs
- App-level changes to support redirect and cluster caching
- Substitute FE-FE calls if two ABs are affected.



Architecture

- Three stages of scale out
  - Bigger server
  - Functional division
  - Data division
- WL uses both of the last two; we have services that own functions, and they do data division within them
  - Driven by integration issues; e.g. ABCH shared between messenger and HM; neither was what we wanted
  - Also driven by org; we need to keep systems manageable in complexity

Windows Live rough architecture

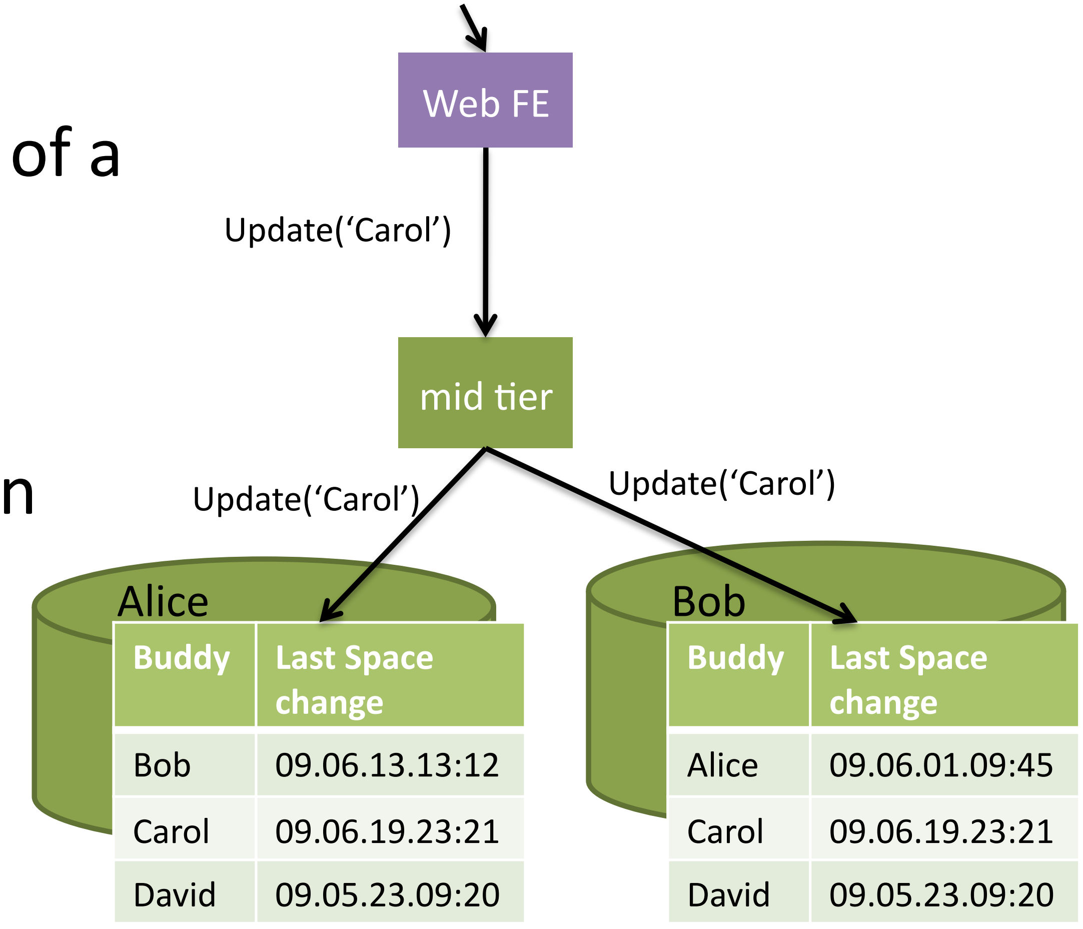


Cross-partition access

- Updates to multiple services and users
  - Examples:
    - Add Messenger buddy
    - Upload photo (write file store and recent activity store)
  - Two-phase commit is out
  - Instead: Ad hoc methods
    - Write A intent, write B, write A
    - Write A and work item, let work item write B
    - Write A, then B, tolerate inconsistency
- Reads for multiple users
  - Needed for social network access
  - Example: Find photo, name, .. for all my friends
  - Typically a join on the user relationship

Inverted data pattern

- Used to find out something about all of a person's friends/ buddies
- Store copy in each buddy's partition, fan out writes
  - Many copies of data
  - High write rate
- Try to use only for binary relationships



Alice

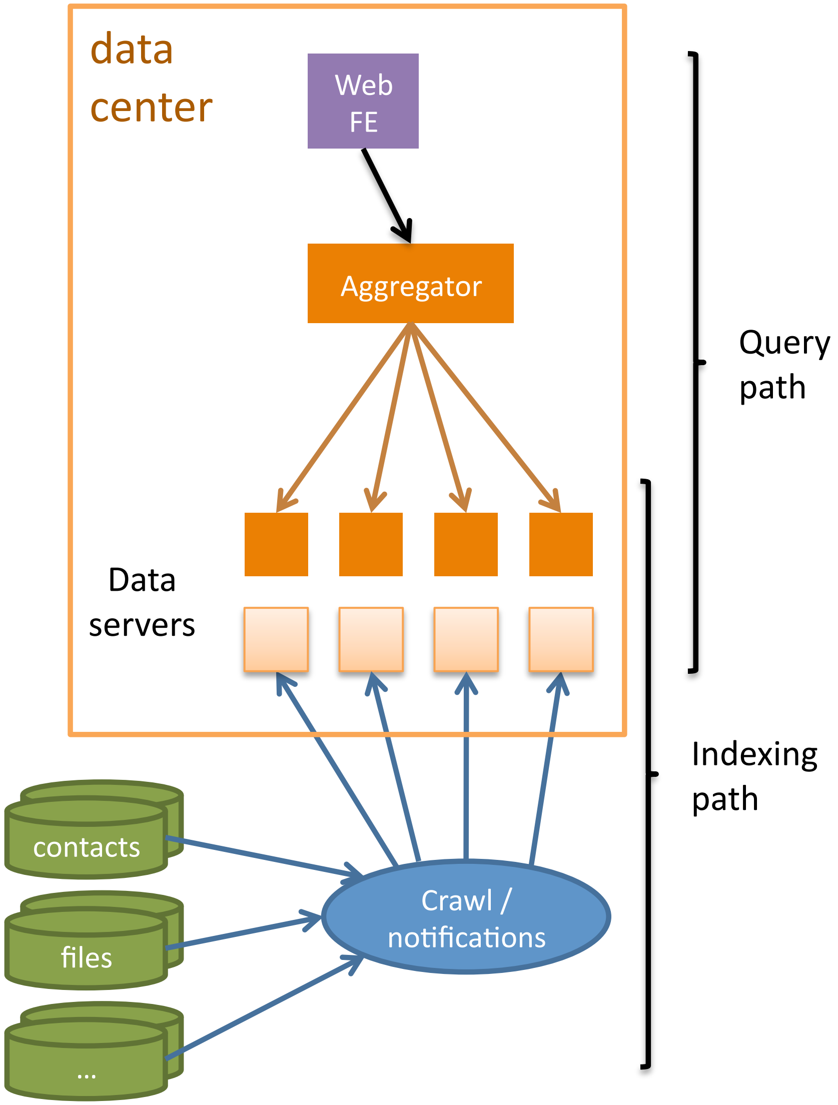
Buddy	Last Space change
Bob	09.06.13.13:12
Carol	09.06.19.23:21
David	09.05.23.09:20

Bob

Buddy	Last Space change
Alice	09.06.01.09:45
Carol	09.06.19.23:21
David	09.05.23.09:20

Aggregated user data pattern

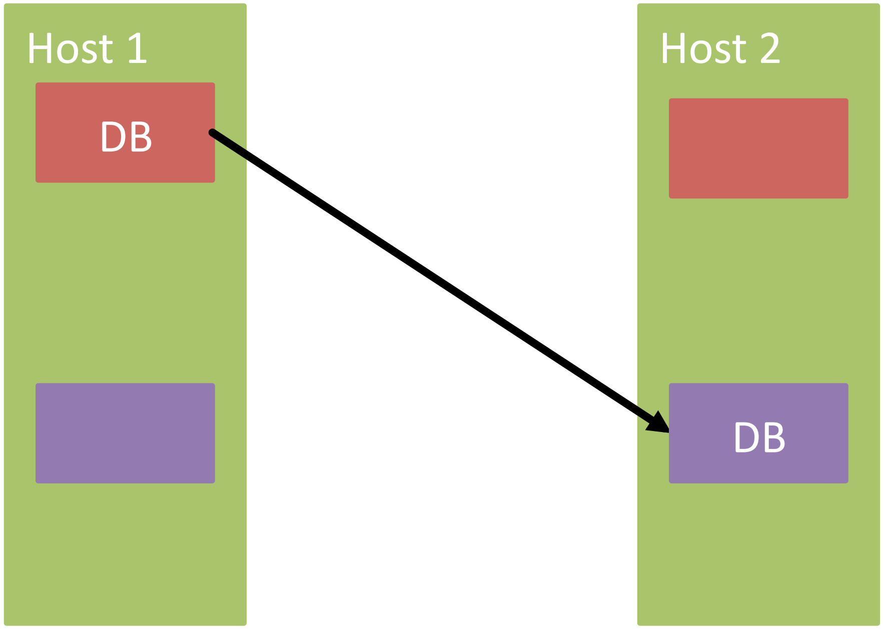
- For social network scenarios requiring data about many users of interest to viewer
- Gather data subset in one place
  - Combine many users
  - Combine several stores
- Data is partitioned, but in single data center
- Fan-out queries go to many/all servers in row
- Caching keeps most data in memory for speed
- Optional: Extra data rows for query capacity



DATA AVAILABILITY AND RELIABILITY

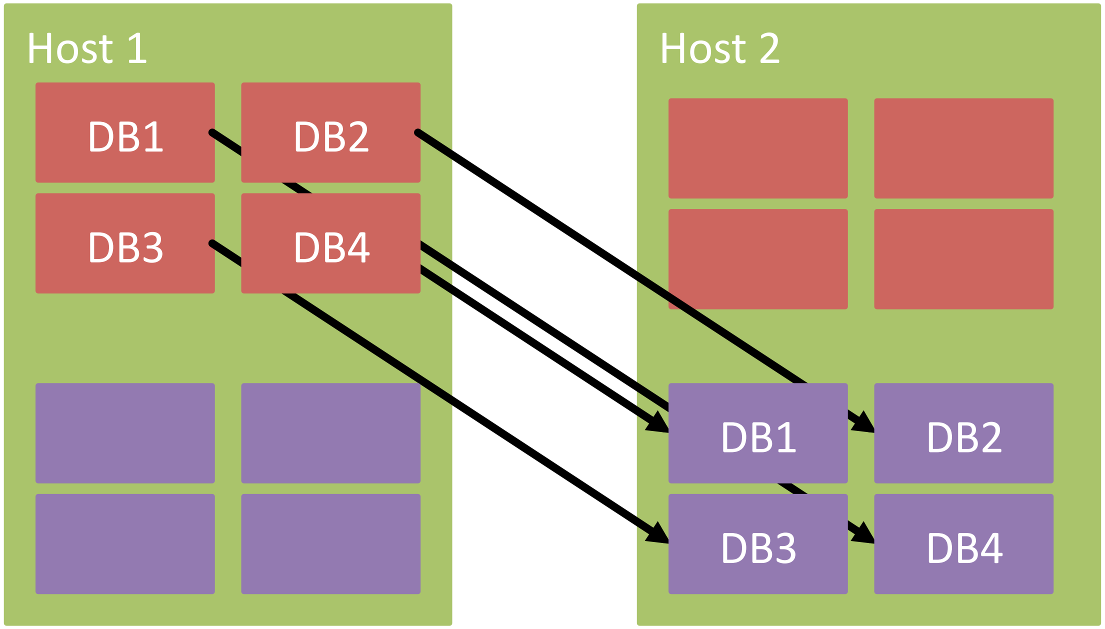
Replication

- We replicate all user data using SQL replication
- Front ends have library (WebStore) to notice failures and switch to secondary
- Original scheme was one-to-one.
    - Too slow due to parallel transactions vs. single replication stream.



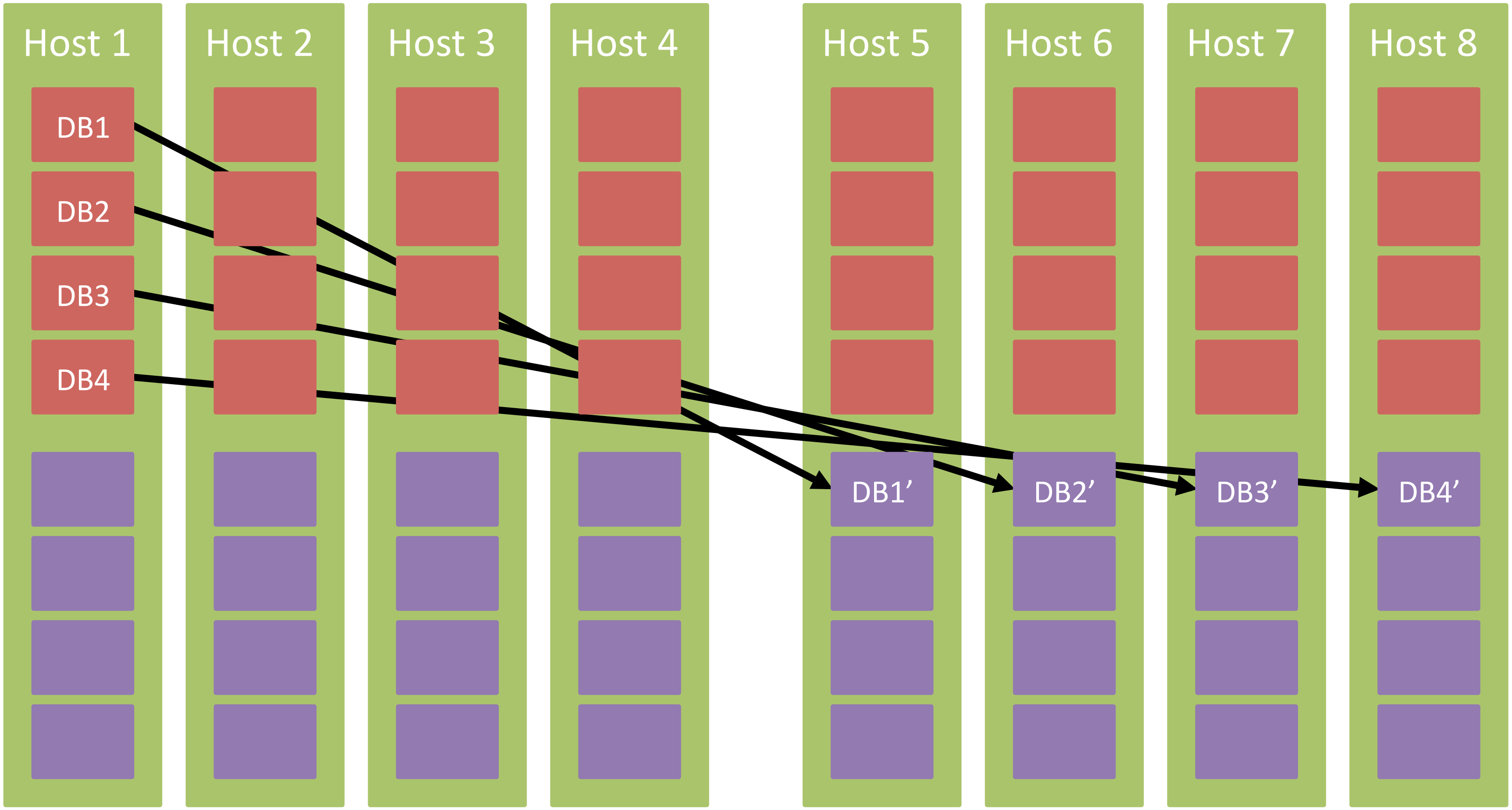
Replication

- Next try was four DB to four DB.
  - Fixed most speed problems.
  - Too much load on secondary after a failure.



Replication

- Current configuration uses 8-host pods
- 25% load increase for secondaries on failure



Replication

- Still not fast enough for key tables
  - Again, 100's of writing threads vs. 4 replication streams
  - "Manual" replication
    - FE's run SProcs at both primary and secondary
    - But small probability of inconsistent data
- Replication runs a few seconds behind
  - Ops reluctant to auto-promote secondary due to potential data in replication stream
  - New SQL tech should fix this

Data loss causes

- External application
- Old data

Above the app

- Software bugs
  - Esp. migration logic bugs

In the app

- Controller failure
- Disk failure

Below the app

Data loss mitigations

- External application
- Old data
- Software bugs
  - Esp. migration logic bugs

- Controller failure
- Disk failure

- Audit trails
- Soft delete
- Per-user backup

- Tape backup
- SQL Replication
- RAID

Data reliability lessons

- "Replication solves availability, backup solves data loss."
- Moving toward logical recovery
- Soft delete everything
- 4-day rule

TOOLS

WebStore

- Manages the thousands of databases in a deployment
  - Sets up replication
  - Deploys schema and SPROCs
  - Provides monitoring and self-healing
- Associated client library
  - Routes requests to proper databases
  - Routes to replicas during server outages
- “App platform for FEs, ops platform for BEs”

Managing Replication

- Fail safe set
  - A set of databases in some sort of replication relationship
  - Typical fail safe set is two to four databases (most are two)
  - Fail safe sets are the true targets of partitioned operations (hundreds or thousands in a deployment)
- WebStore on SQL servers:
  - Sets up SQL replication/mirroring between the databases in a fail safe set
  - Allows operators to promote secondary to primary status after failure
- WebStore on mid tier hosts:
  - Monitors state of each database in a fail safe set
  - Automatically chooses healthy database for each request
  - Informs application whether current operation is aimed at read/write copy or a read-only copy

Upgrade options

- Upgrade partitions.
  - Run DDL in each partition (via WebStore)
  - This is complicated by replication
  - After all DBs are done, upgrade FEs
    - SProcs are compatible; changed APIs get new names
- Migrate users
  - Can take various forms: between servers, within a server, between services(!)
  - Complex, slow, error-prone; nobody's favorite
- Either way, mixed versions in production are a fact of life

Other tools

- Background jobs
  - Examples: User expiration, Indexing for search
  - Also help shed peak load
    - Ex. Delete is expensive, so soft delete and use batch job
  - Mostly automatic, but occasionally ops has to reschedule to avoid peak load
- Migration
- Reporting
  - Too much load for live site
  - Now use backup/restore to non-production servers

OPERATIONS

Things fail

- A lot*:
  - 2-3 disks fail/day.
  - 2 or so “sick machines” need reboot/day.
  - Several bad memory, motherboard, etc./week.
  - 1 disk controller failure/month.
  - 1 machine on fire in last three years.
- Of these, disk controller failure is worst.
  - Data corruption on disk.

*Old stats based on subset of whole service.

How to crash a cluster

- Slow down a SQL server a bit
  - Could happen randomly or due to maintenance, e.g. re-indexing a DB.
  - Connection pools will ask for extra connection.
  - Now all the FEs need a new connection to the DB.
  - SQL dies with lots of new connections.

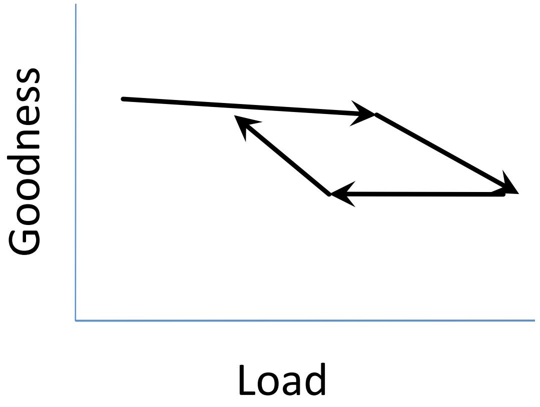
- WebStore now implements connection throttling.

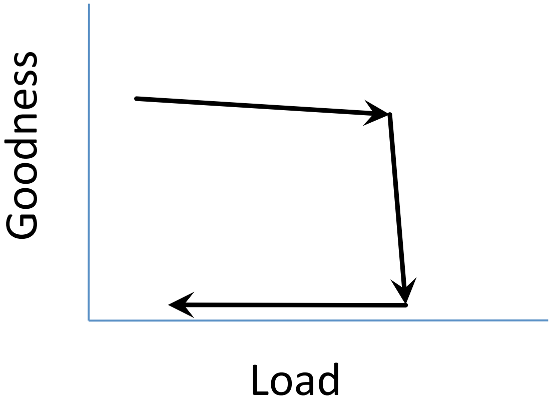
Crashing a cluster: Plan B

- Switch to secondary due to failure or maintenance.
  - Blacklist one machine, add 25% load to host in other half of cluster.
  - Now those get slow enough that one of them gets blacklisted.
  - Lather, rinse, repeat.
- “Double or nothing”

Capacity management

- Growth is in units of servers. When to buy more?
- Test team provides one opinion
- Ops team aims to find max resource and stay below limit.
  - Two kinds of limit: graceful and catastrophic.
- Challenge: balancing multiple resources on machines





Ops lessons

- Never do the same thing to all machines at once
  - Stats queries, re-indexing have all crashed clusters in the past
- Smaller DBs are better
  - Already coping with many DBs
  - Re-indexing, backups, upgrades, etc. all faster
- Read-only mode is powerful
  - Failure, maintenance, migration all use it
- Use the live site to try things out
  - New code (after test), new SQL settings, etc.
  - “Taste vs. Test”

CONCLUSIONS

Conclusions

- SQL can be tamed
  - Real issues, but mostly manageable with some infrastructure. Ops cost not out of line.
- Hard to do better
  - It keeps improving.
  - Each time we go to design something, we find that SQL already does it. Perhaps not in the form we want exactly, but…
- Not always the best solution
  - E.g. Distributed cache (memcached) lets you use cheaper memory and control what's in it.

SQL wish list

- Easier ones:
  - – Partitioned data support
    - Easy migration/placement control, reporting, jobs
  - – Supporting the aggregated data pattern (GMVs)
  - – Improved manageability
- Harder:
  - – DB schema evolution tamed
  - – Soft delete/versioning support of some kind
  - – A--D transactions